Speed Scaling with a Solar Cell

Nikhil Bansal[1], Ho-Leung Chan[2], and Kirk Pruhs[2] *

[1] IBM T.J. Watson Research, P.O. Box 218, Yorktown Heights, NY.
nikhil@us.ibm.com
[2] Computer Science Department, University of Pittsburgh.
{hlchan,kirk}@cs.pitt.edu

Abstract. We consider the setting of a device that obtains it energy from a battery and some regenerative source such as a solar cell. We consider the speed scaling problem of scheduling a collection of tasks with release times, deadlines, and sizes so as to minimize the energy recharge rate of the regenerative source. This is the first theoretical investigation of speed scaling for devices with a regenerative energy source. We show that the problem can be expressed as a polynomial sized convex program. We show that using the KKT conditions, one can obtain an efficient algorithm to verify the optimality of a schedule. We show that the energy optimal YDS schedule, is 2-approximate with respect to the recharge rate. We show that the online algorithm BKP is $O(1)$-competitive with respect to recharge rate.

1 Introduction

Chip manufacturers such as Intel, AMD and IBM have made it a priority to redesign their chips to consume less power and to provide various hardware and software capabilities for power management. All of these chip manufacturers make chips that use dynamic speed scaling as a power management technique. Typically, the power consumed varies as the cube of the processor speed, and hence this can yield significant energy/temperature reductions.

The first theoretical investigation of speed scaling algorithms was in the seminal paper by Yao, Demers and Shenker [15]. They considered the problem of feasibly scheduling a collection of jobs with arbitrary release times, deadlines and sizes to minimize the energy consumed, and gave both offline and online algorithms. Subsequently, there has been a lot of work on improving these results and extending them to optimize various other objectives such as, flow time, throughput and so on [6, 3, 4, 7, 10, 1, 14, 16, 13, 2, 12].

All of these theoretical investigations of speed scaling as an energy management technique involve problems where the goal is to minimize the total energy used. This would be the appropriate objective if the energy source was a battery, and the goal was to extend the battery's lifetime. But some devices, most notably some sensors, also contain technologies that allow then to harvest energy from their environment. The most common energy harvesting technology

* Supported in part by NSF grants CNS-0325353, CCF-0514058 and IIS-0534531.

is solar cells. Some sensors also contain technology that allows then to scavenge energy from ambient vibrations. To give some feel for state of technology, batteries can store about 1 Joule of energy per cubic millimeter, while solar cells can provide approximately 100 micro-Watt per square centimeter in bright sunlight, and vibration devices can provide nano-Watts per cubic millimeter.

In this paper we initiate a study of speed scaling for energy management in devices that contain both a battery and an energy harvesting device, which we will henceforth assume for simplicity is a solar cell. Our goal is understand how the presence of a solar cell will affect the resulting speed scaling problems. For simplicity, we assume that the solar cell generates energy at a fixed rate (although many of a our results apply to a more general setting). We consider the deadline feasibility problem introduced in [15] because it is the most studied, and probably the best understood, speed scaling problem in the literature. We consider the objective minimizing the recharge rate, subject to the deadline feasibility constraint.

1.1 Related Results

Before explaining our results, let us recap what is known about speed scaling with a deadline feasibility constraint on battery only devices. The standard assumption is that when the processor is run at speed s, then the power consumption is $P(s) = s^\alpha$ for some constant $\alpha > 1$ [9, 3, 11]. For CMOS based devices, which will likely remain the dominant technology for the near term future, the well known cube-root rule is that the speed s is roughly proportional to the cube-root of the power P, or equivalently, $P(s) = s^3$. [15] gave an optimum greedy algorithm YDS. For the online version, they gave an algorithm AVR and showed that the competitive ratio of AVR is at most $2^{\alpha-1}\alpha^\alpha$. It was recently shown that AVR is in fact $(2-\epsilon)\alpha^\alpha$ competitive, where ϵ goes to zero as α increases. [15] also proposed another algorithm OA, which was shown by [3] to be α^α competitive. [3] gave another online algorithm BKP and showed that is was $2(\alpha/(\alpha-1))^\alpha e^\alpha$ competitive(this is the best known competitive ratio for large α). It is also known that any algorithm must have competitive ratio of at least $e^{\alpha-1}/\alpha$ [5] and hence the result cannot be improved substantially. Improved results for the practically interesting cases of $\alpha = 2$ and $\alpha = 3$ have also been obtained recently [5].

1.2 Our Results

We consider both the offline and online versions of the minimum recharge rate problem. In Section 3 we show that the offline problem can be expressed as a convex program and hence can be solved to any desired accuracy by using standard techniques such as the Ellipsoid Method. We then explore this convex program further in Section 4. We analyze the consequences of the KKT conditions for optimality for this program and obtain an equivalent set of combinatorial properties that are both necessary and sufficient for a solution to be optimal. This

gives us several insights into the structure of an optimum solution and also allows us to obtain an efficient test to determine whether a solution is optimum or not.

