Online Algorithms for Maximizing Weighted Throughput of Unit Jobs with Temperature Constraints

Martin Birks[1], Daniel Cole[2], Stanley P. Y. Fung[1], and Huichao Xue[2]

[1] Department of Computer Science, University of Leicester, Leicester LE1 7RH, United Kingdom. {mb259,pyfung}@mcs.le.ac.uk
[2] Department of Computer Science, University of Pittsburgh, Pittsburgh, USA. {dcc20,hux10}@cs.pitt.edu

Abstract. We consider a temperature-aware online deadline scheduling model. The objective is to schedule a number of unit jobs, with release dates, deadlines, weights and heat contributions, to maximize the weighted throughput subject to a temperature threshold. We give an optimal randomized algorithm and another resource-augmented constant-competitive randomized algorithm for the problem. We also give almost tight upper and lower bounds for the multiple processor case.

Keywords: Online algorithms, scheduling, competitive analysis, temperature, resource augmentation.

1 Introduction

Background on Temperature. There has been a tremendous increase in processing power and packing density in microprocessors in recent years; thermal management in processors has therefore become an important issue in microarchitecture. High temperature affects reliability, incurs higher cooling costs, accelerates failure mechanisms such as electromigration and dielectric breakdown which can lead to permanent device failures, and worsens leakage which leads to increased power consumption [9]. Commonly, there is a critical *temperature threshold* that cannot be exceeded. Modern processors are usually 'multicore' where multiple processing units are put together on a single chip. One of the reasons for this architecture is that it allows a lower operating frequency (and hence temperature) when delivering the same computational power.

There has been quite a lot of recent work on a related problem of minimizing energy consumption in an online scheduling context (see e.g. [13, 3] among many others). However, optimizing for energy/power and for temperature require different techniques [3]. Arguably, thermal management is even more important than energy management in at least the following sense: as Bansal et al. [3] put it, "*if the processor in a mobile device exceeds its energy bound, then the battery is exhausted. If a processor exceeds its thermal threshold, it is destroyed.*"

However, algorithmic work on temperature issues is comparatively limited so far.

The temperature of a processor is related to its power usage and the cooling mechanism. Since the power usage is a convex function of processor speed [3], one possible way of managing temperature is to slow down the processor when the temperature is too high. This can be supported in the hardware level by *dynamic voltage scaling* (DVS). Online and offline thermal management algorithms using DVS have been designed and analysed in [3]. However, these algorithms are concerned with minimizing the maximum temperature, i.e., the peak temperature of the online algorithm is at most a constant factor higher than that of the offline optimal algorithm. Arguably, it is more useful to impose a fixed temperature threshold, and subject to this threshold, maximize some measure such as throughput; after all, the processor may die even if the threshold is exceeded by just a little. We consider a different model, described below.

Our model. Instead of slowing down the processor, we allow the algorithms to use heat characteristics information of the jobs to make scheduling decisions. This can be done at the operating systems level. Different types of jobs tend to have different patterns of heat contributions (e.g. a CPU-intensive job vs. a memory-intensive one). Recent empirical work [11] suggests that using this information can help produce better schedules with minimal performance degradation while maintaining a good thermal profile.

As a simplified scenario, we consider a set of unit-length jobs, where each job j has a release time $r(j)$, a deadline $d(j)$, a weight $w(j)$ and a heat contribution $h(j)$. All release times and deadlines are integers. At each integral time step, some jobs arrive, and the algorithm selects (at most) one job to schedule. The temperature of the processor will be increased due to the job execution, but there is also a cooling factor $R > 1$, so that the temperature is reduced by a factor of R after every time step. Thus, if the temperature at time t is τ, and the job scheduled at time $[t, t+1)$ has heat contribution h, then the temperature at time $t+1$ is $(\tau + h)/R$. We assume, without loss of generality, that the initial temperature is 0, and the temperature threshold of the processor is 1.

