Computers & Operations Research 39 (2012) 1206–1212

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

Resource-constrained flowshop scheduling with separate resource recycling operations

T.C.E. Cheng [a,1], B.M.T. Lin [b,*], H.L. Huang [b]

[a] Department of Logistics and Maritime Studies, The Hong Kong Polytechnic University, Kowloon, Hong Kong
[b] Institute of Information Management, Department of Information and Finance Management, National Chiao Tung University, Hsinchu 300, Taiwan

ARTICLE INFO

Available online 7 August 2010

Keywords:
Relocation problem
Resource constraint
Flowshop scheduling
Recycling operations
Complexity

ABSTRACT

This paper considers the relocation problem arising from public re-development projects cast as a two-machine flowshop scheduling problem. In such a project, some buildings need to be torn down and re-constructed. The two processes of tearing down and re-constructing each building are often viewed as a single operation. However, under certain circumstances, the re-construction process, i.e., the resource recycling process, can be viewed as a separate operation. In this paper we regard these two processes as separate on the assumption that they are handled by different working crews. We formulate the problem as a resource-constrained two-machine flowshop scheduling problem with the objective of finding a feasible re-development sequence that minimizes the makespan. We provide problem formulations, discuss the complexity results, and present polynomial algorithms for various special cases of the problem.

1. Introduction

Scheduling is a decision-making process concerning the allocation of limited resources to activities over time so as to optimize one or more objective functions. The most commonly considered resources are machines that are deployed to perform (process) the activities (tasks or jobs). When resource-constrained scheduling is considered, extra resources such as money, personnel, energy, etc., are required for the machines to process their assigned tasks. Resource constraints have been considered in scheduling problems [1–3]. A sequence of the tasks is called feasible if all constraints, especially the resource constraints, are satisfied and all operations are completed. In this paper we consider a variant of the relocation problem that can be converted into a two-machine flowshop scheduling problem with generalized resource constraints.

The relocation problem was first proposed and formulated by Kaplan and his colleagues [4–7] for a public house re-development project in Boston [8]. It was a successful operations research application in the public sector. In the basic setting of the relocation problem, there is a set of buildings to be re-constructed. Each building is specified by two capacity-related parameters: the number of current tenants and the number of tenants that can be accommodated after re-development. When a building is under re-development, the evacuated tenants must be temporarily housed. The tenants are not required to reside at the same site after re-development. Given an initial budget of temporary housing, the authority has to determine a re-development sequence of the buildings such that all tenants can be successfully housed during the re-development process.

In the scheduling literature resources can be classified by type and category. For the relocation problem, all jobs need a single type of resource that is discrete and non-renewable. The original capacity of a building specifies the amount of the resource required to start a job (i.e., temporary housing to accommodate the evacuated tenants) and the new capacity after re-development is the amount of the resource returned by the job upon its completion. The resource constraints involved in the relocation problem differ from the commonly adopted resource constraints in that the amount of the resource a job returns upon its completion is not necessarily equal to the amount of the resource the job acquired for its processing.

The basic relocation problem is concerned with the existence of a feasible re-development sequence. The optimization counterpart of the problem is to determine the minimum initial budget that guarantees the existence of a feasible sequence. Kaplan and Amir [6] showed that the optimization problem is mathematically equivalent to the two-machine flowshop scheduling problem to minimize the makespan [9]. The financial planning problem [10] can be regarded as a special case of the relocation problem. More applications and discussion could be found in Amir and Kaplan [5] and Cheng and Lin [11]. Kononov and Lin [12] investigated the relocation problem on identical parallel machines. They settled

* Corresponding author. Tel.: +886 3513 1472; fax: +886 3572 3792.
E-mail addresses: LGTcheng@polyu.edu.hk (T.C.E. Cheng), bmtlin@mail.nctu.edu.tw, bmtlin@iim.nctu.edu.tw (B.M.T. Lin).
[1] Tel: +852 2766 5215; fax: +852 2364 5245.

the computational complexity of the problem and proposed approximation algorithms with analyses of their performance ratios. Lin and Liu [13] incorporated generalized due dates into the relocation problem to maximize the total rewards. They proposed a branch-and-bound algorithm equipped with two dominance rules and two lower bounds. Kononov and Lin [14] proved the strong NP-hardness of the problem to minimize the total weighted completion time and introduced an equivalence property for the unit-weighted case and the unit processing time case. Sevastyanov et al. [15] considered the relocation problem to minimize the makespan subject to release dates. In addition to the complexity analyses of several problem settings, they proposed a multi-parametric dynamic programming algorithm to deal with the case with a fixed number of distinct release dates. The running time of the dynamic program is pseudo-polynomial, which is tight because the case with two distinct due dates is NP-hard.

The first study on the relocation problem with the resource recycling issue incorporated is due to Lin and Huang [16]. In previous studies of resource-constrained scheduling, once a job is completed, the resource it returns is immediately available, i.e., resource recycling is assumed to be included in the job processing time. It is very similar to the situations where we assume that the setup time and removal time of a job are embedded in the processing time of the job. Under certain circumstances, the resource recycling process of a job can be viewed as a separate operation that can be performed concurrently with the normal processing of the other jobs. For the relocation problem, the re-development of a building consists of two stages, namely demolishing and re-construction of the building. For the basic relocation problem, the two stages are considered as an aggregate job. In this paper we treat the resource recycling process as a separate operation that has to be processed by a dedicated machine. Although the model considered in this paper stems from the relocation problem, it is applicable to situations where there are separate resource recycling operations.

Given a certain amount of the resource, our problem is to determine a feasible schedule that minimizes the makespan. We denote the studied problem by $F2|rp|C_{\max}$ using the three-field notation for scheduling problems [17]. The first field indicates that the machine configuration is a two-machine flowshop, where the first machine is for processing the jobs (demolishing the buildings) and the second machine is dedicated to the resource recycling operations (erecting new buildings). The second field highlights that the job characteristics are due to the context of the relocation problem. Lin and Huang [16] first proposed this problem setting and presented NP-hardness proofs, a branch-and-bound algorithm, and an approximation algorithm. This paper continues this line of study by focusing on the theoretical nature of the problem. One unique nature of the problem is due to the existence of two flowshops that interact in a kind of orthogonal manner. We will elaborate on this nature in the next section.

