The Eighth International Symposium on Operations Research and Its Applications (ISORA'09)
Zhangjiajie, China, September 20–22, 2009

Approximation Schemes for Scheduling on Parallel Machines with GoS Levels*

Weidong Li[1] Jianping Li[1] Tongquan Zhang[2]

[1]Department of Mathematics, Yunnan University, Kunming 650091, PR China
[2]School of Mathematics and Computer Science
Yunnan Nationalities University, Kunming, 650031, PR China

Abstract We consider the offline scheduling problem of minimizing the makespan on m parallel and identical machines with certain feature. Every job and machine are labeled with the grade of service (GoS) levels, and each job can only be processed by the machine whose GoS level is no more than that of the job. In this paper, we present a polynomial time approximation scheme (PTAS) with running time $O(n\log n)$ for the special case where the GoS level is either 1 or 2, where the hidden constant depends exponentially on the reciprocal value of the desired precision. This solves an open problem left in [11] partially. We also present a new full polynomial time approximation scheme (FPTAS) with running time $O(n)$ for the case where the number of machines is fixed.

Keywords Approximation algorithm; GoS level; Makespan; PTAS; FPTAS

1 Introduction

Model: Given a set $\mathscr{M} = \{M_1,\dots,M_m\}$ of machines and a set $\mathscr{J} = \{J_1,\dots,J_n\}$ of jobs, each job J_j has the processing time p_j and is labelled with the grade of service (GoS) level $g(J_j)$, and each machine M_i is also labelled with the GoS level $g(M_i)$; Job J_j is allowed to be processed by machine M_i only when $g(J_j) \geq g(M_i)$. The goal is to partition the set $\mathscr{J}$ into m disjoint bundles, $S_1,\dots,S_m$, such that $\max_{1\leq i\leq m} C_i$ is minimized, where $C_i = \sum_{J_j\in S_i} p_j$ and $J_j \in S_i$ only if $g(J_j) \geq g(M_i)$. Using the three-field notation of Graham et al. [3], we denote this scheduling model as the problem $P|GoS|C_{max}$. Hwang, Chang and Lee [5] first proposed this problem and designed a strongly polynomial 2-approximation algorithm. In this paper, we consider a variant of the problem $P|GoS|C_{max}$ where $g(J_j), g(M_i) \in \{1,2\}$. We denote this problem as $P|GoS_2|C_{max}$. We also consider another variant of the problem $P|GoS|C_{max}$ where the number m of machines is fixed. We denote this problem as $P_m|GoS|C_{max}$.

Previous related work: Zhou, Jiang and He [14] proposed a $\frac{4}{3}+(\frac{1}{2})^r$-approximation algorithm for the problem $P|GoS_2|C_{max}$, where r is the desired number of iterations. Jiang [8] studied the online version of the problem $P|GoS_2|C_{max}$ and proposed an online algorithm with competitive ratio $\frac{12+4\sqrt{2}}{7}$. Ji and Cheng [7] designed an full polynomial time

*The work is fully supported by the National Natural Science Foundation of China [No.10861012], Natural Science Foundation of Yunnan Province [No.2006F0016M] and Foundation of Younger Scholar in Science and Technology of Yunnan Province [No.2007PY01-21]. E-mail addresses: jeeth@126.com (W. Li), jianping@ynu.edu.cn (J. Li), tqzh1979@yeah.net (T. Zhang).

approximation scheme (FPTAS) for the problem $P_m|GoS|C_{max}$. Note that $P_m|GoS|C_{max}$ is a special case of the unrelated parallel machines scheduling problem with fixed number of machines, which possesses many FPTASs [2, 4, 6, 12, 13]. Ou, Leung and Li [11] designed a PTAS with running time $O(mn^{2+\frac{8}{\varepsilon}log_2\frac{4}{\varepsilon}}\log P + m\log m)$ for the problem $P|GoS|C_{max}$, where $P = \sum_{j=1}^{n} p_j$. They also posed as an interesting problem the question of whether there is a PTAS for $P|GoS|C_{max}$ with improved running time. Note that the problem $P|GoS|C_{max}$ is a generalization of identical parallel scheduling problem, which possesses a linear time approximation scheme [1]. Other related results can be found in the recent survey [10].

