Tạp chí Tin học và Điều khiển học, T. 17, S. 1 (2001), 21–30

THE FAST ALGORITHM FOR FOUNDING NONPREEMPTIVE SCHEDULE WITH SOME ON-TIME JOBS IN MINIMAL PROCESSING TIME

TRINH NHAT TIEN

Abstract. In [2] we presented an $O(n^2.\log n)$ algorithm to determine a schedule with maximal number of on-time jobs in minimal processing time for problem $1|\ r_j\ |\sum U_j$ in the case that release dates and due dates are satisfied $I_1 \preceq I_2 \preceq ... \preceq I_n$, where $I_j := [r_j, d_j]$; (i.e., $r_j < r_k \Rightarrow d_j \leq d_k$).

In this paper, we would extend the above algorithm to determine a schedule of the same problem but with any number of on-time jobs in minimal processing time. The time for this problem is $O(n^3.\log n)$.

Tóm tắt. Trong [2] chúng tôi đã trình bày thuật toán $O(n^2.\log n)$ để xác định thời gian biểu với số lượng lớn nhất các công việc đúng hạn và thời gian xử lý ít nhất cho vấn đề $1|\ r_j\ |\sum U_j$, trong đó $I_j := [r_j, d_j]$, mà $r_j < r_k \Rightarrow d_j \leq d_k$.

Trong bài báo này, chúng tôi mở rộng kết quả của thuật toán trên cho bài toán xây dựng thời gian biểu của cùng vấn đề nhưng số lượng công việc đúng hạn là tùy ý nhưng thời gian xử lý là ít nhất.

1. SOME BASIC CONCEPTS

Some conceptions in the paper are presented in [2]. Now we would remind some concepts and notations related to *"job"*, *"realization"* and *"schedule"*. 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 a *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*.

$R_u := [b_u, c_u]$ (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 a job u is said to be *late*.

Let $I_i = [r_i, d_i]$ and $I_j = [r_j, d_j]$ be active areas of corresponding jobs i and j, respectively. 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 $I_i \not\equiv I_j$. $[B, C] := [\operatorname{Min} r_j, \operatorname{Max} d_j]$ is said to be *an active area* of the system; where B is a *release date* and C is a *due date* of it.

Let U=$\{u_1, u_2, ..., u_s\}$ be a subset of jobs on the system. Suppose that $S := \{R_{u_1}, R_{u_2}, ..., R_{u_s}\}$ is a set of realizations of corresponding jobs $u_1, u_2, ..., u_s$ such that $R_{u_i} \cap R_{u_j} = \emptyset, \forall i \neq j\ i, j = 1, 2, ..., s$. Then S (or $\bigcup_{i=1}^{s} R_{u_i}$) is said to be a *schedule* on the set U of the system (or a *schedule* of the system). $\bigcup_{i=1}^{s} R_{u_i}$ is also said to be a *processing area* of the 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 $R_u(S) \subset I_u$, therefore the *schedule* S is regarded as a set of disjunctive realizations of on-time jobs.

We note some following parameters of the schedule S:

- $\sharp S := s$ is a *number* of realizations or a *number* of jobs;
- $t_S := \sum_{i=1}^{s} t_{u_i}$ is a *processing time* (or a *length*);
- $b_S := \operatorname{Min}\{b_{u_i}\}$ is a *starting time*;
- $c_S := \operatorname{Max}\{c_{u_i}\}$ is a *completion time*;

- $[b_S, c_S]$ is an *active area* of schedule S.

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

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 a *schedule* only on the set of *jobs*, not on a set of pre-jobs. A set of jobs $U = \{u_1, u_2, ..., u_s\}$, which can create any schedule, is said to be a *scheduled set*. In this paper, the such set contains all on-time jobs of the schedule. Sometime for schedule S having scheduled set $\{u_1, u_2, ..., u_s\}$, we also write $S = \{u_1, u_2, ..., u_s\}$.

