Makespan Minimization on Unrelated Parallel Machines with a Few Bags*

Daniel R. Page[0000−0002−6317−7431](✉) and Roberto Solis-Oba

Department of Computer Science, Western University, London, Canada
dpage6@uwo.ca, solis@csd.uwo.ca

Abstract. Let there be a set M of m parallel machines and a set J of n jobs, where each job j takes $p_{i,j}$ time units on machine M_i. In makespan minimization the goal is to schedule each job non-preemptively on a machine such that the length of the schedule, the makespan, is minimum. We investigate a generalization of makespan minimization on unrelated parallel machines ($R||C_{max}$) where J is partitioned into b bags $B = (B_1, \ldots, B_b)$, and no two jobs belonging to the same bag can be scheduled on the same machine. First we present a simple b-approximation algorithm for $R||C_{max}$ with bags ($R|bag|C_{max}$). Two machines M_i and $M_{i'}$ have the same machine type if $p_{i,j} = p_{i',j}$ for all $j \in J$. We give a polynomial-time approximation scheme (PTAS) for $R|bag|C_{max}$ with machine types where both the number of machine types and bags are constant. This result infers the existence of a PTAS for uniform parallel machines when the number of machine speeds and number of bags are both constant. Then, we present a $b/2$-approximation algorithm for the graph balancing problem with $b \geq 2$ bags; the approximation ratio is tight for $b = 3$ unless $\mathsf{P} = \mathsf{NP}$ and this algorithm solves the graph balancing problem with $b = 2$ bags in polynomial time. In addition, we present a polynomial-time algorithm for the restricted assignment problem on uniform parallel machines when all the jobs have unit length. To complement our algorithmic results, we show that when the jobs have lengths 1 or 2 it is NP-hard to approximate the makespan with approximation ratio less than 3/2 for both the restricted assignment and graph balancing problems with $b = 2$ bags and $b = 3$ bags, respectively. We also prove that makespan minimization on uniform parallel machines with $b = 2$ bags is strongly NP-hard.

Keywords: makespan minimization · unrelated parallel machines · approximation algorithms · scheduling · bag constraints.

This is an accepted version of our paper that has been accepted at the 12th International Conference on Algorithmic Aspects in Information and Management (AAIM 2018), Dallas, USA, December 3–4, 2018. The final authenticated version is available online at https://doi.org/10.1007/978-3-030-04618-7_3.

* Daniel Page is supported by an Ontario Graduate Scholarship. Roberto Solis-Oba was partially supported by the Natural Sciences and Engineering Research Council of Canada, grant 04667-2015 RGPIN.

1 Introduction

Let M be a set of m *unrelated parallel machines* and J be a set of n *jobs*, where job j has length or processing time $p_{i,j} \in \mathbb{Z}^+$ on machine M_i. In makespan minimization, the goal is to schedule all the jobs on the machines so as to minimize the length of the schedule—the *makespan*. Makespan minimization on unrelated parallel machines is denoted as $R||C_{max}$ in the notation of Graham *et al.* [11]. Two extensively studied machine environments that are special cases of the unrelated parallel machine environment are the identical and uniform parallel machine environments: if the machines are *identical* then job j has the same *length* $p_j \in \mathbb{Z}^+$ on any machine; and if the machines are *uniform* then each machine M_i has a *speed* $s_i \in \mathbb{Z}^+$ and job j has processing time p_j/s_i on M_i.

We consider a generalization of $R||C_{max}$ where the jobs J are partitioned into b sets $B = (B_1, B_2, \ldots, B_b)$ called *bags*, and any feasible solution must satisfy the *bag constraints*: no two jobs from the same bag can be scheduled on the same machine. This problem is called problem *makespan minimization on unrelated parallel machines with bags*, and we denote it as $R|bag|C_{max}$. Notice that if $b = |J|$, then every job is in a distinct bag, and we get the classic setting $R||C_{max}$. As discussed in [7], the bag constraints appear in settings such as in the scheduling of tasks for on-board computers in airplanes. That is, these systems have multiple processors and it is required for some of the tasks to be scheduled on different processors so that the airplane continues to operate safely even if one of the processors were to fail. Thus, parallel machine scheduling problems where the bag constraints are imposed are a kind of fault-tolerant scheduling that finds applications in complex parallel systems where system stability is desired [4].

The best-known approximation algorithms for $R||C_{max}$ have approximation ratio 2 [8,16,19], and it is NP-hard to approximate the makespan with approximation ratio less than 3/2 [16]. Two special cases of $R||C_{max}$ have become of recent interest to try to understand the 3/2-to-2 approximation gap for $R||C_{max}$:

- *Restricted Assignment Problem* ($P|\mathcal{M}_j|C_{max}$). This is makespan minimization on identical parallel machines (i.e., $P||C_{max}$) with the constraint that some jobs are not eligible to be scheduled on some of the machines. That is, for each job $j \in J$, there is a set $\mathcal{M}_j$ of machines where job j can be scheduled. A schedule that assigns each job j to an eligible machine in $\mathcal{M}_j$ is said to satisfy the *eligibility constraints*.
- *Graph Balancing Problem* ($P|\mathcal{M}_j, |\mathcal{M}_j| \leq 2|C_{max}$): This is a special case of the restricted assignment problem where the number of eligible machines for each job is at most 2. An alternate way to interpret an instance of this problem is as a weighted multigraph where the jobs are edges and the machines are vertices, and every edge must be directed to one of its endpoints so as to minimize the maximum sum of the edge lengths directed toward a vertex.

