Parallel Machine Problems with a Single Server and Release Times

SHI Ling

ABSTRACT Parallel machine problems with a single server and release times are generalizations of classical parallel machine problems. Before processing, each job must be loaded on a machine, which takes a certain release times and a certain setup times. All these setups have to be done by a single server, which can handle at most one job at a time. In this paper, we continue studying the complexity result for parallel machine problem with a single and release times. New complexity results are derived for special cases.

KEYWORDS parallel machine problem; single server; release time; complexity

CLC NUMBER O14

Introduction

Parallel machine problems with a single server and release times are the generalizations of classical parallel machine problems and can be formulated as follows. A set \( J = \{J_1, J_2, \ldots, J_n\} \) of \( n \) jobs, identical parallel machines \( M_1, M_2, \ldots, M_m \), and a single server \( M_s \) are given, where each machine can process at most one job at a time. For an operation \( O_{i,j} \) of performing job \( J_i \) (\( j = 1, 2, \ldots, n \)) on machine \( M_i \), the release times take \( r_i (r_i \geq 0) \) time units; the setup time takes \( s_{i,j} (s_{i,j} \geq 0) \) time units and is followed by the processing phase which lasts \( p_{i,j} (p_{i,j} \geq 0) \) time units. Once the setup of some operation is completed, the machine starts the processing phase of that operation, possibly after some idle time. There is no preemption in performing any phase of any operation. A job can be assigned to at most one processing machine at a time. Moreover, during the setup phase of an operation \( O_{i,j} \), both machine \( M_i \) and the server \( M_s \) can only perform the setup of job \( J_i \) on machine \( M_i \). All setups have to be done by a single server \( M_s \), which can handle at most one job at a time. The goal is to determine a feasible schedule, which minimizes a given objective function.

Using the three-field notation scheme for scheduling problem, we denote the parallel machine problems with a single server and release times by \( P_m, S_1 | r_j | C_{max} \). We also consider the special cases where all operations have constant processing times \( p_0 = p \). We denote this problem by \( P_m, S_1 | p_0 = p, r_j | C_{max} \). If we have only \( m = 2 \) machines, the jobs are only processed on machine \( M_1 \) or \( M_2 \). Therefore, the index \( j \) in the notation \( p_{i,j} \) or \( s_{i,j} \) are dropped. The processing times and setup times are denoted by \( p_i \) and \( s_i \). In this case, the \( P_2, S_1 | p = p, r_j | C_{max} \) problem denotes two parallel machines with a single server and constant processing times.

Hall et al. and Kravchenko & Werner obtained complexity results for parallel machine problems with a single server [1, 2]. This paper discusses the parallel machine problems with a single server and release times.

\[ 1 \ P_2, S_1 | r_j | C_{max} \text{ problem} \]

Theorem 1 The \( P_2, S_1 | r_j | C_{max} \) problem is NP-hard in the strong sense.
Proof The proof is based on a reduction from the problem 3-Partition to the P2,S1 | rj | Cmax problem.

The 3-Partition problem, which is known to be strongly NP-hard \(^{19}\), is defined as follows. 3-Partition: Given a positive integer \(b\) and a set \(X = \{x_1, x_2, \ldots, x_m\}\) of positive integers, which satisfies

\[
\frac{b}{4} < x_i < \frac{b}{2}(i = 1,2,\ldots,3m) \quad \text{and} \quad \sum_{i=1}^{3m} x_i = mb
\]

(1)

decide if there exists a partition of \(X\) into \(m\) disjoint 3-element sets \(\{X_1, X_2, \ldots, X_m\}\), then

\[
\sum_{j \in X_i} x_j = b \quad (j = 1,2,\ldots,m)
\]

(2)

Given any instance of 3-Partition, we define the following instance of the scheduling problem P2,S1 | rj | Cmax with four types of jobs.

1. 3m P-jobs with: \(r_i = 0, s_i = x_i, p_i = x_i\) \((i = 1,2,\ldots,3m)\)
2. \(m-1\) Q-jobs with: \(r_i = 2b + (j-1)4b, s_i = 0, p_i = 2b\) \((j = 1,2,\ldots,m-1)\)
3. \(m-1\) R-jobs with: \(r_i = 3b + (l-1)4b, s_i = b, p_i = b\) \((l = 1,2,\ldots,m-1)\)
4. \(m\) U-jobs with: \(r_i = 0, s_i = 0, p_i = 2b\) \((q = 1,2,\ldots,m)\)

The threshold \(y = 4b(m-1) + 2b\) and the corresponding decision problem is: does there exist a schedule \(S\) with makespan \(C(S)\) not greater than \(y = 4b(m-1) + 2b\)?

