Lagrangian Duality based Algorithms in Online Energy-Efficient Scheduling

Nguyen Kim Thang*

IBISC, Université d'Evry-Val-d'Essonne, Evry, France

Abstract

We study online scheduling problems in the general energy model of speed scaling with power down. The latter is a combination of the two extensively studied energy models, speed scaling and power down, toward a more realistic one. Due to the limits of the current techniques, only few results have been known in the general energy model in contrast to the large literature of the previous ones.

In the paper, we consider a Lagrangian duality based approach to design and analyze algorithms in the general energy model. We show the applicability of the approach to problems which are unlikely to admit a convex relaxation. Specifically, we consider the problem of minimizing energy with a single machine in which jobs arrive online and have to be processed before their deadlines. We present an α^α-competitive algorithm (whose the analysis is tight up to a constant factor) where the energy power function is of typical form $z^\alpha + g$ for constants $\alpha > 2$ and $g \geq 0$. Besides, we also consider the problem of minimizing the weighted flow-time plus energy. We give an $O(\alpha/\ln \alpha)$-competitive algorithm; that matches (up to a constant factor) to the currently best known algorithm for this problem in the restricted model of speed scaling.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Online Scheduling, Energy Minimization, Speed Scaling and Power-down, Lagrangian Duality

Digital Object Identifier 10.4230/LIPIcs.SWAT.2016.20

1 Introduction

Energy-efficient algorithms [1] have gained considerable interest in the algorithmic community in the last decade. Many results and techniques have been developed to reduce energy while optimizing some objectives, especially in the domain of scheduling. There are two widely studied models in energy-aware scheduling: *power down* and *speed scaling*. In the power down model, a machine could be set in one of several states, which vary from low-power states to high-power ones and there are transition costs from one state to another. Depending on the required tasks to be performed, an algorithm has to decide when to make a transition and to which states to switch. The goal is to minimize the total energy consumption. In the speed scaling model, there is no state but now one can choose an appropriate speed to process required tasks. The energy power of a machine is a convex function of its speed. An algorithm needs to specify the machine speed and a policy to process jobs in order to optimize some quality of service and the consumed energy. Each model captures partly (but complementarily) a more realistic setting — the speed scaling with power down model. In the latter, a machine could be set in different states and its speed could also be varied as a function of required tasks.

* Research supported by the ANR project OATA n° ANR-15-CE40-0015-01.

© Nguyen Kim Thang;
licensed under Creative Commons License CC-BY

A power management scheduling problem in a realistic setting is an *online* problem [1], meaning that at any time the scheduler is not aware of future events and the decision is irrevocable. The standard performance measure of an algorithm is the *competitive ratio*, defined as the worst ratio between the cost of the algorithm and that of the optimal solution. As having been raised in [1], an important direction is to design efficient algorithms (in term of competitive ratio) for online problems in the speed scaling and power down model (general energy model for short). Attempting efficient algorithms for problems in the general energy model, one encounters several limits of current tools. Hence there are only a few works on the model [2, 7, 13] in contrast to a large literature of previous energy models.

Popular tools in online computation are the charging scheme and the potential function method. The idea of the methods is to show that an algorithm behaves well in an amortized sense. However, such methods yield little insight about the nature of problems. Recently, different approaches based on the duality of mathematical programming to design and analyze online scheduling have been proposed [3, 12, 17]. The approaches reveal the nature of such the problems, hence lead to algorithms which are usually simple and competitive [3, 12, 17, 11, 14, 15, 6, 4]. An essential starting point of those approaches (except [17]) is to formulate a linear or convex relaxation for a given problem. However, problems in the general energy model unlikely admit convex program and that represents the main obstacle to design competitive algorithms. In the paper, we show an approach to bypass this difficulty.

1.1 The Model, Approaches and Contribution

Model. We are given a single machine that can be set either in the *sleep state* or in the *active state*. In the active state, the machine can be either in *working state* in which some job is currently processed or in *idle state* in which no job is currently executed. Each transition of the machine from the sleep state to the active one has cost A, which represents the *wake-up* cost. In the sleep state, the energy consumption of the machine is 0. In the active state, the machine can choose an arbitrary speed $s(t)$ to execute jobs. Hence, if $s(t) > 0$ then the machine is in the working state; otherwise if $s(t) = 0$, the machine is idle.

The power energy consumption of the machine at time t in its active state is $P(s(t)) = s(t)^\alpha + g$ where $\alpha > 2$ and $g = P(0) \geq 0$. The consumed energy (without wake-up cost) of the machine is $\int_0^\infty P(s(t))dt$ where the integral is taken during the machine's active periods. We decompose the latter into *dynamic energy* $\int_0^\infty s(t)^\alpha dt$ and *static energy* $\int_0^\infty g dt$ (where again the integrals are taken during active periods). We call the model as the *general energy model*.

Jobs arrive over time and could be processed preemptively, i.e., a job could be interrupted and resumed later. At any time, the scheduler has to determine the state and the machine speed (if it is active) and also a policy to execute jobs. In the paper, we study the following problems.

In the first problem, each job j has a released time r_j, a deadline d_j, a processing volume p_j. A job j has to be fully processed in $[r_j, d_j]$. The objective is to minimize the total energy consumption (the static, dynamic energy and the wake-up cost).

In the second problem, each job j has released time r_j, weight w_j and requires p_j units of processing volume. The flow-time of a job j is $C_j - r_j$ where C_j is the completion time of the job. The objective is to minimize the total weighted flow-time plus the total energy (including the wake-up cost).

Lagrangian Duality Approach

To overcome the issues in the general energy model, we follow our duality approach presented in [17]. The approach has been applied to non-convex relaxations for several problems in [17]. However, such the problems admit linear relaxations with some lost factors[1] and the consideration of non-convex relaxations permits improvement on the competitive ratio. In this paper, we study the approach for non-convex problems, i.e., the problems unlikely admit a convex relaxation with bounded integrality gap. To the best of our knowledge, it is the first time online algorithms have been designed based on non-convex relaxations.

We first briefly summarize the high level idea of the approach in [17]. Given a problem, formulate a relaxation which is *not* necessarily convex and its Lagrangian dual. Next construct dual variables such that the Lagrangian dual has objective value within a desired factor of the primal one (due to some algorithm). Then by the standard Lagrangian weak duality[2] in mathematical programming, the competitive ratio follows. Since the Lagrangian weak duality also holds in the context of calculus of variations, the approach could be applied for the unknowns which are not only variables but also functions.