We study variants of the above two problems with bag constraints. In addition to this, we investigate $R|bag|C_{max}$ in the setting with so-called machine types. As discussed by Gehrke *et al.* [10], a natural scenario in parallel machine scheduling is where the machines are clusters of processors where each processor in a cluster

is of the same type, e.g. clusters of CPUs and/or GPUs. More formally, two machines M_i and $M_{i'}$ have the same *machine type* if $p_{i,j} = p_{i',j}$ for all $j \in J$. We study $R|bag|C_{max}$ *with machine types*, where the number δ of machine types is constant.

2 Related Work

For the restricted assignment problem with two different job lengths $p_j \in \{\alpha, \beta\}$ for each job j and $\alpha < \beta$, there are approximation algorithms with approximation ratio slightly less than 2 [3,15], most notably a $(2-\gamma)$-approximation algorithm for some small value $\gamma > 0$ and a $(2-\alpha/\beta)$-approximation algorithm, both by Chakrabarty *et al.* [3]. Jansen and Rohwedder [14] showed that in quasi-polynomial time the restricted assignment problem can be approximated within a factor $11/6+\epsilon$ of the optimum for any $\epsilon > 0$. Ebenlendr *et al.* [6] presented a 7/4-approximation algorithm for the graph balancing problem. For the graph balancing problem with two job lengths there are 3/2-approximation algorithms [12,18], and there is no p-approximation algorithm with $p < 3/2$, unless $\mathsf{P} = \mathsf{NP}$ [1,6]. Jansen and Maack [13] presented an efficient PTAS for $R||C_{max}$ with machine types when the number of machine types is constant. For more literature on makespan minimization with machine types see [10,13].

Makespan minimization with bags is a type of conflict scheduling problem, where two jobs conflict if two jobs from the same bag are scheduled on the same machine. A natural way to model this type of conflict is with an incompatibility graph: there is a vertex for each job and an edge $\{j, j'\}$ if jobs j and j' cannot be scheduled on the same machine. Then, makespan minimization with bags is when the incompatibility graph consists of b disjoint cliques. Bodlaender *et al.* [2] developed several results for $P||C_{max}$ with incompatibility graphs. In addition, Dokka *et al.* [5] considered a related, but generalized version of $P|bag|C_{max}$ called the multi-level bottleneck assignment problem, and gave a 2-approximation algorithm for three bags. For further discussion on related variants of conflict scheduling refer to Section 1.3 of [4].

In 2017, Das and Wiese [4] presented a PTAS for $P|bag|C_{max}$, and an 8-approximation algorithm for the restricted assignment problem with bags in the special case when for each bag B_k all the jobs $j \in B_k$ have the same eligibility constraints, i.e. each set of machines on which a job in B_k can be scheduled is the same. For any $\epsilon > 0$, Das and Wiese proved there is no $\left((\log n)^{1/4-\epsilon}\right)$-approximation algorithm for the restricted assignment problem with bags, unless $\mathsf{NP} \subseteq \mathsf{ZPTIME}(2^{(\log n)^{O(1)}})$.

3 Preliminaries

First, we give a couple of basic properties for $R|bag|C_{max}$. If the number b of bags is one, at most one job can be scheduled on each machine. Hence, we can solve $R||C_{max}$ with one bag in polynomial time as follows: build a weighted bipartite

graph $G = (J \cup M, E)$, where $E = \{(j,i) \mid \text{job } j \text{ can be scheduled on machine } M_i\}$, and $w(j,i) = p_{i,j}$ for every $(j,i) \in E$. Compute a maximum cardinality bottleneck matching $\mathscr{M}$ of G and for each arc $(j,i) \in \mathscr{M}$, schedule job j on machine M_i; there is no feasible solution if any job is not scheduled. Thus, in the sequel we focus on $R|bag|C_{max}$ when there are $b > 1$ bags.

Property 1. For any schedule that satisfies the bag constraints with b bags, there are at most b jobs scheduled on a machine.

For some of our algorithmic results, we employ a *p-relaxed decision procedure* as given in Lenstra *et al.* [16]. Let $U \in \mathbb{Z}^+$ be an upper bound on the optimal makespan for some scheduling problem. We use binary search over the interval $[0, U]$ to determine the smallest value $d \in \mathbb{Z}^+$ for which the p-relaxed decision algorithm either: computes a schedule with makespan at most pd; or returns `FAIL` if there is no solution with value at most d. In the binary search, if the p-relaxed decision algorithm returns `FAIL` then the value d is increased, and if a schedule is returned the value d is decreased. If we keep track of the schedule with minimum makespan found, after $O(\log U)$ iterations the binary search guarantees that $d \leq OPT$ and a schedule with makespan at most $p \cdot OPT$ is found. Therefore, if the overall p-relaxed decision procedure takes polynomial time, this is a p-approximation algorithm.

