Speed Scaling to Manage Energy and Temperature

NIKHIL BANSAL AND TRACY KIMBREL

IBM T.J. Watson Research Center

AND

KIRK PRUHS

University of Pittsburgh

Abstract. Speed scaling is a power management technique that involves dynamically changing the speed of a processor. We study policies for setting the speed of the processor for both of the goals of minimizing the energy used and the maximum temperature attained. The theoretical study of speed scaling policies to manage energy was initiated in a seminal paper by Yao et al. [1995], and we adopt their setting. We assume that the power required to run at speed s is $P(s) = s^\alpha$ for some constant $\alpha > 1$. We assume a collection of tasks, each with a release time, a deadline, and an arbitrary amount of work that must be done between the release time and the deadline. Yao et al. [1995] gave an offline greedy algorithm YDS to compute the minimum energy schedule. They further proposed two online algorithms Average Rate (AVR) and Optimal Available (OA), and showed that AVR is $2^{\alpha-1}\alpha^\alpha$-competitive with respect to energy. We provide a tight α^α bound on the competitive ratio of OA with respect to energy.

We initiate the study of speed scaling to manage temperature. We assume that the environment has a fixed ambient temperature and that the device cools according to Newton's law of cooling. We observe that the maximum temperature can be approximated within a factor of two by the maximum energy used over any interval of length $1/b$, where b is the cooling parameter of the device. We define a speed scaling policy to be cooling-oblivious if it is simultaneously constant-competitive with respect to temperature for all cooling parameters. We then observe that cooling-oblivious algorithms are also constant-competitive with respect to energy, maximum speed and maximum power. We show that YDS is a cooling-oblivious algorithm. In contrast, we show that the online algorithms OA and AVR are not cooling-oblivious. We then propose a new online algorithm that we call BKP. We show that BKP is cooling-oblivious. We further show that BKP is e-competitive with respect to the maximum speed, and that no deterministic online algorithm can have a better competitive ratio. BKP also has a lower competitive ratio for energy than OA for $\alpha \geq 5$.

K. Pruhs was supported in part by NSF grants CCR-0098752, CNS-0123705, CNS-0325353, CCF-0448196, CCF-0514058, and IIS-054531.

Authors' addresses: N. Bansal and T. Kimbrel, IBM T. J. Watson Research Center, PO Box 209, Yorktown Heights, NY 10598, e-mail: {nikhil,kimbrel}@us.ibm.com; K. Pruhs, Department of Computer Science, 210 South Bouquet Street, Sennott Square Building, Room 6415, University of Pittsburgh, Pittsburgh , PA 15260, e-mail: kirk@cs.pitt.edu.

Finally, we show that the optimal temperature schedule can be computed offline in polynomial-time using the Ellipsoid algorithm.

Categories and Subject Descriptors: C.4 [**Performance of Systems**]: *Performance attributes*; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Sequencing and scheduling*

General Terms: Algorithms, Performance

Additional Key Words and Phrases: Speed scaling, voltage scaling, power management

ACM Reference Format:

Bansal, N., Kimbrel, T., and Pruhs, K. 2007. Speed scaling to manage energy and temperature. J. ACM 54, 1, Article 3 (March 2007), 39 pages. DOI = 10.1145/1206035.1206038 http://doi.acm.org/10.1145/1206035.1206038

1. *Introduction*

1.1. MOTIVATION. The energy consumption rate of computing devices has increased exponentially for several decades. Since the early 1970s, power densities in microprocessors have doubled every three years [Skadron et al. 2003]. This increased power usage poses two types of difficulties:

Energy Consumption. As energy is power integrated over time, supplying the required energy may become prohibitively expensive, or even technologically infeasible. This is a particular difficulty in devices that rely on batteries for energy, and will will become even more critical since battery capacities are increasing at a much slower rate than energy consumption.

Temperature. The energy used in computing devices is in large part converted into heat. Cooling costs are rising exponentially along with energy consumption and threaten the computer industry's ability to deploy new systems [Skadron et al. 2003]. In fact, in May 2004 Intel publicly acknowledged that it had hit a "thermal wall" on its microprocessor line. Intel scrapped the development of its Tejas and Jayhawk chips in order to rush to the marketplace a more power efficient chip technology. Designers said the escalating heat problems were so severe that they threatened to cause chips to fracture [Markoff 2004]. Apple was unable to develop a G5 laptop due to inadequate heat management in the IBM PowerPC chips, and in the summer of 2005 Apple announced that it was switching to cooler Intel chips [Apple 2005].

These factors have resulted in power becoming a first-class design constraint for modern computing devices [Mudge 2001]. There is an extensive literature on power management in computing devices. Overviews have been given by Brooks et al. [2000], Mudge [2001], and Tiwari et al. [1998].

Both in academic research and practice, voltage/frequency/speed scaling is the dominant technique for power management. Speed scaling involves dynamically changing the speed of the processor. Current microprocessors from AMD, Intel and Transmeta allow the speed of the microprocessor to be set dynamically. Some modern processors are able to sense their own temperatures. Such a device can be slowed down or shut down so that its temperature will stay below its thermal threshold [Skadron et al. 2003].

There is an inherent tradeoff between power reduction and performance; in general, when more power is available, better performance can be achieved. As a result,

it is generally proposed that power reduction techniques be preferentially applied during times when performance is less critical. It is likely that in the future these policies will necessarily incorporate information provided by applications and high-level resource managers in operating systems [Ellis 1999]. This will require policies to determine how essential performance is at any given time, and how to apply a particular power reduction technique.

Our goal here is to make a formal study of a particular power reduction technique, namely speed scaling, in a specific setting, namely scheduling tasks with deadlines, to manage either temperature or energy.

1.2. BACKGROUND. The starting point for our investigations is the seminal paper by Yao et al. [1995], in which the authors proposed formulating speed scaling problems as scheduling problems. That is, the setting is a collection of tasks, and a schedule specifies not only which task to run at each time, but also the speed at which to run the selected task. Each task i has a release time r_i at which it enters the system, a deadline d_i, and an amount of work w_i that must be performed between times r_i and d_i to complete the task.

In some settings, for example, the playing of a video or other multimedia presentation, there may be natural deadlines for the various tasks imposed by the application. In other settings, the system may impose deadlines to better manage tasks or insure a certain quality of service to each task [Buttazzo 1997]. Yao et al. [1995] assumed that tasks can be preempted, that is, the device can suspend the execution of a task, and later resume the task from the point of suspension. Preemption is a necessary feature to obtain reasonable performance in a system with tasks with widely varying work.

Yao et al. [1995] assumed the existence of a function $P(s)$ that specifies the power used when the device is run at speed s. They assumed that $P(s) = s^\alpha$ for some $\alpha > 1$. The key fact about such a function is that it is strictly convex, that is, the slower a task is run, the less energy is used to complete that task. This is a generalization of the well-known cube-root rule for CMOS devices, which states that the speed s is roughly proportional to the cube-root of the power P, or equivalently, $P(s) = s^3$. CMOS is likely to remain the dominant technology for the near-term future. Power in CMOS devices has three components: switching loss, leakage loss, and short circuit loss [Brooks et al. 2000; Mudge 2001]. Switching loss is the energy consumption due to charging and discharging gates. The switching loss is roughly proportional to sV^2, where s is the speed (clock frequency), and V is the voltage. V and s are not independent; there is a minimum voltage required to drive the microprocessor at a desired frequency, and this minimum voltage is approximately proportional to the frequency [Brooks et al. 2000]. Hence, one can conclude that switching loss is roughly proportional to the cube of the speed. Currently, switching loss is responsible for the majority of the energy used by computing devices [Mudge 2001].

Yao et al. [1995] then studied the problem of minimizing the total energy used subject to the deadline feasibility constraints. This is always possible under the assumption that the processor can run at any speed. They gave an optimal offline greedy polynomial-time algorithm, which we call YDS. The YDS schedule is simultaneously optimal for all strictly convex speed-to-power functions. The YDS schedule can also be seen to minimize the maximum speed and hence the maximum power. Yao et al. [1995] also proposed two simple online algorithms. In the

online version of the problem, the scheduler learns about a task only at its release time. At this time, the scheduler also learns the exact work requirement and the deadline of the task. The online algorithm Average Rate (AVR) runs each task i at speed $w_i/(d_i - r_i)$. The online algorithm Optimal Available (OA) schedules the unfinished work optimally (say using YDS) under the assumption that no more tasks will arrive. Yao et al. [1995] state a lower bound of α^α on the competitive ratio for AVR and OA. They prove, using a rather complicated spectral analysis, that the competitive ratio of AVR is at most $2^{\alpha-1}\alpha^\alpha$. They also show that an online algorithm cannot in general construct an optimal energy schedule, even for an instance that contains only two tasks.

1.3. OUR CONTRIBUTIONS. Yao et al. [1995] did not explicitly prove that the YDS algorithm produces the most energy efficient feasible schedule. To the best of our knowledge, no such proof has appeared in the literature. We show in Section 2 that the correctness of YDS is an elegant consequence of the well-known KKT optimality conditions for convex programs. This illustrates the utility of the KKT optimality conditions in power management problems.

In Section 3, we extend the results Yao et al. [1995] on online algorithms for energy minimization. We give explicit instances that show that the competitive ratios of AVR and OA are at least α^α. We then provide a tight α^α bound on the competitive ratio of OA using a potential function argument.

We then turn our attention to speed scaling to manage temperature. To our knowledge, this is the first theoretical investigation of this area. We first need to model the cooling behavior of a device. Cooling is a complex phenomenon that cannot be captured completely accurately by any simple model [Sergent and Krum 1998]. For tractability, we require a simple first-order approximation. Our key assumptions are that heat is lost via conduction, and the ambient temperature of the environment surrounding the device is constant. This is likely a reasonable first-order approximation in some, but certainly not all, settings. Then we appeal to Newton's law of cooling, which states that the rate of cooling is proportional to the difference in temperature between the object and the ambient environmental temperature. Without loss of generality, we may assume that the temperature scale is translated so that the ambient temperature is zero. If we assume that the net change in temperature is the sum of the decrease due to cooling as described above and an increase proportional to the electrical power applied to the device, a first-order approximation for rate of change $T'(t)$ of the temperature $T(t)$ at time t is then given by the equation:

$$T'(t) = aP(t) - bT(t),$$

where $P(t)$ is the supplied power at time t, and a and b are constants [Sergent and Krum 1998; Crusoe 2002]. We call $b \geq 0$ the *cooling parameter* of the device.

We then consider the relationship between temperature and energy. Temperature and energy are physical variables with quite different properties. If the processor in a mobile device exceeds its energy bound, then the battery is exhausted. If a processor exceeds it thermal threshold, it is destroyed. Power management schemes for conserving energy focus on reducing *cumulative* power, while power management schemes for reducing temperature must focus more on *instantaneous* power. Power management schemes designed to conserve energy may not perform well when the goal is to reduce temperature. In fact, many low-power techniques are reported to have little or no effect on temperature [Skadron et al. 2003]. Temperature-aware

design is therefore an area of study distinct from, albeit related to, energy aware design [Skadron et al. 2003].

In Section 2, we consider the relationship between temperature and energy. One consequence of Newton's law is that an un-powered device cools by a constant fraction every $\frac{1}{b}$ time units. This leads us to observe that the maximum temperature is within a factor of two of a times the maximum energy used over any interval of length $\frac{1}{b}$. If $b = 0$, then no energy is ever dissipated from the device and the maximum temperature is the final temperature, which is a times the energy used. Thus, when $b = 0$, the temperature minimization problem is equivalent to the energy minimization problem. In the limit as the cooling parameter b approaches $+\infty$, the maximum temperature is essentially determined by the maximum energy over an infinitesimal interval, and thus the minimization problem intuitively becomes equivalent to the problem of minimizing the maximum power (or equivalently minimizing the maximum speed). The energy minimization problem, when the speed-to-power parameter α is ∞, is also equivalent to minimizing the maximum power.

Because of this exponential cooling, it seems difficult to reason about temperature. However, the above observation about approximating temperature by energy used over some interval makes reasoning about approximate temperature much easier than reasoning about exact temperature. This observation also motivates us to define what we call a *cooling-oblivious* speed scaling algorithm, which is an algorithm that is simultaneously $O(1)$-approximate for minimizing the maximum temperature for all cooling parameters $b \geq 0$. Thus, a cooling-oblivious algorithm is $O(1)$-approximate for total energy. Further, if the schedule produced by a cooling oblivious algorithm does not depend on the value of α and the cooling parameter b, as is the case for all the algorithms that we consider, then the algorithm is also $O(1)$-approximate for minimizing the maximum speed.

In Section 2, we show that while the YDS schedule may not be optimal for temperature, it is cooling-oblivious. More precisely, we show that YDS is 20-approximate with respect to temperature for all cooling parameters b. This constructively shows that there are schedules that are $O(1)$-approximate with respect to both of the dual criteria of temperature and energy.

We then turn to online speed scaling to minimize the maximum temperature that a device ever reaches, again subject to the constraint that all tasks finish by their deadlines. In Section 3, we show that online algorithms OA and AVR, proposed by Yao et al. [1995] in the context of energy management, are not $O(1)$-competitive with respect to temperature. That is, these algorithms are not cooling-oblivious. Recall that both OA and AVR are $O(1)$-competitive with respect to energy. This demonstratively illustrates the observation from practice that power management techniques that are effective for managing energy may not be effective for temperature. One intuitive speed scaling algorithm to manage temperature is to run at the minimum constant speed that will allow all tasks to finish by their deadline. Surprisingly, we show that this algorithm is also not $O(1)$-competitive with respect to temperature.

We propose a new online speed scaling algorithm that we call BKP. In Section 4, we show that BKP is cooling-oblivious. That is, BKP is simultaneously $O(1)$-competitive for total energy, maximum temperature, maximum power, and maximum speed. We show that the competitive ratio for BKP with respect to energy is at most $2(\alpha/(\alpha - 1))^{\alpha} \exp(\alpha)$. Note that for $\alpha \geq 2$, this competitive ratio

is at most $8\exp(\alpha)$. The competitive ratio of BKP is better than the competitive ratios of OA and AVR for $\alpha \geq 5$. We show that the competitive ratio of BKP with respect to temperature is at most $2^{\alpha+1}\exp(\alpha)(6(\alpha/(\alpha-1))^{\alpha}+1)$. We show that BKP is e-competitive with respect to maximum speed, or equivalently, that BKP is $\exp(\alpha)$-competitive with respect to maximum power. We further show that BKP is optimally competitive with respect to both maximum speed and maximum power, that is, no deterministic algorithm can have better competitive ratios. As a consequence of this, one can conclude that if a deterministic online algorithm is $O(k^{\alpha})$-competitive with respect to energy, then the value of the constant k has to be at least e. Thus, the competitive ratio of BKP, with respect to energy, is optimal up to a multiplicative constant for deterministic online algorithms.

We finally turn our attention to offline temperature management. We show in Section 5 that this problem can be posed as a convex optimization problem. Convex optimization problems can be solved arbitrarily precisely in polynomial time using the Ellipsoid algorithm if one can compute a separating hyperplane for a violated constraint in polynomial time. To accomplish this for our temperature problem, we show that the key subproblem is determining the maximum work that can be accomplished during a fixed time period with a fixed starting and a fixed ending temperature. We show how to use techniques from calculus of variations to solve this subproblem. As a consequence of this, we reveal some structure of the optimal temperature schedule: during any maximal time period which contains no release time or deadline, the temperature curve is either (1) an Euler–Lagrange curve, or (2) rises to the thermal threshold along an Euler–Lagrange curve, stays at the thermal threshold for some amount of time, and then falls along an Euler–Lagrange curve.

1.4. RELATED RESEARCH. A naive implementation of YDS runs in time $O(n^3)$. This can be improved to $O(n^2)$ if the intervals have a tree structure [Li et al. 2006a]. Recently, Li et al. [2006b] gave an $O(n^2 \log n)$ implementation for the general case. For hard real-time tasks with fixed priorities, Yun and Kim [2003] show that it is NP-hard to compute a minimum-energy schedule. They also give a fully polynomial time approximation scheme for the problem. Kwon and Kim [2003] give a polynomial time algorithm for the case of a processor with discrete speeds. Li and Yao [2005] give an algorithm with running time $O(d \cdot n \log n)$ where d is the number of speeds.

