The 7th International Symposium on Operations Research and Its Applications (ISORA'08)
Lijiang, China, October 31–Novemver 3, 2008

Approximation Algorithm of Minimizing Makespan in Parallel Bounded Batch Scheduling*

Jianfeng Ren[1,†] Yuzhong Zhang[1] Sun Guo[2]

[1]College of Operations Research and Management Sciences, Qufu Normal University, Shangdong, 276826, China
[2]Department of Mathematics, Qufu Normal University, Shangdong, 276826, China.

Abstract We consider the problem of minimizing the makespan(C_{max}) on m identical parallel batch processing machines. The batch processing machine can process up to B jobs simultaneously. The jobs that are processed together form a batch, and all jobs in a batch start and complete at the same time. For a batch of jobs, the processing time of the batch is equal to the largest processing time among the jobs in the batch. In this paper, we design a fully polynomial time approximation scheme (FPTAS) to solve the bounded identical parallel batch scheduling problem $P_m|B<n|C_{max}$ when the number of identical parallel batch processing machines m is constant.

Keywords Approximation algorithm; Bounded batch scheduling; Makespan; FPTAS; Dynamic programming.

1 Introduction

Model: A batching machine or batch processing machine is a machine that can process up to B jobs simultaneously. The jobs that are processed together form a batch. Specifically, we are interested in the so-called burn-in model, in which the processing time of a batch is defined to the maximum processing time of any job assigned to it. All jobs contained in the same batch start and complete at the same time, since the completion time of a job is equal to the completion time of the batch to which it belongs. This model is motivated by the problem of scheduling burn-in operations for large-scale integrated circuit(IC) chips manufacturing (see Lee [1] for the detailed process). In this paper, we study the problem of scheduling n independent jobs on m parallel machines to minimize the makespan. Using the notation of Graham et al [2], we denote this problem as $P_m|B<n|C_{max}$. Karp [3] showed that the problem $P_2||C_{max}$ is NP-hard in the ordinary sense. Since it contains $P_2||C_{max}$ as a special case, the problem $P_m|B<n|C_{max}$ is also NP-hard in the ordinary sense.

*Supported by the National Natural Science Foundation (Grant Number 10671108) and the Province Natural Science Foundation of Shandong (Grant Number $Y2005A04$).
†Corresponding author.Email: qrjianfeng@163.com (J.F. Ren).

Previous related work: Karp (1972) showed that $P_2||C_{max}$ is NP-hard in the ordinary sense, and Sahnin [4] gave an FPTAS for it. For the problem of $1|B|C_{max}$, Bartholdi (1988) showed that it can be solved optimally by the algorithm of full batch longest processing time (FBLPT). As to minimize the makespan on parallel identical batch processing machines, Lee et al. ([5]) provided efficient algorithms under some assumptions. For the problem $1|r_j, B|C_{max}$, Deng and zhang [5] derived a PTAS algorithm. For the problem $R|B|C_{max}$, Zhang (2005) [7] presented a $(2 - \frac{1}{B} + \varepsilon)$-approximation algorithm.

Our contributions: In this paper, we design a fully polynomial time approximation scheme (FPTAS) to solve the bounded identical parallel batch scheduling problem $P_m|B < n|C_{max}$ when the number of identical parallel batch processing machines m is constant.

2 Problem Description, Notation, and Elementary Definitions

The scheduling model that we analyze is as following. There are n independent jobs $J_1, J_2, \cdots, J_n$ that have to be scheduled on m bounded identical parallel batch machines. Each job J_j $(j = 1, 2, \cdots, n)$ has a non-negative processing time p_j. All jobs are available for processing at time 0. The goal is to scheduling the jobs without preemption on m bounded identical parallel batch machines such that the makespan is minimized.

The set of real numbers is denoted by IR, and the set of non-negative integers is denoted by IN; note that $0 \in IN$. The base two logarithm of z denoted by $\log z$, and the natural logarithm by $\ln z$.