4 Our Results

In Section 5 we provide a simple b-approximation algorithm for $R|bag|C_{max}$. In Section 6 we present a PTAS for $R|bag|C_{max}$ with machine types when there is a fixed number of machine types and bags. As we will explain, this implies the existence of a PTAS for $Q|bag|C_{max}$ when both the number of machine speeds and the number of bags are constant. Then, in Section 7 we give a $b/2$-approximation algorithm for the graph balancing problem with $b \geq 2$ bags, the approximation ratio for this algorithm is tight for $b = 3$ unless $\mathsf{P} = \mathsf{NP}$ and implies that the graph balancing problem with $b = 2$ bags is solvable in polynomial time. Finally, in Section 8 we show that the restricted assignment problem with bags when the machines are uniform and every job has unit length $(Q|bag, p_j = 1, \mathcal{M}_j|C_{max})$ is polynomial-time solvable. As a note, we designed a $O(m \log m)$-time algorithm for $P|bag|C_{max}$ with $b = 2$ bags.

To complement our algorithmic results, we present a series of inapproximability and strong NP-hardness results in Section 9. We first show how to extend the classic 3/2-inapproximability lower bound of Lenstra *et al.* [16] to the restricted assignment problem with job lengths $p_j \in \{1, 2\}$ when there are $b = 2$ bags. Then, we prove that there is no approximation algorithm with approximation ratio less than 3/2 for the graph balancing problem with $b = 3$ bags and job lengths $p_j \in \{1, 2\}$, unless $\mathsf{P} = \mathsf{NP}$. Finally we show that $Q|bag|C_{max}$ with $b = 2$ bags is strongly NP-hard.

5 A b-Approximation Algorithm for $R|bag|C_{max}$

Let $p_{max} = \max_{1 \leq j \leq n, 1 \leq i \leq m}(p_{i,j})$. Our approximation algorithm uses binary search and a b-relaxed decision procedure with $U = p_{max}n$. For makespan estimate $d \leq U$, the idea is to treat each bag independently and simply schedule the jobs j in each bag $B_k \in B$ on machines M_i where $p_{i,j} \leq d$ so that the bag constraints are satisfied. We do this by using a bipartite flow network $N(B, P, d)$ where there is a source s, a *job node* for each $j \in J$, a *bag-machine node* $M_{B_k,i}$ for each $M_i \in M$ and $B_k \in B$, a *machine node* for every $M_i \in M$, and a sink t. Do the following for each bag $B_k \in B$: for each $j \in B_k$, add an arc from s to each job node j and set its capacity to 1. Next, for each $j \in B_k$ and $M_i \in M$, if $p_{i,j} \leq d$, then add an arc from job node j to $M_{B_k,i}$ with capacity 1. For each $M_i \in M$ and $B_k \in B$, add an arc from each bag-machine node $M_{B_k,i}$ to machine node M_i with capacity 1. Finally, from each machine node $M_i \in M$, add an arc from M_i to t with capacity b. The b-relaxed decision algorithm is as follows.

1. Build flow network $N(B, P, d)$ and compute an integral maximum flow f.
2. **For each** $j \in B_k$, if $f(s, j) = 0$ **then return** `FAIL`.
3. **For each** job $j \in B_k$, schedule j on machine M_i **if** $f(j, M_{k,i}) = 1$, and **return** this schedule.

If an integral maximum flow is computed and all the arcs incident on s are saturated, the jobs in bag B_k can be scheduled on the machines so as to satisfy the bag constraints, and such that each job takes at most d time units.

Theorem 1. *There is b-approximation algorithm for $R|bag|C_{max}$.*

6 A PTAS for $R|bag|C_{max}$ with a Constant Number of Machine Types and Bags

Recall that two machines M_i and $M_{i'}$ have the same machine type if, for every $j \in J$, $p_{i,j} = p_{i',j}$. Let $N_t(\upsilon)$ be the number of machines of machine type υ, and let δ be the number of machine types. Now we describe the PTAS. Compute a b-approximate value ρ to the optimal makespan, so that $\rho/b \leq OPT \leq \rho$; value ρ can be computed using the b-approximation algorithm in Section 5. Then use binary search over the interval $[\rho/b, \rho]$ to find the smallest value τ for which the algorithm given below computes a schedule. Consider all possible schedules of length τ. We can simplify the structure of these schedules so that there is only a constant number of different load configurations for the machines (defined below), while increasing the length of the schedule by a factor of at most $(1 + \epsilon)$ for any constant $\epsilon > 0$. This simplification will allow us to design a PTAS.

