Contents lists available at ScienceDirect

Int. J. Production Economics

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

Maximizing the reward in the relocation problem with generalized due dates

B.M.T. Lin[a,*], S.T. Liu[b]

[a] Department of Information and Finance Management, Institute of Information Management, National Chiao Tung University, Hsinchu 300, Taiwan
[b] Department of Information Management, National Chi Nan University, Puli 545, Taiwan

ARTICLE INFO

Article history:
Received 15 September 2006
Accepted 25 April 2008
Available online 10 May 2008

Keywords:
Relocation problem
Resource-constrained scheduling
Generalized due dates
NP-hardness
Branch-and-bound algorithm

ABSTRACT

The relocation problem, based on a public housing project in Boston, USA, is a generalized resource-constrained scheduling problem in which the amount of resources (new housing units) returned by a completed job (building) is not necessarily the same as the amount of resources (original housing units) it started out with for processing. In this paper we consider a variant where several generalized due dates are specified to define the number of new housing units that should be built in the entire duration of the project. Generalized due dates are different from conventional due dates in that they are job independent and common to all jobs. In the present study each generalized due date is given to specify an expected percentage of completion of the project. Given an initial number of temporary housing units, the goal is to find a feasible reconstruction sequence that maximizes the total reward over all generalized due dates. This paper investigates the time complexity of the problem. Two upper bounds and a dominance property are proposed for the design of branch-and-bound algorithms. Computational experiments are carried out to assess the efficiency of the proposed properties. The results show that the proposed properties can significantly reduce the time required for producing an optimal schedule.

1. Introduction

Resource constraints are one of the issues that are the most commonly addressed in project scheduling and management (Al-Fawzan and Haouari, 2005; Drezet and Billaut, 2008; Kobylański and Kuchta, 2007; Mingozzi et al., 1998; Pesenti and Ukovich, 2003). The study on the relocation problem arose from a public housing redevelopment project in Boston (PHRG, 1986; Kaplan, 1986). The goals of the project were to tear down some old buildings and build new ones in the same area. For this housing redevelopment scheme, the authorities provided sufficient temporary housing units for the tenants who would be evacuated from the area being redeveloped. More specifically, the authorities wanted a construction sequence of the new buildings and when and where all the displaced tenants were to be housed during the redevelopment process. This relocation problem can be looked at from an optimization point of view, in order to determine the minimum initial budget guaranteeing a feasible redevelopment sequence of the buildings. In this paper, we will consider a variant of the relocation problem that takes into account check points of the redevelopment.

In most scheduling problems, the due dates are job-dependent. That is, each due date is associated with a particular job and each individual job is expected to be completed before its corresponding due date. Hall (1986) first introduced the concept of generalized due dates. When scheduling using generalized due dates, the due dates are job independent such that a generalized due date is associated with a certain number of jobs that must be completed prior to that point in time. The idea of generalized due dates is commonly adopted in the real world. For example, a company might have a 2-year

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

0925-5273/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.ijpe.2008.04.009

Notation:

$J=\{1, 2, \ldots, n\}$	set of jobs to be processed
V_0	initial resource level
p_i	processing time of job i
a_i	amount of resource (housing units) required for processing job i
b_i	amount of resource (housing units) returned at the completion of job i
D_k	generalized due date k, $k=1, 2, \ldots, m$; note that $D_{m+1}=\sum_{i=1}^{n} p_i$
h_k	expected amount of housing units to be completed by D_k
C_i	completion time of job i
V_t	resource level in the common pool at time t
B_k	cumulative reward gained at due date D_k, i.e. $B_k=\sum_{C_i\leqslant D_k} b_i$
$Z(S)$	$\sum_{k=1}^{m} B_k$, total reward of particular schedule S

contract to produce 100 units of a product with the special requirement that one batch of 40 units must be delivered within the first year.

In this paper, we focus on the relocation problem incorporating generalized due dates. Consider the original setting of the relocation problem in the housing redevelopment project. The authorities and the construction company may set several generalized due dates based upon which they coordinate their transactions, such as pay by installments, project reviews, and so on. In this paper, we define a generalized due date as the time by which an expected percentage of the project is to be completed. For example, 50% of the newly built capacities of the project must be completed within the first year, and the entire project shall be completed at the end of the second year. At each generalized due date, if the actual percentage of the completion of the project is less than that is expected, the authorities may reduce the amount paid to the construction company as a penalty for the delay. On the other hand, it increases the amount paid as a reward for a more rapid progress. Such type of contract is quite common in the real world (Lock, 1996). For both theoretical and practical considerations, our study will investigate this situation.

This paper is organized into seven sections. In Section 2, we present a formal definition of the relocation problem with generalized due dates. This is followed by a literature review on the relocation problem and generalized due dates in Section 3. In Section 4, we present a proof of NP-hardness for the problem considered. Section 5 is dedicated to the design of two upper bounds and one dominance rule for developing a branch-and-bound algorithm. The computational experiments and numerical results are given in Section 6. Finally, we draw our conclusions in Section 7.

2. Problem formulation

In this section, we present a formal description of the problem under study. Relevant literature on the subject will also be addressed. Please note that throughout this paper, job and building, and number of housing units and amount of resource will be used interchangeably.

