Single-machine scheduling of multi-operation jobs without missing operations to minimize the total completion time

T.C.E. CHENG[1], **C.T. NG**[1*] **and J.J. YUAN**[1,2]

[1]Department of Logistics, The Hong Kong Polytechnic University,
Hung Hom, Kowloon, Hong Kong, People's Republic of China
[2]Department of Mathematics, Zhengzhou University,
Zhengzhou, Henan 450052, People's Republic of China

Abstract

We consider the problem of scheduling multi-operation jobs on a singe machine to minimize the total completion time. Each job consists of several operations that belong to different families. In a schedule each family of job operations may be processed as batches with each batch incurring a set-up time. A job is completed when all of its operations have been processed. We first show that the problem is strongly NP-hard even when the set-up times are common and each operation is not missing. When the operations have identical processing times and either the maximum set-up time is sufficiently small or the minimum set-up time is sufficiently large, the problem can be solved in polynomial time. We then consider the problem under the job-batch restriction in which the operations of each batch is partitioned into operation batches according to a partition of the jobs. We show that this case of the problem can be solved in polynomial time under a certain condition.

Keywords: Scheduling, Single machine, Multi-operation jobs, Job-batch restriction, SPT-agreeability.

*Corresponding author

1 Introduction

As introduced in [4], the problem under consideration arises in a food manufacturing environment. The problem can be stated as follows: Let n multi-operation jobs $J_1, J_2, ..., J_n$ and a machine that can handle only one job at a time be given. Each job consists of several operations that belong to different families. There are F families $\mathcal{F}_1, \mathcal{F}_2, ..., \mathcal{F}_F$. We assume that each job has at most one operation in each family. The operation of job J_j ($j = 1, ..., n$) in family $\mathcal{F}_f$ ($f = 1, ..., F$) is denoted by (j, f), and its associated processing time is $p_{(j,f)} \geq 0$. Any operation with a zero processing time is called a trivial operation. Each family $\mathcal{F}_f$ has an associated set-up time s_f. The operations of each family are processed in batches, where a batch of a family is a subset of the operations of this family and the batches of a family form a partition of the operations belonging to this family. Each batch (of family $\mathcal{F}_f$) containing at least one nontrivial operation will incur a set-up time s_f. That is, a trivial operation does not share the set-up time in its family. Hence, a trivial operation is treated as a missing operation. Trivial operations arise when not every job J_j contains all the operations (j, f), $1 \leq f \leq F$. A job is completed when all of its operations have been processed. Hence, the completion time of job J_j is

$$C_j = \max\{C_{(j,f)} : (j, f) \text{ is a nontrival operation of } J_j\},$$

where $C_{(j,f)}$ is the completion time of the operation (j, f). The objective is to find a schedule that minimizes the sum of the job completion times $\sum_j C_j$. Following [4], we denote the problem by

$$1|s_f,\ \textit{assembly}|\sum C_j,$$

where the term "assembly" is used to describe the fact that a job is completed when it becomes available for assembly, i.e., when all of its operations have been processed. If we require that all the operations in any family are to be scheduled contiguously, i.e., each family acts as a single batch, we say that we study the problem under the group technology (GT) assumption. The corresponding problem is denoted by

$$1|s_f,\ \textit{assembly, GT}|\sum C_j.$$

If $p_{(j,f)} > 0$ for each family $\mathcal{F}_f$ and each job J_j, we say that the assembly problem is without missing operations, and the corresponding problem is denoted by

$$1|s_f,\ \textit{assembly},\ p_{(j,f)} > 0|\sum C_j.$$

Example 1: We consider an instance of the problem $1|s_f,\ \textit{assembly},\ p_{(j,f)} > 0|\sum C_j$. Suppose that we have four jobs J_1, J_2, J_3, J_4, and two families $\mathcal{F}_1, \mathcal{F}_2$ of operations with $s_1 = 1$ and $s_2 = 2$. The processing times of the operations are given in Table 1.

Table 1: Processing times.

Jobs	J_1	J_2	J_3	J_4
$p_{(j,1)}$	2	1	2	1
$p_{(j,2)}$	1	2	1	2

Let π be a schedule defined in the following way. The first family is partitioned into two batches $\{(1,1),(2,1)\}$ and $\{(3,1),(4,1)\}$, and the second family $\{(1,2),(2,2),(3,2),(4,2)\}$ acts as a single batch. Then we process the operations in the three batches in the following order:

$$\{(1,1,) \to (2,1)\} \to \{(1,2,) \to (2,2) \to (3,2) \to (4,2)\} \to \{(3,1) \to (4,1)\}.$$

Processing of the operations can be shown in a Gantt chart in Figure 1.

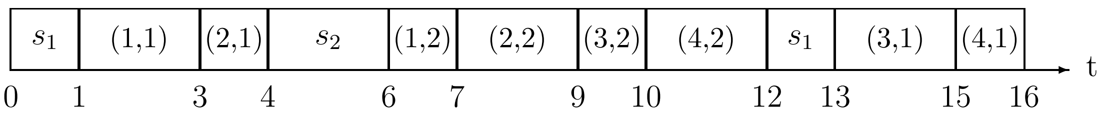

Figure 1: Processing of jobs in π.

The completion times of all the operations and jobs are given in Table 2. The total completion time of the jobs is 47. $\square$

Table 2: Completion times.

Jobs	J_1	J_2	J_3	J_4
$C_{(j,1)}$	3	4	15	16
$C_{(j,2)}$	7	9	10	12
C_j	7	9	15	16

The following complexity results show that the complexity of the assembly problem is different between the versions with and without missing operations. It seems that the problem is more tractable if it is without missing operations.

Gerodimos et al. [4] gave an $O(Fn\log n)$ algorithm for the scheduling problem

$$1|s_f,\ \textit{assembly},\ GT,\ p_{(j,f)} > 0|\sum C_j.$$

But Ng et al. [6] showed that the scheduling problems

$$1|s_f,\ \textit{assembly},\ p_{(j,f)} = 0 \text{ or } 1|\sum C_j$$

and

$$1|s_f,\ \textit{assembly},\ GT,\ p_{(j,f)} = 0 \text{ or } 1|\sum C_j$$

are strongly NP-hard. Cheng et al. [1] showed that the scheduling problem

$$1|s_f,\ \textit{assembly},\ d_j = d,\ p_{(j,f)} = 0 \textit{ or } 1|\sum U_j$$

is strongly NP-hard and that the scheduling problem

$$1|s_f,\ \textit{assembly},\ d_j = d,\ p_{(j,f)} > 0|\sum U_j$$

can be solved by the shortest processing time (SPT) rule in $O(n(\log n + F))$ time, where d_j is the due date of J_j, $d_j = d$ means that the jobs have a common due date d, and $U_j = 1$ if $C_j > d_j$ and 0 otherwise.

