# Makespan Minimization on Unrelated Parallel Machines with Simple Job-Intersection Structure and Bounded Job Assignments*

Daniel R. Page[1][0000−0002−6317−7431] ✉, Roberto Solis-Oba[1], and Marten Maack[2]

[1] Department of Computer Science, Western University, London, Canada
dpage6@uwo.ca, solis@csd.uwo.ca
[2] Department of Computer Science, University of Kiel, Kiel, Germany
mmaa@informatik.uni-kiel.de

**Abstract.** Let there be a set $J$ of $n$ jobs and a set $M$ of $m$ parallel machines, where each job $j$ takes $p_{i,j} \in \mathbb{Z}^+$ time units on machine $i$ and assume $p_{i,j} = \infty$ implies job $j$ cannot be scheduled on machine $i$. In makespan minimization on unrelated parallel machines ($R||C_{max}$), the goal is to schedule each job non-preemptively on a machine so as to minimize the makespan. A job-intersection graph $G_J = (J, E_J)$ is an unweighted undirected graph where there is an edge $\{j, j'\} \in E_J$ if there is a machine $i$ such that both $p_{i,j} \neq \infty$ and $p_{i,j'} \neq \infty$. In this paper we consider two variants of $R||C_{max}$ where there are a small number of eligible jobs per machine. First, we prove that there is no approximation algorithm with approximation ratio better than 3/2 for $R||C_{max}$ when restricted to instances where the job-intersection graph contains no diamonds, unless $\mathsf{P} = \mathsf{NP}$. Second, we match this lower bound by presenting a 3/2-approximation algorithm for this special case of $R||C_{max}$, and furthermore show that when $G_J$ is triangle free $R||C_{max}$ is solvable in polynomial time. For $R||C_{max}$ restricted to instances when every machine can process at most $\ell$ jobs, we give approximation algorithms with approximation ratios 3/2 and 5/3 for $\ell = 3$ and $\ell = 4$ respectively, a polynomial-time algorithm when $\ell = 2$, and prove that it is $\mathsf{NP}$-hard to approximate the optimum solution within a factor less than 3/2 when $\ell \geq 3$. In the special case where every $p_{i,j} \in \{p_j, \infty\}$, called the restricted assignment problem, and there are only two job lengths $p_j \in \{\alpha, \beta\}$ we present a $(2 - 1/(\ell - 1))$-approximation algorithm when $\ell \geq 3$.

**Keywords:** makespan minimization · unrelated parallel machines · approximation algorithms · restricted assignment · bounded job assignments · job-intersection graphs.

This is a version of the paper that was accepted at the 12th International Conference on Combinatorial Optimization and Applications (COCOA 2018), Atlanta, USA, December 15–17, 2018. The final authenticated version is available online at https://doi.org/10.1007/978-3-030-04651-4_23.

* 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. Marten Maack was supported by the DAAD (Deutscher Akademischer Austauschdienst).

## 1 Introduction

Let $J$ be a set of $n$ jobs and $M$ a set of $m$ parallel machines, where a job $j$ takes $p_{i,j} \in \mathbb{Z}^+$ time units on machine $i$. The goal in *makespan minimization on unrelated parallel machines* is to produce a schedule where each job is scheduled non-preemptively on a machine so as to minimize the length of the schedule or *makespan*. Makespan minimization on unrelated parallel machines is a classic NP-hard scheduling problem, and is denoted as $R||C_{max}$ in Graham's notation (see [12]). Note that when the processing time $p_{i,j} = \infty$, we say that job $j$ cannot be scheduled on machine $i$, and assume the processing times are given as an $m \times n$ *processing matrix* $P = (p_{i,j})$. In this paper we investigate two versions of $R||C_{max}$:

- $R||C_{max}$ *with simple job-intersection structure.* A *job-intersection graph* $G_J = (J, E_J)$ has a *job vertex* for each job $j \in J$, and for any two jobs $j, j' \in J$, there is an edge $\{j, j'\} \in E_J$ if there is a machine $i$ such that $p_{i,j} \neq \infty$ and $p_{i,j'} \neq \infty$. A set of restrictions on which machines can process a job can be represented as a job-intersection graph. We study $R||C_{max}$ restricted to particular classes of job-intersection graphs. We give an example of a job-intersection graph in Figure 1.

**Instance:** $m = 3$, $n = 4$

$$P = \begin{array}{c} \begin{matrix} j_1 & j_2 & j_3 & j_4 \end{matrix} \\ \begin{pmatrix} 1 & 2 & 1 & \infty \\ 3 & 5 & 1 & \infty \\ \infty & \infty & 1 & 4 \end{pmatrix} \end{array} \begin{matrix} M_1 \\ M_2 \\ M_3 \end{matrix}$$

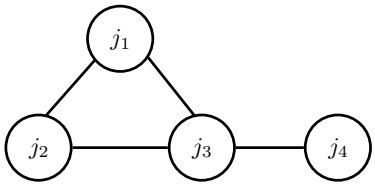


**Fig. 1.** An instance of $R||C_{max}$ (left), and its job-intersection graph $G_J$ (right).

- $R||C_{max}$ *with bounded job assignments.* Let $\mathcal{J}_i$ be the set of jobs that can be processed by machine $i$, i.e., $\mathcal{J}_i = \{j \in J \mid p_{i,j} \neq \infty\}$. Let $\ell > 0$. We consider $R||C_{max}$ restricted to instances when, for each machine $i$, $|\mathcal{J}_i| \leq \ell$. Clearly when $\ell = n$, it is $R||C_{max}$.

Currently the best-known approximation algorithms for $R||C_{max}$ have approximation ratio 2 [10,18,24], and there is no approximation algorithm for $R||C_{max}$ with approximation ratio less than 3/2, unless P = NP [18]. Despite much intensive study, finding an approximation algorithm with approximation ratio strictly less than 2 still remains an open problem and is regarded as one of the most challenging open problems in the study of approximation algorithms today [25]. A natural question is whether there are any "well-structured" and efficient to recognize classes of job-intersection graphs, for which the corresponding instances of $R||C_{max}$ can be efficiently solved or for which there are approximation algorithms with approximation ratio less than 2. As we show, both problems given above are closely related from a hardness of approximation standpoint

and we present algorithms for both. Furthermore, we establish that there is no approximation algorithm with approximation ratio less than 3/2 for $R||C_{max}$ restricted to diamondless job-intersection graphs or for $R||C_{max}$ when every machine can process at most $\ell = 3$ jobs, unless $\mathsf{P} = \mathsf{NP}$. However, in both of these cases we can formulate a relatively simple combinatorial algorithm that has the approximation ratio 3/2, matching the lower bound.

