Energy Efficient Scheduling via Partial Shutdown *

Samir Khuller † Jian Li ‡ Barna Saha §

Abstract

Motivated by issues of saving energy in data centers we define a collection of new problems referred to as "machine activation" problems. The central framework we introduce considers a collection of m machines (unrelated or related) with each machine i having an *activation cost* of a_i. There is also a collection of n jobs that need to be performed, and $p_{i,j}$ is the processing time of job j on machine i. Standard scheduling models assume that the set of machines is fixed and all machines are available. However, in our setting, we assume that there is an activation cost budget of A – we would like to *select* a subset S of the machines to activate with total cost $a(S) \leq A$ and *find* a schedule for the n jobs on the machines in S minimizing the makespan (or any other metric).

We consider both the unrelated machines setting, as well as the setting of scheduling uniformly related parallel machines, where machine i has activation cost a_i and speed s_i, and the processing time of job j on machine i is $p_{i,j} = \frac{p_j}{s_i}$, where p_j is the processing requirement of job j.

For the general unrelated machine activation problem, our main results are that if there is a schedule with makespan T and activation cost A then we can obtain a schedule with makespan $(2+\epsilon)T$ and activation cost $2(1+\frac{1}{\epsilon})(\ln \frac{n}{OPT}+1)A$, for any $\epsilon > 0$. We also consider assignment costs for jobs as in the generalized assignment problem, and using our framework, provide algorithms that minimize the machine activation and the assignment cost simultaneously. In addition, we present a greedy algorithm which only works for the basic version and yields a makespan of $2T$ and an activation cost $A(1+\ln n)$.

For the uniformly related parallel machine scheduling problem, we develop a polynomial time approximation scheme that outputs a schedule with the property that the activation cost of the subset of machines is at most A and the makespan is at most $(1+\epsilon)T$ for any $\epsilon > 0$. For the special case of m identical speed machines, the machine activation problem is trivial, since the cheapest subset of k machines is always the best choice if the optimal solution activates k machines. In addition, we consider the case when some jobs can be dropped (and are treated as outliers).

*Research supported by NSF Award CCF-0728839 and a Google Research Award.

†Department of Computer Science, University of Maryland, College Park, MD 20742. E-mail : samir@cs.umd.edu.

‡Department of Computer Science, University of Maryland, College Park, MD 20742. E-mail : lijian@cs.umd.edu.

§Department of Computer Science, University of Maryland, College Park, MD 20742. E-mail : barna@cs.umd.edu.

1 Introduction

Large scale data centers have emerged as an extremely popular way to store and manage a large volume of data. Most large corporations, such as Google, HP and Amazon have dozens of data centers. These data centers are typically composed of thousands of machines, and have extremely high energy requirements. Data centers are now being used by companies such as Amazon Web Services, to run large scale computation tasks for other companies who do not have the resources to create their own data centers. This is in addition to their own computing requirements.

These data centers are designed to be able to handle extremely high work loads in periods of peak demand. However, since the workload on these data centers fluctuates over time, we could selectively shut down part of the system to save energy when the demand on the system is low. Energy savings results not just from putting machines in a sleep state, but also from savings in cooling costs.

Hamilton (see the recent SIGACT News article [3]) argues that a ten fold reduction in the power needs of the data center may be possible if we can simply build systems that are optimized with power management as their primary goal. Suggested examples (summarizing from the original text) are:

1. Explore ways to simply do less during surge load periods.
2. Explore ways to migrate work in time. The work load on modern cloud platforms is very cyclical, with infrequent peaks and deep valleys. Even valley time is made more expensive by the need to own a power supply to be able to handle the peaks, a number of nodes adequate to handle surge loads, a network provisioned for worst case demand.

This leads to the issue of *which machines can we shut down*, since all machines in a data center are not necessarily identical. Each machine stores some data, and is thus not capable of performing every single job efficiently unless some data is first migrated to the machine. We will formalize this question very shortly.

To quote from the recent article by Birman et al. (SIGACT News [3]) "Scheduling mechanisms that

assign tasks to machines, but more broadly, play the role of provisioning the data center as a whole. As we'll see below, this aspect of cloud computing is of growing importance because of its organic connection to power consumption: both to spin disks, and run machines, but also because active machines produce heat and demand cooling. Scheduling, it turns out, comes down to deciding how to spend money."

