Unpublished Manuscript

Intractability of Operations Scheduling in the Presence of Multitasking

John Sum
Institute of Technology Management, National Chung Hsing University, Taichung 40227, Taiwan ROC, pfsum@nchu.edu.tw

Chi-Sing Leung
Department of Electronic Engineering, City University of Hong Kong, Kowloon Tong, Hong Kong, eeleungc@cityu.edu.hk

Kevin Ho
Department of Computer Science and Communication Engineering, Providence University, Shalu, Taichung, Taiwan ROC, ho@pu.edu.tw

Hall, Leung & Li have recently proposed a new model for human multitasking behavior. In their model, a primary job gets interruptions from all waiting jobs whenever it is being processed. Every time an interruption has been received, the worker switches from the primary job to the interrupting job. After part of the interrupting job has complete, the worker switches back to the primary job. Here, we extend from this model by adding that switching costs (SC) are job-dependent and late jobs are not allowed to interrupt. The scheduling problems of six different criteria are then investigated, including makespan, total weighted completion time (TWCT), maximum weighted tardiness (MWT), maximum weighted lateness (MWL), number of late jobs (NLJ) and weighted number of late jobs (WNLJ). For symmetric SC, we show that the makespan, TWCT, MWT and MWL problems are polynomial-time solvable. The NLJ problem is NP-hard. The WNLJ is strongly NP-hard. For asymmetric SC, we show that the makespan problem is NP-hard. The TCT and NLJ problems are strongly NP-hard. Accordingly, the complexity hierarchies of the scheduling problems in the presence of multitasking are sketched. The problems, in which the effect of multitasking on operations scheduling could be analyzed mathematically, can then be solicited.

Key words: complexity; human multitasking, single machine scheduling, switching costs

1. Introduction

In the presence of multitasking, human worker concurrently processes multiple tasks. On the road, some people drive a car and at the same time talk over the cell phone. At the school, some students open multiple browsers at the same time and search information concurrently for different assignments (Spink, Ozmutlu & Ozmutlu 2002, Ozmutlu, Ozmutlu & Spink 2003, Spink 2004, Spink, Park, Jansen & Pedersen 2006). In a recruiting firm, recruiters need to handle candidate search for one project and at the same time answer enquiries (in the form of emails or phone calls) from their clients for another project (Aral, Brynjolfsson & Van Alstyne 2007). In an Italian court, judges are assigned new cases in every morning, and at the same time they need to process cases which are already in the pipe (Coviello, Ichino & Persico 2010). Pilots have to read multiple sources of data over the panel and at the same time listen over his headset the enquiries and instructions from the air traffic control tower to ensure that the plane can safely take off and land (Loukopoulos, Dismukes & Barshi 2009). Air traffic controllers have to gather multiple sources of information, including radar signals, weather reports and the information from the pilots, to co-ordinate the taking off and landing of the planes (Loukopoulos, Dismukes & Barshi 2009).

1.1. Lab/Field studies and findings regarding multitasking

It is clear that there are many reasons why people multitask. The very first reason is about the job nature. Like plane control and air traffic control, the jobs are very complex and multiple tasks have to be handle. Missing any one task could cause catastrophic effect. The second reason is due to interruptions, from clients, colleagues, friends and even family members. These interruptions are usually not predictable. Some of them can be ignored. But some of them, especially from clients and colleagues, are unavoidable and always need immediate attention. It is especially true for senior managers (Mintzberg 1973, Jett & George 2003). The third reason is due to personality trait. Some people with high levels of impulsivity and sensation seeking often engage in multitasking because they are less able to block distractions (Sanbonmatsu, Strayer, Medeiros-Ward & Watson 2013).

While some studies (Lee & Taatgen 2002, Taatgen & Lee 2003) have found that multitasking can help to improve the skills of a worker in handling multiple tasks, the actual benefit of multitasking has yet to be discovered. Normally, *multitasking is not recommended* (Rosen 2008) and for some mission critical occasions, like flight control and air traffic control, multitasking must be avoided (Loukopoulos, Dismukes & Barshi 2009). Any interruption could cause catastrophic effect. A couple of reasons could explain why multitasking is not recommended. The first reason is due to the cost of job switching (Rogers & Monsell 1995, Rubinstein, Meyer & Evan 2001, Monsell 2003), including response delay (Meuter & Allport 1999) and resumption lag (Trafton, Altmann, Brock & Mintz 2003). (A good summary regarding switching costs in multitasking can be found in (APA 2006).) These delay and lag are job dependent (Gillie & Broadbent 1989). In a field experiment, it is found that the average resumption lag could be as long as 25 minutes (Mark, Gonzalez & Harris 2005, p.326). Rehearsal could reduce the response delay or resumption lag. But still, they can hardly be eliminated (Gillie & Broadbent 1989, Trafton, Altmann, Brock & Mintz 2003). The second is because multitasking could cause problem in learning (Foerde, Knowlton & Poldrack 2006) and lead to attention deficit trait (ADT) (Hallowell 2005). A person with ADT has a normal brain but his behavior is similar to a person with attention deficit disorder, problem in paying attention. The third reason is due to performance degradation. In some laboratory experiments, Staryer, Watson and co-workers (Watson & Strayer 2010, Sanbonmatsu, Strayer, Medeiros-Ward & Watson 2013) have found that the performance of a person in multitasking environment drops as compared with working in sequential processing environment. Only very few people are able to perform equally well in both multitasking and sequential processing environments (Watson & Strayer 2010). People perceived themselves good at multitasking usually underperform in both environments, as compared with the people perceived themselves not good at multitasking (Sanbonmatsu, Strayer, Medeiros-Ward & Watson 2013).

Various field studies have also found that *excessive multitasking could degrade the productivity of a worker*. By analyzing the historical data of an executive recruitment firm, Aral, Brynjolfsson

& Van Alstyne (2007) found that there is an inverted-U shaped relationship exists between multitasking and productivity. Beyond optimum, more multitasking is associated with declining project completion rates. By analyzing the historical data of the judges in an Italian court, Coviello, Ichino & Persico (2010) found that the judges who worked on few cases at a time tended to complete more cases and took less time. To attain a theoretical basis for this finding, Coviello, Ichino & Persico (2014) proposed a stochastic model for single-worker. In their model, it is assumed that (i) all the jobs are of the same type and same size, (ii) the worker could randomly pick any number of jobs in the queue, and (iii) in any moment of time the worker allocates effort equally to all the incomplete jobs. Coviello, Ichino & Persico (2014, Proposition 1) then show that the job completion rate increases as the pick-up rate increases until the pick-up rate has reached a certain value. After that value, the productivity decreases as the pick-up rate increases.

1.2. Key issue in scheduling with multitasking

As switching job will cause resumption lag, multitasking should be avoided. But in reality, a worker always gets interruptions by some clients who enquire the statuses of their jobs. The worker will response by terminating the current job, process each client's job a part of it and then report to the clients the statuses. Then, the terminated job is resumed. Clearly, these resumption lag and interruptions will make longer the completion time of a job. In such case, some important questions are aroused.

- *How to schedule the jobs, in the presence of multitasking, such that the performance of a worker can be optimized?*
- *What would be the performance difference between the worker whose schedule is optimized for handling jobs in the presence of multitasking and the worker whose schedule is optimized for handling jobs in the absence of multitasking?*

The latter question is of paramount important because it tells us what would be the effect of multitasking on operations scheduling. One approach (Hall, Leung & Li 2014) is to find the optimal schedules for both problems either in the presence or absence of multitasking, run simulations to figure out the performance of both schedules and then compare their performance. Another approach is to compare their performance by mathematical analysis. For sure, not all problems could be studied along this latter approach. In particular, those problems which are categorized as NP-hard or strongly NP-hard should be unlikely to take this approach. But we believe that some polynomial-time solvable problems should be possible.

Therefore, the key issue behind these two questions is to analyze the complexities of the scheduling problems in the presence of multitasking. For those problems which are categorized as NP-hard or strongly NP-hard (i.e. intractable), we anticipate that the comparisons could likely be conducted by simulations. For those problems which are categorized as polynomial-time solvable, it would be possible for us to conduct the comparisons by mathematical analysis. This is the primary reason why we present the complexity analysis in this paper.

1.3. Researches on scheduling with multitasking

Given a set of jobs, their processing times (job sizes), the setup times (switching costs) amongst jobs, the number of machines available and the criteria (productivity measure) to be optimized, scheduling theorists are interested to find a schedule, i.e. which job should be processed in which machine and in which time slots, such that the criteria is optimized. Hall, Leung & Li (2014) have recently introduced a new scheduling model, in which switching cost has been introduced to capture multitasking behavior. Even Hall, Leung & Li (2014) have not been aware, their model has implicitly assumed that the worker is self-motivated and the worker realizes that multitasking should be avoided (Rosen 2008). The worker switches to other job only when it is requested by the clients.

In Hall-Leung-Li model, a job J_i can be characterized by a 5-tuple $(p_i, d_i, w_i, g_i(\cdot), f_i(\cdot))$, where p_i, d_i and w_i denote the processing time, due date and weight of J_i as in classical models, $g_i(\cdot)$ the interruption-time function and $f_i(\cdot)$ the switching cost function. Let $S = (J_{\pi_1}, J_{\pi_2}, \cdots, J_{\pi_n})$ be a feasible schedule of a given set J of n jobs, where J_{π_i} is scheduled before $J_{\pi_{i+1}}$. A job J_{π_i} is called *primary* in a time period when it is scheduled to be processed in that period. When J_{π_i} becomes a primary job, all the jobs scheduled before it have completed the execution. The execution of a primary job J_{π_i} will be interrupted by job J_{π_j}, $i < j \leq n$, for $g_{\pi_j}(p'_{\pi_j})$ units of time, where p'_{π_j} is the remaining processing time of J_{π_j}. Hence, the remaining processing time of each unfinished job will be reduced by the amount that it interrupts a primary job. To precisely represent the remaining processing time of J_{π_i} after each interruption, we use $p_{\pi_i}(k)$ to denote the remaining processing time of J_{π_i} after the processing of J_{π_k} $(1 \leq k \leq i-1)$ has been completed.

$$p_{\pi_i}(k) = \begin{cases} p_{\pi_i} & \text{if } k = 0, \\ p_{\pi_i}(k-1) - g_{\pi_i}(p_{\pi_i}(k-1)) & \text{if } 1 \leq k \leq i-1, \\ 0 & \text{if } k \geq i \end{cases} \tag{1}$$

for $i = 1, 2, \cdots, n$. It is reasonable to assume that $p_{\pi_i}(k)$ in (1) is greater than zero for $k = 1, \cdots, i-1$. So, we assume that $\sum_{k=1}^{i-1} g_{\pi_i}(p_{\pi_i}(k-1)) < p_{\pi_i}$. The switching cost function $f_i(j)$ is defined as the cost occurring when J_j interrupts J_i. To simplify the notation, $f_i(j)$ is denoted as f_{ij} in the sequel.

Hall, Leung & Li (2014) investigate four different scheduling criteria, including (i) total weighted completion time, (ii) maximum lateness, (iii) total number of late jobs and (iv) total weighted number of late jobs, under the assumptions that (1) all the switching cost functions are constant functions, and (2) the late jobs are not discarded and still interrupt the execution of primary jobs. They showed that the problems of minimizing criteria (i) and (ii) are polynomial-time solvable, (iii) is NP-hard (i.e. *NP-hard in ordinary sense*) and (iv) is strongly NP-hard (i.e. *NP-hard in strong sense*). Furthermore, if the interruption-time function is defined as $g_{\pi_i}(p_{\pi_i}(k)) = Dp_{\pi_i}(k)$ and $0 \leq D < 1$, they showed that the problem of minimizing the criterion (iii) is polynomial-time solvable by developing an optimal algorithm generalized from the Hodgson-Moore algorithm. A dynamic programming algorithm was also developed for minimizing criterion (iv). The effect of multitasking under different criterion is demonstrated by simulations.

1.4. Behavioral economics researches on multitasking

Allocating effort to handle multiple jobs has long been a research problem in behavioral economics. As those studies appeared long before the recent findings about the negative impact of multitasking, such as the resumption lag (Trafton, Altmann, Brock & Mintz 2003) and the ADT (Hallowell 2005), those studies are definitely needed to be revised to cope with these findings. The problem is solved by two different approaches. The first approach is to model the problem as a Markov decision problem. A worker has to work on multiple jobs in infinite horizon. In each time slot, the worker can only devote his effort on one job. If a job is allocated with effort, its quality level will be either incremented or staying the same. If a job has not been allocated with effort, its quality level will be either decremented or staying the same. The problem is to determine in each time slot which job the effort should be allocated. So that all the quality levels can be optimized. Radner & Rothschild (1975), amongst the first group of researchers, worked on this problem. Instead of determine the optimal effort allocation rule, they analyzed how the quality level changes under three different effort allocation rules, namely *constant proportions*, *putting out fires* and *staying with winner* (see (Radner & Rothschild 1975, p.360)), and concluded that *constant proportions* and *putting out fires* are preferred.

The second approach is from a seminal work by Holmstrom & Milgram (1991). The effort allocation problem is modeled as a principal-agent problem (Itoh 1994, Prasad 2009). A worker (i.e. an agent) has to work on multiple jobs in a period of time and the worker has his own cost function to measure the effort he put. The payment to the worker is determined by the output of the worker. For simplicity, one can consider the quality level is an indicator of the output. The higher the quality level, the more payment will be got. The utility of the worker is thus determined by the payment he get minus the cost on the effort. The manager (i.e. the principal) has his profit function which is determined by the output of the worker. As a manager, he has to design the effort the worker will put on the jobs. So that both the profit is maximized as well as the expected utility of the worker is optimized (see (Holmstrom & Milgram 1991, Section 2)). Once the effort has been determined, it is assume that the worker will work on the jobs in accordance with the determined effort. In contrast to the classical scheduling and the Markov decision problem approach, principal-agent problem does not concern on the actual schedule, i.e. when the worker will put his effort in which job.

While a lot of models have been proposed and analyzed since after the works by (Radner & Rothschild 1975) and (Holmstrom & Milgram 1991), very few of them have introduced the idea of switching cost in their models. None of the follow-up works has considered the resumption lag in their problem settings. Rothschild (1974) added one more term to determine the change of the quality level, as compared with the model in (Radner & Rothschild 1975). If a job gets more consecutive steps of effort, he expected incremental change of a job's quality level will be more. In (Seshadri & Shapira 2001), the switching cost is defined in negative sense. Seshadri & Shapira (2001) consider that a worker has to handle one long-term primary job and many short-term secondary jobs. Similar to the problem setting in (Radner & Rothschild 1975), the worker in each time slot can only devote his effort on one job. If a job is allocated with effort, its quality level will be either incremented or staying the same. If a job has not been allocated with effort, its quality level will be either decremented or staying the same. But more than that, Seshadri & Shapira (2001) assume that the primary job will get an immediate decrement of the quality level if it is interrupted by a short-term secondary job. While Coviello, Ichino & Persico (2014) did mention that resumption lag is a cost of job switching cost, they did not include this factor in their analysis. If non-negative resumption lag is included, we can show that the job completion rate (as a measure of worker's productivity) will be zero. For the principal-agent models, including the works presented in some recent papers like (Corts 2007, Prasad 2009, Matsushima, Miyazaki & Yagi 2010), the effect of resumption lag has not been considered in their models. Therefore, allocation of effort to handle multiple jobs with consideration of switching cost is still an open issue in behavioral economics.