Irani et al. [2003] study online speed scaling algorithms to minimize energy usage for a device that also has a sleep state. They give an offline polynomial-time 2-approximate algorithm. Irani et al. [2003] also give an online algorithm A that uses, as a subroutine, an algorithm B for pure speed scaling. If B is additive and monotone (as AVR, OA and BKP are) then A is $\max(\alpha(R+1)+2, 4)$-competitive, where R is the competitive ratio of B. Thus, our analysis of OA and BKP improve the best known competitive ratio for this problem.

A survey on algorithmic problems in power management was given by Irani and Pruhs [2005].

1.5. DEFINITIONS. In this Section, we recap the definitions introduced so far, and introduce some definitions that we will use throughout the article. We also make some observations about these definitions.

A problem instance consists of n tasks. Task i has a release time r_i, a deadline $d_i > r_i$, and work $w_i > 0$. In the online version of the problem, the scheduler learns about a task only at its release time; at this time, the scheduler also learns

the exact work requirement and the deadline of the task. We assume that time is continuous. A schedule specifies for each time a task to be run and a speed at which to run the task. The speed is the amount of work performed on the task per unit time. A task with work w run at a constant speed s thus takes $\frac{w}{s}$ time to complete. More generally, the work done on a task during a time period is the integral over that time period of the speed at which the task is run. A schedule is *feasible* if for each task i, work at least w_i is done on task i during $[r_i, d_i]$. Note that the times at which work is performed on task i do not have to be contiguous. If a task is run at speed s, then the power is $P(s) = s^\alpha$ for some constant $\alpha > 1$.

The energy used during a time period is the integral of the power over that time period. In the energy version of our problem, the objective is to minimize the total energy $E[S]$ used by the schedule S.

We now turn to temperature. We assume without loss of generality that the initial temperature is 0. We assume that the temperature $T(t)$ at time t is then given by the *cooling equation*:

$$T'(t) = aP(t) - bT(t), \tag{1}$$

where $P(t)$ is the power at time t, and a and b are nonnegative constants. If T is a temperature function, then T' will always refer to the derivative of T with respect to time. We make observations about this cooling equation. Note that by rescaling temperature, or energy, one could assume that $a = 1$. Thus, a will not play much of a role in our analysis, and the key parameter is the *cooling parameter* b. Note that the temperature function will be continuous, even if the power function is not. If we want to maintain a constant temperature T_z, then as T' will equal 0, it is sufficient to run at power bT_z/a once temperature T_z is reached. Solving the cooling equation for $P(t)$ yields:

$$P(t) = \frac{T'(t) + bT(t)}{a}. \tag{2}$$

Thus, one can specify a power function, and hence a speed function, by specifying a temperature function. By Eq. (2), the energy used during an interval $[x, y]$ is:

$$\begin{aligned}\int_x^y P(t)dt &= \int_x^y \frac{T'(t) + bT(t)}{a}dt \\ &= \frac{1}{a}\int_x^y T'(t)dt + \frac{b}{a}\int_x^y T(t)dt \\ &= \frac{T(y) - T(x)}{a} + \frac{b}{a}\int_x^y T(t)dt. \end{aligned} \tag{3}$$

Since $P(s) = s^\alpha$, the work done during a time interval $[x, y]$ is

$$\int_x^y \left(\frac{T'(t) + bT(t)}{a}\right)^{1/\alpha} dt. \tag{4}$$

In the temperature version of our problem, the objective is to minimize the maximum temperature $T[S]$ reached during the schedule S. Let $c = \frac{\ln 2}{b}$. Call a time interval of length c a *c-interval*. As we will see, the problem of minimizing the maximum temperature is related to the problem of minimizing the maximum energy $C[S]$ used in any c-interval during the schedule S.

In the maximum speed version of our problem, the objective function is to minimize the maximum speed reached during the schedule. In the maximum power version of our problem, the objective function is to minimize the maximum power reached during the schedule.

If A is a scheduling algorithm, then $A(I)$ denotes the schedule output by A on input I. $E[A(I)]$ will denote the energy of $A(I)$, and $T[A(I)]$ the maximum temperature. For convenience, we will use $\mathrm{OPT}(I)$ to represent an optimal schedule for the objective under consideration. $E[\mathrm{OPT}(I)]$ will denote the optimal energy, and $T[\mathrm{OPT}(I)]$ the optimal temperature. When I is clearly understood from context, we may drop it from the notation.

A schedule is R-competitive, or R-approximate, for a particular objective function if the value of that objective function on the schedule is at most R times the value of the objective function on an optimal schedule. An online scheduling algorithm A is R-competitive, or has competitive ratio R, if $A(I)$ is R-competitive for all instances. An offline scheduling algorithm A is R-approximate, or has approximation ratio R, if $A(I)$ is R-approximate for all instances.

An online algorithm A is *cooling-oblivious* if A is $O(1)$-competitive with respect to temperature for all cooling parameters $b \geq 0$.

We now define the algorithms that we consider in this paper, along with related concepts. We start with the offline speed scaling algorithm YDS proposed by Yao et al. [1995]. Let $w(t_1, t_2)$ denote the work that has release time at least t_1 and has deadline at most t_2. The *intensity* $I(t_1, t_2)$ of the time interval $[t_1, t_2]$ is defined to be $w(t_1, t_2)/(t_2 - t_1)$.

Algorithm YDS: The algorithm repeats the following steps until all jobs are scheduled:

(1) Let $[t_1, t_2]$ be the maximum intensity time interval.
(2) The processor will run at speed $I(t_1, t_2)$ during $[t_1, t_2]$ and schedule all the jobs comprising $w(t_1, t_2)$, always running the released, unfinished task with the earliest deadline.
(3) Then the instance is modified as if the times $[t_1, t_2]$ didn't exist. That is, all deadlines $d_i > t_1$ are reduced to $\max(t_1, d_i - (t_2 - t_1))$, and all release times $r_i > t_1$ are reduced to $\max(t_1, r_i - (t_2 - t_1))$.

It is easy to see that the YDS algorithm is optimal with respect to maximum speed, and hence, maximum power, by noting that the maximum speed of any schedule must be at least the intensity of the maximum-intensity interval found by YDS at the very beginning (i.e., when no jobs have been scheduled), and the intensity of the maximum-intensity interval (and hence the speed at which YDS schedules jobs) only decreases as YDS proceeds. Note that the YDS schedule has the property that each task is run at a fixed speed. However, this speed may be different for different tasks. Let $y(t)$ be the speed of YDS at time t.

We now define the online speed scaling algorithms OA and AVR proposed by Yao et al. [1995].

Algorithm OA: Maintain the invariant that at all times t, the task with the earliest deadline is run at speed $\max_t w(t)/t$, where $w(t)$ is the unfinished work that has deadline within the next t units of time.

An alternative description is that OA's schedule for the future is always the optimal energy YDS schedule based on the current state.

Algorithm AVR: Maintain the invariant that at all times t, the earliest-deadline task is run at speed $\sum_{i \in J(t)} \frac{w_i}{d_i - r_i}$, where $J(t)$ is the collection of tasks i with $r_i \leq t \leq d_i$.

Intuitively, AVR runs each task at the optimal speed under the assumption that it is the only task in the system.

We now turn to defining our proposed online speed scaling algorithm [Bansal et al. 2004]. For $t_1 \leq t \leq t_2$, let $w(t, t_1, t_2)$ denote amount of work that has release time at least t_1 and deadline at most t_2 and that has already arrived by time t. We define three more terms $q(t)$, $p(t)$ and $v(t)$, which will be useful in the description and analysis of our new online algorithm BKP. Let $q(t)$ be the maximum intensity of an interval containing t, that is,

$$q(t) = \max_{t_1,t_2} I(t_1, t_2) \qquad \text{such that } t_1 < t \leq t_2.$$

Intuitively, $q(t)$ can be viewed as YDS's speed at time t. (Note that this statement is not exactly true, but $q(t)$ is an upper bound on YDS's speed at time t.) Let $p(t)$ be defined by:

$$p(t) = \max_{t_1,t_2} \frac{w(t, t_1, t_2)}{(t_2 - t_1)} \qquad \text{such that } t_1 < t \leq t_2.$$

Intuitively, $p(t)$ is the online algorithm's estimate of the speed at which YDS would work at time t, based on the knowledge of tasks that have arrived by time t. Let $v(t)$ be defined by:

$$v(t) = \max_{t'>t} \frac{w(t, et - (e-1)t', t')}{e(t' - t)}. \tag{5}$$

We are now ready to define the online algorithm BKP.

Algorithm BKP**:** At time t, work at speed $e\ v(t)$ on the unfinished task with the earliest deadline.

Note that $w(t, t_1, t_2)$, $p(t)$ and $v(t)$ may be computed by an online algorithm at time t. It is a matter of taste whether we define BKP to run at speed $e\ v(t)$ or $e\ p(t)$. All of our results hold for both variations. The following lemma, which we use frequently, relates $v(t)$, $p(t)$, and $q(t)$.

LEMMA 1.1. *For all instances I, and for all times t, $v(t) \leq p(t) \leq q(t)$.*

PROOF. The speed $v(t)$ is equivalent to a restricted variant of $p(t)$ where instead of considering the maximum over all t_1, t_2 such that $t_1 < t \leq t_2$, we require that t_1 and t_2 to be related such that $t - t_1 = (e-1)(t_2 - t)$. Thus, $v(t) \leq p(t)$. Finally, it is obvious that $p(t) \leq q(t)$ since $w(t, t_1, t_2) \leq w(t_1, t_2)$ for any $t_1 \leq t \leq t_2$. □

2. *Properties of the* YDS *Schedule*

2.1. ENERGY. We show that the energy optimality of the YDS schedule follows as a direct consequence of the well-known KKT optimality conditions for convex programs.

THEOREM 2.1. YDS *is optimal with respect to energy.*

PROOF. We start by stating the KKT conditions. Next we show how to express the energy problem as a convex program, and then show the result of applying the KKT conditions to this convex program.

Consider a convex program

$$\begin{aligned} &\min f_0(x) \\ &\quad f_i(x) \leq 0 \qquad i = 1, \ldots, n \end{aligned}$$

Assume that this program is strictly feasible, that is, there is some point x where $f_i(x) < 0$ for $i = 1, \ldots, n$. Assume that the f_i are all differentiable. Let λ_i, $i = 1, \ldots, n$ be a variable (Lagrangian multiplier) associated with the function $f_i(x)$. Then, the necessary and sufficient KKT conditions for solutions x and λ to be feasible primal and dual solutions are Boyd and Vandenberghe [2004]:

$$f_i(x) \leq 0 \qquad i = 1, \ldots, n \tag{6}$$

$$\lambda_i \geq 0 \qquad i = 1, \ldots, n \tag{7}$$

$$\lambda_i f_i(x) = 0 \qquad i = 1, \ldots, n \tag{8}$$

$$\nabla f_0(x) + \sum_{i=1}^{n} \lambda_i \nabla f_i(x) = 0. \tag{9}$$

To state the energy minimization problem as a convex program, we break time into intervals $t_0, \ldots t_m$ at release times and deadlines of the tasks. Note that because the set of available jobs does not change over any such interval and because of the convexity of the speed-to-power function, we may assume that the processor runs at constant speed throughout any such interval. Let $J(i)$ be the tasks that can feasibly be executed during the time interval $I_i = [t_i, t_{i+1}]$, and $J^{-1}(j)$ be intervals during which task j can be feasibly executed. We introduce a variable $w_{i,j}$, for $j \in J(i)$, that represents the work done on task j during time $[t_i, t_{i+1}]$. Our (interval indexed) program is then:

$$\min \quad E \tag{10}$$

$$w_j \leq \sum_{i \in J^{-1}(j)} w_{i,j} \qquad j = 1, \ldots, n \tag{11}$$

$$\sum_{i=1}^{m} \left(\frac{\sum_{j \in J(i)} w_{i,j}}{t_{i+1} - t_i} \right)^{\alpha} (t_{i+1} - t_i) \leq E \tag{12}$$

$$w_{i,j} \geq 0 \qquad i = 1, \ldots, m \qquad j \in J(i) \tag{13}$$

It is easy to verify that the program is convex. We now apply the KKT conditions to this program. We associate a dual variable δ_j with inequality j in line 11, a dual variable β with the inequality in line 12, and a dual variable $\gamma_{i,j}$ with inequality i, j in line 13. We now evaluate line 9 of the KKT conditions for our convex program. We have

$$\begin{aligned} \nabla E \; + \; & \sum_{j=1}^{n} \delta_j \nabla \left(w_j - \sum_{i \in J^{-1}(j)} w_{i,j} \right) \\ & + \; \beta \; \nabla \left(\left(\sum_{i=1}^{m} \frac{\sum_{j \in J(i)} w_{i,j}}{t_{i+1} - t_i} \right)^{\alpha} (t_{i+1} - t_i) - E \right) \\ & - \; \sum_{i=1}^{m} \sum_{j=1}^{n} \gamma_{i,j} \nabla w_{i,j} = 0. \end{aligned}$$

Considering the component of this equation corresponding to the variable E, we have $\beta - 1 = 0$, or equivalently $\beta = 1$. Considering the component corresponding to the variable $w_{i,j}$, we have

$$-\delta_j + \beta\alpha \left(\frac{\sum_{k \in J(i)} w_{i,k}}{t_{i+1} - t_i} \right)^{\alpha - 1} - \gamma_{i,j} = 0. \tag{14}$$

Consider a $w_{i,j}$ such that $w_{i,j} > 0$. We know that by complementary slackness (Eq. (8)) that it must be the case that $\gamma_{i,j} = 0$. Hence,

$$\delta_j = \alpha \left(\frac{\sum_{k \in J(i)} w_{i,k}}{t_{i+1} - t_i} \right)^{\alpha - 1}. \tag{15}$$

Hence, the interpretation of the dual variable δ_j is α times the speed at which the processor runs during interval i raised to the power of $(\alpha - 1)$. This quantity, and hence the speed of the processor, must be the same for each i such that $w_{i,j} > 0$, that is, during each interval i in which task j is run.

Now consider a $w_{i,j}$ such that $w_{i,j} = 0$. Rearranging Eq. (14), we find that

$$\gamma_{i,j} = \alpha \left(\frac{\sum_{k \in J(i)} w_{i,k}}{t_{i+1} - t_i} \right)^{\alpha - 1} - \delta_j. \tag{16}$$

Since $\gamma_{i,j}$ is nonnegative, the processor runs at least as fast during interval i as during the intervals where task j is run.

Thus, we can conclude that necessary conditions for a primal feasible solution to be optimal are:

—For each task j, the processor runs at the same speed, say s_j, in all intervals i in which task j is run.

—The processor runs at speed no less than s_j during intervals i such that $j \in J(i)$ and task j is not run.

The YDS schedule clearly has these properties. We can use Eqs. (15) and (16) to find a dual feasible solution satisfying the KKT conditions, and thus we have found optimal primal and dual solutions. The YDS schedule corresponding to the primal solution is thus optimal. □

2.2. The Relationship Between Energy and Temperature. We show in Theorem 2.2 that the maximum temperature $T[S]$ of a schedule S is within a factor of two of $a \cdot C[S]$. Recall the parameter a in the cooling equation 1 and our definition of a c-interval where $c = \frac{\ln 2}{b}$, and that $C[S]$ is the maximum energy expended over any c-interval. Further, recall that we assume that the initial temperature is zero. We show in Theorem 2.7 that the energy optimal YDS schedule is 20-approximate with respect to temperature for all cooling parameters $b \geq 0$, that is, YDS is cooling-oblivious. Given Theorem 2.2, to prove Theorem 2.7, it is sufficient to show that YDS is 5-approximate with respect to the objective of the maximum energy expended over any c-interval.