First subdivide the timeline of a schedule into units of length $(\tau\epsilon)/b^2$ for some constant $\epsilon > 0$. Let the *load configuration* $(\ell_{i,1}, \ell_{i,2}, \ldots, \ell_{i,b})$ of machine M_i be such that if job j from bag B_k is scheduled on M_i then $(\ell_{i,k} - 1)(\tau\epsilon)/b^2 < p_{i,j} \leq \ell_{i,k} \cdot (\tau\epsilon)/b^2$. The number of possible values each $\ell_{i,k}$ can take is at most $1 + \lceil \frac{\tau}{\frac{\tau\epsilon}{b^2}} \rceil = 1 + \lceil \frac{b^2}{\epsilon} \rceil$, and so $\ell_{i,k} \in \{0, 1, 2, \ldots, \lceil b^2/\epsilon \rceil\}$. As there are b values in any load configuration, the number of possible load configurations is then

$(1 + \lceil b^2/\epsilon \rceil)^b =: D$, a constant; label these load configurations $1, 2, \ldots, D$. Let vector $(c_{1,1}, c_{1,2}, \ldots, c_{1,D}, c_{2,1}, c_{2,2}, \ldots, c_{2,D}, \ldots, c_{\delta,1}, c_{\delta,2}, \ldots, c_{\delta,D})$ be a *schedule configuration*, where $c_{\upsilon,\mu}$ is the number of machines with machine type υ that have load configuration μ. There are m machines, so each $c_{\upsilon,\mu} \in \{0, 1, \ldots, m\}$ and because there are δD many elements in a schedule configuration, the total possible number of schedule configurations is $O(m^{\delta D})$, a polynomial function as δ and D are constant. A schedule configuration is *valid* if $\sum_{\upsilon=1}^{\delta} \sum_{\mu=1}^{D} c_{\upsilon,\mu} = m$, and $\sum_{\mu=1}^{D} c_{\upsilon,\mu} = N_t(\upsilon)$, for $\upsilon = 1, 2, \ldots, \delta$. For each valid schedule configuration there are exactly m load configurations, one for each machine. That is, for each $c_{\upsilon,\mu} > 0$, assign load configuration μ to $c_{\upsilon,\mu}$ machines of machine type υ. Then, the makespan of a valid schedule configuration is $\max_{1 \le i \le m} \left\{ \sum_{k=1}^{b} \ell_{i,k}(\frac{\tau\epsilon}{b^2}) \right\}$.

We compute all valid schedule configurations for makespan τ and choose one for which the jobs can be allocated to the machines according to that schedule configuration. If there is a feasible schedule, at least one such schedule configuration exists where for each $j \in J$ with $j \in B_k$, there is a machine M_i with $p_{i,j} \le \ell_{i,k}(\frac{\tau\epsilon}{b^2}) \le \lceil \frac{b^2}{\epsilon} \rceil (\frac{\tau\epsilon}{b^2})$ where $\frac{b^2}{\epsilon}(\frac{\tau\epsilon}{b^2}) = \tau \le \lceil \frac{b^2}{\epsilon} \rceil (\frac{\tau\epsilon}{b^2})$. To find this schedule we proceed as follows. For each valid schedule configuration, assign to machine M_i a load configuration L_i as described above. Then consider each bag B_k, $k = 1, 2, \ldots, b$, and build a bipartite graph $G_k = (B_k \cup M, E_k)$, where $E_k = \left\{ (j, M_i) \mid M_i \in M,\ j \in B_k, (\ell_{i,k} - 1)\left(\frac{\tau\epsilon}{b^2}\right) < p_{i,j} \le \ell_{i,k}\left(\frac{\tau\epsilon}{b^2}\right) \right\}$. Compute a maximum matching of G_k, and for each arc (j, M_i) in the matching, schedule j on M_i. Discard the schedule if at least one job of bag B_k is not scheduled. Otherwise, for $k = 1, 2, \ldots, b$, a matching of size $|B_k|$ is computed for G_k and thus every job $j \in B_k$ is scheduled. This will assign at most one job from each bag B_k to each machine, so a feasible schedule is produced.

Let machine M_λ be a machine that finishes last in the schedule configuration with minimum makespan τ^* selected by the algorithm and let $L^*_\lambda = (\ell^*_{\lambda,1}, \ell^*_{\lambda,1}, \ldots, \ell^*_{\lambda,b})$ be its load configuration. Note that for each job $j \in B_k$ on M_λ, $p_{\lambda,j} \le \ell^*_{\lambda,k}(\tau^*\epsilon)/b^2$, but $p_{\lambda,j} > (\ell^*_{\lambda,k} - 1)(\tau^*\epsilon)/b^2$ as otherwise there would be another schedule configuration of lesser makespan where all the jobs can be allocated to the machines. Since $\tau^* \ge OPT$, then

$$\sum_{\substack{\text{job } j \text{ scheduled} \\ \text{on machine } M_\lambda}} p_{\lambda,j} \ge OPT > \sum_{k=1}^{b} \max\left\{ (\ell^*_{\lambda,k} - 1)\frac{\tau^*\epsilon}{b^2}, 0 \right\}.$$

