An Asymptotic PTAS for a Special Case of Minimum Makespan Job Scheduling

Christopher Scarvelis and Prof. Adrian Vetta

August 16, 2019

1 Preliminaries

Minimum makespan job scheduling is a fundamental problem in theoretical computer science. Suppose that we have n jobs $j = 1, ..., n$ and m machines $i = 1, ..., m$. Each job must be assigned to exactly one machine; we call such an assignment of jobs to machines a *schedule*. If some job j is scheduled on machine i, the machine incurs a non-negative cost c_{ij} (or in an alternative interpretation, takes time c_{ij} to process the job).

Let $A_1, ..., A_m$ be a partition of the job set. We interpret A_i as the subset of jobs scheduled on machine i. We define the length of machine i's schedule to be

$$\sum_{j \in A_i} c_{ij}$$

and correspondingly define the *makespan* of the schedule defined by the partition $A_1, ..., A_m$ to be the following quantity:

$$\max_{i \in [m]} \sum_{j \in A_i} c_{ij}$$

where $[m]$ denotes the set $\{1, ..., m\}$. As the name of the problem suggests, the objective of minimum makespan job scheduling is to find a schedule that solves the following combinatorial optimization problem:

$$\min_{A_1, ..., A_m} \max_{i \in [m]} \sum_{j \in A_i} c_{ij}$$

Aside from the obvious interpretation of this problem as one of finding an optimal schedule for a set of jobs, minimizing makespan can be interpeted as the optimization of a fairness criterion in the sense that a minimum makespan schedule attempts to make each machine's schedule length as similar as possible.

This problem in its full generality is known to be APX-hard. In particular, it can be shown by reduction from three-dimensional matching that minimum makespan scheduling on unrelated machines (i.e. with arbitrary non-negative costs c_{ij}) admits no α-approximation algorithm for $\alpha < \frac{3}{2}$ unless P=NP.

In 1990, Lenstra, Shmoys and Tardos developed a 2-approximation algorithm based on rounding a solution to the LP relaxation of the integer program formulation of the scheduling problem. This result remains essentially the best known performance guarantee for an algorithm which solves this problem. (Shchepin and Vakhania have slightly improved this result to obtain a $(2 - \frac{1}{m})$-approximation algorithm that is structurally similar to the LST algorithm.)

However, there are special cases of the job scheduling problem for which substantially stronger results are known. In particular, there exists a polynomial-time approximation scheme (PTAS) for the special case of *identical machines*, where the cost of each job is the same for all machines, so

$$c_{i_1 j} = c_{i_2 j} \ \forall i_1, i_2 \in [m]$$

There also exists a PTAS for the case of *uniform machines*, where the cost vectors of any pair of machines i_1, i_2 are scalar multiples of each other.

Now suppose that we fix three positive parameters B, ϵ, δ and consider an instance I of minimum makespan job scheduling satisfying the following two assumptions:

1. $c_{i1} \geq \cdots \geq c_{in}$ for all machines $i \in [m]$. (**Order property**)
2. $\sum_{j=1}^{n} c_{ij} = B$ for all machines $i \in [m]$. (**Sum property**)

These two properties are motivated by applications to fair division problems. In particular, the sum property may be motivated by considering an instance of the problem where the machines represent agents who have private information regarding their costs. A planner must solicit costs from the agents, and the agents expect that the planner is unlikely to assign a given job to an agent that reports a high cost for that job. Clearly, each agent is incentivized to "cheat" by reporting arbitrarily high costs for all jobs. The sum property represents a natural way to exclude such a strategy - we can give each agent B points to allocate among the n jobs.

I claim that there exists an asymptotic PTAS for this problem. That is, I claim that the following theorem is true.

Theorem 1: *There exists an algorithm A that runs in time polynomial in n and m and returns a schedule whose makespan OPT_A satisfies:*

$$OPT_A \leq (1+\delta)OPT + \epsilon$$

2 The Algorithm

The basic approach is to divide the job set into "big" jobs and "small" jobs and then operate in two distinct phases. In the first phase, we enumerate all possible assignments of big jobs to machines - one of these corresponds to the assignment of big jobs in the optimal schedule for all jobs. In the second phase, we assign the small jobs using the LP-based method of Lenstra, Shmoys and Tardos. The main issue is that it is not clear *a priori* that the enumeration of schedules for big jobs can be done in polynomial time. It turns out that the order and sum properties together ensure that the number of big jobs is bounded above by a constant, which implies that there is at most a polynomial number of schedules for big jobs. The details are as follows.

We call a job j *big* if there exists some machine $i \in [m]$ such that $c_{ij} > \epsilon$. Otherwise, a job is said to be *small.* In particular, a small job j satisfies $c_{ij} \leq \epsilon$ for all machines i. We first prove a Markov-type inequality that supplies a constant upper bound on the number of jobs j satisfying $c_{ij} > \epsilon$ for each machine i.