Assuming that the answer to 3-Partition is 'yes', let \(\{X_1, X_2, \ldots, X_m\}\) be a partition satisfying Eq. (2), where \(X_i = \{\sigma(3j-2), \sigma(3j-1), \sigma(3j)\}\) \((j = 1,2,\ldots,m)\).

The desired schedule \(S_c\) exists and can be described as follows. No machine has intermediate idle time. Machine \(M_1\) processed the P-jobs and Q-jobs in order of permutation \(\sigma\), i.e. in the sequence \(\sigma = \{P_{\sigma(1)}, P_{\sigma(2)}, P_{\sigma(3)}, Q_{\sigma(1)}, Q_{\sigma(2)}, P_{\sigma(3)}, P_{\sigma(1)}, Q_{\sigma(2)}, \ldots, P_{\sigma(1)}, P_{\sigma(2)}, Q_{\sigma(1)}\}\), while machine \(M_2\) processes the R-jobs and U-jobs in the sequence \(\tau = \{U_1, R_1, \ldots, U_m, R_m, U_m\}\).

It is easy to check that the suggested schedule is feasible, i.e. produces no server interference, and that \(C_{\text{max}}(S_c) = y\) (see Fig. 1).

Conversely, suppose that a desired schedule \(S_c\) exists. Since the total workload on each machine is equal to \(y\), it follows that \(C_{\text{max}}(S_c) = y\) and neither machine has any idle time. Without loss of generality, we may assume that machine \(M_1\) processes the P-jobs and Q-jobs in increasing order of their numbering, machine \(M_2\) processes R-jobs and U-jobs in increasing order of their numbering.

Let \(I_l = [x_l, x_l+b]\) denote the time interval in which machine \(M_2\) performs the set-up of job \(R_l\), \(l = 1,2,\ldots,m-1\). For \(l, 2 \leq l \leq m-1\), introduce the interval \(I_l = [x_l, x_l+b]\). We also introduce the time interval \(I_m = [x_m, x_m+b]\), where \(x_m = 0\) for convenience, and the interval \(I_0 = [x_0, x_0+b]\).

We may conclude that

1. job \(U_1\) is processed at the first position on machine \(M_2\), since \(r_{U_1} = 0\); 0
2. job \(U_1\) is processed at the last position on machine \(M_2\), since all U-jobs are processed in increasing order of their numbering;
3. machine \(M_1\) processed jobs in the interval \([0,y]\) without idle time;
4. machine \(M_2\) processed jobs in the interval \([0,y]\) without idle time;
5. machine \(M_3\) processed jobs in the interval \(I_l\).

Since all nonzero set-up times and all processing times on machine \(M_2\) are equal to \(b\), we conclude that the length of each interval \(I_l\) \((1 \leq l \leq m)\) is a multiple of \(b\). Since machine \(M_1\) has no idle time, it follows that machine \(M_1\) must be busy in every interval \(I_l\). Let \(Y_i\) denote the set of indices of the P-jobs, which are set up in interval \(I_l\) for \(l = 1,2,\ldots,m\). Therefore, set \(Y_i\) is not empty. Since \(r_{U_1} = 0\), the \(Y_i\) jobs are the first jobs that are processed in machine \(M_1\), and the length of interval \(I_l\) is equal to \(\sum_{i \in Y_i} (s_{P_i} + p_{P_i})\)

\[
\sum_{i \in Y_i} x_i + |Y_i|b
\]

(see Fig. 1).
1), the length of interval $I_t$ is equal to $\sum_{i \in Y_t} (s_{pi} + p_{pi}) = \sum_{i \in Y_t} x_i + |Y_t| \cdot b - 2b$. Before the interval $I_t$, machine $M_x$ processes at least one job, therefore set $Y_t$ is not empty, and the length of interval $I_t$ is equal to $\sum_{i \in Y_t} (s_{pi} + p_{pi}) = \sum_{i \in Y_t} x_i + |Y_t| \cdot b$.