It should be noticed that in the NP-hardness proofs of the above three NP-hard problems in [6] and [1], the operations with a zero processing time were treated as missing operations and they did not share the set-up times. Since the two problems

$$1|s_f,\ \textit{assembly},\ GT,\ p_{(j,f)} > 0|\sum C_j.$$

and

$$1|s_f,\ \textit{assembly},\ d_j = d,\ p_{(j,f)} > 0|\sum U_j$$

can be solved in polynomial time, this means that the version without missing operations is distinct from the version with missing operations. Thus the computational complexity of the problem

$$1|s_f,\ \textit{assembly},\ p_{(j,f)} > 0|\sum C_j$$

is still open. Furthermore, to the best of our knowledge, the computational complexity of the same problem is still unaddressed even if the number of families F is 2 or any fixed number.

We show in Section 2 that the assembly problem $1|s_f,\ \textit{assembly},\ p_{(j,f)} > 0|\sum C_j$ remains strongly NP-hard even when the set-up times are common. We show in Section 3 that when the operations have identical processing times and either the maximum set-up time is sufficiently small or the minimum set-up time is sufficiently large, the problem can be solved in polynomial time.

We say that the jobs are of SPT-agreeability if the jobs can be re-indexed such that

$$p_{(1,f)} \leq p_{(2,f)} \leq \cdots \leq p_{(n,f)}$$

for $1 \leq f \leq F$. Gerodimos et al. [4] provided an $O(F^2 n^{F+1})$ algorithm for the assembly problem $1|s_f,\ \textit{assembly},\ \textit{SPT-agreeability}|\sum C_j$, which is a polynomial-time algorithm if the number of families F is fixed. When F is arbitrary, the complexity of the problem $1|s_f,\ \textit{assembly},\ \textit{SPT-agreeability}|\sum C_j$ is still open [4]. According to Gerodimos et al. [4], if the jobs are of SPT-agreeability, then there is an optimal schedule such that the operations of each batch are processed in the shortest processing time (SPT) order.

Hence, we consider a variation of the problem called job-batch, assembly scheduling, in which we have the following Job-batch Restriction: The batches of the families are determined by the jobs, i.e., the jobs are first partitioned into k $(1 \leq k \leq n)$ subsets $B_1, B_2, ..., B_k$, and then the batches of each family $\mathcal{F}_f$ $(\leq f \leq F)$ are formed by

$$\mathcal{F}_{(1,f)}, \mathcal{F}_{(2,f)}, ..., \mathcal{F}_{(k,f)},$$

where

$$\mathcal{F}_{(i,f)} = \{(j,f) : j \in B_i\}, \quad 1 \leq i \leq k.$$

Such a scheduling problem is denoted by

$$1|s_f,\ \textit{job-batch assembly}|\sum C_j,$$

which is clearly a generalization of the problem $1|s_f,\ \textit{assembly, GT}|\sum C_j$. When missing operations are allowed, this problem is still strongly NP-hard since the problem $1|s_f,\ \textit{assembly, GT},\ p_{(j,f)} = 0 \text{ or } 1|\sum C_j$ is strongly NP-hard [6]. Hence, we consider the problem under the restriction that there are no missing operations. We show in Section 4 that the problem

$$1|s_f,\ \textit{job-batch assembly},\ \textit{SPT-agreeability}|\sum C_j$$

can be solved in $O(Fn^3)$ time.

2 NP-hardness proofs

Our reduction uses the NP-complete linear arrangement problem of graphs. We first introduce some graph theory terminology.

The graphs considered here are finite and simple. For a graph G, let $V = V(G)$ and $E = E(G)$ denote its sets of vertices and edges, respectively. An edge e with end vertices u and v is denoted by $e = uv = vu$. For $e = uv \in E$, we say that e is incident to u and v. The set of edges incident to a vertex v is denoted by $E_v = E_v(G)$, i.e.,

$$E_v = \{e \in E : e \text{ is incident to } v \text{ in } G\}.$$

The degree of a vertex $v \in V$, denoted by $d(v)$, is defined by

$$d(v) = |E_v|.$$

It is well-known that

$$\sum_{v \in V} d(v) = 2|E|.$$

Given a graph G, a labelling σ of G is a permutation

$$\sigma : V \longrightarrow \{1, 2, 3, ..., |V|\}.$$

The linear sum of G under the labelling σ is defined by

$$S(G, \sigma) = \sum_{xy \in E} |\sigma(x) - \sigma(y)|.$$

The linear arrangement problem of graphs is defined as follows:

Linear arrangement problem: For a given graph G and a positive integer Y, is there a labelling σ of G such that $S(G, \sigma) \leq Y$?

By [2, 3], it is known that the linear arrangement problem is NP-complete in the strong sense. We will make use of this result for the reduction.

The following lemma is implied in [5]. We give a short proof of the result for the sake of completeness.

Lemma 2.1 For a labelling σ of a graph G,

$$\sum_{uv\in E} 2\max\{\sigma(u),\sigma(v)\} - \sum_{v\in V} d(v)\sigma(v) = \sum_{uv\in E} |\sigma(u)-\sigma(v)|.$$

Proof By noting the facts that

$$\sum_{v\in V} d(v)\sigma(v) = \sum_{uv\in E}(\sigma(u)+\sigma(v))$$

and

$$2\max\{\sigma(u),\sigma(v)\} - (\sigma(u)+\sigma(v)) = |\sigma(u)-\sigma(v)|,$$

we see that the result follows. □

Theorem 2.2 The scheduling problem

$$1|s_f = s,\ \textit{assembly},\ p_{(j,f)} > 0|\sum C_i$$

is strongly NP-hard.

Proof The decision version of our scheduling problem is clearly in NP. To prove the strong NP-completeness of the problem, we use the NP-complete linear arrangement problem of graphs for our reduction.