Let $L(x, \lambda)$ be the Lagrangian function with primal and dual variables x and λ, respectively. Let $\mathcal{X}$ and $\mathcal{Y}$ are feasible sets of x and λ. Intuitively, the approach could be seen as a game between an algorithm and an adversary. The algorithm chooses dual variables $\lambda^* \in \mathcal{Y}$ in such a way that whatever the choice (strategy) of the adversary, the value $\min_{x\in\mathcal{X}} L(x, \lambda^*)$ is always within a desirable factor c of the objective due to the algorithm. We emphasize that $\min_{x\in\mathcal{X}} L(x, \lambda^*)$ is taken over x feasible solutions of the primal.

An advantage of the approach is the flexibility of the formulation. As convexity is not required, we can study a (non-convex) formulation of a given problem. The main core of the approach is to determine the dual variables and to prove the desired competitive ratio. Determining such dual variables is the crucial step in the analysis. Sometimes, the dual variables have intuitive interpretations that inspire their construction.

It is worthy to note that in the analyses (of both problems), we consider mathematical programs in which the machine state variable remains 0-1 (without being relaxed). An advantage of keeping the variables integer, which is allowed due to the flexibility of the approach, is that we can additionally use charging-scheme-liked arguments. Hence, the analyses are carried out by benefiting properties from both mathematical programming (weak duality) and amortized method (charging scheme).

Our Results

1. For the problem of minimizing the total consumed energy, we formulate a natural *non-convex* formulation using the Dirac delta function. The Dirac function is useful to represent the wake-up cost — an issue that causes the problems in the general energy model to be non-convex. We present an α^α-competitive algorithm and the analysis follows the Lagrangian duality approach described above. In the construction of dual variables, a key step of the analysis, we define these variables in such a way that they subtly capture the marginal increase of the energy consumption. Note that the analysis is tight since our algorithm, in the restricted speed scaling model (without static energy and wake-up cost), turns out to be the OA algorithm, which has competitive ratio exactly α^α [10]. Hence, even the competitive ratio has been only slightly improved from $\max\{4, \alpha^\alpha + 2\}$ [13]

[1] factors polynomial in $1/\epsilon$ in the resource augmentation model in which the machine has speed $1 + \epsilon$.
[2] For completeness, the weak duality is given in the appendix.

to α^α, it suggests that the duality-based approach is seemingly a right tool for online scheduling. Besides, the formulation and the analysis give basics to study the second problem.

2. For the problem of minimizing energy plus weighted flow-time, we derive an $O(\alpha/\ln\alpha)$-competitive algorithm that matches the currently best known competitive ratio (up to a constant factor) for the same problem in the restricted speed scaling model (where the wake-up cost and the static energy cost are 0). The dual variables and the analysis are built upon the salient ideas from the ones in the previous problem but in a more involved manner. Informally, the dual solutions are constructed in order to balance the weighted flow-time cost and the energy cost at any moment in the schedule (the same idea of previous algorithms in the speed scaling model). However, in the general energy model this cannot be guaranteed for every moment in the schedule. Hence, we introduce an additional term to dual variables that covers the moments where the two costs are not balanced. Intuitively, the additional term represents the loss due to the non-convexity of the problem. It turns out that the loss in term of competitive ratio is only a constant factor.
   Due to the space constraint, the analysis is partly given. The full version can be found on the website of the author.

1.2 Related work

Anand et al. [3] proposed studying online scheduling by linear (convex) programming and dual fitting. By this approach, they gave simple algorithms and simple analyses with improved performance for problems where the analyses based on potential functions are complex or it is unclear how to design such functions. Gupta et al. [12] gave a primal-dual algorithm for a class of scheduling problems with cost function $f(z) = z^\alpha$. In [17] we generalized the approach in [3] and proposed to study online scheduling by non-convex programming and the weak Lagrangian duality. Using that technique, [17] derive competitive algorithms for problems related to weighted flow-time. The approaches based on duality become more and more popular. Subsequently, many competitive algorithms have been designed for different problems in online scheduling [3, 12, 17, 11, 14, 15, 6].

For the online problem of minimizing the energy consumption in the model of speed scaling, Bansal et al. [10] gave a $2(\frac{\alpha}{\alpha-1})^\alpha e^\alpha$-competitive algorithm. Later on, Bansal et al. [8] showed that no deterministic algorithm has better competitive ratio than e^α/α. In the general energy model, Irani et al. [16] were the first who studied the problem and they derived an algorithm with competitive ratio $(2^{2\alpha-2}\alpha^\alpha + 2^{\alpha-1} + 2)$. Subsequently, Han et al. [13] presented an algorithm which is $\max\{4, \alpha^\alpha + 2\}$-competitive. In offline setting, the problem is recently showed to be NP-hard [2]. Moreover, they [2] also gave a 1.171-approximation algorithm, which improved the 2-approximation algorithm in [16]. If the instances are agreeable then the problem is polynomial [7]. Recently, Antoniadis et al. [5] have given a FPTAS for the problem.

To the best of our knowledge, the objective of minimizing the total weighted flow-time plus energy has not been studied in the speed scaling with power down energy model. However, this objective has been widely studied in speed scaling energy model. Bansal et al. [9] gave an $O(\alpha/\log\alpha)$-competitive algorithm for weighted flow-time plus energy in a single machine where the energy function is s^α. Based on linear programming and dual-fitting, Anand et al. [3] proved an $O(\alpha^2)$-competitive algorithm for unrelated machines. Subsequently, Nguyen [17] and Devanur and Huang [11] presented an $O(\alpha/\log\alpha)$-competitive algorithms for unrelated machines by dual fitting and primal dual approaches, respectively. It turns out

that the different approaches lead to the same algorithm. To the best of our knowledge, no competitive algorithm is known in the general energy model for this problem.