Theorem 2.2. *For any schedule S, and for any cooling parameter $b \geq 0$,*

$$\frac{a\, C[S]}{2} \leq T[S] \leq 2a\, C[S].$$

PROOF. If $b = 0$, then $a \cdot C[S] = T[S] = a \cdot E[S]$, and the result holds. So assume from now on that $b > 0$. We rewrite our cooling equation $\frac{dT(t)}{dt} = aP(t) - bT(t)$ as

$$\frac{d(\exp(bt)T(t))}{dt} = a\exp(bt)P(t).$$

Integrating this equation over a c-interval that ends at some time t_0 we get

$$T(t_0)\exp(bt_0) - T(t_0 - c)\exp(bt_0 - cb) = a\int_{t_0-c}^{t_0} \exp(bt)P(t)dt, \tag{17}$$

(note $T(x) = 0$ for $x \leq 0$).

We first show that $T[S] \leq 2a\ C[S]$. Suppose that the temperature $T[S]$ is achieved at time t_0. We simplify Eq. (17) as follows. Since $\exp(bt)$ is increasing in t, we have that $\int_{t_0-c}^{t_0} \exp(bt)P(t)dt \leq \exp(bt_0)\int_{t_0-c}^{t_0} P(t)dt$. Thus,

$$T(t_0) \leq T(t_0 - c)\exp(-cb) + a\int_{t_0-c}^{t_0} P(t)dt.$$

As $T(t_0 - c) \leq T(t_0) = T[S]$, it follows that

$$T[S](1 - \exp(-cb)) \leq a\int_{t_0-c}^{t_0} P(t)dt \leq a\ C[S],$$

and as $cb = \ln 2$, it follows that

$$T[S] \leq \frac{a\ C[S]}{1 - \exp(-cb)} = 2a\ C[S].$$

We now show that $T[S] \geq a\ C[S]/2$. Let $[t_0 - c, t_0]$ be a c-interval where $C[S]$ energy is used. Again we start with Eq. (17). Using the fact that the temperature at any time is nonnegative (as the environmental temperature is 0) and hence in particular that $T(t_0 - c) \geq 0$, and that $\exp(bt)$ is an increasing function of t, it follows that

$$\begin{aligned} T(t_0)\exp(bt_0) &\geq a\int_{t_0-c}^{t_0} \exp(bt)P(t)dt \geq a\exp(bt_0 - cb)\int_{t_0-c}^{t_0} P(t)dt \\ &= \exp(bt_0 - cb)a\ C[S]. \end{aligned}$$

Thus,

$$T[S] \geq T(t_0) \geq \exp(-cb)a\ C[S] = \frac{a\ C[S]}{2}.$$ □

Note that YDS is not optimal for minimizing the maximum temperature, nor for minimizing the maximum total energy in any c-interval. The fact that YDS is not optimal for temperature can be seen on single-task instances where the optimal-temperature schedule must run at a speed that follows some non-constant Euler–Lagrange temperature curve (see Theorem 3.6 for more details). That YDS is not optimal for minimizing the maximum energy used in any c-interval can be seen from the following instance. Without loss of generality we can normalize so that $c = 2$. There are two tasks with work 1 each, both arriving at time 0, with deadlines

1 and 3, respectively. If the first task is run at speed 1 from time 0 to 1 and the second at speed 1 from 2 to 3, the maximum energy in any interval of length 2 is 1. YDS runs the first task at speed 1 from time 0 to 1 and the second at speed 1/2 from 1 to 3. The energy used from time 0 to 2, for instance, is larger than 1. Note that this holds for any speed to power function of the form $P(s) = s^\alpha$, with $\alpha \geq 0$.

2.3. ENERGY IN A c-INTERVAL. For the rest of this section we only consider the objective of minimizing the maximum energy in any c-interval. This will culminate in Lemma 2.6, which states that the YDS schedule is 5-approximate with respect to this objective.

We first require some preliminary definitions and observations. Let $CY(I)$ denote a c-interval in YDS(I) that uses energy $C[\text{YDS}(I)]$, a maximum-energy c-interval YDS(I). Let ϵ be a small positive constant that we will define precisely later. Let the speed s_0 be defined as

$$s_0 = (\epsilon C[\text{YDS}(I)]/c)^{1/\alpha}$$

We call a task in I *slow* if it runs at a speed strictly less than s_0 in YDS(I). This notion is well defined because each task runs at constant speed in the YDS schedule. The rest of the tasks are called *fast*. Let $s(t)$ denote the speed at time t in YDS(I). Define an *island* to be a maximal interval of time where $s(t) \geq s_0$.

We now give some simple but useful properties of the schedule YDS(I).

CLAIM 2.3. *Let $G = [t_1, t_2]$ be an island.* YDS *schedules within G exactly those tasks k such that $t_1 \leq r_k \leq d_k \leq t_2$.*

PROOF. It is trivial that all such tasks must be executed wholly in G. To see that only such tasks are executed in G, observe that if a task can feasibly be executed outside G and is (partially or wholly) executed in G, we would have a contradiction to the energy optimality of YDS(I) since work could be shifted from an interval of higher speed to an interval of lower speed. □

CLAIM 2.4. *For any island G of length no more than c in any instance I, $C[\text{OPT}(I)]$ is at least the energy consumed in G by* YDS.

PROOF. Follows from Claim 2.3 and the energy optimality of YDS, since OPT must execute in G at least the tasks executed by YDS in G. □

We now show that most of the energy in $CY(I)$ is contained in fast tasks.

LEMMA 2.5. *Let $H(I)$ denote the set of islands that intersect $CY(I)$, and let $E[H(I)]$ denote the energy consumed under* YDS *in the islands $H(I)$. Then, we have that $E[H(I)] \geq (1 - \epsilon)C[\text{YDS}(I)]$.*

PROOF. Consider the times in $CY(I)$ where $s(t) \geq s_0$. Those periods are contained in $H(I)$, so $C[\text{YDS}(I)] - E[H(I)]$ is at most the energy used by the YDS(I) during the times t in the c-interval $CY(I)$ when $s(t) < s_0$. Thus, $C[\text{YDS}(I)] - E[H(I)] \leq cs_0^\alpha$, which is at most $\epsilon C[\text{YDS}(I)]$ by the definition of s_0. The claimed inequality then follows. □

We now show that YDS is 5-approximate with respect to minimizing the maximum energy used over any c-interval.

LEMMA 2.6. *For any instance I,*

$$C[\mathrm{OPT}(I)] \geq \min\left(\frac{\epsilon\, C[\mathrm{YDS}(I)]}{2}, \frac{(1-\epsilon)C[\mathrm{YDS}(I)]}{3}\right)$$

Choosing $\epsilon = 2/5$, it follows that $C[\mathrm{OPT}(I)] \geq C[\mathrm{YDS}(I)]/5$.

PROOF. Consider an island G of I and let $|G|$ be the length of G. As the YDS schedule for I runs at speed at least s_0 during G, the total energy consumed by YDS is at least $|G|s_0^\alpha$. By Claim 2.3, all the tasks in I that YDS runs in G must also be run in G in any feasible schedule, so it must be the case that the total energy consumed by any feasible schedule for I during G has to also be at least $|G|s_0^\alpha$. If $|G| \geq c$, then by a simple averaging argument, for any feasible schedule there is some c-interval that is totally contained in G with the property that the energy used during this c-interval is at least $|G|s_0^\alpha/\lceil |G|/c \rceil$. In turn, this is at least $cs_0^\alpha/2$, which by the definition of α equals $\epsilon\, C[\mathrm{YDS}(I)]/2$. Thus, $C[\mathrm{OPT}(I)] \geq \epsilon\, C[\mathrm{YDS}(I)]/2$ if $|G| \geq c$.

If all the islands have length no more than c, then consider the islands that intersect $CY(I)$. If some such island G has energy at least $(1-\epsilon)C[\mathrm{YDS}(I)]/3$, the result follows by Claim 2.4. Now suppose that all islands that intersect $CY(I)$ have energy less than $(1-\epsilon)C[\mathrm{YDS}(I)]/3$. By Lemma 2.5, we know that in $\mathrm{YDS}(I)$ the total energy during the islands intersecting $CY(I)$ is at least $(1-\epsilon)C[\mathrm{YDS}(I)]$. As at most two islands can lie partially in $CY(I)$, at least $(1-\epsilon)C[\mathrm{YDS}(I)]/3$ energy is in islands that are totally contained inside $CY(I)$, and hence the result follows by Claim 2.4. □

We can now conclude that YDS is cooling-oblivious.

THEOREM 2.7. *The energy optimal algorithm* YDS *is a* 20*-approximation with respect to maximum temperature.*

PROOF. By Lemma 2.6, we know that $C[\mathrm{OPT}(I)] \geq C[\mathrm{YDS}(I)]/5$. By Theorem 2.2, we know that $\frac{2\,T[\mathrm{OPT}(I)]}{a} \geq C[\mathrm{OPT}(I)]$, and $C[\mathrm{YDS}(I)] \geq \frac{T[\mathrm{YDS}(I)]}{2a}$. Combining these three inequalities gives that $20\,T[\mathrm{OPT}(I)] \geq T[\mathrm{YDS}(I)]$. □

3. *The OA, AVR, and Constant Temperature Algorithms*

In this section we consider the online algorithms AVR and OA proposed by Yao et al. [1995], and the class of constant temperature algorithms. Recall the definitions of AVR and OA in Section 1.5. We show that both AVR and OA have a competitive ratio of at least α^α with respect to energy. We then show that OA is in fact exactly α^α-competitive. We show that both AVR and OA are not $O(1)$-competitive with respect to temperature. We also show that another natural algorithm, that we call the constant temperature algorithm, is not $O(1)$-competitive with respect to temperature under a natural definition of competitiveness for this class of algorithms.

3.1. ENERGY.

Before giving an explicit lower bound instance for the competitive ratio of OA and AVR with respect to energy, we need the following technical lemma.

LEMMA 3.1. *If $\alpha > 1$ and $x \geq y > 0$, then*

$$(x-y)^\alpha \geq x^\alpha - \alpha x^{\alpha-1} y.$$

PROOF. Setting $t = y/x$ and dividing through by x^α, the inequality above is equivalent to $(1-t)^\alpha \geq 1 - \alpha t$ or $(1-t)^\alpha - 1 + \alpha t \geq 0$, which we need to show holds when $0 < t \leq 1$. Note that the left-hand side is 0 when $t = 0$. Now differentiating the left-hand side with respect to t gives $-\alpha(1-t)^{\alpha-1} + \alpha$, which is always positive when $0 < t \leq 1$. Thus, $(1-t)^\alpha - 1 + \alpha t$ is increasing and thus positive when $0 < t < 1$. □

LEMMA 3.2. *The competitive ratio of* AVR *and* OA *with respect to energy is at least* α^α.

PROOF. The instance is defined as follows: All tasks have the same deadline n. For $i = 0, 1, \ldots, n-1$, a task of work $(1/(n-i))^{1/\alpha}$ arrives at time i. Observe that for instances with a common deadline, as is the case here, AVR and OA behave identically.

The optimal energy algorithm YDS completes the task that arrives at time i by time $i+1$ running at speed $(1/(n-i))^{1/\alpha}$ during the time interval $(i, i+1)$. The resulting energy usage for YDS is then $\sum_{i=0}^{n-1}(1/(n-i)^{1/\alpha})^\alpha = \sum_{i=0}^{n-1} 1/(n-i) = H_n$, where H_n is the nth Harmonic number.

We now analyze the energy usage of OA and AVR. Let $s(i)$ be the speed of AVR during the time interval $(i, i+1)$. Then, for $i = 0, \ldots, n-1$,

$$\begin{aligned} s(i) &= \sum_{j=0}^{i} \frac{1/(n-j)^{1/\alpha}}{n-j} \\ &= \sum_{j=0}^{i} \frac{1}{(n-j)^{(1+1/\alpha)}} \\ &\geq \int_{j=0}^{j=i-1} 1/(n-j)^{1+1/\alpha} dj \\ &= \alpha(n-i+1)^{-1/\alpha} - \alpha n^{-1/\alpha}. \end{aligned} \tag{18}$$

The first equality above is by the definition of AVR. Then, the energy used by AVR is

$$\begin{aligned} E[\mathrm{AVR}(I)] &= \sum_{i=0}^{n-1} s(i)^\alpha \\ &\geq \sum_{i=1}^{n-1} \left(\alpha(n-i+1)^{-1/\alpha} - \alpha n^{-1/\alpha}\right)^\alpha \\ &= \alpha^\alpha \sum_{i=1}^{n-1} \left((n-i+1)^{-1/\alpha} - n^{-1/\alpha}\right)^\alpha \\ &\geq \alpha^\alpha \sum_{i=1}^{n-1} \left(\frac{1}{(n-i+1)} - \alpha \left(\frac{1}{(n-i+1)}\right)^{(\alpha-1)/\alpha} n^{-1/\alpha}\right) \\ &= \alpha^\alpha \left(\sum_{i=1}^{n-1} \frac{1}{(n-i+1)} - \alpha n^{-1/\alpha} \sum_{i=1}^{n-1} \left(\frac{1}{(n-i+1)}\right)^{(\alpha-1)/\alpha}\right) \end{aligned}$$

$$= \alpha^\alpha \left((H_n - 1) - \alpha^2 n^{-1/\alpha}\Theta(n^{1/\alpha})\right)$$
$$= \alpha^\alpha (H_n - \Theta(1)).$$

The first equality above is by the definition of YDS. The first inequality comes from the lower bound on $s(i)$ from Eq. (18). The second inequality comes from applying Lemma 3.1 with $x = (n - i + 1)^{-1/\alpha}$ and $y = n^{-1/\alpha}$. Choosing n large enough, the competitive ratio can be made arbitrarily close to α^α. □

We now turn to the main result of this section, that the competitive ratio of OA with respect to energy is exactly α^α. Before we begin, we need the following algebraic fact.

LEMMA 3.3. *Let $q, r, \delta \geq 0$ and $\alpha \geq 1$. Then, $(q+\delta)^{\alpha-1}(q - \alpha r - (\alpha-1)\delta) - q^{\alpha-1}(q - \alpha r) \leq 0$.*

PROOF. We need to show that

$$(q+\delta)^{\alpha-1}(q - \alpha r) - (q+\delta)^{\alpha-1}(\alpha-1)\delta - q^{\alpha-1}(q - \alpha r) \leq 0$$

or equivalently that,

$$(q - \alpha r)[(q+\delta)^{\alpha-1} - q^{\alpha-1}] - (q+\delta)^{\alpha-1}(\alpha-1)\delta \leq 0.$$

Since $[(q+\delta)^{\alpha-1} - q^{\alpha-1}] \geq 0$, it suffices to show that

$$q[(q+\delta)^{\alpha-1} - q^{\alpha-1}] - (q+\delta)^{\alpha-1}(\alpha-1)\delta \leq 0.$$

Substituting $\delta = zq$, the left-hand side of the above can be written

$$q^\alpha[(1+z)^{\alpha-1} - 1] - q^\alpha[(1+z)^{\alpha-1}(\alpha-1)z].$$

Thus, it will be enough to show that for $z \geq 0$,

$$(1+z)^{\alpha-1} - 1 - (1+z)^{\alpha-1}(\alpha-1)z \leq 0.$$

Differentiating this with respect to z, we get

$$((\alpha-1)(1+z)^{\alpha-2}[1 - (\alpha-1)z] + (1+z)^{\alpha-1}(-\alpha+1))$$
$$= ((\alpha-1)(1+z)^{\alpha-2}[1 - (\alpha-1)z - (1+z)]$$
$$= -\alpha(\alpha-1)z(1+z)^{\alpha-2}$$
$$\leq 0,$$

where the last inequality holds since $\alpha > 1$ and $z \geq 0$. Thus, the maximum of this expression is attained at $z = 0$, where it has value 0. This implies the result. □

THEOREM 3.4. *The algorithm Optimum Available is α^α-competitive with respect to energy.*

PROOF. Let $s_{OA}(t)$ denote the speed at which OA works at time t and $s_{OPT}(t)$ denote the speed that the optimal algorithm YDS works at time t. At any time t, either a task arrives or finishes, or else an infinitesimal interval of time dt elapses and OA consumes $s_{OA}(t)^\alpha dt$ units of energy. We will define a potential function $\phi(t)$ that satisfies the following properties:

—The potential function $\phi(t)$ does not increase as a result of any of the following events: the arrival of a task, the completion of a task by OA, the completion of a task by OPT.

