Annals of Operations Research
https://doi.org/10.1007/s10479-021-04059-3

ORIGINAL RESEARCH

Optimal algorithms for scheduling under time-of-use tariffs

Lin Chen[1] · Nicole Megow[2] · Roman Rischke[3] · Leen Stougie[4] · José Verschae[5]

Accepted: 24 March 2021

Abstract

We consider a natural generalization of classical scheduling problems to a setting in which using a time unit for processing a job causes some time-dependent cost, the time-of-use tariff, which must be paid in addition to the standard scheduling cost. We focus on preemptive single-machine scheduling and two classical scheduling cost functions, the sum of (weighted) completion times and the maximum completion time, that is, the makespan. While these problems are easy to solve in the classical scheduling setting, they are considerably more complex when time-of-use tariffs must be considered. We contribute optimal polynomial-time algorithms and best possible approximation algorithms. For the problem of minimizing the total (weighted) completion time on a single machine, we present a polynomial-time algorithm that computes for any given sequence of jobs an optimal schedule, i.e., the optimal set of time slots to be used for preemptively scheduling jobs according to the given sequence. This result is based on dynamic programming using a subtle analysis of the structure of optimal solutions and a potential function argument. With this algorithm, we solve the unweighted problem optimally in polynomial time. For the more general problem, in which jobs may

A preliminary version of this paper with a subset of results appeared in the Proceedings of MFCS 2015 as Chen et al. (2015).

✉ Roman Rischke
roman.rischke@gmail.com

Lin Chen
chenlin198662@gmail.com

Nicole Megow
nicole.megow@uni-bremen.de

Leen Stougie
stougie@cwi.nl

José Verschae
jverschae@uc.cl

[1] Department of Computer Science, Texas Tech University, Lubbock, USA

[2] Faculty of Mathematics and Computer Science, University of Bremen, Bremen, Germany

[3] Artificial Intelligence Department, Fraunhofer Heinrich Hertz Institute, Berlin, Germany

[4] Department of Operations Analytics, Vrije Universiteit Amsterdam, CWI and Erable-INRIA, Amsterdam, The Netherlands

[5] Instituto de Ingeniería Matemática y Computacional, Escuela de Ingeniería and Facultad de Matemáticas, Pontificia Universidad Católica de Chile, Santiago, Chile

Published online: 08 April 2021

have individual weights, we develop a polynomial-time approximation scheme (PTAS) based on a dual scheduling approach introduced for scheduling on a machine of varying speed. As the weighted problem is strongly NP-hard, our PTAS is the best possible approximation we can hope for. For preemptive scheduling to minimize the makespan, we show that there is a comparably simple optimal algorithm with polynomial running time. This is true even in a certain generalized model with unrelated machines.

Keywords Scheduling · Time-of-use tariffs · Dynamic programming · Polynomial-time approximation scheme (PTAS) · Total weighted completion time · Makespan

1 Introduction

One of the classical problems in operations research is the Production Planning problem. It appears in almost any introductory course in Operations Research (Hillier and Lieberman 2014; Taha 2007). In its deterministic form, a production plan at lowest total cost is required to meet known demands over the next few weeks, given holding cost for keeping inventory at the end of the week, and with unit production cost varying over the weeks. It is a very early example of a problem model in which unit cost, or tariffs, for production, service, labor, energy, etc., *vary over time*.

Nowadays, new technologies allow direct communication of a much larger variety of time-of-use tariffs to customers. For example, in practice, electricity prices can differ largely from hour to hour. Producers or providers of these resources use this kind of variable pricing more and more to spread demand for their services, which can save enormously on the excessive costs that are usually involved to serve high peak demands. Customers are persuaded to direct their use of the scarce resources to time slots that are offered at cheaper rates. From the provider's point of view variable pricing problems have been studied quite extensively. For instance, revenue management is a well established subfield of operations research (e.g., Talluri and Van Ryzin 2006).

As in the Production Planning problem, in this paper we advocate models from the point of view of the user of the resources, who may take advantage of variable pricing by traveling, renting labor, using electricity, etc. at moments at which these services are offered at a lower price. This point of view forms a rich class of optimization problems in which in addition to classical objectives, the cost of using services needs to be taken into account.