In Section 5, we show that the YDS algorithm, which is optimal for the no-recharge case, is in fact a 2-approximation for the minimum recharge rate problem. We also show that this bound is tight. In the special case when the release times and deadlines of jobs are similarly ordered, we show that YDS is optimal. Finally, in Section 6, we consider the online setting, and show that the BKP algorithm is $O(1)$ competitive for the problem. In particular, BKP achieves a competitive ratio of $4(\frac{\alpha}{\alpha-1})^\alpha e^\alpha$.

In summary, our results seem to suggest that speed scaling algorithms that perform well when the energy source is a battery, should also perform reasonably well when there is also a regenerative energy source. As evidence of this, the optimal energy schedule YDS, is an $O(1)$-approximate schedule for recharge rate, and the algorithm BKP is $O(1)$-competitive for both energy and recharge rate. The KKT conditions reveal that cutting the recharge-rate optimal schedule at the points where the battery is empty, partitions the schedule into energy optimal YDS schedules. So there is some relationship between energy optimal and recharge-rate optimal schedules. However, computing the recharge-rate optimal schedule is still seemingly much harder than computing an energy optimal schedule because it is not clear how to partition the work amongst these YDS subschedules of the recharge-rate optimal schedule.

1.3 Formal Problem Statement

We consider a system that consists of a battery that can be charged at a rate of R, i.e. the energy reserve of the battery increases by R units per unit time, from an external source such as a solar cell. The battery is used to run a processor.

The input is a collection of jobs, where each job i has an integer release time r_i when it arrives into the system, an integer work w_i that must be performed to complete the job, and an integer deadline d_i by which this work must be completed. In the online version of the problem, the scheduler learn about job i at time r_i. At this point it also learns w_i and d_i. A schedule specifies at each time which job is run, and at what speed. Note that if the processor runs at speed s, the power is consumed at rate $R - s^\alpha$. Thus the energy level at any time t' is $\int_{t=0}^{t'}(R-s(t)^\alpha)dt$. We say that a schedule is feasible if the system never runs out of power. That is, at any time the energy level of the battery is non-negative. In the minimum recharge rate problem, that we consider in this paper, the goal is to construct a feasible schedule that minimizes the recharge rate R required. An online algorithm A is c-competitive, or equivalently has competitive ratio c, if with recharge rate R, A misses the deadline of some job, then this instance is not feasibly schedulable with recharge rate R/c.

We assume that the energy level of the battery at $t = 0$ is 0. This is without loss of generality; given an instance I with battery level E_0 at $t = 0$, we can construct another instance I' with battery level 0 at $t = 0$ and with all the jobs

in I shifted forward in time by E_0/R units. We also assume that there is no upper bound on the amount of energy that the battery can hold.

2 Preliminaries

We begin by reviewing the algorithms YDS and BKP for energy efficient scheduling, as well as the KKT conditions for convex programming.

The algorithm YDS. Given an instance I, let the density of any time interval $[t, t']$ be defined as $\text{den}(t, t') = w(t, t')/(t' - t)$, where $w(t, t')$ is the total size of jobs with release time at least t and deadline at most t'. Intuitively, $\text{den}(t, t')$ is the minimum average speed at which any feasible algorithm must work during the interval $[t, t']$. YDS applies the following steps until all jobs are scheduled: Let $[t, t']$ be the highest density interval. The speed is set to $\text{den}(t, t')$ during $[t, t']$. Then the instance is modified such that times $[t, t']$ did not exist. That is, all deadlines $d_i > t$ are modified $d'_i = \max\{t, d_i - (t' - t)\}$, and all release times $r_i > t$ are modified to $r'_i = \max\{t, r_i - (t' - t)\}$, and the process is repeated. The jobs are scheduled in the earliest deadline first order.

We note that each job is run at fixed speed in the YDS schedule. This speed is fixed with respect to time, but may be different for different jobs. Moreover, if job i runs at speed s, then the speed at any time during $[r_i, d_i]$ is at least s.

Another useful (but non-algorithmic) view of YDS is the following. Start with an arbitrary schedule, and keep improving the schedule as follows until no longer possible: If some job i runs at time t when the processor speed is s, and there is some other time t' where job i can run (i.e. $r_i \leq t' \leq d_i$) but the speed at t' is less than s, then move infinitesimally small work of job i from t to t'.

