A Note on Multiprocessor Speed Scaling with Precedence Constraints

Evripidis Bampis
Sorbonne Universités
UPMC Univ. Paris 06
UMR 7606, LIP6
F-75005, Paris, France
Evripidis.Bampis@lip6.fr

Dimitrios Letsios
Sorbonne Universités
UPMC Univ. Paris 06
UMR 7606, LIP6
F-75005, Paris, France
Dimitrios.Letsios@lip6.fr

Giorgio Lucarelli
Sorbonne Universités
UPMC Univ. Paris 06
UMR 7606, LIP6
F-75005, Paris, France
Giorgio.Lucarelli@lip6.fr

ABSTRACT

We consider the problem of scheduling a set of jobs, under precedence constraints, on a set of speed scalable parallel processors. The goal is to minimize the makespan of the schedule, i.e. the time at which the last job finishes its execution, without violating a given energy budget. This situation finds applications in computer devices whose lifetime depends on a limited battery efficiency. In order to handle the energy consumption we use the energy model introduced in [Yao et al., FOCS'95], which captures the intuitive idea that the higher is the processor's speed the higher is the energy consumption. We propose a $(2 - \frac{1}{m})$-approximation algorithm improving the best known poly-$\log(m)$-approximation algorithm for the problem [Pruhs et al., TOCS 2008], where m is the number of the processors. We also extend the simple idea used for the above problem, in order to propose a generalized framework that finds applications to other scheduling problems in the speed scaling setting.

Categories and Subject Descriptors

F.2.2 [**Analysis of algorithms and problem complexity**]: Sequencing and scheduling

Keywords

Speed scaling; Scheduling; Approximation algorithms; Convex programming

1. INTRODUCTION

Due to the increasing use of computing devices and the need of more computing power in computer centers and of more autonomy in personal/mobile devices, the energy consumption in computer systems has become an important issue today. One standard way to handle the energy consumption is through the dynamic scaling of the voltage and the frequency of a processor, which is known as the *speed scaling* mechanism. According to the cube-root rule for the CMOS devices, if a processor runs at speed s then the power needed is $P(s) = s^3$. However, recent experiments showed that the exponent is in practise smaller [10] (e.g., the exponent is equal to 1.11 for Intel PXA 270, 1.62 for Pentium M770 and 1.66 for a TCP offload engine). So, we can describe the power as $P(s) = s^\alpha$, where $\alpha > 1$ is a small constant that depends on the processor. The energy consumption is the power integrated over time. Intuitively, the speed scaling mechanism captures the idea that the higher is the processor's speed the higher is the energy consumption. The algorithmic study of the speed scaling mechanism is initiated by Yao et al. [11] in 1995. Since then, there is a series of works in the speed scaling setting (see the surveys [1, 2]).

A classical objective in scheduling is the minimization of the makespan, i.e., the time at which the last job finishes its execution. Unfortunately, makespan minimization and energy minimization are conflicting objectives. One of the ways to take into account the bicriteria nature of speed scaling problems is by adopting a budget approach where a fixed budget of energy is given and the only objective is the minimization of the makespan (see for example [3, 7]). This is a quite natural assumption in our setting where the energy of a battery may be assumed to be fixed.

In this context, Pruhs et al. considered in [7] the problem of scheduling a set of jobs on a set of speed scalable parallel processors subject to precedence constraints among the jobs. The goal is to minimize the makespan of the schedule without exceeding a given energy budget. The approach in [7] is based on *constant power schedules*, which are schedules that keep the total power of all processors constant over time. Based on this property and by performing a binary search to determine the value of the power, they transform the problem to the classical problem of minimizing the makespan for scheduling a set of jobs with precedence constraints on related parallel processors, in which each processor runs at a single predefined speed. Using the known $O(\log m)$-approximation algorithm for the latter problem presented in [4, 5], they give an approximation algorithm of ratio $O(\log^{1+2/\alpha} m)$ for the speed scaling problem with precedence constraints, where m is the number of the processors.

Note that, when the energy consumption is not taken into account, the speed scaling problem reduces to the classical makespan minimization scheduling problem on identical parallel processors with precedence constraints, in which all processors run at the same predefined constant speed which is NP-hard. Graham [6] proved that the simple list scheduling algorithm is a $(2 - \frac{1}{m})$-approximation algorithm for it. On the negative side, Svensson [9] showed that it is NP-hard to improve upon the approximation ratio obtained by Graham, assuming a new variant of the unique games conjecture.