This widely applicable framework is particularly well suited for scheduling problems, in which jobs need to be scheduled over time. Processing jobs requires labor, energy, computer power, or other resources that often exhibit variable tariffs over time. It leads to the natural generalization of scheduling problems, in which using a time slot incurs a certain cost, varying over time, which we refer to as *utilization cost*, that must be paid in addition to the actual scheduling cost. However natural and practicable this may seem, there appears to be very little theoretical research on such scheduling models. The only work we are aware of is by Wan and Qi (2010), Kulkarni and Munagala (2013), Fang et al. (2016) and Chen and Zhang (2019), where variable tariffs concern the cost of labor or the cost of energy.

The goal of this paper is to expedite the theoretical understanding of fundamental scheduling problems within the framework of time-varying costs or tariffs. We contribute optimal polynomial-time algorithms and best possible approximation algorithms for fundamental single-machine scheduling problems with the objectives of minimizing the sum of weighted completion times and the makespan. We further consider an extension of the makespan problem to unrelated machines.

1.1 Problem definition

We first describe the underlying classical scheduling problems. We are given a set of jobs $J := \{1, \dots, n\}$ where every job $j \in J$ has a given processing time $p_j \in \mathbb{N}$ and possibly a weight $w_j \in \mathbb{Q}_{\geq 0}$. Our main focus is on the problem of finding a preemptive schedule on a single machine such that the total (weighted) completion time, $\sum_{j \in J} w_j C_j$, is minimized; here C_j denotes the completion time of job j. Preemption means that the processing of a job may be interrupted at any time and can continue at any time later at no additional cost. In the three-field scheduling notation (Graham et al. 1979), this problem is denoted as $1 \mid pmtn \mid \sum w_j C_j$. We also consider the objective of minimizing the makespan $C_{\max} := \max_{j \in J} C_j$ and investigate a more general multi-machine environment, namely, unrelated machines, $R \mid pmtn \mid C_{\max}$. In this setting, there is given a set of machines M, and each job $j \in J$ has an individual processing time $p_{ij} \in \mathbb{N}$ for running on machine $i \in M$.

In this paper, we additionally consider time-of-use tariffs when making scheduling decisions. We assume that time is discretized into unit-size time slots. We are given a tariff or cost function $e : \mathbb{N} \to \mathbb{Q}_{\geq 0}$, where $e(t)$ denotes the tariff for processing job(s) at time slot $[t, t+1)$. We assume that e is a piecewise constant function, i.e., we assume that the time horizon is partitioned into given intervals $I_k = [s_k, d_k)$ with $s_k, d_k \in \mathbb{N}$, $k \in \{1, 2, \dots, K\}$, within which e has the same value e_k. It holds $s_1 = 0$ and $s_k = d_{k-1}$, for $k \in \{2, 3, \dots, K\}$. To ensure that all jobs can be scheduled in the given time intervals, it must hold that $d_K \geq C^0_{max}$, where C^0_{max} denotes the minimum makespan independent of time-of-use tariffs, i.e., when all time slots have utilization cost 0.

Regarding the input encoding, we remark that the cost function e is compactly specified by the utilization cost e_k for each of the K time intervals. In this paper we aim for algorithms with a running time that is polynomial in the input encoding, that is, polynomial in n, the number of jobs, and K, the number of intervals and possibly logarithmic in the values for processing times, weights and cost. In general, it is not even clear that a schedule can be encoded polynomially in the input. However, for our completion-time based minimization objectives, it is easy to observe that if an algorithm utilizes p unit-size time slots in an interval of equal cost, then it utilizes the first p slots within this interval, which simplifies the structure and the output of an optimal solution in a crucial way.

Given a schedule $\mathcal{S}$, let $y(t)$ be a binary variable indicating if any processing is assigned to time slot $[t, t+1)$. The *utilization cost* of $\mathcal{S}$ is $E(\mathcal{S}) = \sum_t e(t) y(t)$. That means, for any time unit that is used in $\mathcal{S}$ we pay the full tariff, even if the unit is only partially used. While the main focus of this paper is on single-machine scheduling, we extend our makespan result also to a more general multiple machine model, where a time slot that is paid for can be used by all machines. This models applications in which paying for a time unit on a resource gives access to all units of the resource. As an example, consider the reservation of a server for a certain time period, in which case all available processors on that server can be utilized.