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 on-time jobs). We know that the problem is strongly NP-hard, authors H. Kise, T. Ibaraki and H. Mine (1979) 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:

[K1]: $1|I_1 \preceq I_2 \preceq ... \preceq I_n|Max \sum \bar{U}_j$, where $\bar{U}_j = 1$ if $c_j \leq d_j$, $\bar{U}_j = 0$ otherwise.

The problem is to build a nonpreemptive schedule with maximal number of on-time jobs.

Now we would pay attention to following special cases:

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

The problem is to construct a nonpreemptive schedule with a maximal number of on-time jobs and furthermore in minimal processing time. In [2] we presented an $O(n^2 \log n)$ algorithm for the problem.

[T2]: $1|I_1 \preceq I_2 \preceq ... \preceq I_n|\sum \bar{U}_j = s$ and Min $\sum \bar{U}_j.t_j$.

The problem is to construct a nonpreemptive schedule with a fixed number of on-time jobs (i.e., s) and furthermore in minimal processing time. In this paper, we extend the $O(n^2 \log n)$ algorithm in [2] to solve the problem [T2].

2. A s-OPTIMAL SCHEDULE

We would remind some following concepts and notations presented in [2].

Let $R_i = [b_i, c_i]$ and $R_j = [b_j, c_j]$ be realizations of corresponding jobs i and j, respectively. 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 from two 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 $R_{u_i} \preceq R_{v_i}$, $\forall i = 1, 2, ..., m$. Similarly we write $P \prec Q$.

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 $P \succ_r Q \leftrightarrow$ 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 $P \succ_{r_i} Q$.

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*.

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

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 realizations $[b_{u_i}, c_{u_i}]$ have following forms:
$b_{u_1} = \text{Max}\{X, r_{u_1}\}$; $c_{u_1} = b_{u_1} + t_{u_1}$;
$b_{u_i} = \text{Max}\{c_{u_{i-1}}, r_{u_i}\}$; $c_{u_i} = b_{u_i} + t_{u_i}$, $\forall i = 2, 3, ..., m$.

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

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.

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 a 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 $b_{v_i}(S) = \text{Max}\{c_{v_{i-1}}(S), r_{v_i}\}$; $c_{v_i}(S) = b_{v_i}(S) + t_{v_i}$, $\forall\ i = 2, 3, ..., q$.

We say that schedule S is *s-optimal* if and only if it is R-schedule having:
(o_1) just s 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 the above conceptions, the *s-optimal* schedule is just *R-best* schedule having s on-time jobs. Therefore solving problem [T2] is just determining the *s-optimal* schedule.

We call the schedule constructed by authors Kise, Ibaraki and Mine (1979) *K-schedule*. We call their algorithm *K-algorithm*. We assume that this schedule has just m jobs, it is the maximal number of on-time jobs.

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

Conclusion. Let U be a set of n jobs on system [T2], $[B, C]$ 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 use some results in [2], for instance K-schedule is just L-best schedule.

We write following notions:

$$U_0 = \{jobs\ \ u \parallel I_u \prec I_{x_1}\}; U_m = \{jobs\ \ u \parallel I_{x_m} \preceq I_u\}. \tag{1}$$

$U_i = \{jobs\ \ u \parallel I_{x_i} \preceq I_u \prec I_{x_{i+1}}\}$ *for* $i = 1, 2, ..., m-1$;

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

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

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

$U_2 = \{x_2, u_2^2, ..., u_2^{n_2}\}$, where $x_2 \doteq u_2^1$;

...

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

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

where $U = U_0 \cup U_1 \cup U_2 \cup ... \cup U_m$; $n = n_0 + n_1 + n_2 + ... + n_m$.

We write $U_i^* = U_i \cup U_{i+1} \cup U_{i+2} \cup ... \cup U_m$; $n_i^* = n_i + n_{i+1} + n_{i+2} + ... + n_m$.

