Improved Approximations for General Minimum Cost Scheduling

Piotr Berman [*]   Marek Karpinski [†]

Abstract. We give improved trade-off results on approximating general *minimum cost scheduling* problems.

[*]Dept. of Computer Science and Engineering, The Pennsylvania State University, University Park, PA 16802. Partially supported by NSF grant CCR-9700053. Email: `berman@cse.psu.edu`.

[†]Dept. of Computer Science, University of Bonn, 53117 Bonn. Supported in part by DFG grants, DIMACS, and IST grant 14036 (RAND-APX). Email: `marek@cs.uni-bonn.de`.

1 Introduction

There exists a number of natural optimization problems related to scheduling that are difficult to approximate.

In recent years, two techniques offered polynomial time algorithms with improved approximation ratios. For some problems these were the first constant factor aproximations. The first technique was introduced by Bar-Noy *et al.* [BGNS99]; the problem is first described as an integer program, then one gets a fractional solution to linear relaxation of this program, and afterwards the results are converted to integers using a new method. One problem with this approach is that it the running time, while polynomial, is rather large. The second technique was simultaneously presented by Bar-Noy *et al.* [BBFNS00] Berman and DasGupta [BD00] . The latter technique is combinatorial and besides being more efficient, it handles well the case when time moments (release times, deadlines) are expressed with large numbers.

Phillips *et al.* [PUW00] presented an extension of the technique of [BGNS99] to two scheduling problems, and one of them was to find a trade-off between the cost and the completion rate in a variety of scheduling problems where the cost has to be minimized. This problem encapsulates many minimization problems for non-preemptive scheduling, for one or more machines, like minimization of the flow time, average completion time, total tardiness, etc.

Because a solution of our paper does not require the cost function to be increasing (in the sense that the later we schedule a particular job, the larger the cost), we can handle the case when many unrelated machines are available. As a result, we can handle some industrial engineering applications, where "machines" are either various machines in single factory or different factories, and where the cost function can be quite arbitrary.

To summarize, our improvements are the following: the new algorithm is an adaptation of the combinatorial approach of [BD00], and thus it is faster and can be adapted to handle time expressed with large numbers (i.e. it offers a polynomial time algorithm rather than pseudo-polynomial); the trade-off obtained is more favorable; limitations on the cost function are removed, which, among others, generalizes the technique to the cases with many machines.

2 Trade-Off for the General Minimum Cost Scheduling

We discuss here the general problem of *minimum cost scheduling*. This problem encapsulates many minimization problems for non-preemptive scheduling, for one or more machines, like minimization of the flow time, average completion time, total tardiness, etc. For the general background of this problem, see, e.g., [PUW00]. Because a solution offered in this paper does not require a cost function to be increasing (*e.g.,* executing a particular job *later* may cost *less*, we can extend our result to the case of many machines (*e.g.,* we can pretend that the job can be scheduled on one machine only, but this machine will run on many days and the cost of executing a job depends only on the hour, not on the day when it is performed).

To present the problem formally, our input describes a set of n jobs $\mathcal{J}$, each job $i \in \mathcal{J}$ has a set of intervals when it can be executed and the weight w_i. For simplicity, assume that $\sum_{i\in\mathcal{J}} w(i) = 1$. An *scheduling entry* a has job $j(a)$, start $s(a)$ and finish $f(a)$ where job $j(a)$ can be executed in the interval $[s(a), f(a))$. Moreover, the scheduling entry a has its *unit cost* $c(a)$ and weight $w(a) = w_{j(a)}$. A *schedule* is a set S of scheduling entries such that

- for each $i \in \mathcal{J}$ there exists at most one $a \in S$ such that $j(a) = i$;
- for each t there exists at most one $a \in S$ such that $s(a) \leq t < f(a)$.

The weight of a schedule S is $w(S) = \sum_{a\in S} w(a)$ and the cost is $cost(S) = \sum_{a\in S} c(a)w(a)$. We assume that there exists a schedule S^* such that $w(S^*) = 1$ and $cost(S^*) = C^*$. Intuitively, C^* is the minimal cost of a *complete* schedule.

