APPROXIMATE ALGORITHMS FOR UNRELATED MACHINE SCHEDULING TO MINIMIZE MAKESPAN

XIANZHAO ZHANG
School of Science, Linyi University, Linyi 276005, China

DACHUAN XU*
Department of Information and Operations Research
Beijing University of Technology, Beijing 100124, China

DONGLEI DU
Faculty of Business Administration
University of New Brunswick, Fredericton, NB, E3B 5A3, Canada

CUIXIA MIAO
School of Mathematical Sciences
Qufu Normal University, Qufu 273165, China

Abstract. We study two unrelated machine scheduling problems with machine dependent release dates to minimize the makespan. For the case with fixed processing time where processing job \( j \) on machine \( i \) requires time \( p_{ij} \) and incurs a cost of \( c_{ij} \), we derive a 2-approximation algorithm. For the problem with variable processing times where the cost increases linearly as the processing time decreases, we propose a \((2 + \epsilon)\)-approximation algorithm.

1. Introduction. The unrelated parallel machine scheduling problem to minimize the makespan, \( R||C_{\text{max}} \) in the three-field notation (see [5]), is one of the classic NP-hard problems in combinatorial optimization. This problem is mentioned in many works concerning approximation algorithms ([12, 13]), and it has received much attention in the past few decades with many extant approximation algorithms, among which the currently best 2-approximation algorithm is due to [7], who also show that the problem cannot be approximated within a factor smaller than 3/2, unless \( P = NP \). Recently, a slightly better algorithm that achieves \((2 - 1/m)\)-approximation is obtained by [9] \((m \) is the number of unrelated machines). A special case of this problem where jobs are restricted to a subset of machines, i.e., \( p_{ij} \in \{p_j, +\infty\} \), has attracted more and more concern in recent years. Ebenlendr et al. [3] propose a 1.75-approximation algorithm when each job is restricted to at most two machines. A polynomial time algorithm is provided that estimates the optimal makespan up to a factor of \((\frac{3}{2} + \epsilon)\) for the restricted assignment problem in [11]. A relevant problem with additional cost \( c_{ij} \) if job \( j \) is processed on machine

2010 Mathematics Subject Classification. Primary: 90C27, 90B35; Secondary: 68W25, 68W40.

Key words and phrases. Scheduling, unrelated parallel machine, approximation algorithm, linear programming, rounding.

The reviewing process of the paper was handled by Yanqin Bai as Guest Editor.
This paper is dedicated to the memory of Professor Xiaoling Sun (1963-2014).
*Corresponding author. Email: xude@bjut.edu.cn.
i is considered by [10]. Shmoys and Tardos [10] prove that, if a schedule with total cost \( C \) and makespan \( T \) exists, a schedule with total cost \( C \) and makespan \( 2T \) can be found in polynomial time. Their proof relies on rounding the solution of an LP relaxation. They obtain the same result even for a more general version of the problem, i.e., the variable model, in which the processing time \( p_{ij} \) of any job-machine pair \((i, j)\) is a linear decreasing function of the associated cost \( c_{ij} \). Another variant of the unrelated parallel machine scheduling with resource dependent processing times is considered in [6]. They also use LP rounding technique to obtain a 4-approximation algorithm and an improved 3.75-approximation algorithm for the problem. Given an energy budget, a \((2+\epsilon)\)-approximation algorithm is given for the makespan minimization problem under two different settings (see [1]). We refer to [2, 4, 8] and references therein for other variants of unrelated parallel machine scheduling problems.

The unrelated machine scheduling problem we are interested in this work is stated as follows: We are given a job set \( \{J_1, \ldots, J_n\} \) and a machine set \( \{M_1, \ldots, M_m\} \) respectively. Each job is to be processed without interruption by exactly one machine. Job \( J_j \) has a release time \( r_{ij} \) (a nonnegative integer) on machine \( M_i \) \((i = 1, \ldots, m, j = 1, \ldots, n)\). We are also provided with a constant \( C \), which is an upper bound on the total cost for processing jobs \( \{J_1, \ldots, J_n\} \). The objective is to find a schedule that minimizes the makespan with total cost no more than \( C \).

