FOUNDATIONS OF COMPUTING AND DECISION SCIENCES
Vol. 39 (2014) No. 1

DOI: 10.2478/fcds-2014-0002 ISSN 0867-6356
e-ISSN 2300-3405

Minimizing Total Completion Time for Preemptive Scheduling with Release Dates and Deadline Constraints

Cheng He *, Hao Lin*, Yixun Lin †, Junmei Dou*

Abstract. It is known that the single machine preemptive scheduling problem of minimizing total completion time with release date and deadline constraints is NP-hard. Du and Leung solved some special cases by the generalized Baker's algorithm and the generalized Smith's algorithm in $O(n^2)$ time. In this paper we give an $O(n^2)$ algorithm for the special case where the processing times and deadlines are agreeable. Moreover, for the case where the processing times and deadlines are disagreeable, we present two properties which could enable us to reduce the range of the enumeration algorithm.

Keywords: Preemptive Scheduling; Release date; Deadline; Total completion time

1 Introduction

Suppose that we are given a set $\mathcal{J} = \{1, 2, \ldots, n\}$ of n independent jobs. These jobs have to be preemptively processed on a single machine that can process at most one job at a time. Each job j has a processing time p_j, release date r_j, and deadline d_j $(j = 1, \ldots, n)$. A *schedule* is for each job an allocation of one or more time intervals to the machine. A schedule is called a *feasible* schedule if each job is completely processed between its release date and deadline. Given a schedule σ, we designate the starting time of job j in σ by $S_j(\sigma)$ and we use $C_j(\sigma)$ to denote its completion time in σ. When there is no ambiguity, we abbreviate $S_j(\sigma)$ and $C_j(\sigma)$ to S_j and C_j. The objective of the problem considered is to find a feasible schedule with minimum total completion time $\Sigma_j C_j(\sigma)$. Following the three-field notation $\alpha|\beta|\gamma$ of Graham et al. [7] (where α specifies the *machine environment*, β specifies the *job character*, and γ denotes the *optimality criterion*), this model may be denoted by $1|r_j, pmtn, d_j|\Sigma C_j$.

*School of Science, Henan University of Technology, Zhengzhou, Henan 450052, China. Email: hech202@163.com

†Department of Mathematics, Zhengzhou University, Zhengzhou, Henan 450052, China.

It is known that the problem $1|r_j, pmtn, d_j|\Sigma C_j$ is NP-hard (see [4, 5]). Yet some special cases can be solved in $O(n \log n)$ time respectively: $1|r_j, pmtn|\Sigma C_j$ by Baker's rule [1] (at each decision point t given by a release time or a finishing time of some job, schedule a job j with the smallest remaining processing time), and $1|pmtn, d_j|\Sigma C_j$ (namely $1|d_j|\Sigma C_j$) by Smith's deadline rule [9] (at each decision point t given by the last job will be completed, schedule a job j with the largest processing time from among all the job k with $d_k \geq t$). If the job set contains no *obstruction* (an ordered triple of jobs (i, j, k) with $r_i, r_k < r_j < d_i < d_j, d_k$ and $p_j < p_i, p_k$ is called an obstruction), then the problem $1|r_j, pmtn, d_j|\Sigma C_j$ can be solved in $O(n^2)$ time by the generalized Baker's algorithm and the generalized Smith's algorithm, respectively [5]. In this paper we present an $O(n^2)$ algorithm for the special case where the processing times and deadlines are agreeable, i.e., $r_i \leq r_j \Rightarrow p_i \leq p_j$. Besides, we present two properties which enable us to reduce the range of the enumeration algorithm.

The paper is organized as follows. In Section 2 we state some preliminaries. Section 3 is dedicated to the polynomially solvable cases and a polynomial-time algorithm. In Section 4 we give a short summary. We shall follow the terminology and notation of [3].

2 Preliminaries

The feasibility of the problem $1|r_j, pmtn, d_j|\Sigma C_j$ can be determined by the EDD rule [8]: At each decision point t given by a release time or a finishing time of some job, schedule a job j with the smallest due date.

Generalized Baker's algorithm [5]: At each decision point t given by a release time or a finishing time of some job, schedule a job j, which can be completed prior to any of the other available jobs, with the smallest remaining processing time (break ties by choosing the one with the earlier possible deadline).

Generalized Smith's algorithm [5]: Choose the job that is completed last in an EDD schedule: At each decision point t, choose to process any job with deadline not later than t, other than the one with the largest processing time (break ties by choosing the one with the earlier possible release time).

Lemma 2.1 [5] If the set $\mathcal{J}$ of jobs contains no obstruction, then both the generalized Baker's algorithm and the generalized Smith's algorithm yield an optimal schedule of $1|r_j, pmtn, d_j|\Sigma C_j$ in $O(n^2)$ time.

