An improved monotone algorithm for scheduling related machines with precedence constraints

Anke van Zuylen[*]

Max Planck Institute for Informatics, Saarbrücken, Germany

Abstract

We answer an open question posed by Krumke, Schwahn, van Stee and Westphal by showing how to turn the algorithm by Chekuri and Bender for scheduling related machines with precedence constraints into a $O(\log m)$-approximation algorithm that is monotone in expectation. This significantly improves on the previously best known monotone approximation algorithms for this problem, by Krumke et al. and Thielen and Krumke, which have an approximation guarantee of $O(m^{2/3})$.

Key words: Scheduling, algorithmic mechanism design, precedence constraints, monotone algorithms

[*]Corresponding author. Max Planck Insitute for Informatics, Campus E 1 4, 66123 Saarbrcken, Germany

Email address: `anke@mpi-inf.mpg.de`

1. Introduction

In this paper we consider the problem of scheduling jobs on related machines when the machines are controlled by selfish agents.

The classical version of the problem, in which the machines are not controlled by selfish agents, is called $Q|prec|C_{\max}$ in the notation introduced by Graham, Lawler, Lenstra and Rinnooy Kan [4]. We are given a set of n jobs with processing times $p_1, \ldots, p_n$, precedence constraints on the jobs in the form of a partial order, and m machines with speeds $s_1, \ldots, s_m$. If job j is processed on machine i, it takes p_j/s_i time units to process. A schedule is an assignment of jobs to machines, plus a starting time for each job, so that a job starts after all jobs that must precede it have been processed, and a machine works on only one job at a time. The goal is to minimize the makespan of the schedule, i.e. the finishing time of the last job.

Motivated by applications on the Internet, in recent years researchers have considered scheduling problems in which the machines are controlled by selfish agents. In our setting, this means that we do not know the speeds of the machines. Our goal is to design a mechanism which asks the agents for their speeds, assigns the jobs to the agents so as to minimize the makespan, and computes payments to pay each agent. Each agent, on the other hand, is trying to maximize her profit, which is equal to the payment received minus the time it takes to process the jobs assigned to the agent. Hence, agents may choose to misrepresent their speed. A *truthful* mechanism is a mechanism in which reporting the true speed is a dominant strategy for each agent, i.e. no matter what the speeds are that are reported by the other agents, the strategy which yields the maximum profit for a given agent is to report its true speed.

The problem we consider belongs to the class of mechanism design problems for one-parameter agents. For a problem in this class, there are m agents, and each agent i holds some piece of private data consisting of a single parameter $t_i \in \mathbb{R}_{\geq 0}$. All other input is public knowledge. The vector $t = (t_1, \ldots, t_n)$ combined with the rest of the input defines a (classical) optimization problem, for which a feasible solution takes the form of a set of loads $w_i(t)$ to be assigned to the agents. A mechanism for a problem in this class solicits a bid b_i from each agent, and based on $b = (b_1, \ldots, b_m)$, it computes feasible loads $w_i(b)$ and payments $p_i(b)$ for agents $i = 1, \ldots, m$. The goal of agent i is to maximize $\text{profit}_i = p_i(b) - w_i(b)t_i$. Let b_{-i} be the bids of the other agents, and let $b = (b_{-i}, b_i)$ denote the vector containing all bids. The mechanism is truthful if for each agent i, reporting her true private value t_i dominates any other bid b_i, i.e., $\text{profit}_i(b_{-i}, t_i) \geq \text{profit}_i(b_{-i}, b_i)$, for all b_{-i}, b_i.

It was shown by Archer and Tardos [1] that a necessary and sufficient condition to obtain a truthful mechanism for this class of mechanism design problems is to have a *monotone* algorithm for the underlying problem, in which t is part of the known input. An algorithm is monotone if $w_i(t_{-i}, t_i)$ is a non-increasing function of t_i. Given such an algorithm, Archer and Tardos show how to compute payments that induce truth telling. In our setting, an agent's private information is $t_i = \frac{1}{s_i}$, and a monotone algorithm means that the amount of work assigned to a machine should not increase if the speed of the machine is decreased (and all other input remains the same). Since the problem $Q|prec|C_{\max}$ is NP-hard, we are concerned with finding monotone approximation algorithms for $Q|prec|C_{\max}$. Combined with the payment scheme of [1], a monotone α-approximation algorithm results in a truthful mechanism that finds a schedule with makespan at most α times optimal.

