Dual Techniques for Scheduling on a Machine with Varying Speed*

Nicole Megow[1] and José Verschae[2]

[1] Department of Mathematics, Technische Universität Berlin, Germany
nmegow@math.tu-berlin.de
[2] Departamento de Ingeniería Industrial and Centro de Modelamiento Matemático, Universidad de Chile, Santiago, Chile
jverscha@ing.uchile.cl

Abstract. We study scheduling problems on a machine of varying speed. Assuming a known speed function (given through an oracle) we ask for a cost-efficient scheduling solution. Our main result is a PTAS for minimizing the total weighted completion time on a machine of varying speed. This implies also a PTAS for the closely related problem of scheduling to minimize generalized global cost functions. The key to our results is a re-interpretation of the problem within the well-known *two-dimensional Gantt chart*: instead of the standard approach of scheduling in the *time-dimension*, we construct scheduling solutions in the *weight-dimension*.

We also consider a dynamic problem variant in which deciding upon the speed is part of the scheduling problem and we are interested in the tradeoff between scheduling cost and speed-scaling cost, which is typically the energy consumption. We obtain two insightful results: (1) the optimal scheduling order is independent of the energy consumption and (2) the problem can be reduced to the setting where the speed of the machine is fixed, and thus admits a PTAS.

1 Introduction

In several computation and production environments we face scheduling problems in which the speed of resources may vary. We distinguish mainly two types of varying speed scenarios: one, in which the speed is a *given* function of time, and another *dynamic* setting in which deciding upon the processor speed is part of the scheduling problem. The first setting occurs, e.g., in production environments where the speed of a resource may change due to overloading, aging, or in an extreme case it may be completely unavailable due to maintenance or failure. The dynamic setting finds application particularly in modern computer architectures, where speed-scaling is an important tool for power-management. Here we are interested in the tradeoff between the power consumption and the quality-of-service. Both research directions—scheduling on a machine with given speed

* Supported by the German Science Foundation (DFG) under contract ME 3825/1, by FONDECYT grant 3130407, and by Nucleo Milenio Información y Coordinación en Redes ICM/FIC P10-024F.

F.V. Fomin et al. (Eds.): ICALP 2013, Part I, LNCS 7965, pp. 745–756, 2013.
© Springer-Verlag Berlin Heidelberg 2013

fluctuation as well as scheduling including speed-scaling—have been pursued quite extensively, but seemingly separately from each other.

The main focus of our work and the main technical contribution lie in the setting with a given speed function. We present a PTAS for scheduling to minimize the sum of weighted completion times $\sum_j w_j C_j$, which is best possible unless P=NP. In addition, we draw an interesting connection between the given speed and dynamic models which allows us to utilize the results for the given speed setting also for the dynamic problem. Very useful in our arguments is the well-known geometric view of the min-sum scheduling problem in a *two-dimensional Gantt chart*, an interpretation originally introduced in [11]. Crucial to our results is the deviation from the standard view of scheduling in the *time dimension* and switching to scheduling in the *weight dimension*. This dual view allows us to cope with the highly sensitive speed changes in the time dimension which prohibit standard rounding, guessing, and approximation techniques.

Previous Work

Research on scheduling on a machine of given varying speed has been mainly focused on the special case of scheduling with non-availability periods, see e.g. [18]. Despite a history of more than 30 years, only recently the first constant approximation for min $\sum w_j C_j$ was derived in [12]. In fact, their $(4+\varepsilon)$-approximation computes a universal sequence which has the same guarantee for any (unknown) speed function. For the setting with release dates, they give an approximation algorithm with the same guarantee for any given speed function. If the speed is only increasing, there is an efficient PTAS [20], if all release dates are equal. In this case the complexity remains an open question, whereas for general speed functions the problem is strongly NP-hard, even when for each job the weight and processing time are equal [22].

The problem of scheduling on a machine of varying speed is equivalent to scheduling on an ideal machine (of constant speed) but minimizing a more general global cost function $\sum w_j f(C_j)$, where f is a nondecreasing function. In this identification, $f(C)$ denotes the time that the varying speed machine needs to process a work volume of C [14]. Also, the special case of only nondecreasing (nonincreasing) speed functions corresponds to concave (convex) global cost functions. Recently, in [14] tight guarantees for the Smith rule for all convex and all concave functions f were given. They also show that the problem for increasing piecewise linear cost function is strongly NP-hard even with only two slopes, and so is our problem when the speed function takes only two distinct values.

Even more general min-sum cost functions have been studied, where each job may have its individual nondecreasing cost function. A $(2+\varepsilon)$-approximation was recently derived in [10]. For the more complex setting with release dates a randomized $\mathcal{O}(\log\log(n \max_j p_j))$-approximation is known [5]. Clearly, these results translate also to the setting with varying machine speed.