Suppose that we are given an instance of the linear arrangement problem of graphs, which inputs a graph G and a positive integer Y and asks whether there is a labelling σ of G such that $S(G,\sigma) \leq Y$. Without loss of generality, we suppose that $|V| \geq 5$. We construct an instance of the decision version of our scheduling problem as follows:

• There are $n = |V|^2 + |V|^8$ jobs, which are of three types: vertex-jobs, edge-jobs and small jobs.

• Each vertex $v \in V$ corresponds to $\alpha(v) = |V| - d(v)$ vertex-jobs $J_{v(1)}, J_{v(2)}, ..., J_{v(\alpha(v))}$.

• Each edge $e \in E$ corresponds to $\alpha(e) = 2$ edge-jobs $J_{e(1)}$ and $J_{e(2)}$. Note that

$$\sum_{v\in V}\alpha(v) + \sum_{e\in E}\alpha(e) = |V|^2 - \sum_{v\in V} d(v) + 2|E| = |V|^2.$$

Hence, the numbers of vertex-jobs and edges-jobs are $|V|^2$.

• There are additional $|V|^8$ small jobs $J_{s(1)}, J_{s(2)}, ..., J_{s(|V|^8)}$.

• There are $F = |V|$ families, with each vertex $v \in V$ corresponding to a family $\mathcal{F}_v$ with a set-up time $s = |V|^5 + 2|V|^{11} + |V|^{17}(|V| + 2) + 2|V|^{15}Y$.

• For $v \in V$, the family $\mathcal{F}_v$ contains $|V|^2+|V|^8$ operations, where we have $\alpha(v) = |V|-d(v)$ vertex-operations

$$(v(1), v), (v(2), v), ..., (v(\alpha(v)), v),$$

with each having a processing time $|V|^{14}$, and $2d(v)$ edge-operations

$$(e(1), v), (e(2), v), \quad e \in E_v,$$

with each also having a processing time $|V|^{14}$; each of the other operations (called small operations) not mentioned here has a processing time 1.

• Each operation (still called small operation) of a small job has processing time 1.

• The decision is whether there exists a schedule such that the total completion time $\sum C_j$ is at most

$$X = |V|^9(s+|V|^2+|V|^8)+|V|^2((|V|-1)s+|V|^3+|V|^9)+(s+2|V|^{15})(\frac{1}{2}|V|^2(|V|+1)+Y).$$

Summarizing the above construction, we have $n = |V|^2 + |V|^8$ jobs and $|V|$ families with each job having $|V|$ operations with a positive processing time belonging to distinct families; we have three types of jobs: vertex-jobs, edge-jobs and small jobs; we also have three types of operations: vertex-operations, edge-operations and small operations, where each of the vertex-operations and edge-operations has processing time $|V|^{14}$, and each small operations has processing time 1; furthermore, for each family $\mathcal{F}_v$, the vertex-operations in it are $(v(1), v), (v(2), v), ..., (v(\alpha(v)), v)$, and the edge-operations in it are $(e(1), v), (e(2), v)$ with $e \in E_v$.

For the sake of a better understanding of the above reduction, we consider an example as follows. Figure 2 is a graph G with vertex set $V(G) = \{x, y, u, v, w\}$ and edge set $E(G) = \{a, b, c, d, e\}$. Using G as an instance of the linear arrangement problem, the constructed instance of the scheduling problem is displayed in Table 3 and Table 4. Table 3 shows the jobs and their operations, and Table 4 shows the families and their operations.

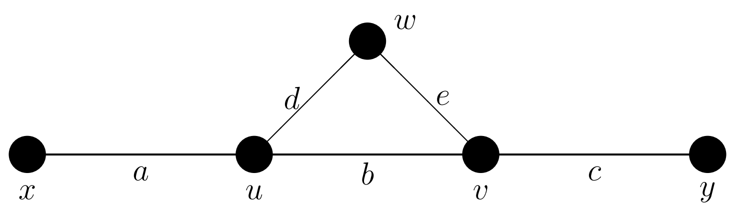

Figure 2: A graph G in the linear arrangement problem.

Types	Jobs	Large operations	Small operations
Vertices	$J_{x(i)}$, $1 \le i \le 4$	$(x(i), x)$	$(x(i), z)$, $z \in V(G) \setminus \{x\}$
	$J_{y(i)}$, $1 \le i \le 4$	$(y(i), y)$	$(y(i), z)$, $z \in V(G) \setminus \{y\}$
	$J_{u(i)}$, $1 \le i \le 2$	$(u(i), u)$	$(u(i), z)$, $z \in V(G) \setminus \{u\}$
	$J_{v(i)}$, $1 \le i \le 2$	$(v(i), v)$	$(v(i), z)$, $z \in V(G) \setminus \{v\}$
	$J_{w(i)}$, $1 \le i \le 3$	$(w(i), w)$	$(w(i), z)$, $z \in V(G) \setminus \{w\}$
Edges	$J_{a(i)}$, $1 \le i \le 2$	$(a(i), x)$, $(a(i), u)$	$(a(i), z)$, $z \in V(G) \setminus \{x, u\}$
	$J_{b(i)}$, $1 \le i \le 2$	$(b(i), u)$, $(b(i), v)$	$(b(i), z)$, $z \in V(G) \setminus \{u, v\}$
	$J_{c(i)}$, $1 \le i \le 2$	$(c(i), v)$, $(c(i), y)$	$(c(i), z)$, $z \in V(G) \setminus \{v, y\}$
	$J_{d(i)}$, $1 \le i \le 2$	$(d(i), u)$, $(d(i), w)$	$(d(i), z)$, $z \in V(G) \setminus \{u, w\}$
	$J_{e(i)}$, $1 \le i \le 2$	$(e(i), v)$, $(e(i), w)$	$(e(i), z)$, $z \in V(G) \setminus \{v, w\}$
Small	$J_{s(i)}$. $1 \le i \le 5^8$	None	$(s(i), z)$, $z \in V(G)$

Table 3: Jobs and their operations.