Therefore,

$$\sum_{\substack{\text{job } j \text{ scheduled} \\ \text{on machine } M_\lambda}} p_{\lambda,j} \le \sum_{k=1}^{b} \ell^*_{\lambda,k}\frac{\tau^*\epsilon}{b^2} \le \sum_{k=1}^{b} \max\left\{ (\ell^*_{\lambda,k} - 1)\frac{\tau^*\epsilon}{b^2}, 0 \right\} + \sum_{k=1}^{b} \frac{\tau^*\epsilon}{b^2}$$

$$< OPT + \sum_{k=1}^{b} \frac{\tau^*\epsilon}{b^2},$$

and since $\rho/b \leq OPT \leq \tau^* \leq \rho$, the makespan is at most

$$OPT + \sum_{k=1}^{b} \frac{\tau^* \epsilon}{b^2} = OPT + \left(\frac{\tau^*}{b}\right)\epsilon \leq OPT + \left(\frac{\rho}{b}\right)\epsilon \leq (1+\epsilon)OPT.$$

Theorem 2. *There is a PTAS for $R|bag|C_{max}$ with machine types when both the number b of bags and the number δ of machine types are constant.*

Consider makespan minimization on uniform machines with bags ($Q|bag|C_{max}$). The processing time for a job on a machine depends on the speed of the machine. Therefore, the number of machine types is in fact the number of machine speeds.

Corollary 1. *There is a PTAS for $Q|bag|C_{max}$ when both the number of distinct machine speeds and the number of bags are constant.*

7 A $b/2$-Approximation Algorithm for the Graph Balancing Problem with $b \geq 2$ Bags

Recall that in the graph balancing problem with bags the jobs and machines can be represented as a weighted multigraph $G = (V, E)$, where the jobs are edges i.e. $E = \bigcup_{k=1}^{b} B_k$, each edge $e \in E$ has length $p_e \in \mathbb{Z}^+$, and the machines are the vertices. We continue to use m and n to be the number of machines and jobs, respectively. Let $G_{B_k} = (V_{B_k}, B_k)$ where vertex $v \in V_{B_k}$ if $v \in e \in B_k$. We call a maximally connected component of G_{B_k} a *bag component*. A *pseudoforest* is a collection of trees and graphs with at most one cycle called 1-*trees*.

Property 2. Consider the graph balancing problem with bags. If there is a feasible schedule S, then for every $B_k \in B$, G_{B_k} is a pseudoforest.

In the sequel we assume that the input multigraph G satisfies the conditions of Property 2. There are at most two possible orientations for a bag component that is a 1-tree: direct each edge to a unique vertex along the cycle of the 1-tree, and then direct all other edges away from the cycle. A tree $T = (V_T, E_T)$ however has at most $|V_T|$ possible orientations: select each vertex as the root of the tree and direct all edges away from it. We use these facts in our algorithm.

We use binary search and a $b/2$-relaxed decision procedure as described in Section 3 with $U = \sum_{e \in E} p_e$ to find the smallest value $d \in \mathbb{Z}^+$ for which there is a schedule with makespan at most $(b/2)d$ that satisfies the bag constraints. If the $b/2$-relaxed decision algorithm below returns `FAIL`, then there is no feasible schedule with makespan at most d for G; otherwise the algorithm computes a feasible schedule with makespan at most $(b/2)d$. Let $l_L(u)$ be the load contributed by the edge with the largest edge length directed toward vertex u in G; hence $l_L(u) = 0$ if no edge is directed toward u. We note that if $l_L(u) > d/2$ then no other edges with length larger than $d/2$ can be directed toward u without the

makespan exceeding d; we call an edge e a *big* edge when its length $p_e > d/2$ and an edge is *small* if $p_e \leq d/2$.

The $b/2$-relaxed decision algorithm uses a set D to store the edges that have been assigned a direction. Initially $D = \varnothing$ and if a schedule exists, at the end D will contain all the edges in G. First, if any edge $e \in E$ has length larger than d return `FAIL`. While there is an edge in $E \setminus D$ do the following. Compute a bag component C of $G \setminus D$ and perform an *expansion* of C by using the procedure described in Step (2) of the algorithm below. For each feasible orientation of the edges in C an expansion forces edges away from vertices in C if doing so does not violate the bag constraints and $l_L(u) + p_e \leq d$ for every $u \in C$ and edge e incident on u. The forcing of edges away "expands" C and this process will continue "expanding" C until no more edges need to be forced in a certain direction or infeasibility is determined. If an expansion is successfully computed, directions for a set C_E of edges is found and so we set $D = D \cup C_E$. The process is then repeated if there are any undirected edges left in $E \setminus D$. Otherwise if no expansion was found another orientation for C is considered and another expansion is computed. The algorithm returns `FAIL` if there are no more orientations to try. We assume below that each $l_L(u)$ is updated as the direction of edges are changed. Now we formally describe the algorithm.