1.5. Objective of our paper

In this paper, our focus is on the second question regarding the scheduling of jobs so as to minimize the effect of multitasking. We continue the study on Hall-Leung-Li model but we consider more realistic situations. *First, the switching cost functions are allowed to be arbitrary functions and they are jobs dependent.* This consideration is in line with the findings in (Gillie & Broadbent 1989, Trafton, Altmann, Brock & Mintz 2003). Moreover, two types of switching costs are considered, asymmetric and symmetric. Switching cost function is asymmetric if $f_{ij} \neq f_{ji}$, and symmetric otherwise. *Second, when consider the problems related to late jobs, we assume that the late jobs are completely discarded.* It means that the late jobs will be excluded from a given job set after scheduling. This assumption is different from the one made in the studies done by Hall, Leung & Li (2014) in which all the jobs have to be processed. In real world, if a job is decided to be a late job to a worker, it will likely be assigned to other workers. The set of late jobs will no longer interrupt the on-time jobs. The performance of the worker will be improved. For the late jobs, if they are assigned to other workers, some of them could turn out to be finished on-time. So, we believe that our assumption is more practical.

With these new assumptions, six scheduling problems are investigated and their complexities are analyzed. Algorithms are developed for some of these problems. Based on our results and the complexity analysis presented in Lageweg, Lenstra, Lawler & Rinnooy Kan (1982), Hall, Leung & Li (2014), Lawler (1977), Lenstra, Rinnooy Kan & Brucker (1977) and Du & Leung (1990), the complexity hierarchy of the scheduling problems regrading human multitasking behavior is illustrated. All these results serve as the foundation for answering the very first question in regard to the effect of multitasking on the productivity of a worker. By simulation, it is not difficult to get the answer. However, we are interested in analytical solution. For the total completion time, we have derived a formulae expressing the total completion time in terms of the switching cost and the degree of multitasking. Due to the page limit, all these analytical results will be presented in another paper.

The rest of this paper is organized as follows. In the next section, the notations, system models and assumptions, and the problems to be investigated will be defined. The complexities of these problems are then presented from Section 3 to Section 8. Both asymmetric and symmetric switching costs are considered. Accordingly, the complexity hierarchy of the scheduling problems are compiled in Section 9. Concluding remarks are drawn in the last section. To be self-contained, Appendix A summarizes the results obtained by Hall, Leung & Li (2014) for the total number of late jobs problem and the total weighted number of late jobs problem.

2. Preliminaries

In this section, the notations to be used in the paper will be introduced. The assumptions on the interruption-time functions and switching costs will be stated. The problems to be investigated are formally defined.

2.1. Notations

To make the paper more concise, the following notations are used throughout the paper.

J	Set of n jobs $\{J_1, J_2, J_3, \cdots, J_n\}$.
$J_{O,S}$	A subset of J containing on-time jobs in schedule S.
$J_{L,S}$	A subset of J containing late jobs in schedule S.
S	A schedule of J represented as $\{J_{\pi_1}, J_{\pi_2}, \cdots, J_{\pi_n}\}$.
J_{π_k}	The k^{th} job to be processed as a primary one.
p_i	Processing time of job J_i.
d_i	Due date of job J_i.
w_i	Weighting factor associated with job J_i.
p'_i	Remaining processing time of job J_i.
$p_i(k)$	Remaining processing time of job J_i after it has interrupted k times.
$g_i(p'_i)$	Interruption-time function of job J_i.
f_{ij}	Switching cost incurred on job J_i if it is interrupted by job J_j.

Note that we assume that the switching costs are not time dependent but job dependent, and they can be positive or negative values. The following notations are related to the performance of processing J_i in S.

$c_i(S)$	Completion time of J_i in S.
$T_i(S)$	Tardiness of J_i in S,
	defined as $T_i(S) = \max\{0, c_i(S) - d_i\}$.
$L_i(S)$	Lateness of i in S,
	defined as $L_i(S) = c_i(S) - d_i$.
$U_i(S)$	$U_i(S) = 1$ if J_i is late, i.e. $c_i(S) > d_i$, in S; and 0 otherwise

The performance criteria related to a schedule S are denoted by the following notations.

$C_{max}(S)$	Makespan, i.e. $\max_{J_i \in J}\{c_i(S)\}$, of S,
$\sum_i c_i(S)$	Total completion time of S.
$\sum_i w_i c_i(S)$	Total weighted completion time of S.
$T_{max}(S)$	Maximum weighted tardiness of S,
	defined as $T_{max}(S) = \max_{J_i \in J}\{w_i T_i(S)\}$.
$L_{max}(S)$	Maximum weighted lateness of S,
	defined as $L_{max}(S) = \max_{J_i \in J}\{w_i L_i(S)\}$.
$\sum_i U_i(S)$	Total number of late jobs in S.
$\sum_i w_i U_i(S)$	Total weighted number of late jobs in S.
$\sum_i T_i(S)$	Total tardiness of S.
$\sum_i w_i T_i(S)$	Total weighted tardiness of S.

2.2. Assumptions

While Hall, Leung & Li (2014) have presented some of the problems in single machine scheduling with interruptions, they have not clearly made the conditions on the interruption-time functions and the switching functions. As mentioned in Hall, Leung & Li (2014), interruption could sometimes lead to positive effect on processing a primary job. To reflect this phenomena, the switching costs (i.e. f_{ij}) could thus be negative. In such case, we need the following inequalities to ensure that $0 < c_{\pi_1}(S) < c_{\pi_2}(S) < \cdots < c_{\pi_n}(S)$.

Assumption 1 *For $i = 1, 2, \cdots, n-1$,*

$$p_{\pi_i}(i-1) + \sum_{j=i+1}^{n} \left(g_{\pi_j}(p_{\pi_j}(i-1)) + f_{\pi_i \pi_j} \right) > 0 \tag{2}$$

and $p_{\pi_n}(n-1) > 0$.

Lemma 1 *If condition (2) is satisfied, $0 < c_{\pi_1} < c_{\pi_2} < \cdots < c_{\pi_n}$.*

Proof. Given p_i, d_i, w_i, $g_i(p'_i)$ and f_{ij} for all $i, j = 1, \cdots, n$, the remaining processing time of job J_i could be obtained by (1). Thus the completion time of J_{π_1} is given by

$$c_{\pi_1}(S) = p_{\pi_1}(0) + \sum_{j=2}^{n} \left(g_{\pi_j}(p_{\pi_j}(0)) + f_{\pi_1 \pi_j} \right).$$

The completion time of J_{π_i} for $i = 2, \cdots, n-1$ is given by

$$c_{\pi_i}(S) = c_{\pi_{i-1}}(S) + p_{\pi_i}(i-1) + \sum_{j=i+1}^{n} \left(g_{\pi_j}(p_{\pi_j}(i-1)) + f_{\pi_i \pi_j} \right).$$

The completion time of J_{π_n} is given by

$$c_{\pi_n}(S) = c_{\pi_{n-1}}(S) + p_{\pi_n}(n-1).$$

If condition (2) is satisfied, it is clear that $0 < c_{\pi_1} < c_{\pi_2} < \cdots < c_{\pi_n}$. The proof is completed. **Q.E.D.**

By the fact that $p_{\pi_i}(0) = p_{\pi_i}$, the completion times can also be written as follows :

$$c_{\pi_i}(S) = \sum_{j=1}^{i} \left(p_{\pi_j} + \sum_{j'=j+1}^{n} f_{\pi_j \pi_{j'}} \right) + \sum_{j=i+1}^{n} \left(\sum_{k=1}^{i} g_{\pi_j}(p_{\pi_j}(k-1)) \right) \tag{3}$$

for $i = 1, 2, \cdots, n-1$ and

$$c_{\pi_n}(S) = p_{\pi_n} + \sum_{j=1}^{n-1} \left(p_{\pi_j} + \sum_{j'=j+1}^{n} f_{\pi_j \pi_{j'}} \right). \tag{4}$$

2.3. Problems

With the notations and assumptions stated above, we now define the problems to be analyzed in this paper. Follow (Hall, Leung & Li 2014), we use 'mt' in the β field to denote 'multitasking'. For the late job problems, our definitions are different from those defined in (Hall, Leung & Li 2014). We assume that the late jobs will be discarded. They have no effect on the on-time jobs. For this reason, we add 'DLJ' in the β field to synonymize discarding late jobs.

Problem 1 (Makespan) *Given a set of jobs $J = \{J_1, J_2, \cdots, J_n\}$ with known p_i, w_i, $g_i(p'_i)$ and f_{ij} for all $i, j = 1, \cdots, n$, which satisfy the condition (2), find a feasible schedule $S = \{J_{\pi_1}, J_{\pi_2}, \cdots, J_{\pi_n}\}$ such that $C_{max}(S)$ is minimum among all feasible schedules. This problem is denoted as $1|mt, f_{ij}|C_{max}$ for the case of asymmetric switching costs and $1|mt, f_{ij} = f_{ji}|C_{max}$ for the case of symmetric switching costs.*

Problem 2 (Total Weighted Completion Time) *Given a set of jobs $J = \{J_1, J_2, \cdots, J_n\}$ with known p_i, w_i, $g_i(p'_i)$ and f_{ij} for all $i, j = 1, \cdots, n$, which satisfy the condition (2), find a schedule $S = \{J_{\pi_1}, J_{\pi_2}, \cdots, J_{\pi_n}\}$ such that $\sum_i w_{\pi_i} c_{\pi_i}(S)$ is minimum among all feasible schedules. This problem is denoted as $1|mt, f_{ij}|\sum_i w_i c_i$ for the case of asymmetric switching costs and $1|mt, f_{ij} = f_{ji}|\sum_i w_i c_i$ for the case of symmetric switching costs.*

Problem 3 (Max Weighted Tardiness) *Given a set of jobs $J = \{J_1, J_2, \cdots, J_n\}$ with known p_i, d_i, $w_i > 0$, $g_i(p'_i)$ and f_{ij} for all $i, j = 1, \cdots, n$, which satisfy the condition (2), find a schedule $S = \{J_{\pi_1}, J_{\pi_2}, \cdots, J_{\pi_n}\}$ such that the $T_{max}(S)$ is minimum among all feasible schedules. This problem is denoted as $1|mt, f_{ij}|T_{max}$ for the case of asymmetric switching costs and $1|mt, f_{ij} = f_{ji}|T_{max}$ for the case of symmetric switching costs.*

Problem 4 (Max Weighted Lateness) *Given a set of jobs $J = \{J_1, J_2, \cdots, J_n\}$ with known p_i, d_i, $w_i > 0$, $g_i(p'_i)$ and f_{ij} for all $i, j = 1, \cdots, n$, which satisfy the condition (2), find a schedule $S = \{\pi_1, \pi_2, \cdots, \pi_n\}$ such that $L_{max}(S)$ is minimum among all feasible schedules. This problem is denoted as $1|mt, f_{ij}|L_{max}$ for the case of asymmetric switching costs and $1|mt, f_{ij} = f_{ji}|L_{max}$ for the case of symmetric switching costs.*

Problem 5 (Total Number of Late Jobs) *Given a set of jobs $J = \{J_1, J_2, \cdots, J_n\}$ with known p_i, d_i, $g_i(p'_i)$ and f_{ij} for all $i, j = 1, \cdots, n$, which satisfy the condition (2), find a schedule S, in which the late jobs are completely discarded, such that the number of late jobs $\sum_{i \in S} U_i(S)$ is minimum among all feasible schedules. This problem is denoted as $1|mt, f_{ij}, DLJ|\sum_i U_i$ for the case* *of asymmetric switching costs and* $1|mt, f_{ij} = f_{ji}, DLJ|\sum_i U_i$ *for the case of symmetric switching costs.*

Problem 6 (Total Weighted Number of Late Jobs) *Given a set of jobs* $J = \{J_1, J_2, \cdots, J_n\}$ *with known* p_i*,* d_i*,* $w_i > 0$*,* $g_i(p_i')$ *and* f_{ij} *for all* $i, j = 1, \cdots, n$*, which satisfy the condition (2), find a schedule* S*, in which the late jobs are completely discarded, such that the weighted number of late jobs* $\sum_{i \in S} w_i U_i(S)$ *is minimum among all feasible schedules. This problem is denoted as* $1|mt, f_{ij}, DLJ|\sum_i w_i U_i$ *for the case of asymmetric switching costs and* $1|mt, f_{ij} = f_{ji}, DLJ|\sum_i w_i U_i$ *for the case of symmetric switching costs.*

3. Makespan

Makespan is one of the simplest criteria in scheduling. For asymmetric switching costs, it will be shown that the makespan problem is NP-hard. For symmetric switching costs, the makespans of all feasible schedules for a given job set are the same.

3.1. Asymmetric switching costs

Now, we consider the problem $1|mt, f_{ij}|C_{max}$. For a schedule $S = \{J_{\pi_1}, J_{\pi_2}, \cdots, J_{\pi_n}\}$,

$$C_{max}(S) = \sum_{i=1}^{n} p_{\pi_i} + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} f_{\pi_i \pi_j}.$$

As $\sum_{i=1}^{n} p_{\pi_i}$ is constant for any schedule, the problem $1|mt, f_{ij}|C_{max}$ is equivalent to finding a schedule S such that $\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} f_{\pi_i \pi_j}$ is minimum. Its NP-hardness is stated in the following theorem.

Theorem 1 *Given a set of jobs* $J = \{J_1, J_2, \cdots, J_n\}$ *with known* p_i*,* $g_i(p_i')$*,* f_{ij} *for all* $i, j = 1, \cdots, n$ *and condition (2) is satisfied, the problem* $1|mt, f_{ij}|C_{max}$ *is NP-hard.*

Proof. The proof is accomplished by reducing the feedback arc set problem to $1|mt, f_{ij}|C_{max}$. In accordance with Garey and Johnson (Garey & Johnson 1979, p.192 Problem GT8), the feedback arc set problem is defined as below.

Problem 7 (Feedback Arc Set (FAS)) *Given a directed graph $G = (V, A)$ and a positive integer $K \leq |A|$, find a subset $A' \subseteq A$ with $|A'| \leq K$ such that A' contains at least one arc from every directed cycle in G.*

Let $G = (V, A)$ be a arbitrary directed graph, where $V = \{V_1, V_2, \cdots, V_n\}$ is a vertex set with n vertices and an arc $(V_i, V_j) \in A$ denotes the arc from V_i to V_j, Given any instance of FAS, G and K, an instance of $1|mt, f_{ij}|C_{max}$ in decision version, J and y, can be constructed as follows. $|J| = n$ and $y = n^2 + K$. For each job $J_i \in J$ which corresponds to $V_i \in V$, its processing time, interruption-time function and the switching costs are defined as follows :

$$p_i = n, \tag{5}$$

$$g_i = 1, \tag{6}$$

$$f_{ij} = \begin{cases} 1 \text{ if } (V_j, V_i) \in A, \\ 0 \text{ otherwise.} \end{cases} \tag{7}$$

Note that the value of $f_{ij} = 1$ if $(V_j, V_i) \in A$ and 0 otherwise. It implies that all the non-zero switching costs incur only when J_j is scheduled after J_i.

