

Energy Aware Scheduling for Weighted Completion Time and Weighted Tardiness

Rodrigo A. Carrasco * Garud Iyengar † Cliff Stein ‡

May 2011, v.arXiv 4.6

Abstract

The ever increasing adoption of mobile devices with limited energy storage capacity, on the one hand, and more awareness of the environmental impact of massive data centres and server pools, on the other hand, have both led to an increased interest in energy management algorithms.

The main contribution of this paper is to present several new constant factor approximation algorithms for energy aware scheduling problems where the objective is to minimize weighted completion time plus the cost of the energy consumed, in the one machine non-preemptive setting, while allowing release dates and deadlines. Unlike previous known algorithms these new algorithms can handle general job-dependent energy cost functions, extending the application of these algorithms to settings outside the typical CPU-energy one. These new settings include problems where in addition, or instead, of energy costs we also have maintenance costs, wear and tear, replacement costs, etc., which in general depend on the speed at which the machine runs but also depend on the types of jobs processed. Our algorithms also extend to approximating weighted tardiness plus energy cost, an inherently more difficult problem that has not been addressed in the literature.

Keywords: energy aware scheduling, approximation algorithms, α -points, weighted tardiness

*rac2159@columbia.edu. Department of Industrial Engineering & Operations Research, Columbia University, Mudd 313, 500W 120th Street, New York, NY 10027. Research partially supported by NSF grants CCF-0728733 and CCF-0915681, and Fulbright/Conicyt Chile Scholarship.

†garud@ieor.columbia.edu. Department of Industrial Engineering & Operations Research, Columbia University, Mudd 314, 500W 120th Street, New York, NY 10027. Research partially supported by NSF grant DMS-1016571, ONR grant N000140310514, and DOE grant DE-FG02-08ER25856.

‡cliff@ieor.columbia.edu. Department of Industrial Engineering & Operations Research, Columbia University, Mudd 326, 500W 120th Street, New York, NY 10027. Research partially supported by NSF grants CCF-0728733 and CCF-0915681.

1 Introduction

Managing energy consumption is a problem of critical interest throughout the world and throughout various industries. Computing devices use a large amount of energy, both in individual devices such as laptops and PDAs and also in large industrial uses such as datacenters. For example, Google states that the servers in its datacenter, which are much more efficient than the average industry server, consume 1kJ per query on average [1]. In January 2011, just in the US, there were an average more than 400 million queries per day [2], and thus the total amount of energy consumed was 44.5 million kWh, equivalent to more than 4,000 average US households [3]. Furthermore, CPUs account for 50-60% of a typical computer's energy consumption [4], making CPU energy management very important. When scheduling on such devices, it is important not only to consider the relevant quality of service (QoS) metrics such as makespan or weighted completion time, but also to take energy consumption into account. Most modern CPUs can be run at multiple speeds; the lower the speed, the less energy used, and the relationship is device-dependent, but typically superlinear. The technique of scheduling while controlling the speed of the processor is known as *speed scaling*.

Starting with the work of Yao, Demers, and Shenker [25], there has by now been tens of papers studying scheduling problems in which energy consumption is taken into account. (See, for example, the surveys by Irani and Pruhs [19] and that by Albers [4]). There are three main settings for energy aware scheduling problem: optimizing a QoS metric with an energy budget [22, 23], minimizing energy subject to a QoS constraint [7, 10, 11, 25], or optimizing some convex combination of a scheduling objective and energy consumption [5, 6, 9, 12]. Underlying the latter setting, which is in the one we will focus in this work, is an assumption that both energy and time can be (implicitly) converted into a common unit, such as dollars.

1.1 Our results

In this paper we consider two commonly studied scheduling metrics, *weighted completion time* and *weighted tardiness*, that have not received attention in the energy aware scheduling literature. Given a schedule in which job i with weight w_i , release time r_i , and deadline d_i is completed at time C_i , the total weighted completion time is $\sum_i w_i C_i$. The tardiness of a job is zero if it is completed before its deadline and otherwise equal to the amount by which it misses, that is, $T_i = \max\{0, C_i - d_i\}$ and total weighted tardiness is $\sum w_i T_i$. For both these metrics, we consider the non-preemptive, off-line problem on one machine, and allow arbitrary precedence constraints. For the weighted completion time we allow arbitrary release dates as well. We consider a metric that is a convex combination of our scheduling metric and energy cost. We are not aware of any previous work on energy aware scheduling algorithms for these metrics. There is a rich literature on minimizing weighted completion time in the absence of energy concerns (e.g. [20, 21, 24]), but we are aware of only one result about weighted tardiness in the absence of energy concerns in the speed scaling/resource augmentation literature [8], where a 2-machine, 24-speed 4-approximation algorithm is presented. Weighted tardiness, in particular, is difficult to analyze because, in contrast to most scheduling objectives, it is a non-linear function of completion time.

In our work we consider a more general model of energy cost than has previously been used. The most common energy model assumes that the rate at which power is consumed is a polynomial function of speed of the form $P(s) = s^\beta$ for some constant β ; typical values of β are 2 or 3. Some recent work[6, 9] uses a more general power function with minimum regularity conditions, like non-negativity, but in all the cases the power function does not depend on the job. Furthermore, most energy aware algorithms assume cost functions that are closely related to energy consumption; however, in practice the actual energy cost is not simply a function of energy consumption, it is a complicated function of discounts, pricing, time of consumption, etc. We consider a more

general class of cost functions that are only restricted to be non-negative and can be different for different jobs. Because we allow job-dependent energy costs, our algorithms can be used outside the CPU-energy setting, where energy cost generally are job independent, and can be applied to more general problems that have additional speed-associated costs. Examples of these costs are maintenance costs, wear and tear of parts, failure rates, etc. all of which not only depend on the speed at which the machine runs, but also the job being processed. We are not aware of any other work that allows such general costs. For the weighted tardiness case we require an additional regularity condition on the energy cost functions that allows us to control its rate of growth.

Our paper contains several results for different scheduling problems, we state here the most general results:

Theorem 1.1. *Given n jobs with precedence constraints and release dates and a general non-negative energy cost function, there is an $O(1)$ -approximation algorithm for the problem of non-preemptively minimizing a convex combination of weighted completion time and energy cost.*

Theorem 1.2. *Given n jobs with precedence constraints and deadlines and a general non-negative energy cost function, there is an $O(1)$ -approximation algorithm for the problem of non-preemptively minimizing a convex combination of weighted tardiness and energy cost.*

The constants in the $O(1)$ are modest. Consider the case where we are given a set of speeds $\mathbf{S} = \{\sigma_1, \dots, \sigma_m\}$, at which the machine can run, with $\sigma_j \leq (1 + \delta)\sigma_{j-1}$, and some $\epsilon > 0$. Then the algorithm for the weighted completion time setting has a $4(1 + \epsilon)(1 + \delta)$ -approximation ratio when only precedence constraints exist, and $(3 + 2\sqrt{2})(1 + \epsilon)(1 + \delta)$ -approximation ratio when release dates are added. The algorithm for the weighted tardiness setting has a $4^\beta(1 + \epsilon)^{\beta-1}(1 + \delta)^{\beta-1}$ -approximation ratio even with arbitrary precedence constraints, where β controls the growth of the energy cost function.

1.2 Our Methodology

The problem of minimizing weighted completion time in the combinatorial setting has been well-studied. The work of Phillips, Stein, and Wein [20] and Hall, Schulz, Shmoys, and Wein [17, 18] introduced the idea of α -points, and these have been used in much of the subsequent work. The idea is that one first formulates a time-indexed integer program in which decision variable x_{it} is 1 if job i completes at time t , and then solves its linear programming relaxation. From the solution to the relaxation, one computes the α -point of each job, that is, the earliest time at which an α fraction of the job has completed in the relaxation. The exact interpretation of when an α fraction completes depends upon the particular problem. One uses these α -points to infer an order on the jobs and then runs the jobs non-preemptively, respecting that order. There are many variants and extensions of these technique including choosing α randomly [13, 14] or choosing a different α for each job [15]. This technique has led to small constant factor approximation algorithms for many weighted completion time scheduling problems [24].

The time-indexed integer program (IP) formulations for this problem are not typically of polynomial size. However, the *interval-indexed* IP, introduced in [18], in which time is divided into geometrically increasing intervals and jobs are assigned to intervals rather than individual time slots, is of polynomial size. By using this linear program one obtains a polynomial sized linear program from which it is still possible to apply the ideas of α -points while suffering only a small additional degradation of the approximation ratio.

