Lagrangian Primal Dual Algorithms in Online Scheduling

Nguyen Kim Thang*

IBISC, University of Evry Val d’Essonne, France

Abstract

We present a primal-dual approach to design algorithms in online scheduling. Our approach makes use of the Lagrangian weak duality and convexity to derive dual programs for problems which could be formulated as convex assignment problems. The constraints of the duals explicitly indicate the online decisions and naturally lead to competitive algorithms.

We illustrate the advantages and the flexibility of the approach through problems in different setting: from single machine to unrelated machine environments, from typical competitive analysis to the one with resource augmentation, from convex relaxations to non-convex relaxations.

1 Introduction

In the online setting, items arrive over time and one must determine how to serve items in order to optimize a quality of service without the knowledge about future. A popular measure for studying the performance of online algorithms is competitive ratio in the model of the worst-case analysis. An algorithm is said to be c-competitive if for any instance its objective value is within factor c of the optimal offline algorithm’s one. Moreover, to remedy the limitation of pathological instances in the worst-case analysis, there is other model called resource augmentation [11]. In the latter, online algorithms are given an extra power and are compared to the optimal offline algorithm without that additional resource. This model has successfully provided theoretical evidence for heuristics with good performance in practice, especially in online scheduling where jobs arrive online and need to be processed on machines. We say a scheduling algorithm is s-speed c-competitive if for any input instance the objective value of the algorithm with machines of speed s is at most c times the objective value of the optimal offline scheduler with unit speed machines.

The most successful tool until now to analyze online algorithms is the potential function method. Potential functions have been designed to show that the corresponding algorithms behave well in an amortized sense. However, designing such potential functions is far from trivial and often yields little insight about how to design such potential functions and algorithms for related problems.

Recently, interesting approaches [1, 8, 14] based on mathematical programming have been presented in the search for a principled tool to design and analyze online scheduling algorithms. The approaches give insight about the nature of many scheduling problems, hence lead to algorithms which are usually simple and competitive.

*thang@ibisc.fr
1.1 Contribution and techniques

In this paper, we present a primal-dual approach to design algorithms for an online convex assignment and applications to online scheduling. The online convex assignment consists of a set of agents and a set of items which arrive over time. At the arrival of item $j$, the item needs to be (fractionally) assigned to some agents $i$. Let $x_{ij}$ be the amount of $j$ assigned to $i$. The problem is to minimize $\sum_i f_i(\sum_j a_{ij} x_{ij})$ under the constraints $g_i(\sum_j b_{ij} x_{ij}) \leq 0$ and $h_j(\sum_i c_{ij} x_{ij}) \leq 0$ for every $i,j$ where functions $f_i, g_i, h_j$'s are convex. In offline setting, the optimal solutions are completely characterized by the KKT conditions (see [3] for example). However, for online setting, the conditions could not be satisfied due to the lack of knowledge on the inputs.

Our approach is the following. We first consider the problem as a primal convex mathematical program. Then we derive a Lagrangian dual program by the standard Lagrangian duality. Instead of analyzing directly the corresponding Lagrangian functions where in general one cannot disentangle the objective and the constraints as well as the primal and dual variables, we exploit the convexity property of given functions and construct a dual program. In the latter, dual variables are separated from the primal ones. The construction is shown in Section 2. As the price of the separation procedure, the strong duality property is not guaranteed. However, the weak duality always holds and that is crucial (and enough) to deduce a lower bound for the given problem. An advantage of the approach lies in the dual program in which the constraints could be maintained online. Moreover, the dual constraints explicitly indicate the online decisions and naturally lead to a competitive algorithm in the primal-dual sense.

We illustrate the advantages and the flexibility of the approach through problems in online scheduling. The problems that we study vary in different settings: from single machine to unrelated machine environments, from typical competitive analysis to the one with resource augmentation. Moreover, we show how to combine the primal-dual approach in this paper and the dual-fitting approach presented in [14] to design and analyze algorithms for problem with non-convex relaxation.

1. We consider the problem in which there is a machine and jobs arrive over time. Each job $j$ is released at time $r_j$, has deadline $d_j$, processing volume $p_j$ and a value $a_j$. Jobs could be executed preemptively and at a time $t$, the scheduler has to choose a set of jobs and the machine speed $s(t)$ in order to process such jobs. The energy cost of a schedule is $\int_0^\infty P(s(t)) dt$ where $P$ is a given convex energy power. Typically, $P(z) = z^\alpha$ for some constant $\alpha \geq 1$. The objective of the problem is to minimize energy cost plus the lost value — which is the total value of uncompleted jobs. Using the primal-dual approach, we derive an algorithm where the competitive ratio is characterized by a system of differential equations. For typical power function $P(z) = z^\alpha$, the competitive ratio turns out to be $\alpha^\alpha$ (and rediscover the result in [12]. With the primal-dual framework, the result is more general and the analysis is simpler.

2. We consider other tradeoff between value and energy. Similar to the previous problem, jobs arrive online, each one has a deadline, a value and could be processed preemptively. However, there are a set of unrelated machines and a job may have different processing volumes on different machines. At a time $t$, the scheduler has to choose a set of jobs to be executed on each machine, and the speeds $s_i(t)$’s for every machine $i$ to process such jobs. Job migration is not allowed, i.e., no job could be executed on more than one machine. The objective now is to maximize the total value of completed jobs minus the energy cost — which is $\sum_i \int_0^\infty P(s_i(t)) dt$ where $P$ is a given convex power function. It has been shown that without resource augmentation no algorithm has bounded competitive ratio even for a single machine.
We study the problem in the resource augmentation model. We give a primal-dual algorithm which is $(1 + \epsilon)$-speed and $1/\epsilon$-competitive for every $\epsilon \geq \epsilon(P) > 0$ where $\epsilon(P)$ depends on function $P$. For typical function $P(z) = z^\alpha$, $\epsilon(P) = 1 - \alpha^{-1/\alpha}$ which is closed to 0 for $\alpha$ large.