For the problem $1|r_j, pmtn, prec|f_{\max}$, Baker et al. [2] have presented an $O(n^2)$ time algorithm. The basic idea of the algorithm is to divide the job set into blocks, which has close connection with our algorithm. Therefore we state the construction of block as the following procedure [10]:

Construction Blocks: All jobs are sequenced by the ERD (the earliest release date first) rule and scheduled with no preemption as early as possible after their release date. Each time a job i is scheduled at its release date, a new block containing job i is created. If a job is processed after its release date, then it is added to the last block created.

By the construction of blocks, there is no idle time between the jobs that are in

the same block. Note that there may be no idle time between the end of a block and the start of the next block if the release date of the first job of this block is equal to the end time of the previous block. Let $B_1, B_2, \cdots, B_l$ denote the blocks. They are clearly a partition of the job set $\mathcal{J}$. Each block B_i has a starting time $s_i = \min_{j \in B_i} r_j$ and an finishing time $f_i = s_i + \Sigma_{j \in B_i} p_j$.

In the next section we study the problem $1|r_j, pmtn, d_j|\Sigma C_j$, denoted PRD in short. It is known that this problem is NP-hard in general (see [4, 5]). We shall present an improved $O(n \log n)$ algorithm for the special case where the processing times and deadlines are agreeable. And the algorithm may solve the case where the processing times and deadlines are disagreeable in $O(n^2)$ time if some additional conditions are satisfied. Our main idea comes from the algorithm of Baker et al. for the problem $1|r_j, pmtn, prec|f_{\max}$, Baker's rule, and the EDD rule.

3 Polynomially solvable cases

First, we study the structural properties of optimal and feasible schedules. Suppose that all jobs are sequenced in the ERD rule ($r_1 \leq r_2 \leq \cdots \leq r_n$) and that $B_1, B_2, \cdots, B_l$ denote the blocks of Construction Blocks, s_i and f_i are the starting time and the finishing time of block B_i ($i = 1, 2, \cdots, l$) respectively.

Theorem 3.1 Suppose that the problem PRD is feasible. Let σ^* be its optimal schedule. Then each job j in block B_i must start at or after time s_i and end at or before time f_i, i.e., $s_i \leq S_j(\sigma^*) \leq C_j(\sigma^*) \leq f_i$ ($i = 1, 2, \cdots, l$) for any $j \in B_i$.

Proof. By the definition of blocks, we have $s_i \leq r_j \leq f_i$ for any $j \in B_i$. So $S_j(\sigma^*) \geq s_i$. Assume that the assertion does not hold. Then there exists a block B_i such that $C_j(\sigma^*) = \max\{C_k(\sigma^*) | k \in B_i\} > f_i$. Furthermore, assume that block B_i is the first such block in schedule σ^*. Since for any job $m \in B_k$ ($i+1 \leq k \leq l$), we have $r_m \geq s_k > f_i$. Thus in σ^* the jobs scheduled in the interval $[s_i, f_i]$ can only belong to B_i. By the definition of blocks, we have $f_i = s_i + \Sigma_{k \in B_i} p_k$. Hence there exists an idle interval $[s, t]$ in the interval $[s_i, f_i]$ by $C_j(\sigma^*) > f_i$ ($j \in B_i$). We claim that in this block there exists some job k, which is processed after this idle interval, with $r_k \leq s$. For otherwise let r be the minimum release date of all jobs processed after this idle interval, then $r > s$ and $[s, r]$ is an idle interval in the schedule produced by Construction Blocks, a contradiction. So we can move job k completely or partially into the idle interval $[s, t]$. Hence we get a new schedule σ with $\Sigma_j C_j(\sigma) < \Sigma_j C_j(\sigma^*)$, which contradicts the optimality of σ^*. This completes the proof.

Theorem 3.1 says that in any optimal schedule of PRD, the decomposition of blocks is unique (with respect to the jobs contained in each block), which coincides with the decomposition obtained by Construction Blocks. From this, we may treat each block separately. In other words, we may find an optimal schedule of problem PRD by rearranging the schedule inside each block B_i in order to reduce the objective $\Sigma_j C_j$.

Let B be such a block, and s and f the starting time and the finishing time of block B, respectively. Without loss of generality, suppose that all n jobs of $\mathcal{J} = \{1, 2, \ldots, n\}$ are in this block. Let $I_j = [r_j, d_j]$ be the processing interval (time window) of job j

$(1 \le j \le n)$. We have the following criterion of feasibility.

Theorem 3.2 The problem PRD has a feasible solution if and only if for any $r \in \{r_1, r_2, \cdots, r_n\}$ and $d \in \{d_1, d_2, \cdots, d_n\}$, $\Sigma_{I_j \subseteq [r,d]} p_j \le d - r$.