Families	Large operations	Small operations
$\mathcal{F}_x$	$(x(i), x)$, $1 \le i \le 4$; $(a(1), x)$, $(a(2), x)$	$(z(i), x)$, $1 \le i \le 5 - d(z)$, $z \in V(G) \setminus \{x\}$; $(h(i), x)$, $1 \le i \le 2$, $h \in E(G) \setminus \{a\}$; $(s(i), x)$, $1 \le i \le 5^8$
$\mathcal{F}_y$	$(y(i), y)$, $1 \le i \le 4$; $(c(1), y)$, $(c(2), y)$	$(z(i), y)$, $1 \le i \le 5 - d(z)$, $z \in V(G) \setminus \{y\}$; $(h(i), y)$, $1 \le i \le 2$, $h \in E(G) \setminus \{c\}$; $(s(i), y)$, $1 \le i \le 5^8$
$\mathcal{F}_u$	$(u(i), u)$, $1 \le i \le 2$; $(a(1), u)$, $(a(2), u)$; $(b(1), u)$, $(b(2), u)$; $(d(1), u)$, $(d(2), u)$	$(z(i), u)$, $1 \le i \le 5 - d(z)$, $z \in V(G) \setminus \{u\}$; $(h(i), u)$, $1 \le i \le 2$, $h \in E(G) \setminus \{a, b, d\}$; $(s(i), u)$, $1 \le i \le 5^8$
$\mathcal{F}_v$	$(v(i), v)$, $1 \le i \le 2$; $(b(1), v)$, $(b(2), v)$; $(c(1), v)$, $(c(2), v)$; $(e(1), v)$, $(e(2), v)$	$(z(i), v)$, $1 \le i \le 5 - d(z)$, $z \in V(G) \setminus \{v\}$; $(h(i), v)$, $1 \le i \le 2$, $h \in E(G) \setminus \{b, c, e\}$; $(s(i), v)$, $1 \le i \le 5^8$
$\mathcal{F}_w$	$(w(i), w)$, $1 \le i \le 3$; $(d(1), w)$, $(d(2), w)$; $(e(1), w)$, $(e(2), w)$	$(z(i), w)$, $1 \le i \le 5 - d(z)$, $z \in V(G) \setminus \{w\}$; $(h(i), w)$, $1 \le i \le 2$, $h \in E(G) \setminus \{d, e\}$; $(s(i), w)$, $1 \le i \le 5^8$

Table 4: Families and their operations.

Clearly, the construction can be done in polynomial time. We show in the sequel that the instance of the linear arrangement problem has a labelling σ such that $S(G, \sigma) \le Y$ if and only if the instance of the decision version of our scheduling problem has a schedule such that $\sum C_j \le X$.

If the linear arrangement problem has a labelling σ of G such that $S(G,\sigma) \leq Y$, we construct a schedule π as follows. The family $\mathcal{F}_v$ with $\sigma(v)=1$ is processed in a single batch $\mathcal{A}_v$. Each of the other families $\mathcal{F}_v$ with $\sigma(v)>1$ is processed in two batches $\mathcal{B}_v$ and $\mathcal{A}_v$; the batch $\mathcal{B}_v$ consists of all the operations in $\mathcal{F}_v$ with processing time 1, and the batch $\mathcal{A}_v$ consists of all the operations in $\mathcal{F}_v$ with processing time $|V|^{14}$. The jobs are scheduled in the following way.

The batches $\mathcal{B}_v$ with $\sigma(v)>1$ are scheduled first in any order with the operations in each batch being scheduled in any order; then the family $\mathcal{F}_v=\mathcal{A}_v$ with $\sigma(v)=1$ is scheduled such that the operations of the small jobs are scheduled first and then the other operations are scheduled in any order; and then the batches $\mathcal{A}_v$ with $\sigma(v)>1$ are scheduled such that $\mathcal{A}_v$ is scheduled before $\mathcal{A}_u$ if and only if $\sigma(v)<\sigma(u)$ and such that the operations in each batch are scheduled in any order.

In the schedule π the first $|V|$ batches include all the operations with processing time 1, each of the first $|V|$ batches in π contains at most $|V|^2+|V|^8$ operations with processing time 1, the first $|V|-1$ batches consist of operations with processing time 1, and the $|V|$-th batch leads the operations with processing time 1. Hence, the completion time of the last small job under π is less than

$$|V|(s+|V|^2+|V|^8).$$

Furthermore, each batch $\mathcal{A}_v$ $(v \in V)$ consists of $|V|+d(v)<2|V|$ operations with processing time $|V|^{14}$. Hence, the completion time of every operation in batch $\mathcal{A}_v$ $(v \in V)$ is less than

$$(|V|-1)s+|V|^3+|V|^9+(s+2|V|^{15})\sigma(v).$$

It follows that for each edge-job $J_{uv(i)}$ with $uv \in E$ and $i=1,2$, the completion time $C_{uv(i)}=\max\{C_{(uv(i),u)},C_{(uv(i),v)}\}$ under π is less than

$$(|V|-1)s+|V|^3+|V|^9+(s+2|V|^{15})\ \max\{\sigma(u),\sigma(v)\}.$$

Now the total completion time of the small jobs is given by

$$\sum_{1\leq i\leq |V|^8} C_{s(i)} < |V|^9(s+|V|^2+|V|^8),$$

the total completion time of the vertex-jobs is given by

$$\begin{aligned}
& \textstyle\sum_{v\in V}\sum_{1\leq i\leq \alpha(v)} C_{v(i)} \\
< \ & \textstyle\sum_{v\in V}\alpha(v)((|V|-1)s+|V|^3+|V|^9+(s+2|V|^{15})\sigma(v)) \\
= \ & \textstyle\sum_{v\in V}(|V|-d(v))((|V|-1)s+|V|^3+|V|^9+(s+2|V|^{15})\sigma(v)) \\
= \ & (|V|^2-2|E|)((|V|-1)s+|V|^3+|V|^9) \\
& \textstyle +(s+2|V|^{15})|V|\sum_{v\in V}\sigma(v)-(s+2|V|^{15})\sum_{v\in V}d(v)\sigma(v) \\
= \ & (|V|^2-2|E|)((|V|-1)s+|V|^3+|V|^9) \\
& \textstyle +(s+2|V|^{15})(\frac{1}{2}|V|^2(|V|+1))-(s+2|V|^{15})\sum_{v\in V}d(v)\sigma(v),
\end{aligned}$$

and the total completion time of the edge-jobs is given by

$$\begin{array}{rl} & \sum_{uv\in E}\sum_{1\le i\le 2} C_{uv(i)} \\ < & \sum_{uv\in E} 2((|V|-1)s+|V|^3+|V|^9+(s+2|V|^{15})\ \max\{\sigma(u),\sigma(v)\}) \\ = & 2|E|((|V|-1)s+|V|^3+|V|^9)+(s+2|V|^{15})\sum_{uv\in E} 2\max\{\sigma(u),\sigma(v)\}. \end{array}$$