3. We study the problem of speed scaling with power down scheduling on a single machine. Again, jobs are released online, each job has a deadline, a processing volume and could be processed preemptively. In the problem, all jobs have to be executed. The machine has could be transitioned into a sleep state or in the active state. In the sleep state, the energy consumption is 0. In the active state, the power energy consumption at time $t$ is $P(s(t)) = s(t)^\alpha + g$ where $\alpha \geq 1$ and $g \geq 0$ are constant. A transition cost from the sleep state to the active state is $A$, which represents the wake-up cost. The objective is to minimize the total energy — the consumed energy in active state plus the wake-up energy — while executing all jobs. For the problem, it is unclear how to formalize a relaxation as a convex program. Therefore, we consider a natural relaxation which is not convex. We first study a special case with no wake-up cost. In this case, the relaxation becomes convex and our framework could be applied to show a $\alpha^\alpha$-competitive algorithm (the algorithm is in fact algorithm OPTIMAL AVAILABLE [15]). Next we study the problem in general (with wake-up cost). The special case effectively gives idea to determine the machine speed in active state. Thus we consider an algorithm which uses that procedure to maintain the machine speed in active state as a subroutine. The algorithm turns out to be algorithm SLEEP-AWARE OPTIMAL AVAILABLE (SOA) [9] with different description (due to the primal-dual view). Han et al. [9] proved that SOA has competitive ratio $\max\{4, \alpha^\alpha + 2\}$. We prove that SOA is indeed $\max\{4, \alpha^\alpha\}$-competitive by the dual-fitting technique presented in [14]. Although the improvement is slight, the analysis is tight$^1$ and it suggests that the duality-based approach is seemingly a right tool for online scheduling. Through the problem, we illustrate the design and analysis of an algorithm for non-convex relaxation. The primal-dual framework (presented in this paper) gives ideas for the construction of an algorithm while the analysis is done using dual-fitting technique.

1.2 Related work

Anand et al. [1] was the first who 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. Subsequently, Nguyen [14] generalized the approach in [1] and proposed to study online scheduling by non-convex programming and the weak Lagrangian duality. Using that technique, [14] designed and analyzed non-water-filling algorithms for problems related to weighted flow-time.

Buchbinder and Naor [4] presented the primal-dual method for online packing and covering problems. Their method unifies several previous potential function based analysis and is a powerful tool to design and analyze algorithms for problems with linear relaxations. Gupta et al. [8] gave a primal-dual algorithm for a general class of scheduling problems with cost function $f(z) = z^\alpha$. Devanur and Jain [7] also used the primal-dual approach to derive optimal competitive ratios for online matching with concave return. Later on, Devanur and Huang [6] considered the problem of

$^1$The algorithm has competitive ratio exactly $\alpha^\alpha$ even without wake-up cost [2].
online scheduling to minimize the sum of energy and weighted flow-time on unrelated machines. They gave an algorithm with almost optimal competitive ratio for arbitrary convex power function by primal-dual approach. The construction of dual programs in [6, 7] is based on convex conjugates and Fenchel duality for primal convex programs in which the objective is convex and the constraints are linear.

An interesting quality of service in online scheduling is the tradeoff between energy and throughput. The online problem to minimize the consumed energy plus lost values with the energy power $P(z) = z^\alpha$ is first studied by [5] where a $(\alpha^\alpha + 2\epsilon\alpha)$-competitive algorithm is given for a single machine. Subsequently, Kling and Pietrzyk [12] derived an improved $\alpha^\alpha$-competitive for identical machines with migration using the technique in [8]. The online problem to maximize the total value of completed jobs minus the consumed energy for a single machine has been considered in [13]. Pruhs and Stein [13] proved that the competitive ratio without resource augmentation is unbounded and gave an $(1 + \epsilon)$-speed, $O(1/\epsilon^3)$-competitive algorithm. The problem of speed scaling with power down scheduling on a single machine has been first studied in [10]. Irani et al. [10] derived an algorithm with competitive ratio $(2^{2\alpha - 2\alpha^\alpha} + 2^{\alpha - 1} + 2)$. Subsequently, Han et al. [9] presented algorithm SOA which is $\max\{4, \alpha^\alpha + 2\}$-competitive.

2 Framework for Online Convex Assignment

Consider the assignment problem where items $j$ arrive online and need to be (fractionally) assigned to some agents $i$ with the following objective and constraints.

$$\min \quad P(x) := \sum_i f_i \left( \sum_j a_{ij} x_{ij} \right)$$

subject to

$$g_i \left( \sum_j b_{ij} x_{ij} \right) \leq 0 \quad \forall i$$

$$h_j \left( \sum_i c_{ij} x_{ij} \right) \leq 0 \quad \forall j$$

$$x_{ij} \geq 0 \quad \forall i, j$$

where $x_{ij}$ indicates the amount of item $j$ assigned to agent $i$ and functions $f_i, g_i, h_j$ are convex, differential for every $i, j$ and $a_{ij}, b_{ij} \geq 0$. Denote $k < j$ if item $k$ is released before item $j$.

Let $F$ be the set of feasible solutions of $(P)$. The Lagrangian dual is $\max_{\lambda, \gamma \geq 0} \min_{x \in F} L(x, \lambda, \gamma)$
where \( L \) is the following Lagrangian function

\[
L(x, \lambda, \gamma) = \sum_i f_i \left( \sum_j a_{ij} x_{ij} \right) + \sum_i \lambda_i g_i \left( \sum_j b_{ij} x_{ij} \right) + \sum_j \gamma_j h_j \left( \sum_i c_{ij} x_{ij} \right)
\]

\[
\geq \sum_{i,j} (x_{ij} - x_{ij}^*) \left[ a_{ij} f'_i \left( \sum_k a_{ik} x_{ik}^* \right) + \lambda_i b_{ij} g'_i \left( \sum_k b_{ik} x_{ik}^* \right) + \gamma_j c_{ij} h'_j \left( \sum_i c_{ij} x_{ij}^* \right) \right]
\]

\[
+ \sum_i f_i \left( \sum_j a_{ij} x_{ij}^* \right) + \sum_i \lambda_i g_i \left( \sum_j b_{ij} x_{ij}^* \right) + \sum_j \gamma_j h_j \left( \sum_i c_{ij} x_{ij}^* \right)
\]

where the inequalities holds for any \( x^* \) due the convexity of functions \( f_i, g_i, h_j \)'s. In the first inequality, we use \( f_i(z) \geq f_i(z^*) + (z - z^*)f'_i(z^*) \) (similarly for functions \( g_i, h_j \)'s) and in the second inequality, we use the monotonicity of \( f'_i \) (and similarly for \( g'_i \)). Denote

\[
M(x, x^*, \lambda, \gamma) := \sum_{i,j} (x_{ij} - x_{ij}^*) \left[ a_{ij} f'_i \left( \sum_k a_{ik} x_{ik}^* \right) + \lambda_i b_{ij} g'_i \left( \sum_k b_{ik} x_{ik}^* \right) + \gamma_j c_{ij} h'_j \left( \sum_i c_{ij} x_{ij}^* \right) \right]
\]

\[
N(x^*, \lambda, \gamma) := \sum_i f_i \left( \sum_j a_{ij} x_{ij}^* \right) + \sum_i \lambda_i g_i \left( \sum_j b_{ij} x_{ij}^* \right) + \sum_j \gamma_j h_j \left( \sum_i c_{ij} x_{ij}^* \right)
\]

We have

\[
L(x, \lambda, \gamma) \geq M(x, x^*, \lambda, \gamma) + N(x^*, \lambda, \gamma) \tag{1}
\]

Intuitively, one could imagine that \( x^* \) is the solution of an algorithm (or a function on the solution of an algorithm). We emphasize that \( x^* \) is not a solution of an optimal assignment. The goal is to design an algorithm, which produces \( x^* \) and derives dual variables \( \lambda, \gamma \), in such a way that the primal objective is bounded by a desired factor from the dual one.