The algorithm BKP. At any time t and $t_1 < t \leq t_2$, let $w(t, t_1, t_2)$ be the total size of jobs that have release time at least t_1, deadline at most t_2 and have been released by time t. Intuitively, it is an estimation for the density of the interval $[t_1, t_2]$ based on the jobs released by t. Let $v(t)$ be defined by

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

Then, at any time t, BKP runs at speed $e \cdot v(t)$ and processes the unfinished job with earliest deadline. BKP is known to be $2(\frac{\alpha}{\alpha - 1})^\alpha e^\alpha$-competitive [3].

The KKT conditions. Consider a convex program

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

Assume the functions f_i are all differentiable. Let λ_i, $i = 1, \ldots, n$ be a variable (Lagrangian multiplier) associated with f_i. Then the necessary KKT conditions for solutions x and λ to be primal and dual optimal are:

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

$$\lambda_i \geq 0 \qquad i = 1, \ldots, n \tag{2}$$

$$\lambda_i f_i(x) = 0 \qquad i = 1, \ldots, n \tag{3}$$

$$\nabla f_0(x) + \sum_{i=1}^{n} \lambda_i \nabla f_i(x) = 0 \tag{4}$$

We refer to the above four equations as Condition 1, 2, 3 and 4 of the KKT conditions, respectively. Condition 3 is commonly known as complementary slackness. If the program is strictly feasible, i.e., there is some point x where $f_i(x) < 0$ for $i = 1, \ldots, n$, then these conditions are also sufficient [8].

3 Convex Programming Formulation

In this section, we give a convex program to find the minimum recharge rate, which implies that the problem can solved optimally in polynomial time. For simplicity of description, we give a pseudo-polynomial sized time indexed formulation, but as we show later the size can be made polynomial.

Let I be any job sequence. Recall that the release time, size and deadline of a job i are denoted as r_i, w_i and d_i, respectively. Without loss of generality, we assume that the release time and deadline of each job are integers. Let $w_{i,j}$ be the amount of work done on job i during time $[j-1, j]$. Then, minimizing the recharge rate R can be written as the following program CP.

$$\min R \quad s.t.$$

$$w_i - \sum_{t: r_i < t \leq d_i} w_{i,t} \leq 0 \qquad \forall i = 1, 2, \ldots \tag{5}$$

$$\sum_{t: t \leq j} \Big(\sum_{x: r_x < t \leq d_x} w_{x,t} \Big)^{\alpha} - Rj \leq 0 \qquad \forall j = 1, 2, \ldots \tag{6}$$

$$-w_{i,j} \leq 0 \qquad \forall i, j = 1, 2, \ldots \tag{7}$$

The constraints (5) enforce that each job is completed. Constraints (6) enforce that the battery is non-negative at any integral time. We need to show that the optimal solution for CP gives the minimum recharge rate. This is not completely obvious since CP does not explicitly enforce that the battery does not run out of energy at some non-integral time.

Lemma 1. *The optimal solution R for CP is the minimum recharge rate to complete all jobs in I.*

Proof. Constraints (5) guarantee that each job is completed. Consider any time j and let E_{j-1} and E_j be the energy in the battery at time $j-1$ and j, respectively. Let $s = \sum_{i: r_i < j \leq d_i} w_{i,j}$ be the speed during $[j-1, j]$ and let $R' = s^{\alpha}$. Then for any $w \in [0, 1]$, at time $j - 1 + w$ the energy in the battery is $E_{j-1} + w(R - R')$. If $R - R' \geq 0$, then $E_{j-1} + w(R - R') \geq E_{j-1} \geq 0$; else if $R - R' < 0$, then $E_{j-1} + w(R - R') \geq E_{j-1} + (R - R') = E_j \geq 0$. Hence, if $E_{j-1} \geq 0$ and $E_j \geq 0$,

then the battery is not depleted at any time during $[j-1, j]$. This implies that the schedule returned by CP is feasible. Conversely, every feasible schedule must satisfy the constraints stated in CP. Hence R is the minimum recharge rate. □

Theorem 1. *The recharge-rate problem can be solved by a convex program.*

Proof. Lemma 1 shows that the problem can be solved by CP. It remains to show that CP is convex. The objective function as well as the constraints (5) and (7) are linear, and hence convex. For constraints (6), we note that the function $f(x) = x^\alpha$ is convex and the sum of convex functions is also convex. Hence, the constraints (6) are also convex. □

Since CP is convex, we can apply the standard methods to determine R to any desired accuracy. We remark that CP has pseudo-polynomial size as the number of variables and equations are depend upon the time horizon. However, given the insight provided by the KKT conditions in the next section, we can reduce the size to polynomial by considering only those time points that are the release time or deadline of a job. We can redefine $w_{i,j}$ to be the work done on job i between the $(j-1)$-th and j-th time points. We also need to modify the left size of (6) such that the speed during that interval is $\sum_i w_{i,j}$ divided by the length of the interval. The resulting convex program gives the minimum recharge rate.