Our contributions: In this paper, we design a PTAS with running time $O(n\log n)$ to solve $P|GoS_2|C_{max}$, which partially answers the open question of whether there is a PTAS for $P|GoS|C_{max}$ with improved running time in [11]. We also design an FPTAS with running time $O(n)$ for the problem $P_m|GoS|C_{max}$. Our FPTAS is simpler than the previous approximation schemes.

2 An Improved PTAS for $P|GoS_2|C_{max}$

Hwang et al. [5] proposed a simple 2-approximation algorithm for the problem $P|GoS|C_{max}$, called lowest grade-longest processing times first (LG-LPT, for short) algorithm, whose running time is $O(n\log n)$. For a given instance $I = (\mathscr{M}, \mathscr{J})$ of the problem $P|GoS_2|C_{max}$, let L be the value of the solution produced by LG-LPT algorithm, we have $L \le 2OPT$, where OPT denotes the optimal value of the instance $I = (\mathscr{M}, \mathscr{J})$.

Without loss of generality, we assume that $g(M_1) = g(M_2) = \cdots = g(M_k) = 1$ and $g(M_{k+1}) = g(M_{k+2}) = \cdots = g(M_m) = 2$. Let $\mathscr{J}_{[i]}$ denote the set of jobs with GoS level i, *i.e.*, $\mathscr{J}_{[i]} = \{J_j \mid g(J_j) = i\}$ $(i = 1, 2)$. The jobs in $\mathscr{J}_{[2]}$ can be processed by any machines, and the jobs in $\mathscr{J}_{[1]}$ can only be processed by the first k machines with GoS level 1.

For any given constant ε, let $\delta = \frac{1}{\lambda} = \frac{1}{\lceil\frac{5}{\varepsilon}\rceil}$. We partition the jobs into two subsets: large jobs set $\mathscr{J}^L$ and small jobs set $\mathscr{J}^S$, where $\mathscr{J}^L = \{J_j|p_j > \delta L\}$ and $\mathscr{J}^S = \{J_j|p_j \le \delta L\}$. For any given instance $I = (\mathscr{M}, \mathscr{J})$ of the problem $P|GoS_2|C_{max}$ and a given constant ε, we construct a *new* instance $\widehat{I} = (\mathscr{M}, \widehat{\mathscr{J}^L} \cup \mathscr{J}^A)$ as follows. Let $\mathscr{J}^S_{[i]} = \mathscr{J}^S \cap \mathscr{J}_{[i]}$ $(i = 1, 2)$. Thus, $\mathscr{J}^S_{[i]}$ contains those small jobs with GoS level i. Similar to [11], we replace these jobs in $\mathscr{J}^S_{[i]}$ by $\lceil \sum_{J_j \in \mathscr{J}^S_{[i]}} p_j/\delta L\rceil$ *auxiliary* jobs with GoS level i, where each auxiliary job has a processing time of δL. Let $\mathscr{J}^A_{[i]}$ be the set of auxiliary jobs with GoS level i and $\mathscr{J}^A = \mathscr{J}^A_{[1]} \cup \mathscr{J}^A_{[2]}$ be the set of all auxiliary jobs. For each job in $J_j \in \mathscr{J}^L$, we round its processing time p_j to $\widehat{p}_j$, where

$$\widehat{p}_j = \lceil \frac{p_j}{\delta^2 L} \rceil \le p_j + \delta^2 L \le (1+\delta)p_j.$$

Let $\widehat{\mathscr{J}^L}$ denote the set of jobs with scaled-up processing times. Note that all jobs in the instance $\widehat{I} = (\mathscr{M}, \widehat{\mathscr{J}^L} \cup \mathscr{J}^A)$ have processing times of the form $k\delta^2 L$, where $k \in \{\lambda, \lambda+1, \ldots, \lambda^2\}$ and $\lambda = 1/\delta$. Thus, $\widehat{\mathscr{J}^L} \cup \mathscr{J}^A$ contains jobs with at most $\lambda^2 - \lambda + 1$ different processing times. From now on, for a job in any instance, we refer it as a *large job* if it has the processing time $> \delta L$ and otherwise as a *small job*.