In this paper we consider two versions of the generalized assignment problem, namely, the fixed model and the variable model. These two models may arise from the following actual conditions: Due to their difference in skills, the workers spend different times and get different pays when executing the same task. Here the workers correspond to the machines in our problem, and their pays are a counterpart to the processing costs. The decision maker always wants to finish all the tasks as soon as possible subject to the condition that the total cost he pays to the workers does not exceed a certain budget. In the fixed model, the processing time and the pay are fixed; and in the variable model, the pay is a linearly decreasing function of the processing time—and hence slower works are paid less.

The linear program formulations we consider are almost the same as those proposed by [10]. The authors in [10] mainly introduce a new rounding technique and show how to obtain a required schedule from a feasible solution of a certain linear program. Our main contribution lies in the following: We declare that the linear programming model possesses some properties (Theorem 3.4), which serve as the foundation for a subsequent binary search procedure to carry through.

The remainder of this paper is organized in the following way: We present the 2-approximation and the \((2+\epsilon)\)-approximation algorithms for the fixed and variable models in sections 2 and 3, respectively. We then offer some concluding remarks in Section 4.

2. A 2-approximation algorithm for the fixed model. For this model, Job \( J_j \) has processing time \( p_{ij} \) (a positive integer) when processed on machine \( M_i \) and thus incurs a cost \( c_{ij} \). Remember that job \( J_j \) has release time \( r_{ij} \) (a nonnegative integer) on machine \( M_i \). Our goal is to find a schedule that minimizes the makespan with the total cost no more than \( C \).

We denote by \( \pi^* \) the optimal schedule to this model, and by \( T^* \) the makespan for \( \pi^* \). Note that \( T^* \) is a positive integer from the definitions for \( r_{ij} \) and \( p_{ij} \). Consider
the following integer linear program \( \text{IP}(t) \) along with its relaxation \( \text{LP}(t) \):

\[
\begin{align*}
\text{IP}(t) : & \quad \sum_{i=1}^{m} x_{ij} = 1, \quad j = 1, 2, \ldots, n, \\
& \sum_{j=1}^{n} p_{ij} x_{ij} \leq t, \quad i = 1, 2, \ldots, m, \\
& \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \leq C, \\
& (r_{ij} + p_{ij}) x_{ij} \leq t, \quad i = 1, 2, \ldots, m, \\
& \quad j = 1, 2, \ldots, n, \\
x_{ij} \in \{0, 1\}, \quad i = 1, 2, \ldots, m, j = 1, 2, \ldots, n.
\end{align*}
\]

\[
\begin{align*}
\text{LP}(t) : & \quad \sum_{i=1}^{m} x_{ij} = 1, \quad j = 1, 2, \ldots, n, \\
& \sum_{j=1}^{n} p_{ij} x_{ij} \leq t, \quad i = 1, 2, \ldots, m, \\
& \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \leq C, \\
& x_{ij} = 0, \quad \text{if} \quad r_{ij} + p_{ij} > t, \\
& \quad i = 1, 2, \ldots, m, j = 1, 2, \ldots, n, \\
x_{ij} \geq 0, \quad i = 1, 2, \ldots, m, j = 1, 2, \ldots, n.
\end{align*}
\]

Note that \( \text{LP}(T^*) \) admits a feasible solution since \( \text{IP}(T^*) \) has a feasible solution. We denote by \( M_{i,j} \) the machine with the minimum cost for job \( J_j \) \((j = 1, 2, \ldots, n)\), i.e., \( i_j = \arg \min_{i} \{c_{ij}\} \). Meanwhile we have the following notations: \( r_{\max} = \max_{ij} r_{ij}, p_j = \max_i p_{ij}, p_{\text{sum}} = \sum_{j=1}^{n} p_j \).

Job \( J_j \) is assigned to machine \( M_{i,j} \), and the jobs on each machine are scheduled in an increasing order of their release times. We denote by \( U_0 \) the makespan of the schedule obtained in such a way. Obviously we have \( U_0 \leq r_{\max} + p_{\text{sum}} \). Assume that \( L_0 = 0 \).