Formally, at time zero there is a common pool of V_0 units of single-type resource (housing units in the relocation problem), and a set of jobs (buildings in the relocation project) $J=\{1, 2, \ldots, n\}$ is to be processed on a single machine. Job $i\in J$ has three integer parameters: processing time p_i, amount of resource required a_i, and amount of resource returned b_i. At any time t, job i can only be considered for processing if the resource level V_t in the common pool is no less than a_i, i.e. $V_t\geqslant a_i$. When job i starts processing, it acquires and immediately consumes a_i units of resource and thus reduces the resource level in the resource pool by a_i. Upon its completion, job i immediately returns b_i units of resource back to the resource pool. No preemption is allowed, and at any moment the machine can process at most one job. In the housing redevelopment project, a_i and b_i, respectively, correspond to the numbers of housing units of building i before and after the redevelopment. The processing of job set J is associated with m generalized due dates (abbreviated as gdd hereafter) $D_1\leqslant D_2\leqslant \cdots \leqslant D_m$ such that each D_k is an integer belonging to the interval $[p_{\min}, P]$, where $p_{\min}=\min\{p_i|i\in J\}$and $P=\Sigma_{i\in J} p_i$. Each D_k is associated with a threshold or a number of new housing units, h_k, which are expected to be completed by D_k. Given a feasible construction sequence, we denote B_k as the total number of new housing units completed by due date D_k. If B_k is smaller than h_k, then the contractor will be penalized by a cost linearly dependent on h_k–B_k. On the other hand, if the construction progresses well and more units have been completed than expected, then a reward that is linearly dependent on B_k–h_k will be earned. We assume the unit penalty and unit reward to be the same. When the difference B_k–h_k is negative, the reward is then interpreted as a penalty. Given a certain amount of initial resource, there could be many feasible schedules of a given job set. The goal of the problem is to find a feasible schedule such that the sum of rewards over generalized due dates, $\sum_{k=1}^{m}(B_k-h_k)$ is maximized. Because $\sum_{k=1}^{m} h_k$ is fixed once the input is given, the problem is equivalent to the maximization of $\sum_{k=1}^{m} B_k$. For notational convenience, we use the standard three-field notation (Graham et al., 1979) $1|\text{rp, gdd}|\Sigma B_k$ to specify the problem of interest. The first field indicates that the problem environment is a single machine. The second field specifies special restrictions on the problems and on the jobs. It shows that the gdd version is considered in the relocation problem (rp). The third field indicates that the objective is the optimization of the cumulated reward gained over all generalized due dates.

Note that when m gdds are specified we do not include the one where the project is completely finished because the total processing length P is fixed once the input is given. Following the convention in the literature on scheduling, we use indices enclosed within brackets to denote the positions in a particular schedule. For example, $a_{[i]}$ is the resource requirement of the i-th job in a particular schedule, and $V_{[i]}$ is the amount of resource available when the i-th job in a schedule is finished.

To better understand the definition of the problem, we consider the set of 5 jobs shown in Table 1. Let us assume that $V_0 = 2$ and two gdds $D_1 = 12$ and $D_2 = 24$ are given. Schedule $S_1 = (3, 4, 2, 5, 1)$ shown in Fig. 1 is feasible and has an objective value of 26. Schedule $S_2 = (3, 5, 2, 4, 1)$ is also feasible and its objective value is 31, which is better than that of S_1. Consider another schedule $S_3 = (4, 5, 2, 3, 1)$ whose solution value is 35. However, S_3 is not feasible because job 4 is not allowed to start its processing with $V_0 = 2$. Therefore, schedule S_3 will not be considered as a candidate of the optimal solution. From the above three example schedules, we can see that it is not easy to find the optimal schedule while taking the feasibility issue into account.

3. Literature review

Kaplan (1986) first formulated, through an analytical study, the relocation problem of determining a construction sequence of buildings subject to an initial budget allocated by the authority. The theoretical significance of the relocation problem lies in its mathematical equivalence, established by Kaplan and Amir (1988), to Johnson's two-machine flow shop problem (Johnson, 1954). For more elaboration of the relation between the relocation problem and two-machine flowshop scheduling, the reader is referred to Cheng and Lin (2008). From a practical point of view, it also relates to the memory management in database systems (Amir and Kaplan, 1988) and a scheduling problem with financial constraints (Xie, 1997). From the aspect of scheduling theory, we may say that the relocation problem is a generalized version of conventional resource-constrained scheduling problems (Blazewicz et al., 1986, 1989).

Unlike most scheduling problems, the original version of the relocation problem mainly centers on the feasibility issue instead of on temporal considerations. However, it is interesting to note that the relocation problem is mathematically equivalent to the classical two-machine flow shop scheduling problem of minimizing the makespan, i.e. the maximum completion time. In Kaplan's study (1986), multiple working crews are assumed, i.e. more than one building can be simultaneously developed if the available resource is sufficient, and if the objective is to design a redevelopment schedule that has the minimum makespan. He proposed a myopic algorithm to construct approximate solutions. Amir and Kaplan (1988) showed that this problem is NP-hard by a polynomial reduction from the Partition problem. For other relevant studies on the relocation problem, the reader is referred to Amir and Kaplan (1988), Kaplan and Berman (1988), Kononov and Lin (2006, 2008), Lin and Tseng (1992, 1993) and Lin and Cheng (1999). Among these, the work by Lin and Tseng (1992) is the most relevant to our study. In it they sought to maximize the number of new rooms that can be completed before a common due date. This problem can be treated as a special case of the 1|rp, gdd|ΣB_k problem considered in this paper.