The following lemmas and their proofs are similar as according lemmas in [2]:

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

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

$$c_{x_k}(K) \le c_{w_j}(W) \ \ and \ \ s - (j+1) < m - k \tag{2}$$

then

$$U_k \ \ doesn't \ \ contain \ \ a \ \ next \ \ job \ \ w_{j+1} \in W; \tag{3}$$

$$U_0 \ \ doesn't \ \ contain \ \ any \ \ job \ \ w \in W. \tag{4}$$

Lemma 2. *The assumptions are the same as in the Lemma 1. We have following result:*

For $k = 1, 2, ..., m-1$, if $U_k \cup \{x_{k+1}\}$ contains $w_j \in W$ such that

$$b_{w_j}(W) < c_{x_k}(K) \ \ and \ \ j - 2 < k \tag{5}$$

then

$$U_k - \{x_k\} \ \ doesn't \ \ contain \ \ a \ \ preceding \ \ job \ \ w_{j-1} \in W; \tag{6}$$

$$U_m \ \ doesn't \ \ contain \ \ a \ \ job \ \ w \in W \ \ such \ \ that \ c_w(W) < c_{x_m}(K). \tag{7}$$

Lemma 3. *The assumptions are the same as in the Lemma 1. We have following result: There is not any integer k $(0 \le k \le m)$ such that $U_k - \{x_k\}$ contains 2 neighbouring jobs w_j, $w_{j+1} \in W$.*

Proof. By the contradiction, suppose that there is integer k $(0 \le k \le m)$ smallest such that $U_k - \{x_k\}$ contains 2 neighbouring jobs w_j, $w_{j+1} \in W$. We consider 2 following cases:

- When $c_{x_k}(K) \le c_{w_j}(W)$. On the other hand, there is $s - (j+1) < m - k$. That is in the contradictory with Lemma 1.

- When $c_{x_k}(K) > c_{w_j}(W)$. It means $b_{w_{j+1}}(W) < c_{x_k}(K)$. On the other hand, there is $(j+1) - 2 < k$. That is in the contradictory with Lemma 2.

Lemma 4. *The assumptions are the same as in the Lemma 1. We have following result:*

$$w_i \in U_i \cup U_{i+1} \cup ... \cup U_{i+m-s} \cup \{x_{i+m-s+1}\}, \ \ \forall i = 1, 2, ..., s-1 \tag{8}$$

and $w_s \in U_s \cup U_{s+1} \cup ... \cup U_m$.

We can easily prove the lemma by the contradiction and the Lemma 3.

4. s^*-OPTIMAL SCHEDULE

From result of Lemma 4 we define concept "s^*-*optimal*" schedule related to the *s-optimal* schedule.

Definition 1. Let $K = \{x_1, x_2, ..., x_m\}$ be K-schedule, let sets of jobs U_i and other notions be such as (1). For $d = 1, 2, ..., m$, with q $(1 \leq q \leq m - d + 1)$, we say that S is q^*-*optimal schedule* on set of jobs $U_d^* \uparrow [B, C]$ if and only if it is *q-optimal* and has following form: $S := \{w_d, w_{d+1}, ..., w_{d+q-1}\}$, where

$$w_i \in U_i \cup U_{i+1} \cup ... \cup U_{i+q-2} \cup \{x_{i+q-1}\}, \quad \forall i = d, d+1, ..., d+q-2 \tag{9}$$

and $w_{d+q-1} \in U_{d+q-1} \cup U_{d+q} \cup ... \cup U_m$.

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

Lemma 5. *The assumptions are the same as in the Definition 1. We have following result:*

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

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

We can prove result 1/. by contradiction and by using definitions of q^*-optimal schedule and K-schedule. Result 2/. is the corollary of result 1/.

