Information Processing Letters 116 (2016) 723–728

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

On some special cases of the restricted assignment problem

Chao Wang[a,1], René Sitters[b,c,*]

[a] East China University of Science and Technology, 200237 Shanghai, China
[b] VU University Amsterdam, 1081HV Amsterdam, The Netherlands
[c] Centrum Wiskunde & Informatica (CWI), 1098XG Amsterdam, The Netherlands

ARTICLE INFO

Article history:
Received 21 April 2015
Received in revised form 17 March 2016
Accepted 16 June 2016
Available online 17 June 2016
Communicated by M. Chrobak

Keywords:
Scheduling
Restricted assignment problem
Design of algorithms

ABSTRACT

We consider some special cases of the restricted assignment problem. In this scheduling problem on parallel machines, any job j can only be assigned to one of the machines in its given subset M_j of machines. We give an LP-formulation for the problem with two job sizes and show that it has an integral optimal solution. We also present a PTAS for the case that the M_j's are intervals of the same length. Further, we give a new and very simple algorithm for the case that $|M_j| = 2$ (known as the graph balancing problem) with ratio 11/6.

1. Introduction

A classic algorithm by Lenstra, Shmoy and Tardos [1] gives a 2-approximation for minimizing the makespan on unrelated parallel machines (denoted by $R||C_{\max}$). In this problem, we are given n jobs and m machines and processing times p_{ij} which give the time needed to process job j on machine i. The goal is to assign the jobs to the machines such that the maximum load (total processing time) over all machines is minimized. The same paper shows that approximating the problem within a factor 3/2 is NP-hard. Today, no better upper or lower bound is known and improving one of the two bounds is considered one of the main open problems in the scheduling theory. Even for the so called *restricted assignment problem* no better ratios are known. Here, each job j has a processing time p_j and *processing set* $M_j \subseteq \{1, 2, \dots, m\}$ and $p_{ij} = p_j$ for $i \in M_j$ and $p_{ij} = \infty$ otherwise.

A breakthrough was made by Svensson [2], who showed that the integrality gap of the configuration LP for the restricted assignment problem is at most 1.942. However, the proof does not give a polynomial time approach for constructing a corresponding schedule. An interesting special case in which $|M_j| \leqslant 2$ for all j was considered by Ebenlendr et al. [3] who called this the *graph balancing problem*. An instance may be seen as a multigraph $G = (V, E)$ where each edge $e_j \in E$ has a weight p_j. The edges should be oriented such that the maximum total weight of incoming edges is minimized. The authors give a 1.75-approximation by LP-rounding and also show that the integrality gap of their LP is 1.75.

Based on the work of Ebenlendr et al. [3], some special cases of graph balancing have been studied. Lee, Leung and Pinedo [4] gave an FPTAS if the graph structure is restricted to a tree. Verschae and Wiese [5] showed that even the configuration-LP has an integrality gap of 1.75 for the graph balancing problem and 2 for the unrelated version where each job may have different job processing times on its two machines.

* Corresponding author at: VU University Amsterdam, 1081HV Amsterdam, The Netherlands.

E-mail addresses: c.wang@mail.ecust.edu.cn (C. Wang), r.a.sitters@vu.nl (R. Sitters).

1 This research done while visiting the VU University Amsterdam and was supported in part by the China Scholarship Council under grant No. 201306740038.

http://dx.doi.org/10.1016/j.ipl.2016.06.007
0020-0190/© 2016 Elsevier B.V. All rights reserved.

Glass and Kellerer [6] explored the restricted assignment problem with only two different processing times: 1 and $k \leqslant 2$, and gave an approximation algorithm with factor $2 - 1/k$. In the same paper, they considered the restricted assignment problem with the additional structure that the M_j's are nested and gave a $2 - 1/m$ approximation by list scheduling. For totally ordered sets a 3/2-approximation was designed. Huo and Leung [7] improved the nested case by giving an algorithm with worst-case bound of 5/3, moreover they also studied so called *tree-hierarchical processing sets*. Here, machines are the nodes of a rooted tree and processing sets correspond with a path from some vertex to the root. For the latter problem, they gave a 4/3-approximation algorithm. Muratore et al. [8] improved this nested case by designing a polynomial time approximation scheme (PTAS). Epstein and Levin [9] also studied the nested and tree-hierarchical cases, and they designed a PTAS for the two cases respectively.

