Tạp chí Tin học và Điều khiển học, T. 15, S. 1 (1999), 66–76

AN $O(n^2.\log n)$ ALGORITHM FOR A NONPREEMPTIVE SCHEDULE ON ONE MACHINE

TRINH NHAT TIEN

Abstract. An $O(n^2)$ algorithm for problem $1|r_j|\sum U_j$ in the case that release dates and due dates are similarly ordered (i.e., $r_j < r_k \Rightarrow d_j \leq d_k$) is provided by authors H. Kise, T. Ibaraki and H. Mine in [4]. In this paper, we would describe an $O(n^2.\log n)$ algorithm to determine a schedule of the same problem but furthermore in minimal processing time.

1. SOME BASIC CONCEPTS

It is known that the concepts "machine" and "job" are in the many materials, now we would remind some related concepts and notations.

The following data can be specified for each job u:

- r_u is a *release date*, on which u becomes available for processing;
- d_u is *due date*, by which u should ideally be completed;
- t_u is a *processing time* (or *length*) of u.

We assume that the above data are nonnegative integers and are regarded as *parameters* of job u.

For convenience we will also use a concept *"pre-job"* u; it is a pair (I_u, t_u), where $I_u = [r_u, d_u]$ is *its active area*. A pre-job u such that $t_u \leq d_u - r_u$ is said to be a *job*.

$R_u := [b_u, c_u]$ with: b_u is a *starting time*, c_u is a *completion time*, is said to be a *realization* of job u on machine.

A job u is said to be completed *on time* (or a *on-time job*) if $c_u \leq d_u$; otherwise job u is said to be *late*.

Definition 1. Let $I_i = [r_i, d_i]$ and $I_j = [r_j, d_j]$ be active areas of corresponding jobs i and j. Then the area I_i is said to be *ahead* of area I_j (or area I_j is *behind* area I_i) and denoted by $I_i \preceq I_j$ if and only if $r_i \leq r_j$ and $d_i \leq d_j$.

Similarly $I_i \prec I_j$ if and only if $I_i \preceq I_j$ and $I_i \not\equiv I_j$.

Similarly as in [3] we denote problem [T] by following:

[T]: $1|r_j|\sum U_j$, where $U_j = 0$ if $c_j \leq d_j$, $U_j = 1$ otherwise.

This problem means that the system has n jobs with different release dates r_j, they are available processing on one machine, we have to construct a nonpreemptive schedule with a minimal number of late jobs (i.e., a maximal number of jobs ideally completed on time). We know that the problem is strongly NP-hard, authors H. Kise, T. Ibaraki and H. Mine in [4] provided an $O(n^2)$ algorithm for problem [T] in the case that release dates and due dates are similarly ordered (i.e., $r_j < r_k \Rightarrow d_j \leq d_k$). We like to express this case by following:

[TK]: $1|I_1 \preceq I_2 \preceq \ldots \preceq I_n|\text{Max}\sum \overline{U}_j$, where $\overline{U}_j = 1$ if $c_j \leq d_j$, $\overline{U}_j = 0$ otherwise.

The problem is to construct a nonpreemptive schedule with a maximal number of jobs completed on time. Now we would pay attention to following special case:

[T1]: $1|I_1 \preceq I_2 \preceq \ldots \preceq I_n|\text{Max}\sum \overline{U}_j$ and $\text{Min}\sum \overline{U}_j.t_j$.

The problem is to construct a nonpreemptive schedule with a maximal number of jobs completed on time and furthermore with a minimal processing time.

We would present an $O(n^2\log n)$ algorithm to solve this problem.

Definition 2. $[B, C] := [\min r_i, \max d_i]$ is said to be an *active area* of the system [T], where B is a *release date* and C is a *due date* of it.

Let $P = \{u_1, u_2, ..., u_m\}$ be a subset of jobs on system [T] ($m \leq n$). Suppose that $S = \{R_{u_1}, R_{u_2}, ..., R_{u_m}\}$ is a set of realizations of corresponding jobs $u_1, u_2, ..., u_m$ such that $R_{u_i} \cap R_{u_j} = \emptyset$, $\forall i \neq j$, $i, j = 1, 2, \ldots, m$. Then S or $\bigcup_{i=1}^{m} R_{u_i}$ is said to be a *schedule* on the set P of the system [T], by abbreviation a *schedule* of the system. $\bigcup_{i=1}^{m} R_{u_i}$ is also said to be a *processing area* of schedule S.

A realization R_u of job u in schedule S is written by $R_u(S)$ or $u(S)$ and sometime only by $\{u\}$. In the paper we assume that $R_u(S) \subset I_u$.

We note some following parameters of the schedule S:

- $\sharp S := m$ is a *number* of realizations or a *number* of jobs;
- $t_S := \sum_{i=1}^{m} t_{u_i}$ is a *processing time* (or a *length*);
- $b_S := \min\{b_{u_i}\}$ is a *starting time*;
- $c_S := \max\{c_{u_i}\}$ is a *completion time*;
- $[b_S, c_S]$ is an *active area* of schedule S.

The set of jobs $P = \{u_1, u_2, ..., u_m\}$, which can create any schedule, is said to be a *scheduled set*. Sometime for schedule S having scheduled set $\{u_q, u_2, ..., u_m\}$, we also write

$S = \{u_1, u_2, ..., u_m\}$.

Let $u = (I_u, t_u)$ be a job, $[X, Y]$ be a time area. We define a pre-job $v = (I_v, t_v)$ on $[X, Y]$ such as $I_v = I_u \cap [X, Y]$, $t_v = t_u$ and we write $v = u \uparrow [X, Y]$.

For a set of jobs $P = \{u_1, u_2, ..., u_m\}$, we denote a set of pre-jobs on $[X, Y]$ by $P \uparrow [X, Y] = \{u_1 \uparrow [X, Y], u_2 \uparrow [X, Y], ..., u_m \uparrow [X, Y]\}$.