The algorithm is *online* meaning that it makes decisions without knowledge of future job arrivals. The objective is to maximize the weighted throughput, i.e. total weight of jobs completed before their deadlines, while not exceeding the temperature threshold. In a single processor environment, this can be denoted, using standard notation, by $1|\textit{online-}r_i, h_i, p_i = 1|\sum w_i U_i$. Such a model was first studied in [8] where the unit jobs represent unit slices in an OS scheduler.

In the multiple processor case, m denotes the number of processors. All processors have the same temperature threshold of 1 and the same cooling factor. The temperature of each processor is updated independently, i.e., the temperature of a processor at time $t+1$ is calculated based on the its temperature at time t, the heat contribution of the job it schedules at t and the cooling factor, in the same way as in the single processor case.

The performance of online algorithms is analysed using competitive analysis. An online algorithm $\mathcal{A}$ is *c-competitive*, if the value obtained by $\mathcal{A}$ on any instance

is at least $1/c$ that of the offline optimal algorithm. For randomized algorithms, we consider the expected value of $\mathcal{A}$, against an oblivious adversary who has to specify the input sequence without knowing the outcome of the random choices made by $\mathcal{A}$. See e.g. [5] for further details on competitive analysis and adversaries.

A job j with $h(j) > R$ can never be scheduled by any algorithm, because the temperature threshold will be immediately exceeded; hence we can assume that all jobs have $h(j) \leq R$. A job j with $h(j) = R$ is said to have the *maximum permissible heat*, or simply *full heat*. We will see that if the heat contribution is bounded away from full heat, more interesting and more useful results can be obtained. Moreover, full heat is not reasonable because it effectively means that once any job (of any positive heat contribution) has been scheduled, then no full heat job can be scheduled, no matter how long afterwards. In practice, after some finite amount of idle time, the processor is effectively at the ambient temperature and can run other jobs. Also, a full heat job will almost 'burn' the processor (starting from the ambient temperature) in just one 'quantum' of time, which is perhaps not really that reasonable.

Previous results. Without temperature constraints, the problem is known as *unit job scheduling* in the literature and received a lot of attention in recent years. Currently, the best deterministic upper and lower bounds on the competitive ratio are 1.828 [10] and 1.618 [7, 1], and the best randomized upper and lower bounds are 1.582 [6] and 1.25 [7].

The paper [8] introduced temperature issues into such a model. They considered the unweighted case, i.e., all jobs have the same weight so that the objective becomes maximizing the number of completed jobs, and that the cooling factor $R = 2$. They showed that computing the offline optimal schedule is NP-hard, and that all 'reasonable' algorithms are 2-competitive and this ratio is the best possible for deterministic algorithms. In [4] this result is extended to all values of R, giving an optimal competitive ratio that increases as R decreases (but is a constant for any fixed R).

Our results. In this paper we consider the case where jobs have different weights, to represent their different importance. We show that this weighted version does not admit constant competitive algorithms for the deteministic case. Then we consider two ways of improving the competitiveness: using randomization and using multiple processors. All of our results work for any $R > 1$.

For the randomized case, we give optimal $\Theta(\log W)$ bounds on the competitive ratio where W is the ratio of maximum to minimum job weights. Furthermore we give a constant competitive randomized algorithm when the heat contribution of all jobs are a constant factor $1-\epsilon$ away from the maximum permissible heat contribution. For any $\epsilon > 0$, the algorithm is $O(\log 1/\epsilon)$-competitive, and this competitive ratio is optimal up to a constant factor. This allows a tradeoff between the maximum permissible heat and the competitive ratio. The result can be interpreted in a resource augmentation setting where the online algorithm has a slightly higher temperature threshold than the offline optimal algorithm.

For the multiple processor case, we give an upper bound of $O(mW^{1/m})$ and a lower bound of $\Omega((mW)^{1/m})$ on the competitive ratio of deterministic algorithms. Thus for constant m this gives a tight bound of $\Theta(W^{1/m})$.