2 Minimizing Energy in Speed Scaling with Power Down Model

In this section, we study the problem of minimizing the total energy. We formulize the problem as a mathematical program. In such a program, we need to incorporate an information about the machine states and the transition cost from the sleep state to the active one. Here we make use of the properties of the Heaviside step function and the Dirac delta function to encode the machine states and the transition cost. Recall that the Heaviside step function $H(t) = 0$ if $t < 0$ and $H(t) = 1$ if $t \geq 0$. Then $H(t)$ is the integral of the Dirac delta function δ (i.e., $H' = \delta$) and it holds that $\int_{-\infty}^{+\infty} \delta(t)dt = 1$. Now let $F(t)$ be a function indicating whether the machine is in active state at time t, i.e., $F(t) = 1$ if the machine is active at t and $F(t) = 0$ if it is in the sleep state. Assume that the machine initially is in the sleep state. Then $A\int_0^{+\infty} |F'(t)|dt$ equals twice the transition cost of the machine (a transition from the active state to the sleep state costs 0 while by the term $A\int_0^{+\infty} |F'(t)|dt$, it costs A).

Let $s_j(t)$ be variable representing the speed of job j at time t. The problem could be formulated as the following (non-convex) program.

$$\min \quad \int_0^{\infty} P\Big(\sum_j s_j(t)\Big)F(t)dt + \frac{A}{2}\int_0^{+\infty} |F'(t)|dt \tag{1}$$

$$\text{subject to} \quad \int_{r_j}^{d_j} s_j(t)F(t)dt \geq p_j \qquad \forall j$$

$$s_j(t) \geq 0,\ F(t) \in \{0,1\} \qquad \forall j,t$$

Observe that each time a job is executed, the machine has to be in the active state. The first constraint ensures that every job j must be fully processed during $[r_j, d_j]$. Note that we do not relax the variable $F(t)$. The objective function consists of corresponding terms to the energy cost during the active periods and the wake-up cost. Recall that $P(z) = z^{\alpha} + g$.

2.1 Algorithm and Dual Variable Construction

Define the *critical* speed $s^c = \arg\min_{s>0} P(s)/s$. It has been observed in [7] that in any algorithm, it would better to set the machine speed at least s^c whenever it executes some job. Let $0 < \beta \leq 1$ be some constant to be chosen later.

Let $s^*(t)$ and $s_j^*(t)$ be the machine speed and the speed of job j at time t by the algorithm, respectively. In the algorithm, we maintain variables, called *virtual* speeds, $s(t)$ and $s_j(t)$. Intuitively, job j would be processed by speed $s_j(t)$ at time t (and the machine would process jobs by speed $s(t)$) *if* the wake-up cost equals A and the parameter $g = 0$. However, it is not the case so the algorithm will process jobs by different speeds but it is the function related to the virtual speeds.

During the execution of the algorithm, we also maintain a set of *active jobs*. Informally, a job is *active* if it has been released but has not been processed by the algorithm. Initially, set auxiliary variables $s(t)$ and $s_j(t)$ equal 0 for every time t and jobs j. If a job is released then it is marked as *active*.

Let τ be the current moment. Set $s(t) \leftarrow s^*(t)$ for every $t \geq \tau$. Consider currently active jobs in the earliest deadline first (EDF) order. (The set of active jobs may include new released job and jobs that have been released before τ but have not been

processed.) For every active job j and $\tau \le t \le d_j$, increase continuously $s_j(t)$ for all $t \in \arg\min_{t'} P'(s(t'))$ and update simultaneously $s(t) \leftarrow s(t) + s_j(t)$ until $\int_{r_j}^{d_j} s_j(t')dt' = p_j$.

Now consider different states of the machine at the current time τ. We distinguish three different states: (1) in *working state* the machine is active and is executing some jobs; (2) in *idle state* the machine is active but its speed equals 0; and (3) in *sleep state* the machine is inactive.

In working state. If $s(\tau) > 0$ then set the machine speed $s^*(t) \leftarrow \max\{s(t), s^c\}$ for $t \ge \tau$ as long as pending jobs exists. Additionally, mark all currently pending jobs as inactive. Otherwise (if $s(\tau) = 0$), switch the machine to the idle state.

In idle state. If $s(\tau) \ge s^c$ then switch to the working state.

If $s^c > s(\tau) > 0$. Do not execute any job; however, mark all currently pending jobs as active. Intuitively, we delay the execution of such jobs until some moment where the machine has to run at speed s^c in order to complete these jobs (assuming that there is no new job released).

Otherwise, if the total duration of idle state from the last wake-up equals A/g then switch to the sleep state.

In sleep state. If $s(\tau) \ge s^c$ then switch to the working state.

Dual variables. Consider a job j and the virtual machine speed $s(t, r_j)$. If $s(t, r_j) > 0$ for every $t \in [r_j, d_j]$, set λ_j such that $\lambda_j p_j/\beta$ equals the marginal increase of the *dynamic* energy due to the arrival of job j. If $s(t, r_j) = 0$ for some moment $t \in [r_j, d_j]$, define λ_j such that $\lambda_j p_j$ equals the marginal increase of the *dynamic and static* energy due to the arrival of job j (assuming no new job will be released later).

2.2 Analysis

The Lagrangian dual of (1) is $\max_{\lambda \ge 0} \min_{s,F} L(s, F, \lambda)$ where the minimum is taken over (s, F) feasible solutions of the primal and L is the following Lagrangian function

$$\begin{aligned} L(s,F,\lambda) &= \int_0^\infty P\Big(\sum_j s_j(t)\Big)F(t)dt + \frac{A}{2}\int_0^{+\infty} |F'(t)|dt + \sum_j \lambda_j \Big(p_j - \int_{r_j}^{d_j} s_j(t)F(t)dt\Big) \\ &\ge \sum_j \lambda_j p_j - \sum_j \int_{r_j}^{d_j} s_j(t)F(t)\Big(\lambda_j - \frac{P(s(t))}{s(t)}\Big)\mathbb{1}_{\{s(t)>0\}}\mathbb{1}_{\{F(t)=1\}}dt + \frac{A}{2}\int_0^{+\infty}|F'(t)|dt \end{aligned} \tag{2}$$

where $s(t) = \sum_j s_j(t)$.

By weak duality, the optimal value of the primal is always larger than the one of the corresponding Lagrangian dual. In the following, with the chosen values of dual variables, we bound the Lagrangian dual value in function of the algorithm cost and show the competitive ratio.

Let $s^*(t, r_j)$ be the machine speed at time $t \ge r_j$ settled by the algorithm at time r_j. In other words, $s^*(t, r_j)$ would be the machine speed at time t if there is no new released job after job j. Similarly, let $s_j^*(t, r_j)$ be the speed of job j at time t settled by the algorithm at time r_j.