The minimum positive integer with which \( \text{LP}(t) \) admits a feasible solution is denoted by \( \bar{T} \). Clearly we have \( \bar{T} \leq T^* \), where \( \bar{T} \) can be found by conducting a binary search over the interval \([L_0, U_0]\) after at most \( \log(r_{\max} + p_{\text{sum}}) \) steps.

Assume that \( x_{\text{LP}} \) is a feasible solution for \( \text{LP}(\bar{T}) \). Then \( x_{\text{LP}} \) can be converted into a feasible solution \( x_{\text{IP}} \) for the following integer program using the technique in [10].

\[
\begin{align*}
\sum_{i=1}^{m} x_{ij} = 1, \quad j = 1, 2, \ldots, n, \\
\sum_{j=1}^{n} p_{ij} x_{ij} \leq \bar{T} + p_{\max}, \quad i = 1, 2, \ldots, m,
\end{align*}
\]
\[
\sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \leq C,
\]
\[
x_{ij} \in \{0, 1\}, \quad i = 1, 2, \ldots, m; j = 1, 2, \ldots, n.
\]

Here \( p_{\text{max}} = \max\{p_{ij} | x_{ij}^l > 0\} \).

The rounding procedure of [10] ensures that in \( x^{\text{IP}} \), except for at most one job on each machine (called an extra job if exists), the total processing time of all the other jobs is no more than \( T^* \) (for details see the proof of Theorem 2.1 in [10]).

This property allows us to respect the release time of each job by ordering the jobs on each machine such that all the jobs start at \( T^* \) unless there exist some extra jobs on some machines, in which case the extra job of each machine is processed first before any other job. Hence we can construct a feasible schedule with makespan no more than \( 2T^* \).

In conclusion, we obtain a 2-approximation algorithm as follows:

**Algorithm 2.1.**

- **Step 1:** Find the minimum positive integer \( \bar{T} \) with which \( LP(t) \) admits a feasible solution by binary search.
- **Step 2:** Solve \( LP(\bar{T}) \) and obtain a feasible solution \( x^{\text{LP}} \).
- **Step 3:** Transform \( x^{\text{LP}} \) into a solution \( x^{\text{IP}} \) for the integer program (3) by the rounding technique of [10].
- **Step 4:** Construct a schedule from \( x^{\text{IP}} \), while respecting the release time of each job.

3. A \((2 + \epsilon)\)-approximation algorithm for the variable model. In this model, job \( J_j \) may be processed by machine \( M_i \) in \( t_{ij} \) time units, where \( l_{ij} \leq t_{ij} \leq u_{ij} \); and the cost linearly increases as the processing time of each job decreases. Accordingly, we are also given the minimum cost \( c_{ij}^l \) and the maximum cost \( c_{ij}^u \) of assigning job \( J_j \) to machine \( M_i \). The cost associated with processing \( J_j \) on machine \( M_i \) in time \( t_{ij} \) is \( \lambda c_{ij}^l + (1 - \lambda)c_{ij}^u \) when \( t_{ij} \) is written as \( t_{ij} = \lambda l_{ij} + (1 - \lambda)u_{ij} \) for \( 1 \geq \lambda \geq 0 \). Job \( J_j \) has release time \( r_{ij} \geq 0 \) on machine \( M_i \). We shall find a schedule that minimizes the makespan with the total cost no more than \( C \).

We still denote by \( \pi^* \) the optimal schedule to this model, and by \( T^* \) the makespan for \( \pi^* \). A key observation is that we can always reduce \( u_{ij} \) to \( T^* - r_{ij} \) for those \( J_j \) satisfying \( r_{ij} + u_{ij} > T^* \) (\( J_j \) is processed on machine \( M_i \) in \( \pi^* \)). This reduction maintains both the structure and the makespan of the optimal schedule. The above observation motivates the following definition.