For two arbitrary job sizes, Kolliopoulos and Moysoglou [10] gave a 1.883-approximation algorithm for computing the optimal value based on Svensson's [2] result. If the jobs can only be assigned on at most two machines, the ratio reduces to 1.652.

1.1. Our results

We give several results for special cases of the restricted assignment problem. First, we briefly discuss the 2-approximation algorithm for $R||C_{\max}$ given by Shmoys and Tardos [11]. We show that this approach leads to a very simple 1.88-approximation algorithm for the graph balancing problem. Although this ratio is worse than the 1.75-approximation given by Ebenlendr et al. [3], the algorithm and analysis are very easy, given the framework of [11]. Further, we study the restricted assignment problem on intervals. Here, the machines can be ordered such that each processing set is a consecutive set of machines.

We show that the special case of two different processing times s, b can be solved exactly in polynomial time, assuming that $s \leqslant b$ and $b > \text{OPT}/2$. We obtain this by formulating an exact LP, i.e., with an integrality gap of 1. Interestingly, our LP is stronger than the configuration LP which has an integrality gap of at least 3/2 in this case.

As mentioned above, the interval case has been well-studied and several polynomial time approximation schemes are known. We extend these results by giving a PTAS for the problem of equal length intervals and by simplifying the analysis of known approximation schemes by using the framework of [11].

2. Preliminaries

The 2-approximation algorithm for $R||C_{\max}$ by Lenstra, Shmoys, and Tardos [1] is well-known. A slightly different algorithm and analysis was given a few years later by Shmoys and Tardos [11]. They introduced a new rounding technique which does not require the fractional solution to be a vertex of the linear programming relaxation. An advantage is that it can therefore be applied to any fractional assignment and for variants of the problem as we shall do in this paper. Below, we give a short explanation of this rounding technique that we will use several times.

The algorithms in [1] and [11] work as follows. First, a guess T for the optimal value is made. The algorithm produces a schedule of length at most $2T$ if indeed $\text{OPT} \leqslant T$. Using a binary search on T, this will give a 2-approximation algorithm. Denote by $x_{ij} \geqslant 0$ the fraction of job J_j assigned to machine M_i, $i \in \{1, \dots, m\}$ and $j \in \{1, \dots, n\}$. If there exists a schedule of length at most T, then the following LP has a feasible solution.

$$\begin{array}{ll}
\sum_{i=1}^{m} x_{ij} = 1 & \text{for all } j \\
\sum_{j=1}^{n} x_{ij} p_{ij} \leqslant T & \text{for all } i \\
x_{ij} = 0 & \text{if } p_{ij} > T, \text{for all } i, j \\
x_{ij} \geqslant 0 & \text{for all } i, j
\end{array}$$

In [1], the authors conclude that any extreme optimal LP-solution has at most m fractional jobs and there is a perfect matching of these jobs to machines in the support graph. Hence, this gives a schedule of length at most $2T$. In [11] the approach is as follows. For each machine i, define a complete ordering $\prec_i$ such that $j \prec_i k$ if $p_{ij} \geqslant p_{ik}$. Given a feasible fractional solution x, let $y_i = \lceil \sum_j x_{ij} \rceil$ be the number of jobs (rounded up) assigned to machine i. For each machine, order the fractions assigned to it by $\prec_i$ and call this the LP-schedule. Then split this schedule up into slots such that the fractions in each slot (except for the last) add up to exactly 1. Thus, a job fraction may be split and be assigned to two contiguous slots, and there are y_i slots for machine i. Define a bipartite graph $G = (\{W, V\}, E)$ as follows. For each job j there is a vertex w_j and for each machine i there are vertices $v_1^i, \dots, v_{y_i}^i$. There is an edge between w_j and v_z^i if job j is processed (partially) in the z-th slot of machine i. It follows directly from Hall's theorem that G has a matching that contains all vertices W, i.e., all jobs.