This paper is organized as follows: We present the problem formulation and introduce several preliminary observations in Section 2. While most previous studies assumed permutation schedules, we discuss the issue of non-permutation schedules in Section 3. We examine in Section 4 the complexity status of several special cases and present a complexity hierarchy of the problem. Finally, we conclude the paper and suggest some topics for future research in Section 5.

2. Problem formulation and preliminary properties

In this section we first introduce the notation that will be used throughout the paper. We then present a formal problem statement and discussions of several preliminary properties. An integer programming model follows.

Notation:

$\mathcal{J}=\{J_1 J_2,\ldots J_n\}$	the set of jobs to be processed;
p_i	processing time of job J_i on machine M_1;
q_i	processing time of job J_i on machine M_2;
α_i	resource requirement of job J_i;
β_i	amount of the resource returned by job J_i;
$\delta_i=\beta_i-\alpha_i$	contribution of job J_i;
$\sigma=(\sigma_1,\sigma_2,\ldots,\sigma_n)$	a particular sequence of the jobs (assumed for the case of permutation schedules);
S	a particular schedule that assigns a starting time to each of the $2n$ operations;
V	initial resource level;
v_t	resource level at time $t\geq 0$;
$Z(\sigma),Z(S)$	makespan of feasible schedule σ and S, respectively;
$s_{m,i}(S)$	starting time of job J_i on machine M_m in schedule S, $m=1,2$;
$C_{m,i}(S)$	completion time of job J_i on machine M_m in schedule S, $m=1,2$.

We formally state the problem as follows: From time zero onwards, a set of jobs $\mathcal{J}=\{J_1 J_2,\ldots J_n\}$ is available to be processed in a two-stage flowshop consisting of machines M_1 and M_2. Initially, the common resource pool contains V units of a single type of resource. Job $J_i\in\mathcal{J}$ can start processing only if machine M_1 is not occupied and the resource level in the pool is not less than α_i. Once job J_i starts processing, it immediately consumes α_i units of the resource from the resource pool and takes p_i units of time on M_1. After the machine-one operation is completed, q_i units of time is required to complete its resource recycling operation on M_2. Upon the completion of job J_i on machine M_2, it returns β_i units of the resource to the resource pool. No preemption on either machine is permitted. The goal is to minimize the makespan, i.e., to find a feasible schedule that completes all jobs in the shortest time.

2.1. Preliminary properties

We first present some known results on the basic relocation problem to facilitate discussion. In the basic relocation problem, the jobs are to be processed on a single machine and each job J_i is associated with two parameters α_i and β_i. The minimum initial resource level guaranteeing the existence of a feasible sequence for the jobs of $\mathcal{J}$ is called the *minimum resource requirement* of set $\mathcal{J}$. It is clear that no temporal parameters are involved. In fact, such a feasibility issue is linked to the classical two-machine flowshop scheduling problem to minimize the makespan. Given job set $\mathcal{J}$ in the basic relocation problem, we define job set $\hat{\mathcal{J}}=\{\hat{J}_1,\hat{J}_2,\ldots\hat{J}_n\}$ for the classical two-machine flowshop scheduling problem by letting α_i and β_i of job J_i as the processing times of job $\hat{J}_i$ on the stage-one and stage-two machines, respectively. The following lemma establishes the equivalence between the relocation problem and the two-machine flowshop scheduling problem.

Lemma 1 (*Kaplan and Amir [6]*). *The minimum resource requirement of job set $\mathcal{J}$ for the basic relocation problem is equivalent to the minimum total idle time on machine M_2 for the two-machine flowshop scheduling of job set $\hat{\mathcal{J}}$.*

Therefore we can determine the minimum resource requirement of a given job set in $O(n\log n)$ time using Johnson's algorithm. Cheng and Lin [11] further elaborated on the relation between the two-machine flowshop scheduling problem and the relocation problem, and highlighted the potential use of the notion of the relocation problem in the study of two-machine flowshop scheduling. The relocation problem provides a more intuitive interpretation of Johnson's algorithm for makespan minimization. To perform a set of projects, it is beneficial to undertake the projects that make profits ($\delta_i=\beta_i-\alpha_i\geq 0$) first and the remaining projects ($\delta_i=\beta_i-\alpha_i<0$) follow. The projects that make profits are arranged in non-decreasing order of their investment requirement (α_i). On the other hand, the remaining projects follow in non-increasing order of their returns (β_i).

In this section we consider permutation schedules, i.e., the processing sequences on both machines are identical. In a later section we will consider the non-permutation case. To illustrate the problem definition, we consider the following instance of three jobs with an initial resource level $V=5$.

Job	p_i	q_i	α_i	β_i
J_1	1	1	5	2
J_2	2	3	2	5
J_3	3	2	3	1

Fig. 1 shows schedule S_1, corresponding to sequence (1,2,3), and schedule S_2, corresponding to sequence (2,3,1). Given the objective of makespan minimization, we apply Johnson's algorithm by considering p_i and q_i and obtain sequence (1,2,3). As can be seen from the Gantt charts, S_1 is inferior to S_2. In fact, the makespan of S_1 is the worst.

