SOP TRANSACTIONS ON APPLIED MATHEMATICS
Accepted version, see http://www.scipublish.com/journals/AM/papers/1097 for published version.
Note there is a minor correction on page 2 of this version as of Aug. 23, 2016.

Approximation Algorithms for Subclasses of the Makespan Problem on Unrelated Parallel Machines with Restricted Processing Times

Daniel R. Page

Department of Computer Science, University of Manitoba
Winnipeg, Manitoba, Canada
drpage@pagewizardgames.com

Abstract—Let there be m parallel machines and n jobs, where a job j can be scheduled on machine i to take $p_{i,j} \in \mathbb{Z}^+$ time units. The makespan C_{max} is the completion time of a machine that finishes last. The goal is to produce a schedule with all n jobs that has minimum makespan. This is the makespan problem on unrelated parallel machines, denoted as $R||C_{max}$. Assume $p, q \in \mathbb{Z}^+$ are constants, let $A(p,q) = \{a \in \mathbb{Z}^+ \mid p \leq a \leq q\}$. We explore a general NP-hard subclass of $R||C_{max}$ when processing times are between p and q inclusive or $p_{i,j} = \infty$ abbreviated as $R|p_{i,j} \in A(p,q) \cup \{\infty\}|C_{max}$, where $p_{i,j} = \infty$ means job j cannot be scheduled to machine i. We give a (q/p)-approximation algorithm for $R|p_{i,j} \in A(p,q) \cup \{\infty\}|C_{max}$. As a consequence, we obtain a 2-approximation algorithm for the NP-hard subclass $R||C_{max}$ with job lengths 1, 2, or $p_{i,j} = \infty$. In addition, we prove $R||C_{max}$ with job lengths 1, 2, and 3 is NP-hard, and present a 3/2-approximation algorithm for this subclass of $R||C_{max}$.

Keywords: approximation algorithms, combinatorial optimization, computational complexity, scheduling, unrelated parallel machines.

I. Introduction

Suppose there are m parallel machines and n jobs to be scheduled non-preemptively. A job j scheduled on machine i takes $p_{i,j}$ time units to complete, where $1 \leq i \leq m$ and $1 \leq j \leq n$. For a given schedule, the makespan is the completion time of a machine that finishes last. The goal is to produce a schedule of all n jobs with minimum makespan. This general scheduling problem is called the *makespan problem on unrelated parallel machines*, which is represented by the Graham notation (see Graham *et al.* [6]) as $R||C_{max}$.

An instance of the makespan problem on UPMs is given by a $m \times n$ matrix $P = (p_{i,j})$ called a *processing requirement matrix*, the number of UPMs m, and the number of jobs n. For a processing requirement matrix P, we denote the optimal makespan as $OPT(P)$. As shorthand, we write $p_{i,j} = \infty$ if job j takes too long to be processed by machine i—restricting jobs to be assigned only to certain machines. Assume without loss of generality that for any processing requirement matrix P that for every column j there exists a row i, such that $p_{i,j} \neq \infty$. Any processing requirement matrix P that does not satisfy this assumption contains jobs that cannot be scheduled on any machine. Since the order jobs are assigned to machines does not matter under this model of scheduling, a schedule can be represented as a 0-1 $m \times n$ matrix $X = (x_{i,j})$ we will call an *assignment matrix.* We say $x_{i,j} = 1$ if job j is scheduled on machine i, and $x_{i,j} = 0$ otherwise.

The scheduling problem $R||C_{max}$ is NP-hard, because a special case, the makespan problem on identical parallel machines (i.e., $P||C_{max}$) when $m = 2$ is NP-hard [5]. Under the assumption that $\mathsf{P} \neq \mathsf{NP}$, researchers seek polynomial-time algorithms that guarantee feasible solutions whose objective value is always within a factor k of the optimal objective value. If a polynomial-time algorithm has such a factor k often called its *approximation factor*, then we call it a *k-approximation algorithm.* Note that an approximation factor is a relative performance guarantee. If a polynomial-time algorithm has instead an absolute performance guarantee k with respect to the optimal objective value, we call it a *k-absolute approximation algorithm.*