Definition 2. The assumptions are the same as in the Definition 1. For $d = 1, 2, ..., m$, with q $(1 \leq q \leq m - d + 1)$, we define following concepts:

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

$$W_d^1 \text{ is } q^* - \text{optimal schedule on } U_d^* \uparrow [B, C] \tag{10}$$

$$W_d^i \text{ is } q^* - \text{optimal schedule on } U_d^* \uparrow [b_{W_d^{i-1}} + 1, C], \tag{11}$$

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

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

$$V_d^1 \text{ is } q^* - \text{optimal schedule on } (U_d^* - \{x_d\}) \uparrow [B, C] \tag{12}$$

$$V_d^i \text{ is } q^*\text{-optimal schedule on } (U_d^* - \{x_d\}) \uparrow [b_{V_d^{i-1}} + 1, C], \tag{13}$$

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

$\mathcal{F}_d^q := (\mathcal{W}_d, \mathcal{V}_d)$ is said to be a *pair* of 2 *sets* of q^*-*optimal* schedules on 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 $t_{S^i} < t_{S^{i+1}}$; $b_{S^i} < b_{S^{i+1}}$; $S^i \preceq S^{i+1}$, $\forall i = 1, 2, ..., p - 1$.

Proposition 1. *The defined sets* $\mathcal{W}_d$ *and* $\mathcal{V}_d$ *have R-order.*

Definition 3. With d $(1 \leq d \leq s - 1)$, let $\mathcal{F}_d^q := (\mathcal{W}_d^q, \mathcal{V}_d^q)$ is a *pair* of 2 *sets* of q^*-*optimal* schedules on set U_d^*.

$\mathcal{A}_d^q := \{\mathcal{F}_d^q, \mathcal{F}_{d+1}^q, ..., \mathcal{F}_{d+m-s}^q\}$ is said to be a *system of pairs* of 2 *sets* (or a *system*) of q^*-*optimal* schedules on set U_d^*.

Lemma 6. *For* d $(1 \leq d \leq s - 1)$, *let* $\mathcal{A}_{d+1}^{q-1} := \{\mathcal{F}_{d+1}^{q-1}, \mathcal{F}_{d+2}^{q-1}, ..., \mathcal{F}_{d+m-(s-1)}^{q-1}\}$ *be a system of* $(q-1)^*$*-optimal schedules on the set* U_{d+1}^*.

Suppose that $\mathcal{A}_d^q := \{\mathcal{F}_d^q, \mathcal{F}_{d+1}^q, ..., \mathcal{F}_{d+m-s}^q\}$ *be a system of* q^**-optimal schedules on the set* U_d^*. *We have following conclusion:*

Every schedule from the system $\mathcal{A}_d^q$ has to contain any schedule from the system $\mathcal{A}_{d+1}^{q-1}$ as its "ending part" with $(q-1)$ jobs.

Corollary. *The s^*-optimal schedule has to contain any schedule from the system $\mathcal{A}_2^{s-1}$ as its "ending part" with $(s-1)$ jobs.*

5. ALGORITHM DETERMINING s^*-OPTIMAL SCHEDULE

5.1. Main idea of algorithm

By the above results, our algorithm will 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 $O(n.\log n)$.

- Lemma 4 and Lemma 5 determine the position of the s-optimal schedule W in comparise with K-schedule. Here if $W = \{w_1, w_2, ..., w_s\}$ then

$$w_i \in U_i \cup U_{i+1} \cup ... \cup U_{i+m-s} \cup \{x_{i+m-s+1}\}, \quad \forall i = 1, 2, ..., s-1; \tag{14}$$

$w_s \in U_s \cup U_{s+1} \cup ... \cup U_m$ *and* $b_K \leq b_W$.

For $d := s, s-1, s-2, ..., 2, 1$, put $q := s - d + 1$.

To create W, we construct the system $\mathcal{A}_1^s$ of all schedules, which could become W, these such schedules equally have property (14). By Lemma 6, this system will created recurssively by 3 following algorithms:

1/ Algorithm SBASE will create the basic system $\mathcal{A}_s^1$, i.e., the system of 1^*-*optimal* schedules on the set U_s^*; one from these schedules will become *"an ending part"* $\{w_s\}$ of optimal schedule W.