1. Set $D = \varnothing$. If any edge $e \in E$ has length $p_e > d$, return `FAIL`.
2. **While** $E \setminus D$ is not empty:
   (a) Compute a bag component C of $G \setminus D$.
   (b) Find a new orientation of the edges in C for which at most one edge from each bag is directed to the same vertex and any two edges e, e' directed to the same vertex satisfy $p_e + p_{e'} \leq d$. If there are no more new orientations to try for C **return** `FAIL`. Let C_v be the set of vertices u where an edge is directed toward u by this step.
   (c) **While** there is a vertex $u \in C_v$ and undirected edge $e = \{u, v\}$ in $E \setminus D$ where $l_L(u) + p_e > d$:
      i. Direct e from u to v; then direct all edges of the same bag as e that are reachable from v away from u. Add to C_v all vertices whose loads increased in this step.
      ii. **If** any vertex $w \in C_v$ has two edges from the same bag directed toward it or **if** there are two edges e and e' directed toward w so that $p_e + p_{e'} > d$ **then** reset all loads and edges directed by this iteration of Step (2) and go to Step (2b).
   (d) Let C_E be the set of edges that were assigned a direction in Steps (2b) and (2c). Set $D = D \cup C_E$.
3. **Return** schedule corresponding to the orientation of the edges.

The time complexity of this algorithm is $O(n^2m + m^2n)$. Let $C_1, C_2, \ldots, C_h$ be the bag components selected by the algorithm in Step (2a) in the order they were chosen.

Lemma 1. *If the expansion of C_h is attempted by the algorithm and it returns `FAIL`, then there is no schedule with makespan at most d. Also, if the algorithm produces a schedule, the makespan of the schedule is at most $(b/2)d$.*

Theorem 3. *There is a $b/2$-approximation algorithm for the graph balancing problem with $b \geq 2$ bags.*

8 A Polynomial-Time Algorithm for $Q|bag, p_j = 1, \mathcal{M}_j|C_{max}$

In this section we consider the restricted assignment problem on uniform parallel machines with bags where every job has the same length. Without loss of generality we can assume that all the machines have speeds $s_1, \ldots, s_m \in \mathbb{Z}^+$. Let the least common multiple of the speeds $s_1, \ldots, s_m$ be c. The above problem is equivalent to when every job $j \in J$ has length $p_j = c$, so for convenience we assume below that every job has length $p_j = c$. Observe that since c is the least common multiple of the speeds, p_j/s_i is integral for all $j \in J$ and $i \in \{1, \ldots, m\}$. We employ binary search with upper bound $U = nc$ and a 1-relaxed decision procedure to compute the smallest value $d \in [1, nc]$ for which there is a schedule with makespan at most d. Note that in the special case when there is one bag for each job, our algorithm is exactly the algorithm of Lin and Li [17] for $Q|p_j = 1, \mathcal{M}_j|C_{max}$.

Let a *conflict machine set for bag* B_k be $\mathcal{C}(B_k) = \{M_i \in M \mid \exists j, j' \in B_k : M_i \in \mathcal{M}_j \cap \mathcal{M}_{j'}\}$, where $\mathcal{M}_j$ and $\mathcal{M}_{j'}$ are the sets of machines where for jobs j and j' can be scheduled, respectively. As a result of this definition, if there is a machine $M_i \notin \mathcal{C}(B_k)$, then at most one job in B_k can be scheduled on machine M_i. In our algorithm we first build a flow network N with a source s and sink t as follows. First, there will be a *job node* for each job $j \in J$; then for each $k = 1, \ldots, b$, create a *conflict machine node* for every machine $M'_i \in \mathcal{C}(B_k)$, and a *machine node* for each machine $M_i \in M$. To avoid ambiguity, we write M'_i whenever we refer to a conflict machine node of a machine M_i, and M_i when we refer to the machine node for machine M_i. We add arcs as follows: add arcs with capacity 1 from the source to each job node; if job $j \in B_k$ can be scheduled on machine M_i: (i) if $M_i \in \mathcal{C}(B_k)$ then add an arc from j to the machine conflict node M'_i of bag B_k with capacity 1, (ii) otherwise add an arc from j to machine node M_i with capacity 1. Add an arc with flow capacity 1 from each machine conflict node M'_i to its corresponding machine node M_i and include an arc from each machine node M_i to the sink with capacity $\lfloor (s_i d)/c \rfloor$.

Lemma 2. *There is an integral flow f that saturates all the arcs incident on s if and only if there is a feasible schedule with makespan at most d.*

The 1-relaxed decision algorithm is the following: build the flow network N described above and compute an integral maximum flow f; if f does not saturate at least one arc incident on the source, return `FAIL`; otherwise all the arcs incident on the source are saturated, and for each job $j \in J$, schedule job j on machine i if there is flow sent from job node j to machine node M_i.