Approximation algorithms for $R||C_{max}$ have been presented over the years [8, 1], but in 1990, Lenstra *et al.* [9] gave the first 2-approximation algorithm for $R||C_{max}$. Their algorithm uses linear programming to obtain a fractional feasible schedule, then rounds all remaining fractional jobs by finding a matching. In addition, the authors showed a hardness of approximation result that says there is no p-approximation algorithm for any $p < 3/2$, unless $\mathsf{P} = \mathsf{NP}$. Later Shchepin and Vakhania [12] gave an approximation algorithm that also uses linear programming, but a more refined rounding scheme to obtain an approximation factor of $(2 - 1/m)$. Most recently, Gairing *et al.* [4] showed that finding a feasible fractional schedule with linear programming and performing a rounding can instead be done with generalized network flows, and presented a more efficient 2-approximation algorithm for $R||C_{max}$.

In response to the best known approximation factor of two for $R||C_{max}$, some researchers consider NP-hard subclasses or special cases of $R||C_{max}$ in order to devise a p-approximation algorithm with $p \leq 2$. Two examples are a polynomial-time algorithm that estimates the optimal makespan within a factor 1.9412 for identical parallel machines with restricted assignments (i.e., $R|p_{i,j} \in \{p_j, \infty\}|C_{max}$) presented by Svensson [13], and the 1.75-approximation algorithm by Ebenlendr *et al.* [3, 2] for a subclass of $R||C_{max}$ called graph balancing, which is $R|p_{i,j} \in \{p_j, \infty\}|C_{max}$ with the restriction that at most two entries satisfy $p_{i,j} \neq \infty$ in each column of the processing requirement matrix. In 2014, Vakhania *et al.* [14] developed a q-absolute approximation algorithm that uses linear programming for $R||C_{max}$ with two distinct integral constants for processing times, which is abbreviated as $R|p_{i,j} \in \{p, q\}|C_{max}$. To this date, for any $k < 2$, no k-approximation algorithm has been given for $R|p_{i,j} \in \{p, q, \infty\}|C_{max}$.

Given two fixed positive integers p and q, define $A(p, q) = \{a \in \mathbb{Z}^+ \mid p \leq a \leq q\}$, where $p \leq q$. Consider scheduling on UPMs when processing times are either from $A(p, q)$ or $p_{i,j} = \infty$. Denote this scheduling problem as $R|p_{i,j} \in A(p, q) \cup \{\infty\}|C_{max}$. This problem is NP-hard, because the special case $R|p_{i,j} \in \{p, q\}|C_{max}$ when $q \neq 2p$ is NP-hard [9]. Note that this problem can also be shown to be NP-hard due to $R|p_{i,j} \in \{1, 2, \infty\}|C_{max}$ being NP-hard [9]. This subclass encompasses numerous scheduling problems of interest where either we are interested in restricted processing times, or restricted assignment instances. In Section II, we give a (q/p)-approximation algorithm for this subclass of $R||C_{max}$.

II. (q/p)-approximation algorithm for $R|p_{i,j} \in \mathsf{A}(p, q) \cup \{\infty\}|C_{max}$

To begin, we present an algorithm that uses matching techniques. Our motivation is to exploit known polynomial-time solvable subclasses in order to develop polynomial-time algorithms that need not require linear programming. If we devise an algorithm for $R|p_{i,j} \in A(p, q) \cup \{\infty\}|C_{max}$, then we have developed an approximation algorithm for a large set of pragmatic scheduling problems with restricted processing times, including $R|p_{i,j} \in \{1, 2, \infty\}|C_{max}$ and $R|p_{i,j} \in \{p, q\}|C_{max}$. The approximation algorithm we present as Algorithm 2 utilizes the polynomial-time solvablility [9] of subclass $R|p_{i,j} \in \{1, \infty\}|C_{max}$. A subroutine of Algorithm 2 is to determine in polynomial time if there is a schedule with makespan at most $d \in \mathbb{Z}^+$ for an instance of $R|p_{i,j} \in \{1, \infty\}|C_{max}$, which we provide as Algorithm 1 and is based on the remarks of Lenstra *et al.* [9]. If there is a schedule with makespan at most d, we say the deadline is "met", and "not met" otherwise. If the deadline is met, Algorithm 1 also describes how to construct the schedule.