## 2 Preliminaries

One $\mathsf{NP}$-hard special case of $R||C_{max}$ of recent interest in the literature is the graph balancing problem. In the *graph balancing problem*, every job takes $p_{i,j} = p_j$ time units and can only be scheduled on one of at most two possible machines. This problem can be described as an edge orientation problem: given a weighted multigraph $G = (V, E)$ with weights $p_e$ for each edge $e \in E$, orient all the edges in $G$ such that the maximum load of the vertices is minimized, where the load of a vertex is the sum of all the weights of edges oriented toward that vertex. In this formulation the edges are the jobs, and the vertices are the machines. We would like to remark that another well-known and intensely studied special case of $R||C_{max}$ is the *restricted assignment problem*, which is a general case of the graph balancing problem where every job has a subset of machines on which it can be scheduled.

A graph is *triangle free* if it does not contain any simple cycles of length 3—triangles. Note that all bipartite graphs contain no odd-length cycles, thus all bipartite graphs are triangle free. The *diamond graph* consists of four vertices and five edges, so it is $K_4$ less one edge. We call a graph *diamondless* if it does not contain the diamond graph as a subgraph. In contrast, a *diamond-free graph* is defined as not having the diamond graph as an induced subgraph. An *induced subgraph* $H = (V', E')$ of a graph $G = (V, E)$ is such that $V' \subseteq V$ and an edge $e = \{u, v\} \in E'$ if both $u, v \in V'$ and $e \in E$; all diamondless graphs are diamond-free, but not all diamond-free graphs are diamondless. For example, the graph $K_4$ is diamond free but is not diamondless. In Figure 2 we give an instance of the graph balancing problem where its job-intersection graph is both diamondless and diamond free.

## 3 Related Work

The best-known approximation algorithms for $R||C_{max}$ and the restricted assignment problem have approximation ratio 2 [10,18,24]. However, for the restricted assignment problem with two job lengths, $\alpha < \beta$, Chakrabarty *et al.* [5] gave a $(2 - \delta)$-approximation algorithm for some small value $\delta > 0$ and a $(2 - \alpha/\beta)$-approximation algorithm. Ebenlendr *et al.* [8] presented a 7/4-approximation algorithm for the graph balancing problem, and in [13,22] 3/2-approximation algorithms are presented for the problem when there are only two job lengths.

The concept of the job-intersection graph goes back to at least Glass and Kellerer [11] with the study of so-called nested-structures and the restricted

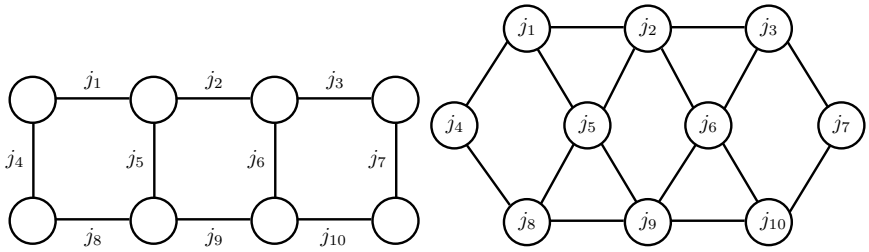


**Fig. 2.** An instance of the graph balancing problem (left) and its job-intersection graph (right).

assignment problem. Research on the restricted assignment problem when instances satisfy certain structural properties is extensive and has grown in interest in recent years [19]. In addition, there has been investigation of scheduling problems on machine-intersection graphs where the machines are the vertices and an edge exists between two vertices when a job can be scheduled on the two corresponding machines [3,14,17]. Jansen *et al.* [14][3] proved that $R||C_{max}$ is fixed-parameter tractable (FPT) in the treewidth $tw$ of the job-intersection graph. That is, if the job-intersection graph $G_J$ has constant treewidth, $R||C_{max}$ can be solved in polynomial time. So when the job-intersection graph belongs to graph classes such as trees ($tw = 1$), cactus graphs ($tw \leq 2$), outerplanar graphs ($tw \leq 2$), and series-parallel graphs ($tw \leq 2$), $R||C_{max}$ is solvable in polynomial time. In this paper we study $R||C_{max}$ restricted to classes of job-intersection graphs that do not have constant treewidth. In Figure 3 we summarize both computational complexity results found in this paper and presently in the literature for $R||C_{max}$ on job-intersection graphs.

To the best of our knowledge $R||C_{max}$ with bounded job assignments has not been previously studied. Bounded job assignments have been considered in other types of scheduling problems, such as in batch scheduling where a batch size bounds the number of jobs simultaneously processed by a batching machine [6,20]. A generalization of $R||C_{max}$ where every machine has a positive integer called a *machine capacity* that bounds the maximum number of jobs each machine can process has also been studied. For this generalization there is a 2-approximation algorithm [23], and there exists an efficient polynomial-time approximation scheme when the machines are identical [7].

It is important to discuss recognition of the instances for which we design algorithms. For any $0 \leq \ell \leq n$, it is trivial to determine if the set $\mathcal{J}_i$ of jobs that every machine $i$ can process has size at most $\ell$. Alon *et al.* [1] gave an algorithm that can test if a graph $(V, E)$ is triangle free in $O(|E|^{1.41})$ time. We note that diamondless graphs can be recognized in $O(|V|^3)$ time by a simple algorithm that looks for a pair of triangles with a common edge. Kloks *et al.* [16] showed that one can recognize if a graph is diamond-free (and give a diamond in the

[3] In this paper the authors refer to the job-intersection graph as the primal graph.