We recall the following well-known properties of binary relations $\preceq$ on a set Z. The relation $\preceq$ is called

- *reflexive*, if for any $z \in Z$: $z \preceq z$,
- *symmetric*, if for any $z, z' \in Z$: $z \preceq z'$ implies $z' \preceq z$,
- *anti-symmetric*, if for any $z, z' \in Z$: $z \preceq z'$ and $z' \preceq z$ implies $z' = z$,
- *transitive*, if for any $z, z', z'' \in Z$: $z \preceq z'$ and $z' \preceq z''$ implies $z \preceq z''$.

A relation on z is called a *partial order*, if it is reflexive, anti-symmetric, and transitive. A relation on Z is called a *quasi-order*, if it is reflexive and transitive. A quasi-order on Z is called a quasi-linear order, if any two elements of Z are comparable.

For $Z' \subseteq Z$, an element $z \in Z'$ is called a *maximum* in Z' with respect to $\preceq$, if $z' \preceq z$ holds for all $z' \in Z'$. The element $z \in Z'$ is called a *maximal* in Z' with respect to $\preceq$, if the only $z' \in Z'$ with $z \preceq z'$ is z itself.

Proposition[6]: For any binary relation $\preceq$ on a set Z, and for any finite subsetZ' of Z the following holds.
(i) If $\preceq$ is a partial order, then there exists a *maximal* element in Z.
(ii) If $\preceq$ is a quasi-line order, then there exists at least one *maximum* element in Z.

Woeginger [6] showed that dynamic programming algorithm with a special structure automatically lead to a *fully polynomial time approximation scheme*. Assume that we have an approximation algorithm that always returns a near-optimal solution whose cost is at most a factor of ρ away from the optimal cost, where $\rho > 1$ is some real number: in minimization problems the near-optimal is at most a multiplicative factor of ρ above the optimum, and in maximinization problems it is at most a factor of ρ below the optimum.

Such an approximation algorithm is called a ρ-approximation algorithm. A family of $(1+\varepsilon)$-approximation algorithms over all $\varepsilon > 0$ with polynomial running time is called a polynomial time approximation scheme (PTAS). If the time complexity of a PTAS is also polynomially bounded in $(\frac{1}{\varepsilon})$, then it is called a fully polynomial time approximation scheme (FPTAS). An FPTAS is the strongest possible polynomial time approximation result that we can derive for an NP-hard problem, unless P=NP. Woeginger et al. considered a GENEric optimization problem (for short GENE) and provided a uniform approach to design the fully polynomial time approximation scheme for it.

A DP-simple optimization problem GENE is called DP-benevolent iff there exist a partial order $\preceq_{dom}$, a quasi-wine order $\preceq_{qua}$, and a degree-vector D such that its dynamic programming formulation DP fulfills the Conditions C.1-C.4. The **lemma1**[6] in section 3 denotes that the DP-benevolent problems are easy to approximate.

3 The Dynamic Programming

As we can firstly schedule all jobs whose processing times are zero and let all non-zero processing times be integers by enlarge the same times and keep the same structure of optimal schedule, we propose that all p_j $(j = 1, 2, \cdots, n)$ are non-negative integers. We renumber the jobs such that $p_1 \geq p_2 \geq \cdots \geq p_n$.

lemma2 There exists an optimal schedule in which all machines process the jobs increasing order of index. Moreover, an optimal schedule will not contain any machine idle time. A straightforward job interchange argument can proof the lemma.