In this paper, we extend the interval-indexed IP to handle speed scaling and then design new α -point based rounding algorithms to obtain the resulting schedules. In doing so we introduce the new concept of α -speeds. We assume, in Sections 2, 3, and 4, that we have a discrete set of m

allowable speeds $\mathbf{S} = \{\sigma_j\}$, and that the rate of power consumption is a polynomial function of the speed. In Section 5.3 we describe how to remove these assumptions. Although the time-indexed IPs are easier to explain, due to limited space, we will describe only the interval-indexed linear programs in this paper. In our interval-indexed IP, a variable x_{ijt} is 1 if job i runs at speed σ_j and completes in interval t . We can then extend the standard interval-indexed integer programming formulation to take the extra dimension of speed into account (see Section 2 for details). Once we have solved its linear program (LP) relaxation, we need to now determine *both an α -point and α -speed*. The key insight is that by “summarizing” each dimension appropriately, we are able to make the correct choice for the other dimension. At a high level, we first choose the α -point by “collapsing” all pieces of a job that complete in the LP in interval t (these pieces have different speeds), being especially careful with the last interval, where we may have to choose only some of the speeds. We then use *only* the pieces of the job that complete before the α -point to choose the speed, where the speed is chosen by collapsing the time dimension and then interpreting the result as a probability mass function (pmf), where the probability that the job is run at speed σ_j depends on the total amount of processing done at that speed. We then define the concept of α -speeds, which is related to the expected value under this pmf, and run the job at this speed (see Section 3 for more details). We combine this new rounding method with extensions of the more traditional methods for dealing with precedence constraints and release dates to obtain our algorithms.

For weighted tardiness, we emphasize again that not much is known about approximating this problem, even in the absence of energy concerns. For this problem, we are able to use the same interval-indexed linear program, with the objective function modified to tardiness. Because the linear program is interval indexed, the non-linear objective function is not a problem. After the solving the linear program, we are able to show that with only a constant factor increase in energy (over the lower bound from the linear program), we obtain only a constant factor (over the linear program) increase in tardiness. Implicit in this analysis is the fact that jobs that receive 0 tardiness in the linear program will receive 0 tardiness in our solution; in some sense the speed scaling makes accomplishing this easier than in the combinatorial setting. We note that our weighted tardiness algorithms does not work in the presence of release dates, as release dates may stop us from being able to keep jobs with 0 tardiness in the LP at 0 tardiness in the schedule.

Finally, in Section 5, we show how to extend our results for the weighted completion and weighted tardiness scheduling metrics to general energy cost functions. We also show how to extend our results to the setting where continuous speeds are used and not just a discrete set \mathbf{S} , while maintaining the same approximation ratio.

2 Problem Formulation

2.1 Problem Setting

We are given n jobs, where job i has a processing requirement of $\rho_i \in \mathbb{N}_+$ machine cycles, release time r_i , and an associated positive weight w_i . Let s_i denote the speed at which job i runs on the machine and C_i denote its completion time. Let $\Pi = \{\pi(1), \dots, \pi(n)\}$ denote the order in which the jobs are processed, i.e. $\pi(k) = i$ implies that job i is the k -th job to be processed. Then $C_{\pi(i)} = \max\{r_{\pi(i)}, C_{\pi(i-1)}\} + \frac{\rho_{\pi(i)}}{s_{\pi(i)}}$ is the completion time of the i -th job to be processed, with $C_{\pi(0)} = 0$. We do not allow preemption.

Let $\mathbf{S} = \{\sigma_1, \dots, \sigma_m\}$, be the set of possible speeds at which the machine can run. We will assume that $\sigma_{j+1} \leq (1 + \delta)\sigma_j$, for some $\delta > 0$. This is a natural assumption because actual speed scaling achieved in CPUs is done via frequency multipliers or dividers. Although a discrete set of speeds is probably the most common case for CPUs, in Section 5.3 we show that our algorithm has the same approximation ratio when a continuous set of speeds is used.

Let $E_i(s_i)$ denote the energy cost of running job i at speed s_i . For simplicity we initially consider $E_i(s_i) = v_i \rho_i s_i^{\beta-1}$, where $\beta \geq 2$ and v_i are known constants. As indicated earlier, an energy cost function of this form is the standard model for these problems, although our model is more general because the energy cost function is job-dependent. In Section 5 we show that our algorithms also work for a much larger class of job-dependent energy cost functions.

The objective is to compute a feasible schedule (Π, \mathbf{C}) , consisting of an order Π and completion times \mathbf{C} , possibly subject to precedence and/or release date constraints, and the vector of job speeds $\mathbf{s} = \{s_1, \dots, s_n\} \in \mathbb{R}_+^n$ that minimizes the total cost,

$$f(\Pi, \mathbf{s}) = \sum_{i=1}^n \left[v_i \rho_i s_i^{\beta-1} + w_{\pi(i)} C_{\pi(i)} \right], \quad (2.1)$$

Since this function is convex we can assume, w.l.o.g., that each job runs at a constant speed.

For convenience we will use an extended version of the notation of Graham et al. [16] to refer to the different energy aware scheduling problems, i.e. $1|r_i, prec| \sum E_i(s_i) + \sum w_i C_i$, will refer to the problem setting with 1 machine, with r_i release dates, precedence constraints, and the weighted completion time as the scheduling performance metric. Similarly, the $1|r_i, prec| \sum E_i(s_i) + \sum w_i T_i$ will refer to the same setting, but with tardiness as the scheduling performance metric. In all of them $E_i(s_i)$ indicates that the energy cost is also added as a performance metric.

2.2 Interval-Indexed Formulation

We now modify and extend the interval-indexed formulation proposed by Hall et al. [18] to accommodate speeds and energy cost.

The interval-indexed formulation divides the time horizon into geometrically increasing intervals, and the completion time of each job is assigned to one of these intervals. Since the completion times are not associated to a specific time, the completion times are not precisely known but are lower bounded. By controlling the growth of each interval one can obtain a sufficiently tight bound.

The problem formulation is as follows. We divide the time horizon into the following geometrically increasing intervals: $[\kappa, \kappa]$, $(\kappa, (1 + \epsilon)\kappa]$, $((1 + \epsilon)\kappa, (1 + \epsilon)^2\kappa]$, \dots , where $\epsilon > 0$ is an arbitrary small constant, and $\kappa = \frac{\rho_{\min}}{\sigma_{\max}}$ denotes the smallest interval size that will hold at least one whole job. We define interval $I_t = (\tau_{t-1}, \tau_t]$, with $\tau_0 = \kappa$ and $\tau_t = \kappa(1 + \epsilon)^{t-1}$. The interval index ranges over $\{1, \dots, T\}$, with $T = \min\{t : \kappa(1 + \epsilon)^{t-1} \geq \max_{i=1}^n r_i + \sum_{i=1}^n \frac{\rho_i}{\sigma_i}\}$; and thus, we have a polynomial number of indices t .

Let

$$x_{ijt} = \begin{cases} 1, & \text{if job } i \text{ runs at a speed } \sigma_j \text{ and completes in the time interval } I_t = (\tau_{t-1}, \tau_t] \\ 0, & \text{otherwise} \end{cases} \quad (2.2)$$

By using the lower bounds τ_{t-1} of each time interval I_t , a lower bound to (2.1) is written as,

$$\min_{\mathbf{x}} \sum_{i=1}^n \sum_{j=1}^m \sum_{t=1}^T \left(v_i \rho_i \sigma_j^{\beta-1} + w_i \tau_{t-1} \right) x_{ijt}. \quad (2.3)$$

The following are the constraints required for the $1|r_i, prec| \sum E_i(s_i) + \sum w_i C_i$ problem:

1. Each job must finish in a unique time interval and speed; therefore for $i = \{1, \dots, n\}$:

$$\sum_{j=1}^m \sum_{t=1}^T x_{ijt} = 1. \quad (2.4)$$

2. Since only one job can be processed at any given time, the total processing time of jobs up to time interval I_t must be at most τ_t units. Thus, for $t = \{1, \dots, T\}$:

$$\sum_{i=1}^n \sum_{j=1}^m \sum_{u=1}^t \frac{\rho_i}{\sigma_j} x_{iju} \leq \tau_t. \quad (2.5)$$

3. Job i running at speed σ_j requires $\frac{\rho_i}{\sigma_j}$ time units to be processed, and considering that its release time is r_i , then for $i = \{1, \dots, n\}$, $j = \{1, \dots, m\}$, and $t = \{1, \dots, T\}$:

$$x_{ijt} = 0, \quad \text{if } \tau_t < r_i + \frac{\rho_i}{\sigma_j}. \quad (2.6)$$

4. For $i = \{1, \dots, n\}$ and $t = \{1, \dots, T\}$:

$$x_{it} \in \{0, 1\}. \quad (2.7)$$

5. The precedence constraint $i_1 \prec i_2$ implies that job i_2 cannot finish in an interval earlier than i_1 . Therefore for every $i_1 \prec i_2$ constraint we have that for $t = \{1, \dots, T\}$:

$$\sum_{j=1}^m \sum_{u=1}^t x_{i_1ju} \geq \sum_{j=1}^m \sum_{u=1}^t x_{i_2ju}. \quad (2.8)$$

It is important to note that this integer program only provides a lower bound for (2.1); in fact its optimal solution may not be schedulable, since constraints (2.5) do not imply that only one job can be processed at a single time, they only bound the total amount of work in $\cup_t I_t$.