When gdds are considered in a scheduling problem, due dates are job independent and related to the entire project. Each gdd can be regarded as a checkpoint or milestone. Scheduling problems involving gdds were first proposed by Hall (1986), who identified two polynomially solvable cases and some NP-hard problems on a single machine. He suggested several applications, such as utility planning problems, survey design and scheduling problems, in various manufacturing environments. Sriskandarajah (1990) described an application from the public service sector: assume that a number of buses of a public bus company are under repair. The repair time for a bus is a function of its mileage and depreciation. All jobs have their own identities. Due dates are not bus dependent, but a given number of buses must be served by a certain date. Hall et al. (1991) described a particular application in the petrochemical industry, where a number of interchangeable heat exchangers must be maintained. Hall et al. (1991) provided an excellent survey on the computational complexities of the scheduling problems with gdds under various environments, such as single machine, parallel machines and shops. The job-dependent due date version is usually as hard as the gdd version. In some cases, there are several efficient algorithms to solve job-dependent

Table 1
A set of 5 jobs

Job	p_i	a_i	b_i
1	3	4	1
2	7	10	9
3	9	2	8
4	9	5	10
5	8	4	6

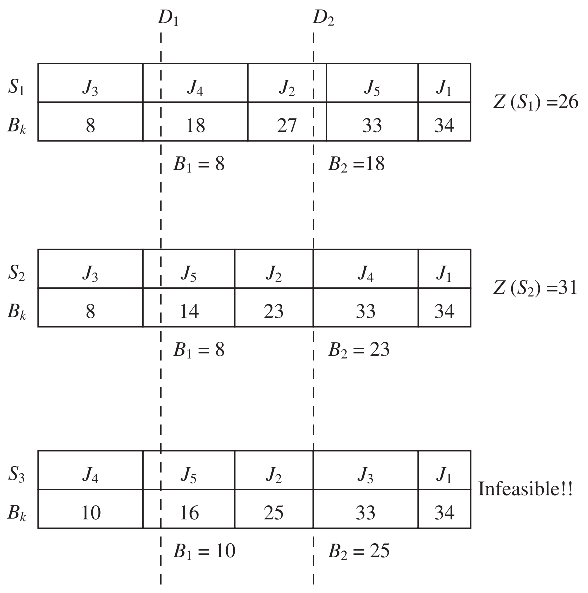

Fig. 1. Example for problem definition.

due date cases in polynomial time, but the gdd versions of these cases are NP-hard (e.g., the single-machine case of minimizing maximum lateness with precedence relations; see Sriskandarajah, 1990). After the publication of Hall et al. (1991), only a limited number of research works on scheduling problems with gdds have been reported. Recently, Mosheiov and Oron (2004) derived two lower bounds for solving maximum tardiness and total tardiness problems and compared the values of these bounds with the SPT heuristic on parallel identical machines.

4. Complexity results

Clarifying the position of a studied problem in the complexity hierarchy is essential to the choice of methodologies for coping with the problem. Therefore, in this section, we present some results concerning the complexity status of the $1|\text{rp}, \text{gdd}|\Sigma B_k$ problem. Our study is based on a polynomial reduction from 3-Partition, which is strongly NP-hard (Garey and Johnson, 1979).

3-Partition: Given a non-negative integer M and a set of non-negative integers $A=\{x_1, x_2, \ldots, x_{3t}\}$ with $M/4<x_i<M/2$ for each x_i and $\sum_{i=1}^{3t} x_i = tM$, is there a partition $A_1, A_2, \ldots, A_t$ for set A such that $\sum_{x_i\in A_k} x_i = M$ for each subset A_k, $1\leqslant k\leqslant t$?

Theorem 1. : *The* $1|\text{rp}, \text{gdd}|\Sigma B_k$ *problem is strongly NP-hard.*

Proof. The decision version of the $1|\text{rp}, \text{gdd}|\Sigma B_k$ problem is clearly in NP. We now present a polynomial reduction from 3-Partition. Given an instance of 3-Partition, we create an instance of $4t$ jobs as follows:

Let ω be a number such that $\omega > tM^2$. Define

Small jobs: $p_i = x_i$, $a_i = M+x_i$, $b_i = 2x_i$ for $i = 1, 2, \ldots, 3t$;

Large jobs: $p_{3t+i} = i\omega$, $a_{3t+i} = M$, $b_{3t+i} = 3M$ for $i = 1, 2, \ldots, t$;

Generalized due dates: $D_k = \sum_{l=1}^{k}(l\omega + M) = k(k+1)\omega/2 + kM$, $k = 1, 2, \ldots, t$;

Initial resource level: $V_0 = M$.

The above two instances show that there is a partition as specified for set A if and only if there is a feasible schedule for the $1|\text{rp}, \text{gdd}|\Sigma B_k$ problem whose total reward is no less than $5t(t+1)M/2$.