We say that a schedule S is *in the area* $[X, Y]$ if its active area $[b_S, c_S] \subseteq [X, Y]$.

Note that we define *schedule* only on the set of *jobs*, not on a set of pre-jobs.

2. *R*-OPTIMAL SCHEDULE AND *L*-OPTIMAL SCHEDULE

First we define some concepts related to problems [TK] and [T1].

Definition 3. Let $R_i = [b_i, c_i]$ and $R_j = [b_j, c_j]$ be realizations of corresponding jobs i and j. We say that R_i is *ahead* of R_j (or R_j is *behind* R_i) and write $R_i \preceq R_j$ if and only if they satisfy one of two the following conditions:

1. $i \equiv j$ and $b_i \leq b_j$;
2. $i \not\equiv j$ and $I_i \preceq I_j$.

Similarly we write $R_i \prec R_j$.

Let $P = \{u_1, u_2, ..., u_m\}$ and $Q = \{v_1, v_2, ..., v_m\}$ be schedules with the same number of jobs. We say that P is *ahead* of Q (or Q is *behind* P) and write $P \preceq Q$ if and only if $R_{u_i} \preceq R_{v_i}$, $\forall i = 1, 2, ..., m$. Similar we write $P \prec Q$.

Definition 4. A schedule $S = \{u_1, u_2, ..., u_m\}$ is said to be *R-schedule* in $[X, Y]$ if it is in the area and realizations $[b_{u_i}, c_{u_i}]$ have following forms:

$c_{u_m} = \min\{d_{u_m}, Y\}$; $b_{u_m} = c_{u_m} - t_{u_m}$;

$c_{u_i} = \min\{d_{u_i}, b_{u_{i+1}}\}$; $b_{u_i} = c_{u_i} - t_{u_i}$, $\forall i = m-1, m-2, ..., 2, 1$.

Let $P = \{u_1, u_2, ..., u_p\}$ and $Q = \{v_1, v_2, ..., v_q\}$ be R-schedules in $[X, Y]$. We say that P is *R - better than* Q and denote by $P \succ_r Q$ if and only if one of the following conditions satisfied:

(r_1) $p > q$ (i.e., P has the number of jobs more than Q);

(r_2) $p = q$ and $t_P < t_Q$ (i.e., P has the processing time less than Q);

(r_3) $p = q$ and $t_P = t_Q$ and $b_P > b_Q$ (i.e., P has the starting time later than Q);

(r_4) $p = q$ and $t_P = t_Q$ and $b_P = b_Q$ and $Q \preceq P$ (i.e., P is behind Q);

With $i = 1, 2, 3, 4$, if $P \succ_r Q$ in the sense (r_i), we write $P \succ_{r_i} Q$.

Definition 5. We say that schedule S is *R-best* if and only if it is R-schedule having:

(ro_1) a maximal number of jobs completed on time;

(ro_2) a minimal processing time t_S from schedules satisfying above condition;

(ro_3) a latest starting time b_S from schedules satisfying above condition;

and it is

(ro_4) *behind* all schedules satisfying above condition.

In the case that the R-best schedule has only 1 job (i.e., 1 realization), we call it R-best *realization.*

Definition 6. Let $P = \{u_1, u_2, ..., u_p\}$ be R-schedule in $[X_P, Y_P]$ and $Q = \{v_1, v_2, ..., v_q\}$ be R-schedule in $[X_Q, Y_Q]$, where $Y_P \leq X_Q$, $I_{u_p} \preceq I_{v_1}$.

We define an operation, which is called *R-connection* and denoted by $P \oplus_r Q$, to connect P to Q. The result of the operation is schedule S, having following realizations:

$[b_{v_i}(S), c_{v_i}(S)] = [b_{v_i}(Q), c_{v_i}(Q)]$, $\forall i = q, q-1, ..., 1$;

$[b_{u_p}(S), c_{u_p}(S)]$ where $c_{u_p}(S) = \min\{d_{u_p}, b_Q\}$; $b_{u_p}(S) = c_{u_p}(S) - t_{u_p}$;

$[b_{u_i}(S), c_{u_i}(S)]$, where $c_{u_i}(S) = \min\{d_{u_i}, b_{u_{i+1}}(S)\}$; $b_{u_i}(S) = c_{u_i}(S) - t_{u_i}$, $\forall i = p-1, p-2, ..., 1$.

Proposition 1.

1. Let $P = \{u_1, u_2, ..., u_p\}$ be R-schedule in $[X_P, Y_P]$, $Q = \{v_1, v_2, ..., v_q\}$ and $Q' = \{v'_1, v'_2, ..., v'_{q'}\}$ be R-schedules in $[X_Q, Y_Q]$, where $Y_P \leq X_Q$. Let $S := P \oplus_r Q$ and $S' := P \oplus_r Q'$, then have following conclusion: With $i = 1, 2, 3, 4$, if $Q \succ_{r_i} Q'$ then $S \succ_{r_i} S'$.

2. Let $P = \{u_1, u_2, ..., u_p\}$, $P' = \{u'_1, u'_2, ..., u'_{p'}\}$ be R-schedules in $[X_P, Y_P]$, $Q = \{v_1, v_2, ..., v_q\}$ be R-schedule in $[X_Q, Y_Q]$, where $Y_P \leq X_Q$. Let $S := P \oplus_r Q$ and $S' := P' \oplus_r Q$, then we have following conclusion: With $i = 1, 2, 3$, if $P \succ_{r_i} P'$ then $S \succ_{r_i} S'$.

Definition 7. A schedule $S = \{u_1, u_2, ..., u_m\}$ is said to be *L-schedule* in $[X, Y]$ if it is in the area and realization $[b_{u_i}, c_{u_i}]$ have following forms:

$b_{u_1} = \max\{X, r_{u_1}\}$; $c_{u_1} = b_{u_1} + t_{u_1}$; $b_{u_i} = \max\{c_{u_{i-1}}, r_{u_i}\}$; $c_{u_i} = b_{u_i} + t_{u_i}$, $\forall i = 2, 3, ..., m$.