Algorithm 1. Assume we are given an arbitrary $m \times n$ processing requirement matrix $\Psi = (\psi_{i,j})$, and let $d \in \mathbb{Z}^+$ be a proposed deadline for the makespan of a feasible schedule X. Create a bipartite graph $G = (M \cup J, E)$ that consists of "machine vertices" M and "job vertices" J. We say a machine

vertex $k \in M$ is of *type* i if it corresponds to machine i. Each job vertex corresponds to a job. For each machine i, create d machine vertices of type i, and for each job j, create a job vertex that corresponds to job j. If $\psi_{i,j} = 1$, for each machine vertex k of type i, create an edge $\{k, j\}$. Find a maximum matching M_{max} of G in polynomial time [7], and if $|M_{max}| = n$, then there is a feasible schedule X. To construct X, consider each edge $\{k, j\} \in M_{max}$. For each edge $\{k, j\} \in M_{max}$, let $x_{i,j} = 1$, where machine vertex k is of type i. Finally, if the makespan is at most d, then the deadline is met. Otherwise, the deadline is not met.

It is straightforward to see that if all the job vertices are saturated in the maximum matching M_{max} for some $d \in \mathbb{Z}^+$ in Algorithm 1, the deadline is met. So to obtain an optimal schedule, simply apply binary search to find the smallest d, such that the deadline is met. This yields our approximation algorithm for $R|p_{i,j} \in A(p, q) \cup \{\infty\}|C_{max}$, given as Algorithm 2.

Algorithm 2. Create another $m \times n$ processing requirement matrix $\Psi = (\psi_{i,j})$ as follows. For each entry $p_{i,j} \neq \infty$, let $\psi_{i,j} = 1$, and $\psi_{i,j} = \infty$ otherwise. Every entry of Ψ is either 1 or ∞. Apply binary search over $[\lfloor n/m \rfloor, n]$ using Algorithm 1 to find an optimal schedule for Ψ, then return the best assignment matrix X of Ψ found, and terminate.

Let's show that this simple algorithm is a (q/p)-approximation algorithm for $R|p_{i,j} \in A(p, q) \cup \{\infty\}|C_{max}$.

Theorem 1. Algorithm 2 is a (q/p)-approximation algorithm for $R|p_{i,j} \in A(p, q) \cup \{\infty\}|C_{max}$.

Proof. Assume we are given an arbitrary instance of $R|p_{i,j} \in A(p, q) \cup \{\infty\}|C_{max}$. First, we show Algorithm 2 terminates in polynomial time. Algorithm 2 begins by creating from P a $m \times n$ processing requirement matrix Ψ in $\Theta(mn)$ steps. Then, Algorithm 2 finds an optimal assignment matrix X for Ψ, which can be done in polynomial time [9]. An optimal schedule for Ψ can be found in $O(\log_2(n)(mn)^{5/2})$ steps. What remains to be shown is that the approximation factor of Algorithm 2 is q/p.

Next, list the entries of $A(p, q)$ as $A(p, q) = \{z_1, \dots, z_{|A(p,q)|}\}$. For $b = 1, 2, \dots, |A(p, q)|$, $p \leq z_b \leq q$. Introduce a processing requirement matrix $P' = (p'_{i,j})$ that is P with rescaled job lengths $p'_{i,j} \in \{z_1/p, z_2/p, \dots, z_{|A(p,q)|}/p, \infty\}$. Denote an optimal schedule for P as S^*, where $x^*_{i,j} = 1$ if job j is scheduled on machine i. Let α be a machine that completes last when we apply Algorithm 2. Call the schedule produced S, and denote its corresponding schedule with P' schedule as S'. The makespan of schedule S' is

$$\sum_{i=1}^{n} p'_{\alpha,j} x_{\alpha,j} = \sum_{b=1}^{|A(p,q)|} \sum_{\substack{\text{j=1,} \\ p_{\alpha,j} = z_b}}^{n} \Big((z_b/p) x_{\alpha,j} \Big).$$

