ONLINE SCHEDULING WITH BOUNDED MIGRATION

(EXTENDED ABSTRACT FOR DAGSTUHL SEMINAR 05031 ON ALGORITHMS FOR OPTIMIZATION WITH INCOMPLETE INFORMATION)

PETER SANDERS, NAVEEN SIVADASAN, AND MARTIN SKUTELLA

ABSTRACT. Consider the classical online scheduling problem where jobs that arrive one by one are assigned to identical parallel machines with the objective of minimizing the makespan. We generalize this problem by allowing the current assignment to be changed whenever a new job arrives, subject to the constraint that the total size of moved jobs is bounded by β times the size of the arriving job.

Our main result is a linear time 'online approximation scheme', that is, a family of online algorithms with competitive ratio $1 + \epsilon$ and constant migration factor $\beta(\epsilon)$, for any fixed $\epsilon > 0$. This result is of particular importance if considered in the context of sensitivity analysis: While a newly arriving job may force a complete change of the entire structure of an optimal schedule, only very limited 'local' changes suffice to preserve near-optimal solutions. We believe that this concept will find wide application in its own right.

We also present simple deterministic online algorithms with migration factors $\beta = 2$ and $\beta = 4/3$, respectively. Their competitive ratio $3/2$ beats the lower bound on the performance of any online algorithm in the classical setting without migration. We also present improved algorithms and similar results for closely related problems. In particular, there is a short discussion of corresponding results for the objective to maximize the minimum load of a machine. The latter problem has an application for configuring storage servers that was the original motivation for this work.

1. INTRODUCTION

A classical scheduling problem. One of the most fundamental scheduling problems asks for an assignment of jobs to m identical parallel machines so as to minimize the makespan. (The makespan is the completion time of the last job that finishes in the schedule; it also equals the maximum machine load.) In the standard classification scheme of Graham, Lawler, Lenstra, & Rinnooy Kan [13], this scheduling problem is denoted by $\mathrm{P}\,||\,C_{\max}$ and it is well known to be strongly NP-hard [10].

The *offline* variant of this problem assumes that all jobs are known in advance whereas in the *online* variant the jobs are incrementally revealed by an adversary and the online algorithm can only choose the machine for the new job without being allowed to move other jobs. Note that dropping this radical constraint on the online algorithm yields the offline situation.

A new online scheduling paradigm. We study a natural generalization of both offline and online problems. Jobs arrive incrementally but, upon arrival of a new job j, we are allowed to migrate *some* previous jobs to other machines. The total size of the migrated jobs however must be bounded by βp_j where p_j is the size of the new job. For *migration factor* $\beta = 0$ we get the online setting and for $\beta = \infty$ we get the offline setting.

Approximation algorithms. For an offline optimization problem, an *approximation algorithm* efficiently (in polynomial time) constructs schedules whose values are within a constant factor $\alpha \geq 1$ of the optimum solution value. The number α is called *performance guarantee* or *performance ratio* of the approximation algorithm. A family of polynomial time approximation algorithms with performance guarantee $1 + \epsilon$ for all fixed $\epsilon > 0$ is called a *polynomial time approximation scheme* (PTAS).

Competitive analysis. In a similar way, *competitive analysis* evaluates solutions computed in the online setting. An online algorithm achieves *competitive ratio* $\alpha \geq 1$ if it always maintains solutions whose objective values are within a factor α of the offline optimum. Here, in contrast to offline approximation results, the achievable values α are not determined by limited computing power but by the apparent lack of information about parts of the input that will only be revealed in the future. As a consequence, for all interesting classical online problems it is rather easy to come up with lower bounds that create a gap between the best possible competitive ratio α and 1. In particular, it is usually impossible to construct a family of $(1 + \epsilon)$-competitive online algorithms for such problems.

2. RELATED WORK

For the online machine scheduling problem, Graham's *list scheduling* algorithm keeps the makespan within a factor $2 - 1/m$ of the offline optimum [11]: Schedule a newly arriving job on the least loaded machine. It can also easily be seen that this bound is tight: adversarial sequence consists of $m(m-1)$ jobs of size $\frac{1}{m}$ followed by one job of size 1. The optimal makespan in this case is 1.

For the offline setting, Graham showed three years later that sorting the jobs in the order of non-increasing size before feeding them to the list scheduling algorithm yields an approximation algorithm with performance

Dagstuhl Seminar Proceedings 05031
Algorithms for Optimization with Incomplete Information
http://drops.dagstuhl.de/opus/volltexte/2005/70

ratio $4/3 - 1/(3m)$ [12]. Later, exploiting the relationship between the machine scheduling problem under consideration and the binpacking problem, algorithms with improved approximation ratios have been obtained in a series of works [7, 9, 17].