4 Recognizing an Optimal Schedule

We now study the consequences of the KKT conditions when applied to CP and the structural properties they impose on an optimal solution. This will lead to a simple algorithm to recognize an optimal schedule.

For our convex program CP, the constraints are differentiable and strictly feasible, so the KKT conditions are both necessary and sufficient for a solution to be optimal. Associate a dual variable α_i for the equation for job i in constraints (5) of CP. Associate a dual variable β_j for the equation for time j in constraints (6). Associate a dual variable $\gamma_{i,j}$ for the equation for job i and time j in (7). Now consider the four KKT conditions 1-4. Condition 1 states that the optimal solution satisfies the constraints of CP (and hence is feasible). Condition 2 states that α_i, β_j and $\gamma_{i,j}$ are non-negative. For Condition 3, the equations become

$$\alpha_i \left(w_i - \sum_{t: r_i < t \le d_i} w_{i,t} \right) = 0 \qquad \forall i = 1, 2, \ldots \tag{8}$$

$$\beta_j \left(\sum_{t: t \le j} \Big(\sum_{x: r_x < t \le d_x} w_{x,t} \Big)^\alpha - Rj \right) = 0 \qquad \forall j = 1, 2, \ldots \tag{9}$$

$$\gamma_{i,j} w_{i,j} = 0 \qquad \forall i, j = 1, 2, \ldots \tag{10}$$

Equation (9) implies that β_j is positive only if the battery is empty at time j. Equation (10) implies that $\gamma_{i,j}$ is zero if job i is processed during $[j-1, j]$.

We now consider Condition 4. We list out separately the terms corresponding to each partial derivative in the gradient. When the derivative is taken with respect to R, we obtain that

$$1 - \sum_j j\beta_j = 0 \tag{11}$$

When the derivative is taken with respect to variable $w_{i,j}$, we obtain that

$$\alpha_i + \gamma_{i,j} = \alpha(\sum_{x:r_x<j\leq d_x} w_{x,j})^{\alpha-1}(\sum_{t:j\leq t} \beta_t) \qquad \forall i, j = 1, 2, \ldots \tag{12}$$

Note that $\sum_{x:r_x<j\leq d_x} w_{x,j}$ is the speed of the schedule during $[j-1, j]$. Hence, the above equation gives a relationship of how the speed depends on α, β and γ.

As we now show, these KKT conditions are equivalent to the following combinatorial properties that a schedule must satisfy.

Lemma 2. *Let I be any job sequence and S be a schedule for I. Then, S is optimal if and only if it satisfies the following 4 properties.*

1. *S completes all jobs and the battery is not depleted at any integral time.*
2. *There exists time $T > 0$ such that the battery has zero energy at T and no job with deadline after T is processed before T.*
3. *Let T be the smallest time satisfying Property 2. Let $0 = t_0 < t_1 < \ldots < t_k = T$ be times up to T such that the battery has zero energy. Then, for each interval $[t_{y-1}, t_y]$, $y = 1, \ldots, k$, the work processed during $[t_{y-1}, t_y]$ is scheduled using the YDS schedule.*
4. *There exists multipliers $m_1, m_2, \ldots, m_{k-1} \geq 1$ for $t_1, t_2, \ldots, t_{k-1}$ with the following property. Let $s_{i,y}$ denote the speed that job i is processed during $[t_{y-1}, t_y]$. Then, if i is processed during both $[t_{y-1}, t_y]$ and $[t_{y'-1}, t_{y'}]$, $y < y'$, we have $s_{i,y'}/s_{i,y} = m_y m_{y+1} \cdots m_{y'-1}$.*

Remark: We note that the multipliers $m_1, \ldots, m_{k-1}$ are independent of the jobs, and hence the ratios $s_{iy'}/s_{iy}$ are identical for each job i.

Proof. We first show sufficiency, that is, if S satisfies the 4 properties stated in Lemma 2, then S also satisfies the KKT conditions for CP and hence S is optimal. We then show that these properties are also necessary. In particular, we show that if S does not satisfy these properties, then there is another feasible schedule with a smaller recharge rate, implying that S is not optimal. We now give the details.