It is easy to see that this matching yields a 2-approximation for $R||C_{\max}$. Consider an arbitrary machine i. For any slot z, let p_z^i be the maximum processing time p_{ij} among all jobs j scheduled in slot z on machine i. (In other words, let $p_z^i = \max\{p_{ij} \mid (w_j, v_z^i) \in E\}$.) Then, the job matched to v_z^i has processing time at most p_z^i. Let L_z^i be the length of slot z on machine i in the LP-schedule. Since jobs are ordered by $\prec_i$, the job matched to v_z^i has processing time at most

$$p_z^i \leqslant L_{z-1}^i \text{ for all } z \geqslant 2. \tag{1}$$

Hence, the total load of machine i in the schedule defined by the matching is at most

$$p_1^i + \sum_{z=1}^{y_i - 1} L_z^i \leqslant p_1^i + \sum_{z=1}^{y_i} L_z^i = p_1^i + \sum_{j=1}^{n} x_{ij} p_{ij} \leqslant 2T.$$

3. A simple 11/6-approximation for graph balancing

We show how the approach by Shmoys and Tardos [11], shown above, can be used to give a simple approximation algorithm for the graph balancing problem with ratio strictly less than 2. Our ratio is larger than the 1.75 given by Ebenlendr et al. [3] but our algorithm and analysis are extremely easy, given the rounding technique of the previous section.

Without loss of generality we assume that the optimal makespan is 1 and is known to the algorithm, i.e., the target value is $T = 1$. Let B be the set of *big* jobs, which are the jobs with processing time $p_j > 1/2$. Then, each machine can contain at most one big job in the optimal schedule. The following set of linear constraints is equal to (LP3) in [3]. Instead of using the graph notation from that paper we define M_j to be the set of machines on which j can be processed and assume $|M_j| \in \{1, 2\}$ for all jobs j.

$$\text{(LP)} \quad \sum_{i \in M_j} x_{ij} = 1, \quad \text{for all jobs } j \tag{2}$$

$$\sum_j x_{ij} p_j \leqslant 1, \quad \text{for all machines } i \tag{3}$$

$$\sum_{j \in B} x_{ij} \leqslant 1, \quad \text{for all machines } i \tag{4}$$

$$x_{ij} = 0, \quad \text{for all } i \notin M_j \tag{5}$$

$$x_{ij} \geqslant 0, \quad \text{for all } i, j. \tag{6}$$

Graph Balancing Algorithm $(0.5 < \beta < 1)$

Step 1. Find a feasible solution for (LP);
Step 2. If j is a big job and $x_{ij} \geqslant \beta$, then assign j to i;
Step 3. For the remaining jobs, find a perfect matching following the approach of Section 2.

Theorem 1. *The algorithm above (with $\beta = 2/3$) gives an $11/6 \approx 1.833$-approximation for the graph balancing problem.*

Proof. In Step 2, at most one big job is assigned to any machine. Further, if a big job is assigned to i in Step 2, then any other big job j with $x_{ij} > 0$ must satisfy $x_{ij} \leqslant 1 - \beta$ and $x_{i'j} \geqslant \beta$ for some other machine i' (since it can only be scheduled on two machines) and it will be assigned to i' in Step 2. Hence, after Step 2, any machine i either contains no fractional big job and at most one complete big job (call this Case 1) or it has at least one fractional big job and no complete big job and any fractional big job j on it satisfies $x_{ij} > 1 - \beta$ (Case 2).

Case 1. Assume big job j is assigned to machine i in Step 2. This causes the load to increase by at most $(1 - \beta)p_{ij} \leq 1 - \beta$. The matching procedure of Step 3 adds at most the largest value p_{ij} among the fractional jobs. Since all remaining fractional jobs on i are small, the increase due to Step 3 is at most 0.5. Therefore, the total load of i is at most $1 + (1 - \beta) + 0.5 = 2.5 - \beta$.

Case 2. Following the notation of Section 2, let p_1^i and p_2^i be the largest processing time in slot 1 and 2 on machine i. Then, $p_1^i \leqslant 1$ and, by constraint (4), $p_2^i \leqslant 0.5$. Further, by the conditions of Case 2, the largest job in slot 1 is a big job j with $x_{ij} > 1 - \beta$. Hence, $L_1^i \geqslant (1 - \beta)p_1^i + \beta p_2^i$. The load $\hat{L}^i$ of machine i after Step 3 satisfies

$$\begin{aligned}
\hat{L}^i &\leqslant p_1^i + p_2^i + \sum_{z=2}^{y_i - 1} L_z^i \\
&\leqslant p_1^i + p_2^i + 1 - L_1^i \\
&\leqslant p_1^i + p_2^i + 1 - (1 - \beta)p_1^i - \beta p_2^i \\
&= 1 + \beta p_1^i + (1 - \beta)p_2^i \\
&\leqslant 1 + \beta + (1 - \beta)/2 \\
&= 3/2 + \beta/2.
\end{aligned}$$

