Computer Science and Engineering

Master's thesis proposal - DATX60 (60 hec)

Pedro Matias 9302087433 mpedro@student.chalmers.se

September 24, 2014

1 Introduction

Nowadays, the amount of energy consumed in processors is huge. Energy is used all the time and all over the world, namely in servers, mobile devices or personal computers. Given our concern towards the global environment, and hence, ourselves, one should try to minimize the impacts of energy consumption. Computing solutions that use less energy is already a good step and it even helps battery supplied devices last longer. This thesis aims to find a better solution to the problem of minimizing the amount of energy consumed in variable speed processors, by finding a better schedule of the jobs to be executed. Variable speed processors can work at different speeds and, naturally, they consume less energy if they work at a lower speed. The idea is to ensure that all requested jobs are fully executed while keeping the amount of energy used as minimum as possible.

2 Problem overview

This work will be a variation of the following problem introduced by Yao, Demers and Shenker in [1]. Given a set of jobs, each with a release time, deadline and a work volume, the goal is to minimize the energy consumed when executing all jobs. There is a single variable speed processor associated with a power function $P(s) = s^{\alpha}(\alpha > 1)$, that must finish executing each job work volume not before its release time nor after its deadline. Otherwise, the schedule is not feasible. The energy consumed is given by the integral of the processor power over the time of execution.

Up until recently, only the *preemptive* case was considered, in which the execution of jobs may be interrupted for a limited amount of time, and restarted on either the same processor or a different one (*migratory* case).

2.1 Goal

In this thesis, one is interested in solving the speed-scaling introduced by Yao et al. [1] in a *non-preemptive* manner using more than one processor and, more specifically, using equal work volumes for all the jobs. The idea is to try to solve the problem without relying on linear programming, but instead by providing a *combinatorial* algorithm.

If all goes well, the thesis could even be extended to achieve a better solution to the problem without restrictions on job work volumes.

2.2 Notation and preliminaries

The release time, deadline and work volume are denoted (respectively) by r_j, d_j and w_j. The maximum and minimum work volumes of all jobs are denoted, respectively, by w_{max} and w_{min}.

In the multiprocessor setting the set of processors can either have equal (*homogeneous* case) or different energy comsumption rates (*heterogeneous* case). In the latter, each processor is assigned an independent constant α_i. Moreover, if each job j has different release time $r_{i,j}$, deadline $d_{i,j}$ and work volume $w_{i,j}$ on each processor i than we denote the set of processors by *fully heterogeneous*.

3 Previous work

The original problem, introduced by Yao et al in [1] was solved optimally with a polynomial-time algorithm that finds a preemptive schedule on a single processor. The non-preemptive version of the problem started being considered in [3] in which it was proved to be strongly NP-hard. In this paper it is also given a $2^{5\alpha-4}$-approximation algorithm. Later on, this result was improved to $2^{\alpha-1}(1+\epsilon)^{\alpha}\tilde{B}_{\alpha}$ (for any $\alpha < 114$) in [5] where $\tilde{B}_{\alpha}$ is the α-generalized version of the Bell number introduced by the authors. After that, Cohen-Addad et al. improved the bound to $(12(1+\epsilon))^{\alpha-1}$ in [2], for any $\alpha > 25$. In [6] the authors explore the idea of transforming an optimal preemptive schedule to a non-preemptive one. They also achieve an approximation ratio of $(1+\frac{w_{max}}{w_{min}})^{\alpha}$, by using as a lower bound the optimal preemptive solution. In the special case of equal work volumes, the ratio becomes constant: 2^α. The current best approximation ratio for this setting is $(1+\epsilon)^\alpha \tilde{B}_\alpha$ for all $\alpha < 77$, described in [8]. For the case when the jobs share the same work volume, optimal polynomial-time algorithms (based on dynamic programming) were given both by Angel et al. [9] and Huang and Ott [4]. Other special cases of the problem (concerning the structure of the life intervals of jobs) were also taken into account in [4]. For *agreeable* instances (see subsection 4.1, item 2a), the solution given by Yao et al. in [1] is optimal since it is never a preemptive schedule.

In the multi-processor environment there are several contributions for the preemptive and (non)migratory settings, but not so much for the non-preemptive version, whose current solutions rely on LP configurations. For the cases when both preemption and migration are allowed there are a few polynomial-time algorithms in the literature, namely in [10, 11, 12, 13]. In the fully heterogeneous version Bampis et al. [5] give a $OPT + \epsilon$ algorithm with complexity polynomial to $1/\epsilon$. The problem becomes strongly NP-hard when preemption is allowed but not migration [14]. In this article is described how a PTAS can be achieved on the special case of equal release dates and deadlines for all the jobs. Without this last restriction and, when the set of processors, is fully heterogeneous there exists an approximation algorithm of ratio $(1+\epsilon)^\alpha \tilde{B}_\alpha$ [5]. Regarding non-preemption and homogeneous processors, Cohen et al. [2] achieve an approximation ratio independent on the number of processors: $(5/2)^{\alpha-1}\tilde{B}_\alpha((1+\epsilon)(1+\frac{w_{max}}{w_{min}}))^\alpha$. This result was recently improved in [8] to $\tilde{B}_\alpha((1+\epsilon)(1+\frac{w_{max}}{w_{min}}))^\alpha$ and it is also valid for the fully heterogeneous setting.