—At any time t between arrivals,

$$s_{OA}(t)^{\alpha} + \frac{d\phi(t)}{dt} \leq \alpha^{\alpha} s_{OPT}(t)^{\alpha}. \tag{19}$$

—The potential function $\phi(t)$ has value 0 before any tasks arrive, and also has value 0 after the last deadline.

Integrating Eq. (19) over time and using the other two stated properties, we can conclude that $E[\mathrm{OA}(I)] \leq \alpha^{\alpha} E[\mathrm{OPT}(I)]$. For more information on the potential function method, see Cormen et al. [2001].

Before we can define the potential function, we need to introduce some notation. Let $s(t)$ denote the speed at which OA would be working at time t if no new tasks were to arrive after the current time. Since OA simply computes the YDS schedule based on the current knowledge of tasks, the speeds $s(t)$ are computed as follows. Let $w_{OA}(t, t')$ denote the unfinished work under OA that is currently available with deadlines in $(t, t']$. We will refer to $w_{OA}(t, t')/(t' - t)$ as the *density* of interval $(t, t']$. Consider a sequence of times defined inductively as follows. Let t_0 always denote the current time. Let t_i, $i > 0$, denote the smallest time such that

$$w_{OA}(t_{i-1}, t_i)/(t_i - t_{i-1}) = \max_{t' > t_{i-1}} w_{OA}(t_{i-1}, t')/(t' - t_{i-1}). \tag{20}$$

Thus, t_1 is the smallest time when the density of tasks with deadlines between t_0 and t_1 is maximized, and so on. It is easy to see that for all $i \geq 0$, $s(t) = s(t_i)$ for $t_i < t \leq t_{i+1}$. We will call the interval $(t_i, t_{i+1}]$, a *critical* interval and denote it by I_i. Note that these intervals are those scheduled in successive steps of the YDS algorithm, assuming that ties are broken by choosing the smallest interval with maximum intensity. We now note that $s(t_i)$ is a nonincreasing sequence. If $s(t_{i+1}) > s(t_i)$, then this contradicts that $s(t_i)$ is the critical density for time t_i, since in this case

$$w_{OA}(t_i, t_{i+2})/(t_{i+2} - t_i) > w_{OA}(t_i, t_{i+1})/(t_{i+1} - t_i) = s(t_i).$$

Analogously, let $w_{OPT}(t, t')$ denote the unfinished work under the optimal offline algorithm at the current time that has deadline in $(t, t']$.

We define the potential function $\phi(t)$ as follows:

$$\phi(t) = \alpha \sum_{i \geq 0} s(t_i)^{\alpha - 1} (w_{OA}(t_i, t_{i+1}) - \alpha w_{OPT}(t_i, t_{i+1})). \tag{21}$$

Note that (by the definition of t_0) OA is always working at speed $s(t_0)$. If no new task arrives, the algorithm continues to work at the same speed until the current critical interval finishes. When the current critical interval finishes, the algorithm enters the next critical interval. The indices of the critical deadlines shift by one, and the new speed $s(t_0)$ is that which was previously $s(t_1)$. Also note that the potential is continuous as a critical interval finishes and we move to the next one. This follows because as the current critical interval finishes $t_1 - t_0$ approaches 0 and both $w_{OA}(t_0, t_1)$ and $w_{OPT}(t_0, t_1)$ approach 0, since both algorithms have to finish this work by time t_1. Thus, the contribution of the first term approaches 0 as the current interval is about to finish.

We now consider the various cases as time progresses, and prove that the potential function satisfies the properties claimed above. It is easy to see that at any time, OA is running some task if and only if the optimal energy algorithm YDS is running

some task, and thus we may assume that all times in our arguments that each of OA and YDS is running some task.

Working Case. We first consider the case that no task arrives and no task is completed during the next dt units of time. Thus, each $s(t_i)$ remains fixed (including $s(t_0)$) during the dt time units. We have to show that

$$s(t)^\alpha - \alpha^\alpha s_{OPT}(t)^\alpha + \frac{d\phi(t)}{dt} \leq 0 \tag{22}$$

or equivalently,

$$s(t_0)^\alpha - \alpha^\alpha s_{OPT}(t_0)^\alpha + \frac{d}{dt}\left(\alpha \sum_{i\geq 0} s(t_i)^{\alpha-1}(w_{OA}(t_i, t_{i+1}) - \alpha w_{OPT}(t_i, t_{i+1}))\right) \leq 0. \tag{23}$$

As OA works, $w_{OA}(t_0, t_1)$ is decreasing at rate $s(t_0)$, and $w_{OA}(t_i, t_{i+1})$ remains fixed for all $i \geq 1$. Let k be the smallest index ≥ 0 such that $w_{OPT}(t_k, t_{k+1}) \neq 0$. Assuming the optimal energy schedule YDS always executes the task with the earliest deadline, we have that $w_{OPT}(t_k, t_{k+1})$ decreases at rate $s_{OPT}(t_0)$, and $w_{OPT}(t_i, t_{i+1})$ is fixed for $i \neq k$.

Thus, evaluating the left-hand side of Eq. (23), we see that it is equivalent to

$$s(t_0)^\alpha - \alpha^\alpha s_{OPT}(t_0)^\alpha + (-\alpha s(t_0)^{\alpha-1} s(t_0) + \alpha^2 s(t_k)^{\alpha-1} s_{OPT}(t_0)) \leq 0. \tag{24}$$

Since $s(t_k) \leq s(t_0)$, Eq. (24) would be implied by

$$(1-\alpha)s(t_0)^\alpha + \alpha^2 s(t_0)^{\alpha-1} s_{OPT}(t_0) - \alpha^\alpha s_{OPT}(t_0)^\alpha \leq 0. \tag{25}$$

Let $z = s(t_0)/s_{OPT}(t_0)$. By substitution, Eq. (25) is equivalent to

$$(1-\alpha)z^\alpha + \alpha^2 z^{\alpha-1} - \alpha^\alpha \leq 0. \tag{26}$$

for $z \geq 0$. Let $u(z)$ be the polynomial on the left-hand side of inequality (26). Note that $u(0) = -\alpha^\alpha$, and $u(+\infty) = -\infty$. Differentiating $u(z)$ with respect to z, we get

$$u'(z) = \alpha(1-\alpha)z^{\alpha-1} + \alpha^2(\alpha-1)z^{\alpha-2}. \tag{27}$$

Solving for $u'(z) = 0$, we get the unique value $z = \alpha$. Because $\alpha \geq 1$, $u'(z) \geq 0$ for $z \leq \alpha$ and $u'(z) \leq 0$ for $z \geq \alpha$, $u(z)$ is maximized at $z = \alpha$, and $u(\alpha) = 0$. Hence, $u(z)$ is nonpositive for nonnegative z. Thus, we have established inequality (22).

Arrival Case. Consider the arrival of a task of work x with deadline t. Let i be such that $t_i < t \leq t_{i+1}$. We must show that the change in potential caused by this arrival is nonpositive.

First, we consider the simplest case when the critical intervals are unchanged, that is, only the values of the critical densities change. Hence, the only effect the new task has is to increase the density $s(t_i)$ of the interval $I_i = (t_i, t_i + 1]$ to

$$\frac{(w_{OA}(t_i, t_{i+1}) + x)}{(t_{i+1} - t_i)},$$

and the quantity $(w_{OA}(t_i, t_{i+1}) - \alpha w_{OPT}(t_i, t_{i+1}))$ decreases by $(\alpha - 1)x$. Thus, the change in the potential function is then

$$\Delta\phi = \alpha \left(\frac{w_{OA}(t_i, t_{i+1}) + x}{t_{i+1} - t_i} \right)^{\alpha-1} ((w_{OA}(t_i, t_{i+1}) + x) - \alpha(w_{OPT}(t_i, t_{i+1}) + x))$$
$$-\alpha \left(\frac{w_{OA}(t_i, t_{i+1})}{t_{i+1} - t_i} \right)^{\alpha-1} (w_{OA}(t_i, t_{i+1}) - \alpha w_{OPT}(t_i, t_{i+1})). \quad (28)$$

Substituting $q = w_{OA}(t_i, t_{i+1})$, $\delta = x$ and $r = w_{OPT}(t_i, t_{i+1})$ and rearranging, we can write

$$\Delta\phi = \frac{\alpha \cdot ((q + \delta)^{\alpha-1}(q - \alpha r - (\alpha - 1)\delta) - q^{\alpha-1}(q - \alpha r))}{(t_{i+1} - t_i)^{\alpha-1}}$$

which is nonpositive by Lemma 3.3.

We now consider the more interesting case when the arrival of a task might change the critical intervals. While this new task may radically change the structure of the critical intervals, we show that we can think of this change as a sequence of smaller changes, where each smaller change affects only two critical intervals. Moreover, each change is essentially equivalent to that in the previous case where the structure of the critical intervals remains unchanged. To explain how to accomplish this, imagine the work of the new task increasing starting from 0. For some amount of work $x' \leq x$, one of the following three events must occur:

—The interval I_i remains a critical interval and its density becomes equal to that of I_{i-1}. In particular, x' is such that $s(t_{i-1}) = (w_{OA}(t_i, t_{i+1}) + x')/(t_{i+1} - t_i)$.

—The interval I_i splits into two critical intervals $I_i' = (t_i, t']$ and $I_i'' = (t', t_{i+1}]$ for some $t_i \leq t' < t_{i+1}$. Since x' is the smallest such work, the densities of I_i' and I_i'' are identical and equal to $(w_{OA}(t_i, t_{i+1}) + x')/(t_{i+1} - t_i)$.

—A sequence $I_0, \ldots, I_i$ of critical intervals merge into one new critical interval. We can think of this event as a sequence of pairwise merges, each of which combines I_0 with I_1 to form a new interval I_0'. In this case, it must be that $x' = 0$.

For each of these events, we can imagine the original task of work x as consisting of two tasks, such that both arrive at the same time and have the same deadline t, but one has work x' and the other has work $x - x'$. We can then first analyze the change in potential as the result of the arrival of the work x', and then repeat this procedure recursively for $x - x'$. Thus, we only need to consider the change in potential for a task of work x' that causes one of the three events described above.

For the first type of event, the only change in the potential function is due to the change of density of I_i. This change is identical to Eq. (28) with x replaced by x', and again the non-positivity of $\Delta\phi$ follows by Lemma 3.3.

For the second type of event, the change in the potential function is only due to I_i being replaced by I_i' and I_i''. Since the densities of I_i' and I_i'' are identical, these intervals can still be considered together as far as the potential function is concerned. Hence, the change in the potential function is again given by Eq. (28) with x replaced by x', and again the non-positivity of $\Delta\phi$ follows by Lemma 3.3.

For the third type of event, the change in the potential function is only due to I_0 and I_1 being replaced by I_0'. Since the densities of I_0, I_1, and I_0' are all identical, these intervals can still be considered together as far as the potential function is

concerned. Hence, the change in the potential function is again given by Eq. (28) with x replaced by $x' = 0$, and again the nonpositivity of $\Delta\phi$ follows by Lemma 3.3.

We can repeat this process until we have used up all of the x work in the arriving task. □

3.2. TEMPERATURE. We start this subsection by showing that AVR and OA are not $O(1)$-competitive with respect to temperature. In the reasonable case that the thermal threshold $T_{\max}$ of the device is known, the most obvious temperature management strategy is to run at a speed that leaves the temperature fixed at $T_{\max}$. In particular, whenever there is work to do, the algorithm works at a speed that maintains the temperature $T_{\max}$, and otherwise it cools according to Newton's law of cooling. We call such a strategy $O(1)$-competitive if on any instance I on which this constant temperature algorithm misses a deadline, every feasible schedule reaches a temperature of $\Omega(T_{\max})$ at some point in time. We then show that staying at the thermal threshold is not an $O(1)$-competitive strategy.

LEMMA 3.5. *The online algorithms* AVR *and* OA *are not* $O(1)$*-competitive with respect to temperature. More precisely, the competitive ratios of these algorithms must depend on either the number of tasks or the cooling rate* b.

PROOF. We use a variation of an instance from Yao et al. [1995]. Choose an arbitrarily large integer n and consider an instance with n tasks, where task i is released at time $r_i = ic$, has work $w_i = c$ and deadline $d_i = nc$ for $0 \leq i \leq n-1$. Again note that since all tasks have a common deadline, AVR and OA behave identically. The YDS schedule runs tasks at a constant speed of 1 and thus uses total energy n and energy c in any c-interval. Using Theorem 2.2, it is sufficient to show that there is some c-interval where the energy used by OA and AVR is $\omega(c)$. As AVR runs task i at speed $1/(n-i)$ during the interval $(ic, nc]$, during the c-interval $[c(n-1), cn]$ AVR and OA run at a speed of $H_n = \Theta(\log n)$, where H_n is the nth harmonic number, and thus the energy used during this c-interval is $\Omega(c \log^{\alpha} n)$. □

THEOREM 3.6. *The speed scaling algorithm that runs at such a speed that the temperature remains constant at the thermal threshold* $T_{\max}$ *is not* $O(1)$*-competitive.*

PROOF. Suppose at time 0 a task with work x (which will be specified later) and deadline ϵ arrives. We will consider the behavior as ϵ goes to 0. Suppose the temperature at time 0 is 0. We choose x such that it is equal to the maximum work that the adversary can get done by time ϵ while keeping the temperature below $T_{\max}/k$ for some constant k. Using Eq. (50) from Section 5 for this maximum work, and substituting $\alpha = 3$, we get $x = \Theta((\frac{bT_{\max}}{ka})^{1/3}\epsilon^{2/3})$. The crucial fact is that the maximum work that the adversary can do depends on ϵ as $\epsilon^{2/3}$. On the other hand, the constant temperature algorithm at temperature $T_{\max}$ has power $P = bT_{\max}/a$, and hence speed $(bT_{\max}/a)^{1/3}$ and work $\Theta((bT_{\max}/a)^{1/3}\epsilon)$, which depends linearly on ϵ. Thus, for any constant k, the ratio of the work completed by the adversary to the work completed by the constant temperature algorithm goes to infinity as ϵ goes to 0. □

4. *The BKP Algorithm*

Recall the definition of our newly introduced algorithm BKP in Section 1.5. We will show that it is cooling-oblivious. That is, BKP is $O(1)$-competitive with respect to energy, maximum power/speed, and temperature. We analyze BKP separately for each of these objectives.

4.1. PRELIMINARIES. Recall that at any time t, BKP works at speed $e \cdot v(t)$ on the unfinished job with the earliest deadline, where $v(t)$ is defined by Eq. (5). In this section, we first prove that BKP always produces a feasible schedule. We then give four inequalities that will allow us to relate BKP to the optimal schedule.

THEOREM 4.1. *The BKP algorithm always outputs a feasible schedule.*

PROOF. Assume for the sake of contradiction that BKP misses some deadline for some problem instance. Of these infeasible instances, consider one with the fewest number of tasks, and let d denote the first deadline that is missed in this instance. We claim that on this instance BKP always works on tasks with deadline no more than d during the interval $(0, d]$. To see this, first observe that if BKP is idle at some time t during $(0, d]$, then we can replace the instance by a smaller one by removing all jobs that finish before t. Similarly, as BKP always works on the job with the earliest deadline, if it worked on a task with deadline greater than d at some time $t \in (0, d]$, then BKP must have finished all work with deadline no more than than d that arrived by time t. Thus, one could obtain another infeasible instance with fewer tasks by considering only those tasks released after the time t.

This implies that if BKP misses the deadline at time d, it must be that the total work done by BKP during $[0, d]$ is strictly less than $w(0, d)$, the total work in the instance with deadline no more than d. Our proof will be to show that this cannot happen.