A natural question arising here is whether it is possible to reduce the gap between the approximability of the scheduling problems on parallel processors with and with-

out speed scaling. In this note, we propose a simple $(2 - \frac{1}{m})$-approximation algorithm for the speed scaling problem with precedence constraints, matching the approximation ratio for the classical setting. In fact, we even generalize the problem studied in [7] by taking into account that each job is available for execution after a given release date. For the more general problem, our algorithm becomes 2-approximate. Our approach is based on several ingredients of Graham's algorithm for the corresponding classical problem with precedence constraints and release dates. Initially, using the lower bounds for the optimal solution used by Graham, we give a convex programming relaxation for the speed scaling problem. By solving this convex program, we define a speed, and hence a processing time, for each job. As these processing times respect the energy budget, we transform our problem to the classical problem without speed scaling and we use the list scheduling algorithm to obtain a feasible schedule.

Interestingly, the idea of using the lower bounds used for the classical setting in order to obtain a convex programming relaxation for the speed scaling setting can be also applied in other speed scaling problems and directly pass the approximation ratios from the classical to the speed scaling environment. In this note, we propose a generalized framework and a characterization of such scheduling problems and we present applications of our method for shop scheduling environments in the speed scaling setting.

Organization of the Paper.

In Section 2 we present the $(2 - \frac{1}{m})$-approximation algorithm for the speed scaling problem with precedence constraints on multiprocessors, improving the best known poly-$\log(m)$-approximation algorithm for this problem [7]. Based on the approach that we used for this problem, we propose in Section 3 a general framework that can directly pass the approximation ratios from a classical problem (without caring about the energy consumption) to the corresponding speed scaling problem, and we present the ingredients needed for this transformation. Then, in Section 3.1 we give an application of our framework to a well-known problem in scheduling, namely the open shop problem. We conclude in Section 4.

2. PRECEDENCE CONSTRAINTS

In this section, we consider the makespan minimization problem of scheduling a set of jobs, with release dates and precedence constraints among them, on parallel speed scalable processors subject to a budget of energy and we present a 2-approximation algorithm. Henceforth, we denote this problem as PREC_S and the corresponding classical problem without speed scaling as PREC. In the case where there are no release dates, our algorithm is $(2 - \frac{1}{m})$-approximate.

Problem Statement and Notation.

In the PREC_S problem, we are given a set of jobs $\mathcal{J} = \{1, 2, \ldots, n\}$ which have to be scheduled on m speed scalable parallel processors. Each job $j \in \mathcal{J}$ is characterized by an amount of work w_j and a release date r_j. We do not allow preemptions of the jobs which means that, for a given job $j \in \mathcal{J}$, we must choose a single processor on which j will be executed without any interruption. There are precedence constraints among the jobs. Specifically, if the job j precedes the job j', then j' cannot start until j is completed. In this case, we call j a predecessor of j'. Our objective is to find a feasible schedule with minimum completion time (makespan) so that the energy consumption is not greater than a given energy budget E.

The precedence constraints are represented in the form of a directed acyclic graph $G = (V, A)$. The set of vertices V contains one vertex for each job and the arc (j, j') belongs to the set of arcs A if and only if the job j is constrained to precede the job j'. We denote by $\mathcal{C}$ the set of all the paths (or chains) in G. Note that these paths are not necessarily maximal. This means that there may exist paths that are completely included in other paths. For a path $c \in \mathcal{C}$, we denote by $\mathcal{J}(c)$ the set of jobs which appear in c and by $r(c)$ the release date of the first job in the path. Given the processing times of the jobs, the length of a path $c \in \mathcal{C}$ is the sum of the processing times of the jobs in c.

Algorithm's Ingredients.

Let us consider first the problem PREC in which the jobs have fixed processing times. That is, each job $j \in \mathcal{J}$ is characterized by a processing time p_j instead of an amount of work w_j. Next, we present two well-known lower bounds on the value C^* of any optimal solution for PREC and an approximation algorithm which is upper bounded by 2 times the maximum of these linear bounds.

In the best case, there is no idle period in the schedule and all the processors complete together. Therefore,

$$C^* \geq \frac{1}{m} \sum_{j \in \mathcal{J}} p_j$$

Because of the precedence constraints, for every $c \in \mathcal{C}$, we have that

$$C^* \geq r(c) + \sum_{j \in \mathcal{J}(c)} p_j$$

For PREC, Graham [6] proposed the well-known list scheduling algorithm which follows.