**Definition 3.1.** For any given parameter \( t > 0 \) and machine-job pair \((i, j)\), we define adjusted upper bound \( u_{ij}(t) \) and the corresponding cost \( c_{ij}^u(t) \) in the following way:

\[
u_{ij}(t) := \begin{cases} 
  u_{ij}, & \text{if } r_{ij} + l_{ij} > t \text{ or } r_{ij} + u_{ij} \leq t; \\
  t - r_{ij}, & \text{if } r_{ij} + l_{ij} < t < r_{ij} + u_{ij},
\end{cases}
\]

\[
c_{ij}^u(t) := c_{ij}^u + \frac{c_{ij}^l - c_{ij}^u}{l_{ij} - u_{ij}}(u_{ij}(t) - u_{ij}).
\]

We now relax the variable model as the following linear program.

\[
LP_1(t) : \sum_{i=1}^{m} (x_{ij}^l + x_{ij}^u) = 1, \quad j = 1, 2, \ldots, n,
\]
Lemma 3.3. Given
We consider the following three cases according to the adjusted upper bounds
Proof. From the definition of \( u_{ij}(t) \), the event \( r_{ij} + u_{ij}(t_2) > t_2 \) occurs only when
Hence, we get \( r_{ij} + l_{ij} > t_1 \), implying that \( r_{ij} + u_{ij}(t_1) = r_{ij} + u_{ij} \geq r_{ij} + l_{ij} > t_1 \).
Lemma 3.3. Given \( 0 < t_1 < t_2 \), we have \( u_{ij}(t_1) \leq u_{ij}(t_2) \) for any \( (i, j) \) satisfying
Proof. We consider the following three cases according to the adjusted upper bounds \( u_{ij}(t_1) \) and \( u_{ij}(t_2) \).
• Case 1. For \( r_{ij} + u_{ij} \leq t_1 \), we get \( u_{ij}(t_1) = u_{ij}(t_2) = u_{ij} \).
• Case 2. For \( r_{ij} + l_{ij} \leq t_1 < r_{ij} + u_{ij} \), we have \( u_{ij}(t_1) = t_1 - r_{ij} \).

\[
\sum_{j=1}^{n} (l_{ij} x_{ij}^l + u_{ij}(t) x_{ij}^u) \leq t, \quad i = 1, 2, \ldots, m,
\]
\[
\sum_{j=1}^{m} \sum_{i=1}^{n} (c_{ij} x_{ij}^l + c_{ij}^u(t) x_{ij}^u) \leq C,
\]
\[
(r_{ij} + u_{ij}(t)) (x_{ij}^l + x_{ij}^u) \leq t, \quad i = 1, 2, \ldots, m,
\]
\[
(x_{ij}^l + x_{ij}^u) \in \{0, 1\}, \quad i = 1, \ldots, m, j = 1, \ldots, n,
\]
\[
x_{ij}^l, x_{ij}^u \geq 0, \quad i = 1, 2, \ldots, m, j = 1, 2, \ldots, n.
\]
In the above, there are two variables \( x_{ij}^l \) and \( x_{ij}^u \) for each machine-job pair \((i, j)\), where \( i = 1, \ldots, m \) and \( j = 1, \ldots, n \). A feasible schedule with makespan \( t \) directly corresponds to a feasible solution to the linear program. Job \( J_j \) is assigned to machine \( M_i \) if \( x_{ij}^l + x_{ij}^u = 1 \), and the assigned time is \( l_{ij} x_{ij}^l + u_{ij}(t) x_{ij}^u \) with the associated cost \( c_{ij} x_{ij}^l + c_{ij}^u(t) x_{ij}^u \). Since \( T^* \) is the makespan for the optimal schedule, \( \text{LP}_1(T^*) \) admits a feasible solution from the above discussion. We now relax \( \text{LP}_1(t) \) to the following linear program.