For all these values being multiples of $b$, the inequalities $|Y_t| \geq 3$ must hold for all $t = 1, 2, \ldots, m$, since $\frac{b}{2} < x_i < \frac{b}{2}$ $(i = 1, 2, \ldots, 3m)$. There are exactly $m$ intervals $I_t$, and in these intervals exactly $3m$ jobs have to be set up. Therefore, we must have that $|Y_t| = 3$ and $\sum_{i \in Y_t} x_i = b$ for all $j = 1, 2, \ldots, m$. Thus, the sets $X_j = Y_j$ give a solution to the 3-Partition.

**2 \ P2, S1 | \ p_i = p, r_j | C_{\max} \ \text{problem}**

### 2.1 Complexity of \( P2, S1 | \ p_i = p, r_j | C_{\max} \ \text{problem} \)

**Theorem 2** The \( P2, S1 | p_i = p, r_j | C_{\max} \) problem is NP-hard in the strong sense, too.

**Proof** Given any instance of 3-Partition, we define the following instance of the scheduling problem \( P2, S1 | p_i = p, r_j | C_{\max} \) with four types of jobs.

1. \( 3m \) \( P \)-jobs with: $r_i = 4b + (j - 1)5b, s_i = 0, p_i = b, (j = 1, 2, \ldots, 3m)$
2. \( m \) \( Q \)-jobs with: $r_i = 4b + (j - 1)5b, s_i = 0, p_i = b, (j = 1, 2, \ldots, m - 1)$
3. \( m \) \( R \)-jobs with: $r_i = 3b + (l - 1)5b, s_i = b, p_i = b, (l = 1, 2, \ldots, m - 1)$
4. \( m \) \( U \)-jobs with: $r_q = 0, s_q = 0, p_q = b, (q = 1, 2, \ldots, m)$

The threshold $y = 5mb$ and the corresponding decision problem is: does there exist a schedule $S$ with makespan $C(S)$ not greater than $y = 5mb$?

Assuming that the answer to 3-Partition is 'yes', let $\{X_1, X_2, \ldots, X_m\}$ be a partition satisfying Eq. (2), where $X_j = (\sigma(3j - 2), \sigma(3j - 1), \sigma(3j))$ $(j = 1, 2, \ldots, m)$.

The desired schedule $S_0$ exists and can be described as follows. No machine has intermediate idle time. Machine $M_1$ processed the $P$-jobs and $Q$-jobs in order of permutation $\sigma$, i.e. in the sequence $\sigma = \{P_{\sigma(1)}, P_{\sigma(2)}, P_{\sigma(3)}, Q_1, P_{\sigma(4)}, P_{\sigma(5)}, P_{\sigma(6)}, Q_1, \ldots, P_{\sigma(3m-2)}, P_{\sigma(3m-1)}, P_{\sigma(3m)}, Q_{m}\}$

while machine $M_2$ processes the $R$-jobs and $U$-jobs in the sequence $r = \{U_{r(1)}, U_{r(2)}, U_{r(3)}, R_1, U_{r(4)}, U_{r(5)}, U_{r(6)}, R_1, \ldots, R_{r(3m-2)}, U_{r(3m-1)}, U_{r(3m)}, R_{m}\}$.

It is easy to check that the suggested schedule is feasible, i.e. produces no server interference, and that $C_{\max}(S_0) = y$ (see Fig. 2).

![Fig. 2 Gantt charts for \( P2, S1 | p_i = p, r_j | C_{\max} \) problem](image)

The following proof is the same as theorem 1, so we omitted it.

### 2.2 An improved heuristic algorithm for \( P2, S1 | p_i = p, r_j | C_{\max} \) problem

The NP-hardness of our parallel problem with a single server and release time motivates the search for polynomial-time approximation algorithms. In this section we show an improved algorithm applied to the \( P2, S1 | p_i = p, r_j | C_{\max} \) problem, which creates a schedule with a makespan that is at most twice as large as the optimal value. We will introduce an example to prove that the bound of 2 is tight.