Let $P = \{u_1, u_2, ..., u_p\}$ and $Q = \{v_1, v_2, ..., v_q\}$ be L-schedules in $[X, Y]$. We say that P is *L-better than* Q and denote by $P \succ_l Q$ if and only if one of the following conditions satisfied:

(l_1) $p > q$ (i.e., P has the number of jobs more than Q);

(l_2) $p = q$ and $c_P < c_Q$ (i.e., P has the completion time earlier than Q);

(l_3) $p = q$ and $c_P = c_Q$ and $c_{u_i}(P) \leq c_{v_i}(Q)$, $\forall i = 1, 2, ..., p-1$;

(l_4) $p = q$ and $c_{u_i}(P) = c_{v_i}(Q)$, $\forall i = 1, 2, ..., p$ and $P \preceq Q$ (i.e., P is ahead of Q);

With $i = 1, 2, 3, 4$ if $P \succ_l Q$ in the sense (l_i), we write $P \succ_{l_i} Q$.

Definition 8. We say that schedule S is *L-best* if and only if it is L-schedule having:

(lo_1) a maximal number of jobs completed on time;

(lo_2) a earlest completion time c_S from schedules satisfying above condition;

(lo_3) a earlest completion time of realizations from schedules satisfying above condition;

and it is (lo_4) *ahead* of all schedules satisfying above condition.

Definition 9. Let $P = \{u_1, u_2, ..., u_p\}$ be L-schedule in $[X_P, Y_P]$ and $Q = \{v_1, v_2, ..., v_q\}$ be L-schedule in $[X_Q, Y_Q]$, where $Y_P \leq X_Q$, $I_{u_p} \preceq I_{v_1}$.

We define an operation, which is called *L-connection* and denoted by $P \oplus_l Q$, to connect Q to P. The result of the operation is schedule S, having following realizations:

$[b_{u_i}(S), c_{u_i}(S)] = [b_{u_i}(P), c_{u_i}(P)]$, $\forall i = 1, 2, ..., p$;

$[b_{v_1}(S), c_{v_1}(S)]$, where $b_{v_1}(S) = \max\{c_P, r_{v_1}\}$; $c_{v_1}(S) = b_{v_1}(S) + t_{v_1}$;

$[b_{v_i}(S), c_{v_i}(S)]$, where $b_{v_i}(S) = \max\{c_{v_{i-1}}(S), r_{v_i}\}$; $c_{v_i}(S) = b_{v_i}(S) + t_{v_i}$, $\forall i = 2, 3, ..., q$.

Proposition 2.

1. Let $P = \{u_1, u_2, ..., u_p\}$ be L-schedule in $[X_P, Y_P]$, $Q = \{v_1, v_2, ..., v_q\}$ and $Q' = \{v'_1, v'_2, ..., v'_{q'}\}$ be L-schedules in $[X_Q, Y_Q]$, where $Y_P \leq X_Q$. Let $S := P \oplus_l Q$ and $S' := P \oplus_l Q'$, then we have following conclusion: if $Q \succ_{l_1} Q'$ then $S \succ_{l_1} S'$.

With $i = 2, 3, 4$, if $Q \succ_{l_i} Q'$ and ($X_Q \leq r_{v'_1}$ or $X_Q = c_P$) then $S \succ_{l_i} S'$.

2. Let $P = \{u_1, u_2, ..., u_p\}$, $P' = \{u'_1, u'_2, ..., u'_{p'}\}$ be L-schedule in $[X_P, Y_P]$, $Q = \{v_1, v_2, ..., v_q\}$ be L-schedule in $[X_Q, Y_Q]$, where $Y_P \leq X_Q$. Let $S := P \oplus_l Q$ and $S' := P' \oplus_l Q$, then we have following conclusion: With $i = 1, 2, 3, 4$ if $P \succ_{l_i} P'$ then $S \succ_{l_i} S'$.

Definition 10. We say that schedule S is *p-optimal* if and only if it is R-schedule having:

(o_1) just p jobs completed on time;

(o_2) a minimal processing time t_S from schedules satisfying above condition;

(o_3) a latest starting time b_S from schedules satisfying above condition;

and it is (o_4) *behind* all schedules satisfying above condition.

Conclusion. According to definitions 5, 10, the *p-optimal* schedule is just *R-best* schedule having p jobs completed on time.

We call the schedule constructed by authors Kise, Ibaraki and Mine [2] *K-schedule*. We call their algorithm *K-algorithm*. We assume that this schedule has just m jobs, it is the maximal number of jobs completed on time. We would problem solve [T1], by above reasons we will determine the *m-optimal schedule*.

3. POSITION OF m-OPTIMAL SCHEDULE WITH K-SCHEDULE

Proposition 3. *K-schedule is just L-best schedule.*

Conclusion. Let U be a set of n jobs on system [TK] or [T1], $[B, C]$ be an active area of the system, let $K = \{x_1, x_2, ..., x_m\}$ be K-schedule, $[b_i(K), c_i(K)] := [b_{x_i}(K), c_{x_i}(K)]$ be the realization $x_i(K)$.

We write following notations:

$$\begin{aligned} U_0 &= \{\text{jobs } u \,\|\, I_u \prec I_{x_1}\}, \\ U_i &= \{\text{jobs } u \,\|\, I_{x_i} \preceq I_u \prec I_{x_{i+1}}\} \text{ for } i = 1, 2, \ldots, m-1, \\ U_m &= \{\text{jobs } u \,\|\, I_{x_m} \preceq I_u\}, \end{aligned} \tag{1}$$

i.e., we can put in order n jobs from U to $m+1$ following subsets:

$U_0 = \{u_0^1, u_0^2, \ldots, u_0^{n_0}\}$;

$U_1 = \{x_1, u_1^2, \ldots, u_1^{n_1}\}$, where $x_1 \doteq u_1^1$;

...

