Discrete Applied Mathematics 195 (2015) 84–100

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

Performance guarantees for scheduling algorithms under perturbed machine speeds☆

Michael Etscheid

Department of Computer Science, University of Bonn, Germany

ARTICLE INFO

Article history:
Received 30 September 2013
Received in revised form 15 April 2014
Accepted 23 May 2014
Available online 25 June 2014

Keywords:
Smoothed analysis
Scheduling
Performance guarantees
Local search

ABSTRACT

We study two local search algorithms and a greedy algorithm for scheduling. The worst-case performance guarantees are well-known but seem to be contrived and too pessimistic for practical applications. For unrestricted machines, Brunsch et al. (2013) showed that the worst-case performance guarantees of these algorithms are not robust if the job sizes are subject to random noise. However, in the case of restricted related machines the worst-case bounds turned out to be robust even in the presence of random noise. We show that if the machine speeds rather than the job sizes are perturbed, one obtains smaller bounds for the performance guarantees also for restricted machines thus yielding a stronger result.

1. Introduction

For many simple scheduling algorithms, the worst-case performance guarantees are known up to a constant factor. However, the instances used to construct lower bounds seem to be artificial and not practically relevant if there is some noise on the input. Therefore, we use the framework of smoothed analysis to identify worst-case bounds which are too pessimistic with high probability if the input is perturbed. We see this as a first step in explaining why the worst case is rarely observed in practice because instances that arise in practice are often subject to some noise coming, e.g., from measurement errors, numerical imprecision, rounding errors, etc. The noise can also model influences that cannot be quantified exactly but for which there is no reason to believe that they are adversarial.

In this section, we define the scheduling problem, introduce the framework of smoothed analysis and compare our results with the worst-case bounds and the bounds given in [3]. In Sections 2 and 3, we provide the proofs for the settings of unrestricted and restricted machines, respectively.

1.1. The scheduling problem

Let $J = \{1, \dots, n\}$ be the set of jobs and $M = \{1, \dots, m\}$ be the set of machines on which the jobs shall be processed. Each machine $i \in M$ has a speed s_i and each job $j \in J$ has a processing requirement p_j. The speeds of the fastest and the slowest machine are denoted by $s_{\max}$ and $s_{\min}$, respectively. We consider two different environments: in the case of unrestricted machines, each job is allowed to run on any machine. In the case of restricted machines, each job $j \in J$ has a set $\mathcal{M}_j \subseteq M$ of allowed machines. These variables form an instance I of the scheduling problem.

In this paper, we define a *schedule* to be an assignment $\sigma : J \to M$. The time a machine i needs to process job j is p_j/s_i if job j is allowed to run on machine i, and ∞ otherwise. Given a schedule σ for an instance I, the *load* of a machine i is

☆ This research was supported by ERC Starting Grant 306465 (BeyondWorstCase).
E-mail address: etscheid@cs.uni-bonn.de.

http://dx.doi.org/10.1016/j.dam.2014.05.041
0166-218X/

Table 1
Performance guarantees for unrestricted machines.

Algorithm	Worst-case	Perturbed job sizes	Perturbed speeds
Jump	$\Theta(\sqrt{m})$ [4,12]	$\Theta(\phi)$ [3]	$\Theta(\phi)$
lex-jump	$\Theta\left(\min\left\{\frac{\log m}{\log\log m}, \log\frac{s_{\max}}{s_{\min}}\right\}\right)$ [5]	$\Theta(\log\phi)$ [3]	$\Theta(\log\phi)$
List	$\Theta(\log m)$ [1,4]	$\Theta(\log\phi)$ [3]	$\Theta(\log\phi)$

Table 2
Performance guarantees for restricted machines. Here, $S = \sum_{i=1}^{m} \frac{s_i}{s_{\min}}$.