Lemma 1: *For any machine $i \in [m]$, we have that $|j \in [n] : c_{ij} > \epsilon| \leq \frac{B}{\epsilon}$.*

Proof: We may decompose the sum of the job costs on machine i as follows:

$$\sum_{j=1}^{n} c_{ij} = \sum_{j: c_{ij} \leq \epsilon} c_{ij} + \sum_{j: c_{ij} > \epsilon} c_{ij} = B$$

As costs are non-negative, it follows that

$$\sum_{j: c_{ij} \leq \epsilon} c_{ij} \geq 0$$

and hence

$$\sum_{j: c_{ij} > \epsilon} c_{ij} = B - \sum_{j: c_{ij} \leq \epsilon} c_{ij} \leq B$$

But by construction, $c_{ij} > \epsilon$ for all terms in the sum on the left-hand side above, so we further have

$$\sum_{j: c_{ij} > \epsilon} c_{ij} > \sum_{j: c_{ij} > \epsilon} \epsilon = |j \in [n] : c_{ij} > \epsilon| \cdot \epsilon$$

We conclude that

$$|j \in [n] : c_{ij} > \epsilon| \cdot \epsilon \leq B$$

which directly implies that

$$|j \in [n] : c_{ij} > \epsilon| \leq \frac{B}{\epsilon}$$

since $\epsilon > 0$. ■

Hence the number of jobs that are big on some given machine i is bounded above by a constant. However, this does not exclude the possibility that a different subset of at most $\frac{B}{\epsilon}$ jobs is big on each machine - in fact, it may still be that *all* jobs are big on some machine. Fortunately, the order property ensures that this cannot be the case. In particular, we can strengthen Lemma 1 to obtain a bound on the number of jobs that are big on some machine. (This was our initial definition of a big job.)

Lemma 2: *The following bound holds:* $|j \in [n] : \exists i \in [m]$ *s.t.* $c_{ij} > \epsilon| \leq \frac{B}{\epsilon}$.

Proof: Let machine i^* have the most jobs with cost $> \epsilon$ and suppose that the number of such jobs on this machine is j^*. By Lemma 1, we have that $j^* = |j \in [n] : c_{i^*j} > \epsilon| \leq \frac{B}{\epsilon}$, so by the order property, $c_{i^*1} \geq \cdots \geq c_{i^*j^*} > \epsilon$ and $c_{i^*j} \leq \epsilon$ for all $j > j^*$. Now consider some other machine $\hat{\imath}$ and suppose that there are $\hat{\jmath} \leq j^*$ jobs with cost $> \epsilon$ on $\hat{\imath}$. Again by the order property, $c_{\hat{\imath}1} \geq \cdots \geq c_{\hat{\imath}\hat{\jmath}} > \epsilon$ and $c_{\hat{\imath}j} \leq \epsilon$ for all $j > \hat{\jmath}$. As $\hat{\jmath} \leq j^*$ (since j^* is the maximum number of jobs with cost $> \epsilon$ for any machine), it follows that $c_{\hat{\imath}j} \leq \epsilon$ for all $j > j^* \geq \hat{\jmath}$. This implies that for any machine i, $c_{ij} \leq \epsilon$ for all $j > j^*$. In particular, $\{j \in [n] : \exists i \in [m] \text{ s.t. } c_{ij} > \epsilon\} = \{1, ..., j^*\}$ and so

$$|j \in [n] : \exists i \in [m] \text{ s.t. } c_{ij} > \epsilon| = j^* \leq \frac{B}{\epsilon}$$

which completes the proof of the lemma. ■

Lemma 2 holds essentially because the ordering property implies that the set of big jobs on any machine i is a subset of the set of big jobs on machine i^*, where i^* is the machine with the most jobs with cost $> \epsilon$.

We have shown that there are at most $\frac{B}{\epsilon}$ big jobs in any instance of the job scheduling problem satisfying the order and sum properties. As each of these jobs must be assigned to one of m machines, there are $O(m^{\frac{B}{\epsilon}})$ (i.e. polynomially many) possible assignments of big jobs to machines. These can clearly be enumerated in polynomial time, and one of them corresponds to the the assignment of big jobs in the optimal schedule for all jobs.

For each assignment of big jobs, we use the method of Lenstra, Shmoys and Tardos to assign the remaining jobs. In particular, let $S = \{j : c_{ij} \leq \epsilon \text{ for all } i \in [m]\}$. Then $S = [n] \setminus \{j \in [n] : \exists i \in [m] \text{ s.t. } c_{ij} > \epsilon\}$. For each machine i, let d_i be the total cost of the big jobs scheduled on i in the assignment currently being considered. Fix some target $T \geq 0$ and consider the following integer program, which we will denote IP(T):

$$\sum_{i=1}^{m} x_{ij} = 1 \ \forall j \in S$$

$$\sum_{j \in S} x_{ij} c_{ij} \leq T - d_i \ \forall i \in [m]$$

$$x_{ij} \in \{0, 1\} \ \forall i \in [m], j \in S$$

It is easily seen that feasible solutions to IP(T) are in bijection with schedules for the small jobs such that the combined schedule for big and small jobs has makespan at most T. In particular, if $T \geq OPT$ and we have chosen the correct assignment of big jobs, then IP(T) must be feasible since the assignment of small jobs in the optimal schedule solves IP(T).