Inequality (1) naturally leads to the following idea of an algorithm. For any item \( j \), we maintain the following invariants

\[
a_{ij} f'_i \left( \sum_k a_{ik} x_{ik}^* \right) + \lambda_i b_{ij} g'_i \left( \sum_k b_{ik} x_{ik}^* \right) + \gamma_j c_{ij} h'_j \left( \sum_i c_{ij} x_{ij}^* \right) \geq 0 \quad \forall i
\]

\[
a_{ij} f'_i \left( \sum_k a_{ik} x_{ik}^* \right) + \lambda_i b_{ij} g'_i \left( \sum_k b_{ik} x_{ik}^* \right) + \gamma_j c_{ij} h'_j \left( \sum_i c_{ij} x_{ij}^* \right) = 0 \quad \text{if } x_{ij}^* > 0
\]

Whenever the invariants hold for every \( j \), \( M(x, x^*, \lambda, \gamma) \geq 0 \) since \( x_{ij} \geq 0 \) for every \( i,j \). Therefore, \( L(x, \lambda, \gamma) \geq N(x^*, \lambda, \gamma) \) and so the dual is lower-bounded by \( N(x^*, \lambda, \gamma) \), which does not depend anymore on \( x \). The procedure of maintaining the invariants dictate the decision \( x^* \) of an algorithm and indicates the choice of dual variables.
Consider the following dual

$$\begin{align*}
\max \quad & N(x^*, \lambda, \gamma) \\
\text{subject to} \quad & a_{ij} f'_i \left( \sum_{k<j} a_{ik} x^*_k \right) + \lambda_i b_{ij} g'_i \left( \sum_{k<j} b_{ik} x^*_k \right) + \gamma_j c_{ij} h'_j \left( \sum_i c_{ij} x^*_i \right) \geq 0 \quad \forall i, j \\
& a_{ij} f'_i \left( \sum_{k<j} a_{ik} x^*_k \right) + \lambda_i b_{ij} g'_i \left( \sum_{k<j} b_{ik} x^*_k \right) + \gamma_j c_{ij} h'_j \left( \sum_i c_{ij} x^*_i \right) = 0 \text{ if } x^*_{ij} > 0 \quad \forall i, j \\
& x^*, \lambda, \gamma \geq 0 \quad \forall i, j
\end{align*}$$

Lemma 1 (Weak Duality) Let $OPT(P)$ and $OPT(D)$ be optimal values of primal program $(P)$ and dual program $(D)$, respectively. Then $OPT(P) \geq OPT(D)$.

Proof. It holds that

$$OPT(P) \geq \max_{\lambda, \gamma \geq 0} \min_{x \in X} L(x, \lambda, \gamma) \geq N(x^*, \lambda, \gamma)$$

where the inequalities follow the weak Lagrangian duality and the constraints of $(D)$ for every feasible solution $x^*, \lambda, \gamma$. Therefore, the lemma follows. \qed

Hence, our framework consists of maintaining the invariants for every online item $j$ and among feasible set of dual variables (constrained by the invariants) choose the ones which optimize the ratio between the primal and dual values. If an algorithm with output $x^*$ satisfies $P(x^*) \leq r N(x^*, \lambda, \gamma)$ for some factor $r$ then the algorithm is $r$-competitive.

3 Minimizing Total Energy plus Lost Values

The problem. We are given a machine with a convex energy power $P$ and jobs arrive over time. Each job $j$ is released at time $r_j$, has deadline $d_j$, processing volume $p_j$ and a value $a_j$. Jobs could be executed preemptively and at a time $t$, the scheduler has to choose a set of pending jobs (i.e., $r_j \leq t < d_j$) and a machine speed $s(t)$ in order to process such jobs. The energy cost of a schedule is $\int_0^\infty P(s(t))dt$. Typically, $P(z) = z^\alpha$ for some constant $\alpha \geq 1$. The objective of the problem is to minimize energy cost plus the lost value — which is the total value of uncompleted jobs.

Formulation. Let $x_j$ and $y_j$ be variables indicating whether job $j$ is completed or it is not. We denote variable $s_j(t)$ as the speed that the machine processes job $j$ at time $t$. The problem could be relaxed as the following convex program.

$$\begin{align*}
\min \quad & \int_0^\infty P(s(t))dt + \sum_j a_j y_j \\
\text{subject to} \quad & s(t) = \sum_j s_j(t) \quad \forall t \\
& x_j + y_j \geq 1 \quad \forall j \\
& \int_{r_j}^{d_j} s_j(t)dt \geq p_j x_j \quad \forall j \\
& x_j, y_j, s_j(t) \geq 0 \quad \forall j, t
\end{align*}$$
In the relaxation, the second constraint indicates that either job $j$ is completed or it is not. The third constraint guarantees the necessary amount of work done in order to complete job $j$.

Applying the framework, we have the following dual.

$$\max \int_0^\infty P\left(\sum_j v_j^*(t)\right) dt - \sum_j \lambda_j \int_{r_j}^{d_j} v_j^*(t) dt + \sum_j \gamma_j$$

subject to

1. For any job $j$, $\gamma_j \leq p_j \lambda_j$. Moreover, if $x_j^* > 0$ then $\gamma_j = p_j \lambda_j$.
2. For any job $j$, $\gamma_j \leq a_j$ and if $y_j^* > 0$ then $\gamma_j = a_j$.
3. For any job $j$ and any $t \in [r_j, d_j]$, it holds that $\lambda_j \leq P'(\sum_k v_k^*(t))$. Particularly, if $v_j^*(t) > 0$ then $\lambda_j = P'(\sum_k v_k^*(t))$.

Note that $v_j^*(t)$ is not equal to $s_j^*(t)$ (the machine speed on job $j$ according to our algorithm) but it is a function depending on $s_j^*(t)$. That is the reason we use $v_j^*(t)$ instead of $s_j^*(t)$. We will choose $v_j^*(t)$’s in order to optimize the competitive ratio. To simplify the notation, we drop out the star symbol in the superscript of every variable (if one has that).