A unique feature of $F2|rp|C_{\max}$ stems from the interaction between two two-machine flowshops. The feasibility issue in the basic relocation problem involves only resource constraints and is equivalent to the idle time issue in the context of two-machine flowshop scheduling. The second flowshop emerges from the separation of the resource consumption (or job processing) operation from the resource recycling operation of a job and the serial-processing requirement of the coupled operations. We may consider the $F2|rp|C_{\max}$ problem by reversing the roles of the two flowshops. Assume that the processing times are $\{\alpha_i\}$ and $\{\beta_i\}$ and the resource-related parameters are $\{p_i\}$ and $\{q_i\}$. The project is required to be completed by a deadline $D=V+\sum_{i=1}^{n}\beta_i$. We want to determine the minimum initial resource level required to ensure the existence of a schedule that is feasible with respect to the resource constraints and the deadline constraint. The orthogonal type of interaction of the two flowshops makes the problem hard to solve but may spur theoretical interest in the structural properties of the problem.

Now we introduce the time symmetry property of $F2|rp|C_{\max}$. Given an instance I and a schedule S, we construct a mirror instance I' as follows: Let $p'_i=q_i$, $q'_i=p_i$, $\alpha'_i=\beta_i$, $\beta'_i=\alpha_i$, and $V'=V+\sum_{J_i\in\mathcal{J}}\delta_i$. Denote the job set by $\mathcal{J}'=\{J'_1,J'_2,\dots,J'_n\}$. We look at the Gantt chart of schedule S from the right hand side and generate a mirror schedule S' of instance I' by setting

$$s_{1,i}(S')=Z(S)-C_{2,i}(S)\quad\text{and}\quad C_{2,i}(S')=Z(S)-s_{1,i}(S). \tag{1}$$

Assume that preemptions are not allowed in schedule S, i.e., if job J'_i starts at $s_{1,i}(S')$ (respectively, $s_{2,i}(S')$), then the processing of its first (respectively, second) operation occupies the time interval $[s_{1,i}(S'),s_{1,i}(S')+p'_i)$ (respectively, $[s_{2,i}(S'),s_{2,i}(S')+q'_i)$). Later, we will also address some results for the case without this assumption. Without preemptions, (1) implies the following equalities:

$$C_{1,i}(S')=Z(S)-s_{2,i}(S)\quad\text{and}\quad s_{2,i}(S')=Z(S)-C_{1,i}(S). \tag{2}$$

Inspecting the Gannt charts, we know, by (1) and (2), that instance I' and schedule S' are mirror to instance I and schedule S, and vice versa. Since schedule S' also satisfies the two-machine flowshop scheduling constraints

$$s_{2,i}(S')-C_{1,i}(S')=s_{2,i}(S)-C_{1,i}(S)\geq 0, \tag{3}$$

it has the same makespan as schedule S, i.e., $Z(S)=Z(S')$. The following lemma confirms the feasibility of schedule S'. The proof is adapted from Kononov and Lin [12].

Lemma 2. *Schedule S' is feasible with respect to the initial resource level V'.*

Proof. We show that the resource requirement is fulfilled at any time point $t\in[0,Z(S')]$. First of all, we examine the time point $t=Z(S')$, at which all jobs are completed. The resource level at $Z(S')$ is given by

$$v_{Z(S')}(S')=V'+\sum_{1\leq i\leq n}(\beta'_i-\alpha'_i)=V'-\sum_{1\leq i\leq n}(\beta_i-\alpha_i)=V\geq 0.$$

If $t<Z(S')$, then $\sum_{\{i|s_{1,i}(S')<t\}}\alpha'_i$ is the total amount of resource consumed and $\sum_{\{i|C_{2,i}(S')\leq t\}}\beta'_i$ is the total amount of resource returned. Therefore, we have

$$\begin{aligned}v_t(S')&=V'-\sum_{\{i|s_{1,i}(S')<t\}}\alpha'_i+\sum_{\{i|C_{2,i}(S')\leq t\}}\beta'_i\\&=\left(V+\sum_{1\leq i\leq n}(\beta_i-\alpha_i)\right)-\sum_{\{i|s_{1,i}(S')<t\}}\beta_i+\sum_{\{i|C_{2,i}(S')\leq t\}}\alpha_i\\&=V+\sum_{\{i|s_{1,i}(S')\geq t\}}\beta_i-\sum_{\{i|C_{2,i}(S')>t\}}\alpha_i\\&=V+\sum_{\{i|s_{1,i}(S')\geq t\}}(\beta_i-\alpha_i)+\sum_{\{i|s_{1,i}(S')\leq t\leq C_{2,i}(S')\}}\alpha_i.\end{aligned}$$

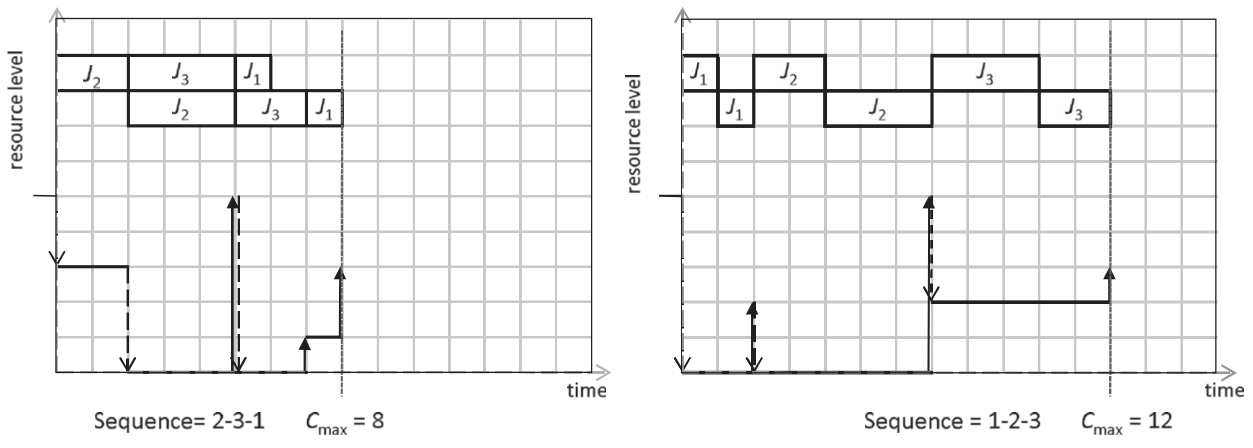

Fig. 1. Two example schedules.