Theorem 4. *There is a polynomial-time algorithm for $Q|bag, p_j = 1, \mathcal{M}_j|C_{max}$.*

9 Inapproximability and Complexity

9.1 Restricted Assignment Problem with $b = 2$ Bags

To begin, we prove that restricted assignment problem with $b = 2$ bags where the job lengths are either 1 or 2 has no approximation algorithm with approximation ratio less than 3/2, unless $\mathsf{P} = \mathsf{NP}$ (Corollary 2). To do this we reduce from the 3-dimensional matching problem ([SP1] in [9]). In the 3-dimensional matching problem there are three disjoint sets $X = \{x_1, x_2, \ldots, x_{m'}\}, Y = \{y_1, \ldots, y_{m'}\}, Z = \{z_1, \ldots, z_{m'}\}$, and a set $T \subseteq X \times Y \times Z$ of triples, and the goal is to determine whether there is a set $T' \subseteq T$ containing m' triples, such that for any pair of triples $(x_k, y_k, z_k), (x_\ell, y_\ell, z_\ell) \in T'$, $x_k \neq x_\ell$, $y_k \neq y_\ell$, and $z_k \neq z_\ell$. We note that our reduction is similar to the one given by Lenstra *et al.* [16], but their reduction assumes there are $b = n$ bags.

Let us describe the reduction. For each triple $t \in T$ where element $z \in t$ and $z \in Z$, create a machine M_t of *type* z. Next, each element in $X \cup Y$ is a job j, where we place j in bag B_1 if $j \in X$ and in bag B_2 if $j \in Y$; j has $p_{t,j} = 1$ on machine M_t if $j \in t$, and $p_{t,j} = \infty$ otherwise. Let $deg(z)$ be the number of triples of T containing element $z \in Z$. Then for each element $z \in Z$, create $(deg(z) - 1)$ *dummy* jobs of *type* z, where each dummy job j takes 2 time units on machines of type z, otherwise $p_{t,j} = \infty$. Put all the dummy jobs in bag B_1.

Theorem 5. *It is* NP*-hard to decide whether there is a schedule with makespan at most* 2 *for the restricted assignment problem with* $b = 2$ *bags when the jobs have lengths either* 1 *or* 2*.*

Corollary 2. *There is no* p*-approximation algorithm with* $p < 3/2$ *for the restricted assignment problem with* $b = 2$ *bags where the job lengths are either* 1 *or* 2*, unless* $\mathsf{P} = \mathsf{NP}$*.*

9.2 Graph Balancing Problem with $b = 3$ Bags

In this section we show that when there are $b \geq 3$ bags, it is NP-hard to approximate the graph balancing problem with b bags with approximation ratio less than 3/2. To do this we extend a reduction of Ebenlendr *et al.* [6], and reduce from a variant of 3-SAT called At-Most-3-SAT($2L$), which is known to be NP-complete [1]. More precisely, let there be a propositional logic formula ϕ in conjunctive normal form (CNF), where there are n' boolean variables $x_1, \ldots, x_{n'}$ and m' clauses $y_1, y_2, \ldots, y_{m'}$. There are at most three literals per clause, and each literal (a variable or its negation) occurs at most twice in ϕ. This problem asks if there is an assignment of values to the variables so that ϕ is satisfied.

Let us first briefly describe the original reduction. Create one vertex for each clause y_i, and two vertices, one for each literal of variable x_i, x_i and $\neg x_i$; let the former be called *clause vertices* and the latter be called *literal vertices*. For each variable x_i, add an edge $\{x_i, \neg x_i\}$ with length 2 called a *tautologous edge*; add a self-loop on clause vertex y_i with length $3 - |y_i|$ if $3 - |y_i| > 0$, where $|y_i|$ is

the number of literals in clause y_i. Finally, for each clause y_i and literal l_j, add a *clause edge* $\{l_i, y_i\}$ if literal l_j is in clause y_i.

Now we describe our extension to this reduction that will assign each edge to a bag. Create a modified version of G called G', where, for each self loop incident on a clause vertex y_i in G, replace the self-loop with a new vertex y'_i and *self edge* $\{y_i, y'_i\}$ in G'; each self-edge in G' corresponds to a self-loop in G.

Lemma 3. *There is an edge colouring of G' that uses at most four colours η_1, η_2, η_3, η_4, this colouring can be computed in polynomial time.*

Proof. Assign colour η_4 to all tautologous edges, then consider the subgraph G'' of G' consisting of the same vertices but only the uncoloured edges. Observe that every edge in G'' either has a literal vertex and a clause vertex as its endpoints or is a self-edge with one endpoint that is a leaf, thus G'' is bipartite. Since G'' is bipartite and the maximum degree of any vertex in G'' is three, there is an edge colouring of G'' using three colours η_1, η_2, η_3, this edge colouring can be computed in polynomial time. □

Using Lemma 3 we can assign the edges in G to three bags: the edges coloured in G' using colours η_1, η_2, and η_3 are placed in bags B_1, B_2, and B_3, respectively. Finally, place the edges coloured with η_4 in any of the three bags.

Theorem 6. *There is no p-approximation algorithm with $p < 3/2$ for the graph balancing problem with $b \geq 3$ bags where job lengths are either 1 or 2, unless* $\mathsf{P} = \mathsf{NP}$.