Now let $\alpha = 1$ and $\beta = 4m$. For $k = 1, 2, \cdots, n$ define the input vector $X_k = [p_k]$. A state $S = [s_1^{(1)}, s_2^{(1)}, s_3^{(1)}, s_4^{(1)}, \cdots, s_1^{(m)}, s_2^{(m)}, s_3^{(m)}, s_4^{(m)}] \in S_k$ encodes a partial schedule for the first jobs $J_1, J_2, \cdots, J_k$: the coordinate $s_1^{(l)}$ measures the total processing time on the l-th machine in the partial schedule, and $s_2^{(l)}$ measures the processing time of the last batch on the l-th machine in the partial schedule. The two additional coordinates $s_3^{(l)}$ and $s_4^{(l)}$, respectively, stores the least and the largest index of the last batch on the l-th machine in the partial schedule. The set $\mathscr{F}$ consists of $4m$ functions $F_1^{(l)}, F_2^{(l)}, F_3^{(l)}, F_4^{(l)}, \quad l = 1, 2, \cdots, m$.

$$F_1^{(l)}[p_k, s_1^{(1)}, s_2^{(1)}, s_3^{(1)}, s_4^{(1)}, \cdots, s_1^{(m)}, s_2^{(m)}, s_3^{(m)}, s_4^{(m)}] =$$

$$[s_1^{(1)}, s_2^{(1)}, s_3^{(1)}, s_4^{(1)}, \cdots, s_1^{(l-1)}, s_2^{(l-1)}, s_3^{(l-1)}, s_4^{(l-1)}, s_1^{(l)}, s_2^{(l)}, s_3^{(l)}, k, s_1^{(l+1)}, s_2^{(l+1)}, s_3^{(l+1)}, s_4^{(l+1)}, \cdots, s_1^{(m)}, s_2^{(m)}, s_3^{(m)}, s_4^{(m)}]$$

$$F_2^{(l)}[p_k, s_1^{(1)}, s_2^{(1)}, s_3^{(1)}, s_4^{(1)}, \cdots, s_1^{(m)}, s_2^{(m)}, s_3^{(m)}, s_4^{(m)}] =$$

$$[s_1^{(1)}, s_2^{(1)}, s_3^{(1)}, s_4^{(1)}, \cdots, s_1^{(l-1)}, s_2^{(l-1)}, s_3^{(l-1)}, s_4^{(l-1)}, s_1^{(l)} + p_k, p_k, k, k, s_1^{(l+1)}, s_2^{(l+1)}, s_3^{(l+1)}, s_4^{(l+1)}, \cdots, s_1^{(m)}, s_2^{(m)}, s_3^{(m)}, s_4^{(m)}]$$

Intuitively speaking, the function $F_1^{(l)}$ schedule the job J_k on the l-th machine and put it into the last batch of the partial schedule $S = [s_1^{(1)}, s_2^{(1)}, s_3^{(1)}, s_4^{(1)}, \cdots, s_1^{(m)}, s_2^{(m)}, s_3^{(m)}, s_4^{(m)}] \in S_k$ for the jobs $J_1, J_2, \cdots, J_k$. i.e, Adds job J_k to the l-th machine so that it does not start the last batch.

The function $F_2^{(l)}$ schedule the job J_k to the l-th machine end of the partial schedule $S=[s_1^{(1)},s_2^{(1)},s_3^{(1)},s_4^{(1)},\cdots,s_1^{(m)},s_2^{(m)},s_3^{(m)},s_4^{m}]\in S_k$ for the jobs $J_1,J_2,\cdots,J_k$. i.e, Adds job J_k so that it starts the last batch.

The functions $H_1^{(l)}$ and $H_2^{(l)}$ in $\mathscr{H}$ correspond to $F_1^{(l)}$ and $F_2^{(l)}$, respectively.

$$H_1^{(l)}[p_k,s_1^{(1)},s_2^{(1)},s_3^{(1)},s_4^{(1)},\cdots,s_1^{(m)},s_2^{(m)},s_3^{(m)},s_4^{(m)}]=k-s_3^{(l)}+1-B$$

$$H_2^{(l)}[p_k,s_1^{(1)},s_2^{(1)},s_3^{(1)},s_4^{(1)},\cdots,s_1^{(m)},s_2^{(m)},s_3^{(m)},s_4^{(m)}]\equiv 0 \qquad l=1,2,\cdots,m.$$