In the following, OPT denotes the optimal offline algorithm, $\mathcal{A}$ denotes an online algorithm and $|\mathcal{A}|$ denotes the value (weighted throughput) of the schedule produced by algorithm $\mathcal{A}$.

2 Deterministic case

We first make the simple observation that deterministic algorithms give poor results on a single processor. It is easy to see that no deterministic algorithms can have competitive ratio better than W, if jobs can have full heat. Consider two jobs J_1 and J_2, with $r(J_1) = 0, d(J_1) = 1, w(J_1) = 1, h(J_1) = R$, and $r(J_2) = 1, d(J_2) = 2, w(J_2) = W, h(J_2) = R$. When J_1 arrives, $\mathcal{A}$ must schedule it or else J_2 will not arrive and it will have an infinite competitive ratio. Once J_1 is started, $\mathcal{A}$ cannot schedule J_2. OPT schedules only J_2. Thus the competitive ratio is W.

It is also easy to give an $O(W)$-competitive algorithm: just ignore all the job weights and use any reasonable algorithms for the unweighted case given in [8, 4]. An algorithm is *reasonable* if, at any time step, (i) it never idles when there is at least one job that can be executed without violating the deadline and temperature constraints, and (ii) it executes a job that is not strictly dominated by another pending job, where a job i strictly dominates another job j if $h(i) \leq h(j)$ and $d(i) \leq d(j)$ and at least one of these two inequalities are strict. Examples of reasonable algorithms include the earliest deadline first, and the coolest first algorithms. It was shown in [4] that, for any constant $R > 1$, all reasonable algorithms are $O(1)$-competitive (with the constant depending on the value of R) with respect to the number of completed jobs. Since we lose a factor of at most W for the weight of each job, the competitiveness of $O(W)$ follows.

3 Randomized case: full heat

For the randomized case, we show that the competitive ratio is also unbounded in the case with full heat. We follow Yao's principle [12] and specify a probabilistic construction of the adversary and bound the competitive ratio against deterministic algorithms with inputs over this distribution.

Theorem 1. *Any randomized algorithm for the problem* $1|\textit{online-}r_i, h_i, p_i = 1|\sum w_i U_i$ *has competitive ratio* $\Omega(\log W)$.

Proof. Choose a large enough positive integer n, and let J_i $(1 \leq i \leq n)$ be a job with $r(J_i) = i - 1$, $d(J_i) = i$, $w(J_i) = 2^{i-1}$ and $h(J_i) = R$.

At $t = 0$, job J_1 arrives. Then at each time step i $(1 \leq i < n)$, with conditional probability $1/2$ the sequence stops, and with conditional probability $1/2$ the adversary continues to release job J_{i+1}. The probability is conditional on the fact

that the time step i is actually reached, i.e. the adversary is not stopped before. If J_n is released, then the sequence stops. Thus, we have a total of n different input sequences, appearing with probability $1/2$, $1/4$, $\ldots$, $1/2^{n-1}, 1/2^{n-1}$. (See Figure 1.)

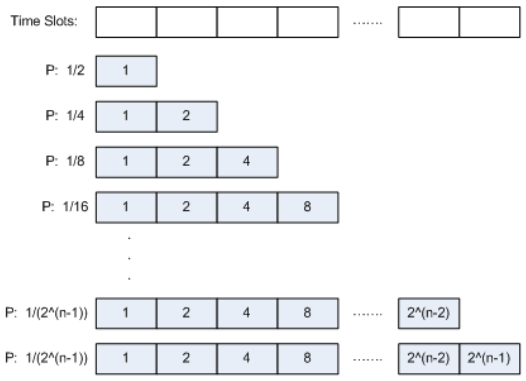

Fig. 1. Input distribution for the randomized lower bound.