Let $S^*$ denote an optimal schedule for the \( P2, S1 | p_i = p, r_j | C_{\max} \) problem, then we have

$$C_{\max}(S^*) \geq \sum_{i=1}^{n} \sum_{j \in N_i} s_{ij}$$ (3)

since the server can do only one setup at a time. Also, for any machine $M_i (i = 1, 2, \ldots, m)$, we have
Algorithm 1:
Step 1: Divide \( n \) jobs into \( q (q \leq n) \) groups according to its release times:

\[
N = \{1, 2, \ldots, n\} = \{N_1, N_2, \ldots, N_q\},
\]

That is in the same groups \( N_k (k = 1, 2, \ldots, q) \), the release times is the same.

Step 2: For any subset \( N_i \in N \), apply LPT (largest processing times) method to arrange the jobs, in which the processing times are equal to \( \{s_{i,j} + p_{i,j}\} \), and omit the subset \( N_i \) from \( N \), until \( N = Q \).

Theorem 3: For the \( P_{m_s} S_1 | p_{r_i} = p \cdot r_i | C_{\text{max}} \) problem, let \( S_0 \) be a schedule created by Algorithm 1. Then

\[
\frac{C_{\text{max}}(S_0)}{C_{\text{max}}(S^*)} \leq 2
\]

and this bound is tight.

Proof: Considering schedule \( S_0 \), and the suppose that machine \( M_i \) terminates this schedule. Let \( I_q(S_0) \) denote the total idle times on machine \( M_i \). Then

\[
C_{\text{max}}(S_0) = \sum_{j \in N_q}(r_j + s_{i,j} + p_{i,j}) + I_q(S_0)
\]

and \( I_q(S_0) \leq \sum_{j \in N_q} s_{i,j} - \sum_{j \in N_q} S_{q,j} \). Since whenever machine \( M_i \) is idle, the server is busy doing the setup of a job on some other machine. Thus, due to Eq. (3) and Eq. (4), we have

\[
C_{\text{max}}(S_0) \leq \sum_{j \in N_q}(r_j + p_{i,j}) + \sum_{j \in N_q} s_{i,j} \leq 2C_{\text{max}}(S^*)
\]

To see that this bound is tight, we consider the following instance of the \( P_{m_s} S_1 | p_{r_i} = p \cdot r_i | C_{\text{max}} \) problem. There are \( n = m^2 \) jobs with the release times, setup times and processing times defined as follows.

\[
s_{i,j} = 0, p_{i,j} = 1 (j \neq m) \quad r_i = m - 1, s_{i,1} = 1,
\]

\[
p_{i,m} = 1;
\]

\[
s_{i,j} = 0, p_{i,j} = 1 (j \neq 2m - 1) \quad r_{2m-1} = m - 2, s_{2m-1} = 1, p_{2m-1} = 1;
\]

\[
s_{i,j} = 0, p_{i,j} = 1 (i \neq m - (i - 1)) \quad r_{m - (i - 1)} = m - (i - 1), s_{m - (i - 1)} = 1, p_{m - (i - 1)} = 1;
\]

\[
s_{m,j} = 0, p_{m,j} = 1 (i \neq (m - 1) m + 1);
\]

A schedule that attains this lower bound can be found by allocating the jobs in the server in the order 1, 2, \ldots, \( m^2 \). None of the machine has intermediate idle time, the server is busy between the first and last setups, and \( C_{\text{max}}(S^*) = m + 1 \) (see Fig. 3(a)).

On the other hand, Algorithm 1 creates a schedule \( S \) in which the jobs are allocated to the server in the order \( m, 2m - 1, \ldots, m^2 \), and \( C_{\text{max}}(S) = 2m \) (see Fig. 3(b)).

Thus, \( \frac{C_{\text{max}}(S)}{C_{\text{max}}(S^*)} = \frac{2m}{m + 1} \to 2 \) as \( m \to \infty \). Thus the proof of Theorem 3 is completed.

Fig. 3 Tightness if algorithm 1

REFERENCES
1. Hall N, Potts C, Sriskandarajah C (2000) Parallel machine scheduling with a common server. *Discrete Applied Math.* 102: 223-243.
2. Kravchenko S K, Werner F (1997) Parallel machine scheduling problems with a single server. *Mathematical and Computer Modelling.* 26(12): 1-11.
3. Garey M R, Johnson D S (1977) *Computers and Intractability: a guide to the theory of NP-completeness.* San Francisco: Freeman.