Each $z_b \leq q$ and $q/p \geq 1$, so

$$\sum_{b=1}^{|A(p,q)|} \sum_{\substack{\text{j=1,} \\ p_{\alpha,j} = z_b}}^{n} \Big((z_b/p) x_{\alpha,j} \Big) \leq (q/p) \cdot \sum_{j=1}^{n} x_{\alpha,j}. \tag{1}$$

In an optimal schedule for P, let machine ϕ be a machine that is assigned the most jobs. Denote machine β as a machine that completes last for schedule S^*. Observe

$$\sum_{j=1}^{n} x^*_{\phi,j} \leq \sum_{j=1}^{n} p'_{\beta,j} x^*_{\beta,j} = OPT(P)/p. \tag{2}$$

Algorithm 2 then finds an optimal schedule for Ψ. The assignment of jobs for schedules S and S' are determined by the assignments made for Ψ. So, the algorithm produces a feasible schedule for which the maximum number of jobs assigned on each machine is minimized. Then it follows that

$$\sum_{j=1}^{n} x_{\alpha,j} \leq \sum_{j=1}^{n} x^{*}_{\phi,j}, \tag{3}$$

and by (1)–(3),

$$(q/p) \cdot \sum_{j=1}^{n} x_{\alpha,j} \leq (q/p) \cdot \sum_{j=1}^{n} x^{*}_{\phi,j} \leq (q/p) \cdot (OPT(P)/p).$$

Thus, the makespan of schedule S is

$$\sum_{i=1}^{n} p_{\alpha,j} x_{\alpha,j} \leq (q/p) \cdot OPT(P).$$

Therefore, this is a (q/p)-approximation algorithm for $R|p_{i,j} \in A(p,q) \cup \{\infty\}|C_{max}$. $\square$

Though the approximation factor of q/p can be undesirable for any p and q, it is of curiosity when $q/p \leq 2$. Algorithm 2 can guarantee a schedule with makespan for certain values of p and q better than twice the optimal makespan or even better than the hardness of approximation result of $3/2$ for $R||C_{max}$. For example if $p_{i,j} \in \{4,5\}$, then Algorithm 2 is a $5/4$-approximation algorithm, and $5/4 < 3/2$. If $q/p > 2$, then one could instead apply known 2-approximation algorithms [9, 12, 4].

Since $R|p_{i,j} \in \{p,q,\infty\}|C_{max}$ and $R|p_{i,j} \in \{1,2,\infty\}|C_{max}$ are special cases of $R|p_{i,j} \in A(p,q) \cup \{\infty\}|C_{max}$ we obtain the following corollaries.

Corollary 1. Algorithm 2 is a (q/p)-approximation algorithm for $R|p_{i,j} \in \{p,q,\infty\}|C_{max}$.

Corollary 2. Algorithm 2 is a 2-approximation algorithm for $R|p_{i,j} \in \{1,2,\infty\}|C_{max}$.

III. Intractability results for subclasses of $R|p_{i,j} \in \{p,q,r\}|C_{max}$

The target of this section are scheduling problems on UPMs with three distinct processing times. This scheduling problem is a natural representation for scheduling jobs on UPMs with *short*, *medium*, and *long* types of processing times. One pragmatic special case of this subclass is $R|p_{i,j} \in \{1,2,3\}|C_{max}$, which is one of the simplest UPM scheduling models with three processing times. We open this section by giving a NP-complete problem [5] called the r-dimensional matching (rDM) problem.

Problem 1 (r-dimensional matching (rDM))**.** Let $r > 2$. Let there be r disjoint sets $A_1 = \{a_{1,1}, \ldots, a_{1,n'}\}, \ldots, A_r = \{a_{r,1}, \ldots, a_{r,n'}\}$, and a family of r-sets $F = \{T_1, \ldots, T_{m'}\}$, where $|T_i \cap A_j| = 1$ for $i = 1, \ldots, m'$, and for $j = 1, \ldots, r$. Does F contain a subfamily $F' \subseteq F$ called a r-dimensional matching (rDM), where $|F'| = n'$ and $\bigcup_{T_i \in F'} T_i = \bigcup_{t=1}^{r} A_t$?