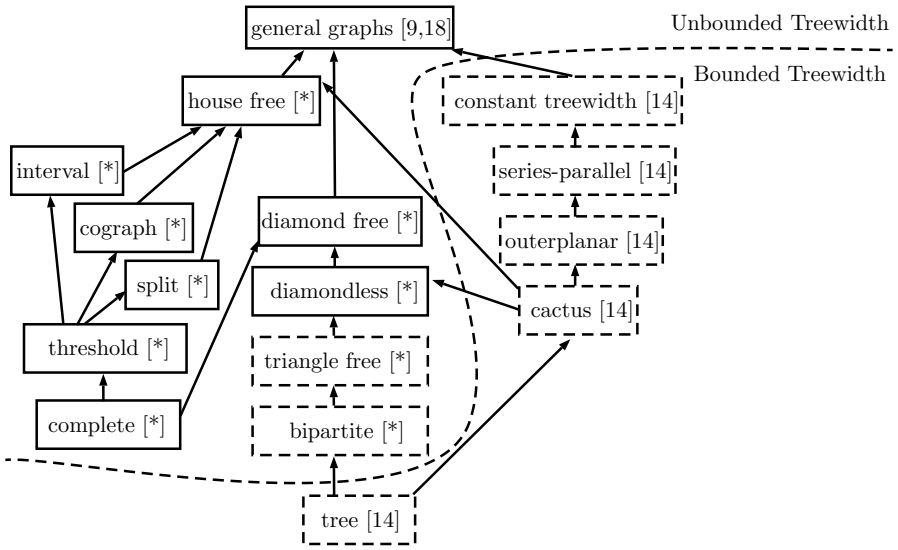


**Fig. 3.** Summary of results for $R||C_{max}$ with simple job-intersection structure. The job-intersection graphs restricting the machine assignments are grouped by graph class. For two graph classes $A$ and $B$, "$A \rightarrow B$" in the diagram means that any graph in $A$ is in graph class $B$. Problems boxed with dashed lines are polynomial-time solvable, and problems with boxed solid lines are strongly NP-hard. The number(s) in brackets are reference numbers, and graph classes with [*] beside them refer to computational complexity results found in this paper.

graph if it is not) in $O(|V|^c + |E|^{3/2})$ time, where $O(|V|^c)$ is the time complexity to compute the square of a $|V| \times |V|$ 0-1 adjacency matrix.

## 4 Our Results

First, we establish that once one admits triangles but forbids diamonds in the job-intersection graph, there is no $k$-approximation algorithm for $R||C_{max}$ with $k < 3/2$, unless $\mathsf{P} \neq \mathsf{NP}$. This matches the same inapproximability bounds as those that exist for the restricted assignment problem with two job lengths [18] and the graph balancing problem with two job lengths [2,9], both special cases of $R||C_{max}$. To do this, we strengthen the inapproximability result of Ebenlendr *et al.* [9] for the graph balancing problem with two job lengths. Employing this result we can also prove that for $R||C_{max}$ when every machine $i$ satisfies $|\mathcal{J}_i| \leq 3$, the inapproximability lower bound of 3/2 holds. In Section 8, we show that $R||C_{max}$ restricted to job-intersection graphs belonging to several well-studied graph classes such as complete graphs, threshold graphs, interval graphs, cographs, split graphs, and house-free graphs do not have any $k$-approximation algorithm with $k < 3/2$, unless $\mathsf{P} = \mathsf{NP}$.

To complement our inapproximability results, we present a flow-based 3/2-approximation algorithm for $R||C_{max}$ when every machine can process at most three jobs. As we will later justify, this problem contains as special cases $R||C_{max}$ when $G_J$ is triangle free and $R||C_{max}$ when $G_J$ is diamondless. Our algorithm can also be used to exactly solve in polynomial time $R||C_{max}$ when every machine can process at most two jobs, as well as $R||C_{max}$ when restricted to triangle-free job-intersection graphs. In addition, the same algorithm is a 5/3-approximation algorithm for $R||C_{max}$ when every machine can process at most four jobs. Finally, in Section 7 we give a $(2-1/(\ell-1))$-approximation algorithm for the restricted assignment problem with two job lengths when every machine can process at most $\ell \geq 3$ jobs.

## 5 Hardness of Approximation on Diamondless Graphs

In this section we prove under the assumption that $\mathsf{P} \neq \mathsf{NP}$ that $R||C_{max}$ has the inapproximability bound 3/2 even in the case when instances have job-intersection graphs that are diamondless. To do this, we employ a similar reduction as that used by Ebenlendr *et al.* [9]. Since Ebenlendr *et al.* showed this reduction yields the inapproximability bound we desire, we must show that the job-intersection graphs from graph-balancing instances produced by the reduction are diamondless.

The reduction by Ebenlendr *et al.* [9] uses a variant of the satisfiability problem we will denote as At-most-3-SAT(2L). Let there be $n'$ boolean variables $x_1, \ldots, x_{n'}$, and $m'$ clauses $\alpha_1, \ldots, \alpha_{m'}$. Given a propositional logic formula $\phi$ in conjunctive normal form (CNF) where there are at most three literals per clause, each variable appears at most three times in $\phi$, and each literal (a variable or its negation) appears at most twice in $\phi$, the problem is to decide whether there is an assignment of values to the variables $x_1, \ldots, x_{n'}$ so that $\phi$ is satisfied. At-most-3-SAT(2L) is known to be $\mathsf{NP}$-complete [2]. Without loss of generality we assume that no clause contains a tautology, and that no clause contains duplicate literals.

Now we describe how to construct the graph balancing instance $I'$ from At-most-3-SAT(2L) instance $I = (\phi, n', m')$. Introduce two types of vertices: literal vertices, and clause vertices. Given a variable $x_i$, a *literal vertex* corresponds to a literal $x_i$ or $\neg x_i$. For each clause $\alpha_j$, a *clause vertex* is created that corresponds to clause $\alpha_j$. There will be two types of edges: tautologous edges, and clause edges. For each variable $x_i$, a *tautologous edge* $\{x_i, \neg x_i\}$ has weight 2. For each clause $\alpha_j$ and literal $\lambda$ that appears in $\alpha_j$, a *clause edge* $\{\lambda, \alpha_j\}$ has weight 1. Finally, for clause $\alpha_j$ add $3 - |\alpha_j|$ self-loops with weight 1 on its clause vertex, where $|\alpha_j|$ is the number of literals in clause $\alpha_j$. The idea is that the orientation of the tautologous edges will determine the assignment of values to the variables of $\phi$. Instance $I'$ can be built from $I$ in polynomial time.