Consider the values of $w_{i,j}$ and R implied by S. The first property above implies that $w_{i,j}$ and R satisfy the constraints of CP and hence Condition 1 of the KKT conditions. The remaining three properties allow us to determine the values of α, β and γ satisfying Condition 2, 3 and 4 of the KKT conditions. We first give some intuition. Assume job i is processed during $[j-1, j]$ for some time j. By (10) it follows that $\gamma_{i,j} = 0$ and by (12) it follows that if $\sum_{t:j\leq t} \beta_t > 0$,

then the speed during $[j-1, j]$ is

$$\sum_{x:r_x<j\leq d_x} w_{x,j} = \left(\frac{\alpha_i}{\alpha \sum_{t:j\leq t} \beta_t}\right)^{1/(\alpha-1)} \tag{13}$$

Note that α_i is a constant for job i. Hence, if job i is processed during $[j-1, j]$ with speed s and is processed during $[j'-1, j']$ with speed s', then we have that $s'/s = (\sum_{t:j\leq t} \beta_t)^{1/(\alpha-1)}/(\sum_{t:j'\leq t} \beta_t)^{1/(\alpha-1)}$, or equivalently $\sum_{t:j\leq t} \beta_t = (s'/s)^{\alpha-1} \sum_{t:j'\leq t} \beta_t$. It means that in any optimum schedule, when a job is processed during two different time intervals, the ratio of speeds should be determined by the values of β. Note that this is exactly what property 4 in Lemma 2 also guarantees. This allows us set the values β can be set consistently. We now give the calculation to derive α, β and γ from the properties of Lemma 2.

Consider $t_1 < \ldots < t_k = T$ and $m_1, \ldots, m_{k-1}$ as defined by the third and fourth property of Lemma 2. We set β_j to zero for all $j \notin \{t_1, \ldots, t_k\}$. Note that it satisfies requirement (9) of the KKT conditions. For $j \in \{t_1, \ldots, t_k\}$, we set β_j such that they satisfy the following system of linear equations.

$$\sum_{t:t_y\leq t} \beta_t = (m_y)^{\alpha-1} \sum_{t:t_{y+1}\leq t} \beta_t \qquad y = 1, \ldots, k-1 \tag{14}$$

$$1 - \sum_{y=1}^{k} t_y \beta_{t_y} = 0 \tag{15}$$

This system has a unique non-negative solution, as (14) can be written as $\beta_{t_y} = ((m_y)^{\alpha-1} - 1) \sum_{t:t_{y+1}\leq t} \beta_t$. Hence, by considering the equation from $y = k-1$ down to $y = 1$, we can express each of $\beta_{t_{k-1}}, \ldots, \beta_{t_1}$ in terms of β_{t_k}. Substituting these expressions into (15), we obtain a unique solution for β_{t_k}, as well as β_{t_y} for $y = k-1, \ldots, 1$. Note that $\beta_{t_k} > 0$ and $\beta_{t_y} \geq 0$ for $y = k-1, \ldots, 1$. This completely specifies β. Note that by (15), the values of β satisfy requirement (11) of the KKT conditions.

To calculate the values of α, we consider each job i. Let $[j_i - 1, j_i]$ be the earliest time interval during which i is processed. Then, α_i is set to

$$\alpha_i = \alpha(\sum_{x:r_x<j_i\leq d_x} w_{x,j_i})^{\alpha-1}(\sum_{t:j_i\leq t} \beta_t). \tag{16}$$

Note that $\alpha_i \geq 0$. As all jobs are completed by S, the KKT condition given by (8) is satisfied for any value of α. Finally, to calculate the values of γ, we consider any job i and any time j, $r_i < j \leq d_i$. We set $\gamma_{i,j}$ to

$$\gamma_{i,j} = \alpha(\sum_{x:r_x<j\leq d_x} w_{x,j})^{\alpha-1}(\sum_{t:j\leq t} \beta_t) - \alpha_i. \tag{17}$$

This guarantees that the KKT conditions specified by (12) are satisfied. It remains to show that $\gamma_{i,j} \geq 0$ and (10) is satisfied. This is trivially true if i has

not been processed until time t_k, because $\gamma_{i,j} = 0$ in that case. If i has been processed by time t_k, recall that $[j_i - 1, j_i]$ is the first interval that i is processed. Consider any time $[j-1, j]$ such that $r_i < j \leq d_i$. Let y and y' be values that $t_{y-1} < j_i \leq t_y$ and $t_{y'-1} < j \leq t_{y'}$. Then, by (17) and (14) we have that

$$\begin{aligned}
\gamma_{i,j} &= \alpha(\sum_{x:r_x<j\leq d_x} w_{x,j})^{\alpha-1}(\sum_{t:j\leq t} \beta_t) - \alpha_i \\
&= \alpha\left(\frac{\sum_{x:r_x<j\leq d_x} w_{x,j}}{m_y m_{y+1} \ldots m_{y'-1}}\right)^{\alpha-1} (m_y m_{y+1} \ldots m_{y'-1})^{\alpha-1}(\sum_{t:t_{y'}\leq t} \beta_t) - \alpha_i \\
&= \alpha\left(\frac{\sum_{x:r_x<j\leq d_x} w_{x,j}}{m_y m_{y+1} \ldots m_{y'-1}}\right)^{\alpha-1} (\sum_{t:t_y\leq t} \beta_t) - \alpha_i \qquad (18)
\end{aligned}$$

