Theoretical Computer Science 661 (2017) 18–34

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Temperature aware online algorithms for minimizing flow time ☆

Martin Birks, Stanley P.Y. Fung *

Department of Informatics, University of Leicester, Leicester LE1 7RH, United Kingdom

ARTICLE INFO

Article history:
Received 19 February 2016
Received in revised form 6 September 2016
Accepted 31 October 2016
Available online 22 November 2016
Communicated by P. Krysta

Keywords:
Online algorithms
Scheduling
Temperature
Flow time

ABSTRACT

We consider the problem of minimizing the total flow time of a set of unit sized jobs in a discrete time model, subject to a temperature threshold. Each job has its release time and its heat contribution. At each time step the temperature of the processor is determined by its temperature at the previous time step, the job scheduled at this time step and a cooling factor. We show a number of lower bound results, including the case when the heat contributions of jobs are only marginally larger than a trivial threshold. Then we consider a form of resource augmentation by giving the online algorithm a higher temperature threshold, and show that the Hottest First algorithm can be made 1-competitive, while other common algorithms like Coolest First cannot. Finally we give some results in the offline case.

1. Introduction

Motivation. Green computing is not just trendy, but is a necessity. For example, data centers around the world consume an enormous amount of energy. Very often, this energy consumption and the associated issue of heat dissipation is the biggest factor affecting system design from data centers to handheld devices. Many ways to tackle the issue have been explored. Among them, the design of energy-efficient algorithms is an active area of research; we refer to [1] for an introduction.

In this paper we are interested in controlling the temperature of a microprocessor. Temperature is an important issue in processor architecture design: high temperature affects system reliability and lifespan, but a powerful processor inevitably comes with a high energy consumption and hence high temperature. It was proposed in [7] that, instead of slowing down the processor to control the temperature, one can use proper scheduling algorithms to help as well. Since then a number of papers [2,4,7] have worked on this model. We explain the model below.

The model. Time is split into discrete time steps. For an integer t, we refer to the time interval between the time instants t and $t+1$ as the *time step* t. A total of n jobs arrive. Each job J has a release time r_J, a heat contribution h_J, and a unit length processing time. All release times are integers. Thus each job fits into one time step. The temperature of a processor changes depending on the heat contribution of the jobs it executes, and a *cooling factor* $R > 1$: specifically, if at time t the temperature is τ_t and a job J is executed at this time step then the temperature at the next time step is given by $\tau_{t+1} = \frac{\tau_t + h_J}{R}$. This is a discrete approximation of the actual cooling which is a continuous time process governed by Faraday's Law, and the unit length jobs represent slices of processes given to a processor. The initial temperature can be set

☆ A preliminary version of this paper appeared in the 10th Annual Conference on the Theory and Applications of Models of Computation (TAMC), 2013.

* Corresponding author.
E-mail addresses: mbirks@gmail.com (M. Birks), pyf1@le.ac.uk (S.P.Y. Fung).

http://dx.doi.org/10.1016/j.tcs.2016.10.022
0304-3975/

at 0 without loss of generality. The temperature can never exceed the *temperature threshold* T. This can be set to 1 without loss of generality. A job J is therefore *admissible* at time t if $\tau_t + h_J \leq R$. This means that any job with $h_J > R$ can never be admissible; without loss of generality we thus assume all jobs have $h_J \leq R$.

One way of quantifying the performance of temperature-aware scheduling algorithms is to optimize some Quality of Service (QoS) measure subject to the temperature threshold. Arguably the most widely used QoS measure for processor scheduling is the *flow time* (or *response time*). The flow time of a job J is defined as the difference between its release time and its completion time. We can consider the total (or average) flow time of all jobs, or the maximum flow time. In this paper we focus on the total flow time.

The scheduling algorithm is *online*, i.e. it is not aware of jobs not yet released. This is of course a natural way to model jobs arriving at a microprocessor. We use the standard *competitive analysis* to analyze the effectiveness of online algorithms: an online algorithm $\mathcal{A}$ is c-competitive if the objective value returned by $\mathcal{A}$ (for a minimization problem) is at most c times that of an offline optimal algorithm $\mathcal{O}$, on all input instances.