($\Rightarrow$) Let $A' \subseteq A$ be an arc set such that $G' = (V, A \setminus A')$ is a directed acyclic graph and $|A'| \leq K$. A schedule S of J can be constructed by sequencing all the jobs according to the topological ordering of the corresponding vertices in G'. Without counting the switching costs introducing by the arcs in A', the makespan of S is n^2. Let $(V_{i_1}, V_{i_2}, \cdots, V_{i_r})$ be a directed path in G' and (V_{i_r}, V_{i_1}) be an arc in A'. It is obvious that scheduling $J_{i_1}, J_{i_2}, \cdots, J_{i_r}$ sequentially will introduce 0 switching cost. Since $(V_{i_r}, V_{i_1}) \in A'$, 1 time unit of switching cost will be added. As a result, the number of arcs in A' is equivalent to the total switching costs incurring in S. Therefore,

$$\begin{aligned} C_{max}(S) &= \sum_{i=1}^{n} p_i + \sum_{(V_j, V_i) \in A'} f_{ij} \\ &\leq n^2 + K \\ &= y. \end{aligned}$$

($\Leftarrow$) Let $S = \{J_{\pi_1}, J_{\pi_2}, \cdots, J_{\pi_n}\}$ be a schedule of the constructed instance such that $C_{max}(S) \le n^2 + K$. The arc set A' can be built as follows.

$$A' = \{(V_{\pi_i}, V_{\pi_j}) | J_{\pi_i} \text{ is scheduled after } J_{\pi_j} \text{ in } S\}$$

We now show that at least one arc in each directed cycle in G is in A'. Assume that there is a directed cycle $C = (V_{i_1}, V_{i_2}, \cdots, V_{i_r}, V_{i_1})$ in G such that none of the arcs in C is in A'. Since S is a single machine schedule, the jobs corresponding to the vertices in C are scheduled as a linear sequence. Therefore, there must exist a job pair $(J_{i_k}, J_{i_{k+1}})$ such that $(V_{i_k}, V_{i_{k+1}}) \in A$ and J_{i_k} is scheduled after $J_{i_{k+1}}$. From the construction of A', $(V_{i_k}, V_{i_{k+1}})$ must be in A', contradicting to the assumption. Since only the job pair corresponding to an arc in A' introduces 1 time unit of switching cost, the size of A' is equal to the total switching cost of S. The total processing time of all the jobs is n^2 and $C_{max}(S) \le n^2 + K$, implying that the total switching costs incurred in S is not greater than K. Hence, $|A'| \le K$.

As it has been proved by Karp (1972) that the feedback arc set problem is NP-Complete, $1|mt, f_{ij}|C_{max}$ must be NP-hard. **Q.E.D.**

3.2. Symmetric switching costs

If switching costs are symmetric, i.e. $f_{ij} = f_{ji}$, it can be shown that the makespans of all the schedules for a given task set are equal to a constant which is stated in the following theorem.

Theorem 2 *Given a set of jobs $J = \{J_1, J_2, \cdots, J_n\}$ with known p_i, $g_i(p'_i)$, $f_{ij} = f_{ji}$ for all $i, j = 1, \cdots, n$ and condition (2) is satisfied,*

$$C_{max}(S) = \sum_{i=1}^{n} p_i + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij}$$

for any schedule S.

Proof. From (4), we have that

$$C_{max}(S) = p_{\pi_n} + \sum_{j=1}^{n-1}\left(p_{\pi_j} + \sum_{j'=j+1}^{n} f_{\pi_j \pi_{j'}}\right)$$

$$= \sum_{j=1}^{n} p_{\pi_j} + \sum_{j=1}^{n-1} \sum_{j'=j+1}^{n} f_{\pi_j \pi_{j'}}. \tag{8}$$

The second term in the right hand side of the last equation is equal to the sum of the upper triangular elements of the matrix

$$\begin{bmatrix} 0 & f_{\pi_1\pi_2} & \cdots & f_{\pi_1\pi_i} & \cdots & f_{\pi_1\pi_{n-1}} & f_{\pi_1\pi_n} \\ f_{\pi_2\pi_1} & 0 & \cdots & f_{\pi_2\pi_i} & \cdots & f_{\pi_2\pi_{n-1}} & f_{\pi_2\pi_n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ f_{\pi_i\pi_1} & f_{\pi_1\pi_3} & \cdots & 0 & \cdots & f_{\pi_i\pi_{n-1}} & f_{\pi_i\pi_n} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \\ f_{\pi_{n-1}\pi_1} & f_{\pi_{n-1}\pi_2} & \cdots & f_{\pi_{n-1}\pi_i} & \cdots & 0 & f_{\pi_{n-1}\pi_n} \\ f_{\pi_n\pi_1} & f_{\pi_n\pi_2} & \cdots & f_{\pi_n\pi_i} & \cdots & f_{\pi_n\pi_{n-1}} & 0 \end{bmatrix}.$$

As the above matrix is symmetric and the diagonal elements are all zeros, the sum of the upper triangular elements is identical to the sum of the lower triangular elements. Moreover, this value is the same for all schedules. As a result, we can get from (8) that

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} f_{ij} = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij}.$$

Hence, for any schedule S,

$$C_{max}(S) = \sum_{i=1}^{n} p_i + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij}$$

and the proof is completed. **Q.E.D.**

4. Total Weighted Completion Time

In the first part of the section, the problems with asymmetric switching costs are analyzed. We show that the problem $1|mt, f_{ij}|\sum_i c_i$ is strongly NP-hard by reduction from the problem $1|prec|\sum_i c_i$. Then, the strongly NP-hardness of the problem $1|mt, f_{ij}|\sum_i w_i c_i$ is implied. In the second part of the section, the problem with symmetric switching costs are analyzed. We show that the problem $1|mt, f_{ij} = f_{ji}|\sum_i w_i c_i$ can be solved by a polynomial-time algorithm.

4.1. Asymmetric switching costs

Theorem 3 *Given a set of jobs $J = \{J_1, J_2, \cdots, J_n\}$ with known p_i, $g_i(p'_i)$ and f_{ij} for all $i, j = 1, \cdots, n$, and condition (2) is satisfied, the problem $1|mt, f_{ij}|\sum_i c_i$ is strongly NP-hard.*

Proof. The proof is accomplished by reducing the problem $1|prec|\sum_i c_i$ to the problem $1|mt, f_{ij}|\sum_i c_i$. Let (J, G, K) be an arbitrary instance of the problem $1|prec|\sum_i c_i$. $J = \{J_1, J_2, \cdots, J_n\}$ denotes a set of n job where p_i, $1 \leq i \leq n$ represent the processing times of J_i. All the jobs are available at time 0. The precedence constraints among jobs are represented by the directed acyclic graph $G = (J, A)$ in which, an arc $(J_i, J_j) \in A$ corresponds to the constraint that J_j cannot start executing before J_i completes. The decision version of $1|prec|\sum_i c_i$ is to ask if there is a schedule S such that the job sequence fulfills all the precedence constraints and $\sum_{J_i \in J} c_i(S) \leq K$.

Let (J^*, y) be the instance of the problem $1|mt, f_{ij}|\sum_i c_i$ constructed based on (J, G, K). The characteristics of each job J_i^*, $1 \leq i \leq n$, in J^* is shown as below. For all $i, j = 1, 2, \cdots, n$,

$$p_i^* = p_i \tag{9}$$

$$g_i^* = 0 \tag{10}$$

$$f_{ij}^* = \begin{cases} K+1 & \text{if } (J_j, J_i) \in A, \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

The threshold y is set to K.

$(\Rightarrow)$ Let $S = \{J_{\pi_1}, J_{\pi_2}, \cdots, J_{\pi_n}\}$ be a schedule fulfilling the precedence constraints and $\sum_{J_i \in J} c_i(S) \leq K$. We can build the schedule S^* for (J^*, y) by sequencing jobs in J^* as S. As S fulfills the precedence constraints, for all $(J_i, J_j) \in A$, J_i must be completed before J_j. It implies that J_i^* must also be completed before J_j^* starts. By the definition of the switching costs, $f_{ij}^* = K + 1$ only incurs when J_j^* is completed before J_i^*. Therefore, no switching cost incurs in the schedule S^*. The completion time of $J_{\pi_i}^*$ must be equal to that of J_{π_i}. As a result, $\sum_{J_i^* \in J^*} c_i(S^*) = \sum_{J_i \in J} c_i(S)$ and thus $\sum_{J_i^* \in J^*} c_i(S^*) \leq y$.

$(\Leftarrow)$ Let $S^* = \{J_{\pi_1}^*, J_{\pi_2}^*, \cdots, J_{\pi_n}^*\}$ be a schedule for (J^*, y) and $\sum_{J_i^* \in J^*} c_i(S^*) \leq y$. A schedule S for (J, G, K) can be formed by sequencing jobs in J as S^*. From the definitions of switching cost functions, it is easy to see that no switching cost is incurred in S^*. Therefore, S satisfies all the precedence constraints. Moreover, the completion time of J_{π_i} in S must be equal to that of $J_{\pi_i}^*$ in S^*. As a result, we can conclude that $\sum_{J_i \in J} c_i(S) \leq K$.

As $1|prec|\sum_i c_i$ has been proved to be strongly NP-hard (Lawler 1978, Lenstra & Rinnooy Kan 1978), $1|mt, f_{ij}|\sum_i c_i$ must be strongly NP-hard. **Q.E.D.**

As the problem $1|mt, f_{ij}|\sum_i c_i$ is strongly NP-hard, we can state without proof the complexity of $1|mt, f_{ij}|\sum_i w_i c_i$ in the following theorem.

Theorem 4 *Given a set of jobs $J = \{J_1, J_2, \cdots, J_n\}$ with known p_i, w_i, $g_i(p_i')$ and f_{ij} for all $i, j = 1, \cdots, n$, and condition (2) is satisfied, the problem $1|mt, f_{ij}|\sum_i w_i c_i$ is strongly NP-hard.*

4.2. Symmetric switching costs

Algorithm WCT, as shown in Figure 1, is a polynomial-time algorithm developed for the problem of minimizing the total weighted completion time if the switching costs are symmetric. There are two major steps in the algorithm. Step 1 is the preprocessing step by which the remaining processing times of all the jobs after they have interrupted k times, for $k = 1, 2, \cdots, n-1$, are calculated. Step 2 is the main step for constructing a schedule in backward fashion. Starting from scheduling the job J_{π_n}, the job $J_{\pi_{n-1}}$ is scheduled next and so on. The idea of the scheduling procedure can be described in the following.

Let $\tilde{S} = \{J_{\pi_{l+1}}, J_{\pi_{l+2}}, \cdots, J_{\pi_n}\}$. To determine which job should be scheduled as job J_{π_l}, we evaluate a value of each job J_i not scheduled in $\tilde{S}$ by using the following formula.

$$\frac{w_i}{p_i(l-1) + \sum_{J_j \in \tilde{S}} g_j(p_j(l-1)) + \sum_{J_j \in \tilde{S}} f_{ij}}. \tag{12}$$

Then, the job to be scheduled as J_{π_l} is determined by the following criteria.

$$\pi_l = \arg\min_i \left\{ \frac{w_i}{p_i(l-1) + \sum_{J_j \in \tilde{S}} g_j(p_j(l-1)) + \sum_{J_j \in \tilde{S}} f_{ij}} \right\}. \tag{13}$$

If there are more than one job having the same minimum value, we arbitrarily select one of them to be job J_{π_l}. The following theorem shows that Algorithm WCT can find a optimal schedule in polynomial-time.

```
Algorithm WCT

Step 1.
FOR $i$ from 1 to $n$,
  $p_i(0) = p_i$;
  FOR $k$ from 1 to $n-1$,
    Step 1.1.  $p_i(k) = p_i(k-1) - g_i(p_i(k-1))$;
  END
END

Step 2.
($\tilde{S}$: Sequence of scheduled jobs.)
$\tilde{S} = \{\}$; $J = \{J_1, J_2, ..., J_n\}$;
FOR $k$ from $n$ to 2 (Backward tracking)
  Step 2.1.  Evaluate $z_i = \sum_{J_j \in \tilde{S}} (g_j(p_j(k-1)) + f_{ij})$ for all $J_i \in J$;
  Step 2.2.  Evaluate $\frac{w_i}{p_i(k-1)+z_i}$ for all $J_i \in J$;
  Step 2.3.  Search all $J_{i'} \in J$ such that $i' = \min_{J_i \in J} \left\{ \frac{w_i}{p_i(k-1)+z_i} \right\}$;
  Step 2.4.  $MT = \left\{ J_{i'} \in J | J_{i'} = \min_{J_i \in J} \left\{ \frac{w_i}{p_i(k-1)+z_i} \right\} \right\}$;
  Step 2.5.  Select arbitrary $J_{i^*} \in MT$;
  Step 2.6.  Remove $J_{i^*}$ from the remaining list, i.e. $J = J \setminus \{J_{i^*}\}$;
  Step 2.7.  $\tilde{S} = \{J_{i^*}\} \| \tilde{S}$;
END
$\tilde{S} = J \| \tilde{S}$;
```

Figure 1 Algorithm WCT for minimizing the weighted completion time.

Theorem 5 *Given a set of jobs $J = \{J_1, J_2, \cdots, J_n\}$ with known p_i, w_i, $g_i(p'_i)$ and $f_{ij} = f_{ji}$ for all $i, j = 1, \cdots, n$, and condition (2) is satisfied, Algorithm WCT finds a optimal schedule for Problem 2, i.e. $1|mt, f_{ij} = f_{ji}|\sum_i w_i c_i$, in $\mathcal{O}(n^2)$ time.*

Proof. The proof is done by contradiction. First of all, the criterion (13) implies the following condition. For $l = 2, 3, \cdots, n$ and $\tilde{S} = \{J_{\pi_{l+1}}, \cdots, J_{\pi_n}\}$,

$$\frac{w_{\pi_{l-1}}}{p_{\pi_{l-1}}(l-1) + \sum_{J_j \in \tilde{S}} g_j(p_j(l-1)) + \sum_{J_j \in \tilde{S}} f_{\pi_{l-1} j}} \geq \frac{w_{\pi_l}}{p_{\pi_l}(l-1) + \sum_{J_j \in \tilde{S}} g_j(p_j(l-1)) + \sum_{J_j \in \tilde{S}} f_{\pi_l j}}. \tag{14}$$

We now assume that there exists a optimal schedule S' in which at least one pair of adjacent jobs, say job J_ν and job J_μ, does not fulfill the condition (14). Let S be the schedule constructed from S' by swapping J_ν and J_μ. Thus, we have that

$$S' = \{J_{\pi_1}, J_{\pi_2}, \cdots, J_{\pi_{i-1}}, J_\nu, J_\mu, J_{\pi_{i+2}}, \cdots, J_{\pi_n}\},$$
$$S = \{J_{\pi_1}, J_{\pi_2}, \cdots, J_{\pi_{i-1}}, J_\mu, J_\nu, J_{\pi_{i+2}}, \cdots, J_{\pi_n}\}.$$

From (3), we have

$$c_\mu(S) = c_{\pi_{i-1}} + p_\mu(i-1) + g_\nu(p_\nu(i-1)) + f_{\mu\nu} + \sum_{j=i+2}^{n} \left(g_{\pi_j}(p_{\pi_j}(i-1)) + f_{\mu\pi_j}\right). \tag{15}$$

$$\begin{aligned} c_\nu(S) = {} & c_{\pi_{i-1}} + p_\mu(i-1) + g_\nu(p_\nu(i-1)) + f_{\mu\nu} + \sum_{j=i+2}^{n} \left(g_{\pi_j}(p_{\pi_j}(i-1)) + f_{\mu\pi_j}\right) \\ & + p_\nu(i) + \sum_{j=i+2}^{n} \left(g_{\pi_j}(p_{\pi_j}(i)) + f_{\nu\pi_j}\right). \end{aligned} \tag{16}$$