By choosing $t' = d$ in the definition of $v(t)$, it follows trivially that

$$v(t) \geq \frac{w(t, et - (e-1)d, d)}{e(d-t)}.$$

Thus, the work done by BKP during the time period $[0, d]$ is

$$\int_0^d e\, v(t)dt \geq \int_0^d \frac{w(t, et - (e-1)d, d)}{(d-t)}dt.$$

We now expand the right hand side of this inequality. Let $b(x)$ denote the rate at which work arrives at time x. Thus, if no work arrives at time x, then $b(x) = 0$. If w units of work arrives at time x, then $b(x)$ is w times the Dirac delta function. Thus, for example $\int_a^b b(x)dx$ is just the work that arrives during the interval $[a, b]$.

$$\begin{aligned}
\int_0^d \frac{w(t, et - (e-1)d, d)}{(d-t)}dt &= \int_0^d \frac{1}{(d-t)} \left(\int_{et-(e-1)d}^{t} b(x)dx \right) dt \\
&= \int_0^d \left(\int_x^{(x+(e-1)d)/e} \frac{b(x)}{(d-t)}dt \right) dx \\
&= \int_0^d b(x) \left(\int_x^{(x+(e-1)d)/e} \frac{1}{(d-t)}dt \right) dx
\end{aligned}$$

$$= \int_0^d b(x) \ln \left(\frac{d-x}{d - ((x + (e-1)d)/e)} \right) dx$$
$$= \int_0^d b(x) dx$$
$$= w(0, d).$$

The second equality follows by interchanging the integrals and observing that for each $x \in [0, d)$, the work $b(x)$ contributes to $w(t, et - (e-1)d, d)/(d-t)$ if and only if $x \in (et - (e-1)d, t)$ or equivalently that $t \in [x, (x + (e-1)d)/e]$.

Thus BKP does at least $w(0, d)$ work during the interval $[0, d]$, which is the contradiction we need. □

We now state two inequalities from Hardy et al. [1952] that are critical in our further analysis of BKP.

FACT 4.2 (HARDY'S INEQUALITY, THEOREM 327 [HARDY ET AL. 1952]). *If it is the case that $\alpha > 1$, $f(x) \geq 0$, and $F(x) = \int_0^x f(t)dt$, then*

$$\int_0^\infty \left(\frac{F(x)}{x} \right)^\alpha dx < \left(\frac{\alpha}{\alpha - 1} \right)^\alpha \int_0^\infty f^\alpha(x) dx.$$

The following fact was first proved by Hardy and Littlewood [1930], and later simplified by Gabriel [1931]. It can also be found in Hardy et al. [1952, Theorem 384 and 385].

FACT 4.3. *Suppose that $f(x)$ is nonnegative and integrable in a finite interval $(0, a)$ and that $\bar{f}(x)$ is the rearrangement of $f(x)$ in decreasing order. Let*

$$M(x) = M(x, f) = \max_{0 \leq y < x} \frac{1}{x - y} \int_y^x f(t) dt.$$

Suppose $s(y)$ is any increasing function of y defined for $y \geq 0$. Then,

$$\int_0^a s(M(x)) dx \leq \int_0^a s \left(\frac{1}{x} \int_0^x \bar{f}(t) dt \right) dx. \tag{29}$$

Roughly, if we think of $f(x)$ as the work arriving at time x, then the definition of $M(x)$ resembles the way we define $r(x)$ in our algorithm. This allows us to argue about the function M in terms of f. We use these two facts to prove the following two lemmas. Lemma 4.4 shows how to relate the work in an arbitrary schedule, which will be the optimal schedule in our arguments, to the speed $q(t)$ that upper bounds the speed $v(t)$ used in the definition of BKP. Lemma 4.5 then shows that the energy used by running at speed $q(t)$ is less than some constant times the energy used by an arbitrary schedule.

LEMMA 4.4. *Let $z(t)$ be the speed at time t for some feasible schedule Z. Then*

$$q(t) \leq \max_{t_1 < t \leq t_2} \frac{1}{t_2 - t_1} \int_{t_1}^{t_2} z(t) dt.$$

PROOF. Since Z is feasible, for any times t_1 and t_2, we have that $\int_{t_1}^{t_2} z(t)dt \geq w(t_1, t_2)$. Thus

$$q(t) = \max_{t_1 < t \leq t_2} I(t_1, t_2) \leq \max_{t_1 < t \leq t_2} \frac{1}{t_2 - t_1} \int_{t_1}^{t_2} z(t)dt. \qquad \square$$

LEMMA 4.5. *Let $q(t) \geq 0$ and $y(t) \geq 0$ be functions such that*

$$q(t) \leq \max_{t_1 < t \leq t_2} \frac{1}{t_2 - t_1} \int_{t_1}^{t_2} y(t)dt.$$

Then, it must be the case that

$$\int_t q(t)^\alpha dt \leq 2 \left(\frac{\alpha}{\alpha - 1}\right)^\alpha \int_t y(t)^\alpha dt.$$

PROOF. We split $q(t)$ into two parts. Let

$$l(t) = \max_{t_1} \frac{1}{t - t_1} \int_{t_1}^{t} y(x)dx \qquad \text{such that } t_1 < t.$$

Similarly, let

$$v(t) = \max_{t_2} \frac{1}{t_2 - t} \int_{t}^{t_2} y(x)dx \qquad \text{such that } t \leq t_2.$$

Clearly, $q(t) \leq \max(l(t), v(t))$, and hence $q(t)^\alpha \leq l(t)^\alpha + v(t)^\alpha$. Thus, to bound $\int_t q(t)^\alpha dt$ it suffices to show that both $\int_t l(t)^\alpha dt$ and $\int_t v(t)^\alpha dt$ are upper bounded by the quantity $(\frac{\alpha}{\alpha-1})^\alpha \int_t y(t)^\alpha dt$.

Let us first consider $\int_t l(t)^\alpha dt$. Consider Fact 4.3, with $f(t) = y(t)$, and $s(x) = x^\alpha$. Note that the definition of $M(t)$ exactly corresponds to that of $l(t)$. Let $\bar{y}(t)$ denote the rearrangement of $y(t)$ in nonincreasing order. Then, by Fact 4.3, it follows that

$$\int_t l(t)^\alpha dt \leq \int_t \left(\frac{1}{t} \int_{x=0}^{t} \bar{y}(x)dx\right)^\alpha dt. \tag{30}$$

Now using Fact 4.2 with $f(x) = \bar{y}(x)$,

$$\int_t \left(\frac{1}{t} \int_{x=0}^{t} \bar{y}(x)dx\right)^\alpha dt \leq \left(\frac{\alpha}{\alpha - 1}\right)^\alpha \int_x \bar{y}(x)^\alpha dx. \tag{31}$$

The desired bound on $\int_t l(t)^\alpha dt$ follows by Eqs. (30) and (31) and observing that

$$\left(\frac{\alpha}{\alpha - 1}\right)^\alpha \int_x \bar{y}(x)^\alpha dx = \left(\frac{\alpha}{\alpha - 1}\right)^\alpha \int_x y(x)^\alpha dx.$$

as $\bar{y}$ is a rearrangement of y.

The analysis of $\int_t v(t)^\alpha dt$ is similar. $\square$

4.2. ENERGY. In this section, we show that the BKP algorithm is $O(1)^\alpha$-competitive with respect to energy.

THEOREM 4.6. *The* BKP *algorithm is $2(\frac{\alpha}{\alpha-1})^\alpha \exp(\alpha)$-competitive with respect to energy.*

PROOF. We first note that

$$E[\text{BKP}] \leq \int_t (e\ v(t))^\alpha dt \leq \int_t (e\ q(t))^\alpha dt = \exp(\alpha) \int_t q(t)^\alpha dt.$$

The first inequality follows by the definition of BKP. The second inequality follows by Lemma 1.1. Setting $y(t)$ to be the speed at which YDS works at time t, by Lemma 4.4, it is the case that

$$q(t) \leq \max_{t_1 < t < t_2} \frac{1}{t_2 - t_1} \int_{t_1}^{t_2} y(t) dt.$$

Finally,

$$\int_t q(t)^\alpha dt \leq 2 \left(\frac{\alpha}{\alpha - 1}\right)^\alpha \int_t y(t)^\alpha dt = 2 \left(\frac{\alpha}{\alpha - 1}\right)^\alpha E[\text{OPT}].$$

The first inequality is by Lemma 4.5. The last equality follows since YDS is the optimal energy schedule. □

4.3. MAXIMUM SPEED AND MAXIMUM POWER. It is not hard to see that our online algorithm BKP is e-competitive with respect to the maximum speed (or equivalently $\exp(\alpha)$-competitive for maximum power). We show this formally in Lemma 4.7. We then show in Lemma 4.8 that this is the best possible competitive ratio for the maximum speed.

Consider an online deterministic algorithm A with the property that the schedule produced by A is identical for all values of $\alpha > 1$. We call such an algorithm α-independent. Note all online algorithms considered in this article (BKP, OA and AVR) are α-independent. Given any fixed schedule produced by an α-independent algorithm A, we can choose α large enough such that the total energy for this schedule is essentially determined by the maximum power used by A. Thus the lower bound of e for maximum speed implies that for any arbitrarily small constant $\epsilon > 0$, there is some α large enough, such that A cannot be $(e - \epsilon)^\alpha$-competitive with respect to total energy (when power varies as speed raised to α). This implies that restricted to the class of α-independent algorithms, the base of the exponent in the competitive ratio of BKP cannot be improved.

LEMMA 4.7. *The online algorithm* BKP *is e-competitive with respect to maximum speed.*

PROOF. YDS is the optimum offline algorithm with respect to maximum speed. The maximum speed at which YDS ever works is exactly equal to $\max_t q(t)$. At any time the BKP algorithm works at speed at most $e\ v(t)$ at time t, which is at most $e\ q(t)$ by Lemma 1.1. □

LEMMA 4.8. *For every deterministic online algorithm A that maintains deadline feasibility, there is some input that causes A at some time to run e times faster than the maximum speed of* YDS.

PROOF. Assume A has a competitive ratio R. We show that it must be the case that $R \geq e$. Let $\epsilon > 0$ be an arbitrarily small constant. Let $a(x) = \frac{-1}{(\ln \epsilon)(1-x)}$. The adversary adopts the following strategy: It releases work at the rate of $a(x)$ until some time $t \leq (1 - \epsilon)$, and then after time t no more work is released. The value of t depends on the behavior of the online algorithm. All work has deadline 1.

The adversary will make $t = 1 - \epsilon$ unless A works at too great a speed at some time before $1 - \epsilon$. So assume for the moment that $t = 1 - \epsilon$.

The work that is released between time f and time g is

$$\int_f^g \frac{-1}{(\ln \epsilon)(1-x)} dx = \frac{1}{\ln \epsilon} \ln \frac{1-g}{1-f}. \tag{32}$$

Let $I(t, t_1, t_2) = w(t, t_1, t_2)/(t_2 - t_1)$ denote the intensity of interval $[t_1, t_2]$ restricted to the work that has arrived by time t. Note that for $t \leq 1 - \epsilon$,

$$w(t, 0, 1) = \frac{1}{\ln \epsilon} \ln(1 - t). \tag{33}$$

In particular, if $t = 1 - \epsilon$, then $w(t, 0, 1) = 1$. For some time $t \leq 1 - \epsilon$ and some $k < t$, by Eq. (32)

$$I(t, k, 1) = \frac{1}{(1-k)\ln \epsilon} \ln \frac{1-t}{1-k}. \tag{34}$$

Given a fixed t, we will be interested in the value of $k \in [0, t]$ that maximizes $I(t, k, 1)$. Let $k^*(t)$ denote this value of k. To determine the value of $k^*(t)$, we differentiate $I(t, k, 1)$ with respect to k. We get

$$\frac{d}{dk} I(t, k, 1) = \frac{1}{(1-k)^2 \ln \epsilon} \ln \frac{1-t}{1-k} + \frac{1}{(1-k)^2 \ln \epsilon}. \tag{35}$$

Setting $I'(t, k, 1) = 0$ and solving for k, we have $k = et - (e - 1)$. Also note that $I'(t, k, 1)$ is positive when $k < et - (e - 1)$, and negative when $k > et - (e - 1)$. Also note that $et - (e - 1)$ is always less than t, and is nonnegative when $t \geq \frac{e-1}{e}$. We can then conclude that if $t \geq \frac{e-1}{e}$ then $k^*(t) = et - (e - 1)$, and if $t < \frac{e-1}{e}$ then $k^*(t) = 0$.

Recall that the YDS schedule is optimal with respect to maximum speed. Suppose the adversary stops bringing in more work at time t. Then, the first interval chosen by YDS on this instance will be $[k^*(t), 1]$, and hence $(t, k^*(t), 1)$ is the maximum speed that YDS will run.

If $t < (e - 1)/e$, then $k^*(t) = 0$, and YDS will run at a constant speed of

$$I(t, k^*(t), 1) = I(t, 0, 1) = \frac{\ln(1-t)}{\ln \epsilon},$$

during the time interval $[0, 1]$. Thus, for each time $x \in [0, \frac{e-1}{e}]$, A cannot work at a greater speed than $R \frac{\ln(1-x)}{\ln \epsilon}$. If it did so, then A would not be R-competitive in the case that adversary stops bringing in work at time x.

If $t \in [\frac{e-1}{e}, 1 - \epsilon]$, then $k^*(t) = et - (e - 1)$, and YDS will run at maximum speed of

$$I(t, k^*(t), 1) = I(t, et - (e-1), 1) = \frac{-1}{e(1-t)\ln \epsilon}.$$

Thus, for each time $x \in [\frac{e-1}{e}, 1 - \epsilon]$, A cannot work at a greater speed than $R \frac{-1}{e(1-x)\ln \epsilon}$. If it did so, then A would not be R-competitive in the case that $t = x$.

If $t = 1 - \epsilon$, then YDS will run at a maximum speed of

$$I(1-\epsilon, k^*(1-\epsilon), 1) = \frac{-1}{e\epsilon \ln \epsilon}.$$

Thus, during the time period $[1-\epsilon, 1]$, A cannot work faster than $R\frac{-1}{e\epsilon \ln \epsilon}$. If it did so, then A would not be R-competitive in the case that $t = 1 - \epsilon$.

Now consider the case that $t = 1 - \epsilon$. By the arguments above, the most work that A can get done is

$$\begin{aligned} & R\int_0^{(e-1)/e} \frac{\ln(1-t)}{\ln \epsilon} dt + R\int_{(e-1)/e}^{1-\epsilon} \frac{-1}{e(1-t)\ln \epsilon} dt + R\int_{1-\epsilon}^{1} \frac{-1}{e\epsilon \ln \epsilon} dt \\ = & \frac{R}{\ln \epsilon}\left(\frac{2}{e} - 1\right) + \frac{R}{e \ln \epsilon}(\ln \epsilon + 1) + \frac{-R}{e \ln \epsilon} \\ = & \frac{R}{e} + \left(\frac{2-e}{e}\right)\left(\frac{R}{\ln \epsilon}\right). \end{aligned}$$

Now as ϵ approaches 0, the term $(\frac{2-e}{e})(\frac{R}{\ln \epsilon})$ approaches 0. Thus, the maximum work that A can get done approaches $\frac{R}{e}$, which must be at least $w(0, 1) = 1$. Thus we conclude that R cannot be less than e. □

4.4. TEMPERATURE. We show in Theorem 4.9 that BKP is $O(1)$-competitive with respect to temperature.

THEOREM 4.9. *The online algorithm* BKP *is* $\exp(\alpha)2^{\alpha+1}(6(\frac{\alpha}{\alpha-1})^\alpha + 1)$-*competitive with respect to temperature for all cooling parameters b satisfying* $0 < b < \infty$.

PROOF. Let X be an arbitrary c-interval. As X is arbitrary, by Theorem 2.2, it is sufficient to show that BKP uses at most a factor of $\exp(\alpha)2^{\alpha-1}(6(\frac{\alpha}{\alpha-1})^\alpha + 1)$ times as much energy as $C[\text{OPT}]$ during the interval X. Here, we use $z(t)$ to denote the speed at time t of a fixed arbitrary schedule OPT that uses energy at most $C[\text{OPT}]$ in every c-interval. Let X_-^k (respectively, X_+^k) denote the kth c-interval immediately to the left (respectively, right) of X. That is, the left endpoint of X_-^k is kc units to the leftmost point of X. Let the interval Z be defined to be $X \cup X_-^1 \cup X_+^1$.