Combining both cases, we see that for $\beta = 2/3$, the total load of machine i is at most

$$\max\{2.5 - \beta, 1.5 + \beta/2\} = 11/6. \qquad \square$$

4. Restricted assignment on intervals

The restricted assignment problem on intervals has been discussed in several papers, [7–9,12]. Here, each processing set forms a consecutive set of machines. These papers only considered the versions with nested, laminar, or tree-hierarchical processing sets. For arbitrary intervals, no better approximation factor than 2 is known. In this section, we show how to solve the interval problem for general intervals if there are only two different processing times $s \leqslant b$ and for which OPT $< 2b$. Although this is also a rather restricted version, it holds for general intervals and can not be solved easily by dynamic programming as holds for the previously considered versions. Also, note that the general restricted assignment problem with processing times $\{\epsilon, 1\}$ and OPT $= 1$ is non-trivial and 2 is the best approximation ratio known. (Approximating just the optimal value can be done within a factor 5/3 in this case, as was shown by Svensson [2].) Our algorithm solves an LP solely for the big jobs, where the constraints are implied by the small jobs. We show that an integral solution can be found in polynomial time. Then, the small jobs are added in a greedy way. Further, we give in Section 4.2 an easy dynamic programming approach for the case of equal length intervals, a special case not yet considered in the literature.

4.1. Intervals and two processing times

We assume here that an instance only consists of *small jobs* with processing time $p_j = s$ and *big jobs* with processing time $p_j = b \geqslant s$. If the optimal value T is at least $2b$, then we immediately get a 3/2-approximation using the original 2-approximation for $R||C_{\max}$. Interestingly, the problem can be solved exactly in polynomial time if OPT $< 2b$.

Theorem 2. *The restricted assignment problem on intervals and with processing times $p_j \in \{s, b\}$ $(s \leqslant b)$ can be solved exactly in polynomial time if OPT $< 2b$.*

As before, we assume w.l.o.g. that the optimal value is guessed correctly and that OPT $= 1$. For integers $i \leqslant i'$, let $[i, i']$ be the set $\{i, \dots, i'\}$. We compute an upper bound $U(i, i')$ on the number of big jobs that can be scheduled on the machine set $[i,i']$, for each such interval $[i,i']$. Denote the interval of job j by $M_j=[l_j,r_j]$. Let $S(i,i')$ be the set of small jobs j for which $M_j \subseteq [i,i']$. Now, an easy upper bound on the number of big jobs that can be scheduled in the interval $[i,i']$ is $\lfloor (i'-i+1-s|S(i,i')|)/b \rfloor$. However, this bound turns out to be not strong enough. To improve this, note that any machine either contains one big job or none. In both cases, we know the number of small jobs that can be added to the machine. Define k_0 and k_1 as the number of small jobs that can be scheduled on a machine if there is, respectively, no or one big job. Then $k_0=\lfloor 1/s \rfloor$ and $k_1=\lfloor (1-b)/s \rfloor$. Now let $U(i,i')$ be the maximum number of big jobs that can be scheduled within the interval $[i,i']$ such that the remaining capacity is enough to add all jobs from $S(i,i')$, assuming that each of these small jobs can be scheduled on any of the machines in $[i,i']$. This upper bound can easily be computed: Denote $S=|S(i,i')|$ and $I=i'-i+1$. Then $U(i,i')$ is the maximum value $z\in\{0,1,\dots,I\}$ such that $zk_1+(I-z)k_0 \geq S$.

This leads to the following linear program, exclusively for the big jobs. Here, x_{ij} is the fraction of $j\in B$ done on machine i.

$$\text{(LP-B)} \quad \sum_{k\in[i,i']}\sum_{j\in B} x_{kj} \leqslant U(i,i') \qquad 1\leqslant i\leqslant i'\leqslant m \tag{7}$$
$$\sum_{i\in M_j} x_{ij}=1 \qquad j\in B$$
$$x_{ij}=0 \qquad i\notin M_j,\ j\in B$$
$$x_{ij}\geqslant 0 \qquad i\in M_j,\ j\in B.$$

Note that the inequalities $\sum_{j\in B} x_{ij}\leqslant 1$ for all machines i are included by taking $i=i'$ in the first constraints.