Lemma 1. The optimal value $\widehat{OPT}$ of the instance $\widehat{I}$ is at most $(1+\delta)OPT + \delta L \leq (1+2\delta)L$, where OPT denotes the optimal value of the instance I.

Proof. In the optimal schedule $(S_1, S_2, \cdots, S_m)$ for the instance I, replace every large job J_j in the instance I by its corresponding job $\widehat{J_j}$ in the instance $\widehat{I}$. This may increase some machine completion times by a factor of at most $1+\delta$, as $\widehat{p}_j \leq (1+\delta)p_j$. let s_i denote total processing times of small jobs which are processed by machine M_i in the solution $(S_1, S_2, \cdots, S_m)$, *i.e.*, $s_i = \sum_{J_j \in S_i \cap \mathscr{J}^S} p_j$. Clearly, we have $\sum_{i=k+1}^{m} s_i \leq \sum_{J_j \in J_{[2]}^S} p_j$. We assign auxiliary jobs with GoS level 2 in $\widehat{I}$ to the last $m-k$ machines with GoS level 2 as many as possible, subjected to that each machine is assigned at most $\lceil s_i/\delta L \rceil$ auxiliary jobs. Then, we assign at most $\lceil s_i/\delta L \rceil$ remaining auxiliary jobs to first k machines. It is easy to see that every auxiliary jobs in $\widehat{I}$ must be assigned in this way. Hence, we conclude that the completion time of machine M_i is at most $(1+\delta)OPT + \delta L \leq (1+2\delta)L$. ∎

Next, we will show how to solve instance $\widehat{I}$ optimally in linear time. The jobs in the instance $\widehat{I}$ can be represented as a set $N = \{\overrightarrow{n^i} \mid \overrightarrow{n^i} = (n^i_\lambda, n^i_{\lambda+1}, \ldots, n^i_{\lambda^2}); i = 1, 2\}$, where n^i_k $(k = \lambda, \lambda+1, \cdots, \lambda^2)$ denote the number of jobs with GoS level i whose processing times are equal to $k\delta^2 L$. An assignment to a machine is a vector $\overrightarrow{v} = (v_\lambda, v_{\lambda+1}, \ldots, v_{\lambda^2})$, where v_k $(k = \lambda, \lambda+1, \cdots, \lambda^2)$ is the number of jobs of processing time $k\delta^2 L$ assigned to that machine. The *length* of assignment $\overrightarrow{v}$ is defined as $l(\overrightarrow{v}) = \sum_{k=\lambda}^{\lambda^2} (v_k \cdot \delta^2 L)$.

Denote by F the set of all possible assignment vectors with length less than $(1+2\delta)L$, *i.e.*, $F = \{\overrightarrow{v} \mid l(\overrightarrow{v}) \leq (1+2\delta)L\}$. By the fact that the processing times of all jobs in instance $\widehat{I}$ are at least δL and Lemma 1, each assignment in F contains at most $\lambda+2$ jobs. Therefore $|F| \leq \lambda^{2\lambda+4}$ holds. Denote by ψ_i the set of all possible vectors that can be allocated to machines with GoS level i $(i = 1, 2)$, *i.e.*, $\psi_i = \{\overrightarrow{u} \in F \mid \overrightarrow{u} \leq \sum_{t=i}^{2} \overrightarrow{n^t}\}$. For every $\overrightarrow{v} \in F$, we define $\psi(i, \overrightarrow{v}) = \{\overrightarrow{u} \in \psi_i \mid l(\overrightarrow{u}) \leq l(\overrightarrow{v})\}$. For every vector $\overrightarrow{u} \in \psi(i, \overrightarrow{v})$, let $x_i^{\overrightarrow{u}}$ be the numbers of machines with GoS level i that are assigned $\overrightarrow{u}$ in $\psi(i, \overrightarrow{v})$. For every $\overrightarrow{v} \in F$, we construct an integer linear programming ILP($\overrightarrow{v}$) with arbitrary objective function, and that the corresponding constraints are:

$$\sum_{\overrightarrow{u} \in \psi(1, \overrightarrow{v})} x_1^{\overrightarrow{u}} = k; \tag{1}$$

$$\sum_{\overrightarrow{u} \in \psi(2, \overrightarrow{v})} x_2^{\overrightarrow{u}} = m-k; \tag{2}$$

$$\sum_{\overrightarrow{u} \in \psi(1, \overrightarrow{v})} x_1^{\overrightarrow{u}} \overrightarrow{u} + \sum_{\overrightarrow{u} \in \psi(2, \overrightarrow{v})} x_2^{\overrightarrow{u}} \overrightarrow{u} = \overrightarrow{n^1} + \overrightarrow{n^2}; \tag{3}$$

$$\sum_{\overrightarrow{u} \in \psi(2, \overrightarrow{v})} x_2^{\overrightarrow{u}} \overrightarrow{u} \leq \overrightarrow{n^2}; \tag{4}$$

$$x_i^{\overrightarrow{u}} \in \mathbb{Z}^+ \cup \{0\} \qquad \forall \overrightarrow{u} \in \psi_i^{\overrightarrow{v}}; i = 1, 2 \tag{5}$$

Here, the constraints (1) and (2) guarantee that each machine is assigned exactly one vector (a set of jobs), and the constraint (3) guarantees that each job is used in exactly once. The constraint (4) guarantees that the machines with GoS level 2 are assigned jobs with GoS level 2. For every $\overrightarrow{v} \in F$, the number of variables in the integer linear programming is at most $|\psi(1,\overrightarrow{v})|+|\psi(2,\overrightarrow{v})|\leq 2|F|\leq 2\lambda^{2\lambda+4}$, and that the number of constrains is at most $2+2(\lambda^2-\lambda+1)+|\psi(1,\overrightarrow{v})|+|\psi(2,\overrightarrow{v})|\leq 2\lambda^2-2\lambda+4+2\lambda^{2\lambda+4}$. Both values are constants, as λ is a fixed constant.

By utilizing Lenstra's algorithm in [9] whose running time is exponential in the dimension of the program but polynomial in the logarithms of the coefficients, we can decide whether the integer linear programming ILP ($\overrightarrow{v}$) has a feasible solution in time $O(log^{O(1)}n)$, where the hidden constant depends exponentially on λ. And since the integer linear programming ILP($\overrightarrow{v}$) can be constructed in $O(n)$ time, we can find the optimal value $\widehat{OPT}=\min\{l(\overrightarrow{v})|$ ILP($\overrightarrow{v}$) has a feasible solution $\}$ of the instance $\widehat{I}$ in time $O(|F|(log^{O(1)}n+n))=O(n)$. Hence, the following lemma holds.