The overall objective in scheduling with time-of-use tariffs is to find a schedule that minimizes the scheduling objective, $\sum_{j \in J} w_j C_j$ resp. $C_{\max}$, *plus* the utilization cost E. We refer to the resulting problems as $1 \mid pmtn \mid \sum w_j C_j + E$ and $R \mid pmtn \mid C_{\max} + E$. We remark that the results in this paper also hold for the minimization of any convex combination of the scheduling and utilization cost.

1.2 Related work

Scheduling with time-of-use tariffs (a.k.a. variable time slot cost) has been studied explicitly by Wan and Qi (2010), Kulkarni and Munagala (2013), Fang et al. (2016) and Chen and Zhang (2019). In their seminal paper, Wan and Qi (2010) consider several *non-preemptive* single machine problems, which are polynomial-time solvable in the classical setting, such as minimizing the total completion time, lateness, and total tardiness, or maximizing the weighted number of on-time jobs. These problems are shown to be strongly NP-hard when taking general tariffs into account, while efficient algorithms exist for special monotone tariff functions. In particular, the problem $1 \mid \mid \sum C_j + E$ is strongly NP-hard, and it is efficiently solvable when the tariff function is increasing or convex non-increasing (Wan and Qi 2010). Practical applications, however, often require non-monotone tariff functions, which lead to wide open problems in the context of preemptive and non-preemptive scheduling. In this paper, we answer complexity and approximability questions for fundamental preemptive scheduling problems.

Kulkarni and Munagala (2013) focus on a related problem in an *online setting*, namely, online flow-time minimization using resource augmentation. Their main result is a scalable algorithm that obtains a constant performance guarantee when the machine speed is increased by a constant factor and there are only two distinct unit tariffs. They also show that, in this online setting, for an arbitrary number of distinct unit tariffs there is no constant speedup-factor that allows for a constant approximate solution. For the problem considered in this paper, offline scheduling without release dates, Kulkarni and Munagala (2013) observed a relation to universal sequencing on a machine of varying speed (Epstein et al. 2012) which implies the following results: a pseudo-polynomial 4-approximation for $1 \mid pmtn \mid \sum w_j C_j + E$, which gives an optimal solution in case that all weights are equal, and a constant approximation in quasi-polynomial time for a constant number of distinct tariffs or when using a machine that processes jobs faster by a constant factor. In this paper, we substantially improve on those results.

Fang et al. (2016) study scheduling on a single machine under time-of-use electricity tariffs. They do not take the scheduling cost into account, but only the energy cost. In their model the time horizon is divided into K regions, each of which has a cost c_k per unit energy. For processing jobs the dynamic variable speed model is used; that is, the energy consumption is s^α per time unit if jobs run at speed s, whence, within region k, the energy cost is $s^\alpha c_k$. The objective is to minimize energy cost such that all jobs are scheduled within the K regions. They prove that the non-preemptive case is NP-hard and give a non-constant approximation, and for the preemptive case, they give a polynomial-time algorithm.

Chen and Zhang (2019) consider non-preemptive scheduling on a single machine so as to minimize the total utilization cost under certain scheduling feasibility constraints such as a common deadline for all jobs or a bound on the maximum lateness, maximum tardiness, maximum flow-time, or sum of completion times. They define a *valley* to be a cost interval I_k that has smaller cost than its neighboring intervals and show the following. General tariffs lead to a strongly NP-hard problem for any of the just mentioned constraints, and even very restricted tariff functions with more than one valley result in NP-hard problems that are not approximable within any constant factor. The problem with a common deadline on the job completion times is shown to admit a pseudo-polynomial-time algorithm when having two valleys, a polynomial-time algorithm for tariff functions with at most one valley, and an FPTAS if there are at most two valleys and $\max_k e_k / \min_k e_k$ is bounded. For the other mentioned constraints, they also present polynomial-time algorithms when having no more than one valley, where the problem with a bound on the sum of completion times requires the number of cost intervals, here K, to be fixed.