Hence, by Lemma 2.1, the total completion time of all the jobs is given by

$$\begin{array}{rl} & \sum_{1\le i\le |V|^8} C_{s(i)} + \sum_{v\in V}\sum_{1\le i\le \alpha(v)} C_{v(i)} + \sum_{uv\in E}\sum_{1\le i\le 2} C_{uv(i)} \\ < & |V|^9(s+|V|^2+|V|^8)+|V|^2((|V|-1)s+|V|^3+|V|^9) \\ & +(s+2|V|^{15})(\frac{1}{2}|V|^2(|V|+1))+(s+2|V|^{15})S(G,\sigma) \\ \le & |V|^9(s+|V|^2+|V|^8)+|V|^2((|V|-1)s+|V|^3+|V|^9) \\ & +(s+2|V|^{15})(\frac{1}{2}|V|^2(|V|+1))+(s+2|V|^{15})Y \\ = & X. \end{array}$$

So the scheduling problem has the required schedule.

Conversely, assume that the scheduling problem has a schedule π such that the total completion time of all the jobs is at most X. Define a labelling σ of the graph G in the following way. For every two vertices $u, v \in V$, $\sigma(u) < \sigma(v)$ if and only if a certain operation (u, x) in family $\mathcal{F}_u$ with processing time $|V|^{14}$ is processed before every operation in family $\mathcal{F}_v$ with processing time $|V|^{14}$ under the schedule π.

If there are a family $\mathcal{F}_u$ and an operation (x, y) with processing time $|V|^{14}$ such that every operation of the small jobs in family $\mathcal{F}_u$ is scheduled after (x, y), then the processing of the operation (x, y) and at least $|V|$ set-ups must be scheduled before any small job is completed. This means that the completion time of every small job is at least $|V|s+|V|^{14}$. Then the total completion time of the $|V|^8$ small jobs is at least $|V|^9 s+|V|^{22}$. By noting the facts that $Y < \frac{1}{2}|V|^3$ and $|V| \ge 5$, we can easily check that

$$|V|^9 s + |V|^{22} > X.$$

This contradicts the fact that the total completion time of all the jobs is at most X. This leads to the following claim.

Claim For every family $\mathcal{F}_u$, there is at least one operation O of a certain small job in family $\mathcal{F}_u$ such that O is scheduled before every operation (x, y) with processing time $|V|^{14}$.

By the above claim, there are at least $|V| - 1$ batches, each of which consisting of the small jobs, such that these batches are scheduled before any operation with processing time $|V|^{14}$. As a rough estimate, it is easy to see that the completion time of the first small job under π is greater than $|V|s$, and the completion time of every operation in family $\mathcal{F}_v$ $(v \in V)$ with processing time $|V|^{14}$ is greater than

$$(|V|-1)s + s\sigma(v).$$

It follows that for each edge-job $J_{uv(i)}$ with $uv \in E$ and $i = 1, 2$, the completion time $C_{uv(i)} = \max\{C_{(uv(i),u)}, C_{(uv(i),v)}\}$ under π is greater than

$$(|V| - 1)s + s\ \max\{\sigma(u), \sigma(v)\}.$$

Now the total completion time of the small jobs is given by

$$\sum_{1 \le i \le |V|^8} C_{s(i)} > |V|^9 s,$$

the total completion time of the vertex-jobs is given by

$$\begin{aligned}
& \textstyle\sum_{v\in V} \sum_{1\le i\le \alpha(v)} C_{v(i)} \\
> & \textstyle\sum_{v\in V} \alpha(v)((|V| - 1)s + s\sigma(v)) \\
= & \textstyle\sum_{v\in V} (|V| - d(v))((|V| - 1)s + s\sigma(v)) \\
= & \textstyle(|V|^2 - 2|E|)(|V| - 1)s + s|V| \sum_{v\in V} \sigma(v) - s \sum_{v\in V} d(v)\sigma(v) \\
= & \textstyle(|V|^2 - 2|E|)(|V| - 1)s + s(\frac{1}{2}|V|^2(|V| + 1)) - s \sum_{v\in V} d(v)\sigma(v),
\end{aligned}$$

and the total completion time of the edge-jobs is given by

$$\begin{aligned}
& \textstyle\sum_{uv\in E} \sum_{1\le i\le 2} C_{uv(i)} \\
> & \textstyle\sum_{uv\in E} 2((|V| - 1)s + s\ \max\{\sigma(u), \sigma(v)\}) \\
= & \textstyle 2|E|(|V| - 1)s + s \sum_{uv\in E} 2 \max\{\sigma(u), \sigma(v)\}.
\end{aligned}$$

Hence, by Lemma 2.1, the total completion time of all the jobs is given by

$$\begin{aligned}
& \textstyle\sum_{1\le i\le |V|^8} C_{s(i)} + \sum_{v\in V} \sum_{1\le i\le n(v)} C_{v(i)} + \sum_{uv\in E} \sum_{1\le i\le 2} C_{uv(i)} \\
> & \textstyle |V|^9 s + |V|^2(|V| - 1)s + s(\frac{1}{2}|V|^2(|V| + 1)) + sS(G, \sigma).
\end{aligned}$$

By the fact that the total completion time under π is at most

$$X = |V|^9(s+|V|^2+|V|^8)+|V|^2((|V|-1)s+|V|^3+|V|^9)+(s+2|V|^{15})(\frac{1}{2}|V|^2(|V|+1)+Y),$$

we have

$$sS(G, \sigma) < sY + |V|^5 + 2|V|^{11} + |V|^{17}(|V| + 2) + 2|V|^{15}Y.$$

Because

$$s = |V|^5 + 2|V|^{11} + |V|^{17}(|V| + 2) + 2|V|^{15}Y,$$

we deduce that $sS(G, \sigma) < sY + s$, and so

$$S(G, \sigma) < Y + 1.$$

The result follows by noting that both $S(G, \sigma)$ and Y are integers.