An instance of the rDM problem is denoted as $(A_1, \ldots, A_r, F, m', n')$, where $A_1, \ldots, A_r$ are the disjoint n'-sets, F is the family of r-sets, m' is the number of r-sets in F, and n' is the number of elements in each disjoint set.

In this section, we develop computational complexity results for subclasses of $R||C_{max}$ that have three distinct processing times, $R|p_{i,j} \in \{p,q,r\}|C_{max}$. In order to do so, we define this problem as a decision problem we call MAKESPANUPM-$\{p,q,r\}$.

Problem 2 (MAKESPANUPM-$\{p,q,r\}$)**.** Let there be three fixed positive integers p, q, r, where $p < q < r$. Given a $m \times n$ processing requirement matrix P with processing times $p_{i,j} \in \{p,q,r\}$, is there a schedule with makespan at most d?

An instance of MAKESPANUPM-$\{p, q, r\}$ is a 4-tuple (P, m, n, d), where P is a $m \times n$ processing requirement matrix, m is the number of machines, n is the number of jobs, and $d \in \mathbb{Z}^+$ is a proposed makespan to be met with a "yes" or "no" answer. Next, we show how to construct a MAKESPANUPM-$\{p, q, r\}$ instance from an instance of the rDM problem in polynomial time.

Construction 1. Assume there are constants $p, q, r \in \mathbb{Z}^+$, where $p < q < r$. Given an instance $I = (A_1, \ldots, A_r, F, n', m')$ of the rDM problem, where $m' \geq n'$, we construct from I an instance $I' = (P, m, n, d)$ of MAKESPANUPM-$\{p, q, r\}$ in polynomial time. Create a $m \times n$ processing requirement matrix P, where $m = m'$ and $n = rn' + p(m' - n')$. There are two types of jobs: dummy jobs, and element jobs. A *dummy job* takes r time units on every machine. For each element in the disjoint sets, there is an element job. For $1 \leq i \leq m'$, an *element job* takes p time units on machine i if element $a_{s,t} \in T_i$ of F and q time units otherwise, where $1 \leq s \leq r$ and $1 \leq t \leq n'$. Let rn' of the jobs be element jobs and $p(m' - n')$ of the jobs be dummy jobs. Assign $d = rp$. Such a processing requirement matrix P consists of elements $p_{i,j} \in \{p, q, r\}$.

We now show a correspondence between the rDM problem and MAKESPANUPM-$\{p, q, r\}$. In particular, we show there is a *rDM F'* if and only if there is a schedule with makespan at most $d = rp$. Our proof extends the correspondence presented by Lenstra *et al* [9] for two-valued subclasses $R|p_{i,j} \in \{p, q\}|C_{max}$ when $q \neq 2p$ to three-valued subclasses $R|p_{i,j} \in \{p, q, r\}|C_{max}$ with $p < q < r$.

Lemma 1. Assume $m' \geq n'$. Given constants $p, q, r \in \mathbb{Z}^+$ where $p < q < r$, and rDM instance $I = (A_1, \ldots, A_r, F, m', n')$, create MAKESPANUPM-$\{p, q, r\}$ instance $I' = (P, m, n, d)$ by applying Construction 1. There is a *rDM F* for I if and only if there exists a schedule for I' with makespan at most $d = rp$.

Proof. First, let us show if there is a rDM for I, then there is a schedule for I' with makespan at most $d = rp$. In order to develop a feasible schedule for I', all the element jobs and dummy jobs need to be scheduled. Since there is a *rDM F'*, each element job corresponds to a distinct element of $A_1 \cup \cdots \cup A_r$. There are n' r-sets in F', each with r distinct elements. Each r-set in F corresponds to a machine. So for each r-set of *rDM F'*, if an element appears in the r-set, schedule its corresponding element job on the machine that corresponds to the r-set. Then n' machines have load rp, and all the element jobs are scheduled. Exactly $(m' - n')$ remaining machines have load 0, and $p(m' - n')$ dummy jobs are left. Schedule on each remaining machine p dummy jobs, and each machine will have load rp. Thus, if there is a rDM, then there is a schedule with makespan at most rp.