$U_i = \{x_i, u_i^2, \ldots, u_i^{n_i}\}$, where $x_i \doteq u_i^1$;

...

$U_m = \{x_m, u_m^2, \ldots, u_m^{n_m}\}$, where $x_m \doteq u_m^1$.

Where $U = U_0 \cup U_1 \cup \ldots \cup U_m$, $n = n_0 + n_1 + \cdots + n_m$.

We write $U_i^* = U_i \cup U_{i+1} \cup \ldots \cup U_m$, $n_i^* = n_i + n_{i+1} + \cdots + n_m$.

Lemma 1. *Let $K = \{x_1, x_2, ..., x_m\}$ be K-schedule, $W = \{w_1, w_2, ..., w_m\}$ be m-optimal schedule. Let sets of jobs U_i, and the notations be such as (1). We have following result:*

For $k = 1, 2, ..., m-1$, if U_k contains $w_j \in W$ such that

$$c_{x_k}(K) \leq c_{w_j}(W) \tag{2}$$

and

$$k < j + 1 \tag{3}$$

then

$$U_k \text{ does not contain a next job } w_{j+1} \in W; \tag{4}$$

$$U_0 \text{ does not contain any job } w \in W. \tag{5}$$

Proof. By contradiction, we suppose that there is U_k satisfying (2), (3), but it contains a subset S of jobs from W, where

$$S = \{w_{j+1}, w_{j+2}, \dots, w_{j+e}\} \neq \emptyset, \quad s := j + e, \quad e \geq 1. \tag{6}$$

Let p $(k+1 \leq p \leq m)$ be smallest such that $x_p \neq$ every job of W. (7)

(Such p always exist since by (3) we see that *behind* job w_{j+1}, a number of jobs from schedule K is more than from schedule W). That result is contradictory with the property of K-schedule K.

Lemma 2. *Let $K = \{x_1, x_2, \dots, x_m\}$ be K-schedule, $W = \{w_1, w_2, \dots, w_m\}$ be m-optimal schedule. Let sets of jobs U_i and the notations be such as (1). We have following result:*
For $k = 1, 2, \dots, m-1$, if $U_k \cup \{x_{k+1}\}$ contains $w_j \in W$ such that

$$b_{w_j}(W) < c_{x_k}(K) \tag{8}$$

and

$$j - 2 < k \tag{9}$$

then

$$U_k - \{x_k\} \text{ does not contain a preceding job } w_{j-1} \in W; \tag{10}$$

$$U_m \text{ does not contain a job } w \in W \text{ such that } c_w(W) < c_{x_m}(K). \tag{11}$$

Lemma 3. *Let $K = \{x_1, x_2, \dots, x_m\}$ be K-schedule, $W = \{w_1, w_2, \dots, w_m\}$ be m-optimal schedule. Let sets of jobs U_i and the notations be such as (1). We have following result:*

$$w_i \in U_i \cup \{x_{i+1}\}, \quad \forall i = 1, 2, \dots, m-1 \text{ and } w_m \in U_m. \tag{12}$$

4. m^* - OPTIMAL SCHEDULE

From results of Lemma 3 we define concept "m^*-optimal schedule" related to the m-optimal schedule.

Definition 11. Let $K = \{x_1, x_2, \dots, x_m\}$ be K-schedule, $[X, Y]$ be a time area, let sets of jobs U_i and the notations be such as (1).

For $d = m, m-1, \dots, 2, 1$, we say that S is $(m-d+1)^*$-*optimal schedule* on set of jobs $U_d^* \uparrow [X, Y]$ if and only if it is $(m-d+1)$-*optimal* and has following form:

$$S := \{w_d, w_{d+1}, \dots, w_m\},$$

where

$$w_i \in U_i \cup \{x_{i+1}\}, \quad \forall i = d, d+1, \dots, m-1 \text{ and } w_m \in U_m. \tag{13}$$

We see that m^*-optimal schedule on set of jobs $U_1^* \uparrow [B, C]$ is just m-optimal and so that we will determine the such schedule.

Lemma 4. *Let $K = \{x_1, x_2, \dots, x_m\}$ be K-schedule, let sets of jobs U_i and the notations be such as (1). We have following result:*

1) For $d = m, m-1, \dots, 2, 1$, if S is $(m-d+1)^*$-optimal schedule on set of jobs $U_d^* \uparrow [B, C]$ then $b_{x_d}(K) \leq b_S$.

2) If W is m^*-optimal schedule then $b_K \leq b_W$.

We can prove result 1) by contradiction and by using definitions of $(m-d+1)^*$-optimal schedule and K-schedule. Result 2) is the corollary of result 1).

Definition 12. Let $K = \{x_1, x_2, \dots, x_m\}$ be K-schedule, let sets of jobs U_i and the notations be such as (1). For $s = 1, 2, \dots, m$ we define following concepts:

$\mathcal{W}_d := \{W_d^1, W_d^2, \dots, W_d^p\}$ is said to be a *full set of* $(m-d+1)^*$-*optimal* schedules on the set U_d^* if and it it satisfies following conditions:

$$W_d^1 \text{ is } (m-d+1)^*\text{-optimal schedule on } U_d^* \uparrow [B, C] \tag{14}$$

and $$W_d^i \text{ is } (m-d+1)^*\text{-optimal schedule on } U_d^* \uparrow [b_{W_d^{i-1}} + 1, C], \tag{15}$$

where $b_{W_d^{i-1}}$ is a starting time of schedule W_d^{i-1}, $\forall i = 2, 3, ..., p$.