3 Approximation Algorithm for Weighted Completion Time

We now describe the approximation algorithm for the weighted completion time, called SCHEDULE BY α -INTERVALS AND α -SPEEDS (SAIAS) which is displayed in Figure 3.1.

Let \bar{x}_{ijt} denote the optimal solution of the linear relaxation of the integer program (2.3)-(2.8), in which we change constraints (2.7) for $x_{ijt} \geq 0$. In step 1 of the algorithm we compute the optimal solution $\bar{\mathbf{x}}$ and in step 2, given $0 \leq \alpha \leq 1$, we compute the α -interval of job i , which is defined as,

$$\tau_i^\alpha = \min \left\{ \tau : \sum_{j=1}^m \sum_{u=1}^{\tau} \bar{x}_{iju} \geq \alpha \right\}. \quad (3.1)$$

SCHEDULE BY α -INTERVALS AND α -SPEEDS (SAIAS)

- Inputs:** set of jobs, $\alpha \in (0, 1)$, $\epsilon > 0$, set of speeds $\mathbf{S} = \{\sigma_1, \dots, \sigma_m\}$.
- 1 Compute an optimal solution $\bar{\mathbf{x}}$ to the linear relaxation (2.3)-(2.8).
 - 2 Compute the α -intervals τ^α and the sets J_t .
 - 3 Compute an order Π^α that has the sets J_t ordered in non-decreasing values of t and the jobs within each set in a manner consistent with the precedence constraints.
 - 4 Compute the α -speeds \mathbf{s}^α
 - 5 Round down each s_i^α to the nearest speed in \mathbf{S} and run job i at this rounded speed, \bar{s}_i^α .
 - 6 Set the i -th job to start at time $\max\{r_{\pi(i)}, \bar{C}_{\pi(i-1)}^\alpha\}$, where $\bar{C}_{\pi(i-1)}^\alpha$ is the completion time of the previous job using the rounded α -speeds, and $\bar{C}_{\pi(0)}^\alpha = 0$.
 - 7 **return** speeds $\bar{\mathbf{s}}^\alpha$ and schedule $(\Pi^\alpha, \bar{\mathbf{C}}^\alpha)$.

Figure 3.1: Schedule by α -intervals and α -speeds

Since several jobs may finish in the same interval, let J_t denote the set of jobs that finish in interval I_t , $J_t = \{i : \tau_i^\alpha = t\}$, and we use these sets to determine the order Π^α as described in step 3.

Next, in step 4, we compute the α -speeds as follows. Since $\sum_{j=1}^m \sum_{u=1}^{\tau_i^\alpha} \bar{x}_{iju} \geq \alpha$, we define auxiliary variable $\{\tilde{x}_{ijt}\}$ as:

$$\tilde{x}_{ijt} = \begin{cases} \bar{x}_{ijt}, & t < \tau_i^\alpha \\ \max \left\{ \min \left\{ \bar{x}_{ij\tau_i^\alpha}, \alpha - \sum_{l=1}^{j-1} \bar{x}_{il\tau_i^\alpha} - \beta_i \right\}, 0 \right\}, & t = \tau_i^\alpha \\ 0, & t > \tau_i^\alpha \end{cases}, \quad (3.2)$$

where $\beta_i = \sum_{j=1}^m \sum_{u=1}^{\tau_i^\alpha - 1} \bar{x}_{iju} < \alpha$. Note that with this auxiliary variable $\sum_{j=1}^m \sum_{u=1}^{\tau_i^\alpha} \tilde{x}_{iju} = \alpha$. This is a key step that allows us to truncate the fractional solution so that for every job i , the sum of \tilde{x}_{ijt} up to time interval τ_i^α for each speed j can be interpreted as a probability mass function. We define this probability mass function (pmf) $\mu_i = (\mu_{i1}, \dots, \mu_{im})$ on the set of speeds $\mathbf{S} = \{\sigma_1, \dots, \sigma_m\}$ as

$$\mu_{ij} = \frac{1}{\alpha} \sum_{u=1}^{\tau_i^\alpha} \tilde{x}_{iju}. \quad (3.3)$$

Let \hat{s}_i define a random variable distributed according to the pmf μ_i , i.e. $\mu_{ij} = \mathbb{P}(\hat{s}_i = \sigma_j)$. Then, the α -speed of job i , s_i^α , is defined as follows:

$$\frac{1}{s_i^\alpha} = \mathbb{E} \left[\frac{1}{\hat{s}_i} \right] = \sum_{j=1}^m \frac{\mu_{ij}}{\sigma_j} \Rightarrow s_i^\alpha = \frac{1}{\mathbb{E} \left[\frac{1}{\hat{s}_i} \right]}. \quad (3.4)$$

We define the α -speeds using the reciprocal of the speeds since the completion times are proportional to the reciprocals instead of the speeds, and we need to bound completion times in the analysis of the algorithm.

Next, in step 5, because the α -speeds s_i^α do not necessarily belong to the set of possible speeds \mathbf{S} we round them down to \bar{s}_i^α , which is the nearest speed in the set such that $\bar{s}_i^\alpha \leq s_i^\alpha$. The following lemma bounds the error introduced by this rounding.

Lemma 3.1. *The cost of the solution with the rounded down speeds $\bar{\mathbf{s}}^\alpha$ is at most $(1 + \delta)$ times the cost of the solution using the α -speeds \mathbf{s}^α .*

Proof. The energy cost function $E_i(s_i)$ is increasing so rounding down does not increase the energy cost, but the completion time is now larger. Let C_i^α be the completion time of job i when the speeds \mathbf{s}^α are used and \bar{C}_i^α when the rounded ones $\bar{\mathbf{s}}^\alpha$ are used. Since the speeds are reduced at most by $(1 + \delta)$, then $(1 + \delta)\bar{s}_i^\alpha \geq s_i^\alpha$, and we have that,

$$\bar{C}_i^\alpha = \max\{r_i, \bar{C}_{i-1}^\alpha\} + \frac{\rho}{\bar{s}_i^\alpha} \leq (1 + \delta) \left(\max\{r_i, C_{i-1}^\alpha\} + \frac{\rho}{s_i^\alpha} \right) = (1 + \delta)C_i^\alpha, \quad (3.5)$$

which implies that $\sum_{i=1}^n w_i \bar{C}_i^\alpha \leq (1 + \delta) \sum_{i=1}^n w_i C_i^\alpha$ and proves the lemma. \square

Finally, in steps 6 and 7 we compute the completion times given the calculated speeds and return the set of speeds $\bar{\mathbf{s}}^\alpha$ and the schedule $(\Pi^\alpha, \bar{\mathbf{C}}^\alpha)$.

We now analyse this algorithm's performance for different energy aware scheduling problems. In the following subsections we will assume w.l.o.g. that $\tau_1^\alpha \leq \tau_2^\alpha \leq \dots \tau_n^\alpha$.

3.1 Single Machine Problem with Precedence Constraints

We first need to prove that the output of the SAIAS algorithm is indeed feasible.

