Multiprocessor Speed Scaling with Precedence Constraints *

Evripidis Bampis[1], Dimitrios Letsios[1,2], Giorgio Lucarelli[1,3]

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

[2] Institut für Informatik, Technische Universität München, Munich, Germany
letsios@in.tum.de

[3] Université Grenoble-Alpes, INP, UMR 5217, LIG, Grenoble, France
giorgio.lucarelli@inria.fr

Mots-clés : *Speed scaling ; Scheduling ; Approximation algorithms ; Convex programming*

1 Introduction

According to the *speed scaling* mechanism for saving energy in computing systems, if a processor runs at speed $s(t)$ at a time t then the power needed is $P(s(t)) = s(t)^\alpha$, where $\alpha > 1$ is a small constant that depends on the processor. The energy consumption is the power integrated over time, i.e., $E = \int P(s(t))dt$. The algorithmic study of the speed scaling mechanism is initiated by Yao et al. [6] in 1995. Since then, there is a series of works in this setting (see the survey [1]).

We consider the following scheduling problem : we are given a set of jobs $\mathcal{J}$ 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, i.e., the interruption of the execution of a job is not permitted. There are precedence constraints among the jobs which 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' (i.e., j' cannot start until j is completed). We denote by $\mathcal{C}$ the set of all the possible paths in G. 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. Our objective is to find a feasible schedule with minimum total completion time (makespan) so that the energy consumption is not greater than a given energy budget E.

The problem without release dates was studied by Pruhs et al. in [4]. Their approach 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 [2], they give an approximation algorithm of ratio $O(\log^{1+2/\alpha} m)$ for the speed scaling problem with precedence constraints.

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 which is known to be NP-hard. Graham [3] proved that the simple *List Scheduling* algorithm is a $(2 - \frac{1}{m})$-approximation algorithm for it. On the negative side, Svensson [5] showed that it is NP-hard to improve upon this approximation ratio, assuming a new variant of the unique games conjecture.

*The authors are partially supported by the project Mathematical Programming and Non-linear Combinatorial Optimization under the program PGMO. D. Letsios is partially supported by the German Research Foundation, project AL-464/7-1. G. Lucarelli is supported by the ANR project Moebus.

2 A 2-approximation algorithm

We propose a simple 2-approximation algorithm for the speed scaling problem with precedence constraints and release dates, generalizing the problem studied by Pruhs et al. in [4], improving upon their proposed ratio of $O(\log^{1+2/\alpha} m)$ and matching the ratio for the classical setting. Our approach is based on the lower bounds for the optimal solution used by Graham in the analysis of the List Scheduling algorithm for the corresponding classical problem.

Theorem 1. [3] *Let C be the makespan of the schedule produced by the List Scheduling algorithm for the classical problem with precedence constraints and release dates, and C^* be the makespan of the corresponding optimal schedule. Then,*

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

Note that, in an optimal schedule for the speed scaling 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 observation and the lower bounds of Theorem 1, we propose a convex programming relaxation for the speed scaling problem. 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.

$$\begin{aligned}
&\min y \\
y &\geq \frac{1}{m}\sum_{j\in\mathcal{J}} x_j \\
y &\geq r(c) + \sum_{j\in\mathcal{J}(c)} x_j && \forall c \in \mathcal{C} \\
&\sum_{j\in\mathcal{J}} \frac{w_j^\alpha}{x_j^{\alpha-1}} \leq E \\
x_j &\geq 0 && \forall j \in \mathcal{J}
\end{aligned}$$

By solving this convex program, we define a processing time, for each job. As these processing times respect the energy budget, we are able to transform our problem to the classical problem without speed scaling and we use the List Scheduling algorithm to obtain a feasible schedule. Note that, the proposed convex program has an exponential number of constraints as the number of paths may be exponential to the size of the instance. However, we can give an equivalent convex programming relaxation of polynomial size. The following theorem follows.

Theorem 2. *There is a polynomial-time 2-approximation algorithm for the makespan minimization speed scaling problem with precedence constraints and release dates.*

Références

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

[2] 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.

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

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

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

[6] 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.