3 Scheduling with identical processing times and restricted set-up times

Consider the scheduling problem $1|s_f,\ \textit{assembly},\ p_{(j,f)} = p|\sum C_i$. Write $s_{\max} = \max\{s_f : 1 \le f \le F\}$ and $s_{\min} = \min\{s_f : 1 \le f \le F\}$. We show that the scheduling problem can be solved in polynomial time if either $s_{\max}$ is sufficiently small or $s_{\min}$ is sufficiently large. Assume that the families of operations have been re-indexed so that $s_1 \le s_2 \le \cdots \le s_F$. Then $s_1 = s_{\min}$ and $s_F = s_{\max}$.

As in [4], in a schedule π, an operation (j, f) is called final if $C_j = C_{(j,f)}$, and non-final if $C_j > C_{(j,f)}$. Furthermore, a batch is called final if it contains at least one final operation, and a batch is called full if it is a family of operations. It can be observed that in a given final batch of an optimal schedule, the final operations are processed before the non-final operations (if any).

Theorem 3.1 If $s_{\min} \ge np$, then each family of operations acts as a full batch in any optimal schedule.

Proof Let π be an optimal schedule. Let B be the first final batch in π. Let t be the completion time of B. We only need to show that B is also the last final batch in π.

Suppose that B is not the last final batch in π. Let B' be the second final batch in π. Let (x, f) be the first operation in B'. Then (x, f) is a final operation. Let B^* be the first batch in π such that $B^* \subseteq \mathcal{F}_f$. Then B^* is not full. Let π^* be a new schedule obtained from π by shifting (x, f) from B' to B^*. If B^* is completed after time t, then π^* is better than π since $C_j(\pi^*) < C_j(\pi)$ and $C_j(\pi^*) \le C_j(\pi)$ for $j \ne x$. Hence, we suppose that B^* is completed by time t. Write $\mathcal{J}^{(t)} = \{J_j : C_j(\pi) \le t\}$. Then, for each $J_j \in \mathcal{J}^{(t)}$, $C_j(\pi^*) \le C_j(\pi) + p$, for each $J_i \notin \mathcal{J}^{(t)} \cup \{J_x\}$, $C_i(\pi^*) \le C_i(\pi)$. Furthermore, $C_x(\pi^*) \le C_x(\pi) - s_f$. Hence, we have

$$\sum_{1 \le j \le n} C_j(\pi^*) \le \sum_{1 \le j \le n} C_j(\pi) + |\mathcal{J}^{(t)}|p - s_f.$$

Since $s_f \ge s_{\min} \ge np > |\mathcal{J}^{(t)}|p$, we conclude that $\sum_{1 \le j \le n} C_j(\pi^*) < \sum_{1 \le j \le n} C_j(\pi)$. This contradicts the assumption that π is optimal. The result follows. □

As a consequence of Theorem 3.1, for the case $s_{\min} \ge np$, the scheduling problem $1|s_f,\ \textit{assembly},\ p_{(j,f)} = p|\sum C_i$ can be solved in $O(Fn)$ time.

When $s_{\max} \le p/n$, we define a schedule σ^* by the following batch partition and processing order:

$$\begin{aligned}&\{(1,1)\},\{(1,2)\},...,\{(1,F-2)\},\{(1,F-1)\},\{(1,F),(2,F)\},\\&\{(2,1)\},\{(2,2)\},...,\{(2,F-2)\},\{(2,F-1),(3,F-1)\},\\&\{(3,1)\},\{(3,2)\},...,\{(3,F-2)\},\{(3,F),(4,F)\},\\&\{(4,1)\},\{(4,2)\},...,\{(4,F-2)\},\{(4,F-1),(5,F-1)\},...\end{aligned}$$

That is, we first obtain a schedule σ by setting, for each operation (j,f), a single batch $B_{(j,f)} = \{(j,f)\}$, and sequencing the batches in the order

$$B_{(1,1)}, B_{(1,2)}, ..., B_{(1,F)}, B_{(2,1)}, B_{(2,2)}, ..., B_{(2,F)}, ..., B_{(n,1)}, B_{(n,2)}, ..., B_{(n,F)}.$$

Then the schedule σ^* is obtained from σ by deleting the batches

$$B_{(2,F)}, B_{(3,F-1)}, B_{(4,F)}, B_{(5,F-1)}, ...$$

and replacing the batches

$$B_{(1,F)}, B_{(2,F-1)}, B_{(3,F)}, B_{(4,F-1)}, ...$$

by

$$B_{(1,F)} \cup B_{(2,F)}, B_{(2,F-1)} \cup B_{(3,F-1)}, B_{(3,F)} \cup B_{(4,F)}, B_{(4,F-1)} \cup B_{(5,F-1)}, ...,$$

respectively.

Theorem 3.2 If $s_{\max} \leq p/n$, then the schedule σ^* is optimal.

Sketch of the proof The details of the proof of this theorem are easy but long. We only give a sketch of the proof. Suppose $n \geq 2$.

Let π be an optimal schedule. Suppose that there are k final batches in π, and suppose that B_i is the i-th final batch in π. Let t_i be the completion time of batch B_i in π. Write $\mathcal{J}^{(i)} = \{J_j : t_{i-1} < C_j(\pi) \leq t_i\}$, where $t_0 = 0$.

(a) By contradiction and shifting arguments, we can show that any non-final batch processed between t_{i-1} and t_i contains only operations of the jobs in $\mathcal{J}^{(i)}$, and such a non-final batch is of size $|\mathcal{J}^{(i)}|$.

(b) By contradiction and shifting arguments, we can show that any final batch B_i contains only operations of the jobs in $\mathcal{J}^{(i)} \cup \mathcal{J}^{(i+1)}$, where $\mathcal{J}^{(k+1)} = \emptyset$. That is $|B_i| = |\mathcal{J}^{(i)}| + |\mathcal{J}^{(i+1)}|$.

(c) Since $p_{(j,f)} = p$ for every operation (j,f), we can assume that $C_1(\pi) < C_2(\pi) < \cdots < C_n(\pi)$.

(d) Based on (a), (b) and (c), we show in the following that $k = n$ and $\mathcal{J}^{(i)} = \{J_i\}$, $1 \leq i \leq n$. Here we assume that $F \geq 3$. The case $F = 2$ can also be proved, but we omit it. If possible, let $x \in \{1, ..., k-1\}$ be the maximum value such that $|\mathcal{J}^{(x)}| \geq 2$. Let y be the maximum value such that $J_y \in \mathcal{J}^{(x)}$. Then $y \geq 2$. There are two possibilities: either $y = n$ or $y \leq n-1$.