If i is processed during $[j-1, j]$, then by property 4 in Lemma 2, the speed $\sum_{x:r_x<j\leq d_x} w_{x,j}$ equals $m_y m_{y+1} \ldots m_{y'-1}$ times that during $[j_i - 1, j_i]$. Hence, (18) implies that

$$\gamma_{i,j} = \alpha(\sum_{x:r_x<j_i\leq d_x} w_{x,j_i})^{\alpha-1}(\sum_{t:j_i\leq t} \beta_t) - \alpha_i$$

which is identically equal to 0 by (16). Thus the KKT conditions given by (10) are satisfied in this case. Finally consider the case when i is not processed during $[j-1, j]$. By property 3 in Lemma 2, the schedule during $[t_{y-1}, t_y]$ is a YDS schedule. Hence, it must be that the speed $\sum_{x:r_x<j\leq d_x} w_{x,j}$ is at least as large as $m_y m_{y+1} \ldots m_{y'-1}$ times the speed during $[j_i - 1, j_i]$. By (18) and (16), this implies that $\gamma_{i,j} \geq 0$ if i is not processed during $[j-1, j]$. This completes the proof that the 4 properties above implies the KKT conditions.

We now show that the properties in Lemma 2 are necessary. The first property is clearly necessary for any feasible solution. For the second property, first we note that the battery must be empty at least once at some time $t > 0$, otherwise the recharge rate can be easily reduced. Now, consider all the times t when the battery is 0. If the second property is not satisfied, then for every such t, there is some job that has deadline after t, but receives some processing by t. Then, it is easy to see that by moving (appropriately chosen) infinitesimally small quantities of this work further in time results in another feasible schedule with a smaller recharge rate. For the third property, assume that the work during some interval $[t_{y-1}, t_y]$ is not scheduled according to YDS. Since the energy remaining at any time j such that $t_{y-1} < j < t_y$ is strictly positive, there is some infinitesimally small movement of work such that after the movement, the total energy consumed during $[t_{y-1}, t_y]$ decreases and the energy at each intermediate time remains positive. This also implies that the energy remaining becomes positive at t'_y for all $y' > y$. Now for each $t_{y'}$ such that $y' < y$, there is some job with deadline after $t_{y'}$ and processed by $t_{y'}$. We move an infinitesimally small amount of this work further in time, which results in a schedule with a smaller recharge rate. For the fourth property, our previous discussion shows that it is implied by the KKT conditions, hence it is necessary. □

Hence, to determine whether a schedule S minimizes the recharge rate, we can simply check for the above 4 properties. This gives our main result for this section.

Theorem 2. *Let I be any job sequence. Given a schedule S, we can determine in polynomial time whether S minimizes the recharge rate.*

Proof. Properties 1, 2 and 3 of Lemma 2 can be checked easily in polynomial time. To check Property 4 we can write as system of linear equations as follows. Let i be a job that is processed in both $[t_{y-1}, t_y]$ and $[t_{y'-1}, t_{y'}]$ for some $y < y'$ with speed $s_{i,y}$ and $s_{i,y'}$ respectively. We include an equation $\ln m_y + \ln m_{y+1} + \ldots + \ln m_{y'-1} = \ln(s_{i,y'}/s_{i,y})$. By considering all jobs and time intervals, we obtain a set of linear equations with variables of the type $\ln m_y$. There is a solution to these equations if and only if Property 4 is satisfied. □

5 Performance of YDS

In this section, we analyze the YDS schedule and show that it requires a recharge rate at most 2 times that of the optimum schedule, and that this bound is the best possible. Later we show that YDS is optimum for instances where the job deadlines and release times are ordered similarly.

Let I be any job sequence, and let OPT denote some optimum schedule. We first state a simple observation used to lower bound the energy usage of OPT.

Lemma 3. *Consider the YDS schedule for I. Let s be any speed and t be any time such that YDS has a speed at least s at t and has a speed strictly less than s immediately after t. Let J_a be the set of all jobs YDS has processed using a speed at least s until time t. Then, the energy usage of OPT for processing jobs in J_a by time t is at least that of YDS.*

Proof. We first notice that all jobs in J_a have deadlines at most t and are actually completed by YDS by time t. Furthermore, in the YDS schedule for I, jobs in J_a are processed identically as they would be in the YDS schedule for the instance J_a, i.e. instance I with jobs in $I \setminus J_a$ removed. Therefore, YDS completes J_a using the minimum amount of energy. OPT needs to complete J_a by time t and must use at least the same amount of energy. □

We are now ready to prove the main result of this section.