Algorithm 1

1: Every time that a processor i becomes available, schedule on i a released job for which all the predecessors have been completed.

THEOREM 1. [6] *Let C be the makespan of the schedule produced by Algorithm 1 for* PREC. *Then,*

$$C \leq 2 \cdot \max \left\{ \frac{1}{m} \sum_{j \in \mathcal{J}} p_j, \max_{c \in \mathcal{C}} \left\{ r(c) + \sum_{j \in \mathcal{J}(c)} p_j \right\} \right\}$$

The Algorithm.

Now, we turn back our attention to PREC_S. Note that, in an optimal schedule for this problem, each job $j \in \mathcal{J}$ is executed with a constant speed s_j due to the convexity of the speed-to-power function. Given this speed we can compute its processing time $p_j = \frac{w_j}{s_j}$ and its energy consumption $E_j = w_j s_j^{\alpha-1} = \frac{w_j^\alpha}{p_j^{\alpha-1}}$.

Based on this, we present a convex programming relaxation for PREC_S. We introduce a variable y for the makespan

and a variable x_j for each job $j \in \mathcal{J}$ which corresponds to the processing time of j. The objective is to minimize y. Given the lower bounds that we described previously for PREC, we add two linear constraints that relate the processing times of the jobs with the makespan, as well as, a constraint which ensures that the energy budget is not exceeded. So, we obtain the following convex program, denoted as CP.

$$\min y$$

$$y \geq \frac{1}{m} \sum_{j \in \mathcal{J}} x_j \tag{1}$$

$$y \geq r(c) + \sum_{j \in \mathcal{J}(c)} x_j \qquad c \in \mathcal{C} \tag{2}$$

$$\sum_{j \in \mathcal{J}} \frac{w_j^\alpha}{x_j^{\alpha-1}} \leq E \tag{3}$$

$$x_j \geq 0 \qquad j \in \mathcal{J}$$

This convex program has an exponential number of constraints as the number of paths may be exponential to the size of the instance. However, we will consider for the moment that we can get an optimal solution of it, and we explain later how we can solve it in polynomial time through a transformation to another convex program of polynomial size. Based on CP, we propose the following algorithm for PREC_S.

Algorithm 2

1: Solve the convex program CP.
2: Let $\vec{x}_{CP}$ be the vector of the processing times of jobs obtained by CP.
3: Apply Algorithm 1 as if the jobs have processing times $\vec{x}_{CP}$ to create a feasible schedule for PREC_S.

THEOREM 2. *Algorithm 2 achieves an approximation ratio of 2 for* PREC_S.

PROOF. Consider an instance of PREC_S, and let SOL and OPT be the value of our algorithm's solution and of an optimal solution, respectively. Given a vector of processing times $\vec{x}$ which corresponds to a feasible solution of the convex program, we denote by $CP(\vec{x})$ the corresponding value of the convex program, i.e. the minimum possible value of y with respect to $\vec{x}$. Furthermore, let $\vec{x}_{CP}$ be the values of the variables in the optimal solution of the convex program produced by the Algorithm 2 and $\vec{x}_{OPT}$ be the processing times of the jobs in an optimal solution of the problem PREC_S. Clearly, $CP(\vec{x}_{CP})$ is the value of the optimal solution of the convex relaxation. We have that

$$SOL \leq \rho \cdot CP(\vec{x}_{CP}) \leq \rho \cdot CP(\vec{x}_{OPT}) \leq \rho \cdot OPT$$

The first inequality comes from the fact that $\rho = 2$ multiplied by the maximum of the lower bounds for PREC is an upper bound on the makespan of the schedule produced by Algorithm 1, i.e., Theorem 1. The second inequality holds because $\vec{x}_{CP}$ is an optimal solution of the convex relaxation and $\vec{x}_{OPT}$ corresponds to just a feasible one for CP. Finally, the third inequality is based on the fact that the convex program is a relaxation of the speed scaling problem. The theorem follows. □

When the jobs have no release dates our result can be slightly improved and we obtain a $(2 - \frac{1}{m})$-approximation algorithm by using the same lower bounds. In this case, when the jobs have fixed processing times, Algorithm 1 is $(2 - \frac{1}{m})$-approximate w.r.t. these lower bounds.

COROLLARY 1. *Algorithm 2 achieves an approximation ratio of* $2 - \frac{1}{m}$ *for* PREC_S *when all jobs have the same release date.*

An Equivalent Polynomial Size Convex Program.