Next, we show if there is a schedule for I' that has makespan at most $d = rp$, then there is a *rDM F'* for I. Suppose there is a schedule for I' with makespan at most rp, but there does not exist a *rDM F'* for I. In order to respect the makespan of rp, $p(m' - n')$ dummy jobs must be assigned to exactly $(m' - n')$ machines to take r time units each. This leaves n' machines and rn' element jobs in the schedule. By our assumption, at least one of the element jobs is assigned to the remaining n' machines and takes q time units, where $q > p$. This implies either the schedule is infeasible, or the makespan of the schedule is at least $(r-1)p + q > rp$, which is a contradiction. Thus, if there exists a schedule for I' that has makespan at most rp, then there is rDM for I.

Therefore, there is a *rDM F'* for I if and only if there is a schedule with makespan at most rp for I'. □

Theorem 2. $R|p_{i,j} \in \{1, 2, 3\}|C_{max}$ is NP-hard.

Proof. Consider the decision variant of $R|p_{i,j} \in \{1, 2, 3\}|C_{max}$, MAKESPANUPM-$\{1, 2, 3\}$ (i.e. a special case of Problem 2). We show $R|p_{i,j} \in \{1, 2, 3\}|C_{max}$ is NP-hard by proving that MAKESPANUPM-$\{1, 2, 3\}$ is NP-complete. To do this, we reduce from the 3DM problem (i.e., a special case of Problem 1), which is NP-complete [5]. Observe that the makespan can be computed in $O(mn)$ steps, so MAKESPANUPM-$\{1, 2, 3\} \in \mathsf{NP}$. What remains to be shown is that 3DM $\leq_P$ MAKESPANUPM-$\{1, 2, 3\}$.

First, let's describe how to construct the MAKESPANUPM-$\{1, 2, 3\}$ instance $I' = (P, m, n, d)$ from any 3DM problem instance $I = (A_1, \ldots, A_r, F, m', n')$ in polynomial time. If $m' \geq n'$, apply Construction 1 with $p = 1$, $q = 2$, and $r = 3$ to obtain I'. By Lemma 1, if there is a feasible schedule for I' with makespan at most $d = 3$, then there is a $3DM$, and say "yes". Otherwise if $m' < n'$, or there does not exist a schedule for I' with makespan at most $d = 3$, say "no". Thus, 3DM $\leq_P$ MAKESPANUPM-$\{1, 2, 3\}$, and MAKESPANUPM-$\{1, 2, 3\}$ is NP-complete.

Therefore, $R|p_{i,j} \in \{1, 2, 3\}|C_{max}$ is NP-hard. □

Applying Lemma 1 for fixed values of p, q, r, we obtain the subsequent results from the argument for Theorem 2.

Corollary 3. $R|p_{i,j} \in \{1, 2, 4\}|C_{max}$ is NP-hard.

Corollary 4. Define constants $p, q, r \in \mathbb{Z}^+$, where $p < q < r$. The scheduling problem $R|p_{i,j} \in \{p, q, r\}|C_{max}$ is NP-hard.

IV. 3/2-approximation algorithm for $R|p_{i,j} \in \{1, 2, 3\}|C_{max}$

In Section III, we showed $R|p_{i,j} \in \{1, 2, 3\}|C_{max}$ is NP-hard. This gives us motivation to develop an approximation algorithm for this scheduling problem. Our goal is to develop a simple to implement algorithm that employs matching techniques as opposed to linear programming. We show that for this subclass, there exists an approximation algorithm that has an approximation factor that matches the hardness of approximation results of $R||C_{max}$. This result is the first of its kind for a NP-hard scheduling problem on UPMs with three different processing times. First, we present our algorithm as Algorithm 3, which adapts a known polynomial-time solvable result by Lenstra *et al.* [9] for $R|p_{i,j} \in \{1, 2\}|C_{max}$. Note that in 2014, Vakhania *et al.* [14] also gave a polynomial-time algorithm that uses linear programming for $R|p_{i,j} \in \{1, 2\}|C_{max}$. Similar linear programming techniques have also been applied in the study of geometric graphs (see [10, 11]).