Lemma 3.2. *If $i_1 \prec i_2$, then constraint (2.8) implies that $\tau_{i_1}^\alpha \leq \tau_{i_2}^\alpha$.*

Proof. Evaluating the LP constraint (2.8) corresponding to $i_1 \prec i_2$, for $t = \tau_{i_2}^\alpha$, we have that,

$$\sum_{j=1}^m \sum_{u=1}^{\tau_{i_2}^\alpha} x_{i_1ju} \geq \sum_{j=1}^m \sum_{u=1}^{\tau_{i_2}^\alpha} x_{i_2ju} \geq \alpha,$$

where the last inequality follows from the definition of $\tau_{i_2}^\alpha$. The chain of inequalities implies that $\sum_{j=1}^m \sum_{u=1}^{\tau_{i_1}^\alpha} x_{i_1ju} \geq \alpha$, so $\tau_{i_1}^\alpha \leq \tau_{i_2}^\alpha$. \square

Since the SAIAS algorithm schedules jobs by first ordering the sets J_t in increasing order of t , and then orders the jobs within each set in a way that is consistent with the precedence constraints, by Lemma 3.2 it follows that the SAIAS algorithm preserves the precedence constraints, and, therefore, the output of the algorithm is feasible. Next, we can prove the following result.

Theorem 3.1. *The SAIAS algorithm with $\alpha = \frac{1}{2}$ is a $4(1 + \epsilon)(1 + \delta)$ -approximation algorithm for the $1|prec|\sum E_i(s_i) + \sum w_i C_i$ problem, with $E_i(s_i) = v_i \rho_i s_i^{\beta-1}$.*

Proof. Let x_{ijt}^* denote an optimal solution to the integer problem (2.3)-(2.8), \bar{x}_{ijt} the fractional solution of its linear relaxation, and \tilde{x}_{iju} the auxiliary variables calculated for the SAIAS algorithm.

Since in (2.3) the completion time for jobs completed in interval I_t is τ_{t-1} , it follows that,

$$\sum_{i=1}^n \sum_{j=1}^m \sum_{t=1}^T \left(v_i \rho_i \sigma_j^{\beta-1} + w_i \tau_{t-1} \right) \bar{x}_{ijt} \leq \sum_{i=1}^n \sum_{j=1}^m \sum_{t=1}^T v_i \rho_i \sigma_j^{\beta-1} x_{ijt}^* + \sum_{i=1}^n w_i C_i^*. \quad (3.6)$$

The energy terms of the algorithm's solution are bounded as follows,

$$\begin{aligned} v_i \rho_i (s_i^\alpha)^{\beta-1} &= v_i \rho_i \left(\frac{1}{s_i^\alpha} \right)^{-(\beta-1)} = v_i \rho_i \left(\mathbb{E} \left[\frac{1}{\hat{s}_i} \right] \right)^{-(\beta-1)} \\ &\leq v_i \rho_i \mathbb{E} \left[\left(\frac{1}{\hat{s}_i} \right)^{-(\beta-1)} \right] = v_i \rho_i \mathbb{E} \left[\hat{s}_i^{\beta-1} \right] = v_i \rho_i \sum_{j=1}^m \mu_{ij} \sigma_j^{\beta-1}, \end{aligned} \quad (3.7)$$

where the inequality follows from Jensen's Inequality applied to the convex function $\frac{1}{s^{\beta-1}}$. Using the definition of μ_{ij} in (3.3) and given that $0 \leq \alpha \leq 1$, $\epsilon > 0$, and $\tilde{x}_{ijt} \leq \bar{x}_{ijt}$, it follows that,

$$v_i \rho_i (s_i^\alpha)^{\beta-1} \leq \frac{v_i \rho_i}{\alpha} \sum_{j=1}^m \sum_{u=1}^{\tau_i^\alpha} \sigma_j^{\beta-1} \tilde{x}_{iju} \leq \frac{(1 + \epsilon)}{\alpha(1 - \alpha)} v_i \rho_i \sum_{j=1}^m \sum_{u=1}^{\tau_i^\alpha} \sigma_j^{\beta-1} \bar{x}_{iju}. \quad (3.8)$$

Since there are no release date constraints there is no idle time between jobs,

$$C_i^\alpha = \sum_{j=1}^i \frac{\rho_j}{s_j^\alpha} = \sum_{j=1}^i \rho_j \mathbb{E} \left[\frac{1}{\hat{s}_j} \right] = \frac{1}{\alpha} \sum_{j=1}^i \sum_{l=1}^m \sum_{u=1}^{\tau_j^\alpha} \frac{\rho_j}{\sigma_l} \tilde{x}_{jlu} \leq \frac{1}{\alpha} \sum_{j=1}^n \sum_{l=1}^m \sum_{u=1}^{\tau_j^\alpha} \frac{\rho_j}{\sigma_l} \bar{x}_{jlu}, \quad (3.9)$$

and from constraint (2.5) for $t = \tau_i^\alpha$ we get, $C_i^\alpha \leq \frac{1}{\alpha} \tau_{\tau_i^\alpha}$.

Let $\bar{C}_i = \sum_{j=1}^m \sum_{t=1}^T \tau_{t-1} \bar{x}_{ijt}$ denote the optimal fractional completion time given by the optimal solution of the relaxed linear program (2.3)-(2.6). Since it is possible that $\sum_{j=1}^m \sum_{t=1}^{\tau_i^\alpha} \bar{x}_{ijt} > \alpha$; we define $X_i^{(1)} = \alpha - \sum_{j=1}^m \sum_{t=1}^{\tau_i^\alpha - 1} \bar{x}_{ijt}$ and $X_i^{(2)} = \sum_{j=1}^m \sum_{t=1}^{\tau_i^\alpha} \bar{x}_{ijt} - \alpha$, thus $X_i^{(1)} + X_i^{(2)} = \sum_{j=1}^m \bar{x}_{ij\tau_i^\alpha}$, and we can rewrite

$$\bar{C}_i = \sum_{j=1}^m \sum_{t=1}^{\tau_i^\alpha - 1} \tau_{t-1} \bar{x}_{ijt} + \tau_{\tau_i^\alpha - 1} X_i^{(1)} + \tau_{\tau_i^\alpha - 1} X_i^{(2)} + \sum_{j=1}^m \sum_{t=\tau_i^\alpha + 1}^T \tau_{t-1} \bar{x}_{ijt}, \quad (3.10)$$

and eliminating the lower terms of the previous sum we get that,

$$\bar{C}_i \geq \tau_{\tau_i^\alpha - 1} X_i^{(2)} + \sum_{j=1}^m \sum_{t=\tau_i^\alpha + 1}^T \tau_{t-1} \bar{x}_{ijt} \geq \tau_{\tau_i^\alpha - 1} X_i^{(2)} + \sum_{j=1}^m \sum_{t=\tau_i^\alpha + 1}^T \tau_{\tau_i^\alpha - 1} \bar{x}_{ijt} = \tau_{\tau_i^\alpha - 1} (1 - \alpha). \quad (3.11)$$

Because $\tau_{\tau_i^\alpha} = (1 + \epsilon) \tau_{\tau_i^\alpha - 1}$, from (3.9) and (3.11) we get that $C_i^\alpha \leq \frac{(1+\epsilon)}{\alpha(1-\alpha)} \bar{C}_i \Rightarrow \sum_{i=1}^n w_i C_i^\alpha \leq \frac{(1+\epsilon)}{\alpha(1-\alpha)} \sum_{i=1}^n w_i \bar{C}_i$. From this, (3.6) and (3.8) it follows that,

$$\sum_{i=1}^n v_i \rho_i (s_i^\alpha)^{\beta-1} + \sum_{i=1}^n w_i C_i^\alpha \leq \frac{(1+\epsilon)}{\alpha(1-\alpha)} \left[\sum_{i=1}^n \sum_{j=1}^m \sum_{t=1}^T v_i \rho_i \sigma_j^{\beta-1} x_{ijt}^* + \sum_{i=1}^n w_i C_i^* \right], \quad (3.12)$$

and we set $\alpha = \arg \min_{0 \leq \alpha \leq 1} \left\{ \frac{1}{\alpha(1-\alpha)} \right\} = \frac{1}{2}$, to minimize the bound. By Lemma 3.1, which bounds the final rounding error, we get the desired approximation ratio. \square

3.2 Single Machine Problem with Precedence and Release Date Constraints

We now analyse the case with precedence constraints and release dates. Release dates makes the problem somewhat harder since they can introduce idle times between jobs.