The general concept of taking into consideration additional (time-dependent) cost for resource utilization when scheduling has been implemented differently in other models. We mention the area of energy-aware scheduling, where energy consumption is taken into account; see Albers (2010) for an overview. Further, the area of scheduling with generalized non-decreasing (completion-) time dependent cost functions, such as minimizing $\sum_j w_j f(C_j)$, e.g. Epstein et al. (2012), Megow and Verschae (2018), Höhn and Jacobs (2015), or even more general job-individual cost functions $\sum_j f_j(C_j)$, e.g. Bansal and Pruhs (2014), Höhn et al. (2018), Cheung and Shmoys (2011), Cheung et al. (2017) has received quite some attention. Our model differs fundamentally from those models since our cost function may decrease with time. In fact, delaying the processing in favor of cheaper time slots may decrease the overall cost. This is not the case in the above-mentioned models. Thus, in our framework we have the additional dimension in decision-making of selecting the time slots that shall be utilized.

Finally, we point out some similarity between our model and *scheduling on a machine of varying speed*, which (with $\sum_j w_j C_j$ as objective function) is an equivalent statement of the problem of minimizing $\sum_j w_j f(C_j)$ on a single machine with constant speed (Epstein et al. 2012; Megow and Verschae 2018; Höhn and Jacobs 2015). We do not see any mathematical reduction from one problem to the other. However, it is noteworthy that the independently studied problem of scheduling with *non-availability periods*, see e.g. the survey by Lee (2004), is a special case of both the varying-speed and the time-varying tariff model. Indeed, machine non/availability can be expressed either by a $0/1$-speed or equivalently by an $\infty/0$ tariff. Results shown in this context imply that our problem $1 \mid pmtn \mid \sum w_j C_j + E$ is strongly NP-hard, even if there are only two distinct tariffs (Wang et al. 2005).

1.3 Our contribution

We contribute optimal polynomial-time algorithms and best possible approximation algorithms for the generalization of fundamental scheduling problems to a framework with time-varying tariffs. We consider the problems of minimizing the makespan and minimizing the sum of weighted completion times on a single machine, which are easy to solve in the classical scheduling setting, and that become considerably more complex when time-of-use tariffs must be considered.

In Sect. 2, we show that the unweighted problem $1 \mid pmtn \mid \sum C_j + E$ can be solved optimally in polynomial time. The key is a polynomial-time algorithm that computes for any given sequence of jobs an optimal schedule, i.e., the optimal set of time slots to be used for preemptively scheduling jobs according to the given sequence. This result is based on dynamic programming, using a subtle analysis of the structure of optimal solutions and a properly chosen potential function. This algorithm solves the unweighted problem optimally in polynomial time when using the observation that the optimal scheduling order is the *shortest processing time* order, independent of the decision which time slots are used.

This algorithm implies almost directly a polynomial-time $(4+\varepsilon)$-approximation algorithm for the more general problem in which jobs have individual weights, $1 \mid pmtn \mid \sum w_j C_j + E$. A 4-approximation was observed by Kulkarni and Munagala (2013) but it has pseudo-polynomial running time. While pseudo-polynomial-time algorithms are relatively easy to derive, it is quite remarkable that our DP's running time is polynomial in the input, in particular, independent of d_K.

Our main result in Sect. 3 is a polynomial-time approximation scheme (PTAS) for the weighted problem $1 \mid pmtn \mid \sum w_j C_j + E$, that is, an algorithm that computes for any fixed $\varepsilon > 0$ a $(1+\varepsilon)$-approximate schedule. Our PTAS is the best possible approximation we can hope for, unless P=NP, since the problem is strongly NP-hard, even if there are only two different tariffs (Wang et al. 2005). Our approach is inspired by a recent PTAS for scheduling on a machine of varying speed (Megow and Verschae 2018) and it uses some of its properties. As discussed before, we do not see a formal mathematical relation between these two seemingly related problems that allows us to apply the result from Megow and Verschae (2018) directly. The key is a dual view on scheduling: instead of directly constructing a schedule in the time-dimension, we first construct a dual scheduling solution in the weight-dimension which has a one-to-one correspondence to a true schedule. We design an exponential-time dynamic programming algorithm which can be trimmed to polynomial time using techniques known for scheduling with varying speed (Megow and Verschae 2018).

In Sect. 4 we consider the makespan minimization problem. We show a simple optimal polynomial-time algorithm for the problem $R \mid pmtn \mid C_{\max} + E$. We design a procedure that selects the optimal time slots to be utilized, given that we know their optimal *number*. That number can be determined by solving the scheduling problem *without* utilization cost, which can be done in polynomial time by solving a linear program (Lawler and Labetoulle 1978).

