AN APPROXIMATION ALGORITHM FOR SINGLE SERVER PARALLEL MACHINE PROBLEM*

SVETLANA A. KRAVCHENKO†, FRANK WERNER‡

†Institute of Engineering Cybernetics, Surganov St. 6, 220012 Minsk, Belarus, kravch@newman.bas-net.by

‡Otto-von-Guericke-Universität, Fakultät für Mathematik, PSF 4120, 39016 Magdeburg, Germany, frank.werner@mathematik.uni-magdeburg.de

Abstract. In this note we consider the problem of scheduling a set of jobs on m identical parallel machines. For each job, a setup has to be done by a single server. The objective is to minimize the sum of the completion times in the case of unit setup times and arbitrary processing times. For this strongly NP-hard problem, we give an approximation algorithm with an absolute error bounded by the product of the number of short jobs (with processing times less than m - 1) and m - 2.

Key Words. Sheduling, parallel machines, single server, unit setup times, total completion time, approximation algorithm

1.INTRODUCTION

In this paper, we consider the problem of scheduling n jobs on a set of m identical parallel machines available for processing at time zero. The processing of a job must be performed on one of the machines without interruption. Before the processing of a job can start, a setup must be performed for this job by a server, which corresponds to a loading of this job on the corresponding machine. Such a setup is not possible during the processing of another job on the corresponding machine. If a server has performed a setup on a machine, the processing of this job can immediately start, but on the other hand, the server is free to perform another setup on another machine. It is assumed that travel times of a server between the machines are equal to zero.

Classical scheduling objectives in connection with this type of problems have been considered by Hall, Potts & Sriskandarajah [3]. In that paper, a study of algorithmic and computational complexity is performed. In dependence on the objective function, the number of machines, and the structure of the processing and setup times, for the majority of these problems there is either given an efficient algorithm, or it has been proved that the corresponding problem is NP-hard which implies that the existence of such an efficient algorithm is rather unlikely. Some of the efficient algorithms have been obtained by adapting classical scheduling algorithms to the corresponding problem with a single server.

A beam search heuristic for parallel machine scheduling problems with a single server in such a static environment described above has been suggested by Koulamas [4]. Kravchenko & Werner [5] consider parallel machine scheduling problems with a single server and the minimization of the makespan as well as of the forced idle time (or interference). In the latter paper, a pseudopolynomial algorithm for the 2-machine problem with unit setup times is given. Moreover, some complexity issues, polynomially special cases and heuristics are discussed. Problems with multiple servers have been considered by Kravchenko & Werner [6].

*) This research was supported by the International Association for the Promotion of Cooperation with Scientists from the Independent States of the Former Soviet Union, Project INTAS-96-820 and by Belarussian Fundamental Research Foundation

In [1], it has been proven that the problem of scheduling a set of jobs on m identical parallel machines $M_1,\ldots,M_m$ with a single server such that the sum of completion times becomes minimal is already strongly NP-hard in the case of unit setup times. The latter problem may be denoted as $P,S1|s_i = 1|\sum C_i$ (see [1, 2]). In this note, we present an approximation algorithm and prove that the derived performance bound is tight.

2. THE ALGORITHM

It is easy to see that an optimal schedule for problem $P,S1|s_i = 1|\sum C_i$ can be found within the class of list schedules. Further we work only with such schedules. The construction of any list schedule consists of n steps: In step i, we schedule the i-th job from the list at the earliest possible time in the partial schedule created in the previous $(i-1)$-th step. We suppose that before the first step, we have a partial schedule with the profile $(0,1,\ldots,m-1)$, i.e. before the first step the machine M_i is available for processing at time $i-1$, where $i\in\{1,\ldots,m\}$.

We say that in some step job j creates a conflict if, after scheduling this job j into the partial schedule obtained in the previous step, there is a scheduled job i such that $C_i=C_j$ holds.