▶ **Lemma 1.** *Let j be an arbitrary job.*

1. *If $s^*(t, r_j) > 0$ for every $t \in [r_j, d_j]$ then $\lambda_j \le \beta P'(s^*(t))$ for every $t \in [r_j, d_j]$.*
2. *Moreover, if $s^*(t, r_j) = 0$ for some $t \in [r_j, d_j]$ then $\lambda_j = P(s^c)/s^c$.*

Proof. We prove the first claim. For any time t, speed $s^*(t)$ is non-decreasing as long as new jobs arrive. Hence, it is sufficient to prove the claim assuming that no other job is released after j, i.e., $\lambda_j \leq \beta P'(s^*(t, r_j))$. The marginal increase in the dynamic energy due to the arrival of j could be written as

$$\frac{1}{\beta}\lambda_j p_j = \int_{r_j}^{d_j} \Big[P(s^*(t,r_j)) - P\big(s^*(t,r_j) - s_j^*(t,r_j)\big)\Big]dt \leq \int_{r_j}^{d_j} P'(s^*(t,r_j))s_j^*(t)dt$$

$$= \min_{r_j \leq t \leq d_j} P'(s^*(t,r_j)) \int_{r_j}^{d_j} s_j^*(t,r_j)dt = \min_{r_j \leq t \leq d_j} P'(s^*(t,r_j)) \cdot p_j$$

where $\min P'(s^*(t,r_j))$ is taken over $t \in [r_j, d_j]$ such that $s_j^*(t,r_j) > 0$. The inequality is due to the convexity of P and the second equality follows the algorithm (that increase the speed of job j at $\arg\min P'(s(t))$ for $r_j \leq t \leq d_j$). So the first claim follows.

We are now showing the second claim. By the algorithm, the fact that $s^*(t, r_j) = 0$ for some $t \in [r_j, d_j]$ means that job j will be processed at speed s^c in some interval $[a, b] \subset [r_j, d_j]$ (assuming that no new job is released after r_j). The marginal increase in the energy is $P(s^c)(b-a)$ while p_j could be expressed as $s^c(b-a)$. Therefore, $\lambda_j = P(s^c)/s^c$. ◀

▶ **Theorem 2.** *The algorithm has competitive ratio at most* $\max\{4, \alpha^\alpha\} = \alpha^\alpha$ *for* $\alpha > 2$.

Proof. Let E_1^* be the dynamic energy of the algorithm schedule. Due to the definition of λ_j's and $0 < \beta \leq 1$ we have $E_1^* = \int_0^\infty [P(s^*(t)) - P(0)]dt \leq \sum_j \lambda_j p_j/\beta$.

Let E_2^* be the static energy plus the wake-up energy of the algorithm, i.e., $E_2^* = \int_0^\infty P(0)F^*(t)dt + \frac{A}{2}\int_0^\infty |(F^*)'(t)|dt$ where $F^*(t)$ is the corresponding state (active or sleep) of the machine at time t by the algorithm. We will lower bound the Lagrangian dual objective by E_1^* and E_2^*.

By Lemma 1 (second statement), for every job j such that $s^*(t, r_j) = 0$ for some $t \in [r_j, d_j]$, $\lambda_j = \frac{P(s^c)}{s^c}$. By the definition of the critical speed, $\lambda_j \leq \frac{P(z)}{z}$ for any $z > 0$. Therefore,

$$\sum_j \int_{r_j}^{d_j} s_j(t)F(t)\left(\lambda_j - \frac{P(s(t))}{s(t)}\right)dt \leq 0 \tag{3}$$

where in the sum is taken over jobs j such that $s^*(t, r_j) = 0$ for some $t \in [r_j, d_j]$.

Define

$$L_1(s,\lambda) := \sum_j \lambda_j p_j - \sum_j \int_{r_j}^{d_j} s_j(t)F(t)\left(\lambda_j - \frac{P(s(t))}{s(t)}\right)\mathbb{1}_{\{s(t)>0\}}\mathbb{1}_{\{F(t)=1\}}dt$$

which is the right-hand side of (2) without the wake-up term.

Let $\bar{s}(t) \in \arg\max_z z\beta P'(s^*(t)) - P(z)$. We have

$$\begin{aligned}
&L_1(s,\lambda)\\
&\geq \beta E_1^* - \max_{s,F} \sum_j \int_{r_j}^{d_j} s_j(t)F(t)\left[\beta P'(s^*(t)) - \frac{P(s(t))}{s(t)}\right]\mathbb{1}_{\{s(t)>0\}}\mathbb{1}_{\{F(t)=1\}}\mathbb{1}_{\{s^*(t)>0\}}dt\\
&\geq \beta E_1^* - \max_{s} \int_0^\infty s(t)\left[\beta P'(s^*(t)) - \frac{P(s(t))}{s(t)}\right]\mathbb{1}_{\{s(t)>0\}}\mathbb{1}_{\{F(t)=1\}}\mathbb{1}_{\{s^*(t)>0\}}dt\\
&\geq \beta E_1^* - \int_0^\infty \Big[\beta P'(s^*(t))\bar{s}(t) - P(\bar{s}(t))\Big]\mathbb{1}_{\{s(t)>0\}}\mathbb{1}_{\{F(t)=1\}}\mathbb{1}_{\{s^*(t)>0\}}dt\\
&\geq \beta E_1^* - \frac{1}{2}\int_0^\infty \Big[\beta P'(s^*(t))\bar{s}(t) - P(\bar{s}(t))\Big]\mathbb{1}_{\{s^*(t)>0\}}dt\\
&\qquad - \frac{1}{2}\int_0^\infty \Big[\beta P'(s^*(t))\bar{s}(t) - P(\bar{s}(t))\Big]\mathbb{1}_{\{F(t)=1\}}dt
\end{aligned}$$

where in the first line, the sum is taken over jobs j such that $s^*(t, r_j) > 0$ for all $t \in [r_j, d_j]$. Note that if $s^*(t, r_j) > 0$ then $s^*(t) \geq s^*(t, r_j) > 0$. The first inequality follows (3) and Lemma 1 (first statement). The second inequality holds since $F(t) \in \{0, 1\}$ and $s(t) \geq \sum_j s_j(t)$ where again the sum is taken over jobs j such that $s^*(t, r_j) > 0$ for all $t \in [r_j, d_j]$. The third inequality is due to the first order derivative and $\bar{s}(t)$ maximizes function $z\beta P'(s^*(t)) - P(z)$ (so $\bar{s}(t)$ is the solution of equation $P'(z(t)) = \beta P'(s^*(t))$).