Algorithm 3. Construct a new $m \times n$ processing requirement matrix P' that will be an instance of $R|p_{i,j} \in \{1, 2\}|C_{max}$. For $i = 1, \ldots, n$ and $j = 1, \ldots, m$, if $p_{i,j} > 2$, then $p'_{i,j} = 2$. Otherwise, let $p'_{i,j} = p_{i,j}$. The result is an instance of the makespan problem on UPMs with processing times $p_{i,j} \in \{1, 2\}$. Next, find an optimal schedule for P', which can be done in polynomial time [9], and call its assignment matrix X. Return the assignment matrix X, then terminate.

Next, we show that Algorithm 3 is a 3/2-approximation algorithm for $R|p_{i,j} \in \{1, 2, 3\}|C_{max}$.

Theorem 3. Algorithm 3 is a 3/2-approximation algorithm for $R|p_{i,j} \in \{1, 2, 3\}|C_{max}$.

Proof. Suppose we are given an arbitrary $m \times n$ processing requirement matrix P with processing times $p_{i,j} \in \{1, 2, 3\}$. First we show that Algorithm 3 terminates in polynomial time. To begin, Algorithm 3 constructs P' in at most $\Theta(mn)$ steps. Next, the algorithm finds an optimal schedule for P' in polynomial time [9]. The algorithm terminates after returning the resulting assignment matrix X, so Algorithm 3 terminates in polynomial time. Now we show that Algorithm 3 when applied to any instance of $R|p_{i,j} \in \{1, 2, 3\}|C_{max}$ always produces a feasible schedule, and has approximation factor 3/2.

Consider the assignment matrix X returned as a result of Algorithm 3, call its corresponding schedule S. Also, let S^* be an optimal schedule for P, and its associated assignment matrix be X^*. Since an optimal schedule for P' is found, the schedule S produced by Algorithm 3 is feasible. Let α be a machine that finishes last in the schedule produced by Algorithm 3. Also, let β be a machine that finishes last in schedule S^*. Since X consists of assignments that correspond to an optimal schedule for P',

$$\sum_{j=1}^{n} p'_{\alpha,j} x_{\alpha,j} \leq \sum_{j=1}^{n} p_{\beta,j} x^*_{\beta,j} = OPT(P) \tag{4}$$

Each entry in P that is 3 time units is replaced with 2 time units in P' by Algorithm 3. By (4), the makespan of the schedule S is

$$\sum_{j=1}^{n} p_{\alpha,j} x_{\alpha,j} \leq \sum_{j=1}^{n} ((3/2) p'_{\alpha,j}) x_{\alpha,j} = (3/2) \sum_{j=1}^{n} p'_{\alpha,j} x_{\alpha,j} \leq (3/2) \cdot OPT(P).$$

Therefore, Algorithm 3 is a 3/2-approximation algorithm. □

Consider $R|p_{i,j} \in \{1,2,4\}|C_{max}$. It is straightforward to see that an argument similar to the one given for Theorem 3 yields the following result for $R|p_{i,j} \in \{1,2,4\}|C_{max}$.

Corollary 5. Algorithm 3 is a 2-approximation algorithm for $R|p_{i,j} \in \{1,2,4\}|C_{max}$.

The approximation factor of Algorithm 3 for $R|p_{i,j} \in \{1,2,4\}|C_{max}$ is on par with the best known for $R||C_{max}$.