As mentioned before, CP has an exponential number of constraints. In order to deal with this, we propose an equivalent convex programming relaxation. Let y_j be a variable indicating the completion time of job $j \in \mathcal{J}$. We replace the constraints (2) of CP with the following constraints, and we obtain a new convex program CP'.

$$y_j \leq y \qquad j \in \mathcal{J} \tag{4}$$

$$r_j + x_j \leq y_j \qquad j \in \mathcal{J} \tag{5}$$

$$y_j + x_{j'} \leq y_{j'} \qquad (j, j') \in A \tag{6}$$

$$y_j \geq 0 \qquad j \in \mathcal{J}$$

Next, we prove that the two convex programs are equivalent. Hence, there exists a polynomial 2-approximation algorithm for PREC_S.

LEMMA 1. *The two convex programs are equivalent.*

PROOF. Assume that we are given a feasible solution $(\tilde{x}_j, \tilde{y})$ for CP. We will show that there exists a feasible solution for CP' of the same cost. The variables x_j and y have equal values in both solutions. So, the constraints (1) and (3) are satisfied. For a given job $j \in \mathcal{J}$, we denote by $\mathcal{C}(j)$ the set of paths that have the job j as a right extremity. We set the value of the variable y_j equal to $\tilde{y}_j = \max_{c \in \mathcal{C}(j)} \left\{ r(c) + \sum_{j' \in \mathcal{J}(c)} \tilde{x}_{j'} \right\}$. It remains to show that the constraints (4), (5) and (6) are satisfied. By considering the path $c \in \mathcal{C}(j)$ that contains only the job j on our definition of $\tilde{y}_j$, the constraints (5) are satisfied. Moreover, assume that $(j, j') \in A$. Based on our definition, let $\tilde{y}_j = r(c) + \sum_{j'' \in \mathcal{J}(c)} \tilde{x}_{j''}$, for some path c. Then, as $r(c) = r(c \cup \{j'\})$,

$$\tilde{y}_{j'} \geq r(c \cup \{j'\}) + \sum_{j'' \in \mathcal{J}(c \cup \{j'\})} \tilde{x}_{j''} = y_j + x_{j'}$$

and the constraints (6) are also satisfied. Given our definition of $\tilde{y}_j$ and the fact that the solution $(\tilde{x}_j, \tilde{y})$ satisfies the constraints (2), we conclude that the solution $(\tilde{x}_j, \tilde{y}_j, \tilde{y})$ satisfies the constraints (4) and it is indeed feasible for CP'.

To the other direction, assume that we have a feasible solution $(\tilde{x}_j, \tilde{y}_j, \tilde{y})$ for CP'. Then, we claim that the solution $(\tilde{x}_j, \tilde{y})$ is feasible for CP. The constraints (1) and (3) are satisfied directly. Next, consider any path $c \in \mathcal{C}$ and assume that it contains the jobs $j^{(1)}, j^{(2)}, \ldots, j^{(k)}$ in this order. We have that

$$r(c) + \sum_{j \in \mathcal{J}(c)} \tilde{x}_j = r_{j^{(1)}} + \sum_{\ell=1}^{k-1} \tilde{x}_{j^{(\ell+1)}}$$

$$\leq r_{j^{(1)}} + \sum_{\ell=1}^{k-1} (\tilde{y}_{j^{(\ell+1)}} - \tilde{y}_{j^{(\ell)}}) \leq \tilde{y}_{j^{(k)}} \leq y$$

where the inequalities follow from the constraints (6), (5) and (4), respectively. □

3. A GENERALIZED FRAMEWORK

In this section, we present a generalized method for obtaining approximation algorithms for speed scaling problems in which the objective function is the minimization of e.g. the makespan and there is a given budget of energy which must not be exceeded. The main assumption is that the energy consumption of any optimal schedule can be expressed as a convex function $E(\vec{x})$ of the vector of the processing times $\vec{x}$ of the jobs (or operations). This assumption is true if a processor satisfies the standard speed-to-power function $P(s) = s^\alpha$, where $\alpha > 1$ is a constant, because each job (or operation) is executed with a constant speed due to the convexity of $P(s)$.

Consider a classical makespan minimization problem Π in which the jobs have fixed processing times. We denote by Π_S the speed scaling variant of the problem where each job (or operation) has an amount of work instead of a fixed processing time, the processors' speed can be varied, the objective remains the same and we are given a budget of energy which must not be exceeded. In order to apply our method, we need the following ingredients.