$\mathcal{V}_d := \{V_d^1, V_d^2, ..., V_d^q\}$ is said to be a *infull set* of $(m-d+1)^*$-optimal schedules on the set U_d^* if and only if it satisfies following conditions:

$$V_d^1 \text{ is } (m-d+1)^*\text{-optimal schedule on } (U_d^* - \{x_d\}) \uparrow [B, C] \tag{16}$$

and $$V_d^i \text{ is } (m-d+1)^*\text{-optimal schedule on } (U_d^* - \{x_d\}) \uparrow [b_{V_d^{i-1}} + 1, C], \tag{17}$$

where $b_{V_d^{i-1}}$ is is a starting time of schedule V_d^{i-1}, $\forall i = 2, 3, ..., q$.

$\mathcal{F}_d := \{\mathcal{W}_d, \mathcal{V}_d)$ is said to be a *pair of 2 sets* of $(m-d+1)^*$-optimal schedules on the set U_d^*.

Let $\mathcal{R} = \{S^1, S^2, ..., S^p\}$ be a set of R-schedules with the same number of jobs. We say that the set has R-*order* if $t_{S^i} < t_{S^{i+1}}$; $b_{S^i} < b_{S^{i+1}}$; $S^i \preceq S^{i+1}$, $\forall i = 1, 2, ..., p-1$.

We note that by Lemma 4 there is $b_{x_d}(K) \leq b_{W_d^1}$, therefore to determine W_d, we consider only set $U_d^* \uparrow [b_{x_d}(K), C]$.

Proposition 4. *The defined sets $\mathcal{W}_d$ and $\mathcal{V}_d$ have R-order.*

Lemma 5. *For $d = m-1, m-2, ..., 2, 1$, let $\mathcal{F}_{d+1} = (\mathcal{W}_{d+1}, \mathcal{V}_{d+1})$ be a pair of 2 sets of $(m-d)^*$-optimal schedule on the set U_{d+1}^*.*

Suppose that $\mathcal{F}_d = (\mathcal{W}_d, \mathcal{V}_d)$ be a pair of 2 sets of $(m-d+1)^*$-optimal schedules on the set U_d^*, then there is following conclusion:

Every schedule $W_d^i \in \mathcal{W}_d$ (or $V_d^i \in \mathcal{V}_d$) has to contain either schedule $W_{d+1}^w \in \mathcal{W}_{d+1}$ or schedule $V_{d+1}^w \in \mathcal{V}_{d+1}$ as its "ending part" with $(m-d)$ jobs.

Corollary. *The m^*-optimal schedule has to contain either schedule $W_2^w \in \mathcal{W}_2$ or $V_2^v \in \mathcal{V}_2$ as its "ending part" with $(m-1)$ jobs.*

5. ALGORITHM DETERMINING s^*-OPTIMAL SCHEDULE

5.1. Main idea of algorithm

By the above results, our algorithm will be constructed by following steps:

- First determine K-schedule $K = \{x_1, x_2, ..., x_m\}$ by K-algorithm with time $O(n^2)$ or by Lawler's algorithm with the time $O(n.\log n)$ (see [5]).

- Lemma 3 and Lemma 4 determine the position of the m-optimal schedule W in comparise with K-schedule. Here if $W = \{w_1, w_2, .., w_m\}$ then

$$w_i \in U_i \cup \{x_{i+1}\}, \quad \forall i = 1, 2, ..., m-1, \quad w_m \in U_m \text{ and } b_K \leq b_W . \tag{18}$$

To create W, we construct the set $\mathcal{W}_1$ of all schedules, which could become W, these such schedules equally have property (12). By Lemma 5, the set $\mathcal{W}_1$ will be created recursively by 3 following algorithms:

A. Algorithm BASE will create the basic pair of sets $\mathcal{F}_m = (\mathcal{W}_m, \mathcal{V}_m)$, i.e., the pair of 2 sets 1^*-optimal schedules on the set $U_m^* = U_m$; one from these schedules will become "an ending part" $\{w_m\}$ of optimal schedule W.

B. Procedure STEPD will form the well-known pair of 2 sets $\mathcal{F}_{d+1}$ of $(m-d)^*$-optimal schedule on the set U_{d+1}^* determine a pair of 2 sets $\mathcal{F}_d$ of $(m-d+1)^*$-optimal schedules on the set U_d^*; one from these schedules will become "an ending part" $\{w_d, w_{d+1}, ..., w_m\}$ of the optimal schedule W.

C. Algorithm USE-STEPD will form the basic pair of sets $\mathcal{F}_m$ apply $(m-1)$ times the procedure STEPD, we will obtain successively pair of 2 sets $\mathcal{F}_{m-1}, \mathcal{F}_{m-2}, ..., \mathcal{F}_2, \mathcal{F}_1$, where $\mathcal{F}_1 = (\mathcal{W}_1, \mathcal{V}_1)$. Suppose $\mathcal{W}_1 = \{W^1, W^2, ..., W^p\}$ then W^1 is just the desirable m^*-optimal schedule.

5.2. Some auxiliary procedures

5.2.1. Procedure finds R-best realisation on the set of jobs

Let a set of jobs $U = \{x^1, x^2, ..., x^k\}$, by Definition 5 we can create a procedure to find R-best schedule with 1 job (i.e., R-best realization) $\{x\}$ on U and write:

$$\{x\} := RB\text{-}JOB(\{x^1, x^2, ..., x^k\}).$$

In the case the set is restricted by the time area $[X, Y]$, we write:

$$\{x\} := RB\text{-}JOB(\{x^1, x^2, ..., x^k\} \uparrow [X, Y]).$$

The processing time of this procedure is $O(k)$.

According to Definition 2, may be $\{x^1, x^2, ..., x^k\} \uparrow [X, Y]$ is not a set jobs, therefore there is not such $\{x\}$.