**Algorithm.** The dual constraints naturally leads to the following algorithm. We first describe informally the algorithm. In the algorithm, we maintain a variable $u_j(t)$ representing the virtual machine speed on job $j$. The virtual speed on job $j$ means that job $j$ will be processed with that speed if it is accepted; otherwise, the real speed on $j$ will be set to 0. Consider the arrival of job $j$. Observe that by the third constraint, we should always increase the machine speed on job $j$ at $\arg\min P'(v(t))$ in order to increase $\lambda_j$. Hence, at the arrival of a job $j$, increase continuously the virtual speed $u_j(t)$ of job $j$ at $\arg\min P'(v(t))$ for $r_j \leq t \leq d_j$. Moreover, function $v(t)$ is also simultaneously updated as a function of $u(t) = \sum_{k \neq j} u_k(t)$ according to a system of differential equations (2) in order to optimize the competitive ratio. The iteration on job $j$ terminates whether one of the first two constraints becomes tight. If the first one holds, then accept the job and set the real speed equal to the virtual one. Otherwise, reject the job.

Define $Q(z) := P(z) - zP'(z)$. Consider the following system of differential equations with boundary conditions: $Q(v) = 0$ if $u = 0$.

$$Q'(v) \frac{dv}{du} + P'(v) \geq \frac{P'(u)}{r},$$

$$(r - 1)P'(u) + rQ'(v) \frac{dv}{du} \geq 0,$$

$$\frac{dv}{du} > 0,$$

where $r$ is some constant. Let $r^* \geq 1$ be a smallest constant such that the system has a solution.

The formal algorithm is given in Algorithm 1.

In the algorithm, machine $i$ processes accepted job $j$ with speed $s_j(t)$ at time $t$. As the algorithm completes all accepted jobs, it is equivalent to state that the machine processes accepted jobs in the earliest deadline first fashion with speed $s(t)$ at time $t$.

By the algorithm, the dual variables are feasible. In the following we bound the values of the primal and dual objectives.
Algorithm 1: Minimizing the consumed energy plus lost values.

1: Initially, set \( s(t), s_j(t) \) and \( u_j(t) \)'s and \( v(t) \), \( v_j(t) \)'s equal to 0.
2: Let \( r^* \geq 1 \) be the smallest constant such that (2) has a solution. During the algorithm, keep \( v(t) \) as a solution of (2) with constant \( r^* \) and \( u(t) = \sum_j u_j(t) \) for every time \( t \).
3: for a job \( j \) arrives do
4: \hspace{1em} Initially, \( u_j(t) \leftarrow 0 \).
5: \hspace{1em} while \( \int_{t^j}^{d_j} u_j(t) dt < p_j \text{ and } \lambda_j p_j < a_j \) do
6: \hspace{2em} Continuously increase \( u_j(t) \) at \( \text{arg min} P'(v(t)) \) for \( r_j \leq t \leq d_j \) and update \( u(t) \leftarrow \sum_{k \neq j} u_k(t) + u_j(t) \) and \( v(t) \) (as a function of \( u(t) \)) and \( \lambda_j \leftarrow \min_{r_j \leq t \leq d_j} P'(v(t)) \) simultaneously.
7: \hspace{1em} end while
8: \hspace{1em} Set \( v_j(t) \leftarrow v(t) - \sum_{k \neq j} v_k(t) \).
9: \hspace{1em} if \( \lambda_j p_j = a_j \text{ and } \int_{t^j}^{d_j} u_j(t) dt < p_j \) then
10: \hspace{2em} Reject job \( j \).
11: \hspace{1em} Set \( \gamma_j \leftarrow a_j \).
12: \hspace{1em} else
13: \hspace{2em} Accept job \( j \).
14: \hspace{2em} Set \( s_j(t) \leftarrow u_j(t), s(t) \leftarrow s(t) + s_j(t) \) and \( \gamma_j \leftarrow \lambda_j p_j \).
15: \hspace{1em} end if
16: end for

Lemma 2: It holds that

\[
\int_0^\infty P(s(t)) dt + \sum_j a_j y_j \leq r^* \left( \int_0^\infty P\left( \sum_j v_j(t) \right) dt - \sum_j \lambda_j \int_{t^j}^{d_j} v_j(t) dt + \sum_j \gamma_j \right)
\]

Proof: By the algorithm, \( \lambda_j = P'(v(t)) \) at every time \( t \) such that \( v_j(t) > 0 \) for every job \( j \). Hence, it is sufficient to show that

\[
\int_0^\infty P(s(t)) dt + \sum_j a_j y_j \leq r^* \left( \int_0^\infty Q\left( \sum_j v_j(t) \right) dt + \sum_j \gamma_j \right)
\]

where recall \( Q(z) = P(z) - z P'(z) \).

We will prove the inequality (3) by induction on the number of jobs in the instance. For the base case where there is no job, the inequality holds trivially. Suppose that the inequality holds before the arrival of a job \( j \). In the following, we consider different cases.

**Job \( j \) is accepted.** Consider any moment \( \tau \) in the while loop related to job \( j \). We emphasize that \( \tau \) is a moment in the execution of the algorithm, not the one in the time axis \( t \). Suppose that at moment \( \tau \), an amount \( du_j(t) \) is increased (allocated) at \( t \). Note that \( du_j(t) = du(t) \) as \( u(t) = \sum_j u_j(t) \). As \( j \) is accepted, \( y_j = 0 \) and the increase at \( \tau \) in the left hand-side of (3) is \( P'(u(t)) du(t) \)
Let $v(t_1, \tau_1)$ be the value of $v(t)$ at moment $\tau_1$ in the while loop. By the algorithm, the dual variable $\gamma_j$ satisfies

$$\gamma_j = \lambda_j p_j \geq \int_{\tau_1} \left( \int_{r_j}^{d_j} u_j(t_2) dt_2 \right) \min_{r_j \leq t_1 \leq d_j} P'(v(t_1, \tau_1)) dt_1$$

$$= \int_{\tau_1} \int_{r_j}^{d_j} u_j(t_2) \min_{r_j \leq t_1 \leq d_j} P'(v(t_1, \tau_1)) dt_2 dt_1$$

where the inequality is due to the fact that at the end of the while loop, $\int_{r_j}^{d_j} u_j(t) dt = p_j$ (by the loop condition) and $P'$ is increasing. Therefore, at moment $\tau$, $d\gamma_j \geq \min_{r_j \leq t_1 \leq d_j} P'(v(t_1, \tau)) du_j(t) = P'(v(t)) du(t)$ where the equality follows since $t \in \arg\min_{r_j \leq t_1 \leq d_j} P'(v(t_1, \tau))$. Hence, the increase in the right-hand side of (3) is at least $r^* [Q'(v(t)) dv(t) + P'(v(t)) du(t)]$.

Due to the system of inequations (2) and the choice of $r^*$, at any moment in the execution of the algorithm, the increase in the left-hand side of (3) is at most that in the right-hand side. Thus, the induction step follows.