Now consider the following LP relaxation of IP(T), which we will denote LP(T):

$$\sum_{i=1}^{m} x_{ij} = 1 \ \forall j \in S$$

$$\sum_{j \in S} x_{ij} c_{ij} \leq T - d_i \ \forall i \in [m]$$

$$x_{ij} \geq 0 \ \forall i \in [m], j \in S$$

This LP is also feasible for $T \geq OPT$ since we have seen that IP(T) is feasible for such values of T.

Now note that $\frac{B}{n}$ is a valid lower bound for OPT since we must have $c_{i1} \geq \frac{B}{n}$ for all machines i (since otherwise we'd have $c_{ij} < \frac{B}{n}$ for all jobs j and the sum property couldn't possibly hold). Furthermore, B is clearly a valid upper bound for OPT. Hence we may carry out a binary search on the interval $[\frac{B}{n}, B]$ to approximately find the smallest value of T such that LP(T) is feasible. In particular, initially set $T = \frac{1}{2}(\frac{B}{n} + B)$ and determine if LP(T) is feasible. If yes, then shrink the search interval to $[\frac{B}{n}, \frac{1}{2}(\frac{B}{n} + B)]$ and carry out another iteration. Otherwise, shrink the search interval to $[\frac{1}{2}(\frac{B}{n} + B), B]$ and carry out another iteration. Terminate the binary search when the width of the interval is less than $\delta \cdot \frac{B}{n}$ - this occurs within $\log_2\left(\frac{n-1}{\delta}\right)$ iterations. Define T^+ to be the upper bound of the search interval after the final iteration and T^* to be the smallest value of T such that LP(T) is feasible. Note that $T^* \leq OPT$ and $T^+ \leq (1+\delta)T^*$, so $T^+ \leq (1+\delta)OPT$ as well.

A feasible solution to $LP(T^+)$ does not necessarily represent a valid schedule since the variables x_{ij} may be fractional; that is, the LP may attempt to divide a job among several machines. To circumvent this issue, we make use of a rounding theorem due to Lenstra, Shmoys and Tardos. I state it without proof - the proof may be found in "Approximation Algorithms for Scheduling Unrelated Parallel Machines" by Lenstra, Shmoys and Tardos or in standard textbooks on approximation algorithms such as Vazirani's book.

Theorem 2: *Define $J_i(\epsilon) = \{j : c_{ij} \leq \epsilon\}$ and $M_j(\epsilon) = \{i : c_{ij} \leq \epsilon\}$. If the following LP has a feasible solution:*

$$\sum_{i \in M_j(\epsilon)} x_{ij} = 1 \; \forall j \in [n]$$

$$\sum_{j \in J_i(\epsilon)} x_{ij} c_{ij} \leq \alpha_i \; \forall i \in [m]$$

$$x_{ij} \geq 0 \; \forall j \in J_i(\epsilon), i \in [m]$$

then any extreme point solution of the LP can be rounded to a feasible solution of the following IP in polynomial time:

$$\sum_{i \in M_j(\epsilon)} x_{ij} = 1 \; \forall j \in [n]$$

$$\sum_{j \in J_i(\epsilon)} x_{ij} c_{ij} \leq \alpha_i + \epsilon \; \forall i \in [m]$$

$$x_{ij} \in \{0,1\} \; \forall j \in J_i(\epsilon), i \in [m]$$

Let $\alpha_i = T^+ - d_i$ and note that $J_i(\epsilon) = S$ for all machines i and $M_j(\epsilon) = [m]$ for all jobs $j \in S$. Theorem 2 then states that a feasible solution to LP(T^+) may be rounded in polynomial time to a feasible solution to the following IP:

$$\sum_{i=1}^{m} x_{ij} = 1 \; \forall j \in S$$

$$\sum_{j \in S} x_{ij} c_{ij} \leq T^+ - d_i + \epsilon \; \forall i \in [m]$$

$$x_{ij} \in \{0,1\} \; \forall i \in [m], j \in S$$

But a feasible solution to this IP combined with the assignment of big jobs used in the optimal schedule corresponds to a schedule for all jobs with makespan at most $T^+ + \epsilon \leq (1+\delta)OPT + \epsilon$. It's easy to see that this algorithm runs in polynomial time, so we have proven Theorem 1 by demonstrating an algorithm that runs in polynomial time and computes a schedule of makespan at most $(1+\delta)OPT + \epsilon$.