If $y = n$, then let (y, f) be any operation such that $(y, f) \notin B_x$. The batch that contains (y, f) is denoted by B'. Then $B' \geq 2$. Write $B^* = B' \setminus \{(y, f)\}$. We obtain a new schedule π^* from π by replacing B' with the new batch B^* and adding a new batch $\{(y, f)\}$ just after B_x. For $J_j \in \mathcal{J}^{(i)}$ with $i \leq i \leq x-1$, we have $C_j(\pi^*) = C_j(\pi)$; for $J_j \in \mathcal{J}^{(x)} \setminus \{J_y\}$, we have $C_j(\pi^*) = C_j(\pi) - p$; for $j = y$, we have $C_j(\pi^*) = C_j(\pi) + s$. Hence, we have

$$\sum_{1 \leq j \leq n} C_j(\pi^*) \leq \sum_{1 \leq j \leq n} C_j(\pi) - p + s_f < \sum_{1 \leq j \leq n} C_j(\pi).$$

This contradicts the assumption that π is optimal.

If $y \leq n-1$, then $x \leq k-1$. Suppose $B_x \subseteq \mathcal{F}_a$. Then, from (b) and (c), we have $(y+1, a) \in B_x$. Let $\bar{B}$ be the the batch just after B_x and assume that $\bar{B} \subseteq \mathcal{F}_b$. Then $\bar{B}$ is not a final batch, since $F \geq 3$. The batch that contains (y, b) is still denoted by B'. Write $B^* = B' \setminus \{(y, b)\}$ and $B^*_x = B_x \setminus \{(y+1, a)\}$. We obtain a new schedule π' from π by replacing B' with B^*, replacing B_x with B^*_x, replacing $\bar{B}$ with $\hat{B} = \bar{B} \cup \{(y, b)\}$ with (y, b) being the first operation in $\hat{B}$, and then inserting a batch $\{(y+1, a)\}$ just after $\hat{B}$. Then we have

$$\sum_{1 \leq j \leq n} C_j(\pi') \leq \sum_{1 \leq j \leq n} C_j(\pi) - p + (n - y + 1)s_f < \sum_{1 \leq j \leq n} C_j(\pi).$$

Again, this contradicts the assumption that π is optimal.

(e) Now the structure of π is clear. Each final batch B_i, $1 \leq i \leq n-1$, contains exactly two operations of J_i and J_{i+1}, respectively. Each of the other batches contains just one operation. The processing time p contributes a fixed amount $Q(p) = Fpn(n+1)/2$ to the objective function. We only need to consider the contribution $R(s_1, s_2, ..., s_F)$ of the set-up times to the objective function. Suppose that $B_i \subseteq \mathcal{F}_{\delta(i)}$, $1 \leq i \leq n$. Then $\delta(i) \neq \delta(i+1)$ for $1 \leq i \leq n-1$. The total set-up time before the completion of each job J_i is calculated by

$$\begin{aligned} & (s_1 + \cdots + s_F) + (s_1 + \cdots + s_F - s_{\delta(1)}) + \cdots + (s_1 + \cdots + s_F - s_{\delta(i-1)}) \\ = \; & i(s_1 + \cdots + s_F) - (s_{\delta(1)} + \cdots + s_{\delta(i-1)}). \end{aligned}$$

It follows that

$$R(s_1, ..., s_F) = \frac{1}{2}n(n+1)(s_1 + \cdots + s_F) - \Big((n-1)s_{\delta(1)} + (n-2)s_{\delta(2)} + \cdots + s_{\delta(n-1)}\Big).$$

To minimize $R(s_1, ..., s_F)$ and guarantee the condition $\delta(i) \neq \delta(i+1)$ for $1 \leq i \leq n-1$, we must have

$$s_F = s_{\delta(1)} = s_{\delta(3)} = s_{\delta(5)} = \cdots$$

and

$$s_{F-1} = s_{\delta(2)} = s_{\delta(4)} = s_{\delta(6)} = \cdots.$$

Denote by $R^*(s_1, ..., s_F)$ the minimum value of $R(s_1, ..., s_F)$, subject to the condition $\delta(i) \neq \delta(i+1)$ for $1 \leq i \leq n-1$. It can be checked that σ^* is a schedule with objective value $R^*(s_1, ..., s_F) + Q(p) \leq R(s_1, ..., s_F) + Q(p) = \sum_j C_j(\pi)$. Hence, we conclude that σ^* is optimal. □

As a consequence of Theorem 3.2, for the case $s_{\max} \leq p/n$, the scheduling problem $1|s_f,\ \mathit{assembly},\ p_{(j,f)} = p|\sum C_i$ can be solved in $O(F \log F + Fn)$ time, where $O(F \log F)$ time is used to sort the set-up times.

4 Job-batch scheduling with SPT-agreeability

Consider the scheduling problem $1|s_f,\ \textit{job-batch assembly},\ \textit{SPT-agreeability}|\sum C_j$. Recall that if the jobs are partitioned into batches $B_1, B_2, ..., B_k$, then the batches of each family $\mathcal{F}_f$ $(1 \leq f \leq F)$ are formed by

$$\mathcal{F}_{(1,f)}, \mathcal{F}_{(2,f)}, ..., \mathcal{F}_{(k,f)},$$

where

$$\mathcal{F}_{(i,f)} = \{(j, f) : j \in B_i\}, \quad 1 \leq i \leq k.$$

For each job J_j, define $P_j = \sum_{1 \leq f \leq F} p_{(j,f)}$. Re-index the jobs such that $P_1 \leq P_2 \leq \cdots \leq P_n$. Since the jobs are of SPT-agreeability, we can see that, for each family $\mathcal{F}_f$, we have $p_{(1,f)} \leq p_{(2,f)} \leq \cdots \leq p_{(n,f)}$.

By the pairwise job exchange argument, we can show the following lemma.

Lemma 4.1 There is an optimal schedule π for the considered problem such that

(1) The jobs are partitioned into job batches $B_1, B_2, ..., B_k$ for some k with $1 \leq k \leq n$ such that, if J_i and J_j are two jobs such that $i < j$, and $J_i \in B_x$ and $J_j \in B_y$ for some x and y with $1 \leq x, y \leq k$, then $x \leq y$. Consequently, each batch B_x consists of jobs with consecutive indices.