**Job $j$ is rejected.** If $j$ is rejected then $y_j = 1$ and so the increase in the left-hand side of (3) is $a_j$. Moreover, by the algorithm $\gamma_j = a_j$. So we need to prove that after the iteration of the for loop on job $j$, it holds that $(r^* - 1)a_j + r^* \int_0^\infty \Delta Q(v(t)) dt \geq 0$. As $j$ is rejected,

$$a_j = \lambda_j p_j > \int_{r_j}^{d_j} \min_{r_j \leq t_1 \leq d_j} P'(v(t_1)) u_j(t) dt$$

Therefore, it is sufficient to prove that

$$(r^* - 1) \int_{r_j}^{d_j} \min_{r_j \leq t_1 \leq d_j} P'(v(t_1)) u_j(t) dt + r^* \int_0^\infty \Delta Q(v(t)) dt \geq 0$$

(4)

Before the iteration of the while loop, the left-hand side of (4) is 0. Similar as the analysis of the previous case, during the execution of the algorithm the increase rate of the left-hand side is $(r^* - 1)P'(v(t)) du(t) + r^* Q'(v(t)) dv(t)$, which is non-negative by equation (2). Thus, inequality (4) holds.

By both cases, the lemma follows.

**Theorem 1** The algorithm is $r^*$-competitive. Particularly, if the energy power function $P(z) = z^\alpha$ then the algorithm is $\alpha^\alpha$-competitive.

**Proof** The theorem follows by the framework and Lemma 2.

If the power energy function $P(z) = z^\alpha$ then $r^* = \alpha^\alpha$ and $u(t) = v(t) / \alpha$ satisfy the system (2). Thus, the algorithm is $\alpha^\alpha$-competitive.

4 Maximizing the Total Value minus Energy

**The problem.** We are given unrelated machines and jobs arrive over time. Each job $j$ is released at time $r_j$, has deadline $d_j$, a value $a_j$ and processing volume $p_{ij}$ if it is executed on machine $i$. 
Jobs could be executed preemptively but migration is not allowed, i.e., no job could be executed on more than one machine. At a time \( t \), the scheduler has to choose a set of pending jobs (i.e., \( r_j \leq t < d_j \)) to be processed on each machine, and the speeds \( s_i(t) \)'s for every machine \( i \) to execute such jobs. The objective now is to maximize the total value of completed jobs minus the energy cost — which is \( \sum_i \int_0^\infty P(s_i(t))dt \) where \( P \) is a given convex power function.

It is shown that without resource augmentation, the competitive ratio is unbounded [13]. In this section, we consider the problem with resource augmentation, meaning that with the same speed \( z \) the energy power for the algorithm is \( P((1 - \epsilon)z) \), whereas the one for the adversary is \( P(z) \). Let \( \epsilon(P) > 0 \) be the smallest constant such that \( zP'((1 - \epsilon(P))z) \leq P(z) \) for all \( z > 0 \). For the typical energy power \( P(z) = z^\alpha \), \( \epsilon(P) \approx 1 - \alpha^{-1/\alpha} \) which is close to 0 for \( \alpha \) large.

**Formulation.** Let \( x_{ij} \) be variable indicating whether \( j \) is completed in machine \( i \). Moreover, variable \( y_j \) equals 1 if \( j \) is not completed and equals 0 otherwise. Let \( s_{ij}(t) \) be the variable representing the speed that machine \( i \) processes job \( j \) at time \( t \). The problem could be formulated as the following convex program.

\[
\max \sum_{i,j} a_{ij}x_{ij} - \sum_i \int_0^\infty P((1 - \epsilon)s_i(t))dt
\]

subject to
\[
s_i(t) = \sum_j s_{ij}(t) \quad \forall i, t
\]
\[
\sum_i x_{ij} \leq 1 \quad \forall j
\]
\[
\int_{r_j}^{d_j} s_{ij}(t)dt \geq p_{ij}x_{ij} \quad \forall i, j
\]
\[
x_{ij}, s_{ij}(t) \geq 0 \quad \forall i, j, t
\]

Note that in the objective, by resource augmentation the consumed energy is \( \sum_i \int_0^\infty P((1 - \epsilon)s_i(t))dt \). Applying the framework, we have the following dual.

\[
\min \sum_j \gamma_j - \sum_i \int_0^\infty P\left(\sum_j u_{ij}^*(t)\right)dt + \sum_{i,j} \lambda_{ij} \int_{r_j}^{d_j} u_{ij}^*(t)dt
\]

subject to

1. For any machine \( i \) and any job \( j \), \( \gamma_j + p_{ij}\lambda_{ij} \geq a_j \).

2. For any machine \( i \), any job \( j \) and any \( t \in [r_j, d_j] \), \( \lambda_{ij} \leq P'((1 - \epsilon)\sum k u_{ik}^*(t)) \) where the sum is taken over all jobs \( k \) released before \( j \), i.e., \( k \preceq j \). Particularly, if \( u_{ij}^*(t) > 0 \) then \( \lambda_{ij} = P'((1 - \epsilon)\sum_{k \preceq j} u_{ik}^*(t)) \).

Similar as in the previous section, the constraints naturally lead to Algorithm 2. In the algorithm and the analysis, to simplify the notation we drop out the star symbol in the superscript of every variable (if one has that).

**Lemma 3** Dual variables constructed by Algorithm 2 are feasible.
Algorithm 2 Minimizing the throughput minus consumed energy.

1: Initially, set $s(t)$ and $u(t)$ equal to 0.
2: The algorithm always runs accepted jobs with speed $s(t)$ in the earliest deadline first fashion.
3: for a job $j$ arrives do
4: Initially, $u_{ij}(t) \leftarrow 0$ for every $t$ and let $\mathcal{I}$ be the set of all machines, $\mathcal{I}' \leftarrow \emptyset$.
5: while $\mathcal{I} \neq \emptyset$ do
6: For every $i \in \mathcal{I}$, increase $u_{ij}(t)$ at $\arg \min P'(u_i(t))$ in the continuous manner for $r_j \leq t \leq d_j$ and update $u_i(t) = \sum_{k \neq j} u_{ik}(t) + u_{ij}(t)$ and $\lambda_{ij} = \min_{r_j \leq t \leq d_j} P'((1 - \epsilon)u_i(t))$ simultaneously.
7: if $\lambda_{ij} p_{ij} = a_j$ and $\int_{r_j}^{d_j} u_{ij}(t) dt < p_j$ for some machine $i$ then
8: $\mathcal{I} \leftarrow \mathcal{I} \setminus \{i\}$.
9: end if
10: if $\lambda_{ij} p_{ij} < a_j$ and $\int_{r_j}^{d_j} u_{ij}(t) dt = p_j$ for some machine $i$ then
11: $\mathcal{I}' \leftarrow \mathcal{I}' \cup \{i\}$ and $\mathcal{I} \leftarrow \mathcal{I} \setminus \{i\}$.
12: end if
13: end while
14: if $\mathcal{I}' = \emptyset$ then
15: Reject job $j$ and set $\gamma_j \leftarrow 0$ (note that $p_{ij} \lambda_{ij} = a_j \forall i$).
16: else
17: Let $i = \arg \min_{i' \in \mathcal{I}'} p_{i'j} \lambda_{i'j}$.
18: Accept and assign job $j$ to machine $i$, i.e., $x_{ij} = 1$.
19: Set $s_{ij}(t) \leftarrow u_{ij}(t)$, $s_i(t) \leftarrow s_i(t) + s_{ij}(t)$ and $\gamma_j \leftarrow a_j - \lambda_{ij} p_j$.
20: end if
21: end for

