Scheduling on Power-Heterogeneous Processors

Susanne Albers[1] Evripidis Bampis[2] Dimitrios Letsios[3]
Giorgio Lucarelli[4] Richard Stotz[1]

[1] Fakultät für Informatik, Technische Universität München
{albers, stotz}@in.tum.de
[2] Sorbonne Universités, UPMC Univ. Paris 06, UMR 7606, LIP6, Paris, France.
Evripidis.Bampis@lip6.fr
[3] Université de Nice - Sophia Antipolis
dletsios@unice.fr
[4] Université Grenoble-Alpes, INP, UMR 5217, LIG, France.
giorgio.lucarelli@imag.fr

Mots-clés : *Scheduling, Speed-scaling, Heterogeneous, Approximation algorithms.*

1 Introduction

One of the main technological alternatives in order to handle the energy consumed in modern computer systems is based on the use of speed-scalable processors where the speed of a processor may be dynamically changed over time. When a processor runs at speed s, then the rate with which the energy is consumed (i.e., the power) is $f(s)$ with f a non-decreasing function of the speed. According to the well-known cube-root rule for CMOS devices, the speed of a device is proportional to the cube-root of the power and hence $f(s) = s^3$. However, the standard model that is usually studied in the literature considers that the power is $f(s) = s^\alpha$ with $\alpha > 1$ a constant. Other works consider that the power is an arbitrary convex function [5].

The algorithmic study of this area started with the seminal paper of Yao et al. [8], where a set of jobs $\mathcal{J}$, each one specified by its work w_j, its release date r_j and its deadline d_j, has to be scheduled preemptively on a single processor so that the energy consumption is minimized. In [8], an optimal polynomial-time algorithm has been proposed for this problem. The *homogeneous multiprocessor* setting in which the *preemption* and the *migration* of the jobs are allowed (without permitting parallel execution of a job) has been also studied. Albers et al. [1] and Angel et al. [2], independently, have proposed an optimal polynomial-time algorithm based on the computation of several maximum flows in appropriate networks. Albers et al. [1] have also considered the online version of the multiprocessor problem and they studied the well-known *Average Rate (AVR)* algorithm which have been proposed in [8] for the single-processor setting. Specifically, they proved that AVR is $(2^{\alpha-1}\alpha^\alpha + 1)$-competitive. Note that, for the single-processor case, the competitive ratio of AVR cannot be better than $2^{\alpha-1}\alpha^\alpha$ [4].

In this paper, we consider the problem of scheduling a set of jobs on a set $\mathcal{P}$ of *power-heterogeneous* processors. In this direction, some recent papers [3, 6, 7] have studied the impact of the introduction of the heterogeneity on the difficulty of some power-aware scheduling problems. In our setting, each processor $p \in \mathcal{P}$ is characterized by each own power function $f_p(s)$. We consider two classes of functions :

1. *Arbitrary Power Functions* : The function $f_p(s)$ of each processor $p \in \mathcal{P}$ is an arbitrary and continuous function of s. However, we require an oracle for computing $f_p(s)$ in polynomial time, for any value of s.
2. *Standard Power Functions* : Each processor $p \in \mathcal{P}$ satisfies the power function $f_p(s) = s^{\alpha_p}$, where $\alpha_p > 1$ is a small constant. This is the usual assumption in the speed-scaling literature. Note that, we denote by α the maximum power exponent, i.e., $\alpha = \max_{p\in\mathcal{P}}\{\alpha_p\}$.

2 Our Results

For the case where the heterogeneous power functions are convex, an algorithm has been proposed in [3] that returns a solution within an additive factor of ϵ far from the optimal and runs in time polynomial to the size of the instance and to $1/\epsilon$. This result generalizes the results of [1, 2] from the homogeneous setting to the heterogeneous one. However, the algorithm proposed in [3] is based on solving a configuration linear program using the Ellipsoid method. Given that this method may not be very efficient in practice, we focus on other approaches. We first propose a polynomial-time algorithm based on a compact linear programming formulation which solves the problem within any desired accuracy. Our algorithm does not need the use of the Ellipsoid method like in [3] and it applies for more general than convex power functions ; it is valid for a large family of continuous non-decreasing power functions.

The above result leaves open a natural question : *is it possible to generalize the flow-based approach used in [1, 2] for the homogeneous multiprocessor problem to the power-heterogeneous case ?* This question is interesting even for standard power functions of the form $f_p(s) = s^{\alpha_p}$. This last case is the goal of our second result. However, when power-heterogeneous processors are considered some structural properties of the optimal schedules of the homogeneous case are no more valid. For instance, in the heterogeneous setting, in any optimal schedule, the speed of a job is not necessarily unique, but it may change when parts of the same job are executed on different processors. A second difficulty comes from the fact that, while in the homogeneous case the processor on which a job is executed at a given time has no influence on the energy consumption, this is a crucial decision when scheduling on heterogeneous multiprocessors. We overcome these subtle difficulties and we propose a max-flow based algorithm which is rather more complicated than its homogeneous counterpart. In particular, we show that it produces a solution arbitrarily close to the optimal for jobs whose density is lower bounded by a small constant depending on the exponents of the power functions. The above assumption ensures that no job is processed with a speed less than one by any processor and allows us to solve the problem by performing maximum flow computations in a principled way. This assumption is reasonable in practice because the speed of a processor is multiple CPU cycles per second.

Our third result concerns the analysis of the well-known online algorithm AVR. Our analysis simplifies the analysis in [1] for the homogeneous case and allows us to extend it in the power-heterogeneous setting. Specifically, we prove that Heterogeneous-AVR is $((1+\epsilon)(\rho+1))$-competitive algorithm for arbitrary power functions, where ρ is the worst competitive ratio of the single-processor AVR algorithm among all processors. This turns to be $((1+\epsilon)(\alpha^\alpha 2^{\alpha-1}+1))$-competitive algorithm for standard power functions of the form $f_p(s) = s^{\alpha_p}$.

Références

[1] S. Albers, A. Antoniadis, and G. Greiner. On multi-processor speed scaling with migration. *J. Comput. Syst. Sci.*, 81(7) :1194–1209, 2015.

[2] E. Angel, E. Bampis, F. Kacem, and D. Letsios. Speed scaling on parallel processors with migration. In *Euro-Par*, volume 7484 of *LNCS*, pages 128–140. Springer, 2012.

[3] E. Bampis, A. V. Kononov, D. Letsios, G. Lucarelli, and M. Sviridenko. Energy efficient scheduling and routing via randomized rounding. In *FSTTCS*, volume 24 of *LIPIcs*, pages 449–460. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013.

[4] N. Bansal, D. P. Bunde, H.-L. Chan, and K. Pruhs. Average rate speed scaling. *Algorithmica*, 60(4) :877–889, 2011.

[5] N. Bansal, H.-L. Chan, and K. Pruhs. Speed scaling with an arbitrary power function. *ACM Transactions on Algorithms*, 9(2) :18, 2013.

[6] A. Gupta, S. Im, R. Krishnaswamy, B. Moseley, and K. Pruhs. Scheduling heterogeneous processors isn't as easy as you think. In *SODA*, pages 1242–1253, 2012.

[7] A. Gupta, R. Krishnaswamy, and K. Pruhs. Scalably scheduling power-heterogeneous processors. In *ICALP*, volume 6198, pages 312–323. Springer, 2010.

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