Lemma 2. For any fixed integer λ, an optimal solution for the new instance $\widehat{I}=(\mathscr{M},\widehat{\mathscr{J}}^L\cup\mathscr{J}^A)$ corresponding to the given instance $I=(\mathscr{M},\mathscr{J})$ of the problem $P|GoS_2|C_{max}$ can be computed in time $O(n)$, where the hidden constant depends exponentially on λ. ∎

Lemma 3. There exists a schedule for the instance I with maximum machine completion time at most $\widehat{OPT}+\delta L$. ∎

Our approximation scheme for the problem $P|GoS_2|C_{max}$ can formulated as follows. For any given instance I, we first compute a bound on OPT using LG-LPT algorithm [5] in time $O(n\log n)$, and construct a corresponding instance $\widehat{I}$ in time $O(n)$. Then, compute an optimal solution for the instance $\widehat{I}$ in time $O(n)$ by utilizing Lenstra's algorithm. Finally, we output a feasible solution for instance I as in Lemma 3. Let OUT be the value of output solution, by Lemma 3, we have $OUT\leq\widehat{OPT}+\delta L$. By Lemma 1, we have $\widehat{OPT}\leq(1+\delta)OPT+\delta L$. Hence, we have $OUT\leq(1+\delta)OPT+2\delta L\leq(1+5\delta)OPT\leq(1+\varepsilon)OPT$, as $L\leq 2OPT$ and $\delta=\frac{1}{\lceil\frac{5}{\varepsilon}\rceil}\leq\frac{\varepsilon}{5}$.

Hence, we achieve our main result as follows.

Theorem 1. The problem $P|GoS_2|C_{max}$ possesses a PTAS with running time $O(n\log n)$, where the hidden constant depends exponentially on $\frac{1}{\varepsilon}$. ∎

3 A new FPTAS for $P_m|GoS|C_{max}$

Although there exists many FPTASs for the problem $P_m|GoS|C_{max}$, only two of them have running time $O(n)$ [2, 6]. As the FPTASs with running time $O(n)$ in [2, 6] are designed for a more general problem, they are complex. In this section, we investigate the special structure of the problem $P_m|GoS|C_{max}$ and design a simpler FPTAS with running time $O(n)$.

Assume that all the machines and jobs are sorted in nondecreasing order of their GoS levels, thus we have $g(M_1)\leq g(M_2)\leq\cdots\leq g(M_m)$ and $g(J_1)\leq g(J_2)\leq\cdots\leq g(J_n)$. For convenience, we denote by $P=\sum_{j=1}^n p_j$ the total processing time, and for each job J_k, we define $v_k=\max\{i\mid g(M_i)\leq g(J_k)\}$.

We first present a standard dynamic program, which can also be found in [13]. In the dynamic program, we will store certain information for certain schedules for the first k jobs ($1\leq k\leq n$): Every such schedule is encoded by a m-dimensional vector $(P_1,P_2,\ldots,P_m)$, where P_i specifies the overall processing times of all jobs assigned to machines M_i for $i=1,2,\ldots,m$. The state space ψ_k consists of all m-dimensional vectors for schedule for the first k job, where for $k=0$, the state space ψ_0 contains only one element $(0,0,\ldots,0)$. For $k \geq 1$, every schedule $(P_1,P_2,\ldots,P_m)$ in state space ψ_{k-1} can be extended in v_k ways by placing job J_k to the machine M_i $(1 \leq i \leq v_k)$. This yields v_k schedules:

$$(P_1,P_2,\ldots,P_m)+p_k e_i; \qquad i=1,2,\ldots,v_k$$

where e_i denotes the m-dimensional vector whose coordinates are 0 except for the ith coordinate as 1. We put these v_k schedules into the state space ψ_k. In the end, the optimal objective value is

$$min\{z \mid \exists (P_1,P_2,\ldots,P_m) \in \psi_n, \text{ which satisfies } \max_{1\leq i\leq m}\{P_i\} \leq z\}.$$

Here, the computational complexity of this dynamic progrmming formulation is clearly $O(nP^m)$, which is a pseudo-polynomial time. As mentioned in [13], following the framework of Woeginger [12], the problem $P_m|GoS|C_{max}$ has an FPTAS. However, the running time is not linear. We will present a new FPTAS with running time $O(n)$ by utilizing a new method of handling small jobs.

