Brief Announcement: A QPTAS for Non-preemptive Speed-scaling

Sungjin Im
EECS, University of California, Merced
Merced, CA
sim3@ucmerced.edu

Maryam Shadloo
EECS, University of California, Merced
Merced, CA
mshadloo@ucmerced.edu

ABSTRACT

Modern processors typically allow dynamic speed-scaling offering an effective trade-off between high throughput and energy efficiency. In a classical model, a processor/machine runs at speed s when consuming power s^α where $\alpha > 1$ is a constant. Yao et al. [FOCS 1995] studied the problem of completing all jobs before their deadlines on a single machine with the minimum energy in their seminal work and gave a nice polynomial time algorithm. The influential work has been extended to various settings. In particular, the problem has been extensively studied in the presence of multiple machines as multi-core processors have become dominant computing units.

However, when jobs must be scheduled *non-preemptively*, our understanding of the problem remains fairly unsatisfactory. Often, preempting a job is prohibited since it could be very costly. Previously, a $O((w_{\max}/w_{\min})^\alpha)$-approximation was known for the non-preemptive setting where $w_{\max}$ and $w_{\min}$ denote the maximum and minimum job sizes, respectively. Even when there is only one machine, the best known approximation factor had a dependency on α. In this paper, for any fixed $\alpha > 1$ and $\epsilon > 0$, we give the *first* $(1+\epsilon)$-approximation for this problem on multiple machines which runs in $n^{O(\text{polylog}(n))}$ time where n is the number of jobs to be scheduled.

1. INTRODUCTION

Energy efficiency is considered a primary goal in modern scheduling at various scales [1]. Energy dissipation is a main concern in mobile devices. Cumulative energy cost over a long time period can easily exceed the hardware procurement cost. At a larger scale, for example, in data centers, the importance of energy conservation is well illustrated by the following quote:

> What matters most to the computer designers at Google is not speed, but power, low power, because data centers can consume as much electricity as a city.
>
> - Dr. Eric Schmidt, CEO of Google [21].

Various methods have been developed to save energy while meeting/optimizing certain qualities of services [18, 1]. One of the most widely used technologies is dynamic speed-scaling where each individual machine can run in different speeds depending on the energy it consumes. A machine can run slowly when processing small workload, and run faster to process high volume of incoming jobs by consuming more power.

Yao et al. studied a fundamental scheduling problem on a single machine with dynamic speed-scaling capabilities in their seminal work [22]. In the problem, which we call the Energy-efficient Deadline Scheduling Problem (EDS), there are n jobs. Each job j has a release time r_j, deadline d_j, and work volume w_j that must be completed during its lifespan (r_j, d_j) on the given single machine. When the machine runs in speed s at time instant t, the machine consumes power $P(s) = s^\alpha$, where $\alpha > 1$ is a constant. A job j completes when it gets processed by w_j units. The goal is to complete all jobs during their respective lifespans while minimizing the total energy consumption. For EDS, Yao et al. gave an elegant polynomial time algorithm. The influential work has been extended to multiple machines [11, 2, 4, 7, 6].

However, all the aforementioned works assume that preemption is allowed for free, meaning that a job being processed can be interrupted to process other jobs and can be resumed later. While preemption is very useful for optimizing certain objectives, it is prohibited in some applications – preempting and resuming a job can be very costly, or may not be allowed due to the way the system works. Unfortunately, non-preemptive scheduling is fundamentally different from preemptive scheduling and gives a lot of algorithmic challenges. For example, adding a job j of small lifespan may change the schedule drastically by forcing other jobs to either complete before j starts or to start after j completes.

Indeed, the EDS problem becomes strongly NP-hard without preemption [5] – we will call the EDS problem with no preemption EDS-N. The best approximation known for EDS-N is $(1+\epsilon)\tilde{B}_\alpha$ due to [8]. Here $\tilde{B}_\alpha := \sum_{k=0}^{\infty} \frac{k^\alpha/e}{k!}$ is the generalized Bell number which grows exponentially in α. A key idea of the $(1+\epsilon)\tilde{B}_\alpha$-approximation was to forbid certain 'enclosing' using a clever configuration LP and rounding scheme. That is, each job j is allowed to start at time a_j and end at time b_j only when the interval (a_j, b_j) includes no other jobs lifespans. However, this constraint is no longer valid when there are multiple machines, rendering the configuration LP inapplicable. Prior to our work

SPAA '16 July 11-13, 2016, Pacific Grove, CA, USA

ACM ISBN 978-1-4503-4210-0/16/07.