Proof. The necessity is obvious. We show the sufficiency below. Suppose that the above-mentioned condition holds. We can construct a feasible solution by running the EDD rule. In more detail, let $t = r_1$ at the beginning. Then, at each decision point t given by a release time or a finishing time of some job, let $A(t)$ be the set of the jobs (called *available jobs*) which have been released but not completely finished at time t. At this time t we schedule a job j in $A(t)$ with the smallest deadline d_j. Keeping this rule all the time, we obtain a schedule σ. We claim that it is feasible. In fact, if there is an idle interval in σ, then we get a contradiction to the assumption of only one block. Moreover, if σ is not feasible, then there is a job j with $C_j(\sigma) > d_j$. We may assume that j is the job with the smallest deadline and missing its deadline; when there is a tie, we choose the one with $C_j(\sigma)$ as large as possible. We perform the following algorithm:

Step 1: Let D_0 be the set of jobs which are processed in the interval $[r_j, C_j]$. Let $k_0 := j$ and $i := 1$.

Step 2: Let job $k_i \in D_{i-1}$ be the job with the minimum release date r_{k_i}. If $r_{k_i} = r_{k_{i-1}}$, stop; otherwise let D_i be the set of jobs which are processed in the interval $[r_{k_i}, r_{k_{i-1}}]$. Let $i := i + 1$, and go back to Step 2.

By the choice of k_i, we claim that $r_{k_i} \le r_{k_{i-1}}$. Otherwise $r_{k_i} > r_{k_{i-1}}$. Thus $S_{k_{i-1}} = r_{k_{i-1}}$ by the EDD rule. Then $k_{i-1} \in D_{i-1}$. Therefore $r_{k_i} \le r_{k_{i-1}}$, a contradiction. Hence the algorithm has at most $j-1$ rounds. Assume that the algorithm terminates in Step 2 when $i = l(< j)$. Then for any job $k \in D_m$ $(0 \le m \le l-1)$, we have $r_k \ge r_{k_{m+1}} \ge r_{k_l}$. On the other hand, since $k_m \in A(r_{k_m}), D_{m-1}$, we have $C_{k_m}(\sigma) > r_{k_{m-1}}$. Therefore, $d_k \le d_{k_m}$ by the EDD rule. Thus $d_{k_m} \le d_{k_{m-1}}$ by $k_m \in D_{m-1}$, i.e., $d_k \le d_{k_m} \le d_{k_{m-1}} \le \cdots \le d_{k_0} = d_j$ for any job $k \in D_m$ $(0 \le i \le l-1)$. Let $D := \cup_{0 \le i \le l-1} D_i$, which be the set of jobs which are processed in the interval $[r_{k_l}, C_j(\sigma)]$. Thus for any job $k \in D$, we have $r_k \ge r_{k_l}$ and $d_k \le d_{k_0} = d_j$. Let $r = r_{k_l}$ and $d = d_j$. Then for any job $k \in D$, we have $I_k \subseteq [r, d]$. Further by the assumption of choosing job j, we have $C_k(\sigma) < C_j(\sigma)$ and the EDD rule, we have $k \in D$ for any job k with $I_k \subseteq [r, d]$. So we have $\Sigma_{I_k \subseteq [r,d]} p_k = \Sigma_{k \in D} p_k = C_j(\sigma) - r_j > d - r$, contradicting the condition of the theorem. Therefore this schedule σ is indeed a feasible solution. The proof is complete.

Since the feasibility of PRD can be decided by the EDD rule, we assume that the problem PRD is always feasible in the sequel.

Lemma 3.3 Let σ^* be an optimal schedule of problem PRD. If $r_j \le r_k$, $p_j < p_k$ and $d_j \le d_k$, then $C_j(\sigma^*) \le S_k(\sigma^*)$.

Proof. Suppose that $C_j(\sigma^*) > S_k(\sigma^*)$. Since $r_j \le r_k$, we may construct a new schedule σ from σ^* as follows. First, take off the processing of job k and insert the part of job j which is processed after $S_k(\sigma^*)$ into the idle intervals as early as possible. Then, we put job k into these remaining idle intervals. Then by $p_j < p_k$ and $d_j \le d_k$, we have $C_j(\sigma) < \min\{C_j(\sigma^*), C_k(\sigma^*)\} \le d_j$, $C_k(\sigma) = \max\{C_j(\sigma^*), C_k(\sigma^*)\} \le d_k$ and $C_l(\sigma) = C_l(\sigma^*)$ $(l \ne k, j)$. So $\Sigma_i C_i(\sigma) < \Sigma_i C_i(\sigma^*)$, a contradiction. This completes the proof.

Definition A schedule σ is said to be *proper* if for each job j with finishing time t, j is scheduled in some of the idle periods produced by Construction Blocks for jobs in $\mathcal{J}(t) \setminus \{j\}$, where $\mathcal{J}(t)$ is the set of jobs finished before t.

Lemma 3.4 Any optimal schedule σ^* must be proper.