2/ Procedure SSTEP will from the well-known system $\mathcal{A}_{d+1}^{q-1}$ of $(q-1)^*$-*optimal* schedules on the set U_{d+1}^* determine a system $\mathcal{A}_d^q$ of q^*-*optimal* schedules on the set U_d^*; one from these schedules will become *"an ending part"* $\{w_d, w_{d+1}, ..., w_{d+(s-1)}\}$ of the optimal schedule W.

3/ Algorithm USE-SSTEP will from the basic system $\mathcal{A}_s^1$ apply $(s-1)$ times the procedure SSTEP, we will obtain successively systems: $\mathcal{A}_{s-1}^2$, $\mathcal{A}_{s-2}^3$, ..., $\mathcal{A}_2^{s-1}$, $\mathcal{A}_1^s$, where $\mathcal{A}_1^s := \{\mathcal{F}_1^s, \mathcal{F}_2^s, ..., \mathcal{F}_{m-(s-1)}^s\}$, $\mathcal{F}_1^s = (\mathcal{W}_1^s, \mathcal{V}_1^s)$. Suppose $\mathcal{W}_1^s = \{W^1, W^2, ..., W^p\}$ then W^1 is just the desirable s^*-*optimal* schedule.

5.2. Some auxiliary procedures

1/ Procedure finds R-best realization on the set of jobs:

Let a set of jobs $U = \{x^1, x^2, ..., x^k\}$, according to the definition of R-best schedule, 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:

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

Processing time of this procedure is $O(k)$. We need note that, may be $\{x^1, x^2, ..., x^k\} \uparrow [X, Y]$ is not a set of jobs, therefore there is not such $\{x\}$.

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

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

- $\mathcal{K} = \{k_1, k_2, ..., k_p\}$ is a set index corresponding to $\mathcal{Z}$; $p = \sharp\mathcal{Z}$.

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 $\{x^{k_1}\} := RB - JOB(\{x^1, x^2, ..., x^k\} \uparrow [b, b_S])$

then $Z_1 := \{x^{k_1}\} \oplus_r S$ and p:=1 else put $\mathcal{Z} := \emptyset$ and p:=0;

Repeat

$i := i + 1$;

if there is $\{x^{k_i}\} := RB - JOB(\{x^{k_{i-1}+1}, x^{k_{i-1}+2}, ..., x^k\} \uparrow [b_{Z_{i-1}} + 1, b_S])$

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 2. *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. $t_{k_1} < t_{k_2} < ... < t_{k_p}$; $t_{k_1} \leq t_{x_j}$, $\forall x^j \in U \uparrow [b, b_S]$;
   $t_{k_i} \leq t_{x_j}$, $\forall x^j \in U \uparrow [b_{Z_{i-1}} + 1, b_S]$, $\forall i = 2, 3, ..., p$;
2. $\{x^{k_i}\}$ *is R-best realization of 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. $\mathcal{Z} = \{Z_1, Z_2, ..., Z_p\}$ *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–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 realizations to S. This method needs only the time $O(k.\log k)$.

Proposition 3. *For $i = 1, 2, ..., p$, let S in 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".*

3/ Procedure connects a set of jobs to a set of schedules: $JOB–SCHEDULES(U, b, \mathcal{R}; \mathcal{Z}, p)$

Input: - $U = \{x^1, x^2, ..., x^k\}$ is the set of jobs such as $I_{x^1} \preceq I_{x^2} \preceq ... \preceq I_{x^k}$;

- 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 $I_{x^k} \prec I_{y^1} \preceq I_{y^2} \preceq ... \preceq I_{y^h}$; where $r = \sharp\mathcal{R}$.