DOI: http://dx.doi.org/10.1145/2935764.2935824

the best approximation factor of any polynomial or quasi-polynomial time algorithms known for EDS-N in the multiple machines setting was $\Omega((w_{\max}/w_{\min})^{\alpha})$ [15, 8], where $w_{\max}$ and $w_{\min}$ denote the maximum and minimum job sizes, respectively. The dependency on $w_{\max}/w_{\min}$ was unavoidable in the previous work since machines had to run at a rate of $\Theta(w_{\max}/w_{\min})$ to treat jobs as if they were of an equal size.

1.1 Our result

Our main result is the following.

THEOREM 1. *For the problem* ENERGY-EFFICIENT DEADLINE SCHEDULING WITH NO PREEMPTION *(*EDS-N*), for any $\epsilon > 0$, there exists a $(1+\epsilon)$-approximation algorithm which runs in $n^{O\left(\frac{4^{\alpha}\log^3 n}{\epsilon^2}\right)}$ time. Further, this result holds even when there are an arbitrary number of identical machines.*

As mentioned before, the best known approximation factor prior to our work had an exponential dependency on α. Further, no constant approximations were known even for a fixed α when there are more than one machine. Although we only present our result for identical machines, we note that our result can be easily extended to a constant number of heterogeneous machines.

Hence a natural looming question was if one could obtain an approximation arbitrarily close to the optimum. We answer this question positively by giving the first $(1+\epsilon)$-approximation, albeit with a quasi-polynomial running time. This is the first $(1+\epsilon)$-approximation for the general case. Previously, polynomial time algorithms were known for instances where all jobs are unit sized [3, 16] or jobs have agreeable deadlines [22]. [1] Quasi-polynomial-time approximation schemes (QPTAS) were known only for restricted instances where jobs lifespans form a laminar structure [5, 16]. That is, for any two distinct jobs, one job's lifespan is completely included in the other's, or the two have disjoint lifespans. The dynamic programming used in [5, 16] strongly relies on the fact that laminar instances have a tree representation. In contrast, our work makes no assumptions on instances, and our approach is very different. Our result implies that EDS-N might admit a PTAS.

1.2 Overview of our algorithm and analysis

Our algorithm is inspired by the dynamic programming framework in [17], which addressed various scheduling objectives, particularly completing as many jobs as possible before their deadlines. A simple yet useful observation made in [17] was that if we know that the scheduling intervals $\mathcal{T}$ where jobs $\mathcal{J}$ are scheduled (but neither the actual assignment of jobs to the intervals, nor the assignment of intervals to machines), then we can find a feasible schedule by finding a matching between $\mathcal{J}$ and $\mathcal{T}$ and greedily packing the intervals $\mathcal{T}$ into machines. This observation allowed [17] to focus on finding a 'good' set $\mathcal{T}$ of intervals that admits a matching between $\mathcal{J}$ and $\mathcal{T}$. Hence [17] used dynamic programming to find such a set. However, to make the dynamic programming efficient, [17] made use of a novel sketching scheme, and was able to find a set $\mathcal{T}$ of intervals that admits a 'fractional' matching between $\mathcal{J}$ and $\mathcal{T}$ where every job is matched by one unit, while every interval in $\mathcal{T}$ is matched by at most $1+\epsilon$ units; here a mild speed augmentation was used and such a fractional matching was said to have congestion $1+\epsilon$. Then, using the integrality of matching, [17] was able to find an intermediate schedule where each interval in $\mathcal{T}$ schedules at most two jobs, which was turned into a feasible schedule by either doubling the speed or the number of machines.