V. Conclusion

As a consequence of this work, an open problem stands for the NP-hard subclass $R|p_{i,j} \in A(p,q) \cup \{\infty\}|C_{max}$, and many of its NP-hard special cases, such as $R|p_{i,j} \in \{p,q\}|C_{max}$ and $R|p \leq p_{i,j} \leq q|C_{max}$. Since there exists a (q/p)-approximation algorithm for the NP-hard subclass $R|p_{i,j} \in A(p,q) \cup \{\infty\}|C_{max}$, the hardness of approximation results for $R||C_{max}$ do not apply when $q/p < 3/2$. The fact that $R|p_{i,j} \in A(p,q) \cup \{\infty\}|C_{max}$ does not apply to the general hardness of approximation result for $R||C_{max}$ may indicate that a PTAS exists. Is there a PTAS for $R|p_{i,j} \in A(p,q) \cup \{\infty\}|C_{max}$? If there is no PTAS for this subclass, are there any non-trivial hardness of approximation results for $R|p_{i,j} \in A(p,q) \cup \{\infty\}|C_{max}$?

Another avenue of research that has not yet been investigated in the literature thoroughly is the makespan problem on UPMs with three distinct processing times, denoted as $R|p_{i,j} \in \{p,q,r\}|C_{max}$. Currently there is little in the literature on this subclass of $R||C_{max}$ Also, is there an approximation algorithm with approximation factor better than 3/2 for $R|p_{i,j} \in \{1,2,3\}|C_{max}$ that does not utilize linear programming, but instead, matching techniques?

By considering the general NP-hard subclass $R|p_{i,j} \in A(p,q) \cup \{\infty\}|C_{max}$, we developed an efficient (q/p)-approximation algorithm that can guarantee feasible schedules with makespan closer to optimal using basic matching techniques than the best known results whenever $q/p \leq 2$. We also extended the known intractability results from two distinct processing times to three distinct processing times. We hope our contributions will motivate researchers to pursue more NP-hard subclasses and tractable subclasses of the makespan problem on UPMs with restricted processing times.

VI. Acknowledgments

Thank you Dr. (Ben) P.C. Li for supervising and providing feedback during the development of this research at the University of Manitoba.

References

[1] E. Davis and J. Jaffe. Algorithms for scheduling tasks on unrelated processors. *J. of the ACM (JACM)*, 28(4):721–736, 1981.

[2] T. Ebenlendr, M. Krčál, and J. Sgall. Graph balancing: A special case of scheduling unrelated parallel machines. *Algorithmica*, 68(1):62–80, 2014.

[3] T. Ebenlendr, M. Krčál, and J. Sgall. Graph balancing: A special case of scheduling unrelated parallel machines. In *Proc. of the* 19th *Annual ACM–SIAM Symposium on Discrete algorithms (SODA)*, pages 483–490, 2008.

[4] M. Gairing, B. Monien, and A. Woclaw. A faster combinatorial approximation algorithm for scheduling unrelated parallel machines. *Theoretical Computer Science*, 380(1):87–99, 2007.

[5] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness.* Freeman New York, 1979.

[6] R. Graham, E. Lawler, J. Lenstra, and K. Rinnooy. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 5:287–326, 1979.

[7] J. Hopcroft and Richard M Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. on Computing*, 2(4):225–231, 1973.

[8] O. Ibarra and C. Kim. Heuristic algorithms for scheduling independent tasks on nonidentical processors. *J. of the ACM (JACM)*, 24(2):280–289, 1977.

[9] J. Lenstra, D. Shmoys, and E. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Program.*, 46(1–3):259–271, 1990.

[10] C. McDiarmid and T. Müller. On the chromatic number of random geometric graphs. *Combinatorica*, 31(4):423–488, 2011.

[11] Y. Shang. Improper coloring of random geometric graphs. *Journal of Advanced Research in Applied Mathematics*, 4(1):1–9, 2012.

[12] E. Shchepin and N. Vakhania. An optimal rounding gives a better approximation for scheduling unrelated machines. *Operations Res. Letters*, 33:127–133, 2005.

[13] O. Svensson. Santa Claus schedules jobs on unrelated machines. In *STOC'11 Proc. of the* 43rd *ACM Symposium on Theory of Computing*, pages 617–626, New York, 2011. ACM.

[14] N. Vakhania, J. Hernandez, and F. Werner. Scheduling unrelated machines with two types of jobs. *International J. of Production Res.*, 53(13):3793–3801, 2014.