Since all jobs have full heat, any algorithm can schedule at most one of those jobs. Hence, without loss of generality, we can restrict our attention to deterministic algorithms $\mathcal{A}_j$ of the following form: do not start the first $j-1$ jobs in the sequence and start the j-th, for some $1 \leq j \leq n$. Then $\mathcal{A}_j$ gets the value of job J_j if the adversary releases J_j, i.e. the adversary has not stopped after releasing the previous $j-1$ jobs. This happens with marginal probability $1/2^{j-1}$. Otherwise, if the adversary stopped before releasing J_j, no value is obtained. Thus $E[|\mathcal{A}_j|] = (1/2^{j-1})(2^{j-1}) = 1$. Note that this is independent of j. Thus the expected profit of any online algorithm is 1.

For OPT, it always schedules the last job released by the adversary; thus $E[|OPT|] = \sum_{i=1}^{n-1}(1/2^i)(2^{i-1}) + (1/2^{n-1})(2^{n-1}) = (n-1)/2 + 1 = (n+1)/2$. Therefore the competitive ratio is at least $(n+1)/2$. Since $W = 2^{n-1}$, we have proven a lower bound of $\Omega(\log W)$ on the competitive ratio. □

This competitive ratio can be attained by a randomized algorithm using the classify-and-random-select technique [2]. We assume the minimum weight is 1 and the maximum weight is W, and that this value of W is known in advance. We partition the weight range $[1, W]$ into $\lceil \ln W \rceil$ classes, with ranges $[1, e), [e, e^2), \ldots, [e^{\lfloor \ln W \rfloor - 1}, e^{\lfloor \ln W \rfloor}), [e^{\lfloor \ln W \rfloor}, W]$. In each class, the job weights differ by a factor of at most e. We randomly choose one of the classes and ignore all jobs not in this class. Then we run the $O(1)$-competitive algorithm for unweighted instances [4] for jobs in this class, ignoring the job weights.

Let $\mathcal{A}_i$ be the schedule of the online algorithm for the i-th class, and OPT_i be the optimal schedule for jobs in the i-th class. Let c_R be the competi-

tive ratio of the algorithm for the unweighted version of the problem. Then $|OPT_i| \leq c_R e|\mathcal{A}_i|$ since the algorithm is c_R-competitive for the number of jobs and at most a factor of e is lost on job weights. So $E[|\mathcal{A}|] = \sum |\mathcal{A}_i|/\lceil \ln W \rceil \geq (\sum |OPT_i|/c_R e)/\lceil \ln W \rceil \geq |OPT|/(c_R e \lceil \ln W \rceil)$. Thus the algorithm is $(c_R e \lceil \ln W \rceil)$-competitive.

Theorem 2. *There is a randomized $O(\log W)$-competitive algorithm for the problem $1|$online-$r_i, h_i, p_i = 1|\sum w_i U_i$.*

4 Randomized Case: Non-full Heat

Assuming that jobs do not have the maximum permissible heat, we can give better algorithms. For example, if all jobs have heat at most $R-1$, then it is easy to verify that the temperature never exceeds the threshold. We can generalise this idea to the case where all jobs have heat at most H, where $H < R$ with an arbitrarily small difference between H and R. We show how to give constant competitive randomized algorithms in such cases. In fact we give a tradeoff between the competitiveness and the maximum permissible heat. We first make this observation.

Lemma 1. *Suppose an algorithm runs jobs of heat at most h every $k \geq 1$ time slots, and keeps idle at other slots. If $h \leq R(1 - 1/R^k)$, then the temperature does not exceed 1 at any point.*

Proof. We claim that the temperature immediately after executing a job of heat at most h is at most 1. For the first such job, the temperature is at most $R(1 - 1/R^k)/R = 1 - 1/R^k$, so this clearly holds. By induction assume this is true after n jobs. Then the temperature before executing the $(n+1)$-th job is at most $1/R^{k-1}$, since there are $k-1$ idle slots. Hence the temperature after executing it is at most $(1/R^{k-1} + h)/R \leq 1$. □