By letting $\tau = Z(S) - t$ and (1), we have

$$v_t(S') = V + \sum_{\{i|C_{2,i}(S) \le \tau\}} (\beta_i - \alpha_i) - \sum_{\{i|s_{1,i}(S) \le \tau \le C_{2,i}(S)\}} \alpha_i. \quad (4)$$

Let $\tau_0 < \tau$ be the latest time point where some job completes or finishes in schedule S. Eq. (4) leads to

$$v_t(S') = V + \sum_{\{i|C_{2,i}(S) \le \tau_0\}} (\beta_i - \alpha_i) - \sum_{\{i|s_{1,i}(S) \le \tau_0 < C_{2,i}(S)\}} \alpha_i = v_{\tau_0}(S) \ge 0. \quad (5)$$

The last inequality follows from the fact that schedule S is feasible with respect to V. The proof is complete. □

With the fact that $Z(S) = Z(S')$ and Lemma 2, solution algorithms and complexity results for instance I can be applied to instance I', and vice versa, because the transformation between the two instances takes only linear time.

2.2. *Integer linear programming formulation*

In the following we provide an integer linear program formulation of the $F2|rp|C_{\max}$ problem using *positional variables*. We use the binary variable u_k^i to indicate if job J_i is scheduled in position k or not. The auxiliary variable $C_{m,k}$ denotes the completion time of the k-th job on machine M_m for $m=1$ or 2. We use the binary variables $y_{\ell,k}$ and $b_{i,\ell,k}$ to handle the resource availability relation between any two jobs. If $y_{\ell,k}=1$ for $\ell < k$, then the ℓ−th job completes on machine M_2 before the k-th job starts on machine M_1. If $b_{i,\ell,k}=1$, then job J_i is scheduled in position ℓ and it is completed on M_2 earlier than the k-th job starts processing on M_1. The variables $y_{\ell,k}$ and $b_{i,\ell,k}$ are related. We use these variables to help identify whether the resource returned by the ℓ−th job could be used by the later k-th job. If the resource is sufficient for processing the later k-th job, variable $b_{i,\ell,k}$ does not have to be 1. Similarly, it is possible to have $y_{\ell,k}=0$ even if ℓ−th job is completed before k-th job starts.

F2RP-Perm.

$$\text{Minimize} \quad C_{2,n} \quad (6)$$

subject to

$$C_{1,1} = \sum_{i=1}^{n} p_i u_1^i, \quad (7)$$

$$C_{1,k} - C_{1,k-1} - \sum_{i=1}^{n} p_i u_k^i \ge 0, \quad 1 < k \le n, \quad (8)$$

$$C_{2,k} - C_{2,k-1} - \sum_{i=1}^{n} q_i u_k^i \ge 0, \quad 1 < k \le n, \quad (9)$$

$$C_{2,k} - C_{1,k} - \sum_{i=1}^{n} q_i u_k^i \ge 0, \quad 1 < k \le n, \quad (10)$$

$$\sum_{i=1}^{n} u_k^i = 1, \quad 1 \le k \le n, \quad (11)$$

$$\sum_{k=1}^{n} u_k^i = 1, \quad 1 \le i \le n, \quad (12)$$

$$V - \sum_{r=1}^{k} \sum_{i=1}^{n} \alpha_i u_r^i + \sum_{r=1}^{k-1} \sum_{i=1}^{n} \beta_i b_{i,r,k} \ge 0, \quad 1 \le k \le n, \quad (13)$$

$$u_\ell^i + y_{\ell,k} - 2b_{i,\ell,k} \ge 0, \quad 1 \le i \le n, 1 \le \ell < k \le n, \quad (14)$$

$$C_{1,k} - C_{2,\ell} - \sum_{i=1}^{n} p_i u_k^i + (1 - y_{\ell,k})M \ge 0, \quad 1 \le \ell < k \le n, \quad (15)$$

$$u_k^i \in \{0,1\}, \quad 1 \le i,k \le n, \quad (16)$$

$$y_{\ell,k} \in \{0,1\}, \quad 1 \le \ell < k \le n, \quad (17)$$

$$b_{i,\ell,k} \in \{0,1\}, \quad 1 \le i \le n, 1 \le \ell < k \le n, \quad (18)$$

$$C_{m,k} \ge 0, \quad 1 \le k \le n, m = 1,2. \quad (19)$$

The objective function (6) is to minimize the makespan, i.e., the completion time of the job scheduled last. Constraints (7)–(10) confine all jobs to the following restrictions of flowshop scheduling: any job cannot start until its preceding job is completed and the machine-two operation of any job cannot be processed unless its machine-one operation is finished. Constraints (11) and (12) specify the one-to-one correspondence between jobs and positions, i.e., any position can accommodate at most one job and any job can occupy only one position. Constraints (13) require that the resource requirement of the k-th job cannot be greater than the current resource level, which is the initial resource level plus the resource returned by the jobs already completed before it can start. Constraints (14) ensure that if job J_i is scheduled in position ℓ and completes before the k-th job starts, then $b_{i,\ell,k}=1$. Constraints (15) check whether the ℓ−th job completes before the k-th job starts. The remaining constraints specify the ranges of the values of the variables.

3. Non-permutation scheduling

One of the main constraints characterizing a flowshop scheduling problem is whether all machines have the same processing sequence of the jobs. A schedule is called *permutational* if the job sequences on all machines in the shop are the same. For the classical two-machine flowshop problem $F2||C_{\max}$, it suffices to consider only permutation schedules [9]. However, the optimal schedule for the m-machine flowshop problem, $Fm||C_{\max}$, is not necessarily permutational, although the sequences on the first two machines or the last two machines are the same. For the $F2|rp|C_{\max}$ problem studied by Lin and Huang [16], permutation sequences were assumed. The issue of whether there exists at least one permutation optimal schedule for the $F2|rp|C_{\max}$ problem has not been investigated before. In this section we discuss the properties of permutation and non-permutation schedules.