The problem of finding a monotone approximation algorithm for $Q|prec|C_{\max}$ was first considered by Krumke, Schwahn, van Stee and Westphal [6]. They showed how to turn an $O(\sqrt{m})$-approximation algorithm by Jaffe [5] into a monotone $O(m^{2/3})$-approximation algorithm. Their approach was generalized by Thielen and Krumke [8] to a general scheme for designing monotone algorithms. Their scheme also yields a monotone algorithm for the problem $Q|prec, r_j|C_{\max}$, in which the existence of a job j is unknown until its release time r_j. The best known (not necessarily monotone) approximation algorithms for $Q|prec|C_{\max}$ and $Q|prec, r_j|C_{\max}$ are $O(\log m)$-approximation algorithms by Chekuri and Bender [2] and Chudak and Shmoys [3]. An open question posed by Krumke et al. [6] is whether one can get better monotone approximation algorithms using these approaches.

In this paper, we slightly weaken the monotonicity requirement and answer Krumke et al.'s question affirmatively under this weakened definition. In particular, we show that a slightly modified and randomized version of the algorithm by Chekuri and Bender [2] is a $O(\log m)$-approximation algorithm that is monotone *in expectation*: the *expected* amount of work assigned to a machine does not increase if the machine's speed is decreased. Using Archer and Tardos [1]'s result, this implies a mechanism that is truthful in expectation: an agent cannot improve its expected payoff by being untruthful. Using a result by Shmoys, Wein and Williamson [7], it is straightforward to extend our approach to $Q|prec, r_j|C_{\max}$ and obtain a $O(\log m)$-approximation algorithm that is monotone in expectation.

2. Problem Definition

In the problem $Q|prec|C_{\max}$, we are given a set of n jobs with processing times $p_1, \dots, p_n$, precedence constraints on the jobs in the form of a partial order $\prec$, and machines with speeds $\{s_1, \dots, s_m\}$. If job j is processed on machine i, it takes p_j/s_i time units to process. A schedule is an assignment of jobs to machines, plus a starting time for each job, so that a job k starts after all jobs j such that $j \prec k$ have been processed, and a machine works on only one job at a time. We assume without loss of generality that $s_1 \geq s_2 \geq \dots, \geq s_m$. The makespan of a schedule is the time when the last job is finished. We'll denote by $C^*_{\max}$ the minimum makespan over all feasible schedules.

We define the amount of work assigned to machine i as the sum of p_j over all jobs j that are assigned to machine i. An algorithm for $Q|prec|C_{\max}$ is monotone if the following holds: if we consider two instances that are identical except for the speed of machine i, which is s in the first instance and $s' > s$ in the second instance, then the amount of work assigned to machine i in the schedule produced for the second instance is not smaller than the amount of work assigned to it in the first schedule. A randomized algorithm is monotone in expectation if the expected amount of work assigned to machine i in the second instance is not smaller than the expected amount of work assigned to it in the first schedule.

The problem $Q|prec, r_j|C_{\max}$ is defined similarly to $Q|prec|C_{\max}$, except that the existence of a job is not known until the time r_j when it is released.

3. The Chekuri-Bender algorithm

We begin by describing the algorithm proposed by Chekuri and Bender [2]. We will need to make only a few changes to achieve monotonicity, which we will describe in the next section.

The algorithm of Chekuri and Bender [2] begins by computing a lower bound B on the makespan of any feasible schedule. To do so, they find a maximal chain decomposition of the jobs: A chain P is a subset of jobs $j_1, \ldots, j_k$ such that $j_i \prec j_{i+1}$ for all $i \in \{1, \ldots, k-1\}$. The length of P is denoted by $|P| = \sum_{i=1}^{k} p_{j_i}$. A chain decomposition is a partition of the precedence order into an ordered collection of chains $(P_1, \ldots, P_r)$ such that P_1 is a longest chain and $(P_2, \ldots, P_r)$ is a maximal chain decomposition with the jobs in P_1 removed. If we consider the first k chains in the chain decomposition, then at any point in time at most k different machines are working on the jobs in these chains. The total speed of these k machines is at most $\sum_{i=1}^{k} s_i$ where we recall that speeds are assumed to be ordered so that $s_i \geq s_{i+1}$. Hence there must be some job in these k chains which completes at $\frac{\sum_{i=1}^{k}|P_i|}{\sum_{i=1}^{k} s_i}$ or later. This observation gives rise to the following lower bound on the makespan:

$$B = \max\left\{\frac{\sum_{j=1}^{n} p_j}{\sum_{i=1}^{m} s_i}, \max_{1 \leq k \leq \min\{r,m\}}\left\{\frac{\sum_{i=1}^{k}|P_i|}{\sum_{i=1}^{k} s_i}\right\}\right\} \leq C^*_{\max}.$$

The Chekuri-Bender algorithm uses an idea of Chudak and Shmoys [3] to reduce an instance with an arbitrary number of speeds to an instance with only $K = O(\log m)$ speeds while losing only a constant factor in the approximation ratio: ignore machines with speed less than $\frac{1}{m}$ times the speed of the fastest machine, and divide the remaining machine speeds into speed classes, by rounding down all speeds to the nearest power of 2.

Let $\bar{s}_1, \ldots, \bar{s}_K$ be the remaining distinct speeds. The algorithm iterates through the chains in the maximal chain decomposition and assigns them to the speed classes. See Figure 1 for a description of the algorithm Chain-Alloc that allocates the chains. For a given assignment, let $k(j)$ be the speed class that (the chain containing) job j is assigned to. Given the output of Chain-Alloc, the jobs are scheduled according to speed-based list scheduling [3]: if a job j is available and there is a free machine i such that machine i is in speed class $k(j)$, then schedule job j on machine i.

```
Chain-Alloc

Compute a maximal chain decomposition of the jobs $\mathcal{P} = (P_1, P_2, \ldots, P_r)$.
Set $B = \max\left\{\frac{\sum_{j=1}^{n} p_j}{\sum_{i=1}^{m} s_i}, \max_{1\le k\le \min\{r,m\}}\left\{\frac{\sum_{i=1}^{k}|P_i|}{\sum_{i=1}^{k} s_i}\right\}\right\}$.

Ignore all machines $i$ with $s_i < \frac{1}{m}s_1$, and round down the speeds of all
other machines to the nearest power of 2.
Let $\bar{s}_1, \ldots, \bar{s}_K$ be the remaining distinct speeds.
Let $m_k$ the number of machines with (rounded) speed $\bar{s}_k$.
Let $\ell = 1$.
For $k = 1$ to $K$ do
        Let $b \le r$ be the maximum index such that $\sum_{j=\ell}^{b} |P_j| \le 4Bm_k\bar{s}_k$.
        Assign jobs in chains $P_\ell, \ldots, P_b$ to speed $k$.
        Let $\ell = b + 1$. If $\ell > r$ return.
return.
```

Figure 1: Algorithm of Chekuri and Bender [2] for allocating chains to speed classes.

We repeat a key point in the analysis by Chekuri and Bender, which we will be using in the next section to show monotonicity. The following lemma is (contained in) Lemma 2 in [2].

Lemma 1 (Chekuri & Bender [2]). *Let $P_{\ell(k)}, \ldots, P_r$ be the chains remaining when Chain-Alloc considers speed k. Then $|P_{\ell(k)}|/\bar{s}_k \le 2B$.*

4. A monotone algorithm

Recall that our goal is to show that if a machine changes its speed from s to $s' > s$, then the amount of work assigned to it does not decrease. We begin by analyzing the original Chekuri-Bender algorithm. Our analysis will highlight certain cases in which monotonicity is not guaranteed, and we show two small adaptations which do ensure monotonicity.

Let m_k be the number of machines with rounded speed $\bar{s}_k$, and define

$$D_k = \frac{1}{m_k\bar{s}_k}\sum_{j:k(j)=k} p_j.$$

We will assume that in the speed-based list scheduling phase of the algorithm, all machines in one speed class are indistinguishable for the algorithm. To be precise, we simulate the speed-based list scheduling where all machines have speed equal to their true speed rounded down to the nearest power of 2, and machines within one speed class are considered in random order. This gives an assignment of jobs to machines, plus a starting time for each job, which is feasible for the real instance. The expected amount of work assigned to a machine in speed class k is equal to $D_k \bar{s}_k$.