To illustrate the reduction, we give an example. Let the propositional logic formula $\phi = (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3)$, where $n' = 3$ and $m' = 2$. Then $\alpha_1 = (x_1 \vee \neg x_2)$ and $\alpha_2 = (\neg x_1 \vee \neg x_2 \vee \neg x_3)$. Applying the reduction we obtain

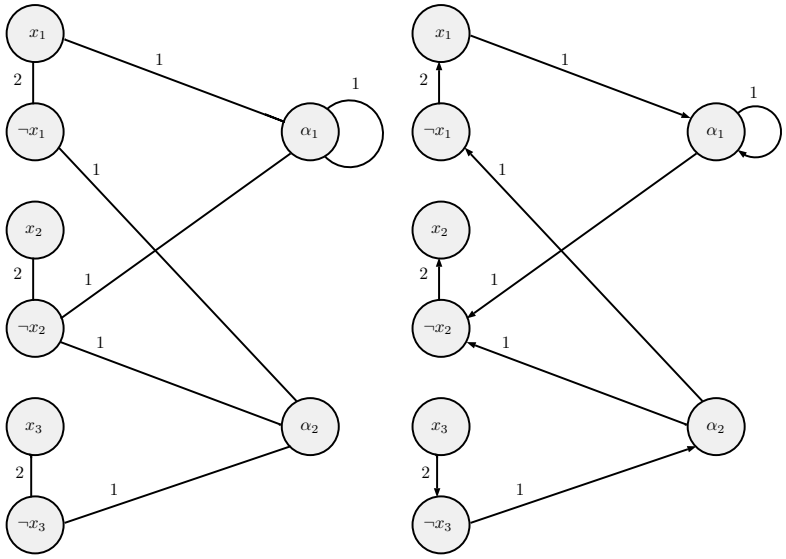


**Fig. 4.** Given the formula $\phi = (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3)$, the resulting graph balancing instance applying the construction is shown on the left. Its optimal orientation is given on the right.

the graph-balancing instance shown in Figure 4. The formula $\phi$ can be satisfied, and the resulting instance has an optimal orientation with makespan 2.

Ebenlendr *et al.* [9] proved that $I'$ has a schedule with makespan at most two if $\phi$ is satisfied, but the makespan is at least three otherwise. Hence, if there were a $k$-approximation algorithm with $k < 3/2$, one could apply the above reduction, apply said $k$-approximation algorithm, then correctly decide whether $\phi$ is satisfiable or not in polynomial time: if the makespan is less than three return "yes"; and if the makespan is at least three, return "no".

**Lemma 1.** *The job-intersection graph $G_J$ of the weighted multigraph $G$ produced by the above reduction contains no diamonds.*

*Proof.* Assume that $G_J$ has at least one diamond. Observe that every vertex in $G$ has at most three incident edges, so $G_J$ can only be comprised of isolated job vertices, paths, or triangles. Then, there must be two triangles that share two job vertices to form a diamond.

First, consider the edges incident on literal vertices in $G$. Recall that each variable appears in at most three clauses in formula $\phi$ and each literal for that variable appears at most twice. So the job vertex corresponding to the tautologous edge $\{x_i, \neg x_i\}$ has degree at most three in $G_J$. Furthermore, this job vertex is only adjacent to job vertices that are clause edges in $G$, and at most two clause edges may have the same literal vertex as an endpoint in $G$. Thus, any job vertex $\{x_i, \neg x_i\}$ along with its adjacent job vertices for clause edges form in $G_J$

either a path with one edge, a path with two edges, a triangle, or a path with an edge plus a triangle, but not two triangles i.e. a bowtie or a diamond.

Next consider the edges incident on clause vertices in $G$. As no two clause vertices have edges in common in $G$ and every clause vertex has three edges incident on it, the edges incident on the clause vertex form a triangle in $G_J$. Thus, the diamond must be comprised of job vertices of two clause edges that are adjacent to the job vertices of a tautologous edge and another clause edge. There is no diamond when these two clause edges are incident on two literal vertices of different variables, hence, there are two possibilities:

1. A diamond formed by two clause edges that are incident on the same literal vertex in $G$. This cannot happen as no clause in formula $\phi$ has duplicate literals.
2. A diamond formed by two clause edges that are incident on two literal vertices for the same variable in $G$. No clause contains tautologies, thus this situation cannot occur.

Therefore, by contradiction, $G_J$ contains no diamonds. □

**Theorem 1.** *There is no $k$-approximation algorithm with $k < 3/2$ for the graph balancing problem with two job lengths when the job-intersection graph $G_J$ contains no diamonds, unless* $\mathsf{P} = \mathsf{NP}$.

**Corollary 1.** *There is no $k$-approximation algorithm with $k < 3/2$ for $R||C_{max}$ restricted to diamondless job-intersection graphs, unless* $\mathsf{P} = \mathsf{NP}$.

If $|\mathcal{J}_i| > 3$ for some machine $i$, then there are at least four jobs $j_1$, $j_2$, $j_3$, $j_4$ such that $p_{i,j_1} \neq \infty$, $p_{i,j_2} \neq \infty$, $p_{i,j_3} \neq \infty$, and $p_{i,j_4} \neq \infty$. This would imply $G_J$ contains a diamond; thus, for any machine $i$, $|\mathcal{J}_i| \leq 3$ is satisfied if $G_J$ is diamondless. Hence, the diamondless case is a special case of when, for each machine $i$, $|\mathcal{J}_i| \leq 3$. Thus our inapproximability results carry over to the special case where every machine can process at most three jobs. Do note that proving this special case has the inapproximability bound stated in the corollary can also be made trivially by simply observing that every vertex has at most three incident edges in the graph-balancing instance constructed in the above reduction.

**Corollary 2.** *There is no $k$-approximation algorithm for the graph balancing problem with two job lengths when every machine can process at most three jobs where $k < 3/2$, unless* $\mathsf{P} = \mathsf{NP}$.