\[
\text{LP}_2(t): \quad \sum_{j=1}^{n} (x_{ij}^l + x_{ij}^u) = 1, \quad j = 1, 2, \ldots, n,
\]
\[
\sum_{j=1}^{n} (l_{ij} x_{ij}^l + u_{ij}(t) x_{ij}^u) \leq t, \quad i = 1, 2, \ldots, m,
\]
\[
\sum_{j=1}^{m} \sum_{i=1}^{n} (c_{ij} x_{ij}^l + c_{ij}^u(t) x_{ij}^u) \leq C,
\]
\[
x_{ij}^l = x_{ij}^u = 0, \quad \text{if} \quad (r_{ij} + u_{ij}(t)) > t, \quad i = 1, 2, \ldots, m, j = 1, 2, \ldots, n,
\]
\[
x_{ij}^l, x_{ij}^u \geq 0, \quad i = 1, 2, \ldots, m, j = 1, 2, \ldots, n.
\]
\( \text{LP}_2(T^*) \) has a feasible solution since \( \text{LP}_2(t) \) is a relaxation for \( \text{LP}_1(t) \). We claim that the linear program \( \text{LP}_2(t) \) possesses some amicable property for later analysis.

We first give two lemmas as preliminary.

**Lemma 3.2.** Given \( 0 < t_1 < t_2 \), we have \( r_{ij} + u_{ij}(t_1) > t_1 \) for any \( (i, j) \) satisfying \( r_{ij} + u_{ij}(t_2) > t_2 \).

**Proof.** From the definition of \( u_{ij}(t) \), the event \( r_{ij} + u_{ij}(t_2) > t_2 \) occurs only when \( r_{ij} + l_{ij} > t_2 \). Hence, we get \( r_{ij} + l_{ij} > t_1 \), implying that \( r_{ij} + u_{ij}(t_1) = r_{ij} + u_{ij} \geq r_{ij} + l_{ij} > t_1 \).

**Lemma 3.3.** Given \( 0 < t_1 < t_2 \), we have \( u_{ij}(t_1) \leq u_{ij}(t_2) \) for any \( (i, j) \) satisfying \( r_{ij} + l_{ij} \leq t_1 \).

**Proof.** We consider the following three cases according to the adjusted upper bounds \( u_{ij}(t_1) \) and \( u_{ij}(t_2) \).

- **Case 1.** For \( r_{ij} + u_{ij} \leq t_1 \), we get \( u_{ij}(t_1) = u_{ij}(t_2) = u_{ij} \).
- **Case 2.** For \( r_{ij} + l_{ij} \leq t_1 < r_{ij} + u_{ij} \), we have \( u_{ij}(t_1) = t_1 - r_{ij} \).
Given foundation for the subsequent binary search procedure.

Proof. Suppose $(i,j)$ satisfying $r_{ij} + u_{ij}(t_1) > t_1$, $x_{ij}^{l(1)} = x_{ij}^{u(1)} = 0$. Therefore, for the rest of the argument, we consider only those machine-job pairs $(i,j)$ with $r_{ij} + u_{ij}(t_1) \leq t_1$, i.e., $r_{ij} + l_{ij} \leq t_1$.

Lemma 3.3 implies that $u_{ij}(t_1) \leq u_{ij}(t_2)$ for any machine-job pair $(i,j)$ satisfying $r_{ij} + l_{ij} \leq t_1$. The relationship between the cost and the processing time implies that $c_{ij}^u(t_2) \leq c_{ij}^u(t_1)$. $c_{ij}^u(t_2) (c_{ij}^u(t_1))$ is the cost associated with processing $J_j$ on machine $M_i$ in time $u_{ij}(t_2)$ ($u_{ij}(t_1)$).

If \( \{x_{ij}^{l(1)}, x_{ij}^{u(1)}\} \) is not a feasible solution for LP\(_2(t_2)\), there must be at least one machine (say $M_i$) satisfying
\[
\sum_{j=1}^{n} (l_{ij} x_{ij}^{l(1)} + u_{ij}(t_2) x_{ij}^{u(1)}) > t_2.
\]

We can transform \( \{x_{ij}^{l(1)}, x_{ij}^{u(1)}\} \) into a feasible solution for LP\(_2(t_2)\) in the following way.