The following lemma allows us to bound the expected amount of work assigned to a machine.

Lemma 2. *For any $1 \le k \le K$ we have that either $3B < D_k \le 4B$, or $D_u = 0$ for $u > k$.*

Proof: From Lemma 1, we know that if there are still chains $P_{\ell(k+1)}, \ldots, P_r$ remaining when considering speed $k+1$ then $|P_{\ell(k+1)}|/\bar{s}_{k+1} \le 2B$. Since $\bar{s}_k \ge 2\bar{s}_{k+1}$, we get that $|P_{\ell(k+1)}|/\bar{s}_k \le B$. Hence if $P_j, \ldots, P_{\ell(k+1)-1}$ are the chains assigned to speed k, we have that

$$D_k = \frac{\sum_{i=j}^{\ell(k+1)-1} |P_j|}{m_k \bar{s}_k} = \frac{\sum_{i=j}^{\ell(k+1)} |P_j|}{m_k \bar{s}_k} - \frac{|P_{\ell(k+1)}|}{m_k \bar{s}_k} > 4B - B = 3B.$$

The upper bound on D_k follows directly from the way the algorithm assigns the chains to the speed classes. □

Now, if a machine changes its speed, the bound B changes:

Lemma 3. *If B is the lower bound on a particular instance, and we change the speed of some machine from s to s' where $s' > s$ and let $\tilde{B}$ be the new lower bound, then $\tilde{B} \ge \frac{s}{s'} B$.*

Proof: Let the speeds in the original instance be $s_1 \ge s_2 \ldots \ge s_m$, and let the speeds in the modified instance be $s'_1 \ge s'_2 \ge \ldots \ge s'_m$. Let the machine of which the speed is changed be the ℓ-th machine in the original ordering, so $s_\ell = s$, and let $u \le \ell$ be the position of this machine in the new ordered set of speeds, so $s'_i = s_i$ for $i < u$ and $i > \ell$, $s'_u = s'$, and $s'_i = s_{i-1}$ for $u+1 \le i \le \ell$.

Note that it is enough to show that for any $k \le m$

$$\sum_{i=1}^{k} s'_i \le \frac{s'}{s} \sum_{i=1}^{k} s_i.$$

We'll show that $s'_i \le \frac{s'}{s} s_i$ for every $i = 1, \ldots, m$: For $i < u$ and $i > \ell$, $s'_i = s_i$ and since $s' > s$, indeed $s'_i \le \frac{s'}{s} s_i$. For $i = u$, $s'_u = \frac{s'}{s} s = \frac{s'}{s} s_\ell \le \frac{s'}{s} s_u$ since $\ell \ge u$ so $s_\ell \le s_u$. For $u+1 \le i \le \ell$, note that $s'_i = s_{i-1} = \frac{s_{i-1}}{s_i} s_i \le \frac{s'}{s} s_i$, where the last inequality follows since (i) $s_{i-1} \le s_u \le s'$ since $i-1 \ge u$ and $s' = s'_u \ge s'_{u+1} = s_u$, and (ii) $s_i \ge s_\ell = s$. □

If we disregard the second case in Lemma 2, then we see that the Chekuri-Bender algorithm is close to being monotone: The expected amount of work processed by a machine in speed class $\bar{s}_k$ is $D_k \bar{s}_k \in (3B\bar{s}_k, 4B\bar{s}_k]$. Now suppose one machine changes its speed from $\bar{s}_k$ to $\bar{s}_{k'} > \bar{s}_k$, i.e., we assume this machine's speed is a power of 2 both before and after its speed is raised. Let $\tilde{B}, \tilde{D}_k$ denote the new bound and average loads. Then the expected amount of work assigned to the machine is $\tilde{D}_{k'} \bar{s}_{k'} \in (3\tilde{B}\bar{s}_{k'}, 4\tilde{B}\bar{s}_{k'}]$, where $3\tilde{B}\bar{s}_{k'} \ge 3B\bar{s}_k$ by

Lemma 3! It is not yet monotone, since it is possible that $D_k \bar{s}_k = 4B\bar{s}_k$ and $\tilde{D}_{k'}\bar{s}_{k'} = 3B\bar{s}_k + \epsilon$, but it is not that far off.