Lemma 3. *The optimal schedule for* $F2|rp|C_{\max}$ *is not necessarily permutational.*

Proof. Consider the following instance of four jobs with an initial resource level $V = 6$.

	p_i	q_i	α_i	β_i
J_1	0	4	3	10
J_2	4	5	3	10
J_3	1	1	9	10
J_4	5	0	11	10

Fig. 2 shows the optimal permutation and non-permutation schedules. It is easy to verify that the optimal solution is obtained by the non-permutation schedule for the given instance. This example also shows that the optimal sequence may not

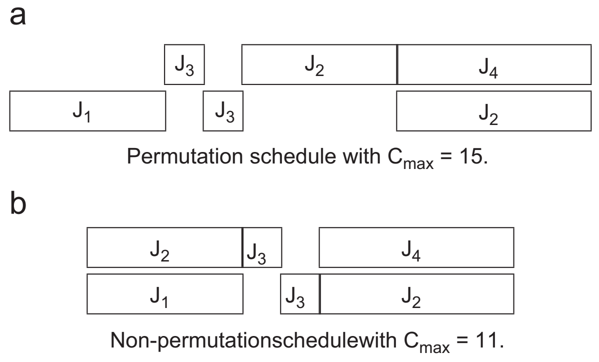
a
J3
J2
J4
J1
J3
J2
Permutation schedule with C max = 15.
b
J2
J3
J4
J1
J3
J2
Non-permutationschedulewith C max = 11.

Fig. 2. Gantt charts of permutation and non-permutation schedules. (a) Permutation schedule with $C_{\max}=15$. (b) Non-permutation schedule with $C_{\max}=11$.

necessarily be permutational even if all jobs return the same amount of the resource. □

The instance given above to illustrate Lemma 3 for non-permutation optimal schedules contains four jobs. It can be verified that instances with two or three jobs always permit permutational optimal schedules. Therefore, the four-job instance given above constitutes the smallest instance for which optimal schedules are not permutational.

Therefore, if we do not require schedules to be permutational, an optimal schedule may have different job sequences on the two machines. In the IP formulation (F2RP-Perm) in Section 2.2, we only considered permutation schedules. Below, we propose another formulation that considers non-permutation schedules. Since the job sequences may not necessarily be the same on both machines, we have to re-define the positional variables. If job J_i is scheduled in position k on machine M_m, then $u^i_{m,k}=1$; 0 otherwise.

F2RP-Non-Perm.

$$\text{Minimize} \quad C_{2,n} \tag{20}$$

subject to

$$C_{1,1}=\sum_{i=1}^{n} p_i u^i_{1,1}, \tag{21}$$

$$C_{1,k}-C_{1,k-1}-\sum_{i=1}^{n} p_i u^i_{1,k} \geq 0, \quad 1<k\leq n, \tag{22}$$

$$C_{2,k}-C_{2,k-1}-\sum_{i=1}^{n} q_i u^i_{2,k} \geq 0, \quad 1<k\leq n, \tag{23}$$

$$C_{2,k}-C_{1,j}-q_i+(1-u^i_{2,k})M+(1-u^i_{1,j})M \geq 0, \quad 1\leq i,j,k\leq n, \tag{24}$$

$$\sum_{i=1}^{n} u^i_{m,k}=1, \quad 1\leq k\leq n, m=1,2, \tag{25}$$

$$\sum_{k=1}^{n} u^i_{m,k}=1, \quad 1\leq i\leq n, m=1,2, \tag{26}$$

$$V-\sum_{r=1}^{k}\sum_{i=1}^{n}\alpha_i u^i_{1,r}+\sum_{\ell=1}^{k-1}\sum_{i=1}^{n}\beta_i b_{i,\ell,k} \geq 0, \quad 1\leq k\leq n, \tag{27}$$

$$u^i_{2,\ell}+y_{\ell,k}-2b_{i,\ell,k} \geq 0, \quad 1\leq i\leq n, 1\leq \ell<k\leq n, \tag{28}$$

$$C_{1,k}-C_{2,\ell}-\sum_{i=1}^{n} p_i u^i_{1,k}+(1-y_{\ell,k})M \geq 0, \quad 1\leq \ell<k\leq n, \tag{29}$$

$$u^i_{m,k}\in\{0,1\}, \quad 1\leq i,k\leq n, m=1,2, \tag{30}$$

$$y_{\ell,k}\in\{0,1\}, \quad 1\leq \ell<k\leq n, \tag{31}$$

$$C_{m,k}\geq 0, \quad 1\leq k\leq n, m=1,2. \tag{32}$$

Most constraints in this formulation are similar to those for the permutational case (F2RP-Perm). Constraints (21)–(24) define the processing characteristics of the two-machine flowshop. The difference between constraints (21)–(24) and (7)–(10) lies in the necessity for new variables, $u^i_{m,k}$, which are required because the two operations of the same job might be scheduled in different positions on the two machines. We use the new variables to determine the completion times of a job on different machines. Constraints (24) verify whether job J_i's machine-two operation starts after its machine-one operation by checking all pairs of positions on different machines.

We have shown that permutation schedules do not always provide optimal solutions in the general situation. Even when all jobs return the same amount of the resource, as shown in the instance of Lemma 3, the optimal schedule is not necessarily permutational. For some special cases, however, permutation schedules do provide the optimal solutions. The following lemma gives a special case that admits permutation optimal schedules.