Scheduling with dynamic speed-scaling was initiated in [23] and became a very active research field in the past fifteen years. Most work focuses on scheduling problems where jobs have deadlines by which they must finish. We refer

to [2,15] for an overview. Closer to our setting is the work initiated by Pruhs et al. [19] where they obtain a polynomial algorithm for minimizing the total flow time given an energy budget if all jobs have the same work volume. This work is later continued by many other; see, e. g., [3, 6, 8] and the references therein. Most of this literature is concerned with online algorithms to minimize total (or weighted) flow time plus energy. The minimization of the weighted sum of completion times plus energy has been considered recently in [4, 7]. These works derive constant approximations for general non-preemptive models with unrelated machines and release dates [4] and additional precedence constraints [7]. For our general objective of speed-scaling with an energy budget, [4] also give a randomized $(2+\varepsilon)$-approximation for unrelated machines with release dates.

Our Results

We give several best possible algorithms for problem variants that involve scheduling to minimize the total weighted completion time on a single machine that may vary its speed.

Our main result is an efficient PTAS (Section 3) for scheduling to minimize $\sum w_j C_j$ on a machine of varying speed (given by an oracle). This is best possible since the problem is strongly NP-hard, even when the machine speed takes only two distinct values [14]. We also provide an FPTAS (Section 5.3) for the case that there is a constant number of time intervals with different uniform speeds (and the max ratio of speeds is bounded). Our results generalize recent previous results such as a PTAS on a machine with only *increasing* speeds [20] and FPTASes for only *one* non-availability period [16, 17].

Our results cannot be obtained with standard scheduling techniques which heavily rely on rounding processing requirements or completion times. Such approaches typically fail on machines that may change their speed since the slightest error introduced by rounding might provoke an unbounded increase in the solution cost. Similarly, adding any amount of idle time to the machine might be fatal. Our techniques completely avoid this difficulty by a change of paradigm. To explain our ideas it is helpful to use a 2D-Gantt chart interpretation [11]; see Section 2. As observed before, e.g., in [13], we obtain a *dual* scheduling problem by looking at the y-axis in a 2D-Gantt chart and switching the roles of the processing times and weights. In other words, a dual solution describes a schedule by specifying the remaining weight of the system at the moment a job completes. This simple idea avoids the difficulties on the time-axis and allows to combine old with new techniques for scheduling on the weight-axis.

In case that an algorithm can set the machine at arbitrary speeds, we show in Section 4 that the optimal scheduling sequence is independent of the available energy. This follows by analyzing a convex program that models the optimal energy assignment for a given job permutation. A similar observation was made independently by Vásquez [21] in a game-theoretic setting. We show that computing this universal optimal sequence corresponds to the problem of scheduling with a particular concave global cost function, which can be solved with our PTAS mentioned above, or with a PTAS for non-decreasing speed [20].

Interestingly, this reduction relies again on a problem transformation from time-space to weight-space in the 2D-Gantt chart. For a given scheduling sequence, we give an explicit formula for computing the optimal energy (speed) assignment. Thus, we have a PTAS for speed-scaling and scheduling for a given energy budget. We remark that the complexity of this problem is open.

In many applications, including most modern computer architectures, machines are only capable of using a given number of discrete power (speed) states. We also provide in Section 5 an efficient PTAS for this complex scenario. This algorithm is again based on our techniques relying on dual schedules. Furthermore, we obtain a $(1+\varepsilon)$-approximation of the Pareto frontier for the energy-cost bicriteria problem. On the other hand, we show that this problem is NP-hard even when there are only two speed states. We complement this result by giving an FPTAS for a constant number of available speeds.

We also notice that in the speed-scaling setting, our (F)PTAS results can be utilized to obtain a $(2+\varepsilon)$-approximation for the more general problem of preemptively scheduling jobs with non-trivial release dates on identical parallel machines. Here, we apply our previous results to solve a *fast single machine relaxation* [9] combined with a trick to control the actual job execution times. Then, we keep the energy assignments computed in the relaxation and apply *preemptive list scheduling* on parallel machines respecting release dates.

We finally remark that all our results for the setting with given speed translate directly to a corresponding result for the equivalent problem $1|\,|\sum_j w_j f(C_j)$ (with f non-decreasing).

2 Model, Definitions, and Preliminaries

Problem Definition. We consider two types of scheduling problems. In both cases we are given a set of jobs $J = \{1, \ldots, n\}$ with work volume (processing time at speed 1) $v_j \geq 0$ and weights $w_j \geq 1$. We seek a schedule on a single machine (permutation of jobs) that minimizes the sum of weighted completion times. The speed of the machine may vary—this is where the problems distinguish.

In the problem *scheduling on a machine of given varying speed* we assume that the speed function is given indirectly by an oracle. Given a value v, the oracle returns the first point in time when the machine can finish v units of work. Thus, for a given order of jobs, we can compute the execution time of each job and then the total cost of the solution (assuming that there is no idle time).