Data is replicated on storage systems for both load balancing during peak demand periods, as well as for fault tolerance. Typically many jobs have to be scheduled on the machines in the data center. In many cases profile information for a set of jobs is available in advance, as well as estimates of cyclical workloads. Jobs may be I/O intensive or CPU intensive, in either case, an estimate of its processing time on each type of machine is available. Jobs that need to access specific data can be assigned to any one of the *subset* of machines that store the needed data. Our goal is to first *select* a subset of machines to activate, and then schedule the jobs on the active machines. From this aspect our problems differ from standard scheduling problems with multiple machines, where the set of active machines is the set of all machines. Here we have to decide *which machines to activate* and then schedule all jobs on the active machines.

The scheduling literature is vast, and one can formulate a variety of interesting questions in this model. We initiate this work by focusing our attention on perhaps one of the most widely studied machine scheduling problems since it matches the requirements of the application. We have a collection of jobs and unrelated machines, and need to decide which subset of machines to activate. The jobs can only be scheduled on active machines. This provides an additional dimension for scheduling problems that was not previously considered. This situation also makes sense when we have a certain set of computational tasks to process, a cost budget, and can purchase access to a set of machines.

One fundamental (and well studied) scheduling problem is as follows: Given a collection of n jobs, and m machines where the processing time of job j on machine i is $p_{i,j}$, assign all jobs to machines such that the makespan, i.e., the time when all jobs are complete, is minimized. This problem is widely referred to as *unrelated parallel machine scheduling* [17, 20]. If machine i does not have the data that job j needs to run, then we set $p_{i,j} = \infty$, otherwise the processing time $p_{i,j}$ is some constant p_j which only depends on job j. This special case is the so-called *restricted scheduling problem* and known to be NP-hard. However, if a schedule exists with makespan T, then the polynomial time algorithm developed by Lenstra, Shmoys and Tardos [17] shows an elegant rounding method to find a schedule with makespan $2T$. The subsequent generalization by Shmoys and Tardos [20], shows in fact that even with a cost function to map each job to a machine, if a mapping with cost C and makespan T exists, then their algorithm finds a schedule with cost C and makespan at most $2T$.

Motivated by the problem of shutting down machines when the demand is low, we define the following "machine activation" problem.

Given a set J of n jobs and a set M of m machines, our goal is to activate a subset S of machines and then map each job to an active machine in S, minimizing the overall makespan. Each machine has an activation cost of a_i. The activation cost of the subset S is $a(S) = \sum_{i \in S} a_i$. We show that if there is a schedule with activation cost A and makespan T, then we can find a schedule with activation cost $2(1+\frac{1}{\epsilon})(\ln \frac{n}{OPT}+1)A$ and makespan $(2+\epsilon)T$ for any $\epsilon > 0$ by the LP-rounding scheme (we call this is a $((2+\epsilon), 2(1+\frac{1}{\epsilon})(\ln \frac{n}{OPT}+1))$-approximation). We also present a greedy algorithm which gives us a $(2, 1+\ln n)$-approximation. Actually, the $\ln n$ term in the activation cost with this general formulation is unavoidable, since this problem is at least as hard to approximate as the set cover problem[1], for which a $(1-\epsilon)\ln n$ approximation algorithm will imply that $NP \subseteq DTIME(n^{O(\log \log n)})$ [8].

We also show that the recent PTAS developed by Epstein and Sgall [7] can be extended to the framework of machine activation problems for the case of scheduling jobs on uniformly related parallel machines. (The original PTAS by Hochbaum and Shmoys [12] is slightly more complicated than the method suggested by Epstein and Sgall [7].)

We also consider a version of the problem in which a subset of the jobs may be dropped to save energy (recall Hamilton's point(1)). In this version of the problem, each job j also has a benefit π_j and we need to process a subset of jobs with total benefit of at least Π. Suppose that a schedule exists with cost C_Π and makespan T_Π that obtains a total benefit at least Π. We show that the method due to Shmoys and Tardos [20] can be extended to find a collection of jobs to perform with expected benefit at least Π and expected cost C_Π, with a makespan guaranteed to be at most $2T_\Pi$ (see Appendix A) . (The recent work by Gupta et al. [11] gives a clever deterministic scheme with

[1] This is easy to see – we can view a set cover instance as a bipartite graph connecting elements (jobs) to corresponding sets (machines). If the element belongs to a set, then the processing time of the corresponding job on the corresponding machine is 0, o.w. it is ∞. An optimal set cover solution corresponds to an optimal set of machines to activate with 0 makespan.

makespan $3T_\Pi$ and cost $(1+\epsilon)C_\Pi$ along with several other results on scheduling with outliers. This has been further improved to $(2+\epsilon)T_\Pi$ and cost $(1+\epsilon)C_\Pi$ in [18].)