As the energy power function $P(z) = z^\alpha + g$ where $\alpha > 2$ and $g \geq 0$, $\bar{s}(t)^{\alpha-1} = \beta(s^*(t))^{\alpha-1}$. Therefore,

$$\begin{aligned}
L_1(s, \lambda) &\geq \beta E_1^* - \frac{1}{2}\int_0^\infty \Big(\beta\alpha(s^*(t))^{\alpha-1}\bar{s}(t) - (\bar{s}(t))^\alpha - g\Big)\mathbb{1}_{\{s^*(t)>0\}}dt \\
&\qquad - \frac{1}{2}\int_0^\infty \Big(\beta\alpha(s^*(t))^{\alpha-1}\bar{s}(t) - (\bar{s}(t))^\alpha - g\Big)\mathbb{1}_{\{F(t)=1\}}dt \\
&= \beta E_1^* - \int_0^\infty (\alpha-1)\beta^{\alpha/(\alpha-1)}(s^*(t))^\alpha dt + \frac{1}{2}\int_0^\infty g\mathbb{1}_{\{s^*(t)>0\}}dt + \frac{1}{2}\int_0^\infty g\mathbb{1}_{\{F(t)=1\}}dt \\
&= \Big[\beta - (\alpha-1)\beta^{\alpha/(\alpha-1)}\Big]E_1^* + \frac{1}{2}\int_0^\infty g\mathbb{1}_{\{s^*(t)>0\}}dt + \frac{1}{2}\int_0^\infty g\mathbb{1}_{\{F(t)=1\}}dt
\end{aligned}$$

Choose $\beta = 1/\alpha^{\alpha-1}$, we have that

$$L(s, F, \lambda) \geq \frac{1}{\alpha^\alpha}E_1^* + \frac{1}{2}\int_0^\infty g\mathbb{1}_{\{s^*(t)>0\}}dt + \frac{1}{2}\int_0^\infty g\mathbb{1}_{\{F(t)=1\}}dt + \frac{A}{2}\int_0^\infty |F'(t)|dt$$

In order to prove the theorem, we prove the following claim.

▶ **Claim 3.** *Define*

$$L_2(F) := \frac{1}{2}\int_0^\infty g\mathbb{1}_{\{s^*(t)>0\}}dt + \frac{1}{2}\int_0^\infty g\mathbb{1}_{\{F(t)=1\}}dt + \frac{A}{2}\int_0^\infty |F'(t)|dt$$

Then, $L_2(F) \geq E_2^*/4$ for any feasible solution (s, F) of the relaxation.

We first show how to deduce the theorem assuming the claim. By the claim, the dual

$$L(s, F, \lambda) \geq E_1^*/\alpha^\alpha + L_2(F) \geq E_1^*/\alpha^\alpha + E_2^*/4$$

whereas the primal is $E_1^* + E_2^*$. Thus, the competitive ratio is at most $\max\{4, \alpha^\alpha\}$. In the remaining, we prove the claim by amortized arguments.

Proof of Claim. Consider the algorithm schedule. An *end-time* u is a moment in the schedule such that the machine switches from the idle state to the sleep state. Conventionally, the first end-time in the schedule is 0. Partition the time line into phases. A *phase* $[u, v)$ is a time interval such that u, v are two consecutive end-times. Observe that in a phase, the schedule has transition cost A and there is always a new job released in a phase (otherwise the machines would not switch to non-sleep state). We will prove the claim on every phase. In the following, we are interested in phase $[u, v)$ and whenever we mention $L_2(F)$, it refers to $\frac{1}{2}\int_u^v g\mathbb{1}_{\{s^*(t)>0\}}dt + \frac{1}{2}\int_u^v g\mathbb{1}_{\{F(t)=1\}}dt + \frac{A}{2}\int_u^v |F'(t)|dt$.

By the algorithm, the static energy of the schedule during the idle time is A, i.e., $\int_u^v g\mathbb{1}_{\{s^*(t)=0\}}dt = A$. Let (s, F) be an arbitrary feasible primal solution.

If during $[u, v)$, the machine according to the solution (s, F) makes a transition from sleep state to non-sleep state (i.e., $F(t') = 0$ and $F(t'') = 1$ for some $u \leq t' < t'' < v$) or

inversely then

$$\begin{aligned} L_2(F) &\geq \frac{1}{2}\int_u^v g\mathbb{1}_{\{s^*(t)>0\}}dt + \frac{A}{2} \\ &\geq \frac{1}{2}\int_u^v g\mathbb{1}_{\{s^*(t)>0\}}dt + \frac{1}{4}\left(\int_u^v g\mathbb{1}_{\{s^*(t)=0\}}dt + A\right) \geq \frac{1}{4}E_2^*\big|_{[u,v)}. \end{aligned}$$

If during $[u, v)$, the machine following solution (s, F) makes no transition (from non-sleep static to sleep state or inversely) then $F(t) = 1$ during $[u, v)$ in order to process jobs released in the phase. Therefore,

$$\begin{aligned} L_2(F) &\geq \frac{1}{2}\int_u^v g\mathbb{1}_{\{s^*(t)>0\}}dt + \frac{1}{2}\int_u^v g\mathbb{1}_{\{F(t)=1\}}dt = \frac{1}{2}\int_u^v g\mathbb{1}_{\{s^*(t)>0\}}dt + \frac{1}{2}\int_u^v g dt \\ &\geq \frac{1}{2}\int_u^v g\mathbb{1}_{\{s^*(t)>0\}}dt + \frac{1}{4}\int_u^v g\mathbb{1}_{\{s^*(t)=0\}}dt + \frac{A}{4} \\ &\geq \frac{1}{4}\left(\int_u^v g\mathbb{1}_{\{s^*(t)>0\}}dt + \int_u^v g\mathbb{1}_{\{s^*(t)=0\}}dt + A\right) = \frac{1}{4}E_2^*\big|_{[u,v)} \end{aligned}$$

where the second inequality follows the algorithm: as the machine switches to sleep state at time v, it means that the total idle duration in $[u, v)$ incurs a cost A. ◀