As in [12], we iteratively thin out the state space of the dynamic program, and collapse solutions that are close to each other, and then bring the size of the state space down to polynomial size. The *trimming* parameter $\triangle$ in our paper here is defined as $\triangle = \frac{\varepsilon^2 P}{4m^2}$, where $\varepsilon > 0$ and $P = \sum_{j=1}^{n} p_j$.

Different from the definition of $\triangle$-domination in [12], we call that a state $s' = (P'_1, P'_2, \ldots, P'_m)$ is $\triangle$-*dominated* by the state $s = (P_1,P_2,\ldots,P_m)$ if and only if

$$P_i - \triangle \leq P'_i \leq P_i + \triangle, \qquad \text{for each } i=1,2,\ldots,m \tag{6}$$

For each state space ψ_k, if the state $s' \in \psi_k$ is dominated by the state $s \in \psi_k$, we remove state s' from ψ_k. Finally, we will get the trimmed state space ψ_k^*. Whenever we compute a new state space ψ_k^* in the trimmed dynamic program, we start from the trimmed state space ψ_{k-1}^* instead of ψ_{k-1} in the original dynamic program.

We can obtain the following results.

Lemma 4. $\mid \psi_k^* \mid = \mathscr{O}(1)$, where $\mid \psi_k^* \mid$ is the cardinality of the trimmed state space ψ_k^* and m is fixed. ∎

Lemma 5. For each state $s' = (P'_1,P'_2,\ldots,P'_m)$ in the original state space ψ_k, there is a state $s = (P_1,P_2,\ldots,P_m)$ in the trimmed state space ψ_k^* which satisfies $P_i - k\triangle \leq P'_i \leq P_i + k\triangle$ $(i=1,2,\ldots,m)$. ∎

We now present our FPTAS for the problem $P_m|GoS|C_{max}$ in the following structural form:

Algorithm: FPTAS
Begin
Step 1 If $(n \leq \frac{2m}{\varepsilon})$ then
Step 1.1 Choose the schedule $(S_1,S_2,\ldots,S_m)$ corresponding to the best state $(P_1, P_2, \ldots, P_m)$ from the final trimmed state space ψ_n^*.
Step 2 If $(n > \frac{2m}{\varepsilon})$ then
Step 2.1 Denote $K = \frac{2m}{\varepsilon}$;
Step 2.2 Choose the first K longest jobs denoted by $L = \{J_{i_1}, J_{i_2}, \ldots, J_{i_K}\}$;

Step 2.3 For the jobs in L, we compute the trimmed state space ψ_K^* in the preceding discuss;
Step 2.4 For each state in ψ_K^*, we assign each job J_j in $\mathscr{J} - L$ to the least-loaded machine M_i with $g(M_i) \leq g(J_j)$ according the increasing order of the GoS level;
Step 2.5 Choose the schedule $(S_1, S_2, \ldots, S_m)$ corresponding to the best solution from the $\mid \psi_K^* \mid$ feasible solutions.
Step 3 Output the solution in either the step 1 or the step 2.
End of Algorithm FPTAS

Lemma 6. In Step 2 of the algorithm FPTAS, the processing time of any job in $J - L$ is at most $\frac{\varepsilon \cdot OPT}{2}$, where OPT denotes the objective value of the optimal solution.

Let $(S_1^*, S_2^*, \ldots, S_m^*)$ be an optimal schedule for the problem $P_m|GoS|C_{max}$ with a fixed number of machines. We show our result in the following theorem.

Theorem 2. The algorithm FPTAS produces a solution for the problem $P_m|GoS|C_{max}$ in which the maximum machine complete time is at most $(1+\varepsilon)OPT$, and the computational complexity is linear, *i.e.*, $O(n)$, where the hidden constant depends exponentially on m.

Proof. We prove the assertion in the following two different cases.