## 6 Approximation Results for Unrelated Scheduling with Bounded Job Assignments

As we stated at the end of the previous section, $R||C_{max}$ restricted to diamondless job-intersection graphs is a special case of $R||C_{max}$ when every machine $i$ satisfies $|\mathcal{J}_i| \leq 3$. We present a 5/3-approximation algorithm for $R||C_{max}$ when

every machine can process at most four jobs. In our analysis we show the same approximation algorithm has approximation ratio 3/2 in the case when every machine can process at most three jobs. Note that for $R||C_{max}$ restricted to triangle-free job-intersection graphs, no machine can process three jobs as doing so implies three jobs share a common machine where they can be scheduled, so every machine $i$ satisfies $|\mathcal{J}_i| \leq 2$ in this particular situation.

Let $OPT$ be the value of an optimal solution for $R||C_{max}$ when every machine $i$ satisfies $|\mathcal{J}_i| \leq 4$. Similar to [18], we perform a binary search procedure to find the smallest value $T$ over the interval $[0, \sum_{i \in M, j \in J}(p_{i,j})]$ such that the algorithm given below produces a schedule with makespan at most $(5/3)T$. If a schedule is produced then we decrease the value of $T$ in the binary search, and if `REJECT` is reported, then there is no schedule with makespan at most $T$ and thus $T$ is increased in the binary search. At the end of the binary search the smallest value for $T$ is found, so it must be the case that $T \leq OPT$ and so the approximation ratio is 5/3. We say a job $j$ is *small* on machine $i$ if its processing time is $p_{i,j} \leq T/2$, and is *big* on machine $i$ if $T/2 < p_{i,j} \leq T$. Observe that if $OPT \leq T$, at most one big job can be scheduled on a machine. Note that by our definitions, there can be a job $j$ that is neither big nor small with respect to its processing time on some machine $i$, if $p_{i,j} > T$.

1. First, there may be machines for which some jobs are neither big nor small. For each machine $i$, if any job $j$ has a processing time $p_{i,j} > T$ on machine $i$, we remove job $j$ from job set $\mathcal{J}_i$. As a result, every job $j$ in each job set $\mathcal{J}_i$ has processing time $p_{i,j} \leq T$.
2. Build a single-source single-sink flow network $N$ with source $s^*$ and sink $t^*$. In this network, create a job node for each job, and add arcs from $s^*$ to each job node with capacity 1. Now, for each machine $i$, we create a machine node and a buffer node with arcs according to the *Machine Plans* given in Figure 5, which we describe now. Let disjoint sets $\mathcal{S}_i, \mathcal{B}_i \subseteq \mathcal{J}_i$, where by default it is assumed that $\mathcal{S}_i$ and $\mathcal{B}_i$ are the small jobs and big jobs in $\mathcal{J}_i$, respectively. Consider the following cases in the order provided:
   (a) If $|\mathcal{J}_i| = 0$, no arcs are added for machine node $i$.
   (b) If $\sum_{j \in \mathcal{J}_i} p_{i,j} \leq T$, then every job $j \in \mathcal{J}_i$ can be scheduled on machine $i$, so we add arcs according to the Machine Plan with $d = |\mathcal{J}_i|$ and set $\mathcal{S}_i = \mathcal{J}_i$ and $\mathcal{B}_i = \varnothing$.
   (c) If $|\mathcal{J}_i| \leq 3$, then use the Machine Plan with $d = |\mathcal{J}_i| - 1$.

   In the last set of cases $|\mathcal{J}_i| = 4$. Sort the jobs of each $\mathcal{J}_i$ in non-increasing order by processing time; let these jobs be denoted as $j_1^{(i)}, j_2^{(i)}, j_3^{(i)}, j_4^{(i)}$.
   (d) If $\sum_{k=2}^{4} p_{i,j_k^{(i)}} > T$, add arcs according to the Machine Plan with $d = 2$.
   (e) If $\sum_{k=2}^{4} p_{i,j_k^{(i)}} \leq T$ and $p_{i,j_1^{(i)}} + p_{i,j_2^{(i)}} > T$, put $j_1^{(i)}$ and $j_2^{(i)}$ (if either is not already) into $\mathcal{B}_i$ and use the Machine Plan with $d = 3$.
   (f) If $\sum_{k=2}^{4} p_{i,j_k^{(i)}} \leq T$ and $p_{i,j_1^{(i)}} + p_{i,j_2^{(i)}} \leq T$, use the Machine Plan with $d = 3$.
3. Now that $N$ is constructed, the algorithm computes an integral maximum flow $f$ on $N$. If any arc leaving the source does not send one unit of flow

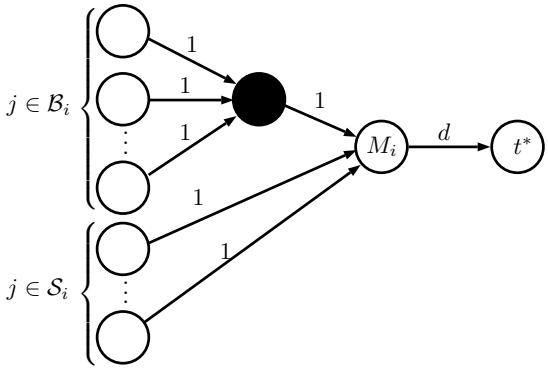


**Fig. 5.** Flow network $N$ is built in part by determining the appropriate machine plan for each machine. Assume an integer value $d$ is provided along with each plan, and unless otherwise stated, let $\mathcal{S}_i, \mathcal{B}_i \subseteq \mathcal{J}_i$ be the set of small jobs and set of big jobs, respectively. The machine plan for machine node $i$ shows the arcs and capacities of the arcs included. Unlabelled white nodes are job nodes, the black node is a *buffer* node of machine $i$ that only allows one unit of flow to be sent from job nodes in $\mathcal{B}_i$, and $t^*$ is the sink of $N$.

then there is a job node that receives no flow, report `REJECT` if this is the case. If all the job nodes receive one unit of flow, we build the schedule as follows: for each job node $j$, if machine node $i$ receives 1 unit of flow from $j$, schedule job $j$ on machine $i$.