Similar, for the schedule S', we have

$$c_\nu(S') = c_{\pi_{i-1}} + p_\nu(i-1) + g_\mu(p_\mu(i-1)) + f_{\nu\mu} + \sum_{j=i+2}^{n} \left(g_{\pi_j}(p_{\pi_j}(i-1)) + f_{\nu\pi_j}\right). \tag{17}$$

$$\begin{aligned} c_\mu(S') = {} & c_{\pi_{i-1}} + p_\nu(i-1) + g_\mu(p_\mu(i-1)) + f_{\nu\mu} + \sum_{j=i+2}^{n} \left(g_{\pi_j}(p_{\pi_j}(i-1)) + f_{\nu\pi_j}\right) \\ & + p_\mu(i) + \sum_{j=i+2}^{n} \left(g_{\pi_j}(p_{\pi_j}(i)) + f_{\mu\pi_j}\right). \end{aligned} \tag{18}$$

As $p_\mu(i-1) = p_\mu(i) + g_\mu(p_\mu(i-1))$, $p_\nu(i-1) = p_\nu(i) + g_\nu(p_\nu(i-1))$, and $f_{\nu\mu} = f_{\mu\nu}$, From (16) and (18), we know that

$$c_\nu(S) = c_\mu(S'). \tag{19}$$

For each of the jobs from 1 to $i-1$ and from $i+1$ to n, as the total switching costs and the total interruption-time from the waiting jobs are not affected by swapping J_μ and J_ν, the completion time of each of these jobs is the same in both S' and S, i.e.

$$c_{\pi_j}(S) = c_{\pi_j}(S'), \tag{20}$$

for $j = 1, \cdots, i-1, i+1, \cdots, n$. Therefore, the difference between $\sum_{i=1}^n w_i c_i(S)$ and $\sum_{i=1}^n w_i c_i(S')$ is given by

$$\sum_{i=1}^n w_i c_i(S) - \sum_{i=1}^n w_i c_i(S') = (w_\mu c_\mu(S) + w_\nu c_\nu(S)) - (w_\nu c_\nu(S') + w_\mu c_\mu(S')). \tag{21}$$

Putting (15), (16), (17), (18) in (21) and further by the fact that $f_{ij} = f_{ji}$, we can get that

$$\begin{aligned}\sum_{i=1}^n w_i c_i(S) - \sum_{i=1}^n w_i c_i(S') &= -w_\mu \left(p_\nu(i) + \sum_{j=i+2}^n g_{\pi_j}(p_{\pi_j}(i)) + \sum_{j=i+2}^n f_{\nu\pi_j} \right) \\ &\quad + w_\nu \left(p_\mu(i) + \sum_{j=i+2}^n g_{\pi_j}(p_{\pi_j}(i)) + \sum_{j=i+2}^n f_{\mu\pi_j} \right).\end{aligned} \tag{22}$$

Based on the assumption that jobs J_ν and J_μ do not fulfill the condition (14), we can get that

$$\frac{w_\nu}{p_\nu(i) + \sum_{j=i+2}^n g_{\pi_j}(p_{\pi_j}(i)) + \sum_{j=i+2}^n f_{\nu\pi_j}} < \frac{w_\mu}{p_\mu(i) + \sum_{j=i+2}^n g_{\pi_j}(p_{\pi_j}(i)) + \sum_{j=i+2}^n f_{\mu\pi_j}}.$$

As a result, $\sum_{i=1}^n w_i c_i(S) < \sum_{i=1}^n w_i c_i(S')$, which contradicts the assumption that S' is optimal. Therefore, Algorithm WCT is able to find a optimal schedule.

Step 1 in Algorithm WCT requires $\mathcal{O}(n^2)$ time. In Step 2, Step 2.2 and Step 2.3 each requires $\mathcal{O}(n)$ time. Repeating k from n to 2, the total number is of order $\mathcal{O}(n^2)$ time. For Step 2.1, it requires $(n-k)k$ running time for each value of k. Summing up for k from n to 2, the total number is of order $\mathcal{O}(n^2)$. Thus, Step 2 requires $\mathcal{O}(n^2)$ time. The overall running time of the algorithm is $\mathcal{O}(n^2)$. The proof is completed. **Q.E.D.**

5. Maximum Weighted Tardiness

In this section, the complexities of the problem $1|mt, f_{ij}|T_{max}$ and the problem $1|mt, f_{ij} = f_{ji}|T_{max}$ will be analyzed. With asymmetric switching costs, the NP-hardness is basically implied from Theorem 1. While with symmetric switching costs, a polynomial-time algorithm is developed.

5.1. Asymmetric switching costs

The complexity of $1|mt, f_{ij}|T_{max}$ is stated in the following theorem.

Theorem 6 *Given a set of jobs $J = \{J_1, J_2, \cdots, J_n\}$ with known p_i, d_i, $w_i > 0$, $g_i(p'_i)$, f_{ij} for all $i, j = 1, \cdots, n$ for all $i = 1, \cdots, n$, and condition (2) is satisfied, the problem $1|mt, f_{ij}|T_{max}$ is NP-hard.*

Proof. It is implied from Theorem 1. **Q.E.D.**

5.2. Symmetric switching costs

Inspired by Lawler (1973) and Pinedo (Pinedo 1995, p.33 Algorithm 3.2.1), Figure 2 shows an algorithm (Algorithm WT) which can find a optimal schedule for a given job set.

Algorithm WT starts by selecting job J_{π_n}, then job $J_{\pi_{n-1}}$ and so on. As from Theorem 2, we know that, for the case of symmetric switching costs, the makespan of any schedule must be equal to $\sum_{i=1}^n p_i + \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n f_{ij}$. So we can set the completion time of the last job in a schedule S as that $c_{\pi_n} = \sum_{i=1}^n p_i + \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n f_{ij}$. From this time point, we can calculate the weighted tardiness of each job and then schedule the job with minimum weighted tardiness as J_{π_n}. Once J_{π_n} has been selected, with the information of interruption-time function and the remaining processing time, $c_{\pi_{n-1}}(S)$ can thus be obtained. Then, we apply the same technique to select the job with minimum weighted tardiness as $J_{\pi_{n-1}}$. The same procedure repeats until job J_{π_2} has been determined. Accordingly, the adjacent pair of jobs of the schedule S must fulfill the following condition.

$$\max\{0, w_{\pi_{i-1}}(c_{\pi_i}(S) - d_{\pi_{i-1}})\} \geq \max\{0, w_{\pi_i}(c_{\pi_i}(S) - d_{\pi_i})\} \tag{23}$$

```
Algorithm WT

Step 1.
FOR $i$ from 1 to $n$,
  $p_i(0) = p_i$;
  FOR $k$ from 1 to $n-1$,
    Step 1.1.  $p_i(k) = p_i(k-1) - g_i(p_i(k-1))$;
  END
END

Step 2.
($\tilde{S}$: Sequence of scheduled jobs.)
$\tilde{S} = \{\}$; $J = \{J_1, J_2, ..., J_n\}$;
$c = \sum_{j=1}^{n} p_j + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} f_{ij}$;
FOR $k$ from $n$ to 2 (Backward tracking)
  Step 2.1.  Evaluate $\max\{0, w_i(c-d_i)\}$ for all $J_i \in J$;
  Step 2.2.  Search all $J_{i'} \in J$ such that $J_{i'} = \min_{J_i \in J} \max\{0, w_i(c-d_i)\}\}$;
  Step 2.3.  $MT = \{J_{i'} \in J | J_{i'} = \min_{J_i \in J}\{\max\{0, w_i(c-d_i)\}\}\}$;
  Step 2.4.  Select $J_{i^*} \in MT$ such that
               $J_{i^*} = \max_{J_i \in MT}\{p_i(k-1) + \sum_{J_j \in \tilde{S}} g_j(p_i(k-2))) + \sum_{J_j \in \tilde{S}} f_{ij}\}$;
  Step 2.5.  Remove $J_{i^*}$ from the remaining list, i.e. $J = J \setminus \{J_{i^*}\}$;
  Step 2.6.  $c = c - p_{i^*}(k-1) - \sum_{J_j \in \tilde{S}} g_j(p_j(k-2)) - \sum_{J_j \in \tilde{S}} f_{ij}$;
  Step 2.7.  $\tilde{S} = \{J_{i^*}\} \| \tilde{S}$;
END
$\tilde{S} = J \| \tilde{S}$;
```

Figure 2 Algorithm WT for minimizing the maximum weighted tardiness with symmetric switching costs.

for $i = 2, \cdots, n$.

To prove the optimality of Algorithm WT, we need to show a characteristic regarding job completion time in a schedule as the following lemma.

Lemma 2 *Given any schedule $S = \{J_{\pi_1}, J_{\pi_2}, \cdots, J_{\pi_{r-1}}, J_{\pi_r}, J_{\pi_{r+1}}, \cdots, J_{\pi_s}, J_{\pi_{s+1}}, \cdots, J_{\pi_n}\}$, we let S' be the schedule obtained from S by rescheduling J_{π_r} immediately after J_{π_s} (i.e. $S' = \{J_{\pi_1}, J_{\pi_2}, \cdots, J_{\pi_{r-1}}, J_{\pi_{r+1}}, \cdots, J_{\pi_s}, J_{\pi_r}, J_{\pi_{s+1}}, \cdots, J_{\pi_n}\}$). If $f_{ij} = f_{ji}$ for all $1 \le i, j \le n$ and condition (2) is satisfied, then $c_{\pi_i}(S') < c_{\pi_i}(S)$ for all $i = r+1, r+2, \cdots, s$ and $c_{\pi_r}(S') = c_{\pi_s}(S)$.*

Proof. By (3), we can express the completion times of J_{π_i} in the schedules S and S' for $i = r+1, r+2, \cdots, s$ as follows :

$$\begin{aligned} c_{\pi_i}(S) = {} & p_{\pi_1} + \cdots + p_{\pi_{r-1}} + p_{\pi_r} + p_{\pi_{r+1}} + \cdots + p_{\pi_i} + \sum_{j=2}^{n} f_{\pi_1\pi_j} + \cdots + \sum_{j=r}^{n} f_{\pi_{r-1}\pi_j} + \sum_{j=r+1}^{n} f_{\pi_r\pi_j} \\ & + \sum_{j=r+2}^{n} f_{\pi_{r+1}\pi_j} + \cdots + \sum_{j=i+1}^{n} f_{\pi_i\pi_j} + \sum_{j=i+1}^{n} g_{\pi_j}(p_{\pi_j}(0)) + \cdots + \sum_{j=i+1}^{n} g_{\pi_j}(p_{\pi_j}(i-1)), \end{aligned} \tag{24}$$

$$\begin{aligned} c_{\pi_i}(S') = {} & p_{\pi_1} + \cdots + p_{\pi_{r-1}} + p_{\pi_{r+1}} + \cdots + p_{\pi_i} + \sum_{j=2}^{n} f_{\pi_1\pi_j} + \cdots + \sum_{j=r}^{n} f_{\pi_{r-1}\pi_j} \\ & + \left(\sum_{j=r+2}^{n} f_{\pi_{r+1}\pi_j} + f_{\pi_{r+1}\pi_r} \right) + \cdots + \left(\sum_{j=i+1}^{n} f_{\pi_i\pi_j} + f_{\pi_i\pi_r} \right) \\ & + \left(\sum_{j=i+1}^{n} g_{\pi_j}(p_{\pi_j}(0)) + g_{\pi_r}(p_{\pi_r}(0)) \right) + \cdots + \left(\sum_{j=i+1}^{n} g_{\pi_j}(p_{\pi_j}(i-2)) + g_{\pi_r}(p_{\pi_r}(i-2)) \right). \end{aligned} \tag{25}$$

Comparing (24) and (25), the difference between $c_{\pi_i}(S')$ and $c_{\pi_i}(S)$ can be written as follows :

$$c_{\pi_i}(S') - c_{\pi_i}(S) = \sum_{j=r+1}^{i} f_{\pi_j\pi_r} + \sum_{j=0}^{i-2} g_{\pi_r}(p_{\pi_r}(j)) - p_{\pi_r} - \sum_{j=r+1}^{n} f_{\pi_r\pi_j} - \sum_{j=i+1}^{n} g_{\pi_j}(p_{\pi_j}(i-1)). \tag{26}$$

Since $f_{ij} = f_{ji}$, we can get that

$$\begin{aligned} c_{\pi_i}(S') - c_{\pi_i}(S) &= -\left(p_{\pi_r} - \sum_{j=0}^{i-2} g_{\pi_r}(p_{\pi_r}(j)) \right) - \sum_{j=i+1}^{n} f_{\pi_r\pi_j} - \sum_{j=i+1}^{n} g_{\pi_j}(p_{\pi_j}(i-1)) \\ &= -p_{\pi_r}(i-1) - \sum_{j=i+1}^{n} f_{\pi_r\pi_j} - \sum_{j=i+1}^{n} g_{\pi_j}(p_{\pi_j}(i-1)). \end{aligned} \tag{27}$$

Further by (2), we get that $c_{\pi_i}(S) < c_{\pi_i}(S')$. Consider J_{π_r},

$$c_{\pi_r}(S') = c_{\pi_s}(S') + p_{\pi_r}(s-1) + \sum_{j=s+1}^{n} f_{\pi_r \pi_j} + \sum_{j=s+1}^{n} g_{\pi_j}(p_{\pi_j}(s-1)).$$

By (27), it is clear that $c_{\pi_r}(S') = c_{\pi_s}(S)$. The proof is completed. **Q.E.D.**

Theorem 7 *Given a set of jobs $J = \{J_1, J_2, \cdots, J_n\}$ with known p_i, d_i, $w_i > 0$, $g_i(p'_i)$ and $f_{ij} = f_{ji}$ for all $i, j = 1, \cdots, n$, and condition (2) is satisfied, Algorithm WT finds a optimal schedule for Problem 3, i.e. $1|mt, f_{ij} = f_{ji}|T_{max}$, in $\mathcal{O}(n^2)$ time.*

Proof. Let S be the schedule generated by Algorithm WT and S' be a optimal schedule for the given job set J. Assume that S' is different from S. Compare S and S' starting from the last scheduled job. Let s be the largest index such that $J_{\pi_s} \neq J_{\pi'_s}$. It implies that, in S', J_{π_s} must be scheduled before $J_{\pi'_s}$ in S', say $J_{\pi'_r} = J_{\pi_s}$. Then, we can generate a schedule S'' from S' by rescheduling $J_{\pi'_r}$ immediately after $J_{\pi'_s}$, i.e.

$$S'' = \{J_{\pi'_1}, \cdots, J_{\pi'_{r-1}}, J_{\pi'_{r+1}}, \cdots, J_{\pi'_s}, J_{\pi'_r}, J_{\pi'_{s+1}}, \cdots, J_{\pi'_n}\}.$$

Note that the completion times of $J_{\pi'_1}, \cdots, J_{\pi'_{r-1}}, J_{\pi'_{s+1}}, \cdots, J_{\pi'_n}$ in S' are the same in S''. It implies that the weighted tardiness of $J_{\pi'_1}, \cdots, J_{\pi'_{r-1}}, J_{\pi'_{s+1}}, \cdots, J_{\pi'_n}$ in S'' are the same as that in S'. For $J_{\pi'_{r+1}}, J_{\pi'_{r+2}}, \cdots, J_{\pi'_s}$ in S'', we can conclude from Lemma 2 that their completion times must be strictly less than those in S'. Since the weighted tardiness function is monotonic increasing function of the job completion time for a given positive weight, the weighted tardiness of these jobs in S'' must be less than that in S'. Combining with the fact that the weighted tardiness of $J_{\pi'_r}$ is not greater than that of $J_{\pi'_s}$ with respect to the time point $c_{\pi'_r}(S'') = c_{\pi'_s}(S')$, the maximum weighted tardiness of S'' is not greater than that of S'.