Proof. Let σ^* be an optimal schedule and let j be the job with the last finishing time $t = f$. We claim that the schedule $\sigma^* - j$, obtained by taking off the parts of job j from σ^*, is an optimal schedule for the jobs in $\mathcal{J} \setminus \{j\}$. In fact, let π be any feasible schedule with $C_j(\pi) = f$. Then $\Sigma_i C_i(\sigma^*) \leq \Sigma_i C_i(\pi)$, and so $\Sigma_{i\in(\mathcal{J}\setminus\{j\})} C_i(\sigma^*) \leq \Sigma_{i\in(\mathcal{J}\setminus\{j\})} C_i(\pi)$, as required. By applying Theorem 3.1 to the optimal schedule $\sigma^* - j$ of $\mathcal{J} \setminus \{j\}$, all jobs of $\mathcal{J} \setminus \{j\}$ are scheduled in the blocks produced by Construction Blocks. Hence j is scheduled in the idle periods produced by the procedure for $\mathcal{J}\setminus\{j\}$.

Suppose that $B_1, B_2, \cdots, B_l$ denote the blocks for $\mathcal{J} \setminus \{j\}$ obtained by Construction Blocks. Let $\sigma_i^* := (\sigma^* - j)|_{B_i}$ denote the subschedule of schedule $\sigma^* - j$ restricted to block B_i, where B_i $(i = 1, 2, \cdots, l)$ is treated as a job set. Then $\sigma^* - j = (\sigma_1^*, \sigma_2^*, \cdots, \sigma_l^*)$. We claim that each subschedule σ_i^* $(1 \leq i \leq l)$ is also optimal. Otherwise assume that σ_k^* $(1 \leq k \leq l)$ is not optimal for B_k, while σ_k is an optimal schedule for B_k. Then by Theorem 3.1 the starting time and the finishing time of σ_k are the same as those of σ_k^*, respectively. Let $\sigma = (\sigma_1^*, \cdots, \sigma_{k-1}^*, \sigma_k, \sigma_{k+1}^*, \cdots, \sigma_l^*)$, which is obtained from $\sigma^* - j$ by σ_k instead of σ_k^*. Then $\Sigma_{i\in(\mathcal{J}\setminus\{j\})} C_i(\sigma^* - j) - \Sigma_{i\in(\mathcal{J}\setminus\{j\})} C_i(\sigma) = \Sigma_{i\in B_k} C_i(\sigma^* - j) - \Sigma_{i\in B_k} C_i(\sigma) = \Sigma_{i\in B_k} C_i(\sigma_k^*) - \Sigma_{i\in B_k} C_i(\sigma_k) > 0$, i.e., $\Sigma_{i\in(\mathcal{J}\setminus\{j\})} C_i(\sigma^* - j) > \Sigma_{i\in(\mathcal{J}\setminus\{j\})} C_i(\sigma)$, which contradicts the optimality of $\sigma^* - j$. Let k be the job with the last finishing time in $\sigma^* - j$. Then in the similar way, we may show that $\sigma_l^* - k$ is optimal for $B_l\setminus\{k\}$, and k is scheduled in the idle periods produced by Construction Blocks for $B_l\setminus\{k\}$. Further, $\sigma^* - j - k = (\sigma_1^*, \sigma_2^*, \cdots, \sigma_l^* - k)$ is optimal for $\mathcal{J} \setminus \{j, k\}$ (it can be proved by contradiction), and the arrange of j and k satisfies the requirement of proper schedule. Thus by using repeatedly, for any finishing time t, we may show that σ^* is also optimal when it is restricted to the job set $\mathcal{J}(t)$. Therefore the assertion holds.

Here we illustrate an optimal schedule which is proper by an example. Figure 1 denotes the optimal schedule for $\mathcal{J} = \{1, 2, 3, 4, 5\}$. Figure 2 indicates that for $t = 15$, job 1 is scheduled in the idle periods produced by Construction Blocks for jobs in $\mathcal{J}(15) \setminus \{1\} = \{2, 3, 4, 5\}$. Figure 3 indicates that for $t = 12$, job 4 is scheduled in the idle periods produced by Construction Blocks for jobs in $\mathcal{J}(12) \setminus \{4\} = \{2, 3, 5\}$. Hence the optimal schedule is proper.

i	1	2	3	4	5
r_i	0	1	5	6	9
p_i	6	2	3	2	2
d_i	15	16	11	12	11

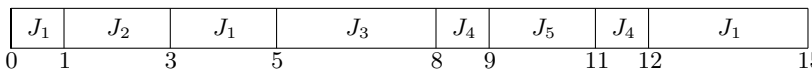

Fig. 1

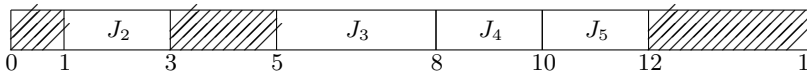

Fig. 2

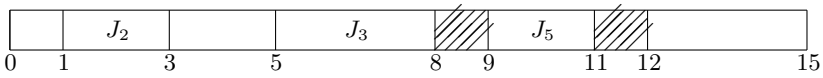

Fig. 3

Due to this property, we can find an optimal schedule from among the proper schedules. Recall that we assume that all jobs form a block B and f is the finishing time of B. The crucial point is to determine the job with the finishing time $t = f$.