Lemma 3. *There is a feasible* 0, 1*-solution to (LP-B) and it can be found in polynomial time.*

Proof. Consider an arbitrary feasible solution x to (LP-B). For any $l\in\{1,\dots,|B|\}$, let $m(l)$ be the smallest machine index i for which $\sum_{k=1}^{i}\sum_{j\in B} x_{kj}\geqslant l$. Let $N_B=\{m(1),\dots,m(|B|)\}$. Note that all $m(l)$ are different since $\sum_{j\in B} x_{ij}\leqslant 1$ for any machine i. Hence, $|N_B|=|B|$. We show that the big jobs B can be scheduled on the set of machines N_B and that any such assignment is feasible for (LP-B).

For any $B'\subseteq B$, let $M(B')=\cup_{j\in B'}M_j$. If $M(B')$ is a consecutive set then by definition of N_B, $|M(B')\cap N_B|\geqslant \lfloor \sum_{k\in M(B')}\sum_{j\in B} x_{kj}\rfloor \geqslant \sum_{k\in M(B')}\sum_{j\in B'} x_{kj}=|B'|$. Hence,

$$|M(B')\cap N_B|\geqslant |B'|. \tag{8}$$

If on the other hand, $M(B')$ is the union of non-intersecting intervals, then (9) follows by considering the non-intersecting intervals of $M(B')$ separately. By Hall's theorem, the big jobs can be assigned to N_B.

Now assume that we do assign the big jobs to N_B. Then, the number of big jobs assigned within an interval $[i,i']$ is

$$N_B\cap[i,i']\leqslant \left\lceil \sum_{k\in[i,i']}\sum_{j\in B} x_{kj}\right\rceil \leqslant U(i,i'). \tag{9}$$

The first inequality in (9) follows by definition of N_B and the second inequality follows from (7) and the fact that $U(i,i')$ is integer. By (9), the rounded solution remains feasible for (LP-B). □

Lemma 4. *Any integral solution of (LP-B) can be completed by adding the small jobs in a greedy way.*

Proof. Given an integral solution of (LP-B), the small jobs are scheduled on the machines in the order $1,2,\dots,m$ as follows. For machine i, consider all yet unscheduled small jobs j for which $i\in M_j$ and add these jobs in non-decreasing order of their right side r_j until the load of the machine becomes more than $1-s$ or all small jobs j with $i\in M_j$ are assigned.

Assume some small job k does not get scheduled this way. Then, any machine in the set $[l_k,r_k]$ has the following two properties: (i) its load is more than $1-s$ and (ii) it only contains small jobs j with $r_j\leqslant r_k$. Let $i\leqslant l_k$ be the smallest machine index such that these two properties hold for all machines in $[i,r_k]$.

Assume $i>1$ and consider an arbitrary job j scheduled in $[i,r_k]$. Since j was not scheduled on machine $i-1$ it follows from the order in which small jobs are scheduled that $i\leqslant l_j$. Hence, for any job j scheduled on a machine in $[i,r_k]$

$$i\leqslant l_j\leqslant r_j\leqslant r_k. \tag{10}$$

Clearly, (10) holds as well for $i=1$. We see that all small jobs scheduled on one of the machines in $[i,r_k]$ are from $S(i,r_k)$ but there is no space to add job k in. This violates constraint (7) for the interval $[i,r_k]$. □

Integrality gaps. We showed that the integrality gap for our LP is 1 in this restricted version. One might also add the small jobs in the LP and get an LP for all jobs and still maintain an exact relaxation. However, as we showed, it is sufficient to consider big jobs only. Interestingly, the integrality gap of the configuration LP is at least 3/2 for this version as shown by the following example. (See [2] for a definition of the configuration LP.) There are 4 machines and 7 jobs. Jobs a, b, c have machine set $[1,2]$ and jobs d, e, f have machine set $[3,4]$. All these have $p_j=1$. The last job g has set $[1,4]$ and processing time 2. For the configuration LP solution, take $x_1(a,b)=x_1(b,c)=x_1(c,a)=1/4$ and $x_1(g)=1/4$. On machine 2, take $x_2(a,b)=x_2(b,c)=x_2(c,a)=1/4$ and $x_2(g)=1/4$. Do the same for machines 3 and 4. The LP-value is 2 while the optimal makespan is 3.