We obtain our QPTAS by carefully adapting this idea for EDS-N. First, observe that it is not enough to find a set $\mathcal{T}$ of scheduling intervals that admits a fractional matching with congestion $1+\epsilon$: Since our objective is minimizing total energy consumption, we will try to find a minimum cost integral matching where the edge between a job of size w and an interval of size p has cost $p(w/p)^{\alpha}$, which is exactly the energy needed to schedule the job on the interval. As in [17], in the integral matching, two jobs may have to be scheduled on the same interval. If we double the speed to convert it into a feasible schedule, we may have to use 2^{α} factor more energy in the worst scenario. Hence, for each pair of w and p, we will keep track of which intervals $\mathcal{T}'$ of length p are used to accommodate (matched with) jobs $\mathcal{J}'$ of size w. The matching between $\mathcal{J}'$ and $\mathcal{T}'$ we find has an additional nice property that only a few scheduling intervals in $\mathcal{T}'$ schedule more than one job from $\mathcal{J}'$. Then, the energy overhead for doubling the machines speed on such intervals can be charged to energy used on other intervals. The actual algorithm is slightly more complicated, but this is a high-level idea.

To discuss other difficulties and how our work is different from [17], we illustrate our dynamic programming at a high level. Suppose the time horizon is $(0, N)$, i.e. all jobs release times and deadlines are between 0 and N. For simplicity, assume that there is only one machine. We first guess the job j_m that is processed at the middle time $N/2$ (if such a job exists). Then, we need to decide which jobs should be scheduled before j_m and after. But such decisions can be exponentially many. This is where [17] uses sketches. The sketching scheme in [17] shows that jobs can be grouped if they have 'similar' release times, deadlines, and sizes. More precisely, it was shown that (1) if there is a feasible schedule for the original instance, then there is a feasible schedule for the instance simplified via the sketch, and (2) if a set of scheduling intervals admits a fractional matching with a congestion 1 for the simplified instance, then it admits a fractional matching with a congestion $(1+\delta)^h$ for the original instance where h is the depth of divide-and-conquer tree in the dynamic programming. Hence, if the dynamic programming has $O(\text{polylog}(n))$ levels and by setting $\delta = O(1/\text{polylog}(n))$ we can find good execution intervals with a small congestion.

However, we need to carefully adapt this dynamic programming to the speed scaling setting since each job can be processed for a very long or short period of time compared to its size. For example, we need to guess how long the middle job is processed in each iteration of the dynamic programming. Further, when some quantities such as job sizes are not polynomially bounded, the number of levels can be super-polylogarithmic. [17] handles this issue by showing that inflexible jobs, whose sizes are not so small compared to their lifespans, are the main concern, and flexible jobs of specific size w are involved in sketching at most $O(\log n)$ levels. Intuitively, flexible jobs are easy to schedule

[1] The instance is said to be agreeable if for any two jobs j and j', $r_j \le r_{j'}$, then $d_j \le d_{j'}$.

since they only need to use machines for short during their lifespans. In our setting, we have to adapt the notion of flexible/inflexible jobs in connection to energy consumption since how long jobs are processed can be very different from their sizes. Additionally, there are many details that should be carefully handled.

1.3 Other related work

We first discuss previous work on the preemptive case. As mentioned earlier, [22] gave a polynomial time algorithm for EDS when there is a single machine. The algorithm was refined in [19, 20]. When there are multiple machines, one can think of two cases, migratory and non-migratory, depending on whether a job can be scheduled on more than one machine or not. Migratory schedule was studied and polynomial-time algorithms were given in [11, 2, 4]. The algorithm in [11] was based on mathematical programing and the others were combinatorial. [6] studied the case when machines are heterogeneous in both migratory and non-migratory settings. In particular, [6] gave a $(1+\epsilon)\tilde{B}_\alpha$-approximation for heterogeneous machines when the schedule is migratory and preemptive.

We now shift our discussion to non-preemptive scheduling. Constant approximations depending on α are known for the single machine case [5, 6, 15]. The current best approximation ratio $(1+\epsilon)\tilde{B}_\alpha$ is due to [8]. [15] gives a $(w_{\max}/w_{\min})^{O(\alpha)}$ approximation when there are multiple machines, and their work can handle heterogeneous machines.