Theorem 3.2. *The SAIAS algorithm with $\alpha = \sqrt{2} - 1$ is a $(3 + 2\sqrt{2})(1 + \epsilon)(1 + \delta)$ -approximation algorithm for the $1|r_i, \text{prec}|\sum E_i(s_i) + \sum w_i C_i$ problem, with $E_i(s_i) = v_i \rho_i s_i^{\beta-1}$.*

Proof. The bound for the energy terms computed in equation (3.7) are still valid when there is idle time between jobs, we have that,

$$v_i \rho_i (s_i^\alpha)^{\beta-1} \leq \frac{(1+\epsilon)}{\alpha(1-\alpha)} v_i \rho_i \sum_{j=1}^m \sum_{u=1}^T \sigma_j^{\beta-1} \bar{x}_{iju} \leq \frac{(1+\epsilon)(1+\alpha)}{\alpha(1-\alpha)} v_i \rho_i \sum_{j=1}^m \sum_{u=1}^T \sigma_j^{\beta-1} \bar{x}_{iju}. \quad (3.13)$$

When bounding the completion time C_i^α , given the sorting done in step 3 of the SAIAS algorithm, now one has to consider all the jobs up to the ones in set $J_{\tau_i^\alpha}$, and thus,

$$C_i^\alpha \leq \max_{j \in \{J_1, \dots, J_{\tau_i^\alpha}\}} r_j + \sum_{j \in \{J_1, \dots, J_{\tau_i^\alpha}\}} \frac{\rho_j}{s_j^\alpha}. \quad (3.14)$$

Since all jobs that have been at least partially processed up to time interval I_t need to be released before τ_t , it follows that $\max_{j \in \{J_1, \dots, J_{\tau_i^\alpha}\}} r_j \leq \tau_{\tau_i^\alpha}$. On the other hand, we also have that,

$$\sum_{j \in \{J_1, \dots, J_{\tau_i^\alpha}\}} \frac{\rho_j}{s_j^\alpha} = \frac{1}{\alpha} \sum_{j \in \{J_1, \dots, J_{\tau_i^\alpha}\}} \sum_{l=1}^m \sum_{u=1}^{\tau_j^\alpha} \frac{\rho_j}{\sigma_l} \tilde{x}_{jlu} \leq \frac{1}{\alpha} \sum_{j=1}^n \sum_{l=1}^m \sum_{u=1}^{\tau_j^\alpha} \frac{\rho_j}{\sigma_l} \bar{x}_{jlu} \leq \frac{1}{\alpha} \tau_{\tau_i^\alpha}, \quad (3.15)$$

where the last inequality follows from constraint (2.5) with $t = \tau_i^\alpha$. Thus, $C_i^\alpha \leq \frac{(1+\alpha)}{\alpha} \tau_i^\alpha$. Since $\bar{C}_i = \sum_{j=1}^m \sum_{t=1}^T \tau_{t-1} \bar{x}_{ijt}$, (3.11) is still valid and because $\tau_{\tau_i^\alpha} = (1+\epsilon)\tau_{\tau_i^\alpha-1}$, we get,

$$C_i^\alpha \leq \frac{(1+\epsilon)(1+\alpha)}{\alpha(1-\alpha)} \bar{C}_i \Rightarrow \sum_{i=1}^n w_i C_i^\alpha \leq \frac{(1+\epsilon)(1+\alpha)}{\alpha(1-\alpha)} \sum_{i=1}^n w_i \bar{C}_i. \quad (3.16)$$

Finally, from (3.13) and (3.16) it follows that,

$$\sum_{i=1}^n v_i \rho_i (s_i^\alpha)^{\beta-1} + \sum_{i=1}^n w_i C_i^\alpha \leq \frac{(1+\epsilon)(1+\alpha)}{\alpha(1-\alpha)} \left[\sum_{i=1}^n \sum_{j=1}^m \sum_{t=1}^T v_i \rho_i \sigma_j^{\beta-1} x_{ijt}^* + \sum_{i=1}^n w_i C_i^* \right], \quad (3.17)$$

and by setting $\alpha = \arg \min_{0 \leq \alpha \leq 1} \left\{ \frac{(1+\alpha)}{\alpha(1-\alpha)} \right\} = \sqrt{2} - 1$, and again using Lemma 3.1 to bound the speed-rounding error, we get the required approximation ratio. \square

If no precedence constraints and release dates exist, there are two versions of this problem that can be optimally solved in polynomial time: when all weights w_i are equal, and when all jobs are of the same size (i.e. $\rho_i = \rho, \forall i$) and all jobs have the same energy cost function. For these cases we have the following result:

Theorem 3.3. *If $w_i = w, \forall i$ or $\rho_i v_i^{\frac{1}{\beta}} = \xi, \forall i$ then the order Π is optimal if*

$$\frac{w_{\pi(i)}}{\rho_{\pi(i)} v_{\pi(i)}^{\frac{1}{\beta}}} \geq \frac{w_{\pi(i+1)}}{\rho_{\pi(i+1)} v_{\pi(i+1)}^{\frac{1}{\beta}}}, \quad \forall i \in \{1, \dots, n-1\}.$$

Proof. For simplicity we will define $\xi_i \equiv \rho_i v_i^{\frac{1}{\beta}}, q = \frac{k-1}{k}$, and $\mathcal{K} \equiv \frac{k}{(k-1)^{\frac{k-1}{k}}}$. First, dual formulation of problem (2.1) with no precedence or release date constraints is given by,

$$\min_{\pi} F(\pi) = \min_{\pi} \sum_{i=1}^n \mathcal{K} \xi_{\pi(i)} \left(\sum_{j=i}^n w_{\pi(j)} \right)^q. \quad (3.18)$$

We now prove both cases by contradiction using the dual formulation.

When $w_i = w, \forall i$, Theorem 3.3 implies that in the optimal order $\xi_{\pi(i+1)} \geq \xi_{\pi(i)}$. By contradiction, let π be an optimal order such that for some index $k, \xi_{\pi(k+1)} < \xi_{\pi(k)}$. For this order the total cost is

$$\begin{aligned} F(\pi) &= \sum_{i=1}^n \mathcal{K} \xi_{\pi(i)} \left(\sum_{j=i}^n w \right)^q = \sum_{i=1}^n \mathcal{K} \xi_{\pi(i)} ((n-i+1)w)^q, \\ &= \mathcal{K} w^q \left\{ \xi_{\pi(k)} (n-k+1)^q + \xi_{\pi(k+1)} (n-k)^q + \sum_{i=1; i \neq k, k+1}^n \xi_{\pi(i)} (n-i+1)^q \right\}. \end{aligned}$$

Let π_k define the order where we switch jobs k and $k+1$ from order π , i.e. $\pi_k(k) = \pi(k+1)$ and $\pi_k(k+1) = \pi(k)$. Given this order we have that

$$\begin{aligned} F(\pi) - F(\pi_k) &= \mathcal{K} w^q \left\{ \xi_{\pi(k)} (n-k+1)^q + \xi_{\pi(k+1)} (n-k)^q - \xi_{\pi(k+1)} (n-k+1)^q - \xi_{\pi(k)} (n-k)^q \right\}, \\ &= \mathcal{K} w^q \left\{ (n-k+1)^q (\xi_{\pi(k)} - \xi_{\pi(k+1)}) - (n-k)^q (\xi_{\pi(k)} - \xi_{\pi(k+1)}) \right\}, \\ &= \mathcal{K} w^q \left\{ (\xi_{\pi(k)} - \xi_{\pi(k+1)}) ((n-k+1)^q - (n-k)^q) \right\}. \end{aligned}$$

By our initial assumption the first term is positive (since $\xi_{\pi(k+1)} < \xi_{\pi(k)}$) and the second one is always positive, hence $F(\pi) - F(\pi_k) > 0$ which is a contradiction, since that implies that π_k has a smaller cost.