As in the proof of Theorem 4.6, we will assume that BKP runs at speed $e\ q(t)$ even if there is no work to do. Thus, we are left to show that

$$\int_{t \in X} q(t)^\alpha dt \le 2^{\alpha-1}\left(6\left(\frac{\alpha}{\alpha-1}\right)^\alpha + 1\right) C[\text{OPT}].$$

Since OPT is feasible, we have by Lemma 4.4

$$q(t) \le \max_{t_1 < t < t_2} \frac{1}{t_2 - t_1} \int_{t_1}^{t_2} z(t) dt.$$

We decompose $z(t)$ as follows: Let $z_1(t) = z(t)$ if $t \in Z$ and 0 at all other times. Let $z_2(t) = z(t) - z_1(t)$ for all t. Let

$$q_1(t) = \max_{t_1 < t \le t_2} \frac{1}{t_2 - t_1} \int_{x=t_1}^{t_2} z_1(x) dx$$

and

$$q_2(t) = \max_{t_1 < t \le t_2} \frac{1}{t_2 - t_1} \int_{x=t_1}^{t_2} z_2(x)dx.$$

Note that $q(t) \le q_1(t) + q_2(t)$ since $z(t) = z_1(t) + z_2(t)$ for all each t. By convexity of the speed-to-power function $P(s)$, it follows that

$$q(t)^\alpha \le (q_1(t) + q_2(t))^\alpha \le 2^{\alpha-1} (q_1(t)^\alpha + q_2(t)^\alpha)$$

and thus

$$\int_{t\in X} q(t)^\alpha dt \le 2^{\alpha-1} \left(\int_{t\in X} q_1(t)^\alpha dt + \int_{t\in X} q_2(t)^\alpha dt \right).$$

We first upper bound $\int_{t\in X} q_1(t)^\alpha dt$. In fact we will upper bound $\int_{t=0}^{\infty} q_1(t)^\alpha dt$. Note that $z_1(t)$ is identically 0 at all points not in Z. Moreover, as Z is an interval of length $3c$, by the definition of $C[\text{OPT}]$ it follows that

$$\int_{t\in Z} z_1(t)^\alpha dt \le 3C[\text{OPT}]. \tag{36}$$

Now we have

$$\begin{aligned} \int_{t=0}^{\infty} q_1(t)^\alpha dt &\le 2 \left(\frac{\alpha}{\alpha - 1} \right)^\alpha \int_0^{\infty} z_1(t)^\alpha dt \\ &= 2 \left(\frac{\alpha}{\alpha - 1} \right)^\alpha \int_{t\in Z} z_1(t)^\alpha dt \\ &\le 6 \left(\frac{\alpha}{\alpha - 1} \right)^\alpha C[\text{OPT}]. \end{aligned} \tag{37}$$

The first inequality follows from Lemma 4.5. The equality follows from the definition of z_1. The final inequality follows from Eq. (36).

We now bound the term $\int_{t\in X} q_2(t)^\alpha dt$. By the definition of $C[\text{OPT}]$, and the convexity of the function s^α for $\alpha \ge 1$, any c-interval contains at most $c(C[\text{OPT}]/c)^{1/\alpha}$ amount of work in OPT. Our next step is to upper bound $q_2(t)$ for any $t \in X$. In particular, we claim that for any t_1 and t_2 such that $t \in X$ and $t_1 < t < t_2$,

$$q_2(t) = \max_{t_1 < t < t_2} \frac{1}{t_2 - t_1} \int_{t_1}^{t_2} z_2(x)dx \le (C[\text{OPT}]/c)^{1/\alpha}.$$

To see this, the crucial observation is that $z_2(t) = 0$ for $t \in Z$. As $Z = X \cup X_-^1 \cup X_+^1$, if $t_1 \in X_-^k$ for $k \ge 1$, then the interval $[t_1, t]$ can contain at most $(k-1) \cdot c(C[\text{OPT}]/c)^{1/\alpha}$ work from z_2. Similarly, if $t_2 \in X_+^k$, then the interval $[t, t_2]$ can contain at most $(k-1) \cdot c(C[\text{OPT}]/c)^{1/\alpha}$ work from z_2. Thus, any interval $[t_1, t_2]$ containing t, and with $(k-1)c \le t_2 - t_1 \le kc$, can contain at most $(k-1) \cdot c(C[\text{OPT}]/c)^{1/\alpha}$ work from z_2. The bound on $q_2(t)$ follows as the integral is just the work from z_2 over the time interval $[t_1, t_2]$.

Thus

$$\int_{t\in X} q_2(t)^\alpha dt \le \int_{t\in X} \left(\left(\frac{C[\text{OPT}]}{c} \right)^{1/\alpha} \right)^\alpha dt = \int_{t\in X} \frac{C[\text{OPT}]}{c} dt = C[\text{OPT}]. \tag{38}$$

Combining Eqs. (37) and (38), we have that for any c-interval X

$$\int_{t\in X} q(t)^\alpha dt \leq \int_{t\in X} 2^{\alpha-1}(q_1(t)^\alpha + q_2(t)^\alpha)dt \leq 2^{\alpha-1}\left(6\left(\frac{\alpha}{\alpha-1}\right)^\alpha + 1\right) C[\text{OPT}].$$

This accomplishes our goal. □

5. *Computing the Optimal Offline Temperature Schedule*

In this section, we consider the offline problem of speed scaling to minimize the maximum temperature. We show how to solve this problem in polynomial time with arbitrary precision using the Ellipsoid algorithm. For basic information on the use of the Ellipsoid algorithm to solve convex problems, see, for example, Nestorov [2003]. We assume a constant $T_{\max}$ that is the thermal threshold for the device. The problem is then to determine whether there is a schedule that is feasible and maintains the invariant that the temperature stays below $T_{\max}$. By binary search, we can then solve the problem of minimizing $T_{\max}$.

Before giving the convex program, we need to make a few definitions. Let $\text{MaxW}(t_x, t_y, T_x, T_y)$ be the maximum work that can be done starting at time t_x at temperature T_x and ending at time t_y at temperature T_y subject to the temperature constraint $T \leq T_{\max}$ throughout the interval $[t_x, t_y]$. Strictly speaking, MaxW is a function of $T_{\max}$ but we suppress this in the notation as we assume throughout that $T_{\max}$ is fixed and given a priori. In order to understand MaxW, we will first need to understand the unconstrained problem $\text{UMaxW}(t_x, t_y, T_x, T_y)$, defined as the maximum possible work that can be done during the interval $[t_x, t_y]$ subject to the boundary constraints that $T(t_x) = T_x$ and $T(t_y) = T_y$. In particular, the temperature at any time is allowed to exceed $T_{\max}$. In Lemma 5.1, we prove the intuitive fact that MaxW and UMaxW are well defined if and only if it is possible to cool as quickly as specified.

LEMMA 5.1. *Suppose that T_x and T_y are at most $T_{\max}$. Each of the quanties* $\text{MaxW}(t_x, t_y, T_x, T_y)$ *and* $\text{UMaxW}(t_x, t_y, T_x, T_y)$ *are well defined if and only if* $T_y \geq T_x \exp(b(t_x - t_y))$.

PROOF. We wish to show that if $T_y < T_x \exp(b(t_x - t_y))$, then there is no feasible solution. Consider the case that the power is zero throughout the interval $[t_x, t_y]$. Then throughout this interval it is the case that $T'(t) = -bT(t)$. Solving this differential equation, we get $T(t) = T_x \exp(b(t_x - t))$ for $t \in [t_x, t_y]$. So in particular, if $T_y < T_x \exp(b(t_x - t_y))$, then there is no feasible solution to either the constrained problem or unconstrained problem since the power must be nonnegative.

We now wish to show that if $T_y \geq T_x \exp(b(t_x - t_y))$, then there is a feasible solution for the constrained problem, and hence trivially, for the unconstrained problem. If $T_y = T_x \exp(b(t_x - t_y))$, then running with power equal to zero is a feasible solution. So assume that $T_y > T_x \exp(b(t_x - t_y))$. Let t_z be some time just after t_x. One feasible solution for the constrained problem is for the temperature to rise to $T_{\max}$ at time t_z, then stay at temperature $T_{\max}$ until the time t_v that solves $T_y = T_{\max} \exp(b(t_v - t_y))$, and then run with power equal to zero until time t_y. □

We are now ready to give the convex program. We divide time into intervals demarcated by the list $t_0, \ldots t_m$ of all release times and deadlines. We introduce a variable T_i that represents $T(t_i)$, the temperature at time t_i. Let $J(i)$ be the set

of tasks j that can feasibly be executed during the time interval $[t_i, t_{i+1}]$, that is, $r_j < t_{i+1}$ and $d_j > t_i$. We introduce a variable $w_{i,j}$, for $j \in J(i)$, that represents the work done on task j during $[t_i, t_{i+1}]$. We can then express our problem as a mathematical program CP in a relatively straightforward way:

$$p_j \leq \sum_{i: j \in J(i)} w_{i,j} \qquad 1 \leq j \leq n \tag{39}$$

$$\sum_{j \in J(i)} w_{i,j} \leq \text{MaxW}(t_i, t_{i+1}, T_i, T_{i+1}) \qquad 1 \leq i \leq m-1 \tag{40}$$

$$T_{i+1} \geq \exp(b(t_i - t_{i+1})) \cdot T_i \qquad 0 \leq i \leq m-1 \tag{41}$$

$$T_i \leq T_{\max} \qquad 0 \leq i \leq m \tag{42}$$

$$0 \leq T_i \qquad 0 \leq i \leq m \tag{43}$$

$$0 \leq w_{i,j} \qquad 0 \leq i \leq m-1, 1 \leq j \leq n. \tag{44}$$

Constraint (39) ensures that enough work is done to finish each job. Constraint (40) ensures that it is feasible to complete the claimed work within an interval. By Lemma 5.1, constraint (41) ensures that the quantity $\text{MaxW}(t_i, t_{i+1}, T_i, T_{i+1})$ is well defined.

LEMMA 5.2. *The mathematical program CP is convex, that is, the feasible region described in CP is convex.*

PROOF. To see that the feasible region is convex, let $\tilde{T}$, and $\hat{T}$ be the temperature curves corresponding to two feasible solutions to this problem. Let $\bar{T} = (\tilde{T} + \hat{T})/2$. The speed curve corresponding to $\bar{T}$ is $\bar{S} = ((\bar{T}' + b\bar{T})/a)^{1/\alpha} = ((\tilde{T}' + \hat{T}' + b\tilde{T} + b\hat{T})/(2a))^{1/\alpha}$. Then, since $x^{1/\alpha}$ is a concave function for $\alpha \geq 1$, $\bar{S} \geq (\tilde{S} + \hat{S})/2$, that is, the average of the two underlying feasible solutions is feasible. □

To apply the Ellipsoid algorithm one needs to give a procedure to determine whether an arbitrary point is feasible, and if not, to determine a separating hyperplane. In particular, if $G \leq 0$ is a violated constraint, then the separating hyperplane for us will be the hyperplane whose normal is the gradient of G evaluated at the current point. The only constraints for which this is not straightforward are the MaxW constraints. So we assume for the rest of this section that all constraints, other than the MaxW constraints, are not violated by the current point, as this is the only interesting case. Our goal is then to explain how to take the gradient of the MaxW function, so that we may compute a separating hyperplane. To better understand MaxW, we first need to understand the unconstrained function UMaxW.

5.1. THE UNCONSTRAINED MAXIMUM WORK PROBLEM. We consider the unconstrained problem $\text{UMaxW}(t_i, T_i, t_{i+1}, T_{i+1})$, where the times and temperatures are arbitrary, other than that we require that $T_{i+1} \geq T_i \exp(b(t_i - t_{i+1}))$ so that a feasible solution exists. Let $\text{UMaxT}(t) = \text{UMaxT}(t_i, t_{i+1}, T_i, T_{i+1})(t)$ denote the temperature as a function of the time t that solves $\text{UMaxW}(t_i, T_i, t_{i+1}, T_{i+1})$ That is, $\text{UMaxT}(t)$ is the temperature curve T that maximizes the quantity

$$\int_{t_i}^{t_{i+1}} P(t)^{1/\alpha} dt = \int_{t_i}^{t_{i+1}} \left(\frac{T'(t) + bT(t)}{a} \right)^{1/\alpha} dt,$$

subject to the constraints that $P(t) \geq 0$, $T(t_i) = T_i$ and $T(t_{i+1}) = T_{i+1}$. This problem falls under the rubric of calculus of variations. We refer the reader to Smith [1974] for the basics on calculus of variations. For notational simplicity, we usually translate time so that $t_0 = 0$. Since temperature is always a function of time, we will drop t in future references to temperature functions.

Let F be the functional

$$F = \left(\frac{T' + bT}{a}\right)^{\frac{1}{\alpha}}$$

and let

$$F_T = \frac{b}{\alpha a^{1/\alpha}}(T' + bT)^{\frac{1}{\alpha}-1}$$

be the partial derivative of F with respect to T, and let

$$F_{T'} = \frac{1}{\alpha a^{1/\alpha}}(T' + bT)^{\frac{1}{\alpha}-1}$$

be the partial derivative of F with respect to T'. Any weak extremum T must satisfy the Euler–Lagrange equation (see, e.g., Smith [1974, page 117]):

$$F_T - \frac{d}{dt}F_{T'} = 0.$$

We call a temperature function T that satisfies the Euler–Lagrange equation an *Euler–Lagrange* curve. We have that

$$\frac{d}{dt}F_{T'} = \frac{1-\alpha}{\alpha^2 a^{1/\alpha}}(T' + bT)^{\frac{1}{\alpha}-2}(T'' + bT').$$

Thus, the Euler–Lagrange equation gives that

$$\frac{b}{\alpha a^{1/\alpha}}(T' + bT)^{\frac{1}{\alpha}-1} - \frac{1-\alpha}{\alpha^2 a^{1/\alpha}}(T' + bT)^{\frac{1}{\alpha}-2}(T'' + bT') = 0.$$

Eliminating common factors and multiplying by $(T' + bT)^{2-1/\alpha}$ gives

$$b^2 T + \left(2 - \frac{1}{\alpha}\right) bT' + \left(1 - \frac{1}{\alpha}\right) T'' = 0.$$

Using the standard Laplace transform technique, we find that the solution to the above differential equation is

$$\text{UMaxT} = T = c\exp(-bt) + d\exp(-bt\alpha/(\alpha - 1)), \tag{45}$$

where the constants c and d are determined by the boundary conditions. Alternatively, one can verify the correctness of this solution by plugging it back into the differential equation.

We now compute the values of c and d. Setting $t = 0$ and $T = T_0$ in Eq. (45), we get $c + d = T_0$. Setting $t = t_1$ and $T = T_1$ in Eq. (45), we get $c\exp(-bt_1) + d\exp(-b\alpha t_1/(\alpha - 1)) = T_1$. Using $c = T_0 - d$, we have $(T_0 - d)\exp(-bt_1) + d\exp(-bt_1\alpha/(\alpha - 1)) = T_1$ or

$$d = \frac{T_0\exp(-bt_1) - T_1}{\exp(-bt_1) - \exp(-bt_1\alpha/(\alpha - 1))} \tag{46}$$

And therefore,

$$c = T_0 - \frac{T_0 \exp(-bt_1) - T_1}{\exp(-bt_1) - \exp(-bt_1\alpha/(\alpha - 1))} \tag{47}$$

This gives a complete description of the curve UMaxT. In particular, the temperature values at two times completely determine the curve UMaxT that passes through these points. Since $T_1 \geq T_0 \exp(-bt_1)$, we can conclude that $d \leq 0$, and hence $c = T_0 - d \geq 0$. We must check that our curve UMaxT is indeed a maximum and not a minimum. This is easily seen by substituting $T_0 - d$ for c in Eq. (45). We have $\text{UMaxT} = T_0 \exp(-bt) + d(\exp(-bt\alpha/(\alpha - 1)) - \exp(-bt))$. Since d and the term in parentheses are both non-positive, their product is non-negative and thus *UMaxT* is at least as great as the nopower curve $T = T_0 \exp(-bt)$. Note that UMaxT is well defined for all times $t \geq t_0$, not just for $t \in [t_0, t_1]$. We will argue about properties of UMaxT on this larger domain.