IF: Let $A_1, A_2, \ldots,$ and A_t be a partition of set A as specified in 3-Partition. We construct a schedule $3t+1, J_1, 3t+2, J_2, \ldots, 4t, J_t$ for the $1|\text{rp}, \text{gdd}|\Sigma B_k$ problem, where J_k, $1\leqslant k\leqslant t$, denotes the set of jobs defined by the elements in A_k. The configuration is shown in Fig. 2. First, it is easy to know that this schedule is feasible. Second, the completion time of the last job in J_k, $1\leqslant k\leqslant t$, is $k(k+1)\omega/2+kM$, which is equal to D_k and the number of new rooms completed at this point is $5kM$. Therefore, the total reward of the schedule gained over all gdds is $5M+10M+\cdots 5tM = 5t(t+1)M/2$.

ONLY_IF: Let S be a feasible schedule with a total reward no less than $5t(t+1)M/2$. Because the large jobs are the same except for the fact that jobs with larger indices have longer processing times, we may assume without loss of generality that in schedule S the large jobs are scheduled in the increasing order of their indices. Because small jobs have negative contributions ($b_i - a_i < 0$), scheduling any small job first will result in an insufficiency of resource for the processing of its successor. Therefore, large job $3t+1$ must be scheduled at the first position so as to ensure feasibility. The remainder of our proof is to show that the inequality $\sum_{l=1}^{k} B_l \leqslant 5\,kM$ will hold.

Let J_1 be the set of small jobs scheduled between jobs $3t+1$ and $3t+2$. We consider the following two cases.

Case 1: $\sum_{i\in J_1} x_i > M$

Because $M/4<x_i<M/2$ for each x_i, $|J_1|\geqslant 3$ must hold. If $|J_1|\geqslant 4$, then the amount of resource available at the completion of jobs in J_1 is

$$
\begin{aligned}
3M + \sum_{i\in J_1}(x_i - M) &= 3M + \sum_{i\in J_1} x_i - |J_1|M \\
&< 3M + \frac{|J_1|M}{2} - |J_1|M \\
&\quad \text{(because } x_i < M/2) \\
&= 3M - \frac{|J_1|M}{2} \\
&\leqslant 3M - 2M \ \ (\text{because} |J_1|\geqslant 4) \\
&= M.
\end{aligned}
$$

In other words, the resource level is less than M, which cannot support the ongoing processing of job $3t+2$. This results in a case of infeasibility. Therefore, $|J_1|$ must be equal to 3. Since $\sum_{i\in J_1} x_i > M$, the last job in J_1 starts before and ends after due date $D_1 = \omega+M$. That is, only job $3t+1$ and the first two jobs can be completed before D_1. Therefore, the number of completed housing units B_1 is less than $5\,M$.

Case 2: $\sum_{i\in J_1} x_i \leqslant M$

If $\sum_{i\in J_1} x_i = M$, then the last job in J_1 finishes at exactly D_1, and thus $B_1 = 3M + \sum_{i\in J_1} 2x_i = 5M$. On the other hand, if $\sum_{i\in J_1} x_i < M$, then the last job in J_1 will be completed before D_1 and the large job $3t+2$ following the jobs of J_1 will commence before D_1. Because $p_{3t+2} = 2\omega > \omega+M = D_1$, the large job $3t+2$ cannot be completed by the first due

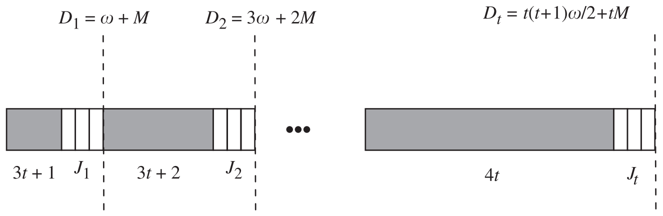

Fig. 2

date. At D_1 only the jobs in $J_1 \cup \{3t+1\}$ can be finished. The number of completed new rooms B_1 therefore is $3M + \sum_{i \in J_1} 2x_i < 5M$.

Excluding the infeasibility cases, we can have that $B_1 \leqslant 5M$. Following this line of analysis, we consider the set of smaller jobs, J_2, scheduled between the large jobs $3t+2$ and $3t+3$. Considering only the feasible cases, we have the following inequality:

$$B_1 + B_2 = (3M + \sum_{i \in J_1} 2x) + (3M + \sum_{i \in J_1} 2x_i) \leqslant 10M.$$

Repeating the same process, we have that in schedule S

$$\sum_{l=1}^{k} B_l \leqslant 5\,\mathrm{kM}, \quad l = 1, 2, \ \ldots, \ t-1,$$

$$\sum_{l=1}^{t} B_l = 5tM.$$

The fact that the total reward of schedule S is no less than $5M + 10M + \cdots + 5tM$ implies

$$\sum_{l=1}^{k} B_l = 5kM, \quad k = 1, 2, \ \ldots, \ t.$$

That is, the number of new housing units of the jobs in J_k is exactly $2M$, or $\sum_{i \in J_k} 2x_i = 2M$. Let A_k denote the subset whose elements define the jobs in J_k. Then, $\sum_{x_i \in A_k} x_i = M$ for any subset A_k, $1 \leqslant k \leqslant t$. Therefore, a partition of set A is derived and the proof is complete. □