The first change we make to the Chekuri-Bender algorithm is that we round down the speeds of the machines to the nearest power of 2 *before* we call the Chain-Alloc algorithm. In Section 4.1, we will show that in an instance in which all speeds are powers of 2, we can strengthen the relation found in Lemma 3 between the value B before and after we change the speed of a single machine, unless the machine that changes its speed was in the first speed class in the original instance. This improved relation between the bounds is enough to deal with speed changes except when the machine that changes its speed is in the first speed class.

We will show how to deal with increase in speed of one of the machines in the first speed class in Section 4.2, by assigning a higher load to the first speed class if it has only one machine.

4.1. Rounding the speeds before calling the Chain-Alloc algorithm

Lemma 4. *Suppose B is the bound on a particular instance with speeds that are powers of 2, and consider some machine with speed $s < s_1$. Let $\tilde{B}$ be the new bound, if we replace this machine with a machine with speed $s' > s$, where s' is again a power of 2. Then $\tilde{B} \geq \frac{4}{3}\frac{s}{s'}B$.*

Proof: Consider the set of chains and machines on which B attains its value, i.e. let

$$B = \frac{\sum_{j=1}^{\ell} |P_j|}{\sum_{j=1}^{k} s_j},$$

where either $\ell = k \leq \min\{r, m\}$, or $\ell = r$ and $k = m$ (in the latter case, the numerator is equal to $\sum_{j=1}^{n} p_j$, since the r chains give a partition of the set of all jobs). Then also $\tilde{B} \geq \frac{\sum_{j=1}^{\ell}|P_j|}{\sum_{j=1}^{k} s'_j}$, where s'_i are the new ordered speeds. Note that $\sum_{j=1}^{k} s'_j - \sum_{j=1}^{k} s_j \leq s' - s$. We'll show that for any $k \geq 1$, $s' - s \leq \frac{3}{4}\frac{s'}{s}\sum_{j=1}^{k} s_j - \sum_{j=1}^{k} s_j$. It follows that $\sum_{j=1}^{k} s'_j \leq \frac{3}{4}\frac{s'}{s}\sum_{j=1}^{k} s_j$, and thus $\tilde{B} \geq \frac{4}{3}\frac{s}{s'}\frac{\sum_{j=1}^{\ell}|P_j|}{\sum_{j=1}^{k} s_j} = \frac{4}{3}\frac{s}{s'}B$.

To show that $s' - s \leq \frac{3}{4}\frac{s'}{s}\sum_{j=1}^{k} s_j - \sum_{j=1}^{k} s_j$, note that $s < s_1$, and since s

and s_1 are powers of two, $s \leq \frac{1}{2}s_1$. Therefore $s \leq \frac{1}{2}\sum_{j=1}^{k} s_j$, which gives:

$$\begin{aligned} s' - s = (\frac{s'}{s} - 1)s &\leq (\frac{s'}{s} - 1)\frac{1}{2}\sum_{j=1}^{k} s_j \\ &= \frac{1}{2}\frac{s'}{s}\sum_{j=1}^{k} s_j - \frac{1}{2}\sum_{j=1}^{k} s_j \\ &= \frac{3}{4}\frac{s'}{s}\sum_{j=1}^{k} s_j - (\frac{1}{4}\frac{s'}{s} + \frac{1}{2})\sum_{j=1}^{k} s_j \\ &\leq \frac{3}{4}\frac{s'}{s}\sum_{j=1}^{k} s_j - \sum_{j=1}^{k} s_j, \end{aligned}$$

where the last inequality follows since $s' \geq 2s$. $\square$

Lemma 5. *If we call the Chekuri-Bender algorithm with speeds rounded down to the nearest power of 2, then the expected amount of work assigned to a machine does not decrease if it changes its speed from s to $s' > s$ unless the machine that changes its speed is in the first speed class before it changes its speed.*

Proof: Since the algorithm only uses the speeds rounded down to the nearest power of 2, we consider a machine that changes its *rounded* speed from s to $s' > s$. We will call the instance where this machine's rounded speed is s the *original* instance, and the instance where every other machine has the same speed, but this machine has rounded speed s' the *modified* instance.