By the way we designed the flow network, it is not hard to see that if $OPT \leq T$, all the arcs leaving the source are saturated, and as a result, a schedule is produced.

Now we analyze the load of each machine. First, it is trivial to observe that the load of any machine $i$ is at most $T$ if either $\sum_{j \in \mathcal{J}_i} p_{i,j} \leq T$ (case (b)) or all the jobs in $\mathcal{J}_i$ are big (case (c) if $|\mathcal{J}_i| \leq 3$, case (d) if $|\mathcal{J}_i| = 4$). Thus, we consider each machine $i$ when there is at least one small job and $\sum_{j \in \mathcal{J}} p_{i,j} > T$ based on the number of jobs in job set $\mathcal{J}_i$:

- $|\mathcal{J}_i| \leq 2$. If $|\mathcal{J}_i| \leq 1$, then either case (a) or case (b) occurs, which we already considered above. If $|\mathcal{J}_i| = 2$ and all the jobs are small, then $\sum_{j \in \mathcal{J}_i} p_{i,j} \leq T/2 + T/2 = T$ and falls under case (b). If $|\mathcal{J}_i| = 2$ and $\sum_{j \in \mathcal{J}_i} p_{i,j} > T$, then the only remaining case is when there is one big job and one small job that cannot be scheduled together. The algorithm applies case (c), which permits only $|\mathcal{J}_i| - 1 = 1$ job to be scheduled on machine $i$, and the load of machine $i$ is at most $T$. Therefore, the load of any machine with $|\mathcal{J}_i| \leq 2$ is at most $T$.
- $|\mathcal{J}_i| = 3$. Since $\sum_{j \in \mathcal{J}_i} p_{i,j} > T$, at most two jobs can be scheduled on machine $i$; case (c) is applied here, and the Machine Plan will allow at most $|\mathcal{J}_i| - 1 = 2$ jobs to be scheduled on machine $i$. If all three jobs in $\mathcal{J}_i$ are small, then at most two jobs are scheduled on machine $i$ and the load is at most $T$. Otherwise at least one job is big and at most two jobs are small in

$\mathcal{J}_i$, and at most one big job will be scheduled with a small job and so the load is at most $T + T/2 = (3/2)T$. Therefore, the load of any machine with $|\mathcal{J}_i| = 3$ is at most $(3/2)T$.

– $|\mathcal{J}_i| = 4$. First, we identify a few key observations that will simplify our analysis. First, if ever case (d) is applied, $d = 2$ in the Machine Plan, so at most one big job is scheduled with one small job and the load is at most $T + T/2 = (3/2)T$. Thus we only need to consider the algorithm in situations when it applies case (e) or case (f). In either of these two cases, $d = 3$, so at most one big job is scheduled with two small jobs, as when three small jobs are scheduled on machine $i$ the load is at most $(3/2)T$. Recall that the jobs in $\mathcal{J}_i$ are sorted in non-increasing order $j_1^{(i)}$, $j_2^{(i)}$, $j_3^{(i)}$, $j_4^{(i)}$. If there are at least three big jobs and at most one small job, then $\sum_{k=2}^{4} p_{i,j_k^{(i)}} > T$ and this falls under case (d); thus we only need to consider below when there are at most two big jobs and at least one small job in $\mathcal{J}_i$.

  - If all four jobs are small, then only case (f) applies as the sum of any two small jobs on machine $i$ cannot exceed $T$. Again, at most three small jobs can be scheduled on machine $i$ and the load is at most $(3/2)T$.
  - If three jobs are small and one job is big, then either case (e) or case (f) is applied by the algorithm. In case (e), if $p_{i,j_2^{(i)}} \leq T/3$ then the sorting of the jobs implies that the load is at most $T + 2(T/3) = (5/3)T$. Then, observe that if $p_{i,j_2^{(i)}} > T/3$ and $\sum_{k=2}^{4} p_{i,j_k^{(i)}} \leq T$, then $\sum_{k=3}^{4} p_{i,j_k^{(i)}} < T - T/3 = (2/3)T$, and the load on machine $i$ is at most $p_{i,j_1^{(i)}} + p_{i,j_3^{(i)}} + p_{i,j_4^{(i)}} \leq T + (2/3)T = (5/3)T$. Next if case (f) is applied, then $p_{i,j_1^{(i)}} + p_{i,j_2^{(i)}} \leq T$ implies the load of machine $i$ is at most $p_{i,j_1^{(i)}} + p_{i,j_2^{(i)}} + p_{i,j_3^{(i)}} \leq T + T/2 = (3/2)T$.
  - If two jobs are small and two jobs are big, only case (e) applies as the sum of any two big jobs exceeds $T$. Job $j_2^{(i)}$ is big, so observe that $\sum_{k=2}^{4} p_{i,j_k^{(i)}} \leq T \Rightarrow \sum_{k=3}^{4} p_{i,j_k^{(i)}} < T - (T/2) = T/2$. Thus, the load of machine $i$ is at most $T + T/2 = (3/2)T$.

  Hence, the maximum load of a machine with $|\mathcal{J}_i| = 4$ is at most $(5/3)T$.

Therefore, we obtain the following results that match the inapproximability bounds given by Corollary 1 and Corollary 2.

**Theorem 2.** *There is a polynomial-time algorithm for $R||C_{max}$ when every machine can process at most two, three, or four jobs with approximation ratio 1, 3/2, or 5/3, respectively.*

**Corollary 3.** *There is a polynomial-time algorithm for $R||C_{max}$ restricted to job-intersection graphs that are either triangle free or diamondless with approximation ratio 1 or 3/2, respectively. Furthermore, there is a polynomial-time algorithm for $R||C_{max}$ restricted to bipartite job-intersection graphs.*

## 7 A $(2 - 1/(\ell - 1))$-Approximation Algorithm for Restricted Assignment with Two Job Lengths and Bounded Job Assignments

Let $\alpha, \beta \in \mathbb{Z}^+$, where $\alpha < \beta$. Recall that the restricted assignment problem with two job lengths is a special case of $R||C_{max}$ where every processing time $p_{i,j} \in \{p_j, \infty\}$ and job length $p_j \in \{\alpha, \beta\}$. Note that if every job has the same job length, this is equivalent to the restricted assignment problem with unit job lengths and can be solved in polynomial time [21]. So below we consider instances where at least one job differs in length, and every machine can process at most $\ell \geq 3$ jobs. By modifying the algorithm we gave in Section 6 along with using some known results, we obtain an approximation algorithm with approximation ratio $2 - 1/(\ell - 1)$. Like in Section 6, there is an estimate $T$ of the optimal makespan where binary search is performed to find the smallest value for $T$ such that the algorithm below produces a schedule with makespan at most $(2 - 1/(\ell - 1))T$. Below we assume if not all of the jobs are scheduled, the algorithm reports `REJECT`. Given estimate $T$, consider the following cases in the order provided.