Let $U = \{j \in \mathcal{J} | d_j \geq f\}$ be the set of the jobs that can be completed at time f. If $|U| > 1$, re-index the jobs in U so that $r_{i_1} \leq r_{i_2} \leq \cdots \leq r_{i_k}$, where $k = |U|$ denotes the cardinality of U, $1 \leq i_1 < i_2 < \cdots < i_k \leq n$ ($p_{i_j} \leq p_{i_{j+1}}$ if $r_{i_j} = r_{i_{j+1}}$). Let $p_{j_1} = \max\{p_j | j \in U\}$, $p_{j_2} = \max\{p_j | j \in U \text{ and } r_j > r_{j_1}\}, \ldots, p_{j_h} = \max\{p_j | j \in U \text{ and } r_j > r_{j_{h-1}}\} = p_{i_k}$, where $j_i (1 \leq i \leq k)$ is chosen as large as possible when there is a tie in the maximization. Let $U_0 = \{j_1, j_2, \ldots, j_h\}$.

Lemma 3.5 For any job $j \in U$, there exists a feasible schedule σ such that job j is completed at time f, i.e., $C_j(\sigma) = f$.

Proof. Assume that $C_j(\sigma) < f$ in any feasible schedule σ. Let π be the feasible schedule in which the completion time of job j is maximal, and $S_k(\pi) = C_j(\pi)$ for some job k. Then it follows that $S_k(\pi) = r_k$, for otherwise $r_k < S_k(\pi)$, we may interchange a part of job j and a part of k so that we get a feasible schedule in which job j is finished later, which contradicts the maximality of $C_j(\pi)$. Furthermore, we can deduce that no job l has $C_l(\pi) > C_j(\pi)$ and $r_l < r_k$, for otherwise we could make job j finished later by interchanging some parts of j and l. Hence all jobs l scheduled after r_k have $r_l \geq r_k$. Thus we get a contradiction to the assumption of only one block. The proof is complete.

Lemma 3.6 If release times and the deadlines of jobs are agreeable, i.e., $r_i < r_j \Rightarrow d_i \leq d_j$, let $f_0 = \Sigma_{1 \leq j < i_1} p_j$. If $r_j + p_j \geq f_0$ for every job j with $j \geq i_1$, then there exists an optimal schedule σ^* such that the jobs in U are scheduled starting at the time f_0 according to Baker's rule.

Proof. Since $r_i < r_j \Rightarrow d_i \leq d_j$, if $i \in U$ and $r_i < r_j$, then $j \in U$. Thus $U = \{i_1, i_1 + 1, i_1 + 2, \cdots, n\}$. By assumption, for every $j \in U$, we have $r_j + p_j \geq f_0$ and so the completion time of j is at least f_0. By Lemma 3.4 and Theorem 3.1, there exists an optimal schedule σ^* such that the jobs in U are scheduled starting at the time f_0. Now we consider the subproblem of PRD restricted on U under the condition that the jobs in $\mathcal{J} \setminus U$ have been optimally scheduled in the time interval $[s, f_0]$. Hence we can regard these jobs in $\mathcal{J} \setminus U$ as "fixed jobs" and the interval $[s, f_0]$ is a "forbidden interval". So this subproblem of U is a problem of $1|r_j, pmtn|\Sigma C_j$ with fixed jobs, which can be solved by Baker's rule.

Theorem 3.7 There exists an optimal schedule σ^* such that if $C_{j^*}(\sigma^*) = f$, then $j^* \in U_0$.

Proof. Suppose that σ^* is an optimal schedule and $C_{j^*}(\sigma^*) = f$, but $j^* \notin U_0$. By the definition of U, it is clear that $j^* \in U$. Note that for any $j \in U$, since $d_j \geq f$, the deadline of j has no restriction on j. So we can ignore the deadlines for jobs in U. As $j^* \notin U_0$, there is a job $k \in U_0$ such that $r_{j^*} \leq r_k$, $p_{j^*} \leq p_k$ and $d_k \geq f$. We distinguish two cases as follows.

(1) If $p_{j^*} < p_k$, then by the method of the proof of Lemma 3.3, we can drive a contradiction to the optimality of σ^* by interchanging the positions of j^* and k.

(2) If $p_{j^*} = p_k$ (this is possible because in the definition of U_0, j_i is chosen as large

as possible when there is a tie in the maximization), then we can construct another optimal schedule σ by interchanging the positions of j^* and k. Thus we obtain an optimal schedule finished by a job k in U_0. This completes the proof.

Therefore, the jobs in U_0 are candidates for scheduling the last job of an optimal schedule. We may call this U_0 the *candidate set*. If $|U_0| = 1$, then we can uniquely determine the last job and the problem can be reduced to a smaller instance. Otherwise the following properties enable us to reduce the size of U_0, which could be of benefit to the enumeration algorithm.