Suppose all jobs have heat at most $H = R(1 - 1/R^k)$, for some $k \geq 1$. Our algorithm first uses an existing online algorithm for scheduling unit jobs without heat consideration. We can use any such existing algorithms, deterministic or randomized ([10, 6] among others), to produce such a schedule S. We then virtually create k schedules $S_1, S_2, \ldots, S_k$: S_1 schedules the same job as S during slots $t = 0, k, 2k, \ldots$, and stays idle at all other slots. Similarly S_2 schedules the same job as S during slots $t = 1, k+1, 2k+1, \ldots$, and stays idle at all other slots. In general, S_i follows S during slots of the form $t = (i-1)+qk$ for $q = 0, 1, 2, \ldots$. Each S_i schedules a job only every k slots, and each slot in S is "covered" by exactly one of the S_i's. By Lemma 1, each of the S_i's will never exceed the temperature threshold. The online algorithm chooses one of $S_1, S_2, \ldots, S_k$ in the beginning, each with probability $1/k$. Clearly this process can be done online.

Let $c < 2$ be the competitive ratio of the underlying unit job scheduling algorithm without heat consideration. The expected weighted throughput $E[|\mathcal{A}|]$ of our algorithm is equal to $1/k$ of the total weighted throughput of $S_1, \ldots, S_k$,

i.e., $E[|\mathcal{A}|] = \frac{1}{k}\sum_{i=1}^{k}|S_i|$. Let OPT' denote the optimal offline schedule for the same input instance but without heat considerations. Then we have:

$$|OPT| \le |OPT'|,$$
$$|S| = \sum_{i=1}^{k}|S_i|, \text{ and}$$
$$|OPT'| \le c|S|.$$

The first inequality is true because putting in temperature consideration clearly cannot result in a schedule with a larger value; the second is from the definition of the algorithm; and the last one comes from the competitiveness of the underlying unit job scheduling algorithm. Combining all these together, we have $|OPT| \le ckE[|\mathcal{A}|]$, and therefore the competitive ratio of our algorithm is $k \cdot c$.

Note that the maximum permissible heat $R(1-1/R^k)$ can be made arbitrarily close to R (the maximum permissible heat) for sufficiently large k. For heat limits not of the form $R(1-1/R^k)$, just take next higher limit of that form and the bound is only affected by a constant factor. We can therefore rephrase the result as follows: let $\epsilon = 1/R^k$. Then $k = \log_R(1/\epsilon) = O(\log(1/\epsilon))$ and therefore we have:

Theorem 3. *For any $0 < \epsilon \le 1/R$, the above algorithm schedules jobs with $H = R(1-\epsilon)$ with competitive ratio $O(\log(1/\epsilon))$.*

We can prove an almost matching lower bound:

Theorem 4. *For jobs with maximum heat $H = R(1-\epsilon) = R(1-1/R^k)$, and for large enough k, no randomized algorithm can have competitive ratio better than k.*

Proof. Let $n \ge 1$ be the largest integer such that $H/R^n + H/R > 1$. (It is required that $H \ge R/2$, i.e. $R^k \ge 2$ for such an n to exist.) Let p be a real number in (0,1), to be determined later. The proof is very similar to that of Theorem 1. Let J_i $(1 \le i \le n)$ be a job with $r(J_i) = i-1$, $d(J_i) = i$, $w(J_i) = 1/p^{i-1}$ and $h(J_i) = H$. At $t = 0$, job J_1 arrives. Then at each time step i $(1 \le i < n)$, with conditional probability $1-p$ the sequence stops, and with conditional probability p the adversary continues to release job J_{i+1}. The probability is conditional on the fact that the adversary is not stopped before time step i. If J_n is released, then the sequence stops.