Finally, polynomial time approximation schemes for a constant number of machines and for an arbitrary number of machines are given in [12, 21] and by Hochbaum & Shmoys [15], respectively. The latter PTAS partitions jobs into large and small jobs. The sizes of large jobs are rounded such that an optimum schedule for the rounded jobs can be obtained via dynamic programming. The small jobs are then added greedily using Graham's list scheduling algorithm. This approach can be refined to an algorithm with linear running time (see, e.g., [14]): replace the dynamic program with an integer linear program on a fixed number of variables and constraints which can be solved in constant time [18].

In a series of papers, increasingly complicated online algorithms with better and better competitive ratios beating the Graham bound 2 have been developed [5, 16, 1]. The best result known to date is a 1.9201-competitive algorithm due to Fleischer and Wahl [8]. The best lower bound 1.88 on the competitive ratio of any deterministic online algorithm currently known is due to Rudin [20]. For randomized online algorithms there is a lower bound of $e/(e-1) \approx 1.58$ [6, 23]. For more results on online algorithms for scheduling we refer to the recent survey articles by Albers [2] and Sgall [24].

Strategies that reassign jobs were studied in the context of online load balancing, jobs arrive in and depart from a system of m machines online and the scheduler has to assign each incoming job to one of the machines. Deviating from the usual approach of comparing against the optimal *peak load* seen so far, Westbrook [25] introduced the notion of competitiveness against *current load*: An algorithm is α–competitive if after every round the makespan is within α factor of the optimal makespan for the current set of jobs. Each incoming job u has size p_u and reassignment cost r_u. For a job, the reassignment cost has to be paid for its initial assignment and then every time it is reassigned. Observe that the optimal strategy has to pay this cost once for each job for its initial assignment. Thus the optimal (re)assignment cost S is simply the sum of reassignment costs of all jobs scheduled till now. Westbrook showed a 6-competitive strategy for identical machines with reassignment cost $3S$ for proportional reassignments, i.e., r_u is proportional to p_u, and $2S$ for unit reassignments, i.e., $r_u = 1$ for all jobs. Later Andrews et al. [3] improved it to 3.5981 with the same reassignment factors. They also showed $3+\epsilon$ and $2+\epsilon$ competitive strategies respectively for the proportional and unit case, the reassignment factor depending only on ϵ. For arbitrary reassignment costs they achieve 3.5981 competitiveness with 6.8285 reassignment factor. They also present a 32–competitive strategy with constant reassignment factor for related machines. Job deletions is an aspect that we do not consider in our work, our focus is primarily on achieving competitive ratios close to 1. Our results can also be interpreted in this framework of online load balancing, with proportional reassignments and without job deletions. We show strategies with better competitive ratios, at the same time achieving reassignment factor strictly less than three. We also show $(1+\epsilon)$–competitive strategies, for any $\epsilon > 0$, with constant reassignment factor $f(\epsilon)$. Our results are also stronger in the sense that a strategy with reassignment factor β ensures that when a job u arrives, the total reassignment cost incurred (for scheduling it) is at most βr_u. This is different from the more relaxed constraint that after t rounds, the total reassignment cost incurred is at most $\beta \sum r_u$ (summing over all jobs seen till round t). Most of our strategies are *robust*, they convert *any* α–competitive schedule to an α–competitive schedule after assigning the newly arrived job, whereas in [25, 3] it is required that the schedule so far is carefully constructed in order to ensure the competitiveness after assigning/deleting a job in the next round.

3. Our Contribution

We describe a simple online algorithm which achieves approximation ratio $3/2$ using a moderate migration factor $\beta = 2$. Notice that already this result beats the lower bound 1.88 (1.58) on the competitive ratio of any classical (randomized) online algorithm without migration. Using a more sophisticated analysis, the migration factor can be decreased to $4/3$ while maintaining competitive ratio $3/2$. On the other hand we show that our approach does not allow for migration factor 1 and competitive ratio $3/2$. Furthermore, an improved competitive ratio $4/3$ can be achieved with migration factor 4. For two machines, we can achieve competitive ratio $7/6$ with a migration factor of one. This ratio is tight for migration factor one.

As our main result, we present a family of online algorithms with competitive ratio $1+\epsilon$ and constant migration factor $\beta(\epsilon)$, for any fixed $\epsilon > 0$. On the negative side, no constant migration factor suffices to maintain competitive ratio one, i.e., optimality. We provide interpretations of these results in several different contexts:

- *Online algorithms.* Online scheduling with bounded job migration is a relaxation of the classical online paradigm. Obviously, there is a tradeoff between the desire for high quality solutions and the requirement to compute them online, that is, to deal with a lack of information. Our result can be interpreted in terms of the corresponding tradeoff curve: Any desired quality can be guaranteed while relaxing the online paradigm only moderately by allowing for a constant migration factor.

- *Sensitivity analysis.* Given an optimum solution to an instance of an optimization problem and a slightly modified instance, can the given solution be turned into an optimum solution for the modified instance without changing the solution too much? This is the impelling question in sensitivity analysis. As indicated above, for the scheduling problem under consideration one has to answer in the negative. Already one additional job can change the entire structure of an optimum schedule. However, our result implies that the answer is positive if we only require near-optimum solutions.
- *Approximation results.* Our result yields a new PTAS for the scheduling problem under consideration. Due to its online background, this PTAS constructs the solution incrementally. That is, it reads the input little by little always maintaining a $(1+\epsilon)$-approximate solution. Indeed, it follows from the analysis of the algorithm that every update only takes constant time. In particular, the overall running time is linear and thus matches the previously best known approximation result.