Proposition 3.8 Let $U_0 = \{j_1, j_2, \ldots, j_h\}$ be the candidate set.

(i) If the release times and the processing times are agreeable, i.e., $r_i \leq r_j \Rightarrow p_i \leq p_j$, then $U_0 = \{j_1\}$, i.e., $j_1 = i_k$.

(ii) For some $j_i \in U_0$ $(2 \leq i \leq h)$, If there exists a job j_m $(1 \leq m \leq i-1)$ such that $p_{j_m} - r_{j_i} + \min\{r_{j_m}, \Sigma_{d_k \leq r_{j_i}} p_k\} \geq p_{j_i}$, then we can reduce the size of U_0 by deleting the job j_i.

(iii) For $j_i \in U_0$ $(1 \leq i \leq h-1)$, we run Construction Blocks for the set $\mathcal{J} \setminus \{j_i\}$ and obtain the blocks $B_1, B_2, \cdots, B_x$ $(1 \leq x \leq |\mathcal{J}| - 1)$ with starting times $s_1, s_2, \cdots, s_x$ and finishing times $f_1, f_2, \cdots, f_x$ respectively. Assume that $j_t \in B_{k_t}$ $(i+1 \leq t \leq h,\ 1 \leq k_t \leq x)$. If there exists a m such that $f - f_{k_t} - \sum_{r_j > f_{k_t}} p_j > p_{j_t}$ for $i+1 \leq t \leq m-1$ and $f - f_{k_m} - \sum_{r_j > f_{k_m}} p_j \leq p_{j_m}$, then we can reduce the size of U_0 by deleting the jobs $j_i, j_{i+1}, \cdots, j_{m-1}$.

Proof. (i) is obvious. We next show (ii). By Theorem 3.7, let σ^* be an optimal schedule such that $C_{j_i}(\sigma^*) = f$ with $j_i \in U_0$. Let the length of j_k $(1 \leq k \leq i-1)$ placed before (after) r_{j_i} be $\widetilde{p}_{j_k}$ (p'_{j_k}). Then $\widetilde{p}_{j_k} \leq \max\{r_{j_i} - r_{j_k}, r_{j_i} - \Sigma_{d_k \leq r_{j_i}} p_k\}$. Therefore, $p'_{j_k} = p_{j_k} - \widetilde{p}_{j_k} \geq p_{j_k} - \max\{r_{j_i} - r_{j_k}, r_{j_i} - \Sigma_{d_k \leq r_{j_i}} p_k\} \geq p_{j_k} - r_{j_i} + \min\{r_{j_k}, \Sigma_{d_k \leq r_{j_i}} p_k\}$. If there exists a job j_m $(1 \leq m \leq i-1)$ such that $p_{j_m} - r_{j_i} + \min\{r_{j_m}, \Sigma_{d_k \leq r_{j_i}} p_k\} \geq p_{j_i}$, i.e., $p'_{j_m} \geq p_{j_i}$, We interchange p_{j_i} with the same length of p'_{j_m} and keep the remaining part of the schedule σ^* unchanged, so that another schedule σ^0 is obtained. Let $\alpha = C_{j_m}(\sigma^*)$. Then the completion time of j_m changes from α to f while the completion time of j_i changes from f to not later than α. Thus σ^0 is also an optimal schedule, but with j_m scheduled last. Therefore j_i can be deleted from U_0. This completes the proof.

Finally we show (iii). By Theorem 3.7, let σ^* be an optimal schedule such that $C_{j_i}(\sigma^*) = f$ with $j_i \in U_0$. By Lemma 3.4, σ^* is a proper schedule, namely j_i is scheduled in the idle periods produced by Construction Blocks for jobs in $\mathcal{J} \setminus \{j_i\}$. Then a part of j_i is placed after f_{k_t} with length $p'_{j_i}(t) = f - f_{k_t} - \sum_{r_j > f_{k_t}} p_j$. By the assumption, we have $p'_{j_i}(m) \leq p_{j_m}$. We interchange the part of $p'_{j_i}(m)$ with the same length of p_{j_m} and keep the remaining part of the schedule σ^* unchanged, so that another schedule σ^0 is obtained. Let $\alpha = C_{j_m}(\sigma^*)$. Then the completion time of j_m changes from α to f while the completion time of j_i changes from f to not later than α. Thus σ^0 is also an optimal schedule, but with j_m scheduled last. Therefore j_i can be deleted from U_0.