Finally, we briefly discuss other work on non-preemptive scheduling when machines have fixed speeds. Several $O(1)$-approximations [10, 14, 17] are known for the throughput objective where each job has some profit, and the goal is to maximize the total profit of the jobs completed by their deadlines. The problem of scheduling all jobs with the minimum number of machines was studied in [12, 13, 17]. Flow time objectives were studied in [9, 17].

Acknowledgements

This work was supported in part by NSF grant CCF-1409130.

2. REFERENCES

[1] S. Albers. Energy-efficient algorithms. *Commun. ACM*, 53(5):86–96, 2010.

[2] S. Albers, A. Antoniadis, and G. Greiner. On multi-processor speed scaling with migration. In *SPAA*, pages 279–288. ACM, 2011.

[3] E. Angel, E. Bampis, and V. Chau. Throughput maximization in the speed-scaling setting. *arXiv preprint arXiv:1309.1732*, 2013.

[4] E. Angel, E. Bampis, F. Kacem, and D. Letsios. Speed scaling on parallel processors with migration. In *Euro-Par 2012 Parallel Processing*, pages 128–140. Springer, 2012.

[5] A. Antoniadis and C.-C. Huang. Non-preemptive speed scaling. *J. of Scheduling*, 16(4):385–394, Aug. 2013.

[6] E. Bampis, A. Kononov, D. Letsios, G. Lucarelli, and M. Sviridenko. Energy efficient scheduling and routing via randomized rounding. In *FSTTCS*, pages 449–460, 2013.

[7] E. Bampis, D. Letsios, and G. Lucarelli. Green scheduling, flows and matchings. In *Algorithms and Computation*, pages 106–115. Springer, 2012.

[8] E. Bampis, D. Letsios, and G. Lucarelli. Speed-scaling with no preemptions. In *Algorithms and Computation*, pages 259–269. Springer, 2014.

[9] N. Bansal, H.-L. Chan, R. Khandekar, K. Pruhs, B. Schieber, and C. Stein. Non-preemptive min-sum scheduling with resource augmentation. In *FOCS*, pages 614–624, 2007.

[10] A. Bar-Noy, S. Guha, J. Naor, and B. Schieber. Approximating the throughput of multiple machines in real-time scheduling. *SIAM J. Comput.*, 31(2):331–352, 2001.

[11] B. D. Bingham and M. R. Greenstreet. Energy optimal scheduling on multiprocessors with migration. In *ISPA*, pages 153–161. IEEE, 2008.

[12] J. Chuzhoy, S. Guha, S. Khanna, and J. S. Naor. Machine minimization for scheduling jobs with interval constraints. In *FOCS*, pages 81–90, 2004.

[13] J. Chuzhoy and J. Naor. New hardness results for congestion minimization and machine scheduling. *J. ACM*, 53(5):707–721, 2006.

[14] J. Chuzhoy, R. Ostrovsky, and Y. Rabani. Approximation algorithms for the job interval selection problem and related scheduling problems. *Math. Oper. Res.*, 31(4):730–738, 2006.

[15] V. Cohen-Addad, Z. Li, C. Mathieu, and I. Milis. Energy-efficient algorithms for non-preemptive speed-scaling. In *Approximation and Online Algorithms*, pages 107–118. Springer, 2014.

[16] C.-C. Huang and S. Ott. New results for non-preemptive speed scaling. In *Mathematical Foundations of Computer Science 2014*, pages 360–371. Springer, 2014.

[17] S. Im, S. Li, B. Moseley, and E. Torng. A dynamic programming framework for non-preemptive scheduling problems on multiple machines [extended abstract]. In *SODA*, pages 1070–1086, 2015.

[18] S. Irani and K. Pruhs. Algorithmic problems in power management. *SIGACT News*, 36(2):63–76, 2005.

[19] M. Li, B. J. Liu, and F. F. Yao. Min-energy voltage allocation for tree-structured tasks. *Journal of Combinatorial Optimization*, 11(3):305–319, 2006.

[20] M. Li, A. C. Yao, and F. F. Yao. Discrete and continuous min-energy schedules for variable voltage processors. *PNAS*, 103(11):3983–3987, 2006.

[21] J. Markoff and S. Lohr. Intel's huge bet turns iffy, new york times, september 29, 2002.

[22] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced cpu energy. In *FOCS*, pages 374–382. IEEE, 1995.