In the problem *scheduling with speed-scaling* an algorithm determines not only a schedule for the jobs but will also decide at which speed $s > 0$ the machine will run at any time. Running a machine at certain speed requires a certain amount of power. Power is typically modeled as a monomial (convex) function of speed, $P(s) = s^\alpha$ with a small constant $\alpha > 1$. Given an energy budget E, we ask for the optimal power (and thus speed) distribution and corresponding schedule that minimizes $\sum_j w_j C_j$. More generally, we are interested in quantifying the tradeoff between the scheduling objective $\sum_{j\in J} w_j C_j$ and the total energy consumption, that is, we aim for computing the Pareto curve for the bicriteria minimization problem. We consider two variants of speed-scaling: If the

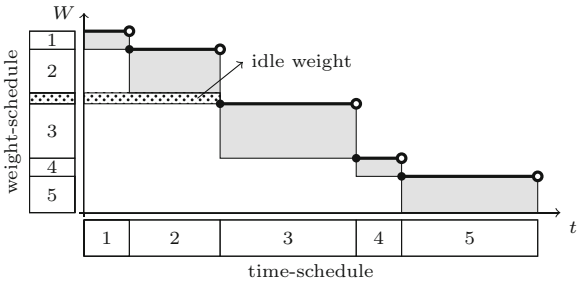

Fig. 1. 2D-Gantt chart. The x-axis shows a schedule, while the y-axis corresponds to $W(t) = \sum_{C_j > t} w_j$ plus the idle weight in the corresponding weight-schedule.

machine can run at an arbitrary power level $p \in \mathbb{R}_+$, we say that we are in the *continuous-speed* setting. On the other hand, if that machine can only choose among a finite set of speeds $\{s_1, \ldots, s_\kappa\}$ we are in an *discrete-speed* environment.

From Time-Space to Weight-Space. For a schedule $\mathcal{S}$, we let $C_j(\mathcal{S})$ denote the completion time of j and we let $W^{\mathcal{S}}(t)$ denote the total weight of jobs completed (strictly) after t. Whenever $\mathcal{S}$ is clear from the context we omit it. It is not hard to see that

$$\sum_{j \in J} w_j C_j(\mathcal{S}) = \int_0^\infty W^{\mathcal{S}}(t) dt. \tag{1}$$

Our main idea is to describe our schedule in terms of the remaining weight function W. That is, instead of determining C_j for each job j, we will implicitly describe the completion time of j by the value of W at the time that j completes. We call this value the *starting weight* of the job j, and denote it by S_j^w. Similarly, we define the *completion weight* of j as $C_j^w := S_j^w + w_j$. This has a natural interpretation in the two axes of the 2D-Gantt chart (see Figure 1): A typical schedule determines completion times for jobs in *time-space* (x-axis), which is highly sensitive when the speed of the machine may vary. We call such a solution a *time-schedule*. Describing a scheduling solution in terms of remaining weight can be seen as scheduling in the *weight-space* (y-axis), yielding a *weight-schedule*.

In weight-space the weights play the role of processing times. All notions that are usually considered in schedules apply in weight-space. For example, we say that a weight-schedule is feasible if there are no two jobs overlapping, and that the machine is idle at weight value w if $w \notin [S_j^w, C_j^w]$ for all j. In this case we say that w is *idle weight*. A weight-schedule immediately defines a non-preemptive time-schedule by ordering the jobs by decreasing completion weights.

Consider a weight-schedule $\mathcal{S}$ with completion weights $C_1^w \geq \ldots \geq C_n^w$, and corresponding completion times $C_1 \leq \ldots \leq C_n$. To simplify notation let $C_0 = C_{n+1}^w = 0$. Then we define the cost of $\mathcal{S}$ as $\sum_{j=1}^n (C_j^w - C_{j+1}^w) C_j$. It is easy to check, even from the 2D-Gantt chart, that this value equals $\sum_{j=1}^n x_j^{\mathcal{S}} C_j^w$, where $x_j^{\mathcal{S}}$ is the execution time of job j (in time-space). Moreover, the last expression is equivalent to Equation (1) if and only if the weight-schedule does

not have any idle weight. In general, the cost of the weight-schedule can only overestimate the cost of the corresponding schedule in time space, given by (1).

On a machine of varying speed, the weight-schedule has a number of technical advantages. For instance, while creating idle *time* can increase the cost arbitrarily, we can create idle *weight* without provoking an unbounded increase in the cost. This gives us flexibility in weight-space and implicitly a way to delay one or more jobs in the time-schedule without increasing the cost. More precisely, we have the following observation that can be easily seen in the 2D-Gantt chart.