- A set of ℓ linear bounds on Π's optimal solution of the form
$$C^* \geq f_k(\vec{p})$$
for $k = 1, 2, \ldots, \ell$, where $\vec{p}$ is the vector of processing times of the jobs and $f_k(\vec{p})$ is a linear function of $\vec{p}$.
- A ρ-approximation algorithm $\mathcal{A}$ for Π which always produces a solution such that $C \leq \rho \cdot \max_{k=1}^{\ell} f_k(\vec{p})$, where C is the value of the $\mathcal{A}$'s solution for Π.

Provided the above ingredients, we may obtain a ρ-approximation algorithm for the speed scaling problem Π_S by using the algorithm $\mathcal{A}$ as a black box. Let us describe in a general manner our approach.

Our first task consists in constructing a convex programming relaxation CP for Π_S by using the lower bounds of Π. We introduce a variable y for the makespan and a vector variable $\vec{x}$ which corresponds to the processing times of the jobs. The objective is to minimize y. We add linear constraints of the form $y \geq f_k(\vec{x})$, for $k = 1, 2, \ldots, \ell$, and a constraint which ensures that the budget of energy is not exceeded. So, we obtain the following convex program.

$$\begin{aligned} \min & \; y \\ y &\geq f_k(\vec{x}) \qquad \forall 1 \leq k \leq \ell \\ E(\vec{x}) &\leq E \\ \vec{x} &\geq 0 \end{aligned}$$

Next, we propose the following algorithm for Π_S.

Algorithm 3

1: Solve the convex program CP.
2: Let $\vec{x}_{CP}$ the vector of the processing times obtained.
3: Apply the algorithm $\mathcal{A}$ as if the jobs have processing times $\vec{x}_{CP}$ to create a feasible schedule for Π_S.

The following theorem can be proved using the same arguments as for Theorem 2.

THEOREM 3. *Algorithm 3 achieves an approximation ratio of ρ for Π_S.*

3.1 An Example: Open Shop

In this section, we consider the speed scaling problem of minimizing the makespan in an open shop environment and we present a 2-approximation algorithm. We denote this problem as SHOP$_S$.

Problem Statement.

An instance of SHOP$_S$ contains a set of n jobs $\mathcal{J} = \{1, 2, \ldots, n\}$ which have to be scheduled by a set of m parallel processors $\mathcal{P} = \{1, 2, \ldots, m\}$. The job $j \in \mathcal{J}$ consists of m operations $O_{1,j}, O_{2,j}, \ldots, O_{m,j}$ and the operation $O_{i,j}$, $i \in \mathcal{P}$, has to be entirely executed by the processor i. Every operation $O_{i,j}$, $i \in \mathcal{P}$ and $j \in \mathcal{J}$, is associated with an amount of work $w_{i,j} \geq 0$. The open shop constraint enforces that no pair of operations of the same job are executed at the same time. We do not allow preemptions of operations which means that each operation has to be executed without interruptions. Our objective is to find a feasible schedule with minimum completion time (makespan) whose energy consumption does not exceed a given budget E.

The study of the open shop problem is motivated by applications where each task is composed by operations that have to be executed on special purpose machines, for example machines for floating point operations or graphics operations, etc. The open shop constraint is implied by the fact that the operations of the same task have access to the same physical resources, and hence they cannot be executed at the same time.

Algorithm's Ingredients.

Let us consider the problem SHOP in which each operation $O_{i,j}$ has a fixed processing time $p_{i,j}$ instead of an amount of work $w_{i,j}$. We denote by C^* the makespan of an optimal solution for this problem. Next, we give a set of lower bounds for C^*.

Each processor completes when all the operations assigned to it are finished. Therefore, for every $i \in \mathcal{P}$, we have that

$$C^* \geq \sum_{j=1}^{n} p_{i,j}$$

Similarly, a job completes when all its operation are finished. Hence, because of the open shop constraint, for every $j \in \mathcal{J}$ it must hold that

$$C^* \geq \sum_{i=1}^{m} p_{i,j}$$

For SHOP, Racsmány[1] proposed the well-known list scheduling algorithm which follows.

Algorithm 4

1: Whenever a processor $i \in \mathcal{P}$ becomes available, schedule on i an operation $O_{i,j}$ of a job $j \in \mathcal{J}$ which is not processed by any other machine at the same time.

[1]Check [8] for more details.