Now the iterative computation in Line 5 of DP for all functions in $\mathscr{F}$ reads

If $k-s_3^{(l)}+1-B\le 0$ then add
$[s_1^{(1)},s_2^{(1)},s_3^{(1)},s_4^{(1)},\cdots,s_1^{(l-1)},s_2^{(l-1)},s_3^{(l-1)},s_4^{(l-1)},s_1^{(l)},s_2^{(l)},s_3^{(l)},k,s_1^{(l+1)},s_2^{(l+1)},s_3^{(l+1)},s_4^{(l+1)},\cdots,s_1^{(m)},s_2^{(m)},s_3^{(m)},s_4^{(m)}]$

If $0\le 0$ then add
$[s_1^{(1)},s_2^{(1)},s_3^{(1)},s_4^{(1)},\cdots,s_1^{(l-1)},s_2^{(l-1)},s_3^{(l-1)},s_4^{(l-1)},s_1^{(l)}+p_k,p_k,k,k,s_1^{(l+1)},s_2^{(l+1)},s_3^{(l+1)},s_4^{(l+1)},\cdots,s_1^{(m)},s_2^{(m)},s_3^{(m)},s_4^{(m)}]$

$l=1,2,\cdots,m.$

Finally, set
$G[s_1^{(1)},s_2^{(1)},s_3^{(1)},s_4^{(1)},\cdots,s_1^{(l-1)},s_2^{(l-1)},s_3^{(l-1)},s_4^{(l-1)},s_1^{(l)},s_2^{(l)},s_3^{(l)},s_4^{(l)},s_1^{(l+1)},s_2^{(l+1)},s_3^{(l+1)},s_4^{(l+1)},\cdots,s_1^{(m)},s_2^{(m)},s_3^{(m)},s_4^{(m)}]=max[s_1^{(1)},s_1^{(2)},\cdots,s_1^{(l-1)},s_1^{(l)}\cdots,s_1^{(l+1)},\cdots,s_1^{(m)}]$ and initialize the space $S_0=\{[0,0,\cdots,0]\}$.

Next we proof the problem $P_m|B<n|C_{max}$ is benevolent.

Let the degree vector $D=[1,0,0,0,1,0.0.0,\cdots,1,0,0,0,\cdots,1,0,0,0]$. Note that the coordinates according to the state variable $s_1^{(l)}$ $(l=1,2,\cdots,m)$ are 1 and all other coordinates are 0.

Let $S=[s_1^{(1)},s_2^{(1)},s_3^{(1)},s_4^{(1)},\cdots,s_1^{(m)},s_2^{(m)},s_3^{(m)},s_4^{(m)}]\in S_k$
$S'=[s_1'^{(1)},s_2'^{(1)},s_3'^{(1)},s_4'^{(1)},\cdots,s_1'^{(m)},s_2'^{(m)},s_3'^{(m)},s_4'^{(m)}]\in S_k$

The dominance relation is defined:
$S\preceq_{dom}S'\Leftrightarrow s_1'^{(l)}\le s_1^{(l)}$ and $s_h'^{(l)}=s_h^{(l)}$ for $h=2,3,4;\ l=1,2,\cdots,m$

The quasi-linear order is defined:
$S\preceq_{qua}S'\Leftrightarrow \sum_{l=1}^{m}s_1'^{(l)}\le\sum_{l=1}^{m}s_1^{(l)}$ for $l=1,2,\cdots,m$