Finally, we remark that job preemption is crucial for obtaining constant worst-case performance ratios for the makespan and the min-sum problems considered in this paper. More precisely, there is no bounded approximation factor possible for scheduling with tariffs, unless P=NP, even if there are only two different tariffs, 0 and ∞. This can be shown by a reduction similar to one by Yuan et al. (2007) for scheduling with fixed jobs; an observation that has been mentioned also in Megow and Verschae (2009). It is due to the fact that it is NP-hard to decide whether a set of jobs can be partitioned into two sets with equal total processing time (2- PARTITION). To see the inapproximability, consider a set of non-preemptive jobs and two equal-length time intervals of cost 0 of total length equal to the sum of all job processing times and all other time slots with cost ∞. A polynomial-time algorithm with a bounded worst-case ratio decides whether all jobs fit into cheap time slots, which would solve 2- PARTITION in polynomial time.

2 An optimal algorithm for minimizing total completion time

In this section, we show how to solve the unweighted problem $1 \mid pmtn \mid \sum C_j + E$ to optimality. Our main result is as follows.

Theorem 1 *There is a polynomial-time algorithm for* $1 \mid pmtn \mid \sum C_j + E$.

An algorithm for the scheduling problem with time-of-use tariffs has to make essentially two types of decisions: (i) which time slots to use and (ii) how to schedule the jobs in these slots. It is not hard to see that these two decisions can be handled separately. In fact, the following observation on the optimal sequencing of jobs holds independently of the utilization decision and follows from a standard interchange argument.

Observation 1 *In an optimal schedule $\mathcal{S}^*$ for the problem $1 \mid pmtn \mid \sum C_j + E$, jobs are processed according to the Shortest Processing Time First (SPT) rule.*

Thus, in the remainder of the section we can focus on determining which time slots to use. We design an algorithm that computes, for any given (not necessarily optimal) scheduling sequence σ, an optimal utilization decision. In fact, we show this result even for the more general problem in which jobs have arbitrary weights.

Theorem 2 *Given an instance of* $1 \mid pmtn \mid \sum w_j C_j + E$ *and an arbitrary processing sequence of jobs* σ*, we can compute an optimal utilization decision for* σ *in polynomial time.*

Combining the optimal choice of time slots (Theorem 2) with the optimal processing order SPT (Observation 1) immediately implies Theorem 1.

The remainder of the section is devoted to proving Theorem 2. Thus, we choose any (not necessarily optimal) order of jobs, $\sigma = (1, \ldots, n)$, in which the jobs must be processed. We want to characterize an optimal schedule $\mathcal{S}^*$ for σ, that is, the optimal choice of time slots for scheduling σ. Among all optimal schedules we shall consider an optimal solution $\mathcal{S}^*$ that minimizes the value $\sum_{t=0}^{d_K-1} t \cdot y(t)$, where $y(t)$ is a binary variable that indicates if time slot $[t, t+1)$ is utilized or not.

We firstly identify structural properties of an optimal solution. Essentially, we give a full characterization which we can compute efficiently by dynamic programming. More precisely, we establish a closed form that characterizes the relationship between the tariff of a utilized slot and job weights in an optimal solution. This relationship allows us to decompose an optimal schedule into a series of sub-schedules. Our algorithm will first compute all possible sub-schedules and then use a dynamic programming approach to select and concatenate suitable sub-schedules.

In principle, an optimal schedule may preempt jobs at fractional time points. However, since time slots must be paid for entirely, any reasonable schedule on a single machine uses the utilized slots entirely as long as there are unprocessed jobs. It can be shown by a standard interchange argument that this is also true if we omit the requirement that time slots must be utilized entirely; for details, see Rischke (2016).

Lemma 1 *There is an optimal schedule* $\mathcal{S}^*$ *for* $1 \mid pmtn \mid \sum w_j C_j + E$ *in which all utilized time slots are entirely utilized and jobs are preempted only at integral points in time.*

Next, we introduce the concept of *split points*. Intuitively, split points are relevant time points for decomposing an optimal schedule into smaller sub-schedules. Our goal is to define a polynomial number of such time points in such a way that we can compute an optimal utilization decision for a given region between two consecutive split points when given a job sequence to be processed in this region. This is the basis for our dynamic program.