Algorithm 1

Step i; (i=1,...,n).
Among all unscheduled jobs choose the shortest job j which does not create a conflict.
If this is impossible, take any unscheduled job j.
Schedule j in the partial schedule.

Denote by $\sum\tilde{C}_i$ the sum of completion times for the schedule created by Algorithm 1, and by $\sum C_i^*$ the sum of completion times for an optimal schedule. Let $n'=|\{ i| p_i<m-1\}|$ be the number of jobs with the total length (p_i+s_i) being less than the number of machines.

Theorem. Algorithm 1 creates a schedule for which the following estimation holds:

$$\sum_{i=1}^{n}\tilde{C}_i-\sum_{i=1}^{n}C_i^*\le n'(m-2).$$

Proof: Assume that Algorithm 1 generates some schedule s. For convenience, divide all jobs into two sets. Namely, the set of jobs $\{i| p_i<m-1\}$ (further we denote these jobs as short jobs), and the set of jobs $\{ i| p_i\ge m-1\}$ (further we denote them as long jobs).

There is some step in Algorithm 1, say step z, when the last short job is scheduled in s. Note that, if there are no short jobs, Algorithm 1 turns out to be SPT rule, and in this case it produces an optimal schedule. Now we show that the server works without idle times during the interval $[0,z+1]$ in schedule s.

Let $q\le z$ be the first step of Algorithm 1 when an idle time is created, i.e. in $[0,q]$ the server works without being idle, and in $[q,q+1]$ the server is idle. It means that all m machines are working in $[q,q+1]$ and therefore, there is no job finished at time q. Then, it is possible to show that the set

$$J'=\{ i| p_i<m-1 \text{ and } i \text{ is unscheduled up to step } q+1\}$$

is empty. Really, J' cannot contain job i with $p_i=k$, where $0\le k\le m-1$, since otherwise in step $q-k$ of Algorithm 1 such a job i has to be scheduled, but this is impossible since at time q there is no job finishing its processing. Thus, set J' is empty at step q, and we obtain a contradiction to the fact that some short job is scheduled in step z of Algorithm 1.

Now we show that all long jobs $\{i|p_i\ge m-1\}$ are scheduled by Algorithm 1 in nondecreasing order of their processing times. Since the algorithm chooses the shortest job which can be scheduled by the list procedure without a conflict, in each step we get some partial schedule with a profile $(t_1,\ldots,t_m)$, where t_i denotes the time when machine M_i becomes free and

$$\max_i\{t_i\}-\min_i\{t_i\}<\min\{p_i| p_i\ge m-1 \text{ and } i \text{ is unscheduled }\}$$

holds, i.e. in any step of Algorithm 1, each unscheduled long job does not create a conflict. Hence, if Algorithm 1 schedules a long job, it chooses the shortest unscheduled long job. Thus, all jobs $\{ i |p_i \ge m-1\}$ are scheduled in nondecreasing order of their processing times.

In schedule s produced by Algorithm 1, we enumerate all jobs in nondecreasing order of their processing times, say $\{ i_1,\ldots,i_n\}$, i.e.

$$p_{i_1}\le p_{i_2}\le\ldots\le p_{i_n}.$$

In an optimal schedule we enumerate all jobs in nondecreasing order of their completion times, say $j_1,\ldots,j_n$, i.e.

$$C_{j_1}^*\le C_{j_2}^*\le\ldots\le C_{j_n}^*.$$

In fact, in the previous two paragraphs we proved that in the first z steps Algorithm 1 schedules all short jobs and $z-n'$ long jobs, i.e. all jobs $\{i_1, i_2,\ldots, i_z\}$ are scheduled during the first z steps of Algorithm 1. Since the server is not idle during the interval $[0,z+1]$, we get that

$$\sum_{k=1}^{g} \widetilde{C}_{i_k} = 0 + 1 + \ldots + (g-1) + \sum_{k=1}^{g} s_{i_k} + \sum_{k=1}^{g} p_{i_k} \qquad (1)$$