Theorem1 For any $\Delta>1$, for any $F\in\mathscr{F}$, for any $S,S'\in IN^{4m}$, the following holds:
(i) If S is $[D,\Delta]$-close to S' and if $S\preceq_{qua}S'$, then (a) $F(X,S)\preceq_{qua}F(X,S')$ holds and $F(X,S)$ is $[D,\Delta]$-close to $F(X,S')$, or (b) $F(X,S)\preceq_{dom}F(X,S')$.
(ii) If $S\preceq_{dom}S'$, then $F(X,S)\preceq_{dom}F(X,S')$. Where $X=[p_k]$. $k=1,2,\cdots,n$

Proof. (i) Consider a real number $\Delta>1$, two vectors $S=[s_1^{(1)},s_2^{(1)},s_3^{(1)},s_4^{(1)},\cdots,s_1^{(m)},s_2^{(m)},s_3^{(m)},s_4^{(m)}]$, $S'=[s_1'^{(1)},s_2'^{(1)},s_3'^{(1)},s_4'^{(1)},\cdots,s_1'^{(m)},s_2'^{(m)},s_3'^{(m)},s_4'^{(m)}]$ that fulfill S is $[D,\Delta]$-close to S' and $S\preceq_{qua}S'$. From S is $[D,\Delta]$-close to S', we get that

$$\Delta^{-1}s_1^{(l)}\le s_1'^{(l)}\le\Delta s_1^{(l)},\text{ and } s_h'^{(l)}=s_h^{(l)} \qquad \text{for } l=1,2,\cdots,m;\ h=2,3,4 \tag{1}$$

$$\text{As } S\preceq_{qua}S',\text{ so }\sum_{l=1}^{m}s_1'^{(l)}\le\sum_{l=1}^{m}s_1^{(l)}. \qquad \text{for } l=1,2,\cdots,m \tag{2}$$

From (2), we have

$$\sum_{l=1}^{m} s_1'^{(l)} \le \sum_{l=1}^{m} s_1^{(l)} \text{ and } \sum_{l=1}^{m} s_1'^{(l)} + p_k \le \sum_{l=1}^{m} s_1^{(l)} + p_k \tag{3}$$

(3) yields that $F_1^{(l)}(X,S) \preceq_{qua} F_1^{(l)}(X,S')$ and $F_2^{(l)}(X,S) \preceq_{qua} F_2^{(l)}(X,S')$ for $l = 1,2,\cdots,m$.

From S is $[D,\Delta]$-close to S' and (2), we have

$$\Delta^{-1}\sum_{l=1}^{m} s_1^{(l)} \le \sum_{l=1}^{m} s_1'^{(l)} \le \Delta\sum_{l=1}^{m} s_1^{(l)} \text{ and } s_h^{(l)} = s_h'^{(l)} \quad \text{for } l = 1,2,\cdots,m;\ \ h = 2,3,4 \tag{4}$$

From S is $[D,\Delta]$-close to S' and (3), we have

$\Delta^{-1}(\sum_{l=1}^{m} s_1^{(l)} + p_k) \le \sum_{l=1}^{m} s_1'^{(l)} \le \Delta(\sum_{l=1}^{m} s_1^{(l)} + p_k), \quad s_h^{(l)} = s_h'^{(l)} \quad$ for $l = 1,2,\cdots,m;\ \ h = 2,3,4$ (5)

(4) , (5) imply that $F_1^{(l)}(X,S)$ is $[D,\Delta]$-close to $F_1^{(l)}(X,S')$ and $F_2^{(l)}(X,S)$ is $[D,\Delta]$-close to $F_2^{(l)}(X,S')$ (for $l = 1,2,\cdots,m$). Hence, for functions $F_1^{(l)}$ and $F_2^{(l)}$, Theorem1(i) hold.