Observation 1. *Consider a weight-schedule $\mathcal{S}$ with enough idle weight so that decreasing the completion weight of some job j, while leaving the rest untouched, yields a feasible weight-schedule. If the order of the jobs is changed, then the corresponding time-schedule is also modified. However, since the order of jobs in the time-schedule is reversed, job j gets delayed but the completion* time *of each job $j' \neq j$ is not increased. Thus, this operation does not increase the cost of $\mathcal{S}$.*

3 A PTAS for Scheduling on a Machine with Given Speeds

In what follows we give a PTAS for minimizing $\sum_j w_j C_j$ on a machine with a given speed function. In order to gain structure, we start by applying several modifications to the instance and optimal solution. First we round the weights of the jobs to the next integer power of $1+\varepsilon$, which can only increase the objective function by a factor $1+\varepsilon$. Additionally, we discretize the weight-space in intervals that increase exponentially. That is, we consider intervals $I_u = [(1+\varepsilon)^{u-1}, (1+\varepsilon)^u)$ for $u \in \{1, \ldots, \nu\}$ where $\nu := \left\lceil \log_{1+\varepsilon} \sum_{j \in J} w_j \right\rceil$. We denote the length of each interval I_u as $|I_u| := \varepsilon(1+\varepsilon)^{u-1}$. We will apply two important procedures to modify weight-schedules. They are used to create idle weight so to apply Observation 1, and they only increase the total cost by a factor $1+\mathcal{O}(\varepsilon)$. Similar techniques, applied in time-space, were used by Afrati et al. [1].

Weight Stretch: We multiply by $1+\varepsilon$ the completion weight of each job. This creates an idle weight interval of length εw_j before the starting weight of job j.

Stretch Intervals: We delay the completion weight of each job j with $C_j^w \in I_u$ by $|I_u|$, so that C_j^w belongs to I_{u+1}. Then $|I_{u+1}| - |I_u| = \varepsilon^2(1+\varepsilon)^{u-1} = \varepsilon |I_{u+1}|/(1+\varepsilon)$ units of weight are left idle in I_{u+1} after the transformation, unless there was only one job completely covering I_{u+1}. By moving jobs within I_u, we can assume that this idle weight is consecutive.

3.1 Dynamic Program

We now show our dynamic programming (DP) approach to obtain a PTAS. We first describe a DP table with exponentially many entries and then discuss how to reduce its size. Consider a subset of jobs $S \subseteq J$ and a partial schedule of S in the weight-space. In our dynamic program, S will correspond to the set of jobs at the beginning of the weight-schedule, i. e., if $j \in S$ and $k \in J \setminus S$

then $C_j^w < C_k^w$. A partial weight-schedule $\mathcal{S}$ of jobs in S implies a schedule in time-space with the following interpretation. Note that the makespan of the time-schedule is completely defined by the total work volume $\sum_j v_j$. We impose that the last job of the schedule, which corresponds to the first job in $\mathcal{S}$, finishes at the makespan. This uniquely determines a value of C_j for each $j \in S$, and thus also its execution time $x_j^{\mathcal{S}}$. The total cost of this partial schedule is $\sum_{j \in S} x_j^{\mathcal{S}} C_j^w$ (which has a simple interpretation in the 2D-Gantt chart).

Consider $\mathcal{F}_u := \{S \subseteq J : w(S) \leq (1+\varepsilon)^u\}$. That is, a set $S \in \mathcal{F}_u$ is a potential set to be scheduled in I_u or before. For a given interval I_u and set $S \in \mathcal{F}_u$, we construct a table entry $T(u, S)$ with a $(1 + \mathcal{O}(\varepsilon))$-approximation to the optimal cost of a weight-schedule of S subject to $C_j^w \leq (1+\varepsilon)^u$ for all $j \in S$.

Consider now $S \in \mathcal{F}_u$ and $S' \in \mathcal{F}_{u-1}$ with $S' \subseteq S$. Let $\mathcal{S}$ be a partial schedule of S where the set of jobs with completion weight in I_u is exactly $S \setminus S'$. We define $\mathrm{APX}_u(S', S) = (1+\varepsilon)^u \sum_{j \in S \setminus S'} x_j^{\mathcal{S}}$, which is a $(1+\varepsilon)$-approximation to $\sum_{j \in S \setminus S'} x_j^{\mathcal{S}} C_j^w$, the partial cost associated to $S \setminus S'$. We remark that the values $\sum_{j \in S \setminus S'} x_j^{\mathcal{S}}$ and $\mathrm{APX}_u(S', S)$ do not depend on the whole schedule $\mathcal{S}$, but only on the total work volume of jobs in S'. We can compute $T(u, S)$ with the following formula, $T(u, S) = \min\{T(u-1, S') + \mathrm{APX}_u(S', S) : S' \in \mathcal{F}_{u-1}, S' \subseteq S\}$.

The set $\mathcal{F}_u$ can be of exponential, and thus also this DP table. In the following we show that there is a polynomial size set $\tilde{\mathcal{F}}_u$ that yields $(1+\varepsilon)$-approximate solutions. We remark that the set $\tilde{\mathcal{F}}_u$ will not depend on the speed of the machine. Thus, the same set can be used in the speed-scaling scenario.

3.2 Light Jobs

We structure an instance by classifying jobs by their size in weight-space.

Definition 1. *In a given schedule, a job j is said to be* light *if $w_j \leq \varepsilon |I_u|$, where u is such that $S_j^w \in I_u$. A job that is not light is* heavy.