Lemma 4. *If machine M_1 is dominant, i.e.,* $\min\{p_i\}\geq\max\{q_i\}$, *then it suffices to consider permutation schedules only.*

Proof. Consider an optimal non-permutation schedule S. Assume that the jobs are re-numbered by their processing order on machine M_1. Let J_i be the job with the smallest index whose machine-two operation does not start immediately after the completion of its machine-one operation, $C_{1,i}$. Since the recycling process does not demand any resource, it could be started once its machine-one operation is completed and machine M_2 is available. By assumption, machine M_2 is idle while job J_{i+1} is processed on machine M_1. We shift the recycling operation of job J_i backward to start at the completion of its machine-one operation. Clearly, the new schedule remains feasible. Moreover, since machine M_1 is dominant, $C_{2,i}\leq C_{1,i+1}$ must hold, implying that $C_{2,i+1}$ remains unchanged. Repeating the same argument, we will obtain a permutation optimal schedule. The proof is complete. □

By Lemmas 2 and 4, we can say that if machine M_2 is dominant, there exists an optimal schedule that is permutational.

Lemma 4a. *If machine M_2 is dominant, i.e.,* $\max\{p_i\}\leq\min\{q_i\}$, *then it suffices to consider permutation schedules only.*

4. Complexity results

Lin and Huang [16] showed that $F2|rp|C_{\max}$ without preemption is NP-hard in the strong sense even if only one parameter is fixed. The complexity status of the preemptive case ($F2|rp,pmtn|C_{\max}$) and some special cases, such as more than one parameter are assumed to be fixed, has not been settled. In this section we provide the complexity results for $F2|rp,pmtn|C_{\max}$ and various special cases of the $F2|rp|C_{\max}$ problem.

4.1. Hard cases

First we present a strong NP-hardness proof for the preemptive case, $F2|rp,pmtn|C_{\max}$, by a reduction from the 3-Partition problem, which is known to be NP-complete in the strong sense [18].

3-Partition. *Instance*: A positive integer B, a set of $3m$ elements $\{x_1,x_2,\ldots,x_{3m}\}$ such that $B/4<x_i<B/2$ for each index $i\in N=\{1,2,\ldots,3m\}$ and $\sum_{i=1}^{3m}x_i=mB$.

Question: Can index set N be partitioned into m disjoint sets $N_1,N_2,\ldots,N_m$ such that for each $1\le k\le m, \sum_{i\in N_k}x_i=B$?

Theorem 1. *The decision version of the $F2|rp,pmtn|C_{\max}$ problem is NP-complete in the strong sense even if all the jobs have the same resource requirement.*

Proof. The decision version of $F2|rp,pmtn|C_{\max}$ is $F2|rp,pmtn,C_{\max}\le\overline{C}|-$, where $\overline{C}$ is the threshold parameter imposing an upper bound on the schedule length. To establish the membership of this decision problem in NP, we have to show that the number of preemptions is bounded by a polynomial in the length of the problem input and that if preemptions occur at non-integer time points, the encoding of the length of each fractional part of an operation in an optimal schedule is also bounded by a polynomial in the length of the problem input. The two issues have been addressed by [19] for much more general scheduling problems with resource constraints and for a wide variety of objective functions, including all the classical ones. Given a schedule of the studied problem, the makespan can be calculated in polynomial time. Hence this decision problem belongs to NP.

Given an instance of 3-Partition, we construct an instance of $F2|rp,pmtn,C_{\max}\le\overline{C}|-$ with an initial resource level $V=2B$ as follows:

$3m$ ordinary jobs $J_i, 1\le i\le 3m$: $p_i=x_i, q_i=0,\ \alpha_i=B;\ \beta_i=B+x_i$.

m enforcer jobs J_i, $3m+1\le i\le 4m$: $p_i=0, q_i=B, \alpha_i=B, \beta_i=0$.

The threshold is $\overline{C}=mB$. We claim that a partition exists for the 3-Partition problem if and only if there exists a feasible schedule whose makespan is not greater than $\overline{C}$.

Let subsets $N_1,N_2,\ldots,N_m$ constitute a partition of set N in 3-Partition. We first schedule an enforcer job, say J_{3m+1}, followed by the three jobs, say J_1, J_2, J_3, defined by the elements of set N_1. On machine M_2, the processing of enforcer job J_{3m+1} is preempted when job J_1 completes on machine M_1. The preemption prevents job J_2 from being blocked due to a shortage of resource. Similarly, job J_2 preempts the processing of job J_{3m+1} on machine M_2 to facilitate the processing of job J_3. When the four jobs are finished, the resource level is again $2B$. Repeating the procedure for dispatching an enforcer job J_{3m+k} followed by the three ordinary jobs defined by the subset N_k, $1\le k\le m$, we can develop a feasible schedule with a makespan of mB.

To show the if-part of the claim, we assume that there is a feasible schedule S with the makespan equal to mB. While the total machine load of either machine is mB, the assumption implies that no idle time is allowed on either machine. Since all enforcers are identical, we assume that they are arranged in increasing order of their job indices. To avoid any idle time on machine M_2, enforcer job J_{3m+1} must be scheduled first. We consider the set of ordinary jobs scheduled to start on machine M_1 in the time interval $[0,C_{2,3m+1}(S))$. Let N_1 be the set of elements defining these ordinary jobs. If $\sum_{i\in N_1}x_i>B$, then there must be some ordinary job completed on machine M_1 later than B, which is the completion time of the first enforcer job J_{3m+1}. That is, idle time occurs on machine M_2. On the other hand, if $\sum_{i\in N_1}x_i<B$, then the resource level at the completion time of these ordinary jobs is $B+\sum_{i\in N_1}x_i<2B$. Then the second enforcer job J_{3m+2} starts its processing and reduces the resource level to $(B+\sum_{i\in N_1}x_i)-B$, which is less than B. At this moment, none of the unscheduled jobs, either ordinary or enforcer, can be successfully started. Therefore $\sum_{i\in N_1}x_i<B$ cannot be true. Equality $\sum_{i\in N_1}x_i=B$ must hold. Following the same line of reasoning, we can come up with sets N_2, …, N_m with $\sum_{i\in N_k}x_i=B, 2\le k\le m$, satisfied. A partition is identified for the 3-Partition problem and the proof is complete. □