We now turn our attention to evaluating the work UMaxW done by the curve UMaxT. Differentiating Eq. (45) gives

$$\text{UMaxT}' = T' = -bc \exp(-bt) - \frac{bd\alpha}{\alpha - 1} \exp(-bt\alpha/(\alpha - 1)) \tag{48}$$

Adding Eq. (48) to b times Eq. (45), the term $bc \exp(-bt)$ cancels out, leaving just

$$T' + bT = -\frac{bd}{\alpha - 1} \exp(-bt\alpha/(\alpha - 1)) \tag{49}$$

By Eq. (2), the power function corresponding to the temperature curve UMaxT is then

$$\frac{T' + bT}{a} = -\frac{bd}{a(\alpha - 1)} \exp(-bt\alpha/(\alpha - 1)).$$

Hence, by Eq. (4),

$$\begin{aligned} \text{UMaxW} &= \int_0^{t_1} \left(\frac{T' + bT}{a}\right)^{1/\alpha} dt \\ &= \int_0^{t_1} \frac{1}{a^{1/\alpha}} \left(\frac{-bd}{\alpha - 1}\right)^{1/\alpha} \exp(-bt/(\alpha - 1))dt \\ &= \left(\frac{-d}{a}\right)^{1/\alpha} \left(\frac{b}{\alpha - 1}\right)^{(1/\alpha)-1} (1 - \exp(-bt_1/(\alpha - 1))), \end{aligned} \tag{50}$$

We now state several intuitive, but technical, properties of the UMaxT. When the property is not obvious, we will give some explanation why the property holds. In Fact 5.3, we observe that it is obvious from Eq. (45) for UMaxT that the temperature approaches zero as time goes to infinity. Lemma 5.5 observes that UMaxT has a unique maximum, and characterizes where this maximum occurs. Lemma 5.6 observes that, if $T_0 = T_1$, then the temperature will always stay above T_0. Lemma 5.7 observes that the maximum temperature is a nondecreasing function of t_1. And finally Lemma 5.8 observes that the maximum temperature will exceed $T_{\max}$ for sufficiently large t_1.

FACT 5.3. *For any t_0, t_1, T_0 and T_1, the curve* $\text{UMaxT}(t_0 = 0, T_0, t_1, T_1)(t)$ *approaches* 0 *as t approaches infinity.*

LEMMA 5.4. *Let* $U_1 = \text{UMaxT}(t_0, T_0, t_1, T_1)$ *and* $U_2 = \text{UMaxT}(\hat{t}_0, \hat{T}_0, \hat{t}_1, \hat{T}_1)$ *be the two curves that intersect at distinct points* (t_a, T_a) *and* (t_b, T_b)*, where* $[t_a, t_b] \subseteq [\max(t_0, \hat{t}_0), \min(t_1, \hat{t}_1)]$*. Let* $U_3 = \text{UMaxT}(t_a, T_a, t_b, T_b)$*. Then,* $U_1(t) = U_2(t) = U_3(t)$ *for all* $t \geq t_a$*.*

PROOF. Follows immediately from the uniqueness of the maximum determined by two points. □

LEMMA 5.5. *Consider the curve* $\text{UMaxT} = \text{UMaxT}(t_i, t_{i+1}, T_i, T_{i+1})$*.*

—*The curve* UMaxT *has at most one point where* UMaxT′*, the derivative of* UMaxT *with respect to time* t*, is 0.*

—*If* $\text{UMaxT}'(x) = 0$ *for some time* x*, then* $\text{UMaxT}'(t) < 0$ *for all* $t > x$*.*

—*If* $\text{UMaxT}'(0) \leq 0$*, then the maximum of* UMaxT *is at* $t = 0$*.*

—*If* $\text{UMaxT}'(t_1) \geq 0$*, then the maximum of* UMaxT *is at* $t = t_1$*.*

—*If* $\text{UMaxT}'(0) > 0$ *and* $\text{UMaxT}'(t_1) < 0$*, then the maximum of* UMaxT *is at the unique point* $t_x \in [0, t_1]$ *where* $\text{UMaxT}'(t_x) = 0$*.*

PROOF. If $d = 0$, it follows from Eq. (48) that $T' = -bc\exp(-bt) = -bT_0\exp(-bt)$ and hence that $\text{UMaxT}' \leq 0$ everywhere, and it is easy to see that the above claims hold. Now assume that $d < 0$. Using $c = T_0 - d$ and multiplying by $\exp(bt)/b$, Eq. (48) can be rewritten as

$$\frac{\exp(bt)\text{UMaxT}'}{b} = -T_0 + d - \frac{d\alpha}{\alpha - 1}\exp(-bt/(\alpha - 1)).$$

Note that since $\exp(bt)/b > 0$ for all t, the sign of UMaxT' is the same as the sign of the right hand side of the above equation, which is a strictly decreasing function of t. In particular, observe that this implies that UMaxT' cannot go from negative to positive. All the above claims are then simple consequences of this observation. □

LEMMA 5.6. *Consider two points* $(0, T_0)$ *and* (t_1, T_0)*. Let* L_1 *denote the constant temperature curve between* $(0, T_0)$ *and* (t_1, T_0)*. Let* L_2 *denote a temperature curve such that* $L_2(0) = L_2(t_1) = T_0$ *and* $L_2(t) \leq L_1(t)$ *for* $t \in [0, t_1]$*. Then at least as much work is completed by following* L_1 *as by following* L_2*.*

PROOF. Consider a temperature curve T such that $T(0) = T(t_1) = T_0$. Applying Eq. (3), the energy used during $[0, t_1]$ by following T is:

$$\frac{b}{a}\int_0^{t_1} T(t)dt$$

Under the temperature constraint $T(t) \leq T_0$, this integral is maximized with $T(t) = T_0$ throughout, and thus the total energy E_2 of L_2 is at most the total energy E_1 of L_1. By convexity, for a given energy budget, working at constant power maximizes the work done; that is, the maximum work that can be done by any curve L with energy at most E is $t_1(E/t_1)^{1/\alpha}$ and is achieved by staying at constant power. As L_1 stays at constant temperature, and hence constant power, it follows that the work completed by following the curve L_2 is no more than that of L_1. □

LEMMA 5.7. *Let* T_0 *and* T_1 *be fixed, and consider the class of unconstrained curves* $\text{UMaxT}(t_0 = 0, T_0, t_1, T_1)$ *for* $t_1 \geq 0$*. In particular, for* $t_1 < t_2$*, let* U_1 *and* U_2

denote the curves $UMaxT(t_0 = 0, T_0, t_1, T_1)(t)$ *and* $UMaxT(t_0 = 0, T_0, t_2, T_1)(t)$, *respectively. Then,*

—*If* $T_0 \geq T_1$, *we have that for all times* $t \geq 0$, $U_1(t) \leq U_2(t)$.
—*If* $T_0 \leq T_1$, *then for all* t *such that* $0 \leq t \leq t_1$, $U_1(t) \leq U_2(t + t_2 - t_1)$.

In particular, this implies the following:

—*The maximum temperature reached by the curve* $\text{UMaxT}(t_0 = 0, T_0, t_1, T_1)$ *in the interval* $[0, t_1]$ *is a nondecreasing function of* t_1.
—*If* $T_0 \geq T_1$, *then* $\text{UMaxT}'(t_0 = 0, T_0, t_1, T_1)(0)$ *is a nondecreasing function of* t_1.
—*If* $T_0 \leq T_1$, *then* $\text{UMaxT}'(t_0 = 0, T_0, t_1, T_1)(t_1)$ *is a nonincreasing function of* t_1.

PROOF. We first consider the case that $T_0 \geq T_1$. As $U_1(0) = U_2(0)$, it must be true that either $U_1(t) = U_2(t)$ for all $t \geq 0$ or one of these temperature curves is always larger than the other, since by Lemma 5.4, if they intersect at two points they are the same. By the mean value theorem, there must be a time $\tilde{t} \in [0, t_1]$ where $U_1'(\tilde{t}) \leq 0$. By Lemma 5.5, this implies that $U_1'(x) < 0$ for all $x > \tilde{t}$. In particular, since $t_2 > t_1 \geq \tilde{t}$, this implies that $U_1(t_2) < U_1(t_1) = T_1 = U_2(t_2)$. Hence, $U_1(t) \leq U_2(t)$. It is then obvious that the maximum temperature must be a nondecreasing function of the ending time, and the derivative at time 0 must be a nondecreasing function of the ending time.

We now consider the case that $T_0 \leq T_1$. Instead of U_1, it will be more convenient to work with the curve $U_3 = \text{UMaxT}(t_0 = t_2 - t_1, T_0, t_2, T_1)$, which is just the curve U_1 translated to the right by $t_2 - t_1$ time units. Now since $U_3(t_2) = U_2(t_2) = T_1$, either U_3 and U_2 are identical or one these temperature curves is always larger than the other. If $U_3(t) > U_2(t)$, then this would in particular mean that $U_2(t_2 - t_1) < U_3(t_2 - t_1) = U_1(0) = T_0$. But, by Lemma 5.5, it is clear that $U_2(t) \geq T_0$ for all $t \in [0, t_2]$. Thus, it must be the case that $U_3(t) \leq U_2(t)$ for all $t \in [t_2 - t_1, t_2]$. It is then obvious that the maximum temperature must be a nondecreasing function of the ending time and the derivative at the ending time must be a nonincreasing function of the ending time. □

LEMMA 5.8. *Consider the temperature curve* $\text{UMaxT}(t_0 = 0, T_0, t_1, T_1)$ *as* $T_0 \leq T_{\max}$ *and* $T_1 \leq T_{\max}$ *are fixed, and* t_1 *is varied. Then, there is a finite time* $\tilde{t}$ *such that the maximum temperature reached by the curve* $\text{UMaxT}(t_0 = 0, T_0, t_1, T_1)$ *is at least* $T_{\max}$ *for all* $t_1 \geq \tilde{t}$.

PROOF. Consider the curve $U = \text{UMaxT}(0, T_0, 1, T_{\max})$. As $U(t)$ approaches 0 as t approaches infinity, there exists a finite time $\hat{t} \geq 1$ such that $U(\hat{t}) = T_1$. Consider the curve $U_2 = \text{UMaxT}(0, T_0, \hat{t}, T_1)$. As U and U_2 share the points $(0, T_0)$ and $(\hat{t}, T_1)$, U and U_2 must be identical. Thus, $U_2 = \text{UMaxT}(0, T_0, \hat{t}, T_1)$ attains a maximum temperature of at least $T_{\max}$ and hence by Lemma 5.7, the maximum temperature reached by $\text{UMaxT}(0, T_0, t_1, T_1)$ is at least $T_{\max}$ for all $t_1 \geq \hat{t}$. □

5.2. THE TEMPERATURE CONSTRAINED MAXIMUM WORK PROBLEM. We now turn our attention to $\text{MaxW}(t_0 = 0, T_0, t_1, T_1)$ and $\text{MaxT}(t_0 = 0, T_0, t_1, T_1)$, again requiring that $T_1 \geq T_0 \exp(-bt_1)$ so that a feasible solution exists. Also we now require that both T_0 and T_1 are at most $T_{\max}$. That is, we assume the existence of a temperature constraint $T \leq T_{\max}$. It is known that when such a global constraint is added the solution can be decomposed into subcurves, where each subcurve is

either an Euler–Lagrange curve corresponding to some unconstrained problem, or else follows the boundary [Smith 1974, page 240]. We are fortunate in our case that the portion of MaxT, for which MaxT $= T_{\max}$, is a single line segment. This is an immediate consequence of Lemma 5.6.

We now know that either MaxT = UMaxT, or MaxT consists of three parts: an Euler–Lagrange curve up to $T_{\max}$, a line segment at $T_{\max}$, and an Euler–Lagrange curve down to T_1. Note that either or both of the Euler–Lagrange curves may have length zero. The Euler–Lagrange curve up to $T_{\max}$ is of length zero if and only if $T_0 = T_{\max}$. The Euler–Lagrange curve down from $T_{\max}$ is of length zero if and only if $T_1 = T_{\max}$.

Before we characterize the curve MaxT in Lemma 5.11, we define some notation. Let $\tilde{t}$ denote the supremum of values of t_1 for which the curve UMaxT$(0, T_0, t_1, T_1)$ does not exceed temperature $T_{\max}$ at any time. The time $\tilde{t}$ is well defined and finite by Lemma 5.8. Note that $\tilde{t}$ is a function of T_0 and T_1. Define γ to be the time (unique by Lemma 5.5) at which the maximum temperature is attained on the curve UMaxT$(0, T_0, \tilde{t}, T_1)$. Define $\beta = \tilde{t} - \gamma$. We now provide alternate characterizations of γ and β, and show that these points occur where the derivative of UMaxT is zero.

LEMMA 5.9. *γ is the largest value of t_1 for which the maximum temperature attained by the curve* UMaxT$(0, T_0, t_1, T_{\max})$ *during the interval* $[0, t_1)$ *is no more than $T_{\max}$. Similarly, β is the largest value of t_1 for which the maximum temperature attained by the curve* UMaxT$(0, T_{\max}, t_1, T_1)$ *in* $(0, t_1]$ *is no more than $T_{\max}$.*

PROOF. By Lemma 5.4, UMaxT$(0, T_0, \tilde{t}, T_1)$ = UMaxT$(0, T_0, \gamma, T_{\max})$; call this curve $U_1(t)$. Let $\gamma' > \gamma$ and consider the curve $U_2(t)$ = UMaxT$(0, T_0, \gamma', T_{\max})$. We know that $U_1(\gamma') < U_2(\gamma') = T_{\max}$, so using Lemma 5.4, again we know that $U_2(\gamma) > U_1(\gamma) = T_{\max}$. The proof for β is similar. □

LEMMA 5.10. UMaxT$(0, T_0, \gamma, T_{\max})'(\gamma) = 0$ *and* UMaxT$(0, T_{\max}, \beta, T_1)'(0) = 0$.

PROOF. Again using UMaxT$(0, T_0, \tilde{t}, T_1)$ = UMaxT$(0, T_0, \gamma, T_{\max})$, we see that the derivative must be 0 at $t = \gamma$ since the maximum is attained there.

The argument for β is similar. □

LEMMA 5.11. *Consider the curve* MaxT$(0, T_0, t_1, T_1)$. *Let γ and β be defined as above. If $t_1 \le \gamma + \beta$, then* MaxT = UMaxT. *If $t_1 > \gamma + \beta$, then the curve* MaxT *travels along the curve* UMaxT$(0, T_0, \gamma, T_{\max})$, *then stays at $T_{\max}$ until time $t_1 - \beta$, and finally travels along the curve* UMaxT$(t_1 - \beta, T_{\max}, t_1, T_1)$.

PROOF. Assume first that $t_1 \le \tilde{t} = \gamma + \beta$. By Lemma 5.7, the maximum temperature reached by UMaxT$(0, T_0, t_1, T_1)$ during $[0, t_1]$ is a nondecreasing function of t_1. Hence, by the definition of $\tilde{t}$, at no time can the function UMaxT$(0, T_0, t_1, T_1)$ exceed $T_{\max}$. Therefore, MaxT = UMaxT.

Now consider the case that $t_1 > \gamma + \beta$. Let t_x and t_y denote the first and last times, respectively, at which MaxT equals $T_{\max}$. We will show that $t_x = \gamma$ and $t_y = t_1 - \beta$.

By definition of t_x and t_y, the curve MaxT restricted to $[0, t_x]$ is identical to the curve UMaxT$(0, T_0, t_x, T_{\max})$ and the curve MaxT restricted to $[t_y, t_1]$ is identical to the curve UMaxT$(0, T_{\max}, t_1 - t_y, T_1)$. If $t_x > \gamma$ then, by Lemma 5.7 and Lemma 5.9, this contradicts the definition of γ. A similar argument shows that $t_y \ge t_1 - \beta$.