For the case when $\xi_i = \xi$, $\forall i$, Theorem 3.3 implies that an order π is optimal then $w_{\pi(i)} \geq w_{\pi(i+1)}$. Let π be an optimal order such that for some index k , $w_{\pi(k)} < w_{\pi(k+1)}$. The total cost for this solution is

$$\begin{aligned} F(\pi) &= \sum_{i=1}^n \mathcal{K}\xi \left(\sum_{j=i}^n w_{\pi(j)} \right)^q = \mathcal{K}\xi \left\{ \sum_{i=1}^k \left(\sum_{j=i}^n w_{\pi(j)} \right)^q + \left(\sum_{j=k+1}^n w_{\pi(j)} \right)^q + \sum_{i=k+2}^n \left(\sum_{j=i}^n w_{\pi(j)} \right)^q \right\}, \\ &= \mathcal{K}\xi \left\{ \sum_{i=1}^k \left(\sum_{j=i}^n w_{\pi(j)} \right)^q + \left(w_{\pi(k+1)} + \sum_{j=k+2}^n w_{\pi(j)} \right)^q + \sum_{i=k+2}^n \left(\sum_{j=i}^n w_{\pi(j)} \right)^q \right\}. \end{aligned}$$

Let π_k define the order where we switch jobs k and $k+1$ from order π . Given this new order we have

$$F(\pi) - F(\pi_k) = \mathcal{K}\xi \left\{ \left(w_{\pi(k+1)} + \sum_{j=k+2}^n w_{\pi(j)} \right)^q - \left(w_{\pi(k)} + \sum_{j=k+2}^n w_{\pi(j)} \right)^q \right\} > 0,$$

since $w_{\pi(k+1)} > w_{\pi(k)}$ by our initial assumption, which is a contradiction since this result implies that order π_k has a lower cost. \square

4 Extension to the Weighted Tardiness Problem

In this section we extend our results to the weighted tardiness setting. We still allow for arbitrary precedence constraints but no release dates. In this case, each job i also has a deadline d_i . The tardiness T_i of job i is defined as $T_i = \max\{0, C_i - d_i\}$, and the objective function is now given by,

$$g(\Pi, \mathbf{s}) = \sum_{i=1}^n v_i \rho_i s_i^{\beta-1} + \sum_{i=1}^n w_{\pi(i)} (C_{\pi(i)} - d_{\pi(i)})^+. \quad (4.1)$$

We now formulate the problem using a modification of the interval-and-speed-indexed formulation presented in Section 2. Because the completion time can be bounded by $\sum_{j=1}^m \sum_{t=1}^T \tau_{t-1} x_{ijt}$, we can bound (4.1) from below by the following optimization problem,

$$\min_{\mathbf{x}} \sum_{i=1}^n \sum_{j=1}^m \sum_{t=1}^T \left(v_i \rho_i \sigma_j^{\beta-1} + w_i (\tau_{t-1} - d_i)^+ \right) x_{ijt}, \quad (4.2)$$

together with constraints (2.4)-(2.8) from the interval-indexed formulation. Note that although the objective (4.1) is non-linear, because we have a interval-indexed formulation, (4.2) is linear.

We approximately solve (4.1) using the SCHEDULE BY α -INTERVALS AND α -SPEEDS FOR TARDINESS (SAIAS-T) Algorithm displayed in Figure 4.1. The main difference with the SAIAS algorithm, is that in step 4 we scale up the α -speeds. This scaling makes the completion time of the relaxed LP comparable to the completion time of the algorithm's output, and thus jobs that have 0 tardiness in the LP also have 0 tardiness in our algorithm. If we rounded speeds down, jobs with 0 tardiness in the LP could, at a lower speed, miss their deadline, and thus the approximation ratio could be arbitrary large.

We now analyse the algorithm assuming w.l.o.g. that $\tau_1^\alpha \leq \tau_2^\alpha \leq \dots \leq \tau_n^\alpha$. Since Lemma 3.2 remains valid, arguments identical to those in Section 2 show that the output of the SAIAS-T algorithm is feasible; thus, we have the following theorem:

SCHEDULE BY α -INTERVALS AND α -SPEEDS FOR TARDINESS (SAIAS-T)

- Inputs:** set of jobs, $\alpha \in (0, 1)$, $\epsilon > 0$, $\gamma > 1$, set of speeds $\mathbf{S} = \{\sigma_1, \dots, \sigma_m\}$.
- 1 Compute an optimal solution $\bar{\mathbf{x}}$ to the linear relaxation (4.2), (2.4)-(2.8).
 - 2 Compute the α -intervals τ^α and the sets J_t as in the SAIAS algorithm.
 - 3 Compute an order Π^α that has the sets J_t ordered in non-decreasing values of t and the jobs within each set in a manner consistent with the precedence constraints.
 - 4 Compute the α -speeds \mathbf{s}^α and scale each s_i^α to $\tilde{s}_i^\alpha = \gamma s_i^\alpha$.
 - 5 Round up each \tilde{s}_i^α to the next speed in \mathbf{S} , \bar{s}_i^α and run each job i at this new speed.
 - 6 Set the i -th job to start at time $\max\{r_{\pi(i)}, \bar{C}_{\pi(i-1)}^\alpha\}$, where $\bar{C}_{\pi(i-1)}^\alpha$ is the completion time of the previous job using the rounded α -speeds, and $\bar{C}_{\pi(0)}^\alpha = 0$.
 - 7 **return** speeds $\bar{\mathbf{s}}^\alpha$ and schedule $(\Pi^\alpha, \bar{\mathbf{C}}^\alpha)$.

Figure 4.1: Schedule by α -intervals and α -speeds for Tardiness Algorithm

Theorem 4.1. *The SAIAS-T algorithm with $\gamma = \frac{(1+\epsilon)}{\alpha(1-\alpha)}$ and $\alpha = \frac{1}{2}$ is a $4^\beta(1+\epsilon)^{\beta-1}(1+\delta)^{\beta-1}$ -approximation algorithm for the $1|prec|\sum E_i(s_i) + \sum w_i T_i$ problem, with $E_i(s_i) = v_i \rho_i s_i^{\beta-1}$.*

Proof. Let $\bar{C}_i = \sum_{j=1}^m \sum_{t=1}^T \tau_{t-1} \bar{x}_{ijt}$ denote the optimal fractional completion time of the relaxed linear program. $(\bar{C}_i - d_i)^+$ is a lower bound for the optimal tardiness $(C_i^* - d_i)^+$, since $\sum_{jt} (\tau_{t-1} - d_i)^+ \bar{x}_{ijt} \geq (\bar{C}_i - d_i)^+$. Thus,

$$\sum_{i=1}^n \sum_{j=1}^m \sum_{t=1}^T v_i \rho_i \sigma_j^{\beta-1} \bar{x}_{ijt} + \sum_{i=1}^n w_i (\bar{C}_i - d_i)^+ \leq \sum_{i=1}^n \sum_{j=1}^m \sum_{t=1}^T v_i \rho_i \sigma_j^{\beta-1} x_{ijt}^* + \sum_{i=1}^n w_i (C_i^* - d_i)^+. \quad (4.3)$$

Let \tilde{C}_i^α denote the completion time of job i using speeds $\tilde{\mathbf{s}}^\alpha$ and C_i^α the one using speeds \mathbf{s}^α . Because there are no release date constraints, there is no idle time in between jobs; therefore,

$$\tilde{C}_i^\alpha = \sum_{j=1}^i \frac{\rho_j}{\tilde{s}_j^\alpha} = \frac{1}{\gamma} \sum_{j=1}^i \frac{\rho_j}{s_j^\alpha} = \frac{1}{\gamma} C_i^\alpha. \quad (4.4)$$

Since (3.9) remains valid, it follows that $C_i^\alpha \leq \frac{(1+\epsilon)}{\alpha(1-\alpha)} \bar{C}_i \Rightarrow \tilde{C}_i^\alpha \leq \frac{1}{\gamma} \frac{(1+\epsilon)}{\alpha(1-\alpha)} \bar{C}_i$. The key step is that by setting $\gamma = \frac{(1+\epsilon)}{\alpha(1-\alpha)}$, which makes the two completion times comparable, we have that,

$$\sum_{i=1}^n w_i (\tilde{C}_i^\alpha - d_i)^+ \leq \sum_{i=1}^n w_i \left(\frac{1}{\gamma} \frac{(1+\epsilon)}{\alpha(1-\alpha)} \bar{C}_i - d_i \right)^+ = \sum_{i=1}^n w_i (\bar{C}_i - d_i)^+. \quad (4.5)$$

The energy term is bounded in a manner analogous to (3.8):

$$v_i \rho_i (\tilde{s}_i^\alpha)^{\beta-1} = \gamma^{\beta-1} v_i \rho_i (s_i^\alpha)^{\beta-1} \leq \frac{(1+\epsilon)^{\beta-1}}{(\alpha(1-\alpha))^\beta} v_i \rho_i \sum_{j=1}^m \sum_{t=1}^T \sigma_j^{\beta-1} \bar{x}_{ijt}, \quad (4.6)$$

where the last inequality follows from (3.8) that remains valid.