4 Workflow

Given the fact that the majority of this work is theoretical and my inexperience in research, the proposed thesis can be unpredictabile both in time and accomplishment. Therefore, it is hard to establish proper deadlines for the tasks found in the workflow. Hence, some time estimates might be overestimated for safety reasons or underestimated given the circunstances. Overall, the plan is thought to last 40 weeks. (All the tasks below, naturally, expect a proof).

1. **(2 weeks)** Do a more detailed reading and analysis on the current literature (on both the same and similar problems) and eventual recently submitted articles.

2. **(30 weeks)** Design a *combinatorial* (as opposed to using LP methods) and if, possible, improve the best approximation ratio so far: $\tilde{B}_\alpha((1+\epsilon)(1+\frac{w_{max}}{w_{min}}))^\alpha$

   - General idea: solving each (or more than one) of the problem special instances described in subsection 4.1 and generalize results to main problem.

     (a) **(5 weeks)** Analyze the contribution of Huang and Ott in section 4 of [4] and try to adapt solution to more than 1 machine:

       i. **(2 weeks)** Prove that argument is only valid for 1 machine, otherwise problem does not need an approximation ratio.

       ii. **(3 weeks)** Try to adapt given solution to the main problem using an approximation algorithm and prove its ratio.

     (b) **(5 weeks)** If the previous approximation ratio is not satisfactory then think of different strategies for special instance 1:

       i. Begin with small number of machines (eg 2) and exploit with an increasing number of processors.

       ii. Generalize result for n machines. Give approximation algorithm with a ratio independent on n.

     (c) **(10 weeks)** If previous step is not possible or the ratio is unsatisfactory then develop strategies (with proofs) for special instances 2(a) and 2(b) keeping the number of machines unspecified:

       i. **(3 weeks)** try first using agreeable deadlines

         – **(2 week)** If too complicated, try both special instances 1 and 2(a)

       ii. **(3 weeks)** in case of no satisfactory result switch to special instance 2(b)

         – **(2 week)** If too complicated, try both special instances 1 and 2(b)

(d) **(10 weeks)** If an algorithm was developed for any of the special instances above, try to adapt it to solve more generalized problem.

   – If main problem can be solved, give proof that it works.
   – Otherwise, accept results and try to give insight on why the argument only works for the special structure of the input.

3. **(8 weeks)** Do the actual thesis writing

   • Collect all intermediary results and proofs and describe them in a simple way, so it is easier to understand, possibly simplifying proofs that are, eventually, more complex.

Given the unpredictability of the proposed workflow it is possible that, when finished, there is still some time left. In that case, try to solve online version of the problem (see subsection 4.1, item 3) or the following:

• Classify the problems in terms of complexity
• If problem is, at least, NP-hard then
  – Check if it is APX-hard
  – If not, try to give a (Q)PTAS algorithm

4.1 Special instances

Possible workarounds and ways of tackling the problem in case of getting stuck, involve trying to solve special cases of the problem:

1. Try specific number of machines, namely, 2 machines.
2. Jobs have life intervals of specific structure:
   (a) *agreeable deadlines*: no job has a deadline inferior to the one of a job whose release time is not greater than the latter.
   (b) *laminar*: for any pair of jobs, their life intervals intersection corresponds to either an empty set or the life interval of one of the jobs.
3. Online version: no knowledge on the release times and deadlines of jobs

5 Relevant courses from the master's programme (CS-ALL)

- Algorithms
- Advanced algorithms
- Discrete Optimization

References

[1] F. Yao, A. Demers and S. Shenker.
A Scheduling Model for Reduced CPU Energy. FOCS 1995, 374-382

[2] V. Cohen-Addad, Z. Li, C. Mathieu, I. Milis.
Energy-efficient algorithms for non-preemptive speed-scaling. Workshop on Approximation and Online Algorithms (WAOA) 2014

[3] Antonios Antoniadis, Chien-Chung Huang.
Non-Preemptive Speed Scaling. SWAT 2012

[4] Chien-Chung Huang, Sebastian Ott.
New Results for Non-Preemptive Speed Scaling. MFCS 2014

[5] E. Bampis, A. Kononov, D. Letsios, G. Lucarelli, and M. Sviridenko.
Energy Efficient Scheduling and Routing via Randomized Rounding. FSTTCS 2013

[6] E. Bampis, A. Kononov, D. Letsios, G. Lucarelli, I. Nemparis.
From Preemptive to Non-preemptive Speed-Scaling Scheduling. COCOON 2013

[7] E. Bampis, G. Lucarelli, I. Nemparis.
Improved Approximation Algorithms for the Non-preemptive Speed-scaling Problem. arXiv preprint arXiv:1209.6481 (2012)

[8] E. Bampis, D. Letsios, G. Lucarelli.
Speed-scaling with no Preemptions. arXiv preprint arXiv:1407.7654 (2014)

[9] E. Angel, E. Bampis, V. Chau.
Throughput maximization in the speed-scaling setting. CoRR 2013

[10] S. Albers, A. Antoniadis, G. Greiner.
On multi-processor speed scaling with migration: extended abstract. SPAA 2011

[11] E. Angel, E. Bampis, F. Kacem, D. Letsios.
Speed scaling on parallel processors with migration. Euro-Par 2012

[12] E. Bampis, D. Letsios, G. Lucarelli.
Green scheduling, flows and matchings. ISAAC 2012

[13] B. D. Bingham, M. R. Greenstreet.
Energy optimal scheduling on multiprocessors with migration. ISPA 2008

[14] S. Albers, F. Müller, S. Schmelzer.
Speed scaling on parallel processors. SPAA 2007