9.3 $Q|bag|C_{max}$ with $b = 2$ Bags

For $P|bag|C_{max}$ with $b = 3$ bags and $Q|bag|C_{max}$ with $b = 2$ bags, we show that both are strongly NP-hard. We reduce from numerical 3-dimensional matching ([SP16] in [9]), which is known to be NP-complete in the strong sense. In the numerical 3-dimensional matching problem $3m'$ elements are contained in 3 disjoint sets $X = \{a_1, a_2, \ldots, a_{m'}\}, Y = \{a_{m'+1}, \ldots, a_{2m'}\}, Z = \{a_{2m'+1}, \ldots, a_{3m'}\}$, and every element $a_j \in X \cup Y \cup Z$ has a size $s(a_j) \in \mathbb{Z}^+$. Given a value $\beta \in \mathbb{Z}^+$ the goal is to determine whether there are disjoint triples $A_1, \ldots, A_{m'}$ where each triple A_i contains exactly one element of X, one element of Y, and one element of Z, such that $\sum_{a_j \in A_i} s(a_j) = \beta$.

Notice that if an instance of $P|bag|C_{max}$ with $b = 3$ bags has exactly $3m$ jobs, Property 1 implies that every machine in a feasible schedule processes 3 jobs. We obtain a straightforward reduction from numerical 3-dimensional matching to $P|bag|C_{max}$ with $b = 3$ bags: set $m = m'$, $n = 3m'$, every element $a_j \in X \cup Y \cup Z$ is a job $j \in J$ with length $p_j = s(a_j)$, and $B_1 = X$, $B_2 = Y$, and $B_3 = Z$. This reduction was independently presented by Dokka *et al.* [5].

Theorem 7 (Dokka *et al.* [5]). *$P|bag|C_{max}$ with $b = 3$ bags is strongly* NP*-hard.*

When the machines are uniform we can eliminate the third bag necessary in the above reduction. Instead $n = 2m'$, and associate each machine M_i with a unique element $z_i \in Z$ and set the speed of M_i to $s_i = (\beta - s(z_i))/\beta$.

Theorem 8. *$Q|bag|C_{max}$ with $b = 2$ bags is strongly* NP*-hard.*

References

1. Y. Asahiro, J. Jansson, E. Miyano, H. Ono, and K. Zenmyo. Approximation algorithms for the graph orientation minimizing the maximum weighted outdegree. *J. of Combinatorial Optimization*, 22(1):78–96, 2011.
2. H. Bodlaender, K. Jansen, and G. Woeginger. Scheduling with incompatible jobs. *Discrete Applied Mathematics*, 55(3):219–232, 1994.
3. D. Chakrabarty, S. Khanna, and S. Li. On (1, ε)-restricted assignment makespan minimization. In 26^{th} *Ann. ACM-SIAM Symp. on Discrete Algorithms*, 1087–1101 (2015).
4. S. Das and A. Wiese. On minimizing the makespan when some jobs cannot be assigned on the same machine. In 24^{th} *Ann. European Symp. on Algorithms*. LIPIcs, vol. 87. 31:1–31:14 (2017).
5. T. Dokka, A. Kouvela, and F. Spieksma. Approximating the multi-level bottleneck assignment problem. *Operations Research Letters*, 40:282–286, 2012.
6. T. Ebenlendr, M. Krčál, and J. Sgall. Graph balancing: A special case of scheduling unrelated parallel machines. In 19^{th} *Ann. ACM–SIAM Symp. on Discrete Algorithms*, 483–490 (2008).
7. F. Eisenbrand, K. Kesavan, R. Mattikalli, M. Niemeier, A. Nordsieck, M. Skutella, J. Verschae, and A. Wiese. Solving an avionics real-time scheduling problem by advanced IP-methods. In *European Symp. on Algorithms*, 11–22 (2010).
8. 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.
9. M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. 1979.
10. J. Gehrke, K. Jansen, S. Kraft, and J. Schikowski. A PTAS for scheduling unrelated machines of few different types. *Int. J. of Foundations of Computer Science*, 2018.
11. 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.
12. C. Huang and S. Ott. A combinatorial approximation algorithm for graph balancing with light hyper edges. In 24^{th} *Ann. European Symp. on Algorithms*. LIPIcs, vol. 57, 49:1–49:15 (2016).
13. K. Jansen and M. Maack. An EPTAS for scheduling on unrelated machines of few different types. In *Workshop on Algorithms and Data Structures*, 497–508. Springer, 2017.
14. K. Jansen and L. Rohwedder. A quasi-polynomial approximation for the restricted assignment problem. In *Int. Conference on Integer Programming and Combinatorial Optimization*, 305–316. Springer, 2017.
15. S. Kolliopoulos and Y. Moysoglou The 2-valued case of makespan minimization with assignment constraints. *Information Processing Letters*, 113(1):39–43, 2013.

16. J. Lenstra, D. Shmoys, and E. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, 46(1–3):259–271, 1990.
17. Y. Lin and W. Li. Parallel machine scheduling of machine-dependent jobs with unit-length. *European Journal of Operational Research*, 156(1):261–266, 2004.
18. D. Page and R. Solis-Oba. A 3/2-approximation algorithm for the graph balancing problem with two weights. *Algorithms*, 9(2):38, 2016.
19. E. Shchepin and N. Vakhania. An optimal rounding gives a better approximation for scheduling unrelated machines. *Operations Research Letters*, 33:127–133, 2005.