5.2.2. Procedure connects a set of jobs to a schedule: $JOB\text{-}SCHED(U, b, S; Z, K, p)$

Input:

- $U = \{x^1, x^2, ..., x^k\}$ is the set of jobs such as $I_{x^1} \preceq I_{x^2} \preceq ... \preceq I_{x^k}$.
- b is a starting time of the area time;
- S is R-schedule on the set of jobs $\{y^1, y^2, ..., y^h\}$ such as $I_{x^k} \prec I_{y^1} \preceq I_{y^2} \preceq ... \preceq I_{y^h}$.

Output:

- $Z = \{Z_1, Z_2, ..., Z_p\}$ is a set of R-schedules, every schedule Z_i is created by R-connection of R-best realization on U to S;
- $K = \{k_1, k_2, ..., k_p\}$ is a set index corresponding to Z, $p = \sharp Z$.

Method: The algorithm applies the procedure $RB\text{-}JOB$ to determine a R-best realization on U, if there is the such realization then connects it to S.

Algorithm:

Begin
  $i := 1$;
  if there is $\{x^{k_1}\} := RB\text{-}JOB(\{x^1, x^2, ..., x^k\} \uparrow [b, b_S])$
  then $\{Z_1 := \{x^{k_1}\} \oplus_r S$ and $p := 1\}$ else { put $Z := \emptyset$ and $p := 0\}$;
  Repeat
    $i := i + 1$;
    if there is $\{x^{k_i}\} := RB\text{-}JOB(\{x^{k_{i-1}+1}, x^{k_{i-1}+2}, ..., x^k\} \uparrow [b_{Z_{i-1}} + 1, b_S])$
    then $\{Z_i := \{x^{k_i}\} \oplus_r S$ and $p := i\}$
  Until $p < i$ (i.e., there is not $\{x^{k_i}\}$);
End.

Proposition 5. *Let r_{k_i}, d_{k_i}, t_{k_i} be parameters of job x^{k_i}, $i = 1, 2, ..., p$. The procedure JOB-SCHED gives following conclusions:*

1. $t_{k_1} < t_{k_2} \cdots < t_{k_p}$; $t_{k_1} \le t_{x^j}$, $\forall x^j \in U \uparrow [b, b_S]$; $t_{k_i} \le t_{x^j}$, $\forall x^j \in U \uparrow [b_{Z_{i-1}} + 1, b_S]$, $\forall i = 2, 3, ..., p$.
2. $\{x^{k_i}\}$ *is R-best realization job* $x^{k_i} \uparrow [b, d_{k_i}]$, $\forall i = 1, 2, ..., p - 1$.
3. $\{x^1, x^2, ..., x^{k_i}\} \uparrow [b_{Z_i} + 1, b_S]$ *is not a set of jobs*, $\forall i = 1, 2, ..., p - 1$.
4. $Z = \{Z_1, Z_2, ..., Z_p\}$ *has R-order.*
5. *The processing time of the procedure is $O(k^2)$, where $k = \sharp U$.*

For simple we presented the procedure $JOB\text{-}SCHED$ by the such method. Practically we use the fast algorithm (for instance Quicksort or Heapsort) to sort realizations on U according to R-order, then connect the realization to S. This method needs only the time $O(k.\log k)$.

Proposition 6. *For $i = 1, 2, ..., p$, let S in procedure JOB-SCHED be d-optimal schedule, then Z_i is $(d+1)$-optimal schedule on corresponding set of jobs and contains S as its "ending part".*

5.2.3. Procedure connects a set of jobs to a set of schedules: $JOB\text{-}SCHEDULES(U, b, \mathcal{R}; Z, p)$

Input:

- $U = \{x^1, x^2, ..., x^k\}$ is the set of jobs such as $I_{x^1} \preceq I_{x^2} \preceq \cdots \preceq I_{x^k}$;
- b is a starting time of the area time;
- $\mathcal{R} = \{S^1, S^2, ..., S^r\}$ is the set of R-schedules having the same number of jobs on a set of jobs $\{y^1, y^2, ..., y^h\}$ such as $I_{x^k} \prec I_{y^1} \preceq I_{y^2} \preceq \ldots \preceq I_{y^h}$; where $r = \sharp\mathcal{R}$.

Output:

$Z = \{Z_1, Z_2, ..., Z_p\}$ is a set of R-schedules, every schedule Z_i is created by R-connection of R-best realization on U to $S \in \mathcal{R}$, where $p = \sharp Z$.

Method: The algorithm applies procedure *JOB-SCHED* r times.

Algorithm:

Begin

$JOB\text{-}SCHED(\{x^1, x^2, ..., x^k\}, b, S^1; \{Z_1^1, Z_2^1, ..., Z_{p_1}^1\}, \{k_1^1, k_2^1, ..., k_{p_1}^1\}, p_1)$;

if $p_1 = 0$ then $\{$put $b_{Z_{p_1}^1} + 1 := b$ and $k_{p_1}^1 := 1\}$;

For $i := 2$ to r do

begin

$JOB\text{-}SCHED(\{x^{k_{p_{i-1}}^{i-1}}, ..., x^k\}, b_{Z_{p_{i-1}}^{i-1}} + 1, S^i; \{Z_1^i, Z_2^i, ..., Z_{p_i}^i\}, \{k_1^i, k_2^i, ..., k_{p_i}^i\}, p_i)$;

if $p_i = 0$ then $\{$put $b_{Z_{p_i}^i} := b_{Z_{p_{i-1}}^{i-1}}$ and $k_{p_i}^i := k_{p_{i-1}}^{i-1}\}$;

end;

Put

$Z := \{Z_1^1, Z_2^1, ..., Z_{p_1}^1; Z_1^2, Z_2^2, ..., Z_{p_2}^2; ...; Z_1^r, Z_2^r, ..., Z_{p_r}^r\} \stackrel{\text{def}}{=} \{Z_1, Z_2, ..., Z_p\}$;

$p := p_1 + p_2 + \cdots + p_r$;

End.