The proof of the claim completes the theorem proof. ◀

3 Minimizing Energy plus Weighted Flow-Time in Speed Scaling with Power Down Model

In this section, we study the problem of minimizing total weighted flow-time plus energy. Let $F(t)$ be a function indicating whether the machine i is in active state at time t, i.e., $F(t) = 1$ if the machine is active at t and $F(t) = 0$ if it is in the sleep state. Let $s_j(t)$ be the variable that represents the speed of job j at time t. Let C_j be a variable representing the completion time of j. Similar as the previous section, the problem can be formulized as the following (non-convex) program.

$$\begin{aligned} \min \int_0^\infty 2P\left(\sum_j s_j(t)\right)F(t)dt + 2\sum_j\left(\int_{r_j}^{C_j} s_j(t)F(t)dt\right)\frac{w_j}{p_j}(C_j - r_j) \\ + A\int_0^\infty |F'(t)|dt \qquad (4) \end{aligned}$$

$$\begin{aligned} \text{subject to} \quad & \int_{r_j}^{C_j} s_j(t)F(t)dt = p_j \quad \forall j \\ & s_j(t) \geq 0, \quad F(t) \in \{0, 1\} \quad \forall j, t \end{aligned}$$

The first constraint ensures that every job j must be completed by some moment C_j which is its completion time. In the objective function, the first and second terms represent twice the consumed energy and the total weighted flow-time, respectively. Note that in the second term, $\int_{r_j}^{C_j} s_j(t)F(t)dt = p_j$ by the constraints. The last term stands for twice the transition cost.

Notations. We say that a job j is *pending* at time t if it has not been completed, i.e., $r_j \leq t < C_j$. At time t, denote $q_j(t)$ the *remaining* processing volume of job j. The *total weight* of pending jobs at time t is denoted as $W(t)$. The *density* of a job j is $\delta_j = w_j/p_j$. Recall that the *critical speed* $s^c \in \arg\min_{z\geq 0} P(z)/z$. As $P(z) = z^\alpha + g$, by the first order condition, s^c satisfies $(\alpha - 1)(s^c)^\alpha = g$.

3.1 The Algorithm

We first describe intuitively the ideas of the algorithm. In the speed scaling model, all previous algorithms explicitly or implicitly balance the weighted flow-time of jobs and the consumed energy to process such jobs. That could be done by setting the machine speed at any time t proportional to some function of the total weight of pending jobs (precisely, proportional to $W(t)^{1/\alpha}$ where $W(t)$ is the total weight of pending jobs). Our algorithm follows the idea of balancing the weighted flow-time and the energy. However, in the general energy model, the algorithm would not be competitive if the speed is always set proportionally to $W(t)^{1/\alpha}$ since the static energy might be large due to the long active periods of the machine. Hence, even if the total weight of pending jobs on the machine is small, in some situation the speed is maintained larger than $W(t)^{1/\alpha}$. In fact, it will be set to be the critical speed s^c.

An issue while dealing with the general model is to determine the state of the machine at a given time (active or inactive). If the total weight of pending jobs is small and the machine is active for a long time, then the static energy is large. Otherwise if pending jobs remain for long time then the weighted flow-time is large. The algorithm, together with dual variables, are constructed in order to bypass this difficulty.

Description of algorithm

At any time t, we distinguish different states of the machine: the *working state* (the machine is active and currently processes some job), the *idle state* (the machine is active but currently processes no job) and the *sleep state*. At time t, we (re)compute the total weight of pending jobs and consider different scenarios corresponding to the current machine state.

In working state. If $\alpha W(t)^{(\alpha-1)/\alpha} > P(s^c)/s^c$ then the machine speed is set as $W(t)^{1/\alpha}$. Otherwise, the speed is set as s^c. At any time, the machine processes the highest density job among the pending ones.

In idle state. **1.** If $\alpha W(t)^{(\alpha-1)/\alpha} > P(s^c)/s^c$ then switch to the working state.

2. If $0 < \alpha W(t)^{\frac{\alpha-1}{\alpha}} \leq P(s^c)/s^c$ then make a plan to process pending jobs with speed (exactly) s^c in non-increasing order of their density. The plan consists of a single block (with no idle time) and the block length could be explicitly computed (given the processing volumes of all jobs and speed s^c). Hence, the total consumed energy in the plan can be computed and it is independent of the starting time of the plan.

Choose the starting time of the plan in such a way that the total energy consumption in the plan equals the total weighted flow-time of all jobs in the plan. There always exists such starting time since if the plan begins immediately at the current time, the energy is larger than the weighted flow-time; and inversely if the starting time is large enough, the weighted flow-time dominates the energy consumption.

At the starting time of a plan, switch to the working state. (Note that the plan together with its starting time could be changed due to the arrival of new jobs.)

3. Otherwise, if the total duration of idle state from the last wake-up equals A/g then switch to sleep state.

In sleep state. Use the same policy as the first two steps of the idle state.

3.2 Analysis

The Lagrangian dual of program (4) is $\max \min_{s,C,F} L$ where L is the corresponding Lagrangian function and the maximum is taken over dual variables. We need to choose appropriate dual variables and prove that for any feasible solution (s, C, F) of the primal, the Lagragian dual is bounded by a desired factor from the primal.

Dual variables

Denote the dual variables corresponding to the first constraints of (4) as λ_j's. Set all dual variables (corresponding to the primal (4)) except λ_j's equal 0. The values of dual variables λ_j's is defined as follows.

Fix a job j. At the arrival of a job j, rename pending jobs as $\{1, \ldots, k\}$ in non-increasing order of their densities, i.e., $p_1/w_1 \leq \ldots \leq p_k/w_k$ (note that p_a/w_a is the inverse of job a's density). Denote $W_a = w_a + \ldots + w_k$ for $1 \leq a \leq k$.

Define λ_j such that

$$\lambda_j p_j = w_j \sum_{a=1}^{j} \frac{q_a(r_j)}{W_a^{1/\alpha}} + W_{j+1} \frac{q_j(r_j)}{W_j^{1/\alpha}} + P(s^c) \frac{q_j(r_j)}{s^c} \tag{5}$$

Note that $q_j(r_j) = p_j$. If job j is processed with speed larger than s^c then the first term stands for the weighted flow-time of j and the second term represents an upper bound on the increase of the weighted flow-time of jobs with density smaller than δ_j due to the arrival of j. Observe that as j arrives, the jobs with higher density than δ_j are completed earlier and the ones with smaller density than δ_j may have higher flow-time. Here, the second term in (5) captures the marginal change in the total weighted flow-time. The third term in (5) is introduced in order to cover energy consumption during the execution periods of job j if it is processed by speed s^c. That term is necessary since during such periods the energy consumption and the weighted flow-time are not balanced.