Since $H/R^n + H/R > 1$, any algorithm can schedule at most one of those jobs. Hence, similar to Theorem 1, we only need to consider deterministic algorithms $\mathcal{A}_j$ that do not start the first $j-1$ jobs in the sequence and start the j-th, for each $1 \le j \le n$. Then $\mathcal{A}_j$ gets the value of job J_j if the adversary releases J_j, which happens with marginal probability p^{j-1}. Thus $E[\mathcal{A}_j] = (p^{j-1})(1/p^{j-1}) = 1$. For OPT, if the adversary stops at job J_j, it only schedules job J_j; thus

$$E[OPT] = \sum_{i=1}^{n-1}(p^{i-1}(1-p))(1/p^{i-1}) + (p^{n-1})(1/p^{n-1}) = (n-1)(1-p) + 1.$$

Therefore, the competitive ratio is at least $(n-1)(1-p)+1$. Choose p to be arbitrarily close to 0, then the ratio can be made arbitrarily close to n. The condition on $H/R^n + H/R > 1$ is equivalent to $n < \log_R(RH/(R-H))$. It follows that the lower bound is $\lceil \log_R(RH/(R-H)) \rceil - 1$. Since $H = R(1-\epsilon) = R(1-1/R^k)$, we have that the lower bound is equal to $\lceil \log_R(R^2(1-1/R^k)/(1/R^{k-1})) \rceil - 1 = \lceil \log_R(R^{k+1}-R) \rceil - 1$. This lower bound is equal to k for any $R > 1$ and sufficiently small ϵ, and thus the upper bound is optimal up to a constant factor of c. □

Resource Augmentation. Alternatively, we can use the above idea to give constant competitive algorithms even for jobs with full heat, but against a slightly weaker adversary; this follows the idea of resource augmentation. In our case, we compare an online algorithm with temperature threshold $1+\epsilon$ against an offline optimal algorithm with threshold 1.

Observe that if the heat contribution of all jobs in a certain schedule are multiplied by a factor λ, then the temperature at any point in the schedule is also multiplied by λ. Hence, when an online algorithm with temperature threshold $1+\epsilon$ is given an instance $\mathcal{I}$ with maximum heat $H \leq R$, it scales the heat contribution of all jobs by a factor of $1/(1+\epsilon)$, and applies the algorithm in Theorem 3 on this new instance. Since the algorithm in Theorem 3 produces a schedule not exceeding temperature 1, the schedule will not exceed temperature $1+\epsilon$ on the original instance. This transformed instance $\mathcal{I}'$ has maximum heat contribution $R/(1+\epsilon) = R(1-\frac{\epsilon}{1+\epsilon})$.

Let $\mathcal{A}(\mathcal{I})$ and $\mathcal{A}(\mathcal{I}')$ be the schedule returned by the online algorithm on $\mathcal{I}$ and $\mathcal{I}'$ (with temperature threshold $1+\epsilon$ and 1 respectively). Similarly let $OPT(\mathcal{I})$ and $OPT(\mathcal{I}')$ be the schedule produced by OPT on $\mathcal{I}$ and $\mathcal{I}'$ (with a temperature threshold of 1 in both cases). We have $|OPT(\mathcal{I})| \leq |OPT(\mathcal{I}')|$ as the jobs in $\mathcal{I}$ are hotter than those in $\mathcal{I}'$; $|\mathcal{A}(\mathcal{I})| = |\mathcal{A}(\mathcal{I}')|$ as they are the same schedule; and $|OPT(\mathcal{I}')| \leq O(\log(1/\frac{\epsilon}{1+\epsilon}))|\mathcal{A}(\mathcal{I}')|$ as this is the competitive ratio given in Theorem 3. Putting these together it therefore follows that $|OPT(\mathcal{I})| \leq O(\log(1/\frac{\epsilon}{1+\epsilon}))|\mathcal{A}(\mathcal{I})| = O(\log(1+1/\epsilon))|\mathcal{A}(\mathcal{I})|$. This represents a tradeoff between competitiveness and extra resource (on temperature threshold).