Proof By the algorithm (line 6), $\lambda_{ij} \leq P'((1 - \epsilon)\sum_k u_{ik}(t))$ where the sum is taken over all jobs $k$ released before $j$ ($k \leq j$) and if $u_{ij}(t) > 0$ then $\lambda_{ij} = P'((1 - \epsilon)\sum_{k < j} u_{ik}(t))$. Consider the first constraint. If $j$ is rejected then $\gamma_j = a_j$. Otherwise, by the assignment (line 17), it always holds that $\gamma_j + p_{ij} \lambda_{ij} \geq a_j$ for every $i, j$. □

In the following we bound the values of the primal and dual objectives in the resource augmentation model.

Lemma 4 For every $\epsilon \geq \epsilon(P)$, it holds that

$$\frac{1}{\epsilon} \left( \sum_{i, j} a_j x_{ij} - \sum_i \int_0^\infty P((1 - \epsilon)s_i(t)) dt \right) \geq \sum_j \gamma_j - \sum_i \int_0^\infty P\left( \sum_j u_{ij}(t) \right) dt + \sum_{i, j} \lambda_{ij} \int_{r_j}^{d_j} u_{ij}(t) dt$$
Proof We have
\[
\sum_{i,j} \lambda_{ij} \int_{t_j}^{d_j} u_{ij}(t)dt - \sum_i \int_0^\infty P(u_i(t))dt \\
\leq \sum_i \int_0^\infty \left( \max_{j: r_j \leq t \leq d_j} \lambda_{ij} \right) \sum_j u_{ij}(t)dt - \sum_i \int_0^\infty P(u_i(t))dt \\
\leq \sum_i \int_0^\infty P'((1 - \epsilon)u_i(t))u_i(t)dt - \sum_i \int_0^\infty P(u_i(t))dt \leq 0
\]
where the second inequality is due to the algorithm, and the last inequality follows by the definition of \( \epsilon(P) \). Therefore, it is sufficient to prove that
\[
\frac{1}{\epsilon} \left( \sum_{i,j} a_j x_{ij} - \sum_i \int_0^\infty P((1 - \epsilon)s_i(t))dt \right) \geq \sum_j \gamma_j \tag{5}
\]
We prove inequality (5) by induction on the number of released jobs in the instance. For the base case where there is no job, the inequality holds trivially. Suppose that the inequality holds before the arrival of a job \( j \).

If \( j \) is rejected then \( x_{ij} = 0, s_j(t) = 0 \) for every \( i, t \) and \( \gamma_j = 0 \). Therefore, the increases in both side of inequality (5) are 0. Hence, the induction step follows.

In the rest, assume that \( j \) is accepted and for simplicity let \( i \) be the machine to which \( j \) is assigned. We have
\[
\int_0^\infty \left[ P\left( (1 - \epsilon)u_i(t) \right) - P\left( (1 - \epsilon)(u_i(t) - u_{ij}(t)) \right) \right]dt \leq (1 - \epsilon) \int_0^\infty P'\left( (1 - \epsilon)u_i(t) \right)u_{ij}(t)dt \\
\leq (1 - \epsilon) \int_0^\infty P'(u_i(t))u_{ij}(t)dt = (1 - \epsilon)\lambda_{ij} \int_0^\infty u_{ij}(t)dt = (1 - \epsilon)\lambda_{ij} p_{ij} \leq (1 - \epsilon)a_j
\]
The first inequality is due to the convexity of \( P \)
\[
P\left( (1 - \epsilon)(u_i(t) - u_{ij}(t)) \right) \geq P\left( (1 - \epsilon)u_i(t) \right) - (1 - \epsilon)u_{ij}(t)P'\left( (1 - \epsilon)u_i(t) \right).
\]
The second inequality holds because \( P' \) is increasing. The first equality follows since \( u_{ij}(t) \neq 0 \) only at \( t \) such that \( P'(u_i(t)) = \lambda_{ij} \) (by the algorithm). The last inequality is due to the loop condition in the algorithm. Thus, at the end of the iteration (related to job \( j \)) in the for loop, the increase in the left hand side of inequality (5) is
\[
\frac{1}{\epsilon} \left( a_j x_{ij} - \int_0^\infty \left[ P\left( (1 - \epsilon)u_i(t) \right) - P\left( (1 - \epsilon)(u_i(t) - u_{ij}(t)) \right) \right]dt \right) \geq \frac{1}{\epsilon} \left( a_j - (1 - \epsilon)a_j \right) = a_j
\]
Besides, the increase in the right hand-side of inequality (5) is \( \gamma_j \leq a_j \). Hence, the induction step follows; so does the lemma.

Theorem 2 The algorithm is \((1 + \epsilon)\)-augmentation, \(1/\epsilon\)-competitive for \( \epsilon \geq \epsilon(P) \).

Proof By resource augmentation, with the same speed \( z \) the energy power for the algorithm is \( P((1 - \epsilon)z) \), whereas the one for the adversary is \( P(z) \). So in the inequality of Lemma 4, the left hand side represents exactly the objective value of the algorithm in the model of resource augmentation. Hence, the theorem follows.
5 Speed Scaling with Power Down Scheduling

The problem. We are given a single machine that could be transitioned into a sleep state or an active state. Each transition from the sleep state to the active state costs $A > 0$, which is called the wake-up cost. Jobs arrive online, each job has a released time $r_j$, a deadline $d_j$, a processing volume $p_j$ and could be processed preemptively. In the problem, all jobs have to be completed. In the sleep state, the energy consumption of the machine is 0. In the active state, the power consumption is $P(s(t)) = s(t)^\alpha + g$ where $\alpha \geq 1$ and $g \geq 0$ are constant. Thus, the consumed energy of the machine in active state is $\int_0^{\infty} P(s(t))dt$, that can be decomposed into dynamic energy $\int_0^{\infty} s(t)^\alpha dt$ and static energy $\int_0^{\infty} gd t$ (where the integral is taken over $t$ at which the machine is in active state). At any time $t$, the scheduler has to decide the state of the machine and the speed if the machine is in active state in order to execute and complete all jobs. The objective is to minimize the total energy — the consumed energy in active state plus the wake-up energy.