4.2. A PTAS for intervals of equal length

A PTAS for laminar set systems was given by Muratore et al. [8] and Epstein and Levin [9]. For laminar sets we have that for any two sets M_j, M_k, either $M_j\subseteq M_k$, $M_j\supseteq M_k$ or $M_j\cap M_k=\emptyset$. It is easy to see that laminar sets can be represented by a collection of intervals and it is therefore a special case of the interval problem. Here, we give a PTAS for another special case: that of equal length intervals. In fact, it holds for any instance where, for any two jobs j, k, the inequality $l_j \leqslant l_k$ implies $r_j \leqslant r_k$. Our PTAS uses similar ideas as has been used for laminar sets, such as partitioning jobs in small and large jobs. However, for scheduling the small jobs we do not construct a separate greedy procedure (as in [8] and [9]) but simply refer to the general 2-approximation for $R||C_{\max}$ as described in Section 2. We note that this idea applies to laminar set systems as well and gives a simplification over the analysis in [8] and [9].

4.2.1. The dynamic program

Let K be the number of different job sizes. Jobs of the same size are said to be of the same *type*. Let n_k be the number of jobs of type k ($k = 1, \dots, K$). For any type k label the jobs by non-decreasing values r_j. Denote the jobs in this order by $J_1^{(k)}, J_2^{(k)}, \dots J_{n_k}^{(k)}$. Polynomiality of the DP follows from the observation that there is an optimal schedule in which, for each type k, the jobs appear in order $J_1^{(k)}, J_2^{(k)}, \dots$. (If two jobs are in reversed order they can be swapped.) For every machine i, we define vectors $(b_1, b_2, \dots, b_K)$ to record the number of jobs from each type assigned to machines $1, 2, \dots, i$. Say $F_i(b_1, b_2, \dots, b_K) = 1$ if it is feasible to schedule the jobs $\cup_k \{J_1^{(k)}, \dots, J_{b_k}^{(k)}\}$ within a makespan of T on the first i machines and let it be 0 otherwise. Define $F_0(0, \dots, 0) = 1$. Then $F_i(b_1, b_2, \dots, b_K) = 1$ if and only if there are numbers $a_k \le b_k$ ($k = 1 \dots K$) such that

- $F_{i-1}(a_1, a_2, \dots, a_K) = 1$.
- For each k, all jobs $J_{a_k+1}^{(k)}, \dots, J_{b_k}^{(k)}$ contain i in their interval.
- The total length of the jobs in $\bigcup_k \{J_{a_k+1}^{(k)}, \dots, J_{b_k}^{(k)}\}$ is at most T.

A feasible schedule exists if and only if $F_m(n_1, \dots, n_K) = 1$. The size of the DP table is $O(mn^K)$ and the computation of each value takes $O(n^K)$ time.

4.2.2. Rounding the instance

Let $T = 1$ be the target value and assume that $OPT \leqslant 1$ (hence, $p_j \leqslant 1$ for all j). Let $1/\epsilon$ be a large positive integer. Call a job j *big* if $p_j > \epsilon$ and call it *small* otherwise. Round the processing time of every big job down to $p'_j = \lfloor \frac{p_j}{\epsilon^2} \rfloor \epsilon^2$. Consequently, the number of different processing times of the rounded big jobs is at most $1/\epsilon^2$. For every small job, round its processing time down to $p'_j = \left\lfloor \frac{np_j}{\epsilon} \right\rfloor \frac{\epsilon}{n}$. Next, split each rounded small job into $\lfloor \frac{np_j}{\epsilon} \rfloor$ tiny jobs with processing time $\frac{\epsilon}{n}$ for each of them. The intervals for these tiny jobs are unchanged. So in the rounded instance, we only have the rounded big jobs and the tiny jobs of fixed size $\frac{\epsilon}{n}$. The total number of job sizes is $K = O(1/\epsilon^2)$. If $\text{OPT} \leqslant 1$, then there is feasible schedule of length at most 1 for the rounded instance as well. We can use the DP to find an optimal solution for the rounded instance in polynomial time.