The natural problem to consider in this context is to find a minimum cost complete schedule. However, this problem is known to be NP-hard. Moreover, if we insist on finding a complete schedule, no polynomial time algorithm can approximate the minimal cost of a complete schedule with an approximation factor $o(\sqrt{n})$, as it was shown by Kellerer *et al.* [KTW95]. Phillips *et al.* [PUW00] investigated the following trade-off: for a fraction φ we want to find a schedule of weight at least φ with as low cost as possible. The state of the art is such that for $\varphi > \frac{1}{2}$ we cannot find any schedule of weight φ, and for $\varphi = \frac{1}{2}$ we can barely manage to find a schedule, so we would rather not think about minimizing its cost. Thus we consider only

$\varphi < \frac{1}{2}$. An algorithm for this problem can be characterized by its *guarantee function.*

Formally, we say that the general minimum cost scheduling is solved with a guarantee $g(\varphi)$, if a polynomial time algorithm finds, given an instance of the problem and a fraction $\varphi < \frac{1}{2}$, a schedule with weight at least φ and cost at most $C^*g(\varphi)$.

Phillips *et al.* [PUW00] use linear relaxation of the problem to obtain a guarantee function $g(\varphi) = \frac{1-\varphi}{1-2\varphi}$. Their discussion explicitly handles the case when all jobs have equal weights and it generalizes the method to the arbitrarily weighted jobs.

We will describe a more direct approach that will yield a smaller guarantee function. We run our algorithm with parameter C that estimates C^*.

In our algorithm, we define for each scheduling entry a the *unit profit* $p_C(a) = C - c(a)(1-2\varphi)$; the profit of this entry is $p_C(a)w(a)$ so that we can define $profit_C(S) = \sum_{a \in S} p_C(a)w(a) = Cw(S) - cost(S)(1-2\varphi)$. Clearly, $profit_C(S^*) = C - C^*(1-2\varphi)$.

Bar-Noy *et al.* [BBFNS00] as well as Berman and DasGupta [BD00] described efficient algorithms that guarantee to find an approximation S of S^* such that $profit_C(S) \geq \frac{1}{2} profit_C(S^*)$. We use S_C to denote the outcome of such an algorithm. We consider two cases.

Case 1: $profit_C(S_C) \leq C\varphi$. Then

$$\begin{aligned} profit_C(S^*) \leq 2C\varphi \equiv C - C^*(1-2\varphi) \leq 2C\varphi &\equiv \\ C(1-2\varphi) \leq C^*(1-2\varphi) &\equiv C \leq C^*. \end{aligned}$$

Case 2: $profit_C(S_C) \geq C\varphi$. Then $Cw(S_C) - cost(S_C)(1-2\varphi) \geq C\varphi$ which implies

$$w(S_C) \geq \varphi + \frac{1-2\varphi}{C} cost(S_C) \text{ and} \tag{1}$$

$$cost(S_C) \leq C\frac{w(S_C)-\varphi}{1-2\varphi} \leq C\frac{1-\varphi}{1-2\varphi}. \tag{2}$$

In this case we know that the weight of S_C is large enough. Moreover, if $C \leq C^*(1+\varepsilon)$, then

$$cost(S_C) \leq C^* \tfrac{1-\varphi}{1-2\varphi}(1+\varepsilon)$$

This case analysis shows that if $C = C_0/(1+\varepsilon)$ satisfies Case 1 and $C = C_0$ satisfies Case 2, then $w(S_{C_0})$ is large enough and $cost(S_{C_0})$ exceeds the guarantee of Phillips *et al.* [PUW00] by a factor not larger than $1+\varepsilon$. Clearly, for any ε we can find an appropriate C_0 using Newton iteration. We start from a low estimate for C being 0 and a high estimate being $\sum_{i \in \mathcal{J}} \max_{a:j(a)=i} c(a)w(a)$. If the average of the two estimates satisfies Case 1, it becomes the new low estimate, if it satisfies Case 2, it becomes the new high estimate. We stop when the difference between the estimates drops below ε.

While we already obtained a solution that satisfies the guarantee of Phillips *et al.*, we can observe that our actual guarantee is a bit stronger. To simplify the reasoning, assume that $\varepsilon = 0$ and thus $C = C^*$. We will describe two algorithms, with smaller guarantee functions.

We will use δ to denote the maximum job weight. If $\delta \geq \varepsilon$, both of our algorithm return schedule $\{a\}$ where a is a scheduling entry with $w(a) = \delta$ and the minimum cost. Clearly, $cost(\{a\}) \leq C^*$.

The design of our first algorithm starts with the observation that in (2) the cost of S_C is equal to $C\frac{1-\varphi}{1-2\varphi}$ only if S_C includes all the jobs, *i.e.* $w(S_C) = 1$. On the other hand, if $w(S_C)$ is just as we promised, *i.e.* equal to φ, then $cost(S_C) = 0$. In the latter case we are clearly obtaining a lower cost than we initially wanted to guarantee. If $w(S_C) > \varphi$, we will try to compute another solution S. We start with $S = S_C$. Then we can pick an entry $a \in S$ with the maximum unit cost and remove it. As a result, the average unit cost will not go up. We can repeat this until $w(S) - w(a) < \varphi$, clearly we end up with $w(S) \leq \varphi + \delta$.