Given a weight-schedule for heavy jobs, we can greedily find a $(1 + \mathcal{O}(\varepsilon))$-approximate solution for the complete instance. To show this, consider any weight-schedule $\mathcal{S}$. First, remove all light jobs. Then we move jobs within each interval I_u, such that the idle weight inside each interval is consecutive. Clearly, this can only increase the cost of the solution by a $1+\varepsilon$ factor. After, we apply the following preemptive greedy algorithm to assign light jobs, which we call *Algorithm Smith in Weight-Space*: For $u = 1, \ldots, \nu$ and each idle weight $w \in I_u$, process a job j maximizing v_j/w_j among all available jobs with $w_j \leq \varepsilon |I_u|$.

To remove preemptions, we apply the Stretch Interval subroutine[1] twice, creating an idle weight interval in I_u of length at least $2\varepsilon|I_u|/(1+\varepsilon) \geq \varepsilon|I_u|$ (for $\varepsilon \leq 1$). This gives enough space in each interval I_u to completely process the (unique) preempted light job with starting weight in I_u. Then, Observation 1 implies that we can remove preemptions, obtaining a new schedule $\mathcal{S}'$. We now show that the cost of $\mathcal{S}'$ is at most a factor of $1 + \mathcal{O}(\varepsilon)$ larger than the cost of $\mathcal{S}$.

[1] The Stretch Interval procedure also applies to preemptive settings by interpreting each piece of a job as an independent job.

To do so we need a few definitions. For any weight-schedule $\mathcal{S}$, let us define the *remaining volume function* as $V^{\mathcal{S}}(w) := \sum_{j:C_j^w \geq w} v_j$. For a given w, let $I_j(w)$ be equal 1 if the weight-schedule processes j at weight w, and 0 otherwise. Then, $f_j(w) := (1/w_j) \int_w^\infty I_j(w')dw'$ corresponds to the fraction of job j processed after w. With this we define the *fractional remaining volume function*, which is similar to the remaining volume function but treats light jobs as "liquid":

$$V_f^{\mathcal{S}}(w) := \sum_{j:j \text{ is light}} f_j(w) \cdot v_j + \sum_{j:j \text{ is heavy}, C_j^w \geq w} v_j \qquad \text{for all } w \geq 0.$$

We notice that $V_f^{\mathcal{S}}(w) \leq V^{\mathcal{S}}(w)$ for all $w \geq 0$.

Consider now the function $f(v)$ corresponding to the earliest point in time in which the machine can process a work volume of v. Notice that this is the same function used when transforming our problem to $1||\sum_j w_j f(C_j)$. It is easy to see—even from the 2D-Gantt chart—that $\int_0^\infty f(V^{\mathcal{S}}(w))dw$ corresponds to the cost of the weight-schedule $\mathcal{S}$. Also, notice that $f(v)$ is non-decreasing, so that $V^{\mathcal{S}}(w) \leq V^{\mathcal{S}'}(w)$ for all w implies that the cost of $\mathcal{S}$ is at most the cost of $\mathcal{S}'$.

Lemma 1. *The cost of $\mathcal{S}'$ is at most $1 + \mathcal{O}(\varepsilon)$ times larger than the cost of $\mathcal{S}$.*

Proof (Idea). If we assume the position of heavy jobs as given, the schedule $\mathcal{S}_f$ returned by Algorithm Smith in Weight Space minimizes $V_f^{\mathcal{S}_f}(w)$ for any given $w \geq 0$. The result follows by combining this insight plus the ideas above.

Corollary 1. *At a loss of a $1 + \mathcal{O}(\varepsilon)$ factor in the objective function, we can assume the following. For a given interval I_u, consider any pair of jobs j, k whose weights are at most $\varepsilon|I_u|$. If both jobs are processed in I_u or later and $v_k/w_k \leq v_j/w_j$, then $C_j^w \leq C_k^w$.*

3.3 Localization and Compact Search Space

The objective of this section is to compute, for each job $j \in J$, two values r_j^w and d_j^w so that job j is scheduled completely within $[r_j^w, d_j^w)$ in some $(1+\mathcal{O}(\varepsilon))$-approximate weight-schedule. We call r_j^w and d_j^w the *release-weight* and *deadline-weight* of job j, respectively. Crucially, we need that the length of the interval $[r_j^w, d_j^w)$ is not too large, namely that $d_j \in \mathcal{O}(\text{poly}(1/\varepsilon)r_j)$. Such values can be obtained by using Corollary 1 and techniques from [1]; we skip the details.