- For those machines $M_i$ satisfying $\sum_{j=1}^{n} \left( l_{ij} x_{ij}^{l(1)} + u_{ij}(t_2) x_{ij}^{u(1)} \right) \leq t_2$, let $x_{ij}^{l(2)} = x_{ij}^{l(1)}$, $x_{ij}^{u(2)} = x_{ij}^{u(1)}$; $j = 1, \ldots, n$.
- For those machine $M_i$ satisfying $\sum_{j=1}^{n} \left( l_{ij} x_{ij}^{l(1)} + u_{ij}(t_2) x_{ij}^{u(1)} \right) > t_2$, there must be some jobs $J_j$ such that $u_{ij}(t_2) > u_{ij}(t_1)$.
- For those jobs $J_j$ with $u_{ij}(t_2) = u_{ij}(t_1)$, let $x_{ij}^{l(2)} = x_{ij}^{l(1)}$, $x_{ij}^{u(2)} = x_{ij}^{u(1)}$; $j = 1, \ldots, n$.
- For the remaining jobs (i.e., jobs $J_j$ with $u_{ij}(t_2) > u_{ij}(t_1)$), let
\[
\lambda_{ij} = \frac{u_{ij}(t_2) - u_{ij}(t_1)}{u_{ij}(t_2) - l_{ij}} x_{ij}^{u(1)}
\]
and $x_{ij}^{l(2)} = x_{ij}^{l(1)} + \lambda_{ij}$, $x_{ij}^{u(2)} = x_{ij}^{u(1)} - \lambda_{ij}$.

Note that for machine-job pair $(i,j)$ with $u_{ij}(t_2) > u_{ij}(t_1)$, we have $l_{ij} \leq u_{ij}(t_1) < u_{ij}(t_2)$ (see Lemma 3.3).

Obviously the following equalities hold for each machine-job pair $(i,j)$ with $u_{ij}(t_2) > u_{ij}(t_1)$:
\[
\begin{align*}
x_{ij}^{l(2)} + x_{ij}^{u(2)} &= x_{ij}^{l(1)} + x_{ij}^{u(1)}, \\
l_{ij} x_{ij}^{l(2)} + u_{ij}(t_2) x_{ij}^{u(2)} &= l_{ij} x_{ij}^{l(1)} + u_{ij}(t_1) x_{ij}^{u(1)}, \\
x_{ij}^{u(2)} &\geq 0.
\end{align*}
\]

Furthermore, we have
\[ c^l_{ij}x^l_{ij} + c^u_{ij}(t_2)x^u_{ij} = c^l_{ij}x^l_{ij} + c^u_{ij}(t_2)x^u_{ij} + \lambda_{ij}(c^l_{ij} - c^u_{ij}(t_2)) = \frac{c^l_{ij}x^l_{ij}}{\lambda_{ij}} + \frac{c^u_{ij}(t_2)x^u_{ij}}{\lambda_{ij}} + \frac{\lambda_{ij}(c^l_{ij} - c^u_{ij}(t_2))}{\lambda_{ij}} = \frac{c^l_{ij}x^l_{ij}}{\lambda_{ij}} + \frac{c^u_{ij}(t_1)x^u_{ij}}{\lambda_{ij}}.\]

For those machine-job pair \((i, j)\) with \(r_{ij} + l_{ij} > t_1\), let \(x^l_{ij} = x^u_{ij} = 0\).

We declare that \(\{x^l_{ij}, x^u_{ij}\} (i = 1, \ldots, m; j = 1, \ldots, n)\) is a feasible solution for \(LP_2(t_2)\).

As before, we denote by \(M_{ij}\) the machine with the minimum cost for job \(J_j\), i.e., \(i_j = \arg\min \{c_{ij}\}\). Meanwhile we have the following notations: \(r_{ij} = \max_{i_j} r_{ij}\), \(p_j = \max u_{ij}, p\text{sum} = \sum_{j=1}^{n} p_j\). Job \(J_j\) is processed on machine \(M_{ij}\) with processing time \(u_{ij} (j = 1, 2, \ldots, n)\); The jobs on each machine are scheduled in an increasing order of their release times. We denote by \(U_0\) the makespan of the schedule obtained in such a way. Obviously we have \(U_0 \leq r_{\text{max}} + p\text{sum}\). Let \(L_0 = \frac{1}{2} \min_{i_j} \{l_{ij}\}\). Note that \(L_0 > 0\), and the feasible set of \(LP_2(L_0)\) is \(\phi\).