5 Multiple Processors

We have seen that in the single processor case, no deterministic algorithms can have a bounded competitive ratio. In this section we consider the multiprocessor (and deterministic) case, and shows that the bounds can be improved, depending on the number of processors m.

Theorem 5. *No deterministic algorithm can be better than $(mW)^{1/m}$-competitive for $P|$online-$r_i, h_i, p_i = 1|\sum w_i U_i$, when $W \geq m^{m-1}$.*

Proof. Fix a deterministic algorithm $\mathcal{A}$. Consider a sequence of jobs J_i for $1 \leq i \leq m$, where $r(J_i) = i-1, d(J_i) = i$ and $h(J_i) = R$. The weights of the jobs are given by $w(J_1) = 1$, and $w(J_i) = (c-1)\sum_{j=1}^{i-1} w(J_j)$ for $i > 1$ where $c > 1$

is some value to be chosen later. Each J_i is released successively. Note that the job J_i cannot be scheduled on any processor that has already scheduled another job J_j for $j < i$ as all of the jobs have full heat contribution. If $\mathcal{A}$ chooses not to schedule some job J_k on any processor then the subsequent jobs will not be released. In this case $\mathcal{A}$ have scheduled the first $k-1$ jobs and so has a weighted throughput of $\sum_{i=1}^{k-1} w(J_i)$. OPT will schedule all k jobs and so have a weighted throughput of $\sum_{i=1}^{k-1} w(J_i) + w(J_k)$. As $w(J_k) = (c-1)\sum_{i=1}^{k-1} w(J_i)$, this gives OPT a weighted throughput of $c\sum_{i=1}^{k-1} w(J_i)$ giving a competitive ratio of c in this case.

Otherwise $\mathcal{A}$ does not miss any of the J_i jobs, $1 \leq i \leq m$. This means that $\mathcal{A}$ have scheduled exactly one job on each of its processors. At time m we then release m jobs $X_1, \ldots, X_m$ that have heat contributions of R, tight deadlines of $m+1$ and weights of $(c\sum_{i=1}^{m} w(J_i))/m$. $\mathcal{A}$ will be too hot to schedule any of them, while OPT can skip all the J_i and schedule all $X_1, X_2, \ldots, X_m$ one on each processor. This gives OPT a weighted throughput of $c\sum_{i=1}^{m} w(J_i)$ while $\mathcal{A}$ has a weighted throughput of $\sum_{i=1}^{m} w(J_i)$ which again gives a competitive ratio of c.

Each job J_i for $i > 1$ has weight $(c-1)\sum_{j=1}^{i-1} w(J_j)$ and $w(J_1) = 1$. Solving the recursion we get $w(J_i) = c^{i-1} - c^{i-2}$. Thus $w(X_i) = c(\sum_{i=2}^{m}(c^{i-1}-c^{i-2})+1)/m = c^m/m$. The minimum job weight is 1 and the maximum weight job is either J_m or one of the X_i's. This gives a weight of the heaviest job (and therefore a ratio of the heaviest to lightest job) of:

$$W = \max\{c^{m-1} - c^{m-2}, \frac{c^m}{m}\}$$

We have that $c^{m-1} - c^{m-2} \leq c^m/m$ as long as $cm \leq c^2 + m$, which means in this case $W = c^m/m$. Setting $c = (mW)^{1/m}$ therefore gives the desired competitive ratio. With this value of c, the condition $cm \leq c^2 + m$ is satisfied if, for example, $W \geq m^{m-1}$. □

We can prove a deterministic upper bound of $O(mW^{1/m})$. The idea is similar to the randomized algorithm in Section 3. Split the job weights into m classes $[1..W^{1/m}), [W^{1/m}..W^{2/m}), ..., [W^{(m-1)/m}..W]$. Each processor deals only with jobs in one class. The processor then ignore the weights and use the $O(1)$-competitive 1-processor algorithm in [4] to schedule the jobs.