THEOREM 4. (Racsmány) *Let C be the makespan of the schedule produced by Algorithm 4 for* SHOP. *Then,*

$$C \leq 2 \cdot \max \left\{ \max_{i \in \mathcal{P}} \left\{ \sum_{j \in \mathcal{J}} p_{i,j} \right\}, \max_{j \in \mathcal{J}} \left\{ \sum_{i \in \mathcal{P}} p_{i,j} \right\} \right\}$$

Algorithm.

Let us now describe how we obtain a 2-approximation algorithm for SHOP_S. Initially, we give a convex programming relaxation CP for it. As we described in Section 3, we introduce a variable $x_{i,j}$ which corresponds to the processing time of the operation $O_{i,j}$ and a variable y that corresponds to the makespan. Then, we construct CP as follows.

$$\begin{aligned}
&\min y \\
y &\geq \sum_{j=1}^{n} x_{i,j} && i \in \mathcal{P} \\
y &\geq \sum_{i=1}^{m} x_{i,j} && j \in \mathcal{J} \\
\sum_{i=1}^{m} \sum_{j=1}^{n} &\frac{w_{i,j}^{\alpha}}{x_{i,j}^{\alpha-1}} \leq E \\
x_{i,j} &\geq 0 && i \in \mathcal{P}, j \in \mathcal{J}
\end{aligned}$$

Provided the above convex program, we can apply Algorithm 3 in order to solve SHOP_S by using Algorithm 4 as a black box. Then, Theorems 3 and 4 imply the following theorem.

THEOREM 5. *There exists a 2-approximation algorithm for* SHOP_S.

4. CONCLUSIONS

We have improved from poly-log(m) to $2 - \frac{1}{m}$ the approximation ratio for the multiprocessor speed scaling problem with precedence constraints where the objective is to minimize the makespan of the schedule subject to a given energy budget. In fact we have even generalized the problem by considering that jobs are subject to release dates. Our approach is much simpler than the previous one and it can be used to deal with other problems in the speed scaling setting, like the open shop problem as well as problems on other shop environments where similar linear lower bounds on the makespan are known for the classical setting. An interesting question is whether we can extend this approach to other scheduling problems and more specifically to other performance objectives, i.e., for the average completion time of the jobs. A positive answer to this question could give us an insight to the difficulty that adds the consideration of the energy consumption in scheduling problems.

Finally, we note that our approach can be used not only for power functions of the form $P(s) = s^{\alpha}$, but for any convex speed-to-power function, as we just need the property of convexity.

Acknowledgements

We would like to thank Maxim Sviridenko for our helpful discussions.

Partially supported by the project ALGONOW, co-financed by the European Union (European Social Fund - ESF) and Greek national funds, through the Operational Program "Education and Lifelong Learning", under the program THALES. Partially supported by the project Mathematical Programming and Non-linear Combinatorial Optimization under the program PGMO.

5. REFERENCES

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

[2] S. Albers. Algorithms for dynamic speed scaling. In *STACS*, volume 9 of *LIPIcs*, pages 1–11. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011.

[3] D. P. Bunde. Power-aware scheduling for makespan and flow. In *SPAA*, pages 190–196. ACM, 2006.

[4] C. Chekuri and M. A. Bender. An efficient approximation algorithm for minimizing makespan on uniformly related machines. *Journal of Algorithms*, 41:212–224, 2001.

[5] F. A. Chudak and D. B. Shmoys. Approximation algorithms for precedence-constrained scheduling problems on parallel machines that run at different speeds. *Journal of Algorithms*, 30(2):323–343, 1999.

[6] R. Graham. Bounds for certain multiprocessor anomalies. *Bell System Technical Journal*, 45(9):1563–1581, 1966.

[7] K. Pruhs, R. van Stee, and P. Uthaisombut. Speed scaling of tasks with precedence constraints. *Theory of Computing Systems*, 43:67–80, 2008.

[8] D. B. Shmoys, C. Stein, and J. Wein. Improved approximation algorithms for shop scheduling problems. *SIAM Journal on Computing*, 23(3):617–632, 1994.

[9] O. Svensson. Conditional hardness of precedence constrained scheduling on identical machines. In *STOC*, pages 745–754, 2010.

[10] A. Wierman, L. L. H. Andrew, and A. Tang. Power-aware speed scaling in processor sharing systems. In *INFOCOM*, pages 2007–2015. IEEE, 2009.

[11] F. F. Yao, A. J. Demers, and S. Shenker. A scheduling model for reduced CPU energy. In *FOCS*, pages 374–382. IEEE Computer Society, 1995.