Theorem 5. *Given an optimal schedule for the rounded instance, we can find a schedule for the original instance with makespan $C_{max} \leqslant 1 + 4\epsilon$.*

Proof. We keep the assignment of the big jobs the same as in the optimal schedule for the rounded instance as given by the DP. The rounded processing time of a big job is at least $\lfloor \frac{\epsilon}{\epsilon^2} \rfloor \epsilon^2 > \epsilon - \epsilon^2 > \epsilon/2$ for small enough ϵ, e.g. $\epsilon < 1/2$. Consequently, there are at most $2/\epsilon$ big jobs on a machine giving a total rounding error due to big jobs of at most $2/\epsilon \cdot \epsilon^2 = 2\epsilon$. Note that the DP gives us an assignment for the tiny jobs which can be seen as a fractional assignment for the rounded small jobs. We apply the matching procedure of Section 2 to these fractions. This procedure gives an integer assignment for the rounded small jobs with an increase in the makespan of at most the length of one rounded small job, which is less than ϵ. Finally, rounding up the processing time of the small jobs gives an additional increase of at most $n \cdot \frac{\epsilon}{n} = \epsilon$. Hence, the total increase in the makespan is at most 4ϵ. □

5. Some open problems

The main subject of our research has been the restricted assignment problem on intervals. The problem turned out to be much harder than expected and it is not clear if a PTAS exists. Even for the restricted version described under Problem 1, no better approximation than the general 2-approximation for $R||C_{\max}$ is known.

Problem 1. Give an approximation ratio better than 2 for the restricted assignment problem with interval processing sets. Even for the problem in which all intervals share one machine nothing better is known. We conjecture that a $3/2$-approximation is possible.

Problem 2. Find an approximation ratio better than 2 for the restricted assignment problem with processing sets of size at most 3.

Problem 3. What is the complexity of the interval problem with only two processing times s, b (as in Section 4.1) and $\text{OPT} \geqslant 2b$? For the very special case that $\text{OPT} = 2b = 3a$ and small jobs can go to any machine, we can solve the problem by a similar technique as in Section 4.1. To see this, note that in the optimal solution, a machine can hold two big jobs or three small jobs. If there is only one big job on a machine, then only one more small job can be scheduled in the space left. If there are two big jobs or three small jobs on a machine, then the machine gets fully occupied. So the problem is equivalent to finding the maximal number of paired big jobs.

References

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

[2] O. Svensson, Santa Claus schedules jobs on unrelated machines, in: STOC 2011 Proceedings of the 43rd Annual ACM Symposium on Theory of Computing, 2011, pp. 617–626.

[3] T. Ebenlendr, M. Krcál, J. Sgall, Graph balancing: a special case of scheduling unrelated parallel machines, Algorithmica 68 (1) (2014) 62–80.

[4] K. Lee, J. Leung, M. Pinedo, A note on graph balancing problems with restrictions, Inf. Process. Lett. 110 (2009) 24–29.

[5] J. Verschae, A. Wieese, On the configuration-LP for scheduling on unrelated machines, in: Algorithms-ESA, 2011, pp. 530–542.

[6] C. Glass, H. Kellerer, Parallel machine scheduling with job assignment restrictions, Nav. Res. Logist. 54 (3) (2007) 250–257.

[7] Y. Huo, J. Leung, Fast approximation algorithms for job scheduling with processing set restrictions, Theor. Comput. Sci. 411 (2010) 3947–3955.

[8] G. Muratore, U. Schwarz, G. Woeginger, Parallel machine scheduling with nested job assignment restrictions, Oper. Res. Lett. 38 (2010) 47–50.

[9] L. Epstein, A. Levin, Scheduling with processing set restrictions: PTAS results for several variants, Int. J. Prod. Econ. 133 (2011) 586–595.

[10] S. Kolliopoulos, Y. Moysoglou, The 2-valued case of makespan minimization with assignment constraints, Inf. Process. Lett. 113 (2013) 39–43.

[11] D. Shmoys, E. Tardos, An approximation algorithm for the generalized assignment problem, Math. Program. 62 (1993) 461–474.

[12] J. Leung, C. Li, Scheduling with processing set restrictions: a survey, Int. J. Prod. Econ. 116 (2008) 251–262.