It can be seen that for the instance of the $F2|rp,pmtn,C_{\max}\le\overline{C}|-$ problem constructed for a given instance of the 3-Partition problem, there is no optimal schedule without preemption. Thus the *Preemption Redundancy Property* [19] does not hold for this scheduling problem. Otherwise, if it did, then the NP-completeness of this decision problem would directly follow from its Preemption Redundancy Property and from the NP-completeness of its non-preemptive counterpart $\langle F2|rp,C_{\max}\le\overline{C}|-\rangle$ proved earlier by Lin and Huang [16]. Another note for the theorem is that all jobs in the instance constructed in the proof have the same resource requirement $\alpha_i=B$. By Lemma 2, we have:

Corollary 1. *The decision version of $F2|rp,pmtn|C_{\max}$ is NP-complete in the strong sense even if all jobs return the same amount of resource.*

In the following we discuss some special cases where preemption is not allowed. Lin and Huang [16] showed four special cases of $F2|rp|C_{\max}$ to be NP-hard in the strong sense, i.e., (1) $p_i=p$ for all i, (2) $q_i=q$ for all i, (3) $\alpha_i=\alpha$ for all i, and (4) $\beta_i=\beta$ for all i. That is, the problem remains hard to solve if one of the four parameters is assumed to be fixed. In the following we show that several further restricted cases remain NP-hard.

The following theorem addresses the case where all jobs are almost identical except for the processing times of the stage-two operations. In the following discussion we only outline how the instances of our scheduling problems are constructed from 3-Partition and omit the detailed analysis.

Theorem 2. $F2|rp,p_i=p,\alpha_i=\beta_i=1|C_{\max}$ *is strongly NP-hard.*

Proof. An instance of $F2|rp,p_i=p,\alpha_i=\beta_i=1,\overline{C}|-$ with $4m$ jobs, initial resource level $V=4$, and threshold $\overline{C}=(4m+1)B$ is given as follows:

$3m$ ordinary jobs J_i, $1\le i\le 3m$: $p_i=B, q_i=x_i, \alpha_i=1, \beta_i=1$;

m enforcer jobs J_i, $3m+1\le i\le 4m$: $p_i=B, q_i=3B, \alpha_i=1, \beta_i=1$.

A schedule with a makespan of $\overline{C}$ corresponds to a partition as required. □

The next result readily follows from the time symmetry property.

Corollary 2. $F2|rp,q_i=q,\alpha_i=\beta_i=1|C_{\max}$ *is strongly NP-hard.*

The next theorem concerns the special case where each job requires one unit of time for its stage-one operation and there are only two distinct stage-two processing times among all jobs. This restricted case remains strongly NP-hard.

Theorem 3. $F2|rp,p_i=1,q_i\in\{q_1,q_2\}|C_{\max}$ *is strongly NP-hard.*

Proof. An instance of $F2|rp,p_i=1,q_i\in\{q_1,q_2\},C_{\max}\le\overline{C}|-$ with an initial resource level $V=6B$ is given as follows:

$3m$ ordinary jobs J_i, $1\le i\le 3m$: $p_i=1, q_i=0, \alpha_i=B+x_i, \beta_i=4x_i$;

m enforcer jobs J_i, $3m+1\le i\le 4m$: $p_i=1, q_i=3, \alpha_i=2B, \beta_i=2B$.

A desired partition of 3-Partition exists if and only if there is a feasible schedule whose makespan is not greater than $\overline{C}=4m$. □

4.2. Solvable cases

In this section we study some polynomially solvable cases by developing exact solution algorithms. In Theorem 2 and Corollary 2,

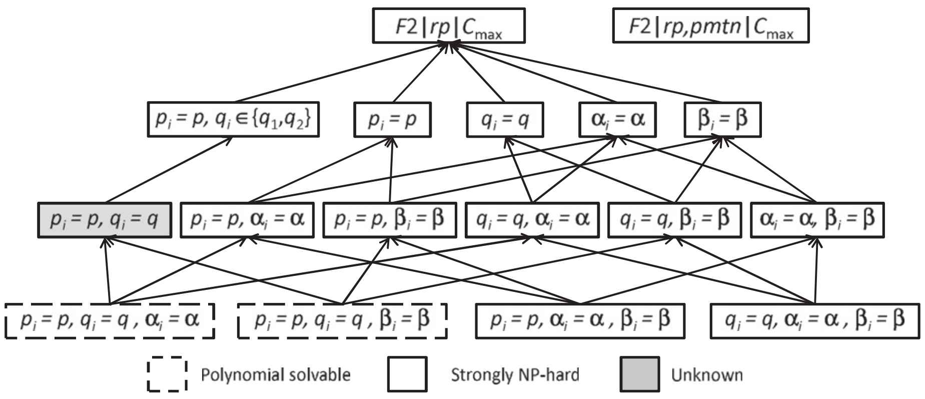
$F2|rp|C_{\max}$
$F2|rp,pmtn|C_{\max}$
$p_i = p,\ q_i \in \{q_1, q_2\}$
$p_i = p$
$q_i = q$
$\alpha_i = \alpha$
$\beta_i = \beta$
$p_i = p,\ q_i = q$
$p_i = p,\ \alpha_i = \alpha$
$p_i = p,\ \beta_i = \beta$
$q_i = q,\ \alpha_i = \alpha$
$q_i = q,\ \beta_i = \beta$
$\alpha_i = \alpha,\ \beta_i = \beta$
$p_i = p,\ q_i = q,\ \alpha_i = \alpha$
$p_i = p,\ q_i = q,\ \beta_i = \beta$
$p_i = p,\ \alpha_i = \alpha,\ \beta_i = \beta$
$q_i = q,\ \alpha_i = \alpha,\ \beta_i = \beta$
Polynomial solvable
Strongly NP-hard
Unknown

Fig. 3. Complexity hierarchy.

we showed that the problem remains strongly NP-hard even if all jobs are common in the three parameter values (p_i,α_i,β_i) or (q_i,α_i,β_i). It is interesting to consider other special cases where all jobs have common values of (p_i,q_i,α_i) or (p_i,q_i,β_i). In the following we show that the two special cases can be solved in polynomial time.