By repeating the above procedure if the newly constructed schedule is different from S. Eventually, S' can be transformed into S without increasing the maximum weighted tardiness. Hence, S, which is generated by Algorithm WT, is optimal.

Step 1 in the Algorithm WT consists of two FOR loops for evaluating $g_{ik} = g_i(p'_i)$, $p'_i = p'_i - g_{ik}$ and $p_{ik} = p'_i$, its complexity is $\mathcal{O}(n^2)$. For Step 2, there is one FOR loop for $k = n, \cdots, 2$. The

Step 2.2 inside the FOR loop requires $|J|-1$ comparisons. The complexity of the second part of the algorithm is again $\mathcal{O}(n^2)$. Therefore, complexity of Algorithm WT is $\mathcal{O}(n^2)$. The proof is completed. **Q.E.D.**

A special case of the above problem is that all the weights are equal. The schedule can be found by the earliest due date (EDD) rule.

Theorem 8 *Given a set of jobs $J=\{J_1,J_2,\cdots,J_n\}$ with known p_i, d_i, $w_i=w$, $g_i(p_i')$ and $f_{ij}=f_{ji}$ for all $i,j=1,\cdots,n$, and condition (2) is satisfied, a optimal schedule for the problem $1|mt,w_i=w,f_{ij}=f_{ji}|T_{max}$ can be obtained by EDD rule in $\mathcal{O}(n\log(n))$ time.*

Proof. The main trick is in the Step 2 in Algorithm WT. First, we assume that job J_{π_i} has earlier due date than job J_{π_j} if $i<j$. That is to say, $d_{\pi_1}<d_{\pi_2}<\cdots<d_{\pi_n}$. Let us start from the time $c=\sum_{i=1}^n p_i+\frac{1}{2}\sum_{i=1}^n\sum_{j=1}^n f_{ij}$. Since all the weighted tardiness functions are the same, we can get that

$$\max\{0,w(c-d_{\pi_n})\}<\max\{0,w(c-d_{\pi_{n-1}})\}<\cdots<\max\{0,w(c-d_{\pi_1})\},$$

where w is the common weight. In regard to the Step 2 in Algorithm WT, it is easy to see that at each time point the job with latest due date is selected. Hence, An optimal schedule can be obtained by simply using EDD rule. Therefore, the time complexity of finding an optimal schedule for a job set with equal weights is $O(n\log(n))$. **Q.E.D.**

6. Maximum Weighted Lateness

In this section, the complexities of the problem $1|mt,f_{ij}|L_{max}$ and the problem $1|mt,f_{ij}=f_{ji}|L_{max}$ will be analyzed. With asymmetric switching costs, the NP-hardness is basically implied from Theorem 1. While with symmetric switching costs, a polynomial-time algorithm similar to Algorithm WT is developed.

6.1. Asymmetric switching costs

The complexity of $1|mt,f_{ij}|L_{max}$ is stated in the following theorem.

Theorem 9 *Given a set of jobs $J = \{J_1, J_2, \cdots, J_n\}$ with known p_i, d_i, $w_i > 0$, $g_i(p_i')$, f_{ij} for all $i, j = 1, \cdots, n$ for all $i = 1, \cdots, n$, and condition (2) is satisfied, the problem $1|mt, f_{ij}|L_{max}$ is NP-hard.*

Proof. It is implied from Theorem 1. **Q.E.D.**

6.2. Symmetric switching costs

It should be noted that Algorithm WT presented in the last section (see Figure 2) can equally be applied to solve the maximum weighted lateness problem by simply replacing $\max\{0, w_i(c - d_i)\}$ in the Step 2.1-2.3 by $w_i(c - d_i)$. Figure 3 shows the algorithm for solving the maximum weighted lateness problem.

The proof of optimality and complexity is similar to the proof for the Algorithm WT in the maximum weighted tardiness problem. Thus, we state without proof the following theorems for the Algorithm WL.

Theorem 10 *Given a set of jobs $J = \{J_1, J_2, \cdots, J_n\}$ with known p_i, d_i, $w_i > 0$, $g_i(p_i')$ and $f_{ij} = f_{ji}$ for all $i, j = 1, \cdots, n$, and condition (2) is satisfied, Algorithm WL finds a optimal schedule for Problem 4, i.e. $1|mt, f_{ij} = f_{ji}|L_{max}$, in $\mathcal{O}(n^2)$ time.*

Theorem 11 *Given a set of jobs $J = \{J_1, J_2, \cdots, J_n\}$ with known p_i, d_i, $w_i = w$, $g_i(p_i')$ and $f_{ij} = f_{ji}$ for all $i, j = 1, \cdots, n$, and condition (2) is satisfied, a optimal schedule for the problem $1|mt, w_i = w, f_{ij} = f_{ji}|L_{max}$ can be obtained by EDD rule in $\mathcal{O}(n \log(n))$ time.*

7. Total Number of Late Jobs

In this and the next section, we work on the problems of minimizing the total number of late jobs and total weighted number of late jobs. In (Hall, Leung & Li 2014), late jobs are scheduled after all on-time jobs have been completed. In our setting, we assume that all the late jobs are completely discarded. In other words, late jobs will not interrupt any on-time jobs. For reference, we state in Appendix A the theorems which are proved by Hall, Leung & Li (2014) in regard to the late job

Algorithm WL

```
Step 1.
FOR $i$ from 1 to $n$,
    $p_i(0) = p_i$;
    FOR $k$ from 1 to $n-1$,
        Step 1.1.   $p_i(k) = p_i(k-1) - g_i(p_i(k-1))$;
    END
END

Step 2.
($\tilde{S}$: Sequence of scheduled jobs.)
$\tilde{S} = \{\}$; $J = \{J_1, J_2, ..., J_n\}$; $c = \sum_{j=1}^{n} p_j + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} f_{ij}$;
FOR $k$ from $n$ to 2 (Backward tracking)
    Step 2.1.   Evaluate $w_i(c - d_i)$ for all $J_i \in J$;
    Step 2.2.   Search all $J_{i'} \in J$ such that $J_{i'} = \min_{J_i \in J}\{w_i(c - d_i)\}$;
    Step 2.3.   $MT = \{J_{i'} \in J | J_{i'} = \min_{J_i \in J}\{w_i(c - d_i)\}\}$;
    Step 2.4.   Select $J_{i^*} \in MT$ such that
                    $J_{i^*} = \max_{J_i \in MT}\{p_i(k-1) + \sum_{J_j \in \tilde{S}} g_j(p_i(k-2))) + \sum_{J_j \in \tilde{S}} f_{ij}\}$;
    Step 2.5.   Remove $J_{i^*}$ from the remaining list, i.e. $J = J \setminus \{J_{i^*}\}$;
    Step 2.6.   $c = c - p_{i^*}(k-1) - \sum_{J_j \in \tilde{S}} g_j(p_j(k-2)) - \sum_{J_j \in \tilde{S}} f_{ij}$;
    Step 2.7.   $\tilde{S} = \{J_{i^*}\} \| \tilde{S}$;
END
$\tilde{S} = J \| \tilde{S}$;
```

Figure 3 Algorithm WL for minimizing the maximum weighted lateness with symmetric switching costs.

problems. In this section, we consider both $1|mt, f_{ij}, DLJ|\sum_i U_i$ and $1|mt, f_{ij} = f_{ji}, DLJ|\sum_i U_i$ problems.

7.1. Asymmetric switching costs

For asymmetric switching costs, the time complexity of the problem $1|mt, f_{ij}, DLJ|\sum_i U_i$ is stated in the following theorem.

Theorem 12 *Problem 5 with asymmetric switching costs and the late jobs are discarded, i.e. $1|mt, f_{ij}, DLJ|\sum_i U_i$, is strongly NP-hard.*

Proof. The proof is conducted by reducing from Exact Cover by 3-Sets (X3C). X3C problem has been proved to be strongly NP-hard (Garey & Johnson 1979).

Problem 8 (Exact Cover by 3-Sets (X3C)) *An X3C instance contains a finite set $X = \{x_1, x_2, \cdots, x_{3q}\}$, where q is a positive integer, and a collection $C = \{C_1, C_2, \cdots, C_r\}$ of 3-element subsets of X, where $r \geq q$. The X3C problem is to ask if there exists a $C' \subseteq C$ such that every element of X appears in exactly one member of C'. In other words, all the members in C' are mutually exclusive and the union of all the members in C' equals X.*

Without loss of the generality, we map each element in X to a unique integer i, where $i \in \{1, 2, \cdots, 3q\}$. Each $C_j \in C$ is denoted by $\{x_{j,1}, x_{j,2}, x_{j,3}\}$, where $x_{j,1} < x_{j,2} < x_{j,3}$.

Given an arbitrary instance of X3C, the corresponding instance of $1|mt, f_{ij}, DLJ|\sum U_i$ with the assumption that all the late jobs are discarded can be constructed. First, a job set $J = \{J_1, J_2, \cdots, J_{3q}, J_{3q+1}, \cdots, J_{3q+r}\}$ is defined. For job $J_i \in J$, its processing time and due date are defined as follows :

$$p_i = \begin{cases} 1 & \text{if } i = 1, 2, 3, \cdots, 3q. \\ M & \text{if } i = 3q+1, 3q+2, \cdots, 3q+r, \end{cases} \tag{28}$$

$$d_i = \begin{cases} \sum_{k=1}^{i}(q+1+k) & \text{if } i = 1, 2, 3, \cdots, 3q. \\ q(M-3q) + \sum_{k=1}^{3q}(q+1+k) & \text{if } i = 3q+1, 3q+2, \cdots, 3q+r, \end{cases} \tag{29}$$

Note that

$$\sum_{k=1}^{i}(q+1+k) = i(q+1)+\frac{i(i+1)}{2},$$
$$q(M-3q)+\sum_{k=1}^{3q}(q+1+k) = qM+3q+\frac{3q(3q+1)}{2}.$$

In (28) and (29), the value of M must fulfill the following inequality.

$$M > 3q(q+1)+\frac{3q(3q+1)}{2}. \tag{30}$$

The interruption-time function is defined as follows :

$$g_i(p') = \begin{cases} 0 \text{ if } i = 1,2,\cdots,3q, \\ 1 \text{ if } i = 3q+1, 3q+2, \cdots, 3q+r. \end{cases} \tag{31}$$

The switching cost is defined as follows :

$$f_{ij} = \begin{cases} i & \text{if } i = 1,\cdots,3q,\ j = 3q+1,\cdots,3q+r \text{ and } i \in C_j, \\ M & \text{if } i = 3q+1,\cdots,3q+r \text{ and } j = 1,\cdots,3q, \\ 0 & \text{otherwise.} \end{cases} \tag{32}$$

The threshold cost y is defined as

$$y = (r-q). \tag{33}$$

For the sake of presentation, we call each of the jobs J_1, J_2, $\cdots$, J_{3q} the *element job* as it corresponds to an element in X. Each of the jobs J_{3q+1}, J_{3q+2}, $\cdots$, J_{3q+r} is called the *member job* as it corresponds to a member in C.

($\Rightarrow$) Let $C' = \{C_{j_1},\cdots,C_{j_q}\}$ be a subset of C such that $\bigcup_{k=1}^{q} C_{j_k} = X$ and $C_{j_m} \bigcap C_{j_n} = \phi$ for $m \neq n$. A schedule S with $(r-q)$ late jobs can be constructed by three steps.

Step 1: Schedule all the element jobs according to the earliest due date (EDD) rule.

Step 2: All the member jobs corresponding to the members in C' are scheduled after the element jobs in arbitrary order.

Step 3: The member jobs corresponding to the members not in C' are late jobs and discarded.

From the definition of the interruption-time functions, each on-time member job J_{3q+j} interrupts all element jobs one unit of time. Besides, from the definition of the switching costs, J_{3q+j} introduces $x_{j,1}$, $x_{j,2}$ and $x_{j,3}$ switching costs to the element jobs $J_{x_{j,1}}$, $J_{x_{j,2}}$ and $J_{x_{j,3}}$ if $C_j = \{x_{j,1}, x_{j,2}, x_{j,3}\}$ is in C'. Since all the members in C' are mutually exclusive, $J_{x_{j,1}}$, $J_{x_{j,2}}$ and $J_{x_{j,3}}$ will need to process $(q+1)+x_{j,1}$, $(q+1)+x_{j,2}$, and $(q+1)+x_{j,3}$, units of time. In other words, the completion time of job J_i for $i = 1, \cdots, 3q$ is $\sum_{k=1}^{i}(q+1+k)$. All element jobs complete their execution by their due dates. The completion time of the last on-time member job is equal to the summation of the processing times of the $3q$ element jobs, the processing times of the q member jobs and the switching costs introduced in the element jobs. So, we get that

$$C_{max} = qM + 3q + \frac{3q(3q+1)}{2}. \tag{34}$$

All the element jobs and the member jobs corresponding to the members in C' are on-time in S. The total number of late jobs is $(r-q) \leq y$.

($\Leftarrow$) Let S be a schedule for the constructed job set J with total number of late jobs less than or equal to $y = (r-q)$. Before we proceed to prove the only-if part, we need to show that the schedule S must satisfy the following properties.

Property 1: Removing inserted idle time between jobs in S will not increase the total number of late jobs.

Property 2: There are exactly q on-time member jobs.

Property 3: All the element jobs must be on-time.

Property 4: All the element jobs must be scheduled before the on-time member jobs.

Property 5: The intersection of any two members corresponding to on-time member jobs in S must be empty.

Since all jobs in the constructed job set are available at time zero, left-shifting an on-time job to eliminate idle time does not make it late. So, **Property 1** holds.

Property 2 is proved by contradiction. Assume $q+v$, where $v \geq 1$, member jobs are scheduled in S. The completion time of the last job in these $q+v$ member jobs must be larger than $(q+v)M$. As $M > 3q+3q(3q+1)/2$, at least one of these $q+v$ member jobs must be late. Therefore, it is not possible to have more than q on-time member jobs in the schedule S. On the other hand, if there are only $q-v$, where $v \geq 1$, on-time member jobs, the number of late jobs will be larger than $(r-q)$. It will violate the condition that the number of late jobs must be less than or equal to $(r-q)$. We can conclude that the number of on-time member jobs in S must be exactly q. **Property 2** holds.

Property 3 is an implication from **Property 2** and the threshold $(r-q)$. As there must be exactly q on-time member jobs and the total number of late jobs must be less or equal to $(r-q)$, $3q$ element jobs must be on-time.

Property 4 is proved by contradiction. If any one of the on-time member job is scheduled just before an element job, say J_i, it will introduce a switching cost M on the on-time member job. Thus, it makes $J_i, J_{i+1}, \cdots, J_{3q}$ late. It contradicts that the total number of late jobs must be $(r-q)$.