For any value computed by the algorithm, we denote by a tilde that it is the value computed when we run the algorithm on the modified instance, e.g. B and $\tilde{B}$ are the bounds computed by the algorithm if the machine's speed is s and s' respectively.

Assume that $s \geq \frac{1}{m}s_1$, since otherwise no jobs are assigned to the machine in the original instance, and the lemma is obviously satisfied. Let k be the speed class that s belongs to, and k' the speed class that s' belongs to. For ease of exposition, we assume that for both the original instance and the modified instance the algorithm considered the speed class $\bar{s}_{k'}$, even though it may be the case that there were no machines with speed $\bar{s}_{k'}$ in the original instance ($m_{k'} = 0$). Note however that adding k' with $m_{k'} = 0$ into the for-loop of the Chain-Alloc algorithm does not change the outcome of the algorithm.

We consider two cases.

Case (i) $\tilde{D}_{k'} > 3\tilde{B}$. By Lemma 2, the expected amount of work assigned to the machine in the original instance is at most $4Bs$, and the expected amount of work for the machine in the modified instance is at least $3\tilde{B}s'$. Since the machine that changed its speed was not in the first speed class, by Lemma 4, $3\tilde{B}s' \geq 4Bs$.

Case (ii) $\tilde{D}_{k'} \leq 3\tilde{B}$.

Claim 6. *If $\tilde{D}_{k'} \leq 3\tilde{B}$ then the jobs assigned to speed class k' in the modified instance form a superset of the jobs assigned to speed classes k' up to k in the original instance.*

When we consider speed class k' in the modified instance, all remaining jobs are assigned to speed class k', by Lemma 2 and the fact that $\tilde{D}_{k'} \leq 3\tilde{B}$. Hence it is enough to show that the chains remaining when we consider k' in the modified instance form a superset of the chains remaining when we consider k' in the original instance.

Let $P_{\ell(j)}, \ldots, P_r$ be the chains remaining when we consider speed j in the original instance, and $P_{\tilde{\ell}(j)}, \ldots, P_r$ be the chains remaining when we consider speed j in the modified instance. We show by induction that $\ell(j) \geq \tilde{\ell}(j)$ for all $j \leq k'$. Clearly, for $j = 1$ this is true. So suppose it holds for $j \leq k' - 1$. Note that $m_j \bar{s}_j$ does not change, and $\tilde{B} \leq B$. So if $\sum_{i=\tilde{\ell}(j)}^{\tilde{\ell}(j+1)-1} |P_i| \leq 4\tilde{B} m_j \bar{s}_j$, then also $\sum_{i=\tilde{\ell}(j)}^{\tilde{\ell}(j+1)-1} |P_i| \leq 4B m_j \bar{s}_j$. Since $\tilde{\ell}(j) \leq \ell(j)$ by the induction hypothesis, we must assign chains $\ell(j), \ldots, \tilde{\ell}(j+1) - 1$ to speed j in the original instance, so $\ell(j+1) \geq \tilde{\ell}(j+1)$. $\diamond$

Let W_k and $W_{k'}$ be the total amount of work assigned to speed class k and k' in the original instance, i.e. $W_k = m_k \bar{s}_k D_k$ and $W_{k'} = m_{k'} \bar{s}_{k'} D_{k'}$. In the original instance, the expected amount of work assigned to the chosen machine is W_k/m_k. By the claim, the expected amount of work assigned to this machine in the modified instance is at least $\frac{W_k + W_{k'}}{m_{k'}+1}$. Since $m_k \geq 1$, it is enough to show that $W_{k'} \geq \frac{m_{k'}}{m_k} W_k$. If $W_k = 0$, then monotonicity clearly holds, so assume $W_k > 0$. By Lemma 2, $W_k \leq m_k \bar{s}_k 4B$. Since $k' < k$, by Lemma 2 we also know that $W_{k'} \geq m_{k'} \bar{s}_{k'} 3B$. Note that $\bar{s}_{k'} \geq 2\bar{s}_k$. Hence $W_{k'} \geq m_{k'} \bar{s}_k 6B \geq \frac{m_{k'}}{m_k} W_k$. $\square$