Lemma 5. *There exists an optimal schedule for $F2|rp,p_i=p,q_i=q,\alpha_i=\alpha|C_{\max}$ in which the jobs are scheduled in non-increasing order of β_i.*

Proof. By Lemma 4, there exists a permutation optimal schedule. Let $\sigma=(\sigma_1,\ldots,\sigma_n)$ be an optimal sequence in which $\beta_{\sigma_{i+1}} > \beta_{\sigma_i}$. Then, evidently, swapping the two jobs in the sequence retains the feasibility of the permutation schedule with the same length (and thus optimal). Repeating the swapping argument, if necessary, we can obtain a schedule with the sequence as required. □

Due to the time symmetry property, we have a similar result for the case $F2|rp,p_i=p,q_i=q,\beta_i=\beta|C_{\max}$.

Lemma 6. *The $F2|rp,p_i=p,q_i=q,\beta_i=\beta|C_{\max}$ problem can be solved in $O(n\log n)$ time by arranging the jobs in non-decreasing order of α_i.*

By Lemma 5, we can determine an optimal schedule for $F2|rp,p_i=p,q_i=q,\alpha_i=\alpha|C_{\max}$ by sorting the values of β_i in non-decreasing order. Similarly, by Lemma 6, we can determine an optimal schedule for $F2|rp,p_i=p,q_i=q,\beta_i=\beta|C_{\max}$ by sorting the values of α_i in non-increasing order. The following theorem thus follows.

Theorem 4. *For any instance of problem $F2|rp,p_i=p,q_i=q,\alpha_i=\alpha|C_{\max}$ or problem $F2|rp,p_i=p,q_i=q,\beta_i=\beta|C_{\max}$ with n jobs, there exists an optimal permutation schedule that could be found in $O(n\log n)$ time.*

5. Conclusion

In this study we considered the relocation problem with separate recycling operations. We examined several properties concerning whether the optimal schedule is permutational or not. We presented integer linear programming models for both the permutational and non-permutational settings. We settled the complexity status of several special cases of the problem, although some cases remain unresolved. Summarizing the results presented in this paper, we depict the complexity status of all studied cases of the problem in Fig. 3. It is seen that the complexity of the case where the processing times are fixed on both machines remains unknown. Furthermore, it is hard to determine the complexity of the most simplified unit execution time case $F2|rp,p_i=q_i=1|C_{\max}$. For further studies, theoretical analysis of approximation algorithms for $F2|rp|C_{\max}$ will be an interesting topic.

Acknowledgements

The authors are grateful to the reviewers for their professional comments, which have improved the presentation of the paper. This research was supported in part by The Hong Kong Polytechnic University under Grant number G-U471. Lin was also supported by the National Science Council of Taiwan under the Grant NSC96-2416-H-009-012-MY2.

References

[1] Blazewicz J, Lenstra JK, Rinnooy Kan AHG. Scheduling subject to resource constraints: classification and complexity. Discrete Applied Mathematics 1989;5:11–34.
[2] Brucker P, Drexl A, Mohring R, Neumann K, Pesch E. Resource-constrained project scheduling: notation, classification, models, and methods. European Journal of Operational Research 1999;112(1):3–41.
[3] Hammer PL. Scheduling under resource constraints - deterministic models. Annals of Operations Research 1986;7.
[4] Kaplan EH. Relocation models for public housing redevelopment programs. Planning and Design 1986;13(1):5–19.
[5] Amir A, Kaplan EH. Relocation problems are hard. International Journal of Computer Mathematics 1988;25:101–10.
[6] Kaplan EH, Amir A. A fast feasibility test for relocation problems. European Journal of Operational Research 1988;35:201–5.
[7] Kaplan EH, Berman O. Orient heights housing projects. Interfaces 1988;18(6):14–22.
[8] PHRG. New Lives for Old Buildings: Revitalizing Public Housing Project 1986. Public Housing Group, Department of Urban Studies and Planning, MIT, Cambridge. MA.
[9] Johnson SM. Optimal two- and three-stage production schedules with setup times included. Naval Research Logistics Quarterly 1954;1:61–7.
[10] Xie JX. Polynomial algorithms for single machine scheduling problems with financial constraints. Operations Research Letters 1997;21:39–42.
[11] Cheng TCE, Lin BMT. Johnson's rule, composite jobs and the relocation problem. European Journal of Operational Research 2009;192:1008–13.
[12] Kononov AVB, Lin MT. On the relocation problems with multiple identical working crews. Discrete Optimization 2006;21(4):368–81.
[13] Lin BMT, Liu ST. Maximizing the reward in the relocation problem with generalized due dates. International Journal of Production Economics 2008;115:55–63.
[14] Kononov AV, Lin BMT. Minimizing the total weighted completion time in the relocation problem. Journal of Scheduling 2010;13(2):123–9.
[15] Sevastyanov SV, Lin BMT, Huang HL. Tight complexity analysis of the relocation problem with arbitrary release dates, in submission.
[16] Lin BMT, Huang HL. On the relocation problem with a second working crew for resource recycling. International Journal of Systems Science 2006;37(1):27–34.
[17] Graham RL, Lawler EL, Lenstra JK, Rinnooy Kan AHG. Optimization and approximation in deterministic sequencing and scheduling: a survey. Annal of Discrete Mathematics 1979;5:87–326.
[18] Garey MR, Johnson DS. Computers and intractability: a guide to the theory of NP-completeness. San Francisco, CA: Freedman; 1979.
[19] Baptiste P, Carlier J, Kononov A, Queyranne M, Sevastyanov S, Sviridenko M. Structural properties of preemptive schedules. Discrete Analysis and Operations Research 2009;16(1):3–36. [in Russian].