Property 5 is proved by using **Property 2**, **Property 3** and **Property 4**. Let α_k be the number of on-time member jobs which interrupt element job J_k resulting in non-zero switching cost. Based on the properties of S, all the possible values for α_k, $1 \leq k \leq 3q$, must satisfy (35) and (36) stated below. From **Property 2** that there must be exactly q member jobs, we can get that

$$\sum_{k=1}^{3q} \alpha_k = 3q. \tag{35}$$

From **Property 3** that all the element jobs must be on-time, we can get that $\sum_{k=1}^{i}(q+1+k\alpha_k) \leq \sum_{k=1}^{i}(q+1+k)$ for $i = 1, 2, \cdots, 3q$. In other words,

$$\sum_{k=1}^{i} k(\alpha_k - 1) \leq 0, \tag{36}$$

for $i = 1, 2, \cdots, 3q$.

To prove that $\alpha_1 = \alpha_2 = \cdots = \alpha_{3q} = 1$ is the unique solution satisfying (35) and (36), we let $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \cdots, \alpha_{3q})$ denote a solution set and introduce a scalar function $V(\boldsymbol{\alpha})$, where

$$V(\boldsymbol{\alpha}) = \sum_{k=1}^{3q} k(\alpha_k - 1). \tag{37}$$

Clearly, $\bar{\boldsymbol{\alpha}} = (1, 1, \cdots, 1)$, *i.e.* $\bar{\alpha}_1 = \bar{\alpha}_2 = \cdots = \bar{\alpha}_{3q} = 1$, is a solution set and $V(\bar{\boldsymbol{\alpha}}) = 0$. We will show that $\bar{\boldsymbol{\alpha}}$ by contradiction that $V(\boldsymbol{\alpha}) > V(\bar{\boldsymbol{\alpha}}) = 0$ for all $\boldsymbol{\alpha} \neq \bar{\boldsymbol{\alpha}}$, implying that all the possible sets of $3q$ integers, except $\bar{\alpha}$, violate (36). Hence, $\bar{\alpha}$ is the only solution.

Assume that there exists a solution set $\boldsymbol{\alpha}' = (\alpha'_1, \alpha'_2, \cdots, \alpha'_{3q})$ in which some values are not equal to 1. Let $t = \min_i\{i | \alpha'_i \neq 1\}$. That is to say, either $\alpha'_t \geq 2$ or $\alpha'_t = 0$. For the first situation that $\alpha'_t \geq 2$, J_t must be late. So, it violates **Property 4**.

Consider the second situation that $\alpha'_t = 0$. By (35), there must exist an element job J_i such that $i > t$ and $\alpha'_i \geq 2$. Let $s = \min_i\{i | i > t \text{ and } \alpha'_i \geq 2\}$. We can construct $\boldsymbol{\alpha}''$ as follows :

$$\alpha''_i = \begin{cases} 1 & \text{if } i = t, \\ \alpha'_s - 1 & \text{if } i = s, \\ \alpha'_i & \text{if } i \neq t, s. \end{cases} \tag{38}$$

Then, we can get that

$$V(\boldsymbol{\alpha}') - V(\boldsymbol{\alpha}'') = (s(\alpha'_s - 1) + t(\alpha'_t - 1)) - (s(\alpha''_s - 1) + t(\alpha''_t - 1)).$$

As $\alpha''_s = \alpha'_s - 1$, $\alpha'_t = 0$ and $\alpha''_t = 1$, $V(\boldsymbol{\alpha}') > V(\boldsymbol{\alpha}'')$.

If some values in $\boldsymbol{\alpha}''$ are not equal to 1, we can apply the same procedure stated as (38) to adjust the values in the set. Eventually, we will obtain a value set equal to $\bar{\boldsymbol{\alpha}}$. From the above analysis, the V function of each newly adjusted value set is strictly less than that of the original one. Hence, $V(\boldsymbol{\alpha}') > V(\boldsymbol{\alpha}'') > \cdots > V(\bar{\boldsymbol{\alpha}}) = 0$. It contradicts that there exists an $\boldsymbol{\alpha} \neq \bar{\boldsymbol{\alpha}}$ such that $V(\boldsymbol{\alpha}) \leq 0$. Therefore, $\bar{\alpha}_1 = \bar{\alpha}_2 = \cdots = \bar{\alpha}_{3q} = 1$ must be the unique solution satisfying (35) and (36).

As a result, each element job will be interrupted by exact one on-time member job such that the switching cost is non-zero. By observing the definition of the switching cost functions, we can conclude that the intersection of any two members corresponding to on-time member jobs in S must be empty. Therefore, **Property 5** holds.

By defining $C' \subseteq C$ as that $C' = \{C_j | J_{3q+j}$ is on-time in $S\}$. Together with **Property 4** and **Property 5**, we can conclude that C' is an exact cover of X. **Q.E.D.**

7.2. Symmetric switching costs

If the switching costs are symmetric, the problem is NP-hard, as stated in the following theorem.

Theorem 13 *Problem 5 with symmetric switching costs and the late jobs are discarded, i.e.* $1|mt, f_{ij} = f_{ji}, DLJ|\sum_i U_i$, *is NP-hard.*

Proof. The NP-hardness of the problem will be shown by reduction from the Independent Set problem. This problem is known to be NP-hard (Garey & Johnson 1979).

Problem 9 (Independent Set) *For a given graph* $G = (V, E)$ *and a positive integer* $K \leq |V|$, *Independent Set problem asks if* G *contains an independent set* V' *of size* K *or more such that no two vertices in* V' *are joined by an edge in* E.

Given an arbitrary instance of the Independent Set problem, we construct the corresponding instance of the problem $1|mt, f_{ij} = f_{ji}, DLJ|\sum U_i$ in the following. By letting $n = |V|$ and a set of jobs $J = \{J_1, J_2, \cdots, J_n\}$. The vertex $V_i \in V$ maps to $J_i \in J$. For jobs J_i and J_j,

$$p_i = 1 \tag{39}$$

$$g_i = 0 \tag{40}$$

$$d_i = K \tag{41}$$

$$f_{ij} = \begin{cases} 1 \text{ if } (V_i, V_j) \in E, \\ 0 \text{ otherwise.} \end{cases} \tag{42}$$

Moreover, the threshold of the number of late jobs y as $n - K$.

($\Rightarrow$) Let V' be the independent set and $|V'| \geq K$. A schedule S of the constructed instance can be obtained by scheduling K arbitrary jobs corresponding to vertices in V' as on-time in any order. Since V' is a independent set, the switching costs among these selected K jobs are all zero. Hence the total time required to process these K job is K, implying that all of them are on-time. As a result, the total number of late jobs in S is $n-K \leq y$.

($\Leftarrow$) Let S be a schedule of J such that $\sum_{i=1}^{n} U_i(S) \leq n-K$. The independent set V' can be formed as $V' = \{V_i | J_i \text{ is on-time in } S\}$. It is easy to verify that $|V'| \geq K$. If there exists a pair of V_i and V_j such that the edge $(V_i, V_j) \in E$, the total time required to process all the on-time jobs in S is at least $K+1$ which is greater the common due date. Hence, V' must be a independent set.

As the Independent Set problem is known to be NP-hard, the problem $1|mt, f_{ij} = f_{ji}, DLJ| \sum_i U_i$ must be NP-hard. **Q.E.D.**

Remark. In (Hall, Leung & Li 2014), they assumed that late jobs are not discarded and the switching costs are constant, i.e. $1|mt, f_{ij} = \xi| \sum_i U_i$. They further showed that the problem of minimizing the number of late jobs is NP-hard. If the interruption-time function of J_i is defined as $g_i(p'_i) = Dp'_i$, they showed that the problem can be solved by an algorithm in $\mathcal{O}(n \log(n))$ time (refer to Algorithm U and Theorem 4 in (Hall, Leung & Li 2014)). The idea of the algorithm is based on the Hodgson-Moore algorithm (Moore 1968). Assume that the due dates of $J_1, J_2, \cdots, J_n$ are in ascending order. Let S be the set of scheduled jobs. Initially, S is empty. For $i = 1, \cdots, n$, $S = S \| \{J_i\}$ if the completion time of J_i is earlier or equal to its due date. Otherwise, $S = S \| \{J_i\} \setminus \{J_h\}$, where J_h is a job in $S \| \{J_i\}$ with the longest processing time. As late jobs will also interrupt, it can be shown that the completion of J_a will not be changed if J_a is used to be scheduled before J_h. The completion of J_a will be earlier if it is used to be scheduled after J_h. So, in each step, at most one job has to be removed. However, this idea cannot be applied to our problem. As the late jobs will not interrupt the on-time jobs, the completion times of the on-time jobs will change whenever a new job is added. More than one jobs could be late. In this regard, the idea of Hodgson-Moore algorithm seems unlikely applicable. Thus, the complexity of the total number of late jobs problem, under the conditions that (1) the late jobs are discarded and (2) the interruption-time function is defined as proportional to the remaining processing time, is still unknown.

8. Total Weighted Number of Late Jobs

For the total weighted number of late jobs problem, it is clear that the problem is strongly NP-hard if the switching costs are asymmetric. While the problem $1|mt, f_{ij}, DLJ|\sum_i U_i$ is strongly NP-hard (from Theorem 12), the problem $1|mt, f_{ij}, DLJ|\sum_i w_i U_i$ must be strongly NP-hard.

8.1. Asymmetric switching costs

For asymmetric switching costs, the NP-hardness of the problem $1|mt, f_{ij}, DLJ|\sum_i w_i U_i$ is stated in the following theorem.

Theorem 14 *Problem 5 with asymmetric switching costs and the late jobs are discarded, i.e.* $1|mt, f_{ij}, DLJ|\sum_i w_i U_i$, *is strongly NP-hard.*

Proof. It is implied from Theorem 12 below. **Q.E.D.**

8.2. Symmetric switching costs

With symmetric switching costs, the problem is strongly NP-hard, as stated in the following theorem.

Theorem 15 *Problem 6 with symmetric switching costs and the late jobs are discarded, i.e.* $1|mt, f_{ij} = f_{ji}, DLJ|\sum_i w_i U_i$, *is strongly NP-hard.*

Proof. The proof is inspired by the proof in (Hall, Leung & Li 2014, Proof of Theorem 5) which is accomplished by reduction from Exact Cover by 3-Sets (X3C), see Problem 8. Similar to the proof for Theorem 12, we map each element in X to a unique integer i, where $i \in \{1, 2, \cdots, 3q\}$. Each $C_j \in C$ is denoted by $\{x_{j,1}, x_{j,2}, x_{j,3}\}$, where $x_{j,1} < x_{j,2} < x_{j,3}$.

Given an arbitrary instance of X3C, the corresponding instance of $1|mt, f_{ij} = f_{ji}, DLJ|\sum w_i U_i$ with no interruption is allowed from the late jobs can be constructed. First, a job set $J = \{J_1, J_2, \cdots, J_{3q}, J_{3q+1}, \cdots, J_{3q+r}\}$ is defined. For each job $J_i \in J$, its processing time, due date and weight are defined as follows :

$$p_i = \begin{cases} 1 & \text{if } i = 1, 2, 3, \cdots, 3q. \\ N + 3(q+1) & \text{if } i = 3q+1, 3q+2, \cdots, 3q+r, \end{cases} \tag{43}$$

$$d_i = \begin{cases} i(q+2) & \text{if } i = 1,2,3,\cdots,3q. \\ 3q(q+2)+qN & \text{if } i = 3q+1, 3q+2, \cdots, 3q+r, \end{cases} \tag{44}$$

$$w_i = \begin{cases} M & \text{if } i = 1,2,3,\cdots,3q. \\ M - x_{j,1} - x_{j,2} - x_{j,3} & \text{if } i = 3q+j \text{ and } C_j \in C. \end{cases} \tag{45}$$

In (43), (44) and (45), the values of N and M must fulfil the following inequalities.

$$N > (q+2), \text{ and } M > 9q^2. \tag{46}$$

The interruption-time function is defined as follows : For $i = 1,2,\cdots,3q$, $g_i(p') = 0$, and for each $C_j \in C$,

$$g_{3q+j}(p') = \begin{cases} 2 \text{ if } p' = N + 3(q+1) - x_{j,1} + 1, \\ 2 \text{ if } p' = N + 3(q+1) - x_{j,2}, \\ 2 \text{ if } p' = N + 3(q+1) - x_{j,3} - 1, \\ 1 \text{ otherwise.} \end{cases} \tag{47}$$

Moreover, we set all the switching costs to zeros. The threshold cost y is defined as

$$y = (r-q)M - \left(A - \frac{3q(3q+1)}{2}\right), \tag{48}$$

where $A = \sum_{C_j \in C}(x_{j,1} + x_{j,2} + x_{j,3})$.

For the sake of presentation, we call each of the jobs J_1, J_2, $\cdots$, J_{3q} the *element job* as it corresponds to an element in X and each of the jobs J_{3q+1}, J_{3q+2}, $\cdots$, J_{3q+r} the *member job* as it corresponds to a member in C.

($\Rightarrow$) Let $C' = \{C_{j_1}, \cdots, C_{j_q}\}$ be a subset of C such that $\bigcup_{k=1}^{q} C_{j_k} = X$ and $C_{j_m} \bigcap C_{j_n} = \phi$ for $m \neq n$. A schedule S with the total number of weighted jobs equal to $(r-q)M - (A - 3q(3q+1)/2)$ can be constructed by three steps.

Step 1: Schedule all the element jobs according to the earliest due date (EDD) rule.

Step 2: All the member jobs corresponding to the members in C' are scheduled after the element jobs in arbitrary order.

Step 3: The member jobs corresponding to the members not in C' are late jobs and discarded.

From the definition of the interruption-time functions, each on-time member job J_{3q+j} interrupts the element jobs $J_{x_{j,1}}$, $J_{x_{j,2}}$ and $J_{x_{j,3}}$ two units of time if $C_j = \{x_{j,1}, x_{j,2}, x_{j,3}\}$ is in C'. Since all the members in C' are mutually exclusive, each element job will need $1+(q-1)+2$ $(=q+2)$ time units, including the processing time of the job and the interruption-time caused by all the on-time member jobs, to complete the process. As the element jobs are scheduled according to EDD rule, the completion time of job J_i is $i(q+2)$. All element jobs complete their execution by their due dates. As the completion time of the last member job is equal to the total processing time, we get that

$$\begin{aligned} C_{max} &= 3q + \sum_{C_j \in C'} (N + 3(q+1)) \\ &= 3q(q+2) + qN. \end{aligned} \tag{49}$$

All the member jobs corresponding to the members in C' are on-time in S.

On the other hand, the total weighted number of late jobs of S is $\sum_{C_j \notin C'}(M - x_{j,1} - x_{j,2} - x_{j,3})$ and

$$\begin{aligned} \sum_{C_j \notin C'} (M - x_{j,1} - x_{j,2} - x_{j,3}) &= (r-q)M - \left(A - \sum_{C_j \in C'} (x_{j,1} + x_{j,2} + x_{j,3}) \right) \\ &= (r-q)M - \left(A - \frac{3q(3q+1)}{2} \right). \end{aligned} \tag{50}$$

The last equation of (50) is due to the facts that C' is a exact cover of X and each element in X is represented by a unique integer selected from $\{1, 2, \cdots, 3q\}$. By (48) and (50), we conclude that $\sum_{C_j \notin C'}(M - x_{j,1} - x_{j,2} - x_{j,3}) \leq y$.