The beginning of any cost interval s_k is a natural candidate for such a relevant point. However, for technical reasons, in particular, for volume shifting arguments, we include also points $s_k + 1$. Let $\mathcal{P} := \bigcup_{k=1}^{K} \{s_k, s_k + 1\} \cup \{d_K\}$ denote the set of potential split points.

Definition 1 (*Split Point*) Consider an optimal schedule $\mathcal{S}^*$. For any job j, let S_j and C_j denote the start time and completion time of j, respectively. A time point $t \in \mathcal{P}$ is a split point for $\mathcal{S}^*$ if all jobs that start before t also finish their processing not later than t, i.e., if $\left\{j \in J : S_j < t\right\} = \left\{j \in J : C_j \le t\right\}$.

Given an optimal schedule $\mathcal{S}^*$, let $0 = \tau_1 < \tau_2 < \cdots < \tau_\ell = d_K$ be the sequence of *all* split points of $\mathcal{S}^*$. Note that there might be consecutive split points with no job processed in between. We denote the interval between two consecutive split points τ_x and τ_{x+1} as *region* $R_x^{\mathcal{S}^*} := \left[\tau_x, \tau_{x+1}\right)$, for $x \in \{1, \ldots, \ell - 1\}$.

Consider now any region $R_x^{\mathcal{S}^*}$ for an optimal schedule $\mathcal{S}^*$ with $x \in \{1, \dots, \ell - 1\}$ and let $J_x^{\mathcal{S}^*} := \left\{ j \in J : S_j \in R_x^{\mathcal{S}^*} \right\}$ denote the set of jobs that start and finish within $R_x^{\mathcal{S}^*}$. The set $J_x^{\mathcal{S}^*}$ might be empty.

We observe that any job j completing at the beginning of a cost interval I_k, i.e., $C_j = s_k \in R_x^{\mathcal{S}^*}$ or $C_j = s_k + 1 \in R_x^{\mathcal{S}^*}$, would make s_k resp. $s_k + 1$ a split point. Thus, no such job $j \in J_x^{\mathcal{S}^*}$ can exist.

Observation 2 *There is no job $j \in J_x^{\mathcal{S}^*}$ with $C_j \in R_x^{\mathcal{S}^*} \cap \mathcal{P}$.*

In the following sequence of lemmas, we characterize which time slots are utilized in an optimal schedule $\mathcal{S}^*$.

Lemma 2 *Consider a region $R_x^{\mathcal{S}^*}$ with the corresponding job set $J_x^{\mathcal{S}^*}$. For every job $j \in J_x^{\mathcal{S}^*}$, except the last job in $J_x^{\mathcal{S}^*}$ (according to our fixed job sequence σ), it holds that the time slots $\left[C_j, C_j + 1\right)$ and $\left[C_j - 2, C_j - 1\right)$ are utilized in $\mathcal{S}^*$.*

Proof The proof is by contradiction and uses the observation that an algorithm that utilizes p unit-size time slots in an interval I of equal cost is best off if it utilizes the first p slots within I, and so does $\mathcal{S}^*$. Let j' be the last job in $J_x^{\mathcal{S}^*}$ according to σ.

Suppose there is a job $j \in J_x^{\mathcal{S}^*} \setminus \left\{j'\right\}$ such that $\left[C_j, C_j + 1\right)$ is not utilized. Let I_k be the interval in which job j completes, i.e. $\left[C_j - 1, C_j\right) \in I_k$. By Observation 2 we know that $C_j \neq s_k + 1$, as otherwise $C_j = \tau_{x+1}$ and thus $j = j'$. Moreover, since an unused time slot cannot be followed by a utilized time slot in the same interval, it means that if $[C_j, C_j + 1)$ is not utilized then no later time slot in I_k is utilized. Thus, j is the last job processed in I_k, hence $d_k = \tau_{x+1}$ is a split point and $j = j'$.