(ii) As $S \preceq_{dom} S'$, we have

$$s_1'^{(l)} \le s_1^{(l)} \text{ and } s_h'^{(l)} = s_h^{(l)} \text{ for } l = 1,2,\cdots,m;\ h = 2,3,4 \tag{6}$$

$$s_1'^{(l)} + p_k \le s_1^{(l)} + p_k \text{ and } s_h'^{(l)} = s_h^{(l)} \text{ for } l = 1,2,\cdots,m;\ h = 2,3,4 \tag{7}$$

Then (6), (7) respectively yields that $F_1^{(l)}(X,S) \preceq_{dom} F_1^{(l)}(X,S')$ and $F_2^{(l)}(X,S) \preceq_{dom} F_2^{(l)}(X,S')$ for $l = 1,2,\cdots,m$.

Theorem2 For any $\Delta > 1$, for any $H \in \mathscr{H}$, for any $S,S' \in IN^{4m}$, the following holds:
(i) If S is $[D,\Delta]$-close to S' and $S \preceq_{qua} S'$, then $H(X,S') \le H(X,S)$.
(ii) If $S \preceq_{dom} S'$, then $H(X,S') \le H(X,S)$.

Proof. (i) By S is $[D,\Delta]$-close to S' and $S \preceq_{qua} S'$, applying the definition of the quasi-order relation, we have

$H_1^{(l)}(X,S) = H_1^{(l)}(X,S')$ and $H_2^{(l)}(X,S) = H_2^{(l)}(X,S')$ for $l = 1,2,\cdots,m$.

(ii) By the definition of the dominance relation, we can easily get

$H_1^{(l)}(X,S) = H_1^{(l)}(X,S')$ and $H_2^{(l)}(X,S) = H_2^{(l)}(X,S')$ for $l = 1,2,\cdots,m$.

Theorem3 Let $g = 1$, then for any $\Delta > 1$, and for any $S,S' \in IN^{4m}$, the following holds:
(i) If S is $[D,\Delta]$-close to S' and if $S \preceq_{qua} S'$, then $G(S') \le \Delta^g G(S) = \Delta G(S)$.
(ii) If $S \preceq_{dom} S'$, then $G(S') \le G(S)$.

Proof. (i) From S is $[D,\Delta]$-close to S' and $S \preceq_{qua} S'$, we get

$\Delta^{-1}s_1^{(l)} \le s_1'^{(l)} \le \Delta s_1^{(l)}$ for $l = 1,2,\cdots,m$.

So $\Delta^{-1}\max_{1\le l\le m}\{s_1^{(l)}\} \le \max_{1\le l\le m}\{s_1'^{(l)}\} \le \Delta\max_{1\le l\le m}\{s_1^{(l)}\}$

i.e, $\Delta^{-1}G(S) \le G(S') \le \Delta G(S)$. Of course, holds $G(S') \le \Delta G(S)$.

(ii) From $S \preceq_{dom} S'$ and (6), we have

$s_1'^{(l)} \le s_1^{(l)}$ for $l = 1,2,\cdots,m$. Then holds $G(S') \le \Delta G(S)$

Theorem4

(i) Every $F \in \mathscr{F}$ can be evaluated in polynomial time. Every $H \in \mathscr{H}$ can be evaluated in polynomial time. The function G can be evaluated in polynomial time. The relation $\preceq_{qua}$ can be decided in polynomial time.

(ii) The cardinality of $\mathscr{F}$ is polynomially bounded in n and $\log\bar{x}$.

(iii) For every instance I of $P_m|B<n|C_{max}$, the state space S_0 can be computed in time that is polynomially bounded in n and $\log\bar{x}$. As a consequence, also the cardinality of the