($\Leftarrow$) Let S be a schedule for the constructed job set J with total weighted number of late jobs less than or equal to $y = (r-q)M - (A - 3q(3q+1)/2)$. Before we proceed to prove the only-if part, we need to show that the schedule S must satisfy the following properties.

Property 1: Removing inserted idle time between jobs in S will not increase the total weighted number of late jobs.

Property 2: Rearranging all the on-time jobs in S according to the EDD rule will not increase the total weighted number of late jobs.

Property 3: There are exactly q on-time member jobs.

Property 4: All the element jobs must be on-time.

Property 5: The intersection of any two members corresponding to on-time member jobs in S must be empty.

Since all jobs in the constructed job set are available at time zero, left-shifting an on-time job to eliminate idle time does not make it late. So, **Property 1** holds.

Consider an adjacency pair of on-time jobs J_{π_i} and $J_{\pi_{i+1}}$ in S such that $d_{\pi_i} > d_{\pi_{i+1}}$. As the switching cost is symmetric and the condition (2) is satisfied, swapping J_{π_i} and $J_{\pi_{i+1}}$ will not increase the time span for executing them. Both jobs will still be on-time after swapping. Repeated swapping every pair of adjacent jobs which violates the EDD rule will generate a schedule in which all the are scheduled according to the EDD rule and all of them on-time. So, **Property 2** holds.

Property 1 and **Property 2** imply that the schedule S must have no idle time between jobs and all the on-time jobs are scheduled in non-decreasing due dates. In other words, the on-time element jobs must be scheduled before the on-time member jobs.

Property 3 will be proved by contradiction. We assume that $q + v$, where $v \geq 1$, member jobs are scheduled on-time in S. The completion time of the last job in these $q + v$ member jobs must be larger than $(q + v)N + 3(q + v)(q + 1)$. It is thus obvious that at least one member job cannot be on-time. Therefore, it is not possible to have more than q on-time member jobs in S. On the contrary, we assume that only $q - v$, where $v \geq 1$, member jobs are on-time in S. Let $J_{L,S}$ be the set of member jobs that are late jobs, $J_{O,S}$ be the set of member jobs that are on-time. We get that

$$\begin{aligned}\sum_{J_{3q+j} \in J_{L,S}} (M - x_{j,1} - x_{j,2} - x_{j,3}) &= (r - (q - v))M - \sum_{J_{3q+j} \in J_{L,S}} (x_{j,1} + x_{j,2} + x_{j,3}) \\ &\geq (r - q)M + vM - \left(A - \sum_{J_{3q+j} \in J_{O,S}} (x_{j,1} + x_{j,2} + x_{j,3}) \right) \\ &> (r - q)M. \end{aligned} \tag{51}$$

The last step is due to (46), $M > 3q^2$. As the total weighted number of late jobs in S must be larger than or equal to $\sum_{J_{3q+j} \in J_{L,S}} (M - x_{j,1} - x_{j,2} - x_{j,3})$, it is clear from (51) that the total weighted number of late jobs must be larger than $(r-q)M - (A - 3q(3q+1)/2)$. Therefore, it is not possible to have few than q on-time member jobs in S. We can conclude that the number of on-time member jobs in S must be exactly q. So, **Property 3** holds.

Property 4 is proved by contradiction. We assume that v, where $v \geq 1$, element jobs in S are late. From **Property 3**, the set of late jobs must include v element jobs and $(r-q)$ member jobs. The total weighted number of late jobs is $vM + \sum_{J_{3q+j} \in J_{L,S}} (M - x_{j,1} - x_{j,2} - x_{j,3})$ and

$$\begin{aligned} vM + \sum_{J_{3q+j} \in J_{L,S}} (M - x_{j,1} - x_{j,2} - x_{j,3}) &\geq (r-q)M + vM - \left(A - \sum_{J_{3q+j} \in J_{O,S}} (x_{j,1} + x_{j,2} + x_{j,3}) \right) \\ &> (r-q)M. \end{aligned} \tag{52}$$

The last step is due to (46), $M > 3q^2$. Clearly, it contradicts to the fact that the total number of weighted late jobs in S is not greater than y. **Property 4** holds.

Property 5 is proved by using **Property 3**, **Property 4** and (48). Let α_k be the number of on-time member jobs which interrupt element job J_k for 2 time units. Based on the properties of S, all the possible values for α_k, $1 \leq k \leq 3q$, must satisfy (53), (54) and (55) stated below. From **Property 3**, we have

$$\sum_{k=1}^{3q} \alpha_k = 3q. \tag{53}$$

From **Property 4**, we have $\sum_{k=1}^{i} (q+1+\alpha_k) \leq i(q+2)$ for $i = 1, 2, \cdots, 3q$. In other words,

$$\sum_{k=1}^{i} \alpha_k \leq i, \tag{54}$$

for $i = 1, 2, \cdots, 3q$. From **Property 3** and **4**, only $(r-q)$ member jobs are late in S. The total weighted number of late jobs in S is given by

$$\begin{aligned} \sum_{J_{3q+j} \in J_{L,S}} (M - x_{j,1} - x_{j,2} - x_{j,3}) &= (r-q)M - \left(A - \sum_{J_{3q+j} \in J_{O,S}} (x_{j,1} + x_{j,2} + x_{j,3}) \right) \\ &\leq (r-q)M - \left(A - \frac{3q(3q+1)}{2} \right). \end{aligned}$$

Note that $\sum_{J_{3q+j}\in J_{O,S}}(x_{j,1}+x_{j,2}+x_{j,3})=\sum_{k=1}^{3q}k\alpha_k$. We can get that $\sum_{k=1}^{3q}k\alpha_k\leq 3q(3q+1)/2$. Based on the fact that $\sum_{k=1}^{3q}k=3q(3q+1)/2$, we thus have the following inequality related to $\alpha_i, 1\leq i\leq 3q$.

$$\sum_{k=1}^{3q}k(\alpha_k-1)\leq 0. \tag{55}$$

Let $\boldsymbol{\alpha}=(\alpha_1,\alpha_2,\cdots,\alpha_{3q})$ denote a solution set satisfying (53), (54) and (55). To facilitate our proof, we define $W(\boldsymbol{\alpha})$ as the left hand side of the inequality (55) :

$$W(\boldsymbol{\alpha})=\sum_{k=1}^{3q}k(\alpha_k-1). \tag{56}$$

Clearly, $\bar{\boldsymbol{\alpha}}=(1,1,\cdots,1)$, *i.e.* $\bar{\alpha}_1=\bar{\alpha}_2=\cdots=\bar{\alpha}_{3q}=1$, is a solution set and $W(\bar{\boldsymbol{\alpha}})=0$. We will show that $\bar{\boldsymbol{\alpha}}$ by contradiction that $W(\boldsymbol{\alpha})>W(\bar{\boldsymbol{\alpha}})=0$ for all $\boldsymbol{\alpha}\neq\bar{\boldsymbol{\alpha}}$. Assume that there exists a solution set $\boldsymbol{\alpha}'=(\alpha_1',\alpha_2',\cdots,\alpha_{3q}')$ in which some values are not equal to 1. Let $t=\min_i\{i|\alpha_i'\neq 1\}$. That is to say, either $\alpha_t'\geq 2$ or $\alpha_t'=0$. If $\alpha_t'\geq 2$, J_t must be late. It violates **Property 4**.

Consider the second situation that $\alpha_t'=0$. By (53), there must exist an element job J_i such that $i>t$ and $\alpha_i'\geq 2$. Let $s=\min_i\{i|i>t \text{ and } \alpha_i'\geq 2\}$. We can construct $\boldsymbol{\alpha}''$ as follows :

$$\alpha_i''=\begin{cases}1 & \text{if } i=t,\\ \alpha_s'-1 & \text{if } i=s,\\ \alpha_i' & \text{if } i\neq t,s.\end{cases} \tag{57}$$

Then, we can get that

$$W(\boldsymbol{\alpha}')-W(\boldsymbol{\alpha}'')=\left(s(\alpha_s'-1)+t(\alpha_t'-1)\right)-\left(s(\alpha_s''-1)+t(\alpha_t''-1)\right).$$

As $\alpha_s''=\alpha_s'-1$, $\alpha_t'=0$ and $\alpha_t''=1$, $W(\boldsymbol{\alpha}')>W(\boldsymbol{\alpha}'')$.

If some values in $\boldsymbol{\alpha}''$ are not equal to 1, we can apply the same procedure stated as (57) to adjust the values in the set. Eventually, we will obtain a value set equal to $\bar{\boldsymbol{\alpha}}$. From the above analysis, the W function of each newly adjusted value set is strictly less than that of the original one. Hence, $W(\boldsymbol{\alpha}')>W(\boldsymbol{\alpha}'')>\cdots>W(\bar{\boldsymbol{\alpha}})=0$. It contradicts that there exists an $\boldsymbol{\alpha}\neq\bar{\boldsymbol{\alpha}}$ such that $W(\boldsymbol{\alpha})\leq 0$. Therefore, $\bar{\alpha}_1=\bar{\alpha}_2=\cdots=\bar{\alpha}_{3q}=1$ must be the unique solution satisfying (53), (54) and (55).

So, we can conclude that each element job is interrupted by exactly one on-time member job for 2 time units. Moreover, there does not exist any two on-time member jobs interrupting the same element job for 2 time units. Therefore, **Property 5** holds.

By defining $C' \subseteq C$ as that $C' = \{C_j | J_{3q+j}$ is on-time in $S\}$. Together with **Property 4** and **Property 5**, we can conclude that C' is an exact cover of X. **Q.E.D.**

9. Complexity Hierarchy

In this section, the complexities of the scheduling problems for human multitasking model with switching cost will be summarized. The complexity hierarchy for the problems regarding asymmetric switching costs will be presented in the first subsection. For completeness, the complexities of the total tardiness and the total weighted tardiness problems are included. The complexity hierarchy for the problems regarding symmetric switching costs will be presented in the second subsection.

The core hierarchy structure is based on Lageweg, Lenstra, Lawler & Rinnooy Kan (1982). The complexity hierarchy of the problems $1|mt|C_{max}$, $1|mt|T_{max}$ and $1|mt|L_{max}$ is based on the same reasoning for the case of asymmetric switching costs. The reduction of $1|mt|C_{max}$ to $1|mt|T_{max}$ is based on the setting $d_i = 0$ and $w_i = 1$ for all $i = 1, \cdots, n$. The reduction of $1|mt|T_{max}$ to $1|mt|L_{max}$ is based on the fact that the instance for the decision if $L_{max} = Z$, for $Z > 0$, is the same as the instance for the decision if $T_{max} = Z$. Moreover, the instance for the decision if $L_{max} \leq 0$ is the same as the instance for the decision if $T_{max} = 0$.

9.1. Asymmetric switching costs

Figure 4 shows the complexity hierarchy of the scheduling problems regarding human multitasking behavior with asymmetric switching costs. For asymmetric switching costs, we have proved in Theorem 1 that the problems $1|mt, f_{ij}|C_{max}$ is NP-hard, in Theorem 4 that problem $1|mt, f_{ij}| \sum_i c_i$ is strongly NP-hard and in Theorem 12 that $1|mt, f_{ij}, DLJ| \sum_i U_i$ is strongly NP-hard. As a result, the problems $1|mt, f_{ij}|L_{max}$ and $1|mt, f_{ij}|T_{max}$ must be NP-hard as stated in Theorem 9 and

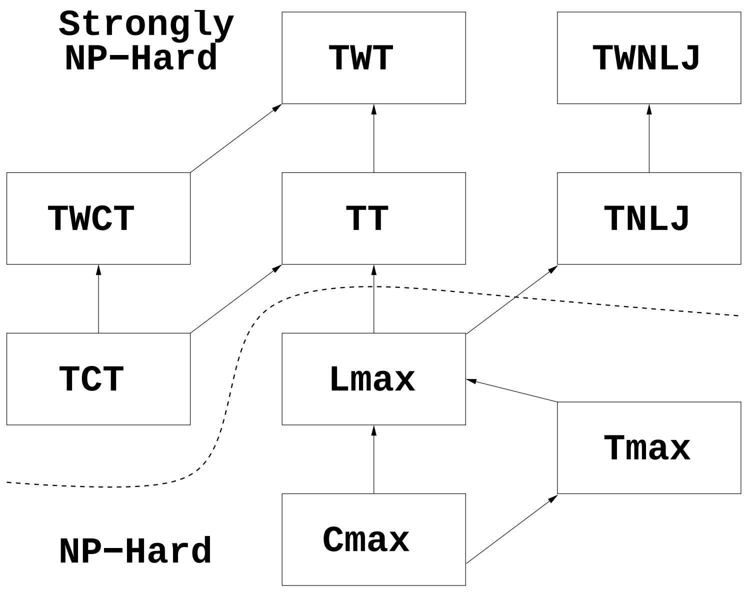

Figure 4 The complexity hierarchy of the scheduling problems regarding human multitasking behavior and asymmetric switching costs.

Theorem 6. The problem $1|mt, f_{ij}|\sum_i T_i$, $1|mt, f_{ij}|\sum_i w_i T_i$ must be strongly NP-hard. Imply from the strongly NP-hardness of $1|mt, f_{ij} = f_{ji}, DLJ|\sum_i w_i U_i$, the problem $1|mt, f_{ij}, DLJ|\sum_i w_i U_i$ is strong NP-hard.

9.2. Symmetric switching costs

Figure 5 shows the complexity hierarchy of the scheduling problems regarding human multitasking behavior with symmetric switching costs. The complexities of the problems $1|mt, f_{ij} = f_{ji}|C_{max}$, $1|mt, f_{ij} = f_{ji}|\sum_i w_i c_i$, $1|mt, f_{ij} = f_{ji}|T_{max}$ and $1|mt, f_{ij} = f_{ji}|L_{max}$ are based on the Theorem 2, Theorem 5, Theorem 7 and Theorem 10. The NP-hardness of the problem $1|mt, f_{ij} = f_{ji}, DLJ|\sum_i U_i$ and the strongly NP-hardness of the problem $1|mt, f_{ij} = f_{ji}, DLJ|\sum_i w_i U_i$ are based on Theorem 13 and Theorem 15. The NP-hardness of the problem $1|mt, f_{ij} = f_{ji}|\sum_i T_i$ is implied from the results in Du & Leung (1990). The strongly NP-hardness of the problem $1|mt, f_{ij} = f_{ji}|\sum_i w_i T_i$ is implied from the results in Lawler (1977) and Lenstra, Rinnooy Kan & Brucker (1977).

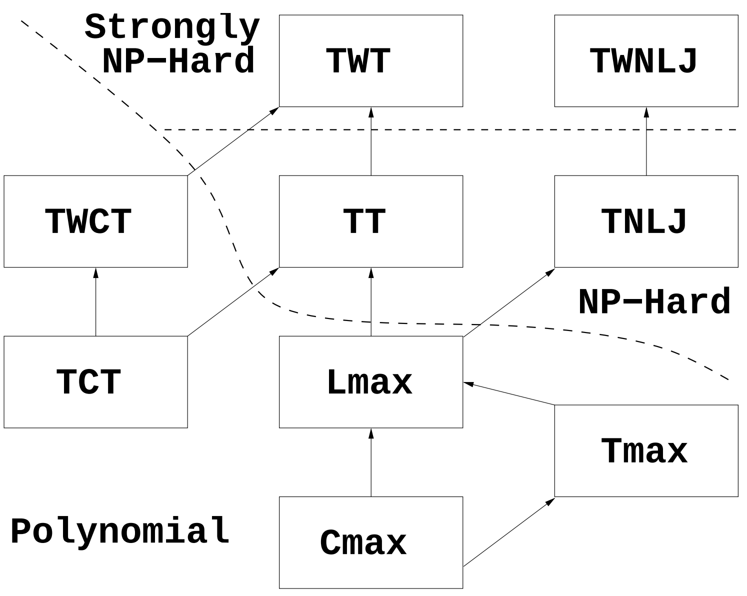

Figure 5 The complexity hierarchy of the scheduling problems regarding human multitasking behavior and symmetric switching costs.