There are several common and simple algorithms that can be used in this temperature model:

- Coolest First (CF): at every time step, schedule the coolest job among all admissible jobs, breaking ties arbitrarily.
- Hottest First (HF): at every time step, schedule the hottest job among all admissible jobs, breaking ties arbitrarily.
- FIFO: at every time step, schedule the job released earliest among all admissible jobs, breaking ties arbitrarily.

They all belong to a natural group of algorithms called *non-idling algorithms*, i.e., they do not idle when they have an admissible job pending. This is a weaker notion than that of *reasonable algorithms* as defined in [7], as reasonable algorithms are non-idling algorithms, but with stricter restrictions on which job must be scheduled.

Related work. Without temperature constraints, the flow time problem is well-studied. It is well known that the SRPT (shortest remaining processing time first) algorithm is 1-competitive with preemption. Since in our case all jobs are of unit length, there is no issue of preemption and all pending jobs have the same 'remaining' processing time; thus in the case without temperature constraints any non-idling algorithm is 1-competitive.

With temperatures, we are not aware of any prior work on flow time, although there were research on other objective functions. One of them is to maximize throughput when each job has a deadline. This was shown to be strongly NP-hard in the offline case [7], even if all jobs have identical release times and identical deadlines. They further showed that in the online case, all 'reasonable' algorithms are 2-competitive for $R = 2$ and this is optimal. When all jobs have identical release times and identical deadlines, CF was shown to have an approximation ratio between 1.3333 and 1.3889 [8]. The model was generalized to all range of R [4], the weighted jobs case [5] and the longer jobs case [6]. It was observed in [7] that their proof also showed strong NP-hardness for minimizing the maximum or total flow time.

Bampis et al. [2] considered the objective of minimizing the makespan on $m > 1$ processors when all jobs are released at time 0. They presented a generic 2ρ-approximate algorithm using a ρ-approximate algorithm for classical makespan scheduling as a subroutine, and a lower bound of $\frac{4}{3}$ on the approximability. For a single processor the algorithm gives an approximation ratio of 2. They also considered other objectives when there is no temperature threshold: they minimize the maximum temperature or the average temperature of the schedule instead (subject to a bound on the finishing time of jobs).

Our contributions. We consider three different cases in this paper:

(1) Bounded job heat: Since the problem is trivial without temperature constraints, it is tempting to believe that the problem is still tractable when the jobs are not very hot; we therefore consider limiting the maximum permissible heat of a job, h_{max}. When h_{max} is allowed to be exactly R then it can be trivially shown that no algorithm can give a bounded competitive ratio. On the other hand, if $h_{max} \leq R - 1$ then it can be easily shown that any algorithm is 1-competitive (details are in Section 3). Therefore we consider the case where $h_{max} = R - \epsilon$ for some $0 < \epsilon < 1$. Unfortunately it turns out that positive results remain rather unlikely. The problem remains NP-hard, and we show that the competitive ratio approaches infinity as ϵ approaches 0. We show even worse results for non-idling algorithms, that they have an unbounded competitive ratio for any $\epsilon < 1$.

(2) Increased temperature threshold: In view of the above, we instead give online algorithms a bit more power by allowing them to have a higher temperature threshold of $1 + \epsilon$ while the offline algorithm still has a threshold of 1. This can correspond to the case where, for example, new technologies make the system more resistant to higher temperatures. This kind of 'resource augmentation' concept was first introduced in [9,11].

Note that when $\epsilon \geq \frac{1}{R-1}$ a 1-competitive upper bound is trivial (see Section 4). In this model we can give a positive result: HF is 1-competitive if $\epsilon \geq \frac{R^2+R+1}{(R-1)(R+1)^2}$. We also show a number of lower bounds as in the bounded job heat model; in particular we show that CF cannot be even constant competitive given any non-trivial higher threshold. This is in stark contrast with the throughput case [7] where CF is optimal but HF can be shown to be not.

For easier illustration, consider the case $R = 2$: our results show that HF is 1-competitive whenever $\epsilon \geq \frac{7}{9}$ (any non-idling algorithm is trivially 1-competitive if $\epsilon \geq 1$), while there are no 1-competitive algorithms whenever $\epsilon < \frac{1}{8}$.