We execute the following binary search procedure.

**Algorithm 3.5.**

**Binary Search:**

**Input:** \(L_0, U_0, \epsilon\).

**Output:** \(U_k\).

**Step 0:** \(k := 0\)

**Step 1:** while \(U_k > (1 + \epsilon)L_k\) do

1. \(t \leftarrow (L_k + U_k) / 2\) and \(k \leftarrow k + 1\)
2. If \(LP_2(t)\) has a feasible solution then \(U_k \leftarrow t\) and \(L_k \leftarrow L_{k-1}\) else \(L_k \leftarrow t\), and \(U_k \leftarrow U_{k-1}\)

End-while

The binary search procedure terminates when \(U_k \leq (1 + \epsilon)L_k\), i.e., \(U_k - L_k \leq \epsilon L_k\), which is equivalent to \(\frac{U_k - L_k}{L_k} \leq \epsilon\). The procedure terminates after at most \(\lceil \log(\frac{L_k}{L_0}) \rceil \leq \lceil \log(\frac{L_k}{L_{max}} + \text{sum}) \rceil\) recursions. We now have \(U_k \leq (1 + \epsilon)L_k < (1 + \epsilon)T^*\), implying that \(LP_2(L_0)\) is not feasible. Theorem 3.4 therefore implies that \(L_k < T^*\) since \(LP_2(T^*)\) admits a feasible solution.

Suppose \(\{x^l_{ij}, x^u_{ij}\} (i = 1, \ldots, m; j = 1, \ldots, n)\) is a feasible solution for \(LP_2(U_k)\).

We consider the following linear program along with the construction of a feasible solution \(x_{ij}\).

\[
LP(U_k): \sum_{i=1}^{m} x_{ij} = 1, \quad j = 1, 2, \ldots, n, \\
\sum_{j=1}^{n} p_{ij}x_{ij} \leq U_k, \quad i = 1, 2, \ldots, m, \\
\sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}x_{ij} \leq C, \\
x_{ij} = 0, \quad \text{if} \quad r_{ij} + p_{ij} > U_k,
\]

\(\sum_{i=1}^{m} x_{ij} = 1, \quad j = 1, 2, \ldots, n, \) implies that \(x_{ij} \in \{0, 1\}\) for all \(i, j\).
Let \( x_{ij} = x_{ij}^l + x_{ij}^u \) (\( i = 1, \ldots, m; j = 1, \ldots, n \)). For those \((i, j)\) such that \( x_{ij} > 0 \), define its processing time and cost respectively as

\[
p_{ij} = \frac{l_{ij} x_{ij}^l + u_{ij}(U_k) x_{ij}^u}{x_{ij}},
\]

\[
c_{ij} = \frac{c_{ij}^l x_{ij}^l + c_{ij}^u(t) x_{ij}^u}{x_{ij}}.
\]

For those \((i, j)\) with \( x_{ij} = 0 \), define \( p_{ij} = A \) (very large number), and \( c_{ij} = \delta \) (very small constant).

Note that \( x_{ij} > 0 \) implies that \( x_{ij}^l + x_{ij}^u > 0 \), and hence \( r_{ij} + u_{ij}(U_k) \leq U_k \).

Obviously we have \( r_{ij} + p_{ij} \leq r_{ij} + u_{ij}(U_k) \leq U_k \).

Therefore \( \{x_{ij}\} \) (\( i = 1, \ldots, m; j = 1, \ldots, n \)) is a feasible solution for the above program with the afore-defined \( p \) and \( c \).

Using again the rounding technique of [10], we can construct a feasible schedule for the variable model with makespan at most \( 2U_k < 2(1 + \epsilon)T^* = (2 + 2\epsilon)T^* \).