Lemma 2. *We can compute in poly-time values r_j^w and d_j^w for each $j \in J$ such that: (i) there exists a $(1 + \mathcal{O}(\varepsilon))$-approximate weight-schedule that processes each job j within $[r_j^w, d_j^w)$, (ii) there exists a constant $s \in \mathcal{O}(\log(1/\varepsilon)/\varepsilon)$ such that $d_j^w \leq r_j^w \cdot (1+\varepsilon)^s$, (iii) r_j^w and d_j^w are integer powers of $(1+\varepsilon)$, and (iv) the values r_j^w an d_j^w are independent of the speed of the machine.*

Now we are ready to express set $\tilde{\mathcal{F}}_u$. Instead of describing a set $S \in \tilde{\mathcal{F}}_u$, we describe $V = J \setminus S$, that is, the jobs with completion weights in I_{u+1} or later. Clearly, Lemma 2 implies that we just need to decide about jobs with release weights $r_j^w = (1+\varepsilon)^v$ with $v \in \{u+1-s, \ldots, u-1\}$. Enumerating over (basically) all possibilities for each $v \in \{u+1-s, \ldots, u-1\}$, we obtain the following.

Lemma 3. *For each interval I_u, we can construct in poly-time a set $\tilde{\mathcal{F}}_u$ that satisfies the following: (i) there exists a $(1+\mathcal{O}(\varepsilon))$-approximate weight-schedule in which the set of jobs with completion weight at most $(1+\varepsilon)^u$ belongs to $\tilde{\mathcal{F}}_u$ for each interval I_u, (ii) the set $\tilde{\mathcal{F}}_u$ has cardinality at most $2^{\mathcal{O}(\log^2(1/\varepsilon)/\varepsilon^3)}$, and (iii) the set $\tilde{\mathcal{F}}_u$ is completely independent of the speed of the machine.*

With the discussion at the beginning of this section we obtain a PTAS, which is a best possible approximation since the problem is strongly NP-hard [14].

Theorem 1. *There exists an efficient PTAS for minimizing the weighted sum of completion times on a machine with given varying speed.*

4 Speed-Scaling for Continuous Speeds

When assuming a continuous spectrum of speeds, each job will be executed at a uniform speed because of the convexity of the power function [23]. Let s_j be the speed at which job j is running. Then j's power consumption is $p_j = s_j^\alpha$, and its execution time is $x_j = v_j/s_j = v_j/p_j^{1/\alpha}$. The energy that is required for processing j is $E_j = p_j \cdot x_j = p_j \cdot \frac{v_j}{s_j} = s_j^{\alpha-1} \cdot v_j = v_j^\alpha / x_j^{\alpha-1}$.

Let π be a sequence of jobs in a schedule, where $\pi(j)$ is the index of the j-th job in the sequence for each $i \in \{1,\dots,n\}$. Computing the optimal energy assignment for all jobs, given a fixed sequence π and using a total amount of energy E, can be done with a convex program. We rewrite the objective function as $\sum_{j=1}^n w_j C_j = \sum_{j=1}^n w_{\pi(j)} \sum_{k=1}^j x_{\pi(k)} = \sum_{j=1}^n x_{\pi(j)} \sum_{k=j}^n w_{\pi(k)}$ and define $W^\pi_{\pi(j)} = \sum_{k=j}^n w_{\pi(k)}$. Note that $x_j = \left(v_j^\alpha / E_j\right)^{1/(\alpha-1)}$, and that W_j^π is the total remaining weight just before j is completed in any schedule concordant with π.

$$\min \left\{ \sum_{j=1}^n W_j^\pi \cdot \left(\frac{v_j^\alpha}{E_j}\right)^{1/(\alpha-1)} : \quad \sum_{j=1}^n E_j \le E, \text{ and } E_j \ge 0 \quad \forall j \in \{1,\dots,n\} \right\}.$$

This program has linear constraints and a convex objective function. The next theorem easily follows by the well-known KKT conditions.

Theorem 2. *For a given job sequence π, a power function $P(s) = s^\alpha$ and an energy budget E, the optimal energy assignment in an optimal schedule for minimizing $\sum_j w_j C_j$ subject to $C_{\pi(1)} < \ldots < C_{\pi(n)}$ is determined by*

$$E_j = v_j \cdot \left(W_j^\pi\right)^{(\alpha-1)/\alpha} \cdot \frac{E}{\gamma_\pi}, \text{ where } \gamma_\pi = \sum_{j=1}^n v_j \cdot \left(W_j^\pi\right)^{(\alpha-1)/\alpha}.$$

Interestingly, the optimal job sequence is independent of the energy distribution, and even stronger, it is independent of the overall energy budget. In other words, one scheduling sequence is universally optimal for all energy budgets. Furthermore, this sequence is obtained by solving *in weight-space* a (standard) scheduling problem with a cost function that depends on the power function.

Theorem 3. *Given a power function $P(s) = s^\alpha$, there is a universal sequence that minimizes $\sum_j w_j C_j$ for any energy budget. The sequence is given by reversing an optimal solution of the scheduling problem $1|\,|\sum w_j C_j^{(\alpha-1)/\alpha}$ on a single machine of unit speed.*

Thus, the scheduling part of the speed-scaling scheduling problem reduces to a problem which can be solved by our PTAS from Sect. 3. Since the cost function $f(x) = x^{(\alpha-1)/\alpha}$ is concave for $\alpha > 1$, the specialized PTAS in [20] also solves it. Combining Theorems 2 and 3 gives the main result.