Case 1. $n \leq \frac{2m}{\varepsilon}$. For the state $(P_1^*, P_2^*, \ldots, P_m^*)$ corresponding to the optimal schedule $(S_1^*, S_2^*, \ldots, S_m^*)$, by Lemma 5, there is a state $s' = (P_1', P_2', \ldots, P_m')$ in the trimmed state space ψ_n^* which satisfies

$$P_i' \leq P_i^* + n\triangle \leq P_i^* + \frac{2m}{\varepsilon}\frac{\varepsilon^2 P}{4m^2} \leq P_i^* + \varepsilon OPT, \quad i = 1, 2, \ldots, m$$

The last inequality comes from the fact that $P \leq m \cdot OPT$. Since we choose the best state $(P_1, P_2, \ldots, P_m)$ from ψ_n^*, we obtain

$$\max_{1 \leq i \leq m} \{P_i\} \leq \max_{1 \leq i \leq m} \{P_i'\} \leq \max_{1 \leq i \leq m} \{P_i^*\} + \varepsilon OPT \leq (1+\varepsilon) \cdot OPT$$

Case 2. $n > \frac{2m}{\varepsilon}$. For the state $(P_1^{*L}, P_2^{*L}, \ldots, P_m^{*L})$ corresponding to the subschedule $(S_1^* \cap L, S_2^* \cap L, \ldots, S_m^* \cap L)$, by Lemma 5, there is a state $(P_1'^L, P_2'^L, \ldots, P_m'^L)$ in the trimmed state space ψ_K^* which satisfies

$$P_i'^L \leq P_i^{*L} + K\triangle = P_i^{*L} + \frac{2m}{\varepsilon}\frac{\varepsilon^2 P}{4m^2} \leq P_i^{*L} + \frac{\varepsilon}{2}OPT, \quad i = 1, 2, \ldots, m$$

Consider the solution $(P_1', P_2', \ldots, P_m')$ which is obtained by assigning the jobs in $\mathscr{J} - L$ to the state $(P_1'^L, P_2'^L, \ldots, P_m'^L)$. Suppose that M_t is the machine with the maximum machine complete time and J_j is the last job in $\mathscr{J} - L$ assigned to the machine M_t. We notice that each job in $\mathscr{J} - L$ assigned before job J_j must be assigned to one of the first v_j machines in any schedule, because we assign the jobs in $\mathscr{J} - L$ according the increasing order of the GoS level. Let $S = \{J_k \mid J_k \in \mathscr{J} - L \text{ and } J_k \text{ is assigned before } J_j\}$. Since we choose the least-loaded machine M_t when we assign job J_j, we obtain the following result:

$$
\begin{aligned}
P'_t &= P'_t - p_j + p_j \le \frac{\sum_{i=1}^{v_j} P_i'^L + l(S)}{v_j} + \frac{\varepsilon}{2}OPT && (7)\\
&\le \frac{\sum_{i=1}^{v_j}(P_i^{*L} + \frac{\varepsilon}{2}OPT) + l(S)}{v_j} + \frac{\varepsilon}{2}OPT && (8)\\
&= \frac{\sum_{i=1}^{v_j} P_i^{*L} + l(S) + \frac{v_j\varepsilon}{2}OPT}{v_j} + \frac{\varepsilon}{2}OPT && (9)\\
&\le \frac{\sum_{i=1}^{v_j} P_i^{*}}{v_j} + \varepsilon OPT \le (1+\varepsilon)OPT && (10)
\end{aligned}
$$

The inequality (7) comes from the fact that machine M_t is the least-loaded machine when we assign the job J_j, the inequality (8) comes from Lemma 5 and the last inequality comes from the fact that average load is no more than OPT.

Since we choose the best schedule $(S_1, S_2, \ldots, S_m)$ from the $\mid \psi_K^* \mid$ feasible solutions, we obtain

$$\max_{1\le i\le m}\{P_i\} \le \max_{1\le i\le m}\{P'_i\} = P'_t \le (1+\varepsilon)OPT$$

Thus, from the results in the preceding both cases, we obtain the fact that the objective value of the output solution $(S_1, S_2, \ldots, S_m)$ is no more than $(1+\varepsilon)OPT$.