Output: $\mathcal{Z} = \{Z_1, Z_2, ..., Z_p\}$ 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 $p = \sharp\mathcal{Z}$.

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

Algorithm:

Begin

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

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

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

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

$p := p_1 + p_2 + ... + p_r$;

End.

Proposition 4. *The procedure JOB–SCHEDULES gives following conclusions:*

1. $t_{Z_1^i} < t_{Z_2^i} < ... < t_{Z_{p_i}^i}$, $\forall i = 1, 2, ..., r$;

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

Proposition 5. *Let $\mathcal{R}$ in procedure JOB-SCHEDULES be the set of d-optimal schedules, then $\mathcal{Z}$ 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 6.
1. *The number of schedules in the set $\mathcal{Z}$ is $p \leq k + r$, where $k = \sharp U$, $r = \sharp \mathcal{R}$.*
2. *The processing time of procedure JOB-SCHEDULES is $r.O(k.\log k)$.*

4/ Procedure unifies 2 sets 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 $P_1 \succ_r P_2 \succ_r ... \succ_r P_p$;
- $\mathcal{Q} = \{Q_1, Q_2, ..., Q_q\}$ is a set of R-schedules, where $Q_1 \succ_r Q_2 \succ_r ... \succ_r Q_q$;
- $[X, Y]$ is the time area.

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

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:

1/ Algorithm SBASE

a. Auxiliary procedure BASE: $BASE(U, b; \mathcal{F})$.

Input: $U := \{x^0, x^1, x^2, ..., x^k\}$; b is a integer.
Output: $\mathcal{F} := (\mathcal{W}, \mathcal{V})$ is the pair of 2 sets of 1^* – *optimal* schedules on U,
$\mathcal{W} = \{W_1, W_2, ..., W_p\}$, $\mathcal{V} = \{V_1, V_2, ..., V_q\}$.

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

Algorithm:

```
Begin
    i := 1;
    if there is {x^{k_1}} := RB-JOB({x^0, x^1, x^2, ..., x^k} ↑ [b, C])
    then W_1 := {x^{k_1}} and p := 1 else put W := ∅ and p := 0;
    Repeat
       i := i + 1;
       if there is {x^{k_i}} := RB-JOB({x^{k_{i-1}+1}, x^{k_{i-1}+2}, ..., x^k} ↑ [b_{W_{i-1}} + 1, C])
       then W_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 - JOB({x^1, x^2, x^3, ..., x^k} ↑ [B, C])
    then V_1 := {y^{h_1}} and q :=1 else put V := ∅ and q :=0 ;
    Repeat
       i := i + 1;
       if there is {x^{h_i}} := RB-JOB({x^{h_{i-1}+1}, x^{h_{i-1}+2}, ..., x^k} ↑ [b_{V_{i-1}} + 1, C])
       then V_i := {x^{h_i}} and q := i;
    Until q < i; (i.e., there is not {x^{h_i}});
End;
```

b. Algorithm SBASE:

Input: U_i^*, $b_i(K)$, for i=s,s-1,...,m are the same as (1).

Output: $\mathcal{A}_s^1 := \{\mathcal{F}_s^1, \mathcal{F}_{s+1}^1, ..., \mathcal{F}_m^1\}$ is the system of 1^*-*optimal* schedules on set U_s^*.

Method: The algorithm applies the procedure $BASE$ $(m - s + 1)$ times.

Algorithm: For $i := s$ To m Do $BASE(U_i^*, b_i(K); \mathcal{F}_i^1)$;

Proposition 7.

1. $\mathcal{A}_s^1$ *is just the system of* 1^*-*optimal schedules on* U_s^*.
2. $\sharp\mathcal{W}_i^1$ *and* $\sharp\mathcal{V}_i^1 \leq n_i^*$, *for* $i = s, s-1, ..., m$, *where* $\mathcal{F}_i^1 = (\mathcal{W}_i^1, \mathcal{V}_i^1)$.
3. $\mathcal{F}_i^1$ *is determined after the time* $O((n_i^*)^2)$.
4. *Processing time of the algorithm is* $\sum_{i=1}^{(m-s+1)} O((n_i^*)^2)$.