Similarly, for any $i+1 \leq t \leq m-1$, let π^* be an optimal schedule such that $C_{j_t}(\pi^*) = f$. Therefore, j_t is scheduled in the idle periods produced by Construction Blocks for jobs in $\mathcal{J} \setminus \{j_t\}$. Since $t \leq m-1$, we have $r_{j_t} < r_{j_m}$ and $p_{j_t} > p_{j_m}$. So $f_{k_t} < s_{k_m}$ (otherwise $k_t = k_m$ and $p'_{j_i}(t) = p'_{j_i}(m) \leq p_{j_m} < p_{j_t}$, which contradicts

the condition $p_{j_t} < p'_{j_i}(t)$). And after $S_{j_t}(\pi^*)$ starting time and finishing time of each block in σ^* are the same as those in π^* by the definition of block. Due to $S_{j_t}(\pi^*) \geq f_{k_t} - p_{j_t} + p'_{j_i}(t) > f_{k_t}$. If $S_{j_t}(\pi^*) \leq s_{k_m}$, then the part of j_t that is placed after s_{k_m} have length $p'_{j_i}(m)$. If $S_{j_t}(\pi^*) > s_{k_m}$, then the part of j_t that is placed after s_{k_m} have length not more than $p'_{j_i}(m)$. To sum up, in π^* the length of j_t is scheduled after $C_{j_m}(\pi^*)$ is not more than p_{j_i}. Thus in the above way we drive an optimal schedule with j_m scheduled last. This completes the proof.

By Proposition 3.8, we have the following recursive algorithm.

Algorithm PRD($\mathcal{J}$)

Step 0: For job set $\mathcal{J}$, compute the candidate set U_0.

Step 1: If $|U_0| = 1$, then set $U_0 := \{j^*\}$ and go to Step 3; if $|U_0| > 1$ and there exists jobs $j_i, j_m \in U_0$ that satisfy the condition in Proposition 3.8(ii), then set $U_0 := U_0 \backslash \{j_i\}$ and go back to Step 1; if $|U_0| > 1$ and there exists job $j_i \in U_0$ and m satisfy the condition in Proposition 3.8(iii), then set $U_0 := U_0 \backslash \{j_i, j_{i+1}, \cdots, j_{m-1}\}$ and go back to Step 1; otherwise choose any job j^* from U_0.

Step 2: Compute Construction blocks for the jobs in $\mathcal{J} \setminus \{j^*\}$. Suppose that we obtain blocks $B_1, B_2, \cdots, B_x$ $(1 \leq x \leq |\mathcal{J}| - 1)$ with starting times $s_1, s_2, \cdots, s_x$ and finishing times $f_1, f_2, \cdots, f_x$ respectively.

Step 3: Call Algorithm PRD(B_i) in each interval $[s_i, f_i]$ $(i = 1, 2, \cdots, x)$. Finally, schedule job j^* in the idle periods $[s, s_1], [f_1, s_2], \ldots, [f_x, f]$.

Example: Suppose that $\mathcal{J} = \{1, 2, 3, 4, 5, 6, 7\}$. The information of jobs is listed in table. First the problem is feasible and $t = 22$. Therefore $U = \{1, 3, 6\}$ and $U_0 = \{1, 3, 6\}$. Let $j = 1$ and compute Construction blocks for the jobs in $\mathcal{J} \setminus \{j\} = \{2, 3, 4, 5, 6, 7\}$ (see Fig. 1). Therefore $p'_1(3) = 4 > p_3$ and $p'_1(6) = 1 < p_6$. By Proposition 3.8 (iii), we have $U_0 = \{6\}$. Let σ^* be an optimal schedule. Then $C_6(\sigma^*) = 22$ and job 6 be scheduled in the idle periods produced by Construction Blocks for jobs in $\mathcal{J} \setminus \{6\} = \{1, 2, 3, 4, 5, 7\}$ (see Fig. 2), and job 7 be scheduled in the interval $[20, 21]$. When $t = 19$, $U = \{1, 3, 5\}$ and $U_0 = \{1, 5\}$. Due to $p'_1(5) = 1 < p_5$ (see Fig. 3), we update $U_0 := U_0 \backslash \{1\} = \{5\}$. Therefore $C_5(\sigma^*) = 19$ and job 6 be scheduled in the interval $[15, 19]$. When $t = 15$, $U = U_0 = \{1, 2, 3\}$. Due to $p_2 - (r_3 - r_2) = 3 = p_3$, we update $U_0 := U_0 \backslash \{3\} = \{1, 2\}$ by Proposition 3.8 (ii). Compute Construction blocks for the jobs in $\{2, 3, 4\}$ (see Fig. 4). So $p'_1(2) = 4 = p_2$ and we update $U_0 := U_0 \backslash \{1\} = \{2\}$. Therefore job 2 be scheduled in the interval $[11, 15]$. When $t = 11$, $U = U_0 = \{1, 3, 4\}$. Due to $p_1 - (r_3 - r_1) = 3 = p_3$, we update $U_0 := U_0 \backslash \{3\} = \{1, 4\}$. Compute Construction blocks for the jobs in $\{3, 4\}$ (see Fig. 5). So $p'_1(4) = 3 > p_4$. Compute Construction blocks for the jobs in $\{1, 3\}$ (see Fig. 6). In an optimal schedule, we cannot determinate whether job 1 or job 2 is scheduled in the last position with completion time 11 (we can schedule any one of jobs 1 and 2 by Algorithm PRD($\mathcal{J}$)), but by enumerating job 1 and job 2 are scheduled in the last position with completion time 11, respectively, we derive an optimal schedule (see Fig. 7).