(2) For each job batch $B_x = \{J_i, J_{i+1}, ..., J_j\}$ and for each family $\mathcal{F}_f$, the jobs in the operation batch $\mathcal{F}_{(x,f)} = \{(i, f), (i+1, f), ..., (j, f)\}$ are processed in the order $(i, f), (i+1, f), ..., (j, f)$ according to increasing order of the indices of their jobs.

(3) If x and y are two job batch indices with $x < y$, then each operation of the jobs in B_x are processed before all the operations of the jobs in B_y.

A schedule for the considered problem that satisfies the three properties in Lemma 4.1 is called a regular schedule.

Let π be an optimal regular schedule for the considered problem for the partial job set $\{J_i, J_{i+1}, ..., J_n\}$. Suppose the job batches in π are $B_1, B_2, ..., B_k$. Then, according to Lemma 4.1, the batches of the operations are processed in the following order:

$$\{\mathcal{F}_{(1,f)} : 1 \le f \le F\}, \{\mathcal{F}_{(2,f)} : 1 \le f \le F\}, ..., \{\mathcal{F}_{(k,f)} : 1 \le f \le F\}.$$

In order to give a backward dynamic programming recursion, we first consider the processing order of the batches in $\{\mathcal{F}_{(1,f)} : 1 \le f \le F\}$. This is equivalent to solving the problem $1|s_f$, *assembly*, *GT*, $p_{(j,f)} > 0|\sum C_j$ for the jobs in B_1, which, by Gerodimos et al. [4], can be solved in $O(F|B_1|\log|B_1|)$ time. But since the jobs are of SPT-agreeability and have been sorted in the SPT order, the solving of the present problem needs only $O(F|B_1|)$ time. In fact, the total completion time of the jobs in B_1 is determined by the processing of the operations in the last batch in $\{\mathcal{F}_{(1,f)} : 1 \le f \le F\}$, and we can enumerate the F possibilities to choose the best one. Suppose that the last job in the job batch B_1 is J_j. We denote the total completion time of the jobs in B_1 in π (which is also optimal when restricted in B_1 under the GT assumption) by $C_{GT}(i, j)$.

Now, let $G(i)$ be the total completion time of the jobs in $\{J_i, J_{i+1}, ..., J_n\}$ under an optimal regular schedule. If the first job batch consists of the jobs $J_i, J_{i+1}, ..., J_j$, then the sum of the set-up times and processing times of all operations of the jobs in the first job batch is calculated by $P(i, j) = \sum_{1 \le f \le F} s_f + \sum_{i \le l \le j} P_l$. Clearly, $P(i, j)$ contributes to the completion time of every jobs in $\{J_{j+1}, ..., J_n\}$. Hence, we have

$$G(i) = C_{GT}(i, j) + (n - j)P(i, j) + G(j + 1).$$

Based on the above discussion, the backward dynamic programming recursion for solving the problem $1|s_f$, *job-batch assembly*, *SPT-agreeability*$|\sum C_j$ is given by

$$G(i) = \min_{i \le j \le n} \left(C_{GT}(i, j) + (n - j)P(i, j) + G(j + 1)\right), \quad 1 \le i \le n.$$

The initial condition is given by $G(n + 1) = 0$. The optimal objective value is given by $G(1)$.

Note that we can calculate all the values $C_{GT}(i, j)$ for $1 \le i \le j \le n$ before invoking the backward dynamic programming recursion, which can be calculated in $O(Fn^3)$ time. All the values $P(i, j)$ can be calculated in $O(F + n^2)$ time in advance.

Each iteration of the above recursion can be calculated in $O(n)$ time. The dynamic recursion function has n states. Hence, the total complexity of the dynamic programming recursion is $O(n^2)$ by using the previously given values $C_{GT}(i, j)$ and $P(i, j)$. Consequently, we have

Theorem 4.2 The problem $1|s_f$, *job-batch assembly*, *SPT-agreeability*$|\sum C_j$ can be solved in $O(Fn^3)$ time.

5 Conclusions

We showed in this paper that the scheduling problem $1|s_f,\ \mathit{assembly},\ p_{(j,f)} > 0|\sum C_j$ is strongly NP-hard even when the set-up times are common. When the operations have identical processing times, the problem can be solved in polynomial time when either $s_{\max}$ is sufficiently small or $s_{\min}$ is sufficiently large. We also discussed the problem $1|s_f,\ \mathit{job\text{-}batch\ assembly},\ \mathit{SPT\text{-}agreeability}|\sum C_j$ and showed that it can be solved in $O(Fn^3)$ time by backward dynamic programming. For future research, the complexities of the problems $1|s_f,\ \mathit{job\text{-}batch\ assembly},\ p_{(j,f)} > 0|\sum C_j$ and $1|s_f,\ \mathit{assembly},$ $p_{(j,f)} > 0|\sum C_j$ with F being fixed are still open. It is also worth devising effective approximation algorithms for the NP-hard problem $1|s_f,\ \mathit{assembly},|\sum C_j$ with or without missing operations.

Acknowledgements

We are grateful for two anonymous referees for their constructive comments on an earlier version of this paper. This research was supported in part by The Hong Kong Polytechnic University under grant number G-YW43. Yuan was also supported in part by NSFC(10671183).

References

[1] T.C.E. Cheng, C.T. Ng and J.J. Yuan, A stronger complexity result for the single machine multi-operation jobs scheduling problem to minimize the number of tardy jobs, *Journal of Scheduling*, **6**(2003), 551-555.

[2] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, CA, 1979.

[3] M.R. Garey, D.S. Johnson and L. Stockmeyer, Some simplified NP-complete graph problem, *Theoretical Computer Science*, **1**(1976), 237-267.

[4] A.E. Gerodimos, C.A. Glass, C.N. Potts and T. Tautenhahn, Scheduling multi-operation jobs on a single machine, *Annals of Operations Research*, **92**(1999), 87-105.

[5] J.K. Lenstra and A.H.G. Rinnooy Kan, Complexity of scheduling under precedence constraints, *Operations Research,* **26**(1978), 22-35.

[6] C.T. Ng, T.C.E. Cheng and J.J. Yuan, Strong NP-hardness of the single machine multi-operation jobs total completion time scheduling problem, *Information Processing Letters*, **82**(2002), 187-191.