Proposition 7. *The procedure JOB-SCHEDULES gives following conclusions:*

1. $t_{Z_1^i} < t_{Z_2^i} < \cdots < t_{Z_{p_i}^i}$, $\forall i = 1, 2, ..., r$.
2. $b_{Z_1^i} < b_{Z_2^i} < \cdots < b_{Z_{p_i}^i} < b_{Z_1^{i+1}}$, $\forall i = 1, 2, ..., r$.
3. $Z_1^i \prec Z_2^i \prec \ldots \prec Z_{p_i}^i \prec Z_1^{i+1}$, $\forall i = 1, 2, ..., r$;

Proposition 8. *Let $\mathcal{R}$ in procedure JOB-SCHEDULES be the set of d-optimal schedules, then Z is the set of $(d+1)$-optimal schedules on corresponding sets of jobs and every such schedule contains corresponding $S^i \in \mathcal{R}$ as its "ending part".*

Proposition 9.

1. *The number of schedules in the set Z is $p \leq k + r$, where $k = \sharp U$, $r = \sharp\mathcal{R}$.*
2. *The processing time of procedure JOB-SCHEDULES is $r.O(k.\log k)$.*

5.2.4. Procedure unifies 2 set of schedules, having R-order: $UNION(\mathcal{P}, \mathcal{Q}, X, Y, \mathcal{T})$

Input:

- $\mathcal{P} = \{P_1, P_2, ..., P_p\}$ is a set of R-schedules, where $P_1 \succ_r P_2 \succ_r \cdots \succ_r P_p$;
- $\mathcal{Q} = \{Q_1, Q_2, ..., Q_q\}$ is a set of R-schedules, where $Q_1 \succ_r Q_2 \succ_r \cdots \succ_r Q_q$;
- $[X, Y]$ is the time area.

Output:

$\mathcal{T} = \{T_1, T_2, ..., T_t\}$ is a set of R-schedules, where $T_1 \succ_r T_2 \succ_r \cdots \succ_r T_t$, $t \leq p + q$.

Method: The procedure is similar as unifying 2 ordered sets of integers.

The processing time of this procedure is $O(p + q)$.

5.3. Main algorithms

Let $K = \{x_1, x_2, ..., x_m\}$ be K-schedule, let sets of jobs U_i and other notions be such as (1). There are 3 following main algorithms:

A. Algorithm BASE:

Input: $U = \{u_m^1, u_m^2, u_m^3, ..., u_m^{n_m}\} := \{x^0, x^1, x^2, ..., x^k\}$;

Output: $\mathcal{F} := (\mathcal{W}, \mathcal{V})$ is the pair of 2 sets of 1*-optimal schedules on $U_m^* = U_m$,
$\mathcal{W} = \{W_m^1, W_m^2, ..., W_m^p\}$, $\mathcal{V} = \{V_m^1, V_m^2, ..., V_m^q\}$.

Method: The algorithm applies procedure *RB-JOB* to determine 1*-optimal schedule on U_m.

Algorithm:

Begin
  $i := 1$;
  if there is $\{x^{k_1}\} := RB\text{-}JOB(\{x^0, x^1, x^2, ..., x^k\} \uparrow [b_m(K), C])$
  then $\{W_m^1 := \{x^{k_1}\}$ and $p := 1\}$ else $\{$put $\mathcal{W}_m := \emptyset$ and $p := 0\}$;
  Repeat
    $i := i + 1$;
    if there is $\{x^{k_i}\} := RB\text{-}JOB(\{x^{k_{i-1}+1}, x^{k_{i-1}+2}, ..., x^k\} \uparrow [b_{W_m^{i-1}} + 1, C])$
    then $\{W_m^i := \{x^{k_i}\}$ and $p := i\}$;
  Until $p < i$; (i.e., there is not $\{x^{k_i}\}$);
  $i := 1$;
  if there is $\{y^{h_1}\} := RB\text{-}JOB(\{x^1, x^2, x^3, ..., x^k\} \uparrow [B, C])$
  then $\{V_m^1 := \{y^{h_1}\}$ and $q := i\}$ else $\{$put $\mathcal{V}_m := \emptyset$ and $q := 0\}$;
  Repeat
    $i := i + 1$;
    if there is $\{x^{h_i}\} := RB\text{-}JOB(\{x^{h_{i-1}+2}, x^{h_{i-2}+2}, ..., x^k\} \uparrow [b_{V_m^{i-1}} + 1, C])$
    then $\{V_m^i := \{x^{h_i}\}$ and $q := i\}$;
  Until $q < i$; (i.e., there is not $\{x^{h_i}\}$);
End.

Proposition 10.

1. $\mathcal{F}_m$ is just the pair of 2 sets of 1 - optimal schedules on U_m^*.*

2. $\sharp\mathcal{W}_m$ and $\sharp\mathcal{V}_m \leq n_m$.

3. Processing time of the algorithm is $O(n_m^2)$.

By the method mentioned in the Proposition 5, the algorithm BASE needs only the time $O(n_m.\log n_m)$.

B. Procedure STEPD: $STEPD(\mathcal{F}_{d+1}, \mathcal{F}_d)$

Input:

$\mathcal{F}_{d+1} := (\mathcal{W}_{d+1}, \mathcal{V}_{d+1})$ is a pair of 2 sets of $(m-d)$*-optimal schedules on the set U_{d+1}^*.

Output:

$\mathcal{F}_d := (\mathcal{W}_d, \mathcal{V}_d)$ is a pair of 2 sets of $(m-d+1)$*-optimal schedules on the set U_d^*.

Method: The algorithm applies procedure *JOB-SCHEDULES* to connect jobs of $U_d \cup \{x_{d+1}\}$ to schedules of $\mathcal{F}_{d+1}$, after that by procedure *UNION* to unify the created sets of schedules.