From (4.3), (4.6), and (4.5) it follows that,

$$\sum_{i=1}^n v_i \rho_i (\tilde{s}_i^\alpha)^{\beta-1} + \sum_{i=1}^n w_i (\tilde{C}_i^\alpha - d_i)^+ \leq \frac{(1+\epsilon)^{\beta-1}}{(\alpha(1-\alpha))^\beta} \left[\sum_{i=1}^n \sum_{j=1}^m \sum_{t=1}^T v_i \rho_i \sigma_j^{\beta-1} x_{ijt}^* + \sum_{i=1}^n w_i (C_i^* - d_i)^+ \right].$$

Because speeds are rounded up, the completion times, and thus the tardiness can only improve, whereas the energy cost increases. Since at most we speed up each job by a factor $(1 + \delta)$, we have that,

$$E_i(\bar{s}_i^\alpha) \leq E_i((1 + \delta)s_i^\alpha) = (1 + \delta)^{\beta-1} E_i(s_i^\alpha) \Rightarrow \sum_{i=1}^n E_i(\bar{s}_i^\alpha) \leq (1 + \delta)^{\beta-1} \sum_{i=1}^n E_i(s_i^\alpha). \quad (4.7)$$

The approximation ratio follows from setting $\alpha = \arg \min_{0 \leq \alpha \leq 1} \left\{ \frac{1}{(\alpha(1-\alpha))^\beta} \right\} = \frac{1}{2}$. Clearly we could use $\frac{(1+\epsilon)^{\beta-1}}{\alpha^\beta(1-\alpha)^{\beta-1}}$ in (4.6) to compute a tighter bound, but the resulting expression is not as simple. \square

We are not able to extend this algorithm for the $1|r_i| \sum E_i(s_i) + \sum w_i T_i$ problem, since it is based on speed scaling to make sure that jobs are finished within a desired time interval. When release dates are present, we do not see how to arbitrarily reduce the completion times.

5 Extension to General Energy Cost Functions

In this section we consider the extension to general energy *cost* functions, as opposed to simply energy *consumption*. We begin by considering discrete speeds, as in the previous sections, but in Section 5.3 we will relax this requirement.

Managers of data centres are clearly interested in the energy cost metric, since they need to balance the penalty for violating the service level agreements with the cost of energy. The energy price curves for industrial consumers are often quite complicated because of energy contracts, discounts, real time pricing etc.; therefore it is very important to consider general cost functions in the scheduling model. Hence, in this section we use $\mathcal{E}_i(s_i)$ as the general energy cost function of running job i at speed s_i . We will require that $\mathcal{E}_i(s_i)$ is non-negative, just as in [6, 9], but no other requirements are needed for the weighted completion time setting. For the weighted tardiness setting we will require an additional regularity condition that bounds the growth of the energy cost function.

Since in practice the processor speed can be dynamically changed during the course of a job, one can replace the general cost function by its lower convex envelope. Hence, without loss of generality, we can assume that $\mathcal{E}_i(s_i)$ is convex. Furthermore, since the machine can only run at the speeds in \mathbf{S} , we can also consider that $\mathcal{E}_i(s)$ is linear in between these speeds. Hence, for every $s \in [\sigma_j, \sigma_{j+1}]$ such that $s = \lambda\sigma_j + (1-\lambda)\sigma_{j+1}$, with $\lambda \in [0, 1]$, then $\mathcal{E}_i(s) = \lambda\mathcal{E}_i(\sigma_j) + (1-\lambda)\mathcal{E}_i(\sigma_{j+1})$.

Note that for bounding the energy cost terms in the weighted completion time setting, we only used the fact that the energy consumption function $E_i(s) = v_i \rho_i s^{\beta-1}$ is convex. Thus, the previous bounds extend to our more general class of functions $\mathcal{E}_i(s)$. In the weighted tardiness case we required also a bound on the growth of the energy cost function, which we will address in Section 5.2.

5.1 Weighted Completion Time Problem with General Energy Cost

The objective function (2.3) is extended as follows,

$$\min_{\mathbf{x}} \sum_{i=1}^n \sum_{j=1}^m \sum_{t=1}^T (\mathcal{E}_i(\sigma_j) + w_i \tau_{t-1}) x_{ijt}, \quad (5.1)$$

where $\mathcal{E}_i(\sigma_j)$ are just coefficients. Given that we only change the energy cost related terms, all the completion time related bounds computed previously are still valid.

The only modification required is in the rounding procedure at the end of the SAIAS algorithm, where it was done by rounding down the α -speeds. Now instead we will round them up or down such that $\mathcal{E}_i(\bar{s}_i^\alpha) \leq \mathcal{E}_i(s_i^\alpha)$, which is always possible since $\mathcal{E}_i(s_i)$ is linear in between the speeds in \mathbf{S} . With this change Lemma 3.1 remains valid and we can extend the algorithm to our general energy cost functions.

Theorem 5.1. *The SAIAS algorithm with $\alpha = \frac{1}{2}$ is a $4(1 + \epsilon)(1 + \delta)$ -approximation algorithm for the $1|prec|\sum \mathcal{E}_i(s_i) + \sum w_i C_i$ problem, for all general non-negative energy cost functions $\mathcal{E}_i(s)$.*

Proof. Because $\mathcal{E}_i(\sigma)$, $i = \{1, \dots, n\}$ are convex functions, (3.7) remains valid since $\mathcal{E}_i(s_i^\alpha) = \mathcal{E}_i(\mathbb{E}[\hat{s}_i]) \leq \mathbb{E}[\mathcal{E}_i(\hat{s}_i)] = \sum_{j=1}^m \mu_{ij} \mathcal{E}_i(\sigma_j)$, and thus, from the definition of μ_{ij} , and from $0 \leq \alpha \leq 1$, $\epsilon > 0$, and $\tilde{x}_{ijt} \leq \bar{x}_{ijt}$,

$$\sum_{i=1}^n \mathcal{E}_i(s_i^\alpha) \leq \frac{1}{\alpha} \sum_{i=1}^n \sum_{j=1}^m \sum_{t=1}^{\tau_i^\alpha} \mathcal{E}_i(\sigma_j) \tilde{x}_{ijt} \leq \frac{(1 + \epsilon)}{\alpha(1 - \alpha)} \sum_{i=1}^n \sum_{j=1}^m \sum_{t=1}^T \mathcal{E}_i(\sigma_j) \bar{x}_{ijt}. \quad (5.2)$$

The proof follows since the bounds for the completion time in Theorem 3.1 remain valid, as well as Lemma 3.1. \square

By the same argument we also have that,

Theorem 5.2. *The SAIAS algorithm with $\alpha = \sqrt{2} - 1$ is a $(3 + 2\sqrt{2})(1 + \epsilon)(1 + \delta)$ -approximation algorithm for the $1|r_i, prec|\sum \mathcal{E}_i(s_i) + \sum w_i C_i$ problem, for all general non-negative energy cost functions $\mathcal{E}_i(s)$.*

5.2 Weighted Tardiness Problem with General Energy Cost

We replace the energy term in (4.2) with the general energy cost term to obtain the new objective

$$\min_{\mathbf{x}} \sum_{i=1}^n \sum_{j=1}^m \sum_{t=1}^T (\mathcal{E}_i(\sigma_j) + w_i (\tau_{t-1} - d_i)^+) x_{ijt}. \quad (5.3)$$

Since the SAIAS-T algorithm speeds up the jobs, we need to add the following regularity condition for the energy cost functions $\mathcal{E}_i(\sigma)$ in order to obtain performance bounds:

Assumption 5.1. $\exists \beta \in \mathbb{N}^+$, such that $\mathcal{E}_i(\gamma \sigma_i) \leq \gamma^{\beta-1} \mathcal{E}_i(\sigma_i)$, $\forall \gamma \geq 1$.