Theorem 6. *The above algorithm has a competitive ratio of $O(mW^{1/m})$ for $P|\text{online-}r_i, h_i, p_i = 1|\sum w_i U_i$.*

Proof. Let $\mathcal{A}_i$ be the online schedule produced by processor i (job weight class i). Let $OPT_{i,j}$ be the OPT schedule of processor j filtered to include jobs in class i only. For each such schedule j, the $O(1)$-competitiveness of the algorithm in [4] implies that the number of jobs completed by $\mathcal{A}_i$ is at least $1/c_R$ that in $OPT_{i,j}$, for some constant c_R. Thus the value of $\mathcal{A}_i$ is at least $1/(c_R W^{1/m})$ that of $OPT_{i,j}$. Summing over all i and j we have that $|\mathcal{A}| \geq 1/(c_R m W^{1/m})|OPT|$. Hence the competitive ratio is $O(mW^{1/m})$. □

6 Conclusion

In this paper we give almost optimal bounds for the online unit job scheduling problem for weighted throughput with temperature constraints. Further work will include closing these bounds and also investigating the use of randomized algorithms or resource augmentation in the multiprocessor case.

Acknowledgement. We thank Kirk Pruhs for useful discussions.

References

1. Nir Andelman, Yishay Mansour, and An Zhu. Competitive queueing policies for QoS switches. In *Proceedings of 14th ACM-SIAM Symposium on Discrete Algorithms*, pages 761–770, 2003.
2. Baruch Awerbuch, Yair Bartal, Amos Fiat, and Adi Rosen. Competitive non-preemptive call control. In *Proceedings of 5th ACM-SIAM Symposium on Discrete Algorithms*, pages 312–320, 1994.
3. Nikhil Bansal, Tracy Kimbrel, and Kirk Pruhs. Dynamic speed scaling to manage energy and temperature. *Journal of the ACM*, 54(1), 2007.
4. Martin Birks and Stanley P. Y. Fung. Temperature aware online scheduling with a low cooling factor. In *Proceedings of 7th Annual Conference on Theory and Applications of Models of Computation*, volume 6106 of *Lecture Notes in Computer Science*, pages 105–116, 2010.
5. Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, New York, 1998.
6. Francis Y. L. Chin, Marek Chrobak, Stanley P. Y. Fung, Wojciech Jawor, Jiri Sgall, and Tomas Tichý. Online competitive algorithms for maximizing weighted throughput of unit jobs. *Journal of Discrete Algorithms*, 4(2):255–276, 2006.
7. Francis Y. L. Chin and Stanley P. Y. Fung. Online scheduling with partial job values: Does timesharing or randomization help? *Algorithmica*, 37(3):149–164, 2003.
8. Marek Chrobak, Christoph Dürr, Mathilde Hurand, and Julien Robert. Algorithms for temperature-aware task scheduling in microprocessor systems. In *Proceedings of 4th International Conference on Algorithmic Aspects in Information and Management*, pages 120–130, 2008.
9. A. Coskun, T. Rosing, and K. Whisnant. Temperature aware task scheduling in MPSoCs. In *Proc. Conference on Design, Automation and Test in Europe*, pages 1659–1664, 2007.
10. Matthias Englert and Matthias Westermann. Considering suppressed packets improves buffer management in QoS switches. In *Proceedings of 18th ACM-SIAM Symposium on Discrete Algorithms*, pages 209–218, 2007.
11. J. Yang, X. Zhou, M. Chrobak, Y. Zhang, and L. Jin. Dynamic thermal management through task scheduling. In *IEEE International Symposium on Performance Analysis of Systems and Software*, pages 191–201, 2008.
12. Andrew C.-C. Yao. Probabilistic computations: Toward a unified measure of complexity. In *Proceedings of 18th IEEE Symposium on Foundations of Computer Science*, pages 222–227, 1977.
13. F. F. Yao, A. J. Demers, and S. Shenker. A scheduling model for reduced CPU energy. In *Proceedings of 36th IEEE Symposium on Foundations of Computer Science*, 1995.