We believe that each of these interpretations constitutes an interesting motivation for results like the one we present here in its own right and can therefore lead to interesting results for many other optimization problems.

The underlying details of the presented online approximation scheme have the same roots as the original PTAS by Hochbaum & Shmoys [15] and its refinements [14]. We distinguish between small and large jobs; a job is called large if its size is of the same order of magnitude as the optimum makespan. Since this optimum can change when a new job arrives, the classification of jobs must be updated dynamically. The size of every large job is rounded such that the problem of computing an optimum schedule for the subset of large jobs can be formulated as an integer linear program of constant size. A newly arriving job causes a small change in the right hand side of this program. This enables us to use results from sensitivity analysis of integer programs in order to prove that the schedule of large jobs needs to be changed only slightly. Our PTAS is very simple, it uses only this structural result and does not use any algorithms from integer programming theory. For a detailed account of linear and integer programming theory we refer to the books [22, 19].

An application of bounded migration occurs in the context of configuring storage servers. This was the original motivation for our work. In this application, the objective is to maximize the minimum load. It is well-known [4] that any online deterministic algorithm for this *machine covering problem* has competitive ratio at least m (the number of machines). There is also a lower bound of $\Omega(\sqrt{m})$ for any randomized online algorithm. We develop a simple deterministic online strategy which is 2-competitive already for migration factor $\beta = 1$.

REFERENCES

[1] S. Albers. Better bounds for online scheduling. *SIAM Journal on Computing*, 29:459–473, 1999.
[2] S. Albers. Online algorithms: a survey. *Mathematical Programming*, 97:3–26, 2003.
[3] M. Andrews, M.X. Goemans, and L. Zhang. Improved bounds for on-line load balancing. *Algorithmica*, 23:278–301, 1999.
[4] Y. Azar and L. Epstein. On-line machine covering. *Journal of Algorithms*, 1:67–77, 1998.
[5] Y. Bartal, A. Fiat, H. Karloff, and R. Vohra. New algorithms for an ancient scheduling problem. *Journal of Computer and System Sciences*, 51:359–366, 1995.
[6] B. Chen, A. van Vliet, and G. J. Woeginger. Lower bounds for randomized online scheduling. *Information Processing Letters*, 51:219–222, 1994.
[7] E. G. Coffman Jr., M. R. Garey, and D. S. Johnson. An application of bin-packing to multiprocessor scheduling. *SIAM Journal on Computing*, 7:1–17, 1978.
[8] R. Fleischer and M. Wahl. Online scheduling revisited. *Journal of Scheduling*, 3:343–353, 2000.
[9] D. K. Friesen. Tighter bounds for the multifit processor scheduling algorithm. *SIAM Journal on Computing*, 13:170–181, 1984.
[10] M. R. Garey and D. S. Johnson. Strong np-completeness results: Motivation, examples and implications. *Journal of the ACM*, 25:499–508, 1978.
[11] R. L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 45:1563–1581, 1966.
[12] R. L. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics*, 17:263–269, 1969.
[13] 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. *Annals of Discrete Mathematics*, 5:287–326, 1979.
[14] D. S. Hochbaum. Various notions of approximation: Good, better, best, and more. In D. S. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems*, chapter 9, pages 346–398. Thomson, 1996.
[15] D. S. Hochbaum and D. B. Shmoys. Using dual approximation algorithms for scheduling problems: Theoretical and practical results. *Journal of the ACM*, 34:144–162, 1987.
[16] D. R. Karger, S. J. Phillips, and E. Torng. A better algorithm for an ancient scheduling problem. *Journal of Algorithms*, 20:400–430, 1996.
[17] M. A. Langston. *Processor scheduling with improved heuristic algorithms*. PhD thesis, Texas A&M University, 1981.
[18] H. W. Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8:538–548, 1983.
[19] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, New York, 1988.
[20] J. F. Rudin III. *Improved bounds for the on-line scheduling problem*. PhD thesis, The University of Texas at Dallas, 2001.
[21] S. Sahni. Algorithms for scheduling independent tasks. *Journal of the ACM*, 23:116–127, 1976.
[22] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, Chichester, 1986.
[23] J. Sgall. A lower bound for randomized on-line multiprocessor scheduling. *Information Processing Letters*, 63(1):51–55, 14 July 1997.
[24] J. Sgall. On-line scheduling — a survey. In A. Fiat and G. J. Woeginger, editors, *Online Algorithms: The State of the Art*, volume 1442 of *Lecture Notes in Computer Science*, pages 196–231. Springer, Berlin, 1998.
[25] J. Westbrook. Load balancing for response time. *J. Algorithms*, 35(1):1–16, 2000.