The Lagrangian function $L(s, C, F, \lambda)$ with the chosen dual variables becomes

$$\begin{aligned}
&A \int_0^\infty |F'(t)|dt + 2 \int_0^\infty P\left(\sum_j s_j(t)\right) F(t)dt + 2 \sum_j \delta_j (C_j - r_j) \int_{r_j}^{C_j} s_j(t) F(t) dt \\
&\qquad + \sum_j \lambda_j \left(p_j - \int_{r_j}^{C_j} s_j(t) F(t) dt\right) \\
&= \sum_j \lambda_j p_j + A \int_0^\infty |F'(t)|dt + \sum_j \int_{r_j}^{C_j} \delta_j (C_j - r_j) s_j(t) F(t) dt \\
&\qquad - \sum_j \int_{r_j}^{C_j} s_j(t) F(t) \left(\lambda_j - 2\frac{P(s(t))}{s(t)} - \delta_j (C_j - r_j)\right) dt
\end{aligned}$$

Notations

We denote $s^*(t)$ the machine speed at time t by the algorithm. So by the algorithm, if $s^*(t) > 0$ then $s^*(t) \geq s^c$. Let $\mathcal{E}_1^*$ and $\mathcal{E}_2^*$ be the total dynamic and static energy consumed by the algorithm schedule, respectively. In other words, $\mathcal{E}_1^* = \int_0^\infty (s^*(t))^\alpha dt$ and $\mathcal{E}_2^* = \int_0^\infty g$ where the integral is taken over all moments t where the machine is active (either in working or in idle states). Additionally, let $\mathcal{E}_3^*$ be the total transition cost of the machine. Moreover, let $\mathcal{F}^*$ be the total weighted flow-time due to the algorithm.

We relate the energy cost of the algorithm schedule and the chosen values of dual variables by the following lemma. Note that by definition of λ_j's, we have that $\sum_j \lambda_j p_j \geq \mathcal{F}^*$.

▶ **Lemma 4.** *It holds that* $2\mathcal{E}_1^* + 2\mathcal{E}_2^* \geq \mathcal{F}^*$ *and* $\sum_j \lambda_j p_j \geq \mathcal{E}_1^*$. *Consequently,* $\sum_j \lambda_j p_j \geq \frac{7}{8}\mathcal{E}_1^* + \frac{1}{16}\mathcal{F}^* - \frac{1}{8}\mathcal{E}_2^*$.

The following lemma is crucial in the analysis.

▶ **Lemma 5.** *Let j be an arbitrary job. Then, for every $t \geq r_j$*

$$\lambda_j - \delta_j(t - r_j) \leq \max\left\{\frac{\alpha}{\alpha-1}W(t)^{\frac{\alpha-1}{\alpha}} + \frac{P(s^c)}{s^c}, 2\frac{P(s^c)}{s^c}\right\}$$

▶ **Theorem 6.** *The algorithm is $O(\alpha/\ln\alpha)$-competitive.*

Proof. Recall that the dual has value at least $\min L(s, C, F, \lambda)$ where the minimum is taken over (s, C, F) feasible solution of the primal. The goal is to lower bound the Lagrangian function.

$$L(s,C,F,\lambda) = \sum_j \lambda_j p_j + A\int_0^\infty |F'(t)|dt + \sum_j \int_{r_j}^{C_j} \delta_j(C_j - r_j)s_j(t)F(t)dt$$
$$- \sum_{i,j}\int_{r_j}^{C_j} s_j(t)F(t)\left(\lambda_j - 2\frac{P(s(t))}{s(t)} - \delta_j(C_j - r_j)\right)\mathbb{1}_{\{s(t)>0\}}dt \tag{6}$$

for any feasible primal solution (s, C, F).

Fix a feasible primal solution (s, C, F). Define $L_1(s, C, F, \lambda)$ as

$$\sum_j \int_{r_j}^{C_j} s_j(t)F(t)\left(\lambda_j - 2\frac{P(s(t))}{s(t)} - \delta_j(C_j - r_j)\right)\mathbb{1}_{\{s(t)>0\}}dt$$

▶ **Claim 7.** *Let (s, C, F) be an arbitrary feasible solution of the primal. Then,*

$$L_1(x,s,C,F,\lambda) \leq \frac{1}{(\alpha-1)^{\frac{1}{\alpha-1}}}\mathcal{F}^* - \frac{1}{2}\int_0^\infty g\mathbb{1}_{\{F(t)>0\}}dt - \frac{1}{2}\int_0^\infty g\mathbb{1}_{\{s^*(t)>0\}}dt$$

▶ **Claim 8.** *Let (s, C, F) be an arbitrary feasible solution of the primal. Define*

$$L_2(F) := \sum_j \int_{r_j}^{C_j} \delta_j(C_j - r_j)s_j(t)F(t)dt + A\int_0^\infty |F'(t)|dt$$
$$+ \frac{1}{2}\int_0^\infty g\mathbb{1}_{\{F(t)>0\}}dt + \frac{1}{2}\int_0^\infty g\mathbb{1}_{\{s^*(t)>0\}}dt$$

Then, $L_2(F) \geq (\mathcal{E}_2^* + \mathcal{E}_3^*)/4$.