Theorem 4. *There is a PTAS for the continuous speed-scaling and scheduling problem with a given energy budget E.*

5 Speed-Scaling for Discrete Speeds

In this section we consider a more realistic setting, where the machine can choose from a set of κ different speeds available $s_1 > \ldots > s_\kappa \geq 1$.

5.1 A PTAS for Discrete Speeds

Let the power function $P(s)$ be an arbitrary computable function. To derive our algorithm, we adopt the PTAS for scheduling on a machine with given varying speed (Sect. 3) and incorporate the allocation of energy.

We adopt the same definitions of weight intervals I_u and sets $\mathcal{F}_u$ as in Sect. 3. For a subset of jobs $S \in \mathcal{F}_u$ and a value $z \geq 0$, let $E[u, S, z]$ be the minimum total energy necessary for scheduling S such that $C_j^w \leq (1+\varepsilon)^u$ for each $j \in S$, and the scheduling cost is at most z, i.e., $\sum_{j\in S} x_j \cdot C_j^w \leq z$ where x_j is the execution time under some feasible speed assignment. Recall that the speed assignment determines the energy. The recursive definition of a state is as follows:

$$E(u, S, z) = \min\{E(u-1, S', z') + \mathrm{APX}_u(S \setminus S', z - z') : S' \in \mathcal{F}_{u-1}, S' \subseteq S\}.$$

Here $\mathrm{APX}_u(S \setminus S', z - z')$ is the minimum energy necessary for scheduling all jobs $j \in S \setminus S'$ with $C_j^w \in I_u$, such that their partial (rounded) cost $\sum_{j\in S\setminus S'} x_j(1+\varepsilon)^u$ is at most $z - z'$.

Lemma 4. *The value $APX_u(S \setminus S', z - z')$ can be computed with an LP.*

We let the DP fill the table for $u \in \{0, \ldots, \nu\}$ with $\nu = \lceil \log \sum_{j\in J} w_j \rceil$ and $z \in [1, z_{\mathrm{UB}}]$ for some upper bound such as $z_{\mathrm{UB}} = \sum_{j\in J} w_j \sum_{k=1}^{j} v_j/s_\kappa$. Then among all end states $[\nu, J, \cdot\,]$ with value at most the energy budget E we choose the one with minimum cost z. Then we obtain the corresponding $(1+\varepsilon)$-approximate solution for energy E by backtracking.

This DP has an exponential number of entries. However, we can apply results from Section 3 and standard rounding techniques to reduce the running time.

Theorem 5. *There is an efficient PTAS for minimizing the total scheduling cost for speed-scaling with a given energy budget.*

5.2 Speed-Scaling for Discrete Speeds Is NP-Hard

We complement Theorem 5 by showing that our problem is NP-hard. Our reduction is based on the NP-hard problem $1|d_j = d|\sum w_j T_j$ [24].

Theorem 6. *The problem of minimizing $\sum_j w_j C_j$ on a single machine for discrete speeds is NP-hard, even if the number of available power levels is 2.*

5.3 An FPTAS for a Constant Number of Speed States

Let $s_1 > \ldots > s_\kappa \geq 1$. A simple interchange argument shows that an optimal solution chooses the speed non-increasing over time. We construct a schedule in weight-space. There are at most κ jobs that run at more than one speed; call them *split jobs*. We guess the split jobs together with their completion weight up to a factor $1 + \varepsilon$. This partitions the weight space into κ subintervals I_i, which we have to fill with the remaining jobs *non-preemptively*. By construction all jobs in one subinterval run at the same uniform speed. The high-level idea now is to use a DP for partitioning the remaining jobs and keeping control on the power consumption and the total cost. One critical point is that we do not know the execution time x_j for split jobs and we cannot guess them: this would cause a running time dependency on the max-speed ratio. However, for each possible objective value for the split jobs, we can compute the minimum energy with an LP similar to the one in the PTAS in Section 5.1.

The main challenge is to reduce the exponential number DP states to a polynomial size. The intuition behind our algorithm is to remove the states with the same (rounded) objective value and nearly the same total work (differing by at most $\varepsilon|I_i|/n$) assigned to an interval I_i. Among them, we want to store those with smallest amount of work in an interval I_i, in order to make sure that enough space remains for further jobs. To show that this approach is feasible we show bounds on the change in the total cost. This yields the next theorem. The theorem after follows by applying these techniques in time-space.