holds for any $1 \le g \le z+1$. Note that

$$\sum_{i=1}^{q} C_i = \sum_{i=1}^{q} t_i^0 + \sum_{i=1}^{q} s_i + \sum_{i=1}^{q} p_i \ge 0 + 1 + \ldots + (q-1) + \sum_{i=1}^{q} s_i + \sum_{k=1}^{q} p_{i_k}$$

holds for any schedule and any $1 \le q \le n$. Hence, by proving (1) we show that

$$\sum_{k=1}^{g} \widetilde{C}_{i_k} \le \sum_{k=1}^{g} C^*_{j_k}$$

holds for any $1 \le g \le z+1$.

Next, we show that, if there is an idle time on some of the machines, then Algorithm 1 produces an optimal schedule. Suppose that under the schedule s, there is an idle time on some machine. Hence, in some step, say step k, all jobs would create a conflict. But any long job cannot create a conflict, and therefore in step k there are no long jobs. Therefore, the server works without idle times and Algorithm 1 produces an optimal schedule.

Now we consider the case when under schedule s, all machines work without idle times. Since only list schedules are considered and all long jobs are scheduled in nondecreasing order of their processing times, each job i_k is scheduled in step k of Algorithm 1 if $k>z$ holds. Remind that z is the number of the step when the last short job is scheduled. Taking into account that there is no idle time in schedule s and that

$$p_{i_1} \le p_{i_2} \le \ldots \le p_{i_n}$$

holds, we obtain that

$$\widetilde{C}_{i_n} + \widetilde{C}_{i_{n-1}} + \ldots + \widetilde{C}_{i_{n-m+1}} = 0 + 1 + \ldots + (m-1) + \sum_{k=1}^{n} s_{i_k} + \sum_{k=1}^{n} p_{i_k} \le C^*_{j_n} + C^*_{j_{n-1}} + \ldots + C^*_{j_{n-m+1}}$$

holds, if $n-m+1>z$, and

$$\widetilde{C}_{i_{n-m}} + \widetilde{C}_{i_{n-m-1}} + \ldots + \widetilde{C}_{i_{n-2m+1}} = 0 + 1 + \ldots + (m-1) + \sum_{k=1}^{n-m} s_{i_k} + \sum_{k=1}^{n-m} p_{i_k} \le C^*_{j_{n-m}} + C^*_{j_{n-m-1}} + \ldots + C^*_{j_{n-2m+1}}$$

holds, if $n-2m+1>z$. In general,

$$\widetilde{C}_{i_{n-(q-1)m}} + \widetilde{C}_{i_{n-(q-1)m-1}} + \ldots + \widetilde{C}_{i_{n-qm+1}} = 0 + 1 + \ldots + (m-1) + \sum_{k=1}^{n-(q-1)m} s_{i_k} + \sum_{k=1}^{n-(q-1)m} p_{i_k} \le C^*_{j_{n-(q-1)m}} + C^*_{j_{n-(q-1)m-1}} + \ldots + C^*_{j_{n-qm+1}} \qquad (2)$$

holds for any q such that $n-qm+1>z$.

Now we show that, if there is a value, say d, such that $n-dm+1=z+1$ or $n-dm+1=z+2$ holds, then the schedule s is optimal. Let $n-dm+1=y$, where

$$y \in \{z+1, z+2\}$$

holds. Then it follows from (2) that

$$\widetilde{C}_{i_{n-(q-1)m}} + \widetilde{C}_{i_{n-(q-1)m-1}} + \ldots + \widetilde{C}_{i_{n-qm+1}} \le C^*_{j_{n-(q-1)m}} + C^*_{j_{n-(q-1)m-1}} + \ldots + C^*_{j_{n-qm+1}}$$

holds for any $q \in [1,d]$. From (1) it follows that

$$\sum_{k=1}^{y-1} \widetilde{C}_{i_k} \le \sum_{k=1}^{y-1} C^*_{j_k}$$

holds. Therefore, inequality

$$\sum_{k=1}^{n} \widetilde{C}_{i_k} \le \sum_{k=1}^{n} C^*_{j_k}$$

holds.