i	1	2	3	4	5	6	7
r_i	0	2	3	5	14	16	20
p_i	6	4	3	2	4	2	1
d_i	22	15	22	11	19	22	21

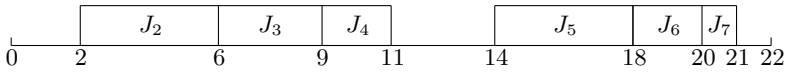

Fig. 1

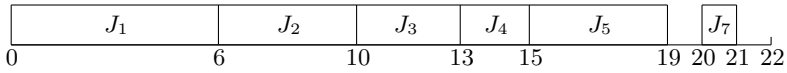

Fig. 2

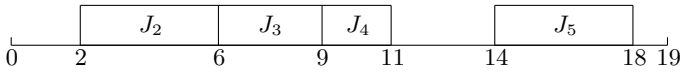

Fig. 3

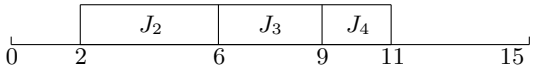

Fig. 4

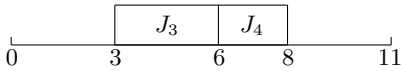

Fig. 5

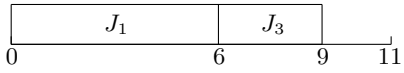

Fig. 6

Fig. 7

Theorem 3.9 When $|U_0| = 1$ in each stage, Algorithm PRD($\mathcal{J}$) solves the problem PRD in $O(n^3)$ time.

Proof. The correctness is based on Theorem 3.6 and Lemma 3.4. We next show the complexity. The algorithm consists of n stages in each of which a job is taken to schedule last. In each stage, the running time of Step 0 is $O(n)$ and the running time of Step 1 is $O(n^2)$, and the Construction blocks can be computed in $O(n)$ time. Note that the sorting of jobs in ERD order takes $O(n \log n)$ time at the beginning. Therefore the overall complexity is $O(n^3)$.

Corollary 3.10 When the release times and the processing times are agreeable, Algorithm PRD($\mathcal{J}$) solves the problem PRD in $O(n^2)$ time.

Therefore, we obtain a backward recursive algorithm, which is different from the forward generalized Baker's algorithm in [5]. The algorithm of [5] includes the case of (i) in Proposition 3.8, but it does not include cases of (ii) and (iii). This two cases could be of benefit to the enumeration algorithm, though they could be invalid in a bad situation.

4 Concluding remarks

In this paper we give an $O(n^2)$ algorithm for $1|r_j, pmtn, d_j|\Sigma C_j$ when the processing times and deadlines are agreeable. And for the case where the processing times and deadlines are disagreeable, we present two properties which could enable us to reduce the range of the enumeration algorithm. The problem $1|d_j|\Sigma C_j$ can be solved by Smith's deadline rule [9] in $O(n \log n)$. Du et al. [6] have showed that $P2|pmtn, r_j|\Sigma C_j$ is NP-hard in the ordinary sense. Du and Leung [5] submit the problem $P2|pmtn, d_j|\Sigma C_j$ to be open. It is worthwhile to settle this open question for our future work.

Acknowledgements

This work was supported by NSFC (grant no. 11201121, 11101383) and young backbone teachers of Henan colleges (2013GGJS-079) and CSC (201309895008). The authors are grateful to the referees for their helpful comments on improving the presentation of the paper.

References

[1] Baker, K.R. Introduction to Sequencing and Scheduling. Wiley, New York, 1974.

[2] Baker, K.R., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G. Preemptive scheduling of a single machine to minimize maximum cost subject to release dates and precedence constraints. Operations Research, 26: 111-120 (1983).

[3] Brucker, P. Scheduling Algorithms (third edition). Springer, Berlin, 2001.

[4] Blazewicz, J., Dror, M. Mathematical programming formulations for machine scheduling: A survey. European journal of Operational Research, 51: 283-300 (1991).

[5] Du, J., Leung, J.Y.T. Minimizing mean flow time with release time and deadline constraints. Journal of Algorithms, 14: 45-68 (1993).

[6] Du, J., Leung, J.Y.T., Young, G.H. Minimizing mean flow time with release time constraints. Theoretical Computer Science, 75: 347-355 (1990).

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

[8] Horn, W.A. Some simple scheduling algorithms. Naval Research Logistics Quarterly, 21: 177-18 (1974).

[9] Smith, W.E. Various optimizers for single-stage production. Naval Research Logistics Quarterly, 3: 59-66 (1956).

[10] Sourd, F. Preemptive scheduling with two minimax criteria. Annals of Operations Research, 107: 303-319 (2001).

Received January,2013