1. If there is a job $j \in J$ with no machine $i$ where $p_{i,j} = p_j \leq T$, report `REJECT`.
2. $\alpha > T/(\ell - 1)$ and $\beta \leq T$. Apply the $(2 - \alpha/\beta)$-approximation algorithm of Chakrabarty *et al.* [5] for estimate $T$. If a schedule exists with makespan $T$, this algorithm will compute a schedule with makespan at most
$$\left(2 - \frac{\alpha}{\beta}\right)T < \left(2 - \frac{\frac{T}{\ell-1}}{T}\right)T = \left(2 - \frac{1}{\ell - 1}\right)T.$$
3. $\alpha \leq T/(\ell - 1)$ and $\beta \leq T/2$. Use the algorithm of Lenstra *et al.* [18]. In this algorithm a fractional solution is computed using linear programming, and then a rounding is performed to integrally assign the remaining fractionally assigned jobs. If $OPT \leq T$, then solving the linear program guarantees the load of each machine is at most $T$, and the rounding step schedules at most one additional job per machine. Thus, the makespan is at most $T + \max\{\alpha, \beta\} \leq T + T/2 = (3/2)T$.
4. $\alpha \leq T/(\ell - 1)$ and $T/2 < \beta \leq T$. Use the algorithm given in Section 6 except than in Step 2, for every machine $i$ proceed as follows:
   - If every job in $\mathcal{J}_i$ is small, it is possible for every job in $\mathcal{J}_i$ to be scheduled on machine $i$, so use the Machine Plan with $d = |\mathcal{J}_i|$. The load of the machine $i$ is at most $|\mathcal{J}_i|\alpha \leq |\mathcal{J}_i|(T/(\ell - 1)) \leq \ell(T/(\ell - 1)) \leq (2 - 1/(\ell - 1))T$ as $\ell \geq 3$.
   - There is at least one big job in job set $\mathcal{J}_i$. If $\beta + \sum_{j \in \mathcal{S}_i} p_j \leq T$, use the Machine Plan with $d = |\mathcal{S}_i| + 1$, where $\mathcal{S}_i$ is the set of small jobs of job set $\mathcal{J}_i$. At most one big job can be scheduled with every job in $\mathcal{S}_i$, so the load of a machine $i$ is at most $\beta + \sum_{j \in \mathcal{S}_i} p_j \leq T$.
     If $\beta + \sum_{j \in \mathcal{S}_i} p_j > T$, then either at most one big job can be scheduled with $|\mathcal{S}_i| - 1$ small jobs or at most all $|\mathcal{S}_i|$ small jobs are scheduled

together. Add arcs according to the Machine Plan with $d = \max\{|\mathcal{S}_i|, 1\}$. Since at least one job in job set $\mathcal{J}_i$ is big, $|\mathcal{S}_i| \leq |\mathcal{J}_i| - 1 \leq \ell - 1$. If every job that is scheduled on machine $i$ is small, then the load is at most $|\mathcal{S}_i|\alpha \leq (\ell-1)(T/(\ell-1)) = T$. Otherwise, at most one big job can be scheduled with $|\mathcal{S}_i|-1$ small jobs and the load of machine $i$ is at most

$$\beta + (|\mathcal{S}_i| - 1)\alpha \leq T + ((\ell-1)-1)\Big(\frac{T}{\ell-1}\Big) = \Big(2 - \frac{1}{\ell-1}\Big)T.$$

**Theorem 3.** *There is a $(2-1/(\ell-1))$-approximation algorithm for the restricted assignment problem with two job lengths when every machine can process at most $\ell \geq 3$ jobs.*

## 8 Inapproximability Results for Job-Intersection Graphs with Cliques

For any instance $I = (P = (p_{i,j}), m, n)$ of $R||C_{max}$ with some $p_{i,j} = \infty$, there is another instance $I' = (P' = (p'_{i,j}), m, n)$ of $R||C_{max}$ with the same optimal solution but every $p'_{i,j} \neq \infty$: set $p'_{i,j} = p_{i,j}$ for any $p_{i,j} \neq \infty$; and if $p_{i,j} = \infty$, set $p'_{i,j}$ to some prohibitively large number, for example, $p'_{i,j} = np_{max} + 1$ where $p_{max}$ is the largest processing time that is not $\infty$ in $P$. For $T \leq np_{max}$, there is a schedule for instance $I$ with makespan $T$ if and only if there is a schedule for instance $I'$ with makespan $T$. Every job in instance $I'$ can be scheduled on any of the machines, so the job-intersection graph $G_J$ for $I'$ is the complete graph $K_n$. We note that an alternate construction to arrive at the complete job-intersection graph is given at the start of Section 4 in [15]. Therefore, we can carry forward the inapproximability lower bound 3/2 from the graph balancing problem with two job lengths given in Section 5.

**Corollary 4.** *There is no $k$-approximation algorithm with $k < 3/2$ for $R||C_{max}$ restricted to instances where the job-intersection graph is the complete graph $K_n$, unless* $\mathsf{P} = \mathsf{NP}$.

From Corollary 4, $R||C_{max}$ restricted to any superclass[4] of the complete job-intersection graphs inherits the 3/2-inapproximability lower bound of $R||C_{max}$. We name some of these graph classes as they are of interest from a graph-theoretic standpoint. To begin, define a job-intersection graph as a *threshold graph* if it can be constructed by repeatedly performing the following two operations: insert an isolated vertex; or insert a vertex and add edges from this vertex to every other vertex presently in the graph, this vertex is called a *dominating* vertex. All complete graphs are threshold graphs, and three superclasses of threshold graphs are interval graphs, cographs, and split graphs [4, Corollary 7.1.1]. Note that all these graphs belong to the house-free graphs. A graph is called *house free* if the graph does not contain as an induced subgraph the *house graph*, shown in Figure 6.