Algorithm	Worst-case	Perturbed job sizes	Perturbed machine speeds
Jump	$\Theta\left(\sqrt{m \cdot \frac{s_{\max}}{s_{\min}}}\right)$ [11]	$\Theta\left(\sqrt{m \cdot \frac{s_{\max}}{s_{\min}}}\right)$ [3]	$\Theta\left(m\sqrt{\phi}\right)$
lex-jump	$\Theta\left(\frac{\log S}{\log\log S}\right)$ [11]	$\Omega\left(\frac{\log m}{\log\log m}\right)$ [3]	$\Theta\left(\min\left\{m, \frac{\log(m\phi)}{\log\log(m\phi)}\right\}\right)$

defined as $L_i(I,\sigma) = \sum_{j\in\sigma^{-1}(i)} p_j/s_i$. The *makespan* $C_{\max}(I,\sigma)$ of σ, i.e., the maximum job-completion time, can be written as $C_{\max}(I,\sigma) = \max_{i\in M} L_i(I,\sigma)$. We write $C^*_{\max}(I)$ for an optimal makespan. The goal is to minimize the makespan. Sometimes we omit the parameters I and σ, respectively, if they are clear from the context.

1.2. Studied scheduling algorithms

We study a greedy and two local search algorithms. We assume that all jobs assigned to a machine finish at the same time, which is the load of the machine.

The *list scheduling* algorithm starts with an empty schedule. Then it iteratively assigns an unscheduled job to the machine on which it will be completed first with respect to the current partial schedule. Any schedule which can be generated this way is called a *list schedule*.

The *jump* and *lex-jump* algorithms start with an arbitrary schedule and then perform local improvement steps. In each step, a job is reassigned to a different machine where it finishes earlier. In the jump algorithm, only jobs assigned to a *critical* machine, i.e., a machine with maximal load, are allowed to be reassigned whereas the lex-jump algorithm does not have this limitation. A schedule which cannot be improved by the (lex-)jump algorithm is called *(lex-)jump optimal*.

We write $\mathrm{Jump}(I)$ for the set of all jump optimal schedules, $\mathrm{Lex}(I)$ for the set of all lex-jump optimal schedules, and $\mathrm{List}(I)$ for the set of all list schedules for a scheduling instance I.

1.3. Smoothed analysis

The framework of smoothed analysis was introduced by Spielman and Teng [14] to explain the good running time of some algorithms in practice despite a bad worst-case running time. We use the more general model suggested by Beier and Vöcking [2]. Let $\phi \geq 1$ be a parameter for the maximum probability density. A *ϕ-smooth* instance $\mathcal{I}$ consists of job sizes $p_1, \ldots, p_n$, subsets $\mathcal{M}_j \subseteq M$, $j \in J$, in the case of restricted machines, the number m of machines, and density functions $f_i\colon [0,1] \to [0,\phi]$ for all machines $i \in \{1,\ldots,m\}$. Each machine speed s_i is then chosen according to the density function f_i independently of the other machine speeds. Thus, any ϕ-smooth instance is a distribution over infinitely many scheduling instances. We then look for ϕ-smooth instances for which the *expected* performance ratio is as bad as possible. For example, we can choose for every s_i an interval of length $1/\phi$ from which it is drawn uniformly at random. For $\phi = 1$, this model complies with an average case analysis, whereas for $\phi \to \infty$ the smoothed analysis tends to a worst-case analysis as the machine speeds can be specified within arbitrary precision.

1.4. Related work and results

Minimizing the makespan in a scheduling instance is a well-known strongly NP-hard problem. There is a polynomial approximation scheme for the unrestricted case by Hochbaum and Shmoys [7] as well as a 2-approximation algorithm for restricted machines by Lenstra et al. [9]. For the algorithms studied in this paper, Table 1 shows an overview of the worst-case and smoothed performance guarantees in the environment of unrestricted machines. We were able to reproduce the same results as Brunsch et al. [3] with perturbed machine speeds instead of perturbed job sizes. Accordingly, we get the same conclusions that the lex-jump algorithm and the list jump algorithm should perform well in practice. An interesting deduction of theirs is that the smoothed price of anarchy for routing games on parallel links is $\Theta(\log\phi)$ as well, as pure Nash equilibria can be seen as local optima according to the lex-jump algorithm. This result carries over to our smoothed model with perturbed link speeds.

As Table 2 shows, the worst-case performance guarantees in the environment of restricted machines are robust against random noise on the job sizes. We calculate the expected values of these worst-case bounds to obtain the smoothed bounds