We first show how to prove the theorem assuming the claims. By (6), we have

$$\begin{aligned}
L(s,C,F,\lambda) &\geq \sum_j \lambda_j p_j + A\int_0^\infty |F'(t)|dt + \sum_j \int_{r_j}^{C_j} \delta_j(C_j - r_j)s_j(t)F(t)dt \\
&\quad - \sum \frac{1}{(\alpha-1)^{1/(\alpha-1)}}\mathcal{F}^* + \frac{1}{2}\int_0^\infty g\mathbb{1}_{\{F(t)>0\}}dt + \frac{1}{2}\int_0^\infty g\mathbb{1}_{\{s^*(t)>0\}}dt \\
&\geq \sum_j \lambda_j p_j - \frac{1}{(\alpha-1)^{1/(\alpha-1)}}\mathcal{F}^* + \frac{1}{4}\mathcal{E}_2^* + \frac{1}{4}\mathcal{E}_3^* \\
&\geq \left(1 - \frac{1}{(\alpha-1)^{1/(\alpha-1)}}\right)\left(\frac{7}{8}\mathcal{E}_1^* + \frac{1}{16}\mathcal{F}^* - \frac{1}{8}\mathcal{E}_2^*\right) + \frac{1}{4}\mathcal{E}_2^* + \frac{1}{4}\mathcal{E}_3^* \\
&\geq \left(1 - \frac{1}{(\alpha-1)^{1/(\alpha-1)}}\right)\left(\frac{7}{8}\mathcal{E}_1^* + \frac{1}{16}\mathcal{F}^*\right) + \frac{1}{8}\mathcal{E}_2^* + \frac{1}{4}\mathcal{E}_3^* \\
&\geq \frac{\ln(\alpha-1)}{2(\alpha-1)}\left(\frac{7}{8}\mathcal{E}_1^* + \frac{1}{16}\mathcal{F}^*\right) + \frac{1}{8}\mathcal{E}_2^* + \frac{1}{4}\mathcal{E}_3^*
\end{aligned}$$

where the first and second inequalities are due to Claim 7 and Claim 8, respectively. The third inequality follows Lemma 4 and $\sum_j \lambda_j p_j \geq \mathcal{F}^*$. The last inequality is due to the fact that $2 \geq (\alpha-1)^{\frac{1}{\alpha-1}} \geq 1 + \frac{\ln(\alpha-1)}{\alpha-1}$ for every $\alpha > 2$.

Besides, the primal objective is at most $2(\mathcal{F}^* + \mathcal{E}_1^* + \mathcal{E}_2^* + \mathcal{E}_3^*)$. Hence, the competitive ratio is $O(\alpha/\ln\alpha)$. ◀

▶ **Theorem 9.** *The algorithm is $O(\alpha/\ln\alpha)$-competitive.*

4 Conclusion

In this paper, we have shown that the Lagrangian duality approach is appropriate to study certain problems which unlikely admit a convex formulation. For many optimization problems, it is challenging to come up with a strong formulation in which the integral constraint of variables is relaxed and the integrality gap is relatively small. The Lagrangian duality approach gives the flexibility to study directly certain problems without relaxing the integrality and without linear/convex formulation. As mentioned earlier and having observed in the analyses, by the approach, one can benefit from techniques in mathematical programming and amortized analysis. It would be interesting to see more work in this direction. For concrete questions, the problems studied in the paper are open for unrelated machine environment. One would expect the existence of algorithms with similar competitive ratio (up to a constant factor).

Acknowledgement. We thank anonymous reviewers for their useful feedbacks that improve the presentation of the paper.

References

1 Susanne Albers. Energy-efficient algorithms. *Commun. ACM*, 53(5):86–96, 2010.
2 Susanne Albers and Antonios Antoniadis. Race to idle: new algorithms for speed scaling with a sleep state. *ACM Transactions on Algorithms (TALG)*, 10(2):9, 2014.
3 S. Anand, Naveen Garg, and Amit Kumar. Resource augmentation for weighted flow-time explained by dual fitting. In *Proc. 23rd ACM-SIAM Symposium on Discrete Algorithms*, pages 1228–1241, 2012.

4 Spyros Angelopoulos, Giorgio Lucarelli, and Kim Thang Nguyen. Primal-dual and dual-fitting analysis of online scheduling algorithms for generalized flow time problems. In *Proc. 23rd European Symposium of Algorithms (ESA)*, pages 35–46. Springer, 2015.

5 Antonios Antoniadis, Chien-Chung Huang, and Sebastian Ott. A fully polynomial-time approximation scheme for speed scaling with sleep state. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1102–1113. SIAM, 2015.

6 Yossi Azar, Nikhil R. Devanur, Zhiyi Huang, and Debmalya Panigrahi. Speed scaling in the non-clairvoyant model. In *Proc. 27th ACM on Symposium on Parallelism in Algorithms and Architectures*, pages 133–142, 2015.

7 Evripidis Bampis, Christoph Dürr, Fadi Kacem, and Ioannis Milis. Speed scaling with power down scheduling for agreeable deadlines. *Sustainable Computing: Informatics and Systems*, 2(4):184–189, 2012.

8 Nikhil Bansal, Ho-Leung Chan, Dmitriy Katz, and Kirk Pruhs. Improved bounds for speed scaling in devices obeying the cube-root rule. *Theory of Computing*, 8(1):209–229, 2012.

9 Nikhil Bansal, Ho-Leung Chan, and Kirk Pruhs. Speed scaling with an arbitrary power function. In *Proc. 20th ACM-SIAM Symposium on Discrete Algorithms*, pages 693–701, 2009.

10 Nikhil Bansal, Tracy Kimbrel, and Kirk Pruhs. Speed scaling to manage energy and temperature. *J. ACM*, 54(1), 2007.

11 Nikhil R. Devanur and Zhiyi Huang. Primal dual gives almost optimal energy efficient online algorithms. In *Proc. 25th ACM-SIAM Symposium on Discrete Algorithms*, 2014.

12 Anupam Gupta, Ravishankar Krishnaswamy, and Kirk Pruhs. Online primal-dual for non-linear optimization with applications to speed scaling. In *Proc. 10th Workshop on Approximation and Online Algorithms*, pages 173–186, 2012.

13 Xin Han, Tak Wah Lam, Lap-Kei Lee, Isaac Kar-Keung To, and Prudence W. H. Wong. Deadline scheduling and power management for speed bounded processors. *Theor. Comput. Sci.*, 411(40-42):3587–3600, 2010.

14 Sungjin Im, Janardhan Kulkarni, and Kamesh Munagala. Competitive algorithms from competitive equilibria: Non-clairvoyant scheduling under polyhedral constraints. In *STOC*, 2014.

15 Sungjin Im, Janardhan Kulkarni, Kamesh Munagala, and Kirk Pruhs. Selfishmigrate: A scalable algorithm for non-clairvoyantly scheduling heterogeneous processors. In *Proc. 55th IEEE Symposium on Foundations of Computer Science*, 2014.

16 Sandy Irani, Sandeep K. Shukla, and Rajesh Gupta. Algorithms for power savings. *ACM Transactions on Algorithms*, 3(4), 2007.

17 Nguyen Kim Thang. Lagrangian duality in online scheduling with resource augmentation and speed scaling. In *Proc. 21st European Symposium on Algorithms*, pages 755–766, 2013.