4.2. Assigning a higher load to unique machine in the first speed class

In the previous section, we did not yet show that monotonicity holds if the machine that changes its speed is in the first speed class, because Lemma 4 does not hold in this case. We now show how to adapt the algorithm so that monotonicity is ensured in this case as well. Note that if a machine in the first speed class increases its speed, then it forms a new speed class in which it is the only machine. To ensure that the expected amount of work assigned to the machine does not decrease in this case, we therefore make a small change in the assignment for the first speed class: if there is only one machine in the first class, it will get a slightly larger load.

In particular, if $m_1 = 1$, we change the upper bound on the length of the chains assigned to speed class 1 in Figure 1:

```
If $k = 1$ and $m_1 = 1$ then
    let $b \le r$ be the maximum index such that $\sum_{j=\ell}^{b} |P_j| \le 5Bm_k\bar{s}_k$,
otherwise
    let $b \le r$ be the maximum index such that $\sum_{j=\ell}^{b} |P_j| \le 4Bm_k\bar{s}_k$.
```

It can be verified that the proof of Lemma 1 given by Chekuri and Bender [2] still holds in this case. This gives the following new version of Lemma 2, the proof of which is similar to the proof of Lemma 2.

Lemma 7. *Let $\alpha_k = 1$ if $k = 1$ and $m_1 = 1$, and $\alpha_k = 0$ otherwise. For any $1 \le k \le K$ we have that either $(3 + \alpha_k)B < D_k \le (4 + \alpha_k)B$, or $D_u = 0$ for $u > k$.*

We now show that the two changes we proposed lead to a monotone algorithm.

Lemma 8. *If we call the Chekuri-Bender algorithm with speeds rounded down to the nearest power of 2, and use the modified assignment rule for the first speed class if it has only one machine, then the amount of work assigned to a machine does not decrease if it changes its speed from s to $s' > s$.*

Proof: As in the proof of Lemma 5, we may assume a machine changes its *rounded* speed from s to $s' > s$. We will call the instance where this machine's rounded speed is s the *original* instance, and the instance where every other machine has the same speed, but this machine has rounded speed s' the *modified* instance. As in the proof of Lemma 5, for any value computed by the algorithm, we denote by a tilde that it is the value computed when we run the algorithm on the modified instance.

If the machine that changes its speed is not in the first speed class, then the proof of Lemma 5 shows that the expected amount of work assigned to the machine does not decrease.

Now suppose the machine is in the first speed class before it changes its speed, and note that it becomes the unique fastest machine after it changes its speed.

If $\tilde{D}_1 \le 4\tilde{B}$, then by Lemma 7 all jobs are assigned to this machine, and hence monotonicity is ensured. If $\tilde{D}_1 > 4\tilde{B}$, we consider the cases when the machine that changed its speed was or was not the unique machine in the first speed class before it changed its speed. In the second case, note that by Lemma 7 it received at most $4Bs$ before it changed its speed, and that by Lemma 3, $\tilde{B} \ge \frac{s}{s'}B$, and hence the amount of work assigned to the machine after it changes its speed is at least $4Bs$. In the first case (i.e. the machine that changes its speed is the unique machine in the first speed class in the original instance), let $P_1, \ldots, P_b$ be the chains allocated to it originally. Then $\sum_{j=1}^{b} |P_j| \le 5Bs$. Moreover, from Lemma 3, $B \le \frac{s'}{s}\tilde{B}$, hence $\sum_{j=1}^{b} |P_j| \le 5\tilde{B}s'$, hence these chains are also allocated to the machine when its speed is s'. □

Theorem 9. *There exists a randomized $O(\log m)$-approximation algorithm for $Q|prec|C_{\max}$ that is monotone in expectation.*

Proof: By Lemma 8, the modified Chekuri-Bender algorithm is monotone in expectation. The correctness of the algorithm follows from the analysis of Chekuri and Bender [2]: in particular, they show that it follows from Lemma 1 that all chains in the maximal chain decomposition (and hence all jobs) are assigned to a speed class, and will hence be scheduled in a feasible way by the speed-based list scheduling algorithm.