Suppose now that there is a job $j \in J_x^{\mathcal{S}^*} \setminus \left\{j'\right\}$ such that $\left[C_j - 2, C_j - 1\right)$ is not utilized. As argued above, because we reserve the earliest time slots within any interval, we know that $[C_j - 2, C_j - 1)$ is not in I_k, whence $[C_j - 1, C_j)$ is the first time slot in I_k and therefore $C_j = s_k + 1$, making $s_k + 1 = \tau_{x+1}$ and $j = j'$. □

We say that interval I_k is *partially utilized* if at least one time slot in I_k is utilized, but not all. The following lemma shows that we may assume that in an optimal schedule that minimizes $\sum_{t=0}^{d_K - 1} t \cdot y(t)$, there is at most one interval per region partially utilized.

Lemma 3 *There exists an optimal schedule $\mathcal{S}^*$ in which for all $x \in \{1, 2, \dots, \ell - 1\}$ at most one interval in $R_x^{\mathcal{S}^*}$ is partially utilized.*

Proof For the sake of contradiction, suppose that there is more than one partially utilized interval in $R_x^{\mathcal{S}^*}$. Consider any two such intervals I_h and $I_{h'}$ with $h < h'$, and all intermediate intervals that are utilized entirely or not at all. Let $[t_h, t_h + 1)$ and $[t_{h'}, t_{h'} + 1)$ be the last utilized time slot in I_h and $I_{h'}$, respectively. If we utilize $[t_{h'} + 1, t_{h'} + 2)$ instead of $[t_h, t_h + 1)$ and redistribute the processing volume accordingly without changing the processing order , then the difference in cost is $\delta_1 := e_{h'} - e_h + \sum_{j \in J'} w_j$ with $J' := \left\{ j \in J : C_j \in \bigcup_{k=h+1}^{h'} I_k \right\}$ because all jobs in J' are delayed by exactly one time unit. This is true by Lemma 2 and because by Observation 2 no job $j \in J'$ finishes at $d_k = s_{k+1}$ for any k. If we utilize $[t_h + 1, t_h + 2)$ instead of $[t_{h'}, t_{h'} + 1)$ and redistribute the processing volume accordingly without changing the processing order , then the difference in cost is $\delta_2 := e_h - e_{h'} - \sum_{j \in J'} w_j$, again using Lemma 2 and Observation 2 to assert that no job $j \in J'$ finishes at $s_k + 1$ for any $h + 1 \leq k \leq h'$. Since $\delta_1 = -\delta_2$ and $\mathcal{S}^*$ is an optimal schedule, it must hold that $\delta_1 = \delta_2 = 0$. This, however, implies that there is another optimal schedule with earlier used time slots which contradicts our assumption that $\mathcal{S}^*$ minimizes the value $\sum_{t=0}^{d_K-1} t \cdot y(t)$. □

The next lemma characterizes the time slots that are used within a region. Let $e^j_{\max}$ be the maximum tariff spent for job j in $\mathcal{S}^*$. Furthermore, let $\Delta_x := \max_{j \in J_x^{\mathcal{S}^*}} (e^j_{\max} + \sum_{j'<j} w_{j'})$ and let j_x be the last job (according to sequence σ) that achieves Δ_x. Suppose there are $b \geq 0$ jobs before and $a \geq 0$ jobs after job j_x in $J_x^{\mathcal{S}^*}$. The following lemma gives for every job $j \in J_x^{\mathcal{S}^*} \backslash \{j_x\}$ an upper bound on the tariff spent in the interval $\left[S_j, C_j\right)$.

Lemma 4 *Consider an optimal schedule $\mathcal{S}^*$ for a given job permutation σ. For any job $j \in J_x^{\mathcal{S}^*} \backslash \{j_x\}$ a slot $[t, t+1) \in \left[S_j, C_j\right)$ is utilized if and only if the tariff $e(t)$ of $[t, t+1)$ satisfies the following upper bound:*

$$e(t) \leq e^{j_x}_{\max} + \sum_{j'=j}^{j_x-1} w_{j'}, \quad \forall j : j_x - b \leq j < j_x,$$

$$e(t) < e^{j_x}_{\max} - \sum_{j'=j_x}^{j-1} w_{j'}, \quad \forall j : j_x < j \leq j_x + a\,.$$

Proof Consider any job $j := j_x - \ell$ with $0 < \ell \leq b$. Suppose there is a job j for which a slot is utilized with cost (tariff) $e^j_{\max} > e^{j_x}_{\max} + \sum_{j'=j}^{j_x-1} w_{j'}$. Then $e^j_{\max} + \sum_{j'<j} w_{j'} > e^{j_x}_{\max} + \sum_{j'<j_x} w_{j'}$, which is a contradiction to the definition of job j_x. Thus, $e^j_{\max} \leq e^{j_x}_{\max} + \sum_{j'=j}^{j_x-1} w_{j'}$.