It is easy to verify that the computational complexity in the case 1 is bounded by $(\frac{P}{\triangle})^m = \mathscr{O}(1)$, by the fact that m is fixed. And for the computational complexity in the case 2, we can obtain the time-consuming: (1) the step 2.2 can be executed in $\mathscr{O}(\frac{mn}{\varepsilon})$; (2) by Lemma 4, the step 2.3 can be executed in $\mathscr{O}(\frac{m}{\varepsilon})$; (3) the step 2.4 can be executed in $\mathscr{O}(n)$. Therefore, the overall computational complexity of the algorithm FPTAS is totally $\mathscr{O}(n)$, where $\varepsilon > 0$ and m are fixed numbers.

Hence, the theorem holds. ∎

4 Conclusion

In this paper, we design a PTAS with running time $O(n\log n)$ for the problem $P|GoS_2|C_{max}$ which is a special case of the problem $P|GoS|C_{max}$. An immediate problem is whether there is a PTAS with running time $O(n\log n)$ for the problem $P|GoS|C_{max}$. It is not clear whether the method in current paper can be extended to $P|GoS|C_{max}$. This may be a direction of the future work although the method does not seem extensible to $P|GoS|C_{max}$.

We also present a new simpler FPTAS with running time $O(n)$ for the problem $P_m|GoS|C_{max}$. The technique of handling small jobs in our FPTAS has independent interest and may be found other applications in the scheduling model under a grade of service provision.

References

[1] N. Alon, Y. Azar, G.J. Woeginger, and T. Yadid. Approximation schemes for scheduling on parallel machines. *Journal of Scheduling*, 1, 55-66, 1998.

[2] A. Fishkin, K. Jansen, and M. Mastrolilli. Grouping techniques for scheduling problems: simpler and faster. *Proceedings of the 9th Annual European Symposium on Algorithms*, 206-217, 2001.

[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, *Annals of Discrete Mathematics*, 5, 287-326, 1979.

[4] E. Horowitz and S. Sahni. Exact and approximate algorithms for scheduling nonidentical processors, *Journal of the ACM*, 23, 317-327, 1976.

[5] H.C. Hwang, S.Y. Chang, and K. Lee. Parallel machine scheduling under a grade of service provision, *Computers and Operations Research*, 31, 2055-2061, 2004.

[6] K. Jansen and L. Porkolab. Improved approximation schemes for scheduling unrelated parallel machines. *31st Annual ACM Symposium on Theory of Computing* (STOC), 408-417, 1999.

[7] M. Ji and T.C.E. Cheng. An FPTAS for parallel machine scheduling under a grade of service provision to minimize makespan. *Information Processing Letters*, 108(4), 171-174, 2008.

[8] Y. Jiang. On line scheduling on parallel machines with two GoS levels. *Journal of Combinatorial Optimization*, 16(1), 28-38, 2008.

[9] H.W. Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8, 538-548, 1983.

[10] J.Y.-T. Leung and C.-L. Li. Scheduling with processing set restrictions: A survey. *International Journal of Production Economics*, 116(2), 251-262, 2008.

[11] J. Ou, J.Y.-T. Leung, and C.-L. Li. Scheduling parallel machines with inclusive processing set restrictions. *Naval Research Logistics*, 55(4), 328-338, 2008.

[12] G.J. Woeginger. When does a dynamic programming formulation guarantee the existence of a fully polynomial time approximation scheme (FPTAS)?, *Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms* (SODA), 820-829, 1999.

[13] G.J. Woeginger. A comment on parallel-machine scheduling under a grade of service provision to minimize makespan. *Information Processing Letters*, 109(7), 341-342, 2009.

[14] P. Zhou, Y. Jiang, and Y. He. Parallel machine scheduling problem with two GoS levels. *Applied Mathematics: A Journal of Chinese Universities (Series A)*, 22(3), 275-284 (in Chinese), 2007.