Theorem 3. *YDS is a 2-approximation for minimizing the recharge rate.*

Proof. For any schedule if $E(t)$ denotes the energy usage until time t, then by definition, the recharge rate required is $\max_t E(t)/t$. Consider some instance where OPT has recharge rate r, but YDS is infeasible even with recharge rate $2r$. Let t' be the earliest time when YDS runs out of energy, and let t be the earliest time after t' when the speed of YDS falls below $r^{1/\alpha}$. Consider the times during $[0, t']$ where speed of YDS is $\geq r^{1/\alpha}$, and let E be the total energy used

during these times. Since YDS is working at speed strictly less than $r^{1/\alpha}$ during other times in $[0, t']$, it follows that the total energy used by YDS during $[0, t']$ is strictly less than $E + rt'$.

We now apply Lemma 3 at time t with $s = r^{1/\alpha}$, and define J_a accordingly. As the energy used by YDS for jobs in J_a is at least $E + (t - t')r$, it follows that the energy usage of OPT on jobs in J_a is at least $E + (t - t')r$. However, as OPT has recharge rate r, it follows that $rt \geq E + (t - t')r$ and hence $E \leq rt'$. However as the total energy used by the YDS during the interval $[0, t']$ is strictly less than $E + rt'$, this implies that the total energy used by YDS is strictly less than $2rt'$ which contradicts the assumption that YDS ran out of energy at t' with recharge rate $2r$. □

We remark that the YDS schedule can be computed in $O(n^2 \log n)$ time [14], where n is the number of jobs. Therefore, this gives a polynomial time constant factor approximation algorithm for the recharge rate minimization problem. We also note that the above bound for YDS cannot be improved.

Observation 1 *The approximation ratio of YDS is at least 2 for the minimum recharge rate problem.*

Proof. Let ϵ be an arbitrarily small parameter such that $1/\epsilon$ is an integer. Consider the instance with two jobs, where job 1 has size $1/\epsilon^{1/\alpha}$, release time $1/\epsilon - 1$ and deadline $1/\epsilon$ and job 2 has size $1/\epsilon^2 - 1/\epsilon$, release time 0 and deadline $1/\epsilon^2$. Consider the schedule that stays idle during 0 to $1/\epsilon - 1$, finishes job 1 during $[1/\epsilon - 1, 1/\epsilon]$ consuming energy $1/\epsilon$, and then works at speed 1 during $[1/\epsilon, 1/\epsilon^2]$ on job 2. It is easily verified that it is feasible with a recharge rate of 1. YDS on the other hand, works at speed $(1/\epsilon^2 - 1/\epsilon)/(1/\epsilon^2 - 1) \approx 1 - \epsilon$ during $[0, 1/\epsilon - 1]$ on job 1. As it needs at least $1/\epsilon$ energy during $[1/\epsilon - 1, 1/\epsilon]$ for job 2, it is easily verified that a recharge rate of $2 - O(\epsilon)$ is necessary. □

Well-ordered Jobs: We also consider the special case where the jobs are *well-ordered*, i.e., for every jobs i_1, i_2, if the release time of i_1 is no later than the release time of i_2, then the deadline of i_1 is no later than the deadline of i_2. We can show that YDS is optimal for job sequences that are well-ordered.

Theorem 4. *For well-ordered job sequences, YDS minimizes the recharge rate.*

Proof. Let $E(t)$ be the energy usage of YDS up to time t, and let $R = \max_t \frac{E(t)}{t}$ be the recharge rate. Also let T be the latest time such that $\frac{E(T)}{T} = R$ and let $s = R^{1/\alpha}$. Note that YDS has speed at least s at T and has speed strictly less than s immediately after T. Let i be the job that is completed by YDS at T. Since the job sequence is well-ordered, every job with deadline later than i has release time at least that of i. By the property of the YDS schedule, these jobs are scheduled completely after T. It means that every job processed by YDS until time T has deadline at most T. Any optimum solution OPT also needs to complete these jobs by time T and the energy usage is at least $E(T)$. Hence, the recharge rate of OPT is at least that of YDS. □

6 An Online Algorithm

We now show that the BKP algorithm is constant competitive in the online setting. For any job sequence I, it is known that BKP uses no more than $2(\frac{\alpha}{\alpha-1})^\alpha e^\alpha$ times the total energy used by YDS [3]. In the following lemma, we show that in fact at any intermediate time t, the energy usage of BKP up to t is at most $2(\frac{\alpha}{\alpha-1})^\alpha e^\alpha$ times that of YDS.

Lemma 4. *Consider any job sequence I. Let $E(t)$ be the energy usage of YDS up to time t, and $E'(t)$ be that of BKP. Then, at any time t, $E'(t) \leq 2(\frac{\alpha}{\alpha-1})^\alpha e^\alpha E(t)$.*

Proof. For the proof we define another algorithm ALG, that any time t runs at speed $p(t) = e \cdot \max_{t_1,t_2} w(t,t_1,t_2)/(t_2-t_1)$, where $t_1 < t \leq t_2$ and $w(t,t_1,t_2)$ denotes the amount of work that has release time at least t_1 and has deadline at most t_2. Recall that the speed of BKP at any time t is no greater than that of ALG. We will show that at any time t, the energy usage of ALG up to t is no greater than $2(\frac{\alpha}{\alpha-1})^\alpha e^\alpha E(t)$, which implies the lemma.