Formulation. Let $H(t)$ be the Heaviside step function, i.e., $H(t) = 0$ if $t < 0$ and $H(t) = 1$ if $t \geq 0$. Then $H(t)$ is the integral of the Dirac delta function $H' = \delta$ and it holds that $\int_{-\infty}^{\infty} \delta(t)dt = 1$. We use this fact to formulate a relaxation for the problem. Let $F(t)$ be a function indicating whether the machine is in the active state, i.e., $F(t) = 1$ if the machine is in the active state and equals 0 otherwise. Assume that initially the machine is in the sleep state. Then $A \int_0^{\infty} |F'(t)|dt$ equals twice the wake-up energy.

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

$$\min \int_0^{\infty} P\left(\sum_j s_j(t)\right) F(t)dt + \frac{A}{2} \int_0^{\infty} |F'(t)|dt$$

subject to:

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

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

5.1 Speed Scaling without Wake-Up Cost

The problem without wake-up cost ($A = 0$) has been extensively studied. We reconsider the problem throughout our primal-dual approach. In case $A = 0$, the machine is put in active state whenever there is some pending job (thus the function $F(t)$ is useless and could be removed from the formulation). In this case, the relaxation above becomes a convex program. Applying the framework and by the same observation as in previous sections, we derive the following algorithm.

At the arrival of job $j$, increase continuously $s_j(t)$ at $\arg \min P'(s(t))$ for $r_j \leq t \leq d_j$ and update simultaneously $s(t) \leftarrow s(t) + s_j(t)$ until $\int_{r_j}^{d_j} s_j(t')dt' = p_j$.

It turns out that the machine speed $s(t)$ of the algorithm equals $\max_{t' > t} V(t, t')/(t' - t)$ where $V(t, t')$ is the remaining processing volume of jobs arriving at or before $t$ with deadline in $(t, t']$. So the algorithm is indeed algorithm \textsc{Optimal Available} [15] that is $\alpha^\alpha$-competitive [2]. However, the primal-dual view of the algorithm gives more insight and that is useful for the general model with wake-up cost (see Lemma 5).
5.2 Speed Scaling with Wake-Up Cost

The Algorithm. Define the critical speed \( s^c = \arg \min_{s > 0} P(s)/s \). In the algorithm, the machine speed is always at least \( s^c \) if it executes some job.

Initially, set \( s(t) \) and \( s_j(t) \) equal 0 for every time \( t \) and jobs \( j \). If a job is released then it is marked as active. Intuitively, a job is active if its speed \( s_j(t) \) has not been settled yet. Let \( \tau \) be the current moment. Consider currently active jobs in the earliest deadline first (EDF) order. Increase continuously \( s_j(t) \) at \( \arg \min P'(s(t)) \) for \( r_j \leq t \leq d_j \) and update simultaneously \( s(t) \leftarrow s(t) + s_j(\tau) \) until \( \int_{r_j}^{d_j} s_j(t')dt' = p_j \). Now consider different states of the machine at the current time \( \tau \). We distinguish three machine states: (1) in working state the machine is active and is processing 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 keep process jobs with the earliest deadline by speed \( \max\{s(\tau), s^c\} \).
Mark all currently pending jobs as inactive. If \( s(\tau) = 0 \), switch to idle state.

In idle state. If \( s(\tau) \geq s^c \) then switch to working state.
If \( s^c > s(t) > 0 \). Mark all currently pending jobs as active. Notice that if there is no new job released then the active jobs will be processed at speed \( s^c \) during some period in the future.
Otherwise, if the total duration of idle state from the last wake-up equals \( A/g \) then switch to sleep state.

In sleep state. If \( s(t) \geq s^c \) then switch to the working state.

In the rest, we denote \( s^*(t) \) as the machine speed at time \( t \) by the algorithm.

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

\[
L(s,F,\lambda) = \int_0^\infty P\left(\sum_j s_j(t)\right)F(t)dt + \frac{A}{2} \int_0^{+\infty} |F'(t)|dt + \sum_j \lambda_j \left(p_j - \int_{r_j}^{d_j} s_j(t)F(t)dt\right)
\[
\geq \sum_j \lambda_j p_j - \sum_j \int_{r_j}^{d_j} s_j(t)F(t)\left(\lambda_j - \frac{P(s(t))}{s(t)}\right)\mathbb{1}_{\{s(t) > 0\}}\mathbb{1}_{\{F(t) = 1\}}dt + \frac{A}{2} \int_0^{+\infty} |F'(t)|dt
\]

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, we bound the Lagrangian dual value in function of the primal objective value due to the algorithm and derive the competitive ratio. That is done by the dual-fitting approach presented in [14].

Dual variables. Let \( 0 < \beta \leq 1 \) be some constant to be chosen later. For jobs \( j \) such that \( s^*_j(t) > 0 \) for every \( t \in [r_j, d_j] \) then define \( \lambda_j \) such that \( \lambda_j p_j/\beta \) equals the marginal increase in the dynamic energy due to the arrival of job \( j \). For jobs \( j \) such that \( s^*_j(t) = 0 \) for some moment \( t \in [r_j, d_j] \) then define \( \lambda_j \) such that \( \lambda_j p_j \) equals the marginal increase in the energy due to the arrival of job \( j \).

Lemma 5 Let \( j \) be an arbitrary job.
1. If \( s^*(t) > 0 \) for every \( t \in [r_j, d_j] \) then \( \lambda_j \leq \beta P'(s^*(t)) \) for every \( t \in [r_j, d_j] \).

2. Moreover, if \( s^*(t) = 0 \) 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 \( j \). So \( s^*(t) \) is the machine speed after the arrival of \( j \). The marginal increase in the dynamic energy due to the arrival of \( j \) could be written as

\[
\frac{1}{\beta} \lambda_j p_j = \int_{r_j}^{d_j} \left( P(s^*(t)) - P(s^*(t) - s^*_j(t)) \right) dt \leq \int_{r_j}^{d_j} P'(s^*(t)) s^*_j(t) dt
\]

\[
= \min P'(s^*(t)) \int_{r_j}^{d_j} s^*_j(t) dt = \min P'(s^*(t)) p_j
\]

where \( \min P'(s^*(t)) \) is taken over \( t \in [r_j, d_j] \) such that \( s^*_j(t) > 0 \). The inequality is due to the convexity of \( P \) and the first equality follows by the algorithm. Moreover, \( \min P'(s^*(t)) \leq P'(s^*(t)) \) for \( t \in [r_j, d_j] \); so the lemma follows.

