Speed-scaling with explorable uncertainty

Evripidis Bampis[1], Konstantinos Dogeas[1], Alexander Kononov[2], Giorgio Lucarelli[3], Fanny Pascual[1]

[1] Sorbonne Université, CNRS, LIP6, F-75005 Paris, France
{evripidis.bampis,konstantinos.dogeas,fanny.pascual}@lip6.fr
[2] Novosibirsk State University, Sobolev Institute of Mathematics, Novosibirsk, Russia
alvenko@math.nsc.ru
[3] Université de Lorraine, LCOMS, F-57000 Metz, France
giorgio.lucarelli@lcoms.fr

Mots-clés : *speed scaling, scheduling, explorable uncertainty.*

1 Introduction

Speed scaling is a well-known mechanism to handle energy consumption in computing systems. Given that the characteristics of the jobs may not be known in advance, many works in speed scaling adopt the frameworks of online optimization [3], or stochastic optimization [4]. However, in some situations it is possible to obtain the exact job characteristics at some extra cost. The operation that allows to obtain the exact value of some part of the input is called a *query*. Dürr et al. introduced recently the classical scheduling problem under explorable uncertainty in [1]. In their model, the uncertain information concerns the processing time of each job for which an upper bound is known in advance. It is possible to learn the exact processing time by querying at a price of a unit cost. If a job is executed without a query, then its execution time is equal to its upper bound. This assumption is motivated by the fact that a query could correspond to a code optimizer as mentioned in [1]. In that case, the code optimizer needs some workload to process the job and potentially reduces its workload. The upper bound on the workload of a job corresponds to the workload of the job when the code optimizer is not executed. Another possible application for this assumption is file compression. Contrary to the previous approaches in the field of explorable uncertainty [6], queries are executed directly on the machine running the jobs and so it is important to balance the time spent on queries and the time spent on the execution of jobs.

2 Model

In the speed-scaling model [2], the speed of a machine can be modified by the scheduler in order to save energy. In this work, we study the general case where the power is described by the function $P(s(t)) = s(t)^\alpha$, where $\alpha > 1$ is considered to be constant. Then, the energy consumption is computed as $E = \int P(s(t))dt$.

In the classical speed-scaling setting, each job j is characterized by a triple (r_j, d_j, w_j), which represents the *release time*, the *deadline* and the *workload* of the job respectively. The workload of j should be entirely executed in the interval $(r_j, d_j]$ which is called its *active interval*. In this paper, we augment this framework by introducing an *uncertainty* on the workload of the jobs. Here, the workload, w_j, is an upper bound rather than an exact value on the actual work needed for the completion of a job. The *exact load*, $w_j^* \leq w_j$, can be revealed to the algorithm only after executing a *query* of additional load $c_j \in (0, w_j]$. Hence, in our setting, each job is characterized by a quintuple $(r_j, d_j, c_j, w_j, w_j^*)$, where w_j^* is not known before the end of the

potential execution of the query. Note that, in the case where the query is not executed, the scheduler is obliged to execute the upper bound of the workload w_j.

We call the above enhanced model as *Query-Based Speed-Scaling* model (QBSS). The QBSS model is online by nature, since the value of w_j^* for each job j is revealed only after the potential execution of the query c_j. However, we distinguish between the *offline* and the *online* versions with respect to the classical scheduling setting. In the offline version, the entire input is known in advance, i.e., the total number of jobs to be scheduled, as well as their characteristics, except for the exact loads w_j^*. In the online version, the input becomes available to the algorithm over time : at time $t = r_j$, a new job j and its characteristics are revealed, except again for the exact loads w_j^*. In other words, the algorithm does not know in advance how many jobs it has to schedule, at which time they will arrive or what are their characteristics.

3 Results

In this work, we study an enhanced speed-scaling setting (called QBSS), where queries can be optionally executed in the system in order to reveal a more accurate value of the workload of jobs. The objective is energy minimization.There are two questions to answer for each job j in the QBSS model : whether the query will be done or not, and, if yes, how to partition the active interval of the job among the execution of its query and its exact load. Both decisions have a crucial impact on the speeds and on the consumed energy. For the first question, doing always the query leads to constant approximation algorithms, whereas never doing it leads to unbounded ratios. Note that the optimal algorithm has complete knowledge of the instance, including the exact loads. Hence, it can take this decision by comparing w_j and $c_j + w_j^*$. For the second question, the algorithm has to determine a *splitting point* $\tau_j \in (r_j, d_j)$, indicating the latest time at which the query has to finish execution and the earliest time at which the exact work of j may start its execution. We first consider the offline case where all jobs have a common release date, and we present a series of results based on different assumptions on the deadlines (e.g. common deadlines, power of two deadlines). For arbitrary deadlines, we obtain an approximation ratio of $(8\phi)^\alpha$ by rounding down the deadlines of the instance to the closest power of two. In addition we consider the online case, and we adapt the well-known AVR [2] and BKP [5] online algorithms for the classical speed-scaling setting to the QBSS model. The competitive ratios of our algorithms (AVRQ and BKPQ) have an additional multiplicative factor with respect to their version in the classical setting : a factor of 2^α for AVRQ in which the query is made for all jobs, and a factor of $(2 + \phi)^\alpha$ for BKPQ in which we decide to do the query or not in function of the value of the ratio c_j/w_j. Finally, we study the QBSS model on parallel identical machines and we propose a modification of the algorithm AVR(m), which turns to be $2^\alpha(2^{\alpha-1}\alpha^\alpha + 1)$-competitive with respect to energy.

Références

[1] Christoph Dürr , Thomas Erlebach , Nicole Megow and Julie Meißner. *Scheduling with Explorable Uncertainty.* ITCS 2018.

[2] F. Frances Yao, Alan J. Demers and Scott Shenker. *A Scheduling Model for Reduced CPU Energy.* 36th Annual Symposium on Foundations of Computer Science, 1995, Milwaukee, Wisconsin, USA.

[3] Susanne Albers. *Energy-efficient algorithms.* Commun. ACM 2010.

[4] Bruno Gaujal, Alain Girault and Stéphan Plassart. *Dynamic speed scaling minimizing expected energy consumption for real-time tasks.* J. Sched. 2020.

[5] Nikhil Bansal, Tracy Kimbrel and Kirk Pruhs. *Speed scaling to manage energy and temperature.* J. ACM 2007.

[6] Thomas Erlebach and Michael Hoffmann. *Query-Competitive Algorithms for Computing with Uncertainty.* Bulletin of the EATCS 2015.