By the method mentioned in the Proposition 2, $\mathcal{F}_i^1$ is determined after the time $O(n_i^*.\log n_i^*)$, therefore the algorithm SBASE needs only the time $\sum_{i=1}^{(m-s+1)} O(n_i^*.\log n_i^*)$.

2/ Procedure SSTEP: $SSTEP(\mathcal{A}_{d+1}^{q-1}, \mathcal{A}_d^q)$; $d = s-1, s-2, ..., 2, 1$.

Input: $\mathcal{A}_{d+1}^{q-1} := \{\mathcal{F}_{d+1}^{q-1}, \mathcal{F}_{d+2}^{q-1}, ..., \mathcal{F}_{d+m-(q-1)}^{q-1}\}$ is the system of $(q-1)^*$-*optimal* schedules on the set U_{d+1}^*.

Output:

$\mathcal{A}_d^q := \{\mathcal{F}_d^q, \mathcal{F}_{d+1}^q, ..., \mathcal{F}_{d+m-q}^q\}$ is the system of q^*-*optimal* schedules on the set U_d^*.

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

Algorithm:

```
Begin
    For i := d To (d + m - s) Do
    begin
      JOB-SCHEDULES(U_i, b_i(K), W_{i+1}^{q-1}; E, e);
      JOB-SCHEDULES(U_i - {x_i}, B, W_{i+1}^{q-1}; G, g);
      JOB-SCHEDULES({x_{i+1}}, B, V_{i+1}^{q-1}; H, h);
      UNION(E, H, b_i(K), C; W_i^q); UNION(G, H, B, C; V_i^q);
    end;
End;
```

Proposition 8. *For* $d := s-1, s-2, ..., 2, 1$ *and* $q := s - d + 1$:

*b*1. $\mathcal{A}_d^q$ *is just the system of* q^*-*optimal schedules on the set* U_d^*.

2. *For* $i = d, d+1, ..., d+m-s$: $\sharp\mathcal{W}_i^q$ *and* $\sharp\mathcal{V}_i^q \leq n_i^*$, *where* $n_i^* = n_i + n_{i+1} + ... + n_m$.

3. *The processing time of the algorithm is* $\sum_{i=d}^{d+m-q} O(n_i.\log n_i).n_{i+1}^*$.

3/ Algorithm USE-SSTEP:

Input: $\mathcal{A}_s^1$ is the system of 1^*-*optimal* schedules on U_s^*.

Output: $\mathcal{A}_{s-1}^2$, $\mathcal{A}_{s-2}^3$, ..., $\mathcal{A}_2^{s-1}$, $\mathcal{A}_1^s$ are the desirable systems of schedules.

Method: The algorithm applies procedure SSTEP (s-1) times with input $\mathcal{A}_s^1$.

Algorithm: For $d := s-1$ DownTo 1 Do $SSTEP(\mathcal{A}_{d+1}^{q-1}, \mathcal{A}_d^q)$;

Theorem 1. *In the output of algorithm USE-SSTEP suppose* $\mathcal{A}_1^s := \{\mathcal{F}_1^s, \mathcal{F}_2^s, ..., \mathcal{F}_{m-(s-1)}^s\}$, $\mathcal{F}_1^s = (\mathcal{W}_1, \mathcal{V}_1)$ *and* $\mathcal{W}_1 = \{W^1, W^2, ..., W^p\}$, *then* W^1 *is just the desirable* s^*-*optimal schedule.*