The approximation factor follows from Lemma 4 in Chekuri and Bender [2], and Theorem 2.1 from Chudak and Shmoys [3]: Let $\mathcal{P}$ be the set of all chains induced by the precedence constraints, and for a given job j let $k(j)$ be the speed class that j gets assigned to, and let $C = \max_{P \in \mathcal{P}} \sum_{j \in P} \frac{p_j}{\bar{s}_{k(j)}}$. Chudak and Shmoys show that the length of the schedule produced by speed-based list scheduling is at most $C + \sum_{k=1}^{K} D_k$. We know from Lemma 7 that $D_k \leq 5B$, and it follows from Lemma 4 in Chekuri and Bender that $C \leq 2KB$. If we had computed B with speeds rounded up rather than rounded down, then B is a lower bound on $C^*_{\max}$. Hence $C^*_{\max} \geq \frac{1}{2}B$. So we find that $C + \sum_{k=1}^{K} D_k \leq 14KC^*_{\max}$. It remains to note that $K \leq \log_2 m$, since $\bar{s}_K \geq \frac{1}{m}\bar{s}_1$ and $\bar{s}_{k+1} \leq \frac{1}{2}\bar{s}_k$ for $k = 1, \ldots, K-1$. □

Theorem 10. *There exists a randomized $O(\log m)$-approximation algorithm for $Q|prec, r_j|C_{\max}$ that is monotone in expectation.*

Proof: Shmoys, Wein and Williamson [7] give a general approach for converting a ρ-approximation algorithm for a scheduling problem without release dates into a 2ρ-approximation algorithm for the problem with release dates: the jobs are divided into blocks, where the first block contains the jobs for which $r_j = 0$, and the jobs in the k-th block are the jobs that are released during the time when the machines are processing the jobs in the $(k-1)$-st block. The jobs in a block are scheduled using the algorithm for the problem without release dates. It is easy to see that if the algorithm for the problem without release dates is monotone (in expectation) then so is the algorithm we thus obtain for the problem with release dates. □

Acknowledgements

The research reported in this paper was performed while the author was at Cornell University and the Institute for Theoretical Computer Science, Tsinghua University. It was supported in part by the National Natural Science Foundation of China Grant 60553001, and the National Basic Research Program of China Grant 2007CB807900,2007CB807901.

References

[1] A. Archer and É. Tardos. Truthful mechanisms for one-parameter agents. In *FOCS '01: 42nd IEEE Symposium on Foundations of Computer Science*, pages 482–491. IEEE Computer Soc., Los Alamitos, CA, 2001.

[2] C. Chekuri and M. Bender. An efficient approximation algorithm for minimizing makespan on uniformly related machines. *J. Algorithms*, 41(2):212–224, 2001. Preliminary version appeared in IPCO '98.

[3] F. A. Chudak and D. B. Shmoys. Approximation algorithms for precedence-constrained scheduling problems on parallel machines that run at different speeds. *J. Algorithms*, 30(2):323–343, 1999. Preliminary version appeared in SODA '97.

[4] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Ann. Discrete Math.*, 5:287–326, 1979. Discrete optimization (Proc. Adv. Res. Inst. Discrete Optimization and Systems Appl., Banff, Alta., 1977), II.

[5] J. M. Jaffe. Efficient scheduling of tasks without full use of processor resources. *Theoret. Comput. Sci.*, 12(1):1–17, 1980.

[6] S. O. Krumke, A. Schwahn, R. van Stee, and S. Westphal. A monotone approximation algorithm for scheduling with precedence constraints. *Oper. Res. Lett.*, 36(2):247–249, 2008.

[7] D. B. Shmoys, J. Wein, and D. P. Williamson. Scheduling parallel machines on-line. *SIAM J. Comput.*, 24(6):1313–1331, 1995. Preliminary version appeared in FOCS'91.

[8] C. Thielen and S. O. Krumke. A general scheme for designing monotone algorithms for scheduling problems with precedence constraints. In *WAOA '08: 6th International Workshop on Approximation and Online Algorithms*, pages 105–118, 2008.