Algorithm:

Begin
  $JOB\text{-}SCHEDULES(U_d, b_d(K), \mathcal{W}_{d+1}; \mathcal{E}, e)$;
  $JOB\text{-}SCHEDULES(U_d - \{x_d\}, B, \mathcal{W}_{d+1}, \mathcal{G}, g)$;

$JOB\text{-}SCHEDULES(\{x_{d+1}\}; B, \mathcal{V}_{d+1}, \mathcal{H}, h)$;
$UNION(\mathcal{E}, \mathcal{H}, b_d(K), C; \mathcal{W}_d); UNION(\mathcal{G}, \mathcal{H}, B, C, \mathcal{V}_d)$;
End.

Proposition 11.

1. $\mathcal{F}_d$ is just pair of 2 sets of $(m-d-1)^*$-optimal schedules on the set U_d^*.

2. $\|\mathcal{W}_d$ and $\|\mathcal{V}_d \le n_d^*$, where $n_d^* = n_d + n_{d+1} + \cdots + n_m$.

3. The processing time of the algorithm is $O(n_d.\log n_d).n_{d+1}^*$.

C. Algorithm USE–STEPD:

Input: $\mathcal{F}_m$ is the pair 2 sets of 1^*-optimal schedules on $U_m^* = U_m$.

Output: $\mathcal{F}_d$ is the pair of 2 sets $(m-d+1)^*$-optimal schedules on U_d^*, for $d = m-1, m-2, \ldots, 2, 1$.

Method: The algorithm applies procedure $STEPD(m-1)$ times with input $\mathcal{F}_m$.

Algorithm: For $d := m-1$ downto 1 do $STEPD(\mathcal{F}_{d+1}, \mathcal{F}_d)$.

Theorem 1. *$\mathcal{F}_1 = (\mathcal{W}_1, \mathcal{V}_1)$ - the result of algorithm USE-STEPD is just the pair of 2 sets of m^*-optimal schedule on U_1^*. Suppose $\mathcal{W}_1 = \{W^1, W^2, \ldots, W^p\}$, then W^1 is just the desirable m^*-optimal schedule.*

Proof. By Proposition 10, $\mathcal{F}_m$ is just the pair of 2 sets of 1^*-optimal schedules on $U_m^* = U_m$. By Proposition 11, $\mathcal{F}_d$ is the pair of 2 sets of $(m-d+1)^*$-optimal schedules on the set U_d^*, for $d = m-1, m-2, \ldots, 2, 1$.

Algorithm USE–STEPS applies procedure STEPD$(m-1)$ times with input $\mathcal{F}_m$, by induction on d we successively obtain the following pairs of 2 sets of schedules: $\mathcal{F}_{m-1}, \mathcal{F}_{m-2}, \ldots, \mathcal{F}_2, \mathcal{F}_1$, where $\mathcal{F}_1 = (\mathcal{W}_1, \mathcal{V}_1)$. Suppose $\mathcal{W}_1 = \{W^1, W^2, \ldots, W^p\}$, then by definitions 11, 12, W^1 is just the desirable m^*-optimal schedule.

Theorem 2. *Processing time of the algorithm USE-STEPD is $O(n^2.\log n)$.*

Proof. According to the Proposition 11, the processing time of the procedure STEPD is $O(n_d.\log n_d).n_{d+1}^*$, for $d = m-1, m-2, \ldots, 2, 1$.

Algorithm USE-STEPD applies procedure STEPD$(m-1)$ times, therefore time for this algorithm is

$$\sum_{d=1}^{m-1} O(n_d.\log n_d).n_{d+1}^* . \tag{19}$$

According to Proposition 10, processing time of the algorithm BASE is $O(n_m.\log n_m)$. Without loss of generality suppose that there is $n_{m+1}^* = 0$, we have following calculations:

$$\sum_{d=1}^{m} n_d.(\log n_d).n_{d+1}^* \le (\log n).\Big(\sum_{d=1}^{m} n_d.n_{d+1}^*\Big),$$

where

$$\begin{aligned}\sum_{d=1}^{m} n_d.n_{d+1}^* &= \sum_{d=1}^{m}(n_d^* - n_{d+1}^*)\,n_{d+1}^* = \sum_{d=1}^{m} n_d^*.n_{d+1}^* - \sum_{d=1}^{m}(n_{d+1}^*)^2 \\ &\le \sum_{d=1}^{m} n_d^*.n_d^* - \sum_{d=1}^{m}(n_{d+1}^*)^2 = \sum_{d=1}^{m}((n_d^*)^2 - (n_{d+1}^*)^2) \\ &= (n_1^*)^2 - (n_2^*)^2 + (n_2^*)^2 - (n_3^*)^2 + \cdots + (n_m^*)^2 - (n_{m+1}^*)^2 = (n_1^*)^2.\end{aligned}$$

The above calculations implies the proof.

Corollary. *Main algorithm determines the optimal schedule of problem [T1] after the time $O(n^2.\log n)$.*

Acknowledgement

The author wish to thank Prof. Michal Chychil, Faculty of Mathematics and Physics, UK in Prague, Prof. Nguyen Huu Ngu, Faculty of Mathematics and Informatics, Hanoi University for their helpful comments.

I wish to express my deep gratitude to Prof. Dr. J. Th. Runnenburg and Faculty of Mathematics and Computer Science, University van Amsterdam for all their useful help during my stay in Amsterdam, where I prepared to write the paper.

REFERENCES

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

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

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

[4] H. Kise, T. Ibaraki, and H. Mine, A solvable case of the one-machine scheduling problem with ready and due times, *Oper. Res.* **26** (1979) 121-126.

[5] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, *Sequencing and Scheduling: Algorithms and Complexity*, Report BS-R8909, Center for Mathematics and Computer Science, P. O. Box 4079, 1009 Amsterdam, The Netherlands, 1989.

Received August 10, 1998

Faculty of Information Technology, Hanoi University
90 Nguyen Trai, Thanh Xuan, Hanoi, Vietnam.