The established computational complexity reveals that it is very unlikely to devise polynomial time algorithms for producing optimal schedules. It is worth noting that the $1|\mathrm{rp}, \mathrm{gdd}|\Sigma B_k$ problem remains difficult to solve even if all jobs have the same processing time and only one generalized due date is given (Lin and Tseng, 1992). When there is only one common gdd, the problem looks similar to the knapsack problem in the sense that the optimization criterion is to maximize the total reward subject to the capacity constraint, or the common due date. However, there is something else worth noting. In the knapsack problem, we do not need to order the selected or discarded items. However, this is not the case in the $1|\mathrm{rp}, \mathrm{gdd}|\Sigma B_k$ problem. The order of the early jobs as well as the late jobs is crucial because we have to ensure the feasibility of the sequence of all jobs. Therefore, a pseudo-polynomial time dynamic programming cannot be directly formulated. It should also be noted that whether the common due date problem is weakly NP-hard or strongly NP-hard remains open.

5. Branch-and-bound algorithm

Branch-and-bound is one of the most commonly used implicit enumerative methods for making optimal decisions. To design a branch-and-bound algorithm, we usually construct an enumeration tree to represent the solution space and then exploit structural properties to reduce the efforts required for probing through the solution space. In this section, we develop a novel branch-and-bound algorithm to deal with the problem under study. Specifically, we design two upper bounds and a dominance rule in order to prune unnecessary branches in the enumeration tree. The details of the proposed branch-and-bound algorithm are discussed in the following.

In an enumeration tree, each node represents a decision point whether or not a specific job is to be scheduled at the first unoccupied position. In the worst case, there will be $O(n!)$ leafs in the tree. The search strategy we use is the depth-first search, which does not demand sophisticated program control and data structures. Before the branch-and-bound algorithm commences, an objective value can be derived by using a heuristic method or simply by branching to the bottom node of the left-most path of the enumeration tree. The objective value is used as the incumbent value, which will be repeatedly updated if better solution values are encountered in the exploration process. In our algorithm, we do not use any heuristic to derive a lower bound. Instead we set the incumbent value to be 0.

For a partial schedule, if its upper bound is less than or equal to the incumbent value, all solutions below this partial schedule can be eliminated because it is impossible to find a better solution in the sub-tree rooted at this partial schedule. First, we develop upper bounds so as to help prune the search tree.

If it exists, let S be a feasible schedule. In the following, we create another job set J' out of the jobs in set J. We remove the association of a_i, b_i and p_i of each job i. So, we have three sets of values $\{a_1, a_2, \ldots, a_n\}$, $\{b_1, b_2, \ldots, b_n\}$ and $\{p_1, p_2, \ldots, p_n\}$. Let $a_{(i)}$ denote the i-th smallest element of $\{a_1, a_2, \ldots, a_n\}$, $b_{(i)}$ the i-th largest element of $\{b_1, b_2, \ldots, b_n\}$ and $p_{(i)}$ the i-th smallest element of $\{p_1, p_2, \ldots, p_n\}$. Job i' of set J' is defined by $a_{(i)}$, $b_{(i)}$ and $p_{(i)}$. Table 2 shows job set J' derived from the set J shown in Table 1.

It is clear that set J' exhibits an ideal structure for us to derive a useful result as given in the following lemma.

Lemma 1. **:** *For any feasible schedule S of set J and the initial resource level V_0, there is a feasible schedule S' of set J' such that $Z(S') \geqq Z(S)$.*

Proof. . In the proof we conduct a three-step transformation from S to another schedule S' of J' without decreasing its objective value.

(a) If in the feasible solution S there are two consecutive jobs i and j such that job i precedes job j and $p_i > p_j$, then we interchange the values of p_i and p_j such that job i has a new processing time p_j and job j has a new processing time p_i. Please note that all other parameters of the jobs are not changed. It is clear that after

Table 2
Job set J' derived from the example in Table 1

Job	p_i	a_i	b_i
1	3	2	10
2	7	4	9
3	8	4	8
4	9	5	6
5	9	10	1

the interchange the new schedule is still feasible and the completion time of any job is not increased. Because there might be more jobs being completed before some due dates after the interchange, the objective value cannot decrease. Repeating this process, we can finally have a new schedule in which the jobs are sequenced in the non-decreasing order of processing times without decreasing the total reward.

(b) With the schedule as derived above, we start the second-step transformation. For the schedule from (a), we interchange the values of a_i and a_j if job i precedes job j and $a_i > a_j$. Please note that all other parameters of the jobs are not altered. After the interchange, the new schedule is feasible because the resource level prior to starting job i is no less than a_i. Moreover, the completion time of any job is not increased and the objective value will not decrease. Repeating this process, we can finally come up with a new schedule in which the jobs are sequenced in the non-decreasing order of resource requirement without decreasing the total reward.

(c) With the schedule derived from (b), we interchange the values of b_i and b_j if job i precedes job j and $b_i < b_j$. All other parameters of all jobs remained unaltered. After the interchange, the new schedule is still feasible and the completion time of any job remains the same. Because there might be more reward earned prior to some due dates after the interchange, the objective value cannot decrease. Repeating this process will result in a new schedule in which the jobs are sequenced in a non-decreasing order of the amount of resource returned without decreasing the total reward. Moreover, the jobs in the final schedule constitute the instance set J'.