Now, suppose for contradiction that $t_x < \gamma$. Since $t_y \geq t_1 - \beta \geq \gamma$, it follows that the point $(\gamma, T_{\max})$ lies on MaxT. Hence, for the non-zero length time interval $[t_x, \gamma]$, it must be the case that MaxT $= T_{\max}$. This contradicts the work optimality of UMaxT$(0, T_0, \gamma, T_{\max})$, and hence the work optimality of MaxT$(0, T_0, t_1, T_1)$. Again, in a similar manner, we can obtain a contradiction from the assumption that $t_y > t_1 - \beta$. □

We now explain how to explicitly compute γ and β. Consider the curve $U =$ UMaxT$(0, T_0, \gamma, T_{\max})$. Recall from Lemma 5.10 that $U'(\gamma) = 0$. Setting $U'(\gamma) = 0$ in Eq. (48) gives

$$c + \frac{d\alpha}{\alpha - 1} \exp(-b\gamma/(\alpha - 1)) = 0.$$

Plugging in the values of c and d from Eqs. (46) and (47), we have that γ is the unique solution of the equation

$$\left(T_0 - \frac{T_0 \exp(-bt_1) - T_1}{\exp(-bt_1) - \exp(-bt_1\alpha/(\alpha - 1))}\right) + \frac{\alpha}{\alpha - 1} \exp(-b\gamma/(\alpha - 1)) \left(\frac{T_0 \exp(-bt_1) - T_1}{\exp(-bt_1) - \exp(-bt_1\alpha/(\alpha - 1))}\right) = 0.$$

By multiplying through by $(\exp(-bt_1) - \exp(-bt_1\alpha/(\alpha - 1)))$, and aggregating like terms, this is equivalent to

$$T_0 \exp(-b\gamma\alpha/(\alpha - 1)) + (\alpha - 1)T_{\max} - \alpha T_{\max} \exp(-b\gamma/(\alpha - 1)) = 0. \quad (51)$$

Similarly, the curve $U =$ UMaxT$(0, T_{\max}, \beta, T_1)$ satisfies $U'(0) = 0$. We can see from Eq. (48) that $U'(0) = 0$ is equivalent to

$$c + \frac{d\alpha}{\alpha - 1} = 0.$$

Plugging in the values of c and d by Eqs. (46) and (47), we have that β is the unique solution of the equation

$$\left(T_0 - \frac{T_0 \exp(-bt_1) - T_1}{\exp(-bt_1) - \exp(-bt_1\alpha/(\alpha - 1))}\right) + \left(\frac{T_0 \exp(-bt_1) - T_1}{\exp(-bt_1) - \exp(-bt_1\alpha/(\alpha - 1))}\right)\left(\frac{\alpha}{\alpha - 1}\right) = 0.$$

By multiplying through by $(\exp(-bt_1) - \exp(-bt_1\alpha/(\alpha - 1)))$, and aggregating like terms, this is equivalent to

$$T_{\max} \exp(-b\beta\alpha/(\alpha - 1)) + (\alpha - 1)T_1 - \alpha T_1 \exp(-b\beta/(\alpha - 1)) = 0 \quad (52)$$

5.3. COMPUTING A SEPARATING HYPERPLANE. We are now finally ready to explain how to compute a separating hyperplane for a violated MaxW constraint. Consider an arbitrary point where each T_i takes the value $\hat{T}_i$, and each $w_{i,j}$ takes the value $\hat{w}_{i,j}$. Assume that the ith MaxW constraint is violated. Given the values of $t_i, t_{i+1}, \hat{T}_i$ and $\hat{T}_{i+1}$, we can compute the values γ and β by binary search using the Eqs. (51) and (52). Note that the left-hand sides of these equations are monotone functions of γ and β respectively.

First, consider the case that $t_{i+1} - t_i \leq \gamma + \beta$. Then we know the maximum temperature constraint is not relevant here. Now Eq. (50) gives that

$$\text{MaxW} = -(d/a)^{1/\alpha} \left(\frac{b}{\alpha - 1}\right)^{1/\alpha - 1} (1 - \exp(-b(t_{i+1} - t_i)/(\alpha - 1))),$$

where by Eq. (46)

$$\begin{aligned} d \;=\; & (T_i \exp(-b(t_{i+1} - t_i)) - T_{i+1}) \\ & \times (\exp(-b(t_{i+1} - t_i)) - \exp(-b(t_{i+1} - t_i)\alpha/(\alpha - 1))). \end{aligned}$$

We can then determine whether the ith MaxW constraint is violated. If this constraint is violated, to compute a separating hyperplane let G be a function of T_i, T_{i+1}, and $w_{i,j}$ for $j \in J(i)$ defined as

$$G = \sum_{j \in J(i)} w_{i,j} - \text{UMaxW}(t_i, t_{i+1}, T_i, T_{i+1}).$$

The ith MaxW constraint is then equivalent to $G \leq 0$. A separating hyperplane is then the plane whose normal is the gradient of G evaluated at the current point. Note that one can easily differentiate G with respect to all variables.

Now consider the case that $t_{i+1} - t_i \geq \gamma + \beta$. MaxW is given by the work done by the Euler–Lagrange curve between (t_i, T_i) and $(t_i + \gamma, T_{\max})$, plus the work done on the constant temperature curve at $T_{\max}$ between time $(t_i + \gamma)$ and time $(t_{i+1} - \beta)$, plus the work done by the Euler–Lagrange curve from $(t_{i+1} - \beta, T_{\max})$ to (t_{i+1}, T_{i+1}). Using Eq. (4) to compute the work done by a constant temperature curve and Eq. (50) to compute the work done by the two Euler–Lagrange curves, we have

$$\begin{aligned} \text{MaxW} \;=\; & -(d_1/a)^{1/\alpha} \left(\frac{b}{\alpha - 1}\right)^{1/\alpha - 1} (1 - \exp(-b\gamma/(\alpha - 1))) \\ & + (t_{i+1} - t_i - \gamma - \beta) \left(\frac{bT_{\max}}{a}\right)^{1/\alpha} \\ & - \left(\frac{d_2}{a}\right)^{1/\alpha} \left(\frac{b}{\alpha - 1}\right)^{1/\alpha - 1} (1 - \exp(-b\beta/(\alpha - 1))), \end{aligned}$$

where, by Eq. (46)

$$d_1 = \frac{T_i \exp(-b\gamma) - T_{\max}}{\exp(-b\gamma) - \exp(-b\gamma\alpha/(\alpha - 1))}$$

and

$$d_2 = \frac{T_{\max} \exp(-b\beta) - T_{i+1}}{\exp(-b\beta) - \exp(-b\beta\alpha/(\alpha - 1))}$$

We can determine whether this MaxW constraint is violated. If this constraint is violated, to compute a separating hyperplane let G be a function of T_i, T_{i+1}, and $w_{i,j}$ for $j \in J(i)$ defined as

$$G = \sum_{j \in J(i)} w_{i,j} - \text{MaxW}(t_i, T_i, t_{i+1}, T_{i+1}).$$

The ith MaxW constraint is then equivalent to $G \leq 0$. A separating hyperplane is then the plane whose normal is the gradient of G evaluated at the current point. Computing the gradient of G, though tedious, is straightforward with the possible exceptions of differentiating γ with respect to T_i and differentiating β with respect to T_{i+1}. These partial derivatives can be computed using the Eqs. (51) and (52) that define γ and β, respectively.

Consider Eq. (51) where T_0 is replaced by T_i, namely,

$$T_i \exp(-b\gamma\alpha/(\alpha-1)) + (\alpha-1)T_{\max} - \alpha T_{\max} \exp(-b\gamma/(\alpha-1)) = 0.$$

Differentiating this equation with respect to T_i yields

$$\exp\left(-\frac{b\alpha\gamma}{\alpha-1}\right) - T_i\left(\frac{b\alpha}{\alpha-1}\right)\exp\left(-\frac{b\alpha\gamma}{\alpha-1}\right)\frac{d\gamma}{dT_i} + \frac{\alpha T_{\max} b}{\alpha-1}\exp\left(-\frac{b\gamma}{\alpha-1}\right)\frac{d\gamma}{dT_i} = 0.$$

Solving for $\frac{d\gamma}{dT_i}$ yields

$$\frac{d\gamma}{dT_i} = \frac{-(\alpha-1)\exp(-b\gamma)}{\alpha b(T_{\max} - T_i \exp(-b\gamma))}.$$

Consider Eq. (52) where T_1 is replaced by T_{i+1}, namely,

$$T_{\max} \exp(-b\beta\alpha/(\alpha-1)) + (\alpha-1)T_{i+1} - \alpha T_{i+1} \exp(-b\beta/(\alpha-1)) = 0.$$

Differentiating Eq. (52) with respect to T_{i+1} yields

$$T_{\max} \cdot \frac{-b\alpha}{\alpha-1}\exp\left(\frac{-b\beta\alpha}{\alpha-1}\right)\frac{d\beta}{dT_{i+1}} + (\alpha-1) - \alpha\exp\left(\frac{-b\beta}{\alpha-1}\right) + \frac{\alpha b T_{i+1}}{\alpha-1}\exp\left(\frac{-b\beta}{\alpha-1}\right)\frac{d\beta}{dT_{i+1}} = 0.$$

Solving for $\frac{d\beta}{dT_{i+1}}$ yields

$$\frac{d\beta}{dT_{i+1}} = \frac{(\alpha-1)^2 - \alpha(\alpha-1)\exp(-b\beta/(\alpha-1))}{\alpha b(T_{\max}\exp(-\alpha b\beta/(\alpha-1)) - T_{i+1}\exp(-b\beta/(\alpha-1)))}$$

6. *Conclusion*

In this article, we have initiated the theoretical study of speed scaling to manage temperature. We assumed a fixed ambient temperature, and that the device cools according to Newton's law of cooling. We have observed that the maximum temperature is within a factor of two of the energy used over an interval with length inversely proportional to the cooling parameter in Newton's law. We have identified the concept of a cooling-oblivious algorithm as an algorithm that is simultaneously $O(1)$-competitive with respect to temperature for all cooling parameters, and observed that cooling-oblivious algorithms are also $O(1)$-competitive with respect to energy and maximum power. We showed that the optimal energy schedule YDS is cooling-oblivious and introduced the first known online cooling-oblivious

algorithm BKP. Further, we have shown that BKP is optimally competitive with respect to maximum power.

We believe that speed scaling to manage energy and temperature is an area deserving further research attention. This area is both academically interesting and has practical applications. The most obvious way to proceed is to consider speed scaling versions of other scheduling problems. We can take essentially any scheduling problem, and consider an energy management version, or a temperature management version. We would then get a dual-criteria optimization problem, in which the first objective is the original quality of service objective for the scheduling problem, and the second objective is either energy or temperature. By solving many such problems, the hope would be that it would be possible to build up a better understanding of speed scaling, and to build an algorithmic toolkit of analysis and design techniques that are useful for speed scaling problems.

ACKNOWLEDGMENTS. We wish to thank Mary Jane Irwin, Bruce Childers, Rami Melhem, Daniel Mosse, Patchrawat Uthaisombut, Gerhard Woeginger, Maxim Sviridenko, Sandy Irani, and Cliff Stein for interesting discussions on power aware computing. We wish to thank the anonymous reviewers for their particularly detailed and helpful reviews.

REFERENCES

APPLE. 2005. Why apple chose intel. http://www.business2.com/b2/web/articles/0,17863,1084781,00.html.

BANSAL, N., KIMBREL, T., AND PRUHS, K. 2004. Dynamic speed scaling to manage energy and temperature. In *Proceedings of the IEEE Syposium on Foundations of Computer Science*. IEEE Computer Society Press, Los Alamitos, CA.

BOYD, S., AND VANDENBERGHE, L. 2004. *Convex Optimization*. Cambridge University Press, Cambridge, MA.

BROOKS, D. M., BOSE, P., SCHUSTER, S. E., JACOBSON, H., KUDVA, P. N., BUYUKTOSUNOGLU, A., WELLMAN, J.-D., ZYUBAN, V., GUPTA, M., AND COOK, P. W. 2000. Power-aware microarchitecture: Design and modeling challenges for next-generation microprocessors. *IEEE Micro 20,* 6, 26–44.

BUTTAZZO, G. 1997. *Hard Real-Time Computing Systems*. Kluwer.

CORMEN, T., LEISERSON, C., RIVEST, R., AND STEIN, C. 2001. *Introduction to Algorithms*. MIT Press, Cambridge, MA.

CRUSOE. 2002. Crusoe tm550/tm5800 thermal design guide. http://crusoe.com/crusoe_docs/TM5800_ThermDesGuide_6-18-02.pdf.

ELLIS, C. S. 1999. The case for higher-level power management. In *Proceedings of the IEEE Workshop on Hot Topics in Operating Systems*. IEEE Computer Society Press, Los Alamitos, CA.

GABRIEL, R. M. 1931. An additional proof of a maximal theorem of hardy and littlewood. *J. London Math. Soc. 6*, 163–166.

HARDY, G. H., AND LITTLEWOOD, J. 1930. A maximal theorem with function-theoretic applications. *Acta Math. 54*, 81–116.

HARDY, G. H., LITTLEWOOD, J. E., AND POLYA, G. 1952. *Inequalities*. Cambridge University Press, Cambridge, MA.

IRANI, S., GUPTA, R. K., AND SHUKLA, S. 2003. Algorithms for Power Savings. In *Proceedings of the ACM/SIAM Symposium on Discrete Algorithms*. 37–46.

IRANI, S., AND PRUHS, K. R. 2005. Algorithmic problems in power management. *SIGACT News 36,* 2, 63–76.

KWON, W., AND KIM, T. 2003. Optimal voltage allocation techniques for dynamically variable voltage processors. In *Design Automation*. 125–130.

LI, M., LIU, B. J., AND YAO, F. F. 2006a. Min-energy voltage allocation for tree-structured tasks. *J. Combin. Optim. 11,* 3, 305–319.

LI, M., YAO, A. C., AND YAO, F. F. 2006b. Discrete and continuous min-energy schedules for variable voltage processors. In *Proceedings of the National Academy of Sciences USA*. Vol. 103. 3983–3987.

LI, M., AND YAO, F. F. 2005. An efficient algorithm for computing optimal discrete voltage schedules. *SIAM J. Comput. 35*, 658–671.

MARKOFF, J. 2004. Intel's big shift after hitting technical wall, *New York Times* (May 17, 2004). http://www.nytimes.com/2004/05/17/business/17intel.html?ex=1400212800&en=482e58801aa02ede&ei=5007&partner=USERLAND.

MUDGE, T. 2001. Power: A first-class architectural design constraint. *Computer 34,* 4, 52–58.

NESTOROV, Y. 2003. Introductory lectures on convex programming.

SERGENT, J. E., AND KRUM, A. 1998. *Thermal Management Handbook*. McGraw-Hill, New York.

SKADRON, K., STAN, M. R., HUANG, W., VELUSAMY, S., SANKARANARAYANAN, K., AND TARJAN, D. 2003. Temperature-aware microarchitecture. In *Proceedings of the International Symposium on Computer Architecture*. 2–13.

SMITH, D. R. 1974. *Variational Methods in Optimization*. Prentice-Hall, Englewood Cliffs, NJ.

TIWARI, V., SINGH, D., RAJGOPAL, S., MEHTA, G., PATEL, R., AND BAEZ, F. 1998. Reducing power in high-performance microprocessors. In *Proceedings of the Design Automation Conference*. 732–737.

YAO, F., DEMERS, A., AND SHENKER, S. 1995. A scheduling model for reduced CPU energy. In *Proceedings of the IEEE Syposium on Foundations of Computer Science*. IEEE Computer Society Press, Los Alamitos, CA, 374–382.

YUN, H., AND KIM, J. 2003. On energy-optimal voltage scheduling for fixed priority hard real-time systems. *ACM Trans. Embed. Comput. Syst. 2,* 3, 393–430.

RECEIVED AUGUST 2005; REVISED JULY 2006; ACCEPTED NOVEMBER 2006