It is shown in [3] that ALG is $2(\frac{\alpha}{\alpha-1})^\alpha e^\alpha$-competitive in total energy usage. To show that the same guarantee holds for any intermediate time, consider any job sequence I and any time t. Let I' be a job sequence constructed based on the YDS schedule for I: At any time $j < t$, a job is released with deadline $j+1$ and size equal to the speed of YDS during $[j, j+1]$, and the last job is released at time $t-1$. As ALG is $2(\frac{\alpha}{\alpha-1})^\alpha e^\alpha$-competitive for total energy, the energy usage of ALG up to t with input I' is at most $2(\frac{\alpha}{\alpha-1})^\alpha e^\alpha$ times that of YDS for I'. To argue back about the job sequence I, we note that at any time up to t, YDS has the same speed for input I and I'. For ALG, we note that at any time $i < t$, the quantity $p(i)$ for input I is at most that for input I', and hence the speed of ALG for I is at most that for I'. This implies that for I, the energy usage of ALG up to time t is at most $2(\frac{\alpha}{\alpha-1})^\alpha e^\alpha$ times that of YDS. Since I and t are arbitrary, the lemma follows. □

By Theorem 3 and Lemma 4 we obtain that

Theorem 5. *The BKP algorithm is $4(\frac{\alpha}{\alpha-1})^\alpha e^\alpha$-competitive for minimizing the recharge rate.*

References

1. Susanne Albers and Hiroshi Fujiwara. Energy-efficient algorithms for flow time minimization. *ACM Transactions on Algorithms*, 3(4), 2007.
2. Susanne Albers, Fabian Müller, and Swen Schmelzer. Speed scaling on parallel processors. In *Proc. ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 289–298, 2007.
3. N. Bansal, T. Kimbrel, and K. Pruhs. Speed scaling to manage energy and temperature. *JACM*, 54(1), 2007.
4. Nikhil Bansal, David Bunde, Ho Leung Chan, and Kirk Pruhs. Average rate speed scaling. In *Latin American Theoretical Informatics Symposium*, 2008.

5. Nikhil Bansal, Ho-Leung Chan, Kirk Pruhs, and Dmitriy Rogozhnikov-Katz. Improved bounds for speed scaling in devices obeying the cube-root rule. In *IBM Research Technical Report*, 2007.
6. Nikhil Bansal and Kirk Pruhs. Speed scaling to manage temperature. In *STACS*, pages 460–471, 2005.
7. Nikhil Bansal, Kirk Pruhs, and Cliff Stein. Speed scaling for weighted flow time. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 805–813, 2007.
8. Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.
9. David M. Brooks, Pradip Bose, Stanley E. Schuster, Hans Jacobson, Prabhakar N. Kudva, Alper Buyuktosunoglu, John-David Wellman, Victor Zyuban, Manish Gupta, and Peter W. Cook. Power-aware microarchitecture: Design and modeling challenges for next-generation microprocessors. *IEEE Micro*, 20(6):26–44, 2000.
10. Ho-Leung Chan, Wun-Tat Chan, Tak-Wah Lam, Lap-Kei Lee, Kin-Sum Mak, and Prudence W. H. Wong. Energy efficient online deadline scheduling. In *ACM-SIAM Symposium on Discrete algorithms*, pages 795–804, 2007.
11. Sandy Irani, Sandeep Shukla, and Rajesh Gupta. Online strategies for dynamic power management in systems with multiple power saving states. *Trans. on Embedded Computing Sys.*, 2003. Special Issue on Power Aware Embedded Computing.
12. W.-C. Kwon and T. Kim. Optimal voltage allocation techniques for dynamically variable voltage processors. In *Proc. ACM-IEEE Design Automation Conf.*, pages 125–130, 2003.
13. Minming Li, Becky Jie Liu, and Frances F. Yao. Min-energy voltage allocation for tree-structured tasks. *Journal of Combinatorial Optimization*, 11(3):305–319, 2006.
14. Minming Li and Frances F. Yao. An efficient algorithm for computing optimal discrete voltage schedules. *SIAM J. on Computing*, 35:658–671, 2005.
15. F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced CPU energy. In *Proc. IEEE Symp. Foundations of Computer Science*, pages 374–382, 1995.
16. H.S. Yun and J. Kim. On energy-optimal voltage scheduling for fixed priority hard real-time systems. *ACM Trans. on Embedded Computing Systems*, 2(3):393–430, 2003.