The lemma directly suggests that there is some schedule of set J' which can provide an upper bound on the optimal solution value of set J. Furthermore, from the three-step transformation used in the proof, we know that the sequence of jobs of J' arranged in the increasing order of their indices is one of interest. Consequently, we can find an upper bound on the optimal solution value of set J by first deriving set J' and then constructing the sequence S' as in the proof of Lemma 1. The computing time required for deriving this bound is $O(n \log n)$ for the sorting stage. However, it can be reduced to $O(n)$ for a partial schedule if the parameters are sorted with a preprocessing procedure after the original data set is given.

In the following, we relax the resource requirement to establish the second upper bound. If none of the jobs need any resource for their processing, then the decision is constrained only by the processing time and the amount of resource returned by each job. Following this relaxation, the problem with one gdd becomes the knapsack problem on condition that p_i and b_i, respectively, correspond to the size and the value of an item, whereas the due date corresponds to the capacity of the knapsack. It is evident that our problem is more general and more sophisticated than the knapsack problem because there are many gdds and the reward is calculated on an accumulative base.

Let S be a feasible schedule, if it exists. In the following, we relax two constraints and find another schedule S' for job set J. First, we redefine the "completion" status of a job. Generally speaking, we call a job "completed" if it has been continuously processed from its starting time to its completion. If a job is allowed to be partially processed and contributes a fraction of its original reward, then it is a preemptive version. In the case where preemption is allowed, the objective function is denoted by $Z'(.)$. Then, $Z'(S')$ is no less than $Z(S)$ because integer solutions are also solutions to the relaxed problem. Second, the resource requirement for each job is relaxed in such a way that each job can be processed at any time without considering the resource requirement. Under the above two conditions, we reorder the jobs in the non-increasing order of b_i/p_i and find another schedule S'. It is clear that schedule S' is feasible because none of the jobs require any resource and $Z'(S')$ is an upper bound of the schedule S. Similarly, this bound can be derived in $O(n)$ time.

To further remove unnecessary branching in the search tree, we introduce the following dominance rule.

Dominance**:** For jobs i and j, if $p_i \leqslant p_j$, $a_i \leqslant a_j$ and $b_i \geqslant b_j$, then there exists some optimal schedule in which job i precedes job j.

The significance of this rule is that it is static over the whole exploration session in the enumeration tree. Therefore, the dominance relation can be easily examined for branching from each partial schedule.

6. Computational study

In this section, we present the computational experiments designed and performed to examine the efficiency of the proposed upper bounds and dominance rule in the branch-and-bound algorithm. The experiments were conducted on a personal computer with a Pentium-IV 1.7 GHz CPU and 1 GB RAM. The operating system is Linux Red Hat 8. The programs were written in C++. In all of the generated cases, processing time p_i, amount of required resource a_i, and amount of returned resource b_i were uniformly drawn from [1, 100]. To generate an appropriate initial resource level V_0, the following approach was adopted. We first determined the minimum amount of initial resource required to ensure the existence of a feasible schedule. This can be achieved by applying the Kaplan and Amir's algorithm. Then, the initial resource level in our experiment was set to be 120% of the determined value. This was done to ensure an allowance of 20% more than the minimum resource requirement.

To generate due dates or deadlines for computational experiments, Hall and Posner (2001) suggested a scheme that incrementally determines the due dates or deadlines job by job. In the present study, the number of distinct gdds is small, and the gdds are job-independent and usually are evenly distributed over the planning horizon. Therefore, the scheme developed by Hall and Posner (2001) was not applied. The number of generalized due dates m in our experiment was 2, 3, 4 or 5. We did not adopt a larger m because in a real project it is unusual to define a great number of checkpoints. For example, in an information system project, the payment can be exercised in three stages: (1) contract is signed; (2) physical facility is constructed and hardware is installed; and (3) information system is implemented and launched. In the computational experiments, the time point of each gdd was generated by dividing the time horizon of makespan P into $m+1$ equal parts. For example, if $m = 2$ and $P = 36$, then $D_1 = 12$, $D_2 = 24$, and the final due date is 36. We tested problems with sizes of $n = 5$, 10, 15 and 20 when $m = 2$ or 3, and $n = 5$, 10, 15, 20 and 25 when $m = 4$ or 5.

For each combination of m and n, 10 independent instances were generated. In the running session, a time limit of 30 min was imposed for each instance. That is, if the algorithm could not successfully report an optimal solution within 30 min it would abort with a failure. Three implementation options of the algorithms were compared: UB, only the upper bound was applied; DR, only the dominance rule was applied; and UB_DR, both the upper bound and the dominance rule were jointly incorporated. It should be noted that UB is the minimum of the two upper bounds defined in the previous section. Regarding the performance statistics, we kept track of the following information: (1) number of job sets that were successfully solved; (2) average number of nodes explored for the test sets that were successfully solved; and (3) average execution time elapsed for the test sets that were successfully solved. The results are summarized and shown in Table 3. In each cell of the columns, there are three sub-columns which summarize the number of solved job sets (referred to as #_Opt), average number of generated nodes (#_Nodes), and average elapsed time (s.) (Time), respectively.