[4] For a comprehensive list of superclasses, we recommend the Java application at `http://www.graphclasses.org`.

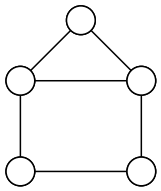

**Fig. 6.** The house graph.

**Corollary 5.** *There is no $k$-approximation algorithm with $k < 3/2$ for $R||C_{max}$ restricted to instances where the job-intersection graphs belong to either the threshold graphs, interval graphs, cographs, split graphs, or house-free graphs, unless* $\mathsf{P} = \mathsf{NP}$.

## 9 Conclusion

In this paper we have established several graph classes where $R||C_{max}$ with simple job-intersection structure is either polynomial-time solvable or 3/2-inapproximable. For $R||C_{max}$ with bounded job assignments we have shown that there are polynomial-time algorithms with approximation ratios less than two when the bounds are small. As we have demonstrated, the structure of a job-intersection graph presents another way of investigating the complexity of $R||C_{max}$. However, our work does not address planar job-intersection graphs. $R||C_{max}$ restricted to planar job-intersection graphs seems like it might not be polynomial-time solvable nor 3/2-inapproximable, we would be interested in its complexity.

## References

1. Alon, N., Yuster, R., Zwick, U.: Finding and counting given length cycles. Algorithmica **17**(3), 209–223 (1997)
2. Asahiro, Y., Jansson, J., Miyano, E., Ono, H., Zenmyo, K.: Approximation algorithms for the graph orientation minimizing the maximum weighted outdegree. J. of Combinatorial Optimization **22**(1), 78–96 (2011)
3. Asahiro, Y., Miyano, E., Ono, H.: Graph classes and the complexity of the graph orientation minimizing the maximum weighted outdegree. Discrete Applied Mathematics **159**(7), 498–508 (2011)
4. Brandstädt, A., Le, V., Spinrad, J.: Graph classes: a survey. SIAM (1999)
5. Chakrabarty, D., Khanna, S., Li, S.: On (1, $\varepsilon$)-restricted assignment makespan minimization. In: $26^{th}$ Ann. ACM-SIAM Symp. on Discrete Algorithms. 1087–1101 (2015)
6. Chang, P.Y., Damodaran, P., Melouk, S.: Minimizing makespan on parallel batch processing machines. Int. J. of Production Research **42**(19), 4211–4220 (2004)
7. Chen, L., Jansen, K., Luo, W., Zhang, G.: An efficient PTAS for parallel machine scheduling with capacity constraints. In: Int. Conf. on Combinatorial Optimization and Applications. 608–623. Springer (2016)

8. Ebenlendr, T., Krčál, M., Sgall, J.: Graph balancing: A special case of scheduling unrelated parallel machines. Algorithmica **68**(1), 62–80 (2014)
9. Ebenlendr, T., Krčál, M., Sgall, J.: Graph balancing: A special case of scheduling unrelated parallel machines. In: $19^{th}$ Ann. ACM–SIAM Symp. on Discrete Algorithms. 483–490 (2008)
10. Gairing, M., Monien, B., Woclaw, A.: A faster combinatorial approximation algorithm for scheduling unrelated parallel machines. Theoretical Computer Science **380**(1), 87–99 (2007)
11. Glass, C., Kellerer, H.: Parallel machine scheduling with job assignment restrictions. Naval Research Logistics (NRL) **54**(3), 250–257 (2007)
12. Graham, R., Lawler, E., Lenstra, J., Rinnooy, K.: Optimization and approximation in deterministic sequencing and scheduling: A survey. Annals of Discrete Mathematics **5**, 287–326 (1979)
13. Huang, C., Ott, S.: A combinatorial approximation algorithm for graph balancing with light hyper edges. In: 24th Ann. European Symp. on Algorithms. LIPIcs, vol. 57, 49:1–49:15 (2016)
14. Jansen, K., Maack, M., Solis-Oba, R.: Structural parameters for scheduling with assignment restrictions. In: Int. Conf. on Algorithms and Complexity. 357–368. Springer (2017)
15. Jansen, K., Maack, M., Solis-Oba, R.: Structural parameters for scheduling with assignment restrictions. CoRR **abs/1701.07242** (2017), `http://arxiv.org/abs/1701.07242`
16. Kloks, T., Kratsch, D., Müller, H.: Finding and counting small induced subgraphs efficiently. Information Processing Letters **74**(3-4), 115–121 (2000)
17. Lee, K., Leung, J.Y.T., Pinedo, M.: A note on graph balancing problems with restrictions. Information Processing Letters **110**(1), 24–29 (2009)
18. Lenstra, J., Shmoys, D., Tardos, E.: Approximation algorithms for scheduling unrelated parallel machines. Mathematical Programming **46**(1–3), 259–271 (1990)
19. Leung, J.Y.T., Li, C.L.: Scheduling with processing set restrictions: A literature update. Int. J. of Production Economics **175**, 1–11 (2016)
20. Li, S., Li, G., Zhang, S.: Minimizing makespan with release times on identical parallel batching machines. Discrete Applied Mathematics **148**(1), 127–134 (2005)
21. Lin, Y., Li, W.: Parallel machine scheduling of machine-dependent jobs with unit-length. European J. of Operational Research **156**(1), 261–266 (2004)
22. Page, D.R., Solis-Oba, R.: A 3/2-approximation algorithm for the graph balancing problem with two weights. Algorithms **9**(2), 38 (2016)
23. Saha, B., Srinivasan, A.: A new approximation technique for resource-allocation problems. In: $1^{st}$ Ann. Symp. on Innov. in Computer Science. 342–357 (2010)
24. Shchepin, E., Vakhania, N.: An optimal rounding gives a better approximation for scheduling unrelated machines. Operations Research Letters **33**, 127–133, 2005.
25. Williamson, D., Shmoys, D.: The Design of Approximation Algorithms. Cambridge University Press (2011)