Theorem 7. *There is an FPTAS for speed-scaling with a given energy budget for $\min \sum w_j C_j$ on a single machine with constantly many discrete speeds.*

Theorem 8. *There exists an FPTAS for non-preemptive[2] scheduling to minimize $\sum w_j C_j$ on a single machine with a constant number of intervals of different, but uniform speed. For the resumable[1] setting, there is an FPTAS in the same setting when the maximum ratio of speeds is bounded.*

References

1. Afrati, F., Bampis, E., Chekuri, C., Karger, D., Kenyon, C., Khanna, S., Milis, I., Queyranne, M., Skutella, M., Stein, C., Sviridenko, M.: Approximation schemes for minimizing average weighted completion time with release dates. In: Proc. of FOCS, pp. 32–43 (1999)
2. Albers, S.: Energy-efficient algorithms. Commun. ACM 53(5), 86–96 (2010)
3. Albers, S., Fujiwara, H.: Energy-efficient algorithms for flow time minimization. ACM Trans. Algorithms 3 (2007)

[2] Resumable jobs may run during a speed-0 interval; non-preemptive jobs must not.

4. Angel, E., Bampis, E., Kacem, F.: Energy aware scheduling for unrelated parallel machines. In: Proc. of GreenCom, pp. 533–540 (2012)
5. Bansal, N., Pruhs, K.: The geometry of scheduling. In: Proc. of FOCS, pp. 407–414 (2010)
6. Bansal, N., Pruhs, K., Stein, C.: Speed scaling for weighted flow time. SIAM J. Comput. 39(4), 1294–1308 (2009)
7. Carrasco, R.A., Iyengar, G., Stein, C.: Energy aware scheduling for weighted completion time and weighted tardiness. arXiv:1110.0685 (2011)
8. Chan, S.-H., Lam, T.-W., Lee, L.-K.: Non-clairvoyant speed scaling for weighted flow time. In: de Berg, M., Meyer, U. (eds.) ESA 2010, Part I. LNCS, vol. 6346, pp. 23–35. Springer, Heidelberg (2010)
9. Chekuri, C., Motwani, R., Natarajan, B., Stein, C.: Approximation techniques for average completion time scheduling. SIAM J. Comput. 31(1), 146–166 (2001)
10. Cheung, M., Shmoys, D.B.: A primal-dual approximation algorithm for min-sum single-machine scheduling problems. In: Goldberg, L.A., Jansen, K., Ravi, R., Rolim, J.D.P. (eds.) APPROX/RANDOM 2011. LNCS, vol. 6845, pp. 135–146. Springer, Heidelberg (2011)
11. Eastman, W.L., Even, S., Isaacs, I.M.: Bounds for the optimal scheduling of n jobs on m processors. Management Sci. 11(2), 268–279 (1964)
12. Epstein, L., Levin, A., Marchetti-Spaccamela, A., Megow, N., Mestre, J., Skutella, M., Stougie, L.: Universal sequencing on a single machine. SIAM J. Comp. 41(3), 565–586 (2012)
13. Goemans, M.X., Williamson, D.P.: Two-dimensional Gantt charts and a scheduling algorithm of Lawler. SIAM J. Disc. Math. 13, 281–294 (2000)
14. Höhn, W., Jacobs, T.: On the performance of Smith's rule in single-machine scheduling with nonlinear cost. In: Fernández-Baca, D. (ed.) LATIN 2012. LNCS, vol. 7256, pp. 482–493. Springer, Heidelberg (2012)
15. Irani, S., Pruhs, K.: Algorithmic problems in power management. SIGACT News 36(2), 63–76 (2005)
16. Kacem, I., Mahjoub, A.: Fully polynomial time approximation scheme for the weighted flow-time minimization on a single machine with a fixed non-availability interval. Computers & Industrial Engineering 56(4), 1708–1712 (2009)
17. Kellerer, H., Strusevich, V.: Fully polynomial approximation schemes for a symmetric quadratic knapsack problem and its scheduling applications. Algorithmica 57(4), 769–795 (2010)
18. Lee, C.-Y.: Machine scheduling with availability constraints. In: Leung, J.-T. (ed.) Handbook of Scheduling. CRC Press (2004)
19. Pruhs, K., Uthaisombut, P., Woeginger, G.J.: Getting the best response for your erg. ACM Transactions on Algorithms 4(3) (2008)
20. Stiller, S., Wiese, A.: Increasing speed scheduling and flow scheduling. In: Cheong, O., Chwa, K.-Y., Park, K. (eds.) ISAAC 2010, Part II. LNCS, vol. 6507, pp. 279–290. Springer, Heidelberg (2010)
21. Vásquez, O.C.: Energy in computing systems with speed scaling: optimization and mechanisms design. arXiv:1212.6375 (2012)
22. Wang, G., Sun, H., Chu, C.: Preemptive scheduling with availability constraints to minimize total weighted completion times. Ann. Oper. Res. 133, 183–192 (2005)
23. Yao, F.F., Demers, A.J., Shenker, S.: A scheduling model for reduced CPU energy. In: Proc. of FOCS, pp. 374–382 (1995)
24. Yuan, J.: The NP-hardness of the single machine common due date weighted tardiness problem. Systems Science and Mathematical Sciences 5(4), 328–333 (1992)