With reference to Table 3, the results first dictate that the upper bounds achieve a better performance, especially in terms of the number of visited nodes, than the dominance rule. For example, the ratio of the numbers of generated nodes of UB and DR for the case of $m = 2$ and $n = 15$ is 1:50, but the ratio between the two running times is only about 1:13. This reflects the fact that for a single node DR takes less time than UB. The second important observation is that when UB and DR were jointly deployed they provide strong synergy effects. Incorporating different properties for solving hard optimization problems usually deteriorates an algorithm's performance. The reason behind such a negative effect is that the sets of nodes cut off by two different properties might greatly overlap such that the gains from the reduction of generated nodes are less than the loss from the increase of running time required by the second property. However, the incorporation of UB and DR in our branch-and-bound algorithm does not suffer from this usual negative effect. The computational results in Table 3 reveal that the joint implementation of UB_DR can solve problems much faster than those of UB and DR separately. Let us look at the particular case of $n = 15$ and $m = 3$. The average time needed by UB_DR is merely 0.1 s, while those by UB and DR are 7.6 and 283 s, respectively. The superiority of UB_DR can also be demonstrated by the fact that the number of nodes explored by UB_DR is only

Table 3
Computational results of the branch-and-bound algorithm

N	*UB*			*DR*			*UB_DR*		
	#_Opt	#_Nodes	Time	#_Opt	#_Nodes	Time	#_Opt	#_Nodes	Time
$m = 2$									
5	10	26	0.0	10	43	0.0	10	9	0.00
10	10	2884	0.0	10	41,354	0.0	10	530	0.00
15	10	2.8E+06	27.7	10	1.4E+08	353.6	10	146,036	1.10
20	0	–	–	0	–	–	8	1.9E+07	399.13
$m = 3$									
5	10	14	0.0	10	43	0.0	10	8	0.00
10	10	683	0.0	10	15,003	0.0	10	161	0.00
15	10	477,126	7.6	10	1.6E+08	283.0	10	15,633	0.10
20	1	6.9E+07	1545.0	0	–	–	9	1.6E+06	45.44
$m = 4$									
5	10	16	0.0	10	44	0.0	10	10	0.0
10	10	1658	0.0	10	23,671	0.0	10	299	0.0
15	10	533,018	21.6	8	5.1E+07	280	10	19,765	0.4
20	5	1.5E+07	584.0	0	–	–	9	6.3E+06	236.1
25	3	1.2E+07	621.3	0	–	–	9	3.2E+06	141.9
$m = 5$									
5	10	16	0.0	10	36	0.0	10	7	0.0
10	10	1370	0.0	10	40,302	0.0	10	430	0.0
15	10	257,926	10.5	9	8.6E+07	426.4	10	124,739	4.6
20	4	1.9E+07	661.3	0	–	–	10	1.1E+06	80.2
25	0	–	–	0	–	–	3	3.8E+07	362.3

about 3.3% of that explored by UB, and less than 0.1% of those explored by DR. It is interesting to note that the synergy effect becomes more significant when the number of jobs increases. For the case of $n = 20$ (a larger number of jobs) and $m = 2$, UB_DR still solved 8 out of the ten instances, while UB and DR could not successfully solve any set of data. Such a phenomenon can also be observed from the results of large-scale instances. Another interesting observation is related to the number of gdds. With regard to the number of nodes or the amount of running time, it appears that the numerical results of the cases with more gdds are better than those with less gdds. However, this trend disappears between $m = 4$ and $m = 5$ for the instances of $n = 20$ or 25. This phenomenon could be attributed to the average number of allocated jobs in between every two consecutive gdds. When the average number of jobs allocated in interval $[D_{i-1}, D_i]$ is large (say $n = 15$ and $m = 2$), the computing time required by determining the optimal solution(s) to a problem similar to the Knapsack problem is relatively longer. On the other hand, when the average number of jobs allocated in interval $[D_{i-1}, D_i]$ becomes too small (say $n = 15$ and $m = 5$), there would be a combinatorial number of schedules that have similar solution values. Such situations may diminish the power of bounding functions.

Before closing this section, we like to address another observation on the two lower bounds. Although there is no strict theoretical relation between UB_1 and UB_2, in our experiments on 240 extra instances we found UB_1 to be greater than or equal to UB_2.

7. Concluding remarks and future research

In this paper, we studied the $1|rp, gdd|\Sigma B_k$ problem, which incorporates the concept of generalized due dates in the relocation problem. This problem demonstrates both theoretical and practical significance. For the $1|rp, gdd|\Sigma B_k$ problem, we have shown its strong NP-hardness by a polynomial-time reduction from 3-Partition. To derive optimal solutions, we developed two upper bounds and one dominance rule as the main ingredients of branch-and-bound algorithms. We designed and conducted computational experiments to study the efficiency of the proposed properties. The numerical results show that the proposed properties can significantly reduce the time (also, the number of nodes generated in the enumeration tree) required for producing optimal solutions. In addition, the synergy effects introduced by the joint deployment of these properties are very informative.

Even though the proposed properties can prune unnecessary explorations during the solution-finding process to a certain degree, the problem scale that our algorithm can solve to optimality is still limited. Therefore, there is considerable room for developing more efficient properties so that optimal solutions of larger problems can be derived in an acceptable time. One possible approach for coping with the $1|rp, gdd|\Sigma B_k$ problem could be the development of Lagrangian relaxation.