1.1 Related Work on Scheduling Generalizations of the work by Shmoys and Tardos [20], have considered the L_p norm. Azar and Epstein [2] give a 2-approximation for any L_p norm for any $p > 1$, and a $\sqrt{2}$-approximation for the L_2 norm. The bounds for $p \neq 2$ have been subsequently improved [16].

In addition, we can have release times r_{ij} associated with each job – this specifies the earliest time when job j can be started on machine i. Koulamas et al. [15] give a heuristic solution to this problem on uniformly related machines with a worst case approximation ratio of $O(\sqrt{m})$.

Minimizing resource usage has been considered before. In this framework, a collection of jobs J needs to be executed – each job has a processing time p_j, a release time r_j and a deadline d_j. In the continuous setting, a job can be executed on any machine between its release time and its deadline. In the discrete setting each job has a set of intervals during which it can be executed. The goal is to minimize the number of machines that are required to perform all the jobs. For the continuous case, Chuzhoy and Codenotti [4] have recently developed a constant factor approximation, improving upon a previous algorithm given by Chuzhoy et al [5]. For the discrete version Chuzhoy and Naor [6] have shown an $\Omega(\log \log n)$ hardness of approximation. However this framework does not model non-uniformity of machines, which is one of the key issues in data centers. In addition, non-uniformity of activation costs is not addressed in their work neither.

1.2 Related Work on Energy Minimization Augustine, Irani and Swamy [1] develop online algorithms to decide when a particular device should transition to a sleep state when multiple sleep states are available. Each sleep state has a different power consumption rate and a different transition cost. They provide deterministic online algorithms with competitive ratio arbitrarily close to optimal to decide in an online way which sleep state to enter when there is an idle period. See also the survey by Irani and Pruhs for other related work [14].

1.3 Our Contributions Our main contributions are:

- A randomized rounding method that approximates both activation cost and makespan for unrelated parallel machines. This method is based on rounding the LP solution of a certain carefully defined LP relaxation and uses ideas from work on dependent rounding [10, 16] (Section 2).
- Extensions of the above method when we have assignment costs in addition to activation costs as part of the objective function (Section 3).
- A greedy algorithm that approximates both activation cost and makespan for unrelated parallel machines and gives a $(2, 1+\ln n)$-approximation (Section 4).
- Extensions of these results to the case of handling outliers using the methods from [11] as well as release times (Section 5).
- A polynomial time approximation scheme for the cost activation problem for uniformly related parallel machines extending the construction given for the version of the problem with no activation costs [7] (Section 6).
- A simple dependent rounding scheme for the partial GAP problem (Appendix A).

2 LP Rounding for Machine Activation on Unrelated Machines

In this section, we first provide a simple rounding-ing scheme with an approximation ratio of $(O(\log n), O(\log n))$. Then we improve it to a $(2+\epsilon, 2(1+\frac{1}{\epsilon})(\ln \frac{n}{OPT}+1))$-approximation by a new rounding scheme. We can formulate the scheduling activation problem as an integer program. We define a variable y_i for each machine i, which is 1 if the machine is open and 0, if it is closed. For every machine-job pair, we have a variable $x_{i,j}$, which is 1, if job j is assigned to machine i and is 0, otherwise. In the corresponding linear programming relaxation, we relax the y_i and $x_{i,j}$ variables to be in $[0,1]$. The first set of constraints require that each job is assigned to some machine. The second set of constraints restrict the jobs to be assigned to only active machines, and the third set of constraints limit the workload on a machine. We require that $1 \geq x_{i,j}, y_j \geq 0$ and if $p_{i,j} > T$ then $x_{i,j} = 0$. The formulation is as shown below:

$$
\begin{aligned}
(2.1) \qquad & \min \sum_{i=1}^{m} a_i y_i \\
s.t. \quad & \sum_{i \in M} x_{i,j} = 1 \ \ \forall j \in J \\
& x_{i,j} \leq y_i \ \ \forall i \in M, j \in J \\
& \sum_j p_{i,j} x_{i,j} \leq T y_i \ \ \forall i
\end{aligned}
$$