In conclusion, the following is a \((2 + \epsilon)\)-approximation algorithm for the variable model:

**Algorithm 3.6.**

**Step 1:** Find a constant \( U_k \) satisfying \( U_k \leq (1 + \epsilon)L_k \) by binary search.

**Step 2:** Solve LP\(_2(U_k)\) and obtain a feasible solution \( \{x_{ij}^l, x_{ij}^u\} \).

**Step 3:** Obtain a feasible solution \( \{x_{ij}\} \) for LP(\( U_k \)) from \( \{x_{ij}^l, x_{ij}^u\} \).

**Step 4:** Transform \( \{x_{ij}\} \) into a solution \( x^{IP} \) for some integer program by rounding technique of [10].

**Step 5:** Construct a schedule from \( x^{IP} \), while respecting the release time of each job.

4. **Concluding remarks.** In this paper, we study two generalized assignment models with machine dependent release dates to minimize the makespan. We derive a 2-approximation algorithm for the fixed model and a \((2 + \epsilon)\)-approximation for the variable model. An interesting future direction of research is to consider the nonlinear cost case, where, for example, the cost is a convex (concave) function of the processing time.

**Acknowledgements.** The research of the second author is supported by NSF of China (No. 11371001). The third author’s research is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) grant 283106. The research of the fourth author is supported by NSF of China (No. 11201259), and the Doctoral Fund of the Ministry of Education (No. 20123705120001).

**REFERENCES**

[1] L. Chen, W. C. Luo, and G. C. Zhang, *Approximation algorithms for unrelated machine scheduling with an energy budget*, in *Proceedings of Frontiers in Algorithmics and Algorithmic Aspects in Information and Management (FAW-AAIM)*, (2011), 244–254.

[2] J. R. Correa, A. Marchetti-Spaccamela, J. Matuschke, L. Stougie, O. Svensson, V. Verdugo, and J. Verschae, *Strong LP formulations for scheduling splittable jobs on unrelated machines*, in *Proceedings of the 17th Integer Programming and Combinatorial Optimization (IPCO)*, (2014), 249–260.
[3] T. Ebenlendr, M. Krčal, and J. Sgall, Graph balancing: a special case of scheduling unrelated parallel machines, *Algorithmica*, 68 (2014), 62–80.

[4] N. K. Freeman, J. Mittenthal and S. H. Melouk, Parallel-machine scheduling to minimize overtime and waste costs, *IIE Transactions*, 46 (2014), 601–618.

[5] 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 (1979), 287–326.

[6] A. Grigoriev, M. Sviridenko and M. Uetz, Machine scheduling with resource dependent processing times, *Mathematical Programming*, 110 (2007), 209–228.

[7] J. K. Lenstra, D. B. Shmoys and E. Tardos, Approximation algorithms for scheduling unrelated parallel machines, *Mathematical Programming*, 46 (1990), 259–271.

[8] W. Li, J. Li, X. Zhang and Z. Chen, Parallel-machine scheduling problem under the job rejection constraint, in *Proceedings of the 8th Frontiers in Algorithmics (FAW)*, (2014), 158–169.

[9] E. V. Shchepin and N. Vakhania, An optimal rounding gives a better approximation for scheduling unrelated machines, *Operational Research Letters*, 33 (2005), 127–133.

[10] D. B. Shmoys and É. Tardos, An approximation algorithm for the generalized assignment problem, *Mathematical Programming*, 62 (1993), 461–474.

[11] O. Svensson, Santa clause schedules jobs on unrelated machines, *SIAM Journal on Computing*, 41 (2012), 1318–1341.

[12] V. V. Vazirani, *Approximation Algorithms*, Springer, Berlin, 2003.

[13] D. P. Williamson and D. B. Shmoys, *The Design of Approximation Algorithms*, Cambridge University Press, London, 2011.

Received July 2014; revised March 2015.

*E-mail address*: xianzhaozhang@126.com
*E-mail address*: xudc@bjut.edu.cn
*E-mail address*: ddu@unb.ca
*E-mail address*: miaocuixia@126.com