Now we consider the case when there is no integer q, such that $n-qm+1=z+1$ or $n-qm+1=z+2$ holds. Then there is a value, say d, such that $n-dm+1>z+2$ and $n-(d+1)m+1<z$ hold. Consider all long jobs from the set

$$\{i_{n-dm}, i_{n-dm-1}, \ldots, i_{n-dm-(m-1)}\},$$

without loss of generality we will suppose that they all are long. Since $n-dm+1>z+2$ holds, job i_{n-dm} is scheduled in step $n-dm$, and job i_{n-dm-1} is scheduled in step $n-dm-1$. In step $n-dm-2$, some short job will be scheduled, if $n-dm-2=z$ holds. Denote all short jobs scheduled between job $i_{n-dm-(m-1)}$ and job i_{n-dm-1} by

$$u_1, u_2, \ldots, u_r.$$

Note that

$$\{u_1, \ldots, u_r\} \subseteq \{i_1, \ldots, i_{n-(d+1)m}\}$$

holds, and all jobs from the set

$$\{i_1, \ldots, i_{n-(d+1)m}\} \setminus \{u_1, \ldots, u_r\}$$

are scheduled in the steps 1, 2,..., $n-(d+1)m-r$ of Algorithm 1, i.e. the server performs the setups for the jobs of this set in the interval

$$[0, n-(d+1)m-r].$$

In the interval

$[n-(d+1)m-r,\ n-dm-2]$,

the server performs the setups for the jobs

$i_{n-dm-2},\ldots,i_{n-dm-(m-1)}$

and jobs

$u_1,\ldots,u_r$.

It means that the setup of job u_r is started no later than at time $n-dm-2$, i.e.

$$t^0_{u_r} \le n-dm-2;$$

the setup of job u_{r-1} is started no later than at time $n-dm-3$, i.e.

$$t^0_{u_{r-1}} \le n-dm-3;$$
$$\ldots;$$

the setup of job u_2 is started no later than at time $n-dm-2-(r-2)$, i.e.

$$t^0_{u_2} \le n-dm-2-(r-2);$$

and the setup of job u_1 is started no later than at time $n-dm-2-(r-1)$, i.e.

$$t^0_{u_1} \le n-dm-2-(r-1).$$

In other words, the inequalities

$$t^0_{u_r} \le n-(d+1)m-r+r+(m-2);$$
$$t^0_{u_{r-1}} \le n-(d+1)m-r+(r-1)+(m-2);$$
$$\ldots;$$
$$t^0_{u_2} \le n-(d+1)m-r+2+(m-2);$$
$$t^0_{u_1} \le n-(d+1)m-r+1+(m-2)$$

hold, i.e. we get

$$\sum_{k=1}^{n-(d+1)m} \widetilde{C}_{i_k} = \sum_{k=1}^{n-(d+1)m} t^0_{i_k} + \sum_{k=1}^{n-(d+1)m} s_{i_k} + \sum_{k=1}^{n-(d-1)m} p_{i_k} \le$$
$$0+\ldots+(n-(d+1)m-1) + \sum_{k=1}^{n-(d+1)m} s_{i_k} + \sum_{k=1}^{n-(d+1)m} p_{i_k} +$$
$$r(m-2) \le \sum_{k=1}^{n-(d+1)m} C^*_{j_k} + r(m-2).$$

Since inequality

$$\sum_{k=n-(d+1)m+1}^{n} \widetilde{C}_{i_k} \le \sum_{k=n-(d+1)m+1}^{n} C^*_{j_k}$$

holds, we obtain that

$$\sum_{k=1}^{n} \widetilde{C}_{i_k} - \sum_{k=1}^{n} C^*_{i_k} \le r(m-2)$$

holds. Because we have $r \le n'$, the theorem has been proved. ■

Now we show that the above bound is tight. Suppose we have an instance with $3m$-1 jobs, there are $2(m$-2) jobs with

$p_1=\ldots=p_{2(m-2)}=m$-1,

there are two jobs

$p_{2m-3}=p_{2m-2}=0$,

and there are m+1 jobs with

$p_{2m-1}=p_{2m}=\ldots=p_{3m-1}=3m$-3.