Proof. By Proposition 7, $\mathcal{A}_s^1$ is just the system of 1^*-*optimal* schedules on U_s^*. By proposition 8, $\mathcal{A}_d^q$ is the system of q^*-*optimal* schedules on the set U_d^*, for $d = s-1, s-2, ..., 2, 1$. Algorithm USE–SSTEP applies procedure SSTEP $(s-1)$ times with input $\mathcal{A}_s^1$, by induction on d, we successively obtain the following systems of schedules: $\mathcal{A}_{s-1}^2$, $\mathcal{A}_{s-2}^3$, ..., $\mathcal{A}_2^{s-1}$, $\mathcal{A}_1^s$, suppose $\mathcal{A}_1^s := \{\mathcal{F}_1^s, \mathcal{F}_2^s, ..., \mathcal{F}_{m-(s-1)}^s\}$, $\mathcal{F}_1^s = (\mathcal{W}_1, \mathcal{V}_1)$ and $\mathcal{W}_1 = \{W^1, W^2, ..., W^p\}$, then by definitions 2, 3, W^1 is just the desirable s^* – *optimal* schedule.

Theorem 2. *Processing time of the algorithm USE–STEPD is* $(m-s+1).O(n^2 \log n)$.

Proof. According to the proposition 8, the processing time of the procedure SSTEP is $\sum_{i=d}^{d+m-s} O(n_i.\log n_i).n_{i+1}^*$, for $d = s-1, s-2, ..., 2, 1$.

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

$$\sum_{d=1}^{s-1}\Big(\sum_{i=d}^{d+m-s} O(n_i.\log n_i).n_{i+1}^*\Big) = \sum_{i=0}^{m-s}\Big(\sum_{d=1}^{s-1} O(n_{d+i}.\log n_{d+i}).n_{d+i+1}^*\Big)$$

$$\leq (m-s+1).\Big(\sum_{k=1}^{m} O(n_k.\log n_k).n_{k+1}^*\Big). \tag{15}$$

Without loss of generality suppose that there is $n_{m+1}^* = 0$, we have following calculations:

$$\sum_{k=1}^{m} n_k.(\log n_k).n_{k+1}^* \leq (logn).\Big(\sum_{k=1}^{m} n_k.n_{k+1}^*\Big),$$

where

$$\sum_{k=1}^{m} n_k.n_{k+1}^* = \sum_{k=1}^{m}(n_k^* - n_{k+1}^*).n_{k+1}^* = \sum_{k=1}^{m} n_k^*.n_{k+1}^* - \sum_{k=1}^{m}(n_{k+1}^*)^2$$

$$\leq \sum_{k=1}^{m} n_k^*.n_k^* - \sum_{k=1}^{m}(n_{k+1}^*)^2 = \sum_{k=1}^{m}((n_k^*)^2 - (n_{k+1}^*)^2)$$

$$= (n_1^*)^2 - (n_2^*)^2 + (n_2^*)^2 - (n_3^*)^2 + ... + (n_m^*)^2 - (n_{m+1}^*)^2 = (n_1^*)^2.$$

The above calculations implies the proof.

Corollary. *Main algorithm determines the optimal schedule of problem* [T2] *after the time* $(m-s+1).O(n^2 logn)$, *where* m *is the maximal number of on-time jobs,* s *is the fixed number of on-time jobs* $(s \leq m)$; *(i.e.,* $O(n^3 \log n)$*).*

Conclusion. We know that m is the maximal number of on-time jobs, therefore the problem [T2] is solved only if $s \leq m$. In the case s=m, the problem [T2] is just the problem [T1], according to the above corollary the time for this case is $O(n^2 \log n)$.

Acknowledgement. I wish to express my deep gratitude to Faculty of Mathematics and Computer science, University van Amsterdam, where the paper was written.

REFERENCES

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

[2] Trinh Nhat Tien, An $O(n^2.logn)$ for a nonpeemptive schedule on one machine, *Journal of Computer Science and Cybernetics* **15** (1) (1999) 66–76.

Received July 14, 2000

Hanoi, Faculty of Technology,
Vietnam National University,
144 Xuan Thuy, Cau Giay, Hanoi, Vietnam.