We are now showing the second claim. By the algorithm, the fact that \( s^*(t) = 0 \) for some \( t \in [r_j, d_j] \) means that job \( j \) will be processed at speed \( s^c \) in some interval \([a, b] \subset [r_j, d_j]\). The marginal increase in the energy is \( P(s^c)(b - a) \) while \( p_j \) could be expressed as \( s^c(b - a) \). Therefore, \( \lambda_j = P(s^c)/s^c \). \( \square \)

**Theorem 3** The algorithm has competitive ratio at most \( \max\{4, \alpha^\alpha\} \).

**Proof** Let \( E_1^* \) be the dynamic energy of the algorithm schedule, i.e., \( E_1^* = \int_0^\infty [P(s^*(t)) - P(0)] dt \leq \sum_j \lambda_j p_j / \beta \) due to the definition of \( \lambda_j \)'s and \( 0 < \beta \leq 1 \). Moreover, let \( E_2^* \) be the static energy plus the wake-up energy of the algorithm, i.e., \( E_2^* = \int_0^\infty P(0) F^*(t) dt + \frac{1}{2} \int_0^\infty (F^*)'(t) dt \). We will bound the Lagrangian dual objective.

By Lemma 5, for every job \( j \) such that \( s^*(t) = 0 \) 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,

\[
\sum_j \int_{r_j}^{d_j} s_j(t) F(t) \left( \lambda_j - \frac{P(s(t))}{s(t)} \right) dt \leq 0 \tag{6}
\]
where in the sum is taken over jobs \( j \) such that \( s^*(t) = 0 \) for some \( t \in [r_j, d_j] \). Therefore,

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

\[
\geq \beta E_1^* + \max_{s,F} \sum_j \int_{r_j}^{d_j} s_j(t) F(t) \left[ \beta P'(s^*(t)) - \frac{P(s(t))}{s(t)} \right] \mathbb{1}_{\{s(t) > 0\}} \mathbb{1}_{\{F(t) = 1\}} \mathbb{1}_{\{s^*(t) > 0\}} dt
\]

\[
\geq \beta E_1^* - \max_s \int_0^{\infty} s(t) \left[ \beta P'(s^*(t)) - \frac{P(s(t))}{s(t)} \right] \mathbb{1}_{\{s(t) > 0\}} \mathbb{1}_{\{F(t) = 1\}} \mathbb{1}_{\{s^*(t) > 0\}} dt
\]

\[
\geq \beta E_1^* - \frac{1}{2} \int_0^{\infty} \left[ \beta P'(s^*(t)) \bar{s}(t) - P(\bar{s}(t)) \right] \mathbb{1}_{\{s^*(t) > 0\}} dt
\]

\[-\frac{1}{2} \int_0^{\infty} \left[ \beta P'(s^*(t)) \bar{s}(t) - P(\bar{s}(t)) \right] \mathbb{1}_{\{F(t) = 1\}} dt
\]

where in the second line, the sum is taken over jobs \( j \) such that \( s^*(t) > 0 \) for all \( t \in [r_j, d_j] \). The first inequality follows (6). The second inequality holds since \( F(t) \leq 1 \) and \( s(t) = \sum_j s_j(t) \). The third inequality is due to the first order derivative and \( \bar{s}(t) \) is the solution of equation \( P'(z(t)) = \beta P'(s^*(t)) \). In fact \( \bar{s}(t) \) maximizes function \( \beta P'(s^*(t)) - P(s(t))/s(t) \).

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

\[
L_1(s, \lambda) \geq \beta E_1^* - \frac{1}{2} \int_0^{\infty} \left( \beta \alpha(s^*(t))^{\alpha - 1} \bar{s}(t) - (\bar{s}(t))^{\alpha - g} \right) \mathbb{1}_{\{s^*(t) > 0\}} dt
\]

\[-\frac{1}{2} \int_0^{\infty} \left( \beta \alpha(s^*(t))^{\alpha - 1} \bar{s}(t) - (\bar{s}(t))^{\alpha - g} \right) \mathbb{1}_{\{F(t) = 1\}} dt
\]

\[
= \beta E_1^* - \int_0^{\infty} (\alpha - 1) \beta^{\alpha/(\alpha - 1)} (s^*(t))^\alpha dt + \frac{1}{2} \int_0^{\infty} g \mathbb{1}_{\{s^*(t) > 0\}} dt + \frac{1}{2} \int_0^{\infty} g \mathbb{1}_{\{F(t) = 1\}} dt
\]

\[
= \left[ \beta - (\alpha - 1) \beta^{\alpha/(\alpha - 1)} \right] E_1^* + \frac{1}{2} \int_0^{\infty} g \mathbb{1}_{\{s^*(t) > 0\}} dt + \frac{1}{2} \int_0^{\infty} g \mathbb{1}_{\{F(t) = 1\}} dt
\]

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 the following, we claim that \( L_2(F) := \frac{1}{2} \int_0^{\infty} g \mathbb{1}_{\{s^*(t) > 0\}} dt + \frac{1}{2} \int_0^{\infty} g \mathbb{1}_{\{F(t) = 1\}} dt + \frac{A}{2} \int_0^{\infty} |F'(t)| dt \) is at least \( E_2^*/4 \). 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 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., \( \int_u^v g 1_{s^*(t)=0} dt = A \). Let \((s, F)\) be a feasible solution of the relaxation.

If during \([u, v)\), the machine following solution \((s, F)\) makes a transition from non-sleep state to sleep state or inversely then \( L_2(F) \geq \frac{1}{4} \int_u^v g 1_{s^*(t)>0} dt + \frac{A}{4} \). Hence

\[
L_2(F) \geq \frac{1}{4} \left( \int_u^v g 1_{s^*(t)>0} dt + \int_u^v g 1_{s^*(t)=0} dt + A \right) = \frac{1}{4} E_2^*|_{[u,v)}.
\]

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,

\[
L_2(F) \geq \frac{1}{2} \int_u^v g 1_{s^*(t)>0} dt + \frac{1}{2} \int_u^v g 1_{F(t)=1} dt = \frac{1}{2} \int_u^v g 1_{s^*(t)>0} dt + \frac{1}{2} \int_u^v g dt
\]

\[
\geq \frac{1}{2} \int_u^v g 1_{s^*(t)>0} dt + \frac{1}{4} \int_u^v g 1_{s^*(t)=0} dt + \frac{A}{4}
\]

\[
\geq \frac{1}{4} \left( \int_u^v g 1_{s^*(t)>0} dt + \int_u^v g 1_{s^*(t)=0} dt + A \right) = \frac{1}{4} E_2^*|_{[u,v)}
\]

where the second inequality is due to the fact that the machine switches to sleep state at time \( v \), meaning that the total idle duration in \([u, v)\) incurs a cost \( A \).

In conclusion, the dual \( L(s, F, \lambda) \geq E_1^*/\alpha + E_2^*/4 \) whereas the primal is \( E_1^* + E_2^* \). Thus, the competitive ratio is at most \( \max\{4, \alpha^2\} \). \( \square \)

References