An optimal schedule is obtained by the list

$2m$-1, 1, ... , m-2, $2m$-3, m-1, ...,
$2(m$-2), $2m$-2, $2m$,..., $3m$-1.

In this schedule the server works without idle times in $[0,3m$-1]. Thus, for the optimal function value we have

$$\sum_{i=1}^{3m-1} C^*_i = \sum_{i=1}^{3m-1} t^0_i + \sum_{i=1}^{3m-1} s_i + \sum_{i=1}^{3m-1} p_i = 0+1+\ldots+$$
$$(3m-2) + \sum_{i=1}^{3m-1} s_i + \sum_{i=1}^{3m-1} p_i.$$

However, Algorithm 1 schedules all jobs by the list

1, ..., $2(m$-2), $2m$-1, ..., $3m$-3,
$2m$-3, $2m$-2, $3m$-2, $3m$-1.

The server is working without idle times during $[0,3m$-2] and $[5m$-6,$5m$-5]. For the value $\sum \widetilde{C}_i$ we have

$$\sum_{i=1}^{3m-1} \widetilde{C}_i = \sum_{i=1}^{3m-1} t^0_i + \sum_{i=1}^{3m-1} s_i + \sum_{i=1}^{3m-1} p_i = 0+1+\ldots+$$
$$3m-3+5m-6+\sum_{i=1}^{3m-1} s_i + \sum_{i=1}^{3m-1} p_i.$$

Therefore, $\sum \widetilde{C}_i - \sum C^*_i = 2(m-2)$ holds.

Consider the example with m=5. There are 14 jobs: six jobs with $p_1=\ldots=p_6=4$, two jobs with $p_7=p_8=0$,

and six jobs with $p_9=...=p_{14}=12$. An optimal schedule is obtained by the list 9, 1, 2, 3, 7, 4, 5, 6, 8, 10, 11, 12, 13, 14, and the server works without idle times during [0,14]. Algorithm 1 schedules all jobs by the list

1, 2, 3, 4, 5, 6, 9, 10, 11, 12, 7, 8, 13, 14.

Under this schedule the server works without idle times during [0,13] and [19,20]. Thus,

$$\Sigma \tilde{C}_i - \Sigma C_i^* = 6 = n'(m-2)$$

holds.

5. REFERENCES

1. Brucker P., Dhaenens-Flipo C., Knust S., Kravchenko S.A., Werner F. Complexity results for parallel machine problems with a single server, Preprint, Heft 219, Osnabrücker schriften zur Mathematik, 2000.
2. Graham R.L., Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G. Optimization and approximation in deterministic machine scheduling: a survey, Annals of Discrete Mathematics, vol. 5, 1979, pp. 287 - 326.
3. Hall N.G., Potts C.N., Sriskandarajah C. Parallel machine scheduling with a common server, Discrete Applied Math., 2000 (to appear).
4. Koulamas C.P. Scheduling on two parallel machines for minimizing machine interference, Working Paper, Department of Decision Sciences and Information Systems, Florida International University, 1993.
5. Kravchenko S.A., Werner F. Parallel machine scheduling problems with a single server, Mathl. Comput. Modelling , vol. 26, No. 12, 1997, pp.1 - 11.
6. Kravchenko S.A., Werner F. Scheduling on parallel machines with a single server, Otto-von-Guericke-Universität Magdeburg, FMA, Preprint 30/98, 1998.