As mentioned earlier, the $1|rp, gdd|\Sigma B_k$ problem remains NP-hard even if all jobs require the same processing time and only one gdd is given. However, there is no clue to the existence of a pseudo-polynomial time dynamic program. Therefore, whether the $1|rp, p_i = p, D_i = D|\Sigma B_k$ problem is weakly NP-hard remains open. For further study, considering generalized relocation problems, say with new machine environments or new performance measures, could be a worthy direction because the relocation problem is a new type of resource-constrained scheduling problem. The introduction of gdds also opens up a new area of scheduling research, that is, to investigate classical scheduling problems with the consideration of gdds.

Acknowledgements

The authors are grateful to the anonymous referees who have provided constructive comments on an earlier version of this paper. The first author was partially supported by a Taiwan–Russia joint grant under contract numbers NSC95-2416-H-009-013 and RP05H01 (05-06-90606-HHCa).

References

Al-Fawzan, M.A., Haouari, M., 2005. A bi-objective model for robust resource-constrained project scheduling. International Journal of Production Economics 96, 175–187.

Amir, A., Kaplan, E.H., 1988. Relocation problems are hard. International Journal of Computer Mathematics 25, 101–110.

Blazewicz, J., Cellary, W., Slowinski, R., Weglarz, J., 1986. In: Hammer, P.L., Baltzer, J.C. (Eds.), Annals of Operations Research: Scheduling under Resource Constraints—Deterministic Models. Scientific Publishing Company, Basel, Switzerland.

Blazewicz, J., Lenstra, J.K., Rinnoy Kan, A.H.G., 1989. Scheduling subject to resource constraints: Classification and complexity. Discrete Applied Mathematics 5, 11–24.

Cheng, T.C.E., Lin, B.M.T., 2008. Johnson's rule, composite jobs and the relocation problem. European Journal of Operational Research (in press).

Drezet, L.-E., Billaut, J.-C., 2008. A project scheduling problem with labour constraints and time-dependent activities requirements. International Journal of Production Economics 112, 217–225.

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

Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., 1979. Optimization and approximation in deterministic sequencing and scheduling: A survey. Annals of Discrete Mathematics 5, 287–326.

Hall, N.G., 1986. Scheduling problems with generalized due dates. IIE Transactions 18, 220–222.

Hall, N.G., Posner, M.E., 2001. Generating experimental data for computational testing with machine scheduling applications. Operations Research (49), 854–865.

Hall, N.G., Sethi, S.P., Sriskandarajah, C., 1991. On the complexity of generalized due date scheduling problems. European Journal of Operational Research 51, 100–109.

Johnson, S.M., 1954. Optimal two- and three-stage production schedules with setup times included. Naval Research Logistics Quarterly 1, 61–67.

Kaplan, E.H., 1986. Relocation models for public housing redevelopment programs. Planning and Design 13, 5–19.

Kaplan, E.H., Amir, A., 1988. A fast feasibility test for relocation problems. European Journal of Operational Research 35, 201–205.

Kaplan, E.H., Berman, O., 1988. Orient Heights housing projects. Interfaces 18, 14–22.

Kobylański, P., Kuchta, D., 2007. A note on the paper by M. A. Al-Fawzan and M. Haouari about a bi-objective problem for robust resource-constrained project scheduling. International Journal of Production Economics 107, 496–501.

Kononov, A.V., Lin, B.M.T., 2006. On the relocation problems with multiple identical working crews. Discrete Optimization 3, 366–381.

Kononov, A.V., Lin, B.M.T., 2008. Minimizing the weighted completion time in the relocation problem (in submission).

Lin, B.M.T., Cheng, T.C.E., 1999. Minimizing the weighted number of tardy jobs and maximum tardiness in relocation problem with due date constraints. European Journal of Operational Research 116, 183–193.

Lin, B.M.T., Tseng, S.S., 1992. On the relocation problems of maximizing new capacities under a common due-date. International Journal of Systems Science 23, 1433–1448.

Lin, B.M.T., Tseng, S.S., 1993. Generating the best *K* sequences in the relocation problems. European Journal of Operational Research 69, 131–137.

Lock, D., 1996. Project Management, sixth ed. Gower Publishing, Hampshire, UK.

Mingozzi, A., Maniezzo, V., Ricciardelli, S., Bianco, L., 1998. An exact algorithm for the resource-constrained project scheduling problem based on a new mathematical formulation. Management Science 44, 714–729.

Mosheiov, G., Oron, D., 2004. A note on the SPT heuristics for solving scheduling problems with generalized due dates. Computers & Operations Research 31, 645–655.

Pesenti, R., Ukovich, W., 2003. Economic lot scheduling on multiple production lines with resource constraints. International Journal of Production Economics 81, 469–481.

PHRG, 1986. New Lives for Old Buildings: Revitalizing Public Housing Project. Public Housing Group, Department of Urban Studies and Planning, MIT, Cambridge, MA.

Sriskandarajah, C., 1990. A note on the generalized due dates scheduling problems. Naval Research Logistics 37, 587–597.

Xie, J.-X., 1997. Polynomial algorithms for single machine scheduling problems with financial constraints. Operations Research Letters 21, 39–42.