state space S_0 is polynomially bounded in n and $\log\bar{x}$.
(iv) For an instance I of $P_m|B<n|C_{max}$, and for a coordinate l $(1 \le l \le 4m)$, let $V_l(I)$ denote the set of the l-th components of all vectors in all state spaces S_k $(1 \le k \le n)$. Then the following holds for every instance I.
For all coordinate l $(1 \le l \le 4m)$, the natural logarithm of every value in $V_l(I)$ is bounded by a polynomial $\pi_1(n,\log\bar{x})$ in n and $\log\bar{x}$. Moreover, for coordinate l with $d_l = 0$, the cardinality of $V_l(I)$ is bounded by a polynomial $\pi_2(n,\log\bar{x})$ in n and $\log\bar{x}$.
Proof. (i), (ii), (iii) are straightforward. For (iv), note that the coordinates are 0 only take the n sums of job processing or the index elements $1,2,\cdots,n$, hence, (iv) is also holds.
Based on the above dynamic programming, we gave following algorithm (named MTDP).
Algorithm MTDP.
Step 0 Delete all jobs with zero processing times and change all other processing times into integers by multipling the same parameter. We denote the new instance I'. For I' go to **Step 1**
Step 1 Initialize $\mathscr{T}_0 := S_0$
Step 2 For $k = 1$ to n do
Step 3 Let $\mathscr{U}_k := \phi$
Step 4 For every $T \in \mathscr{T}_{k-1}$ and every $F \in \mathscr{F}$ do
Step 5 If $H_F(X_k,T) \le 0$ then add $F(X_k,T)$ to $\mathscr{U}_k$
Step 6 Endfor
Step 7 Compute a trimmed copy $\mathscr{T}_k$ of $\mathscr{U}_k$
Step 8 Endfor
Step 9 Output min $\{G(S) : S \in S_n\}$
Step 10 Schedule the jobs of instance I according to I' and insert sufficient zero batches schedule jobs (deleted in **Step 0**) with zero processing times in the ahead of partial scheduling.

Theorem1-4 shows that the problem $P_m|B<n|C_{max}$ is DP-benevolent. Applying lemma1 we can get the following lemma5.
Theorem5 Algorithm MTDP is an FPTAS for problem $P_m|B<n|C_{max}$.
Proof. From Lemma1 we get that MTDP is an FPTAS for problem $P_m|B<n|C_{max}$.
Note: $|\mathscr{F}|$=4m, $\mathscr{T}_k \le \lceil(1+(\frac{2gn}{\varepsilon})\pi_1(n,\log\bar{x})+1+\pi_2(n,\log\bar{x})\rceil^{4m}$ and the running time of deciding the relation $\preceq_{qua}$ on $\mathscr{T}_k$ is $O(\frac{m(m+1)}{2})$. So the total running time of the algorithm MTDP is $O[(\frac{nm(m+1)}{2})\lceil(1+(\frac{2gn}{\varepsilon})\pi_1(n,\log\bar{x})+1+\pi_2(n,\log\bar{x})\rceil^{4m}$, where $\bar{x} = \prod_{j=1}^{n} p_j$.

References

[1] C Y. Lee, R. Uzsoy, and L. A. Martin Vega, Efficient algorithms for scheduling semiconductor burn-in operations, *Operation Research*, 1992, **40**:764-775.

[2] R. L. Graham, Lawler, J. K. Lenstra and A. H. G. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey, *Annals of Discrete Mathematics*, 1979, **5**:31-54.

[3] R.M. Karp, Reducibility among combinatorial problems, In R.E. Miller and J.W. Thatcher, editors, *Complexity of Computer Computations*, 1972, 85-104.

[4] S. Sahni, Algorithms for scheduling independent task, *Journal of the ACM* , 1976, **23**:116-127.

[5] X. Deng, C. K. Poon and Y. Zhang, Approximation algorithms in batch processing, In The 8th Annual International Symposium on Algorithms and Computation, Chennai, India , December, *Lecture Notes in Computer Science*, 1999, **1741**:153-162.

[6] G. J. Woeginger, When does a dynamic programming formulation guarantee the existence of an FPTAS, *Technical Report Woe-27, Tu-Graz, Austria*, 1998.

[7] Yuzhong Zhang, Chunsong Bai and Shouyang Wang, Duplicating and its applications in batch scheduling, *International Symposium on OR and Its Applications*, 2005, 108-110.