10. Conclusions

Six scheduling problems regarding human multitasking behavior have been investigated in this paper. For the late job problems, we assume that human workers will discard all the late jobs. For each scheduling problem, we have analyzed the complexities of the problem under both asymmetric and symmetric switching costs environments.

For asymmetric switching costs, we have shown that all problems are either NP-hard or strongly NP-hard. By reduction from Feedback Arc Set problem, we have shown that the problem $1|mt, f_{ij}|C_{max}$ is NP-hard. Thus, the problems $1|mt, f_{ij}|T_{max}$ and $1|mt, f_{ij}|L_{max}$ are NP-hard. By reduction from $1|prec|\sum_i c_i$, we have shown that the problem $1|mt, f_{ij}|\sum_i c_i$ is strongly NP-hard. Thus, the problems $1|mt, f_{ij}|\sum_i w_i c_i$, $1|mt, f_{ij}|\sum_i T_i$ and $1|mt, f_{ij}|\sum_i w_i T_i$ must also be strong NP-hard. By reduction from Exact Cover by 3-Sets problem, we have further shown that $1|mt, f_{ij}, DLJ|\sum_i U_i$ is strongly NP-hard. Thus, the problem $1|mt, f_{ij}, DLJ|\sum_i w_i U_i$ is strongly NP-hard.

For symmetric switching costs, we have shown that the problem $1|mt, f_{ij} = f_{ji}|C_{max}$ is an easy problem as $C_{max} = \sum_{i=1}^n p_i + \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n f_{ij}$ for any feasible schedule of a given job set. With this important result, three polynomial-time algorithms have been developed for solving the problems $1|mt, f_{ij} = f_{ji}|\sum_i w_i c_i$, $1|mt, f_{ij} = f_{ji}|T_{max}$ and $1|mt, f_{ij} = f_{ji}|L_{max}$. For solving the problem $1|mt, f_{ij} = f_{ji}|\sum_i w_i c_i$, Algorithm WCT has been developed which is a $\mathcal{O}(n^2)$-time algorithm. For solving the problems $1|mt, f_{ij} = f_{ji}|T_{max}$ and the $1|mt, f_{ij} = f_{ji}|L_{max}$, Algorithm WT and Algorithm WL have been developed. Their complexities are $\mathcal{O}(n^2)$ for jobs with different weights. We have further shown that these two problems can be solved by earliest due date (EDD) rule if all the job weights are equal. By reduction from the Independent Set problem, we have shown that $1|mt, f_{ij} = f_{ji}, DLJ|\sum_i U_i$ is NP-hard. Then, by reduction from Exact Cover by 3-Sets (X3C) problem, we have shown that the problem $1|mt, f_{ij} = f_{ji}, DLJ|\sum_i w_i U_i$ is strongly NP-hard. For the problems $1|mt, f_{ij} = f_{ji}|\sum_i T_i$ and $1|mt, f_{ij} = f_{ji}|\sum_i w_i T_i$, based on the results Lawler (1977), Lenstra, Rinnooy Kan & Brucker (1977) and Du & Leung (1990), they are respectively NP-hard and strongly NP-hard.

Finally, together with the complexity hierarchies presented by Lageweg, Lenstra, Lawler & Rinnooy Kan (1982), the complexity hierarchy of the scheduling problems regrading human multitasking behavior are complied and shown in Figure 4 and Figure 5.

While we have shown the complexities of the problems regarding human multitasking behavior and developed algorithms for those problems which are polynomial-time solvable, we have not developed any pseudo polynomial-time or approximation algorithms for those problems which are NP-hard or strongly NP-hard. In particular, the problems $1|mt, f_{ij} = f_{ji}, DLJ|\sum_i U_i$ and $1|mt, f_{ij} = f_{ji}, DLJ|\sum_i w_i U_i$ would be two our future focuses. As mentioned earlier in the paper, another important future work is to investigate the productivity of a worker, as compared with sequential processing, if multitasking is unavoidable.

References

American Psychological Association 2006. Multitasking: Switching costs. Information available online at `https://www.apa.org/research/action/multitask.aspx`.

Aral, S., E. Brynjolfsson & M. Van Alstyne. 2007. Information, technology and information worker productivity: Task level evidence. National Bureau of Economic Research (NBER) Working Paper 13172.

Brucker, P., 2007. *Scheduling Algorithms*, 5th Edition, Springer-Verlag Berlin Heidelberg.

Corts, K.S. 2007. Teams versus individual accountability: Solving multitask problems through job design. *RAND Journal of Economics*, Vol.38(2), 467-479.

Coviello, D., A. Ichino, and N. Persico. 2010. Dont spread yourself too thin: the impact of task juggling on workers speed of job completion. National Bureau of Economic Research (NBER) Working paper 16502.

Coviello, D., A. Ichino, & N. Persico. 2014. Time allocation and task juggling. *The American Economic Review*, 104(2), 609-623.

Deshmukh, S. D., & D.S. Chikte. 1977. Stochastic evolution and control of an economic activity. *Journal of Economic Theory*, 15(1), 112-122.

Du, J., J.Y.T. Leung. 1990. Minimizing total tardiness in one machine is NP-hard. *Mathematics of Operations Research* **15** 483-495.

Foerde, K., B.J. Knowlton, & R.A. Poldrack. 2006. Modulation of competing memory systems by distraction. *Proceedings of the National Academy of Sciences*, 103(31), 11778-11783.

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

Gillie, T., & D. Broadbent. 1989. What makes interruptions disruptive? A study of length, similarity, and complexity. *Psychological Research*, 50(4), 243-250.

Hall, N.G., J.Y.T. Leung, C.L. Li. 2014. The effects of multitasking on operations scheduling. In submission.

Hallowell, E.M. 2005. Overloaded circuits: Why smart people underperform. *Harvard Business Review*, 83(1), 55-62.

Holmstrom, B. & P. Milgrom. 1991. Multitask principal-agent analyses: Incentive contracts, asset ownership, and job design. *Journal of Law, Economics, & Organization*, Vol.7, 24-52.

Itoh, H. 1994. Job design, delegation and cooperation: A principal-agent analysis. *European Economic Review*, 38(3), 691-700.

Jackson, J.R. 1955. *Scheduling a production line to minimize maximum tardiness.* Research Report 43, Management Science Research Report, UCLA.

Jett, Q.R., J.M. George. 2003. Work interrupted: A closer look at the role of interruptions in organizational life. *Academy of Management Review* **28** 494-507.

Karp, K. M. 1972. Reducibility among combinatorial problems. R.E. Miller, J.W. Thatcher, eds. *Complexity of Computer Computations*. Plenum, New York, 85-103.

Lageweg, B.J., J.K. Lenstra, E.L. Lawler, A.H.G. Rinnooy Kan. 1982. Computer-aided complexity classification of combinatorial problems. *Communications of ACM*, **25** 817-822.

Lawler, E.T. 1973. Optimal sequencing of a single machine subject to precedence constraints. *Management Science* **19** 544-546.

Lawler, E.L. 1977. A 'pseudopolynomial' time algorithm for sequencing jobs to minimize total tardiness. *Annals of Discrete Mathematics* **1** 331-342.

Laeler, E.L. 1978. Sequencing jobs to minimize total weighted completion time subject to precedence constraints. *Annals of Discrete Mathematics*, **2** 75-90.

Lee, F.J., N.A. Taatgen NA. 2002. Multi-tasking as skill acquisition. *Proceedings of the twenty-fourth annual conference of the cognitive science society* pp. 572-577. Mahwah, NJ: Erlbaum. Fairfax, VA: August, 2002.

Lenstra, J.K., A.H.G. Rinnooy Kan, P. Bruker 1977. Complexity of machine scheduling problems. *Annals of Discrete Mathematics* **1** 343-362.

Lenstra, J.K., A.H.G. Rinnooy Kan 1978. Complexity of scheduling under precedence constraints. *Operations Research*, **26** (1) 22-35.

Loukopoulos, L.D., R.K. Dismukes, I. Barshi 2009. *The Multitasking Myth: Handling complexity in real-world operations*. Ashgate Publishing Limited, England.

Mark, G., V.M. Gonzalez, & J. Harris. 2005. No task left behind?: examining the nature of fragmented work. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 321-330). ACM.

Matsushima, H., K. Miyazaki, & N. Yagi. 2010. Role of linking mechanisms in multitask agency with hidden information. *Journal of Economic Theory*, 145(6), 2241-2259.

McNaughton, R. 1959. Scheduling with deadlines and loss functions. *Management Science*, **6** 1-12.

Meuter, R.F., & A. Allport. 1999. Bilingual language switching in naming: Asymmetrical costs of language selection. *Journal of Memory and Language*, 40(1), 25-40.

Mintzberg, H. 1973. *The Nature of Managerial Work*. Harper & Row, New York.

Monsell, S. 2003. Task switching. *Trends in cognitive sciences*, 7(3), 134-140.

Moore JM (1968). An n job, one machine sequencing algorithm for minimizing the number of late jobs. *Management Science*, **14** 102-109.

Ozmutlu, S., H.C. Ozmutlu, A. Spink. 2003. Multitasking web searching: implications for design. In ASIST03: Annual meeting of the American society for information science and technology, October 18-22, 2003. Long Beach, CA.

Pinedo, M. 1995. *Scheduling Theory, Algorithms, and Systems*, Prentice Hall, Englewood Cliffs, NJ.

Prasad, S. 2009. Task assignments and incentives: generalists versus specialists. *RAND Journal of Economics*, 40(2), 380-403.

Radner, R., & M. Rothschild. 1975. On the allocation of effort. *Journal of Economic Theory*, 10(3), 358-376.

Rogers, R. D., & S. Monsell. 1995. Costs of a predictible switch between simple cognitive tasks. *Journal of experimental psychology: General*, 124(2), 207.

Rosen, C. 2008. The myth of multitasking. *The New Atlantis*, 20(Spring), 105-110.

Rothschild, M. 1974. Further notes on the allocation of effort. Princeton University, Econometric Research Program No. 171.

Rubinstein, J. S., D.E. Meyer, & J.E. Evans. 2001. Executive control of cognitive processes in task switching. *Journal of Experimental Psychology: Human Perception and Performance*, 27(4), 763.

Salvucci, D.D., N.A. Taatgen, J.P. Borst. 2009. Toward a unified theory of the multitasking continuum: From concurrent performance to task switching, interruption, and resumption. *In Human Factors in Computing Systems: CHI 2009 Conference Proceedings* 1819-1828. New York: ACM Press.

Sanbonmatsu, D. M., D.L. Strayer, N. Medeiros-Ward & J.M. Watson. (2013). Who multi-tasks and why? Multi-tasking ability, perceived multi-tasking ability, impulsivity, and sensation seeking. *PloS One*, 8(1), e54402.

Seshadri, S., Z. Shapira. 2001. Managerial allocation of time and effort: the effects of interruptions. *Management Science* **47** 647-662.

Spink, A., H.C. Ozmutlu, S. Ozmutlu. 2002. Multitasking information seeking and searching processes. *Journal of the American Society for Information Science and Technology* **53**(8) 639-652.

Spink, A. 2004. Multitasking information behavior and information task switching: An exploratory study. *Journal of Documentation* **60**(3) 336-345.

Spink, A., M. Park, B.J. Jansen, J. Pedersen. 2006. Multitasking during web search sessions. *Information Processing and Management* **42**(1) 264-275.

Taatgen, N.A., F.J. Lee. 2003. Production compilation: A simple mechanism to model complex skill acquisition. *Human Factors: The Journal of the Human Factors and Ergonomics Society* **45**(1) 61-67.

Trafton, J.G., E.M. Altmann, D.P. Brock, F.E. Mintz. 2003. Preparing to resume an interrupted task: Effects of prospective goal encoding and retrospective rehearsal. *International Journal of Human-Computer Studies* **58** 583-603.

Watson, J. M., & D.L. Strayer. 2010. Supertaskers: Profiles in extraordinary multitasking ability. *Psychonomic Bulletin & Review*, 17(4), 479-485.

Zukerman, M. 2014. *Introduction to Queueing Theory and Stochastic Teletraffic Models.* Information available online at `www.ee.cityu.edu.hk/~zukerman/classnotes.pdf` (last visited 2014/6/21).

Appendix A: Results in Hall, Leung & Li (2014)

Here, we summarize the results obtained by Hall, Leung & Li (2014). Supposing the switching costs are constant, the problems of minimizing the total weighted completion time and the maximum lateness can be solved in $\mathcal{O}(n^2)$ and $\mathcal{O}(n\log(n))$ time.

Theorem 16 (Theorem 1 in (Hall, Leung & Li 2014)) *The problem $1|mt, f_{ij} = \xi|\sum_i w_i c_i$ is able to be solved in $\mathcal{O}(n^2)$ time.*

Theorem 17 (Theorem 2 in (Hall, Leung & Li 2014)) *The problem $1|mt, f_{ij} = \xi, w_i = 1|L_{max}$ is able to be solved in $\mathcal{O}(n\log(n))$ time.*

In regard to the late job problems, they assume that the switching costs are constant and the late jobs are scheduled arbitrarily after the on-time jobs.

Theorem 18 (Theorem 3 in (Hall, Leung & Li 2014)) *If the late jobs are scheduled arbitrarily after all the on-time jobs have completed, the problem $1|mt, f_{ij} = \xi|\sum_i U_i$ is NP-hard.*

Theorem 19 (Theorem 5 in (Hall, Leung & Li 2014)) *If the late jobs are scheduled arbitrarily after all the on-time jobs have completed, the problem* $1|mt, f_{ij} = \xi|\sum_i w_iU_i$ *is strongly NP-hard.*

For the special case that $g_i(p'_i) = Dp'_i$ for $0 < D < 1$, a polynomial-time algorithm has been developed. Hence, the total number of late jobs problem is polynomial-time solvable.

Theorem 20 (Theorem 4 in (Hall, Leung & Li 2014)) *If* $g_i(p'_i) = Dp'_i$ *for* $0 < D < 1$ *and the late jobs are scheduled arbitrarily after all the on-time jobs have completed, the problem* $1|mt, f_{ij} = \xi|\sum_i U_i$ *is able to be solved in* $\mathcal{O}(n\log(n))$ *time.*

Likewise, for the total weighted late job problem, two pseudo polynomial-time algorithms have been developed. Let $W = \sum_i^n w_i$ and $P = \sum_i^n p_i$.

Theorem 21 (Theorem 6 in (Hall, Leung & Li 2014)) *If* $g_i(p'_i) = Dp'_i$ *for* $0 < D < 1$ *and the late jobs are scheduled arbitrarily after all the on-time jobs have completed, the problem* $1|mt, f_{ij} = \xi|\sum_i w_iU_i$ *is able to be solved in* $\mathcal{O}(n^2 \min\{W, P\})$ *time.*