Theorem 5.3. *The SAIAS-T algorithm with $\gamma = \frac{(1+\epsilon)}{\alpha(1-\alpha)}$ and $\alpha = \frac{1}{2}$, is a $4^\beta(1 + \epsilon)^{\beta-1}(1 + \delta)^{\beta-1}$ -approximation algorithm for the $1|prec|\sum \mathcal{E}_i(s_i) + \sum w_i T_i$ problem, for all non-negative energy cost functions $\mathcal{E}_i(s)$ that satisfy Assumption 5.1.*

Proof. As before, all the completion time related bounds (4.4) and (4.5) remain valid, so only a bound analogous to (4.6) is needed. From Assumption 5.1 it follows that,

$$\mathcal{E}_i(\tilde{s}_i^\alpha) \leq \gamma^{\beta-1} \mathcal{E}_i(s_i^\alpha) \leq \frac{(1 + \epsilon)^{\beta-1}}{\alpha^\beta(1 - \alpha)^\beta} \sum_{j=1}^m \sum_{t=1}^T \mathcal{E}_i(\sigma_j) \bar{x}_{ijt}. \quad (5.4)$$

Thus, from (4.5) it follows that,

$$\sum_{i=1}^n \mathcal{E}_i(\tilde{s}_i^\alpha) + \sum_{i=1}^n w_i (\tilde{C}_i^\alpha - d_i)^+ \leq \frac{(1 + \epsilon)^{\beta-1}}{\alpha^\beta(1 - \alpha)^\beta} \left[\sum_{i=1}^n \sum_{j=1}^m \sum_{t=1}^T \mathcal{E}_i(\sigma_j) x_{ijt}^* + \sum_{i=1}^n w_i (C_i^* - d_i)^+ \right].$$

Since we are rounding speeds up, equation (4.7) remains valid and thus taking $\alpha = \frac{1}{2}$ completes the proof. \square

5.3 Continuous Speeds

As commented previously, our algorithms are also applicable for the case when a continuous set of speeds is possible. In this case we modify the SAIAS and SAIAS-T algorithms, eliminating the rounding step required at the end of each algorithm.

When the operating range of the machine is given, i.e. the speed limits σ_{\min} and σ_{\max} , since our IP requires a speed index, we need to quantize the set $[\sigma_{\min}, \sigma_{\max}]$ in m different speeds. We can do this by setting $\sigma_1 = \sigma_{\min}$, and as before we define speed $\sigma_j = (1 + \delta)\sigma_{j-1}$, for some $\delta > 0$, making sure that $\sigma_m \geq \sigma_{\max}$ in order to cover the whole operating range. Just by rounding as described in Section 5.1 for the weighted completion time setting and rounding up for the weighted tardiness setting we can prove the following lemma:

Lemma 5.1. *The optimal solution for the IP (2.3)-(2.8) is at most $(1 + \delta)$ times the optimal solution of the energy aware problem in the weighted completion time and continuous speed setting, and the optimal solution for the IP (4.2), (2.4)-(2.8) is at most $(1 + \delta)^{\beta-1}$ times the optimal solution of the energy aware problem in the weighted tardiness and continuous speed setting.*

The proof is similar to Lemma 3.1 for the weighted completion time and similar to equation (4.7) for the weighted tardiness setting.

Since there is no additional rounding at the end of the algorithm, using Lemma 5.1 we get the same approximation ratios as in Theorems 5.1, 5.2, and 5.3.

When the operating range of the machine is not given, and we are interested in determining a set \mathbf{S} that covers the optimal speeds from the continuous case, we need the following additional regularity condition on the energy cost functions: $\exists \xi < \infty$ such that $\mathcal{E}_j(s_i)$ is increasing $\forall s_i \geq \xi$. It is easy to prove that this is a necessary and sufficient conditions for the problem to be well defined, and thus we can compute σ_{\min} and σ_{\max} such that the optimal speeds $s_i^* \in [\sigma_{\min}, \sigma_{\max}]$, for all i . Then we can apply the same procedure as before to quantize and build the set of speeds, and proceed to compute an approximate solution.

6 Conclusion

In this work we described new techniques for developing constant approximation algorithms for energy aware scheduling problems with very general job-dependent energy cost functions, that work on both discrete and continuous speed sets. Furthermore, we present the first algorithm, to the best of our knowledge, that tackles the energy aware weighted tardiness setting, even in the presence of arbitrary precedence constraints.

We believe that our methodology, which extends the idea of α -points to the energy aware setting by developing the α -speeds concept, should have many more applications. We suspect that, via techniques such as using randomly chosen values of α or using different α values for different jobs, we could obtain tighter bounds, and also that these techniques could be extended to other settings, such as multiple parallel machines among others.

References

- [1] Google Datacentre Webpage. <http://goo.gl/44nDs>, 2009.
- [2] Comscore May 2011 Ranking. <http://goo.gl/HuX0p>, 2011.
- [3] US Department of Energy. http://www.oe.energy.gov/information_center/faq.htm, 2011.
- [4] ALBERS, S. *Algorithms for Energy Saving*, vol. 5760 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 173–186.

- [5] ALBERS, S., AND FUJIWARA, H. Energy-efficient algorithms for flow time minimization. *ACM Transactions on Algorithms* 3, 4 (Nov. 2007), 49–es.
- [6] ANDREW, L. L., WIERMAN, A., AND TANG, A. Optimal speed scaling under arbitrary power functions. *ACM SIGMETRICS Performance Evaluation Review* 37, 2 (Oct. 2009), 39.
- [7] BANSAL, N., BUNDE, D., CHAN, H. L., AND PRUHS, K. R. Average rate speed scaling. In *Proceedings of the 8th Latin American conference on Theoretical informatics* (Dec. 2008), Springer-Verlag, pp. 240–251.
- [8] BANSAL, N., CHAN, H. L., KHANDEKAR, R., PRUHS, K. R., STEIN, C., AND SCHIEBER, B. Non-preemptive min-sum scheduling with resource augmentation. *Small* (2007).
- [9] BANSAL, N., CHAN, H. L., AND PRUHS, K. R. Speed scaling with an arbitrary power function. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms* (2009), Society for Industrial and Applied Mathematics, pp. 693–701.
- [10] BANSAL, N., KIMBREL, T., AND PRUHS, K. R. Dynamic speed scaling to manage energy and temperature. *Energy* (2004).
- [11] BANSAL, N., KIMBREL, T., AND PRUHS, K. R. Speed scaling to manage energy and temperature. *Journal of the ACM (JACM)* 54, 1 (Mar. 2007), 3.
- [12] BANSAL, N., PRUHS, K. R., AND STEIN, C. Speed scaling for weighted flow time. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms* (2007), vol. pages, Society for Industrial and Applied Mathematics, p. 813.
- [13] CHEKURI, C., MOTWANI, R., NATARAJAN, B., AND STEIN, C. Approximation Techniques for Average Completion Time Scheduling. *SIAM Journal on Computing* 31, 1 (2001), 146.
- [14] GOEMANS, M. X. Improved approximation algorithms for scheduling with release dates. *ACM-SIAM symposium on Discrete algorithms* (1997), 591–598.
- [15] GOEMANS, M. X., QUEYRANNE, M., SCHULZ, A. S., SKUTELLA, M., AND WANG, Y. Single Machine Scheduling with Release Dates. *SIAM Journal on Discrete Mathematics* 15, 2 (2002), 165.
- [16] GRAHAM, R., LAWLER, E. L., LENSTRA, J. K., AND RINNOOY KAN, A. H. G. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Discrete optimization* 5 (1979), 287–326.
- [17] HALL, L. A., SCHULZ, A. S., SHMOYS, D. B., AND WEIN, J. Scheduling to Minimize Average Completion Time : Off-line and On-line Approximation Algorithms. *Industrial Engineering* 22, 3 (1997), 513–544.
- [18] HALL, L. A., SHMOYS, D. B., AND WEIN, J. Scheduling to minimize average completion time: Off-line and on-line algorithms. *Mathematics of Operations Research* 22 (1997), 513–544.
- [19] IRANI, S., AND PRUHS, K. R. Algorithmic problems in power management. *ACM SIGACT News* 36, 2 (June 2005), 63.
- [20] PHILLIPS, C. A., STEIN, C., AND WEIN, J. Minimizing average completion time in the presence of release dates. *Mathematical Programming* 82, 1-2 (June 1998), 199–223.

- [21] PINEDO, M. *Scheduling: Theory, Algorithms, and Systems*, 3rd ed. Springer New York, New York, NY, 2008.
- [22] PRUHS, K. R., STEE, R., AND UTHAISOMBUT, P. Speed Scaling of Tasks with Precedence Constraints. *Theory of Computing Systems* 43, 1 (Oct. 2007), 67–80.
- [23] PRUHS, K. R., UTHAISOMBUT, P., AND WOEGINGER, G. Getting the best response for your erg. *ACM Transactions on Algorithms* 4, 3 (June 2008), 1–17.
- [24] SKUTELLA, M. *List Scheduling in Order of α -Points on a Single Machine*, vol. 3484 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 250–291.
- [25] YAO, F., DEMERS, A., AND SHENKER, S. A scheduling model for reduced CPU energy. In *Proceedings of IEEE 36th Annual Foundations of Computer Science* (1995), IEEE Comput. Soc. Press, pp. 374–382.