The above reasoning shows that we start with the cost of $C\frac{w(S_C)-\varphi}{1-2\varphi}$ and we decrease it by the factor of $(\varphi+\delta)/w(S_C)$ to obtain

$$\frac{\varphi+\delta}{w(S_C)} \times C\frac{w(S_C)-\varphi}{1-2\varphi} = C\frac{(\varphi+\delta)(1-\varphi/w(S_C))}{1-2\varphi} \leq C\frac{(\varphi+\delta)(1-\varphi)}{1-2\varphi}.$$

Thus we have proven the following theorem.

Theorem 1. *We can solve in polynomial time the general minimum cost scheduling problem with the performance guarantee* $\frac{(\varphi+\delta)(1-\varphi)}{1-2\varphi}$ *if* $\delta < \varphi$ *and* 1 *if* $\delta \geq \varphi$.

Using a slower algorithm, we can provide a performance guarantee that does not depend on δ, more precisely, the guarantee of $\max(\frac{\varphi(1-\varphi)}{1-2\varphi}, 1)$. Note that $\frac{\varphi(1-\varphi)}{1-2\varphi} \geq 1$ if $\varphi \geq \frac{3-\sqrt{5}}{2}$.

We consider only the case when $\delta < \varphi$; we will assume that $\delta = w_i$.

Our new algorithm will consider every possible scheduling entry $\mathbf{a}$ for job i. For a given $\mathbf{a}$, we remove from consideration all scheduling entries that are in conflict (schedule job i or have interval that overlaps the interval of $\mathbf{a}$). We find the schedule that consists of $\mathbf{a}$ and the solution of the following residual problem:

all weights are divided by $1-\delta$ so they add to 1,
the target fraction of weight is $(\varphi-\delta)/(1-\delta)$,
the optimum cost is $C^* - \mathbf{c}$, where $\mathbf{c} = c(\mathbf{a})\delta$,
the maximum job weight is $\delta/(1-\delta)$.

By Theorem 1, the resulting schedule has the cost at most

$$\mathbf{c} + \frac{\left(\frac{\varphi-\delta}{1-\delta}+\frac{\delta}{1-\delta}\right)\left(1-\frac{\varphi-\delta}{1-\delta}\right)}{\left(1-2\frac{\varphi-\delta}{1-\delta}\right)}(C^*-\mathbf{c}) = \mathbf{c} + \frac{\varphi(1-\varphi)}{(1-2\varphi+\delta)(1-\delta)}(C^*-\mathbf{c}) =$$
$$\mathbf{c} + \frac{\varphi(1-\varphi)}{1-2\varphi+\delta(2\varphi-\delta)}(C^*-\mathbf{c}) \le \mathbf{c} + \frac{\varphi(1-\varphi)}{1-2\varphi}(C^*-\mathbf{c}) \le$$
$$\max(1, \frac{\varphi(1-\varphi)}{1-2\varphi})\mathbf{c} + \max(1, \frac{\varphi(1-\varphi)}{1-2\varphi})(C^*-\mathbf{c}) = \max(1, \frac{\varphi(1-\varphi)}{1-2\varphi})C^*$$

This entails the following theorem.

Theorem 2. *We can solve in polynomial time the general minimum cost scheduling problem with the performance guarantee* $\max(1, \frac{\varphi(1-\varphi)}{1-2\varphi})$.

□

References

[BGNS99] A. Bar-Noy, S. Guha, J. Naor and B. Schieber, *Approximating the Throughput of Real-Time Multiple Machine Scheduling*, 31st STOC, 1999.

[BBFNS00] A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. Naor and B. Schieber, *A Unified Approach to Approximating Resource Allocation and Scheduling*, 32nd STOC, 2000.

[BD00] P. Berman and B. DasGupta, *Improvements in Throughput Maximization for Real-Time Scheduling*, 32nd STOC, 2000.

[KTW95] H. Kellerer, T. Tautenhahn and G.J. Woeginger, *Approximability and Nonapproximability Results for Minimizing Total Flow Time on a Single Machine*, 28th STOC, 1995.

[PUW00] C. Phillips, R.N. Uma and J. Wein, *Off-Line Admission Control for General Scheduling Problems*, 11th SODA, ACM and SIAM, 2000.