Now suppose that there is a slot $[t, t+1) \in \left[S_j, C_j\right)$ with cost $e(t) \leq e^{j_x}_{\max} + \sum_{j'=j}^{j_x-1} w_{j'}$ that is not utilized. By definition of $e^{j_x}_{\max}$, there must be a time slot $\left[t', t'+1\right) \in \left[S_{j_x}, C_{j_x}\right)$ with cost exactly $e^{j_x}_{\max}$. If we utilize slot $[t, t+1)$ instead of $\left[t', t'+1\right)$ and redistribute the processing volume accordingly without changing the processing order , then the difference in cost is at most $e(t) - e(t') - \sum_{j'=j_x-\ell}^{j_x-1} w_{j'} \leq 0$, because by Observation 2 and Lemma 2 the completion times of at least ℓ jobs ($j = j_x - \ell, \ldots, j_x - 1$ and maybe also j_x) decrease by one. This contradicts either the optimality of $\mathcal{S}^*$ or our assumption that $\mathcal{S}^*$ minimizes $\sum_{t=0}^{d_K-1} t \cdot y(t)$.

The proof of the statement for any job $j_x + \ell$ with $0 < \ell \leq a$ follows a similar argument, but now using the fact that for every job $j := j_x + \ell$ we have $e^j_{\max} < e^{j_x}_{\max} - \sum_{j'=j_x}^{j-1} w_{j'}$, because j_x was the last job with $e^j_{\max} + \sum_{j'<j} w_{j'} = \Delta_x$. □

Corollary 1 *If the interval $[S_j, C_j)$ for processing a job $j \in J_x^{\mathcal{S}^*} \backslash \{j_x\}$ intersects interval I_k but job j does not complete in I_k, i.e., $C_j > d_k$, then either all time slots in I_k are fully utilized or no time slot in I_k is utilized.*

To decide on an optimal utilization decision for the sub-schedule of the jobs in $J_x^{\mathcal{S}^*}$ of region $R_x^{\mathcal{S}^*}$, we need the following two lemmas.

Lemma 5 *If there is a partially utilized interval I_k in region $R_x^{\mathcal{S}^*}$, then (i) I_k is the last interval of $R_x^{\mathcal{S}^*}$, or (ii) j_x is the last job being processed in I_k and $e_k = e^{j_x}_{\max}$.*

Proof Suppose there exists a partially utilized interval I_k in region $R_x^{\mathcal{S}^*}$. Suppose j with $j \neq j_x$ is the last job that is processed in I_k, hence (ii) does not hold. Then either $C_j < d_k$, in which case $d_k = s_{k+1}$ is a split point and thus I_k is the last interval in the region, whence (i) is true. Or, we are in the situation of Corollary 1 and have a contradiction, because then I_k must be fully utilized.

Now suppose j_x is the last job being processed in I_k. If $C_{j_x} < d_k$, then again I_k is the last interval in the region. Otherwise $C_{j_x} \notin I_k$. If $e_k = e_{\max}^{j_x}$, then by the definition of j_x case (ii) of the lemma holds. If not, by definition of $e_{\max}^{j_x}$ we have $e_k < e_{\max}^{j_x}$. A simple exchange argument suffices to show that optimality of S^* implies that interval I_k comes after the last utilized "expensive" interval with cost $e_{\max}^{j_x}$. Hence, job j_x is processed in an expensive interval, then continued in I_k and is completed in yet a later interval. But then we can utilize an extra time slot in I_k instead of a time slot in the expensive interval, without increasing the completion time. This contradicts optimality, and, hence, $e_k = e_{\max}^{j_x}$, which completes the proof. □

Lemma 6 *There exists an optimal schedule $\mathcal{S}^*$ for a given job permutation σ with the following property. If the last interval I_k of a region $R_x^{\mathcal{S}^*}$ is only partially utilized then all time slots in $\left[S_{j_x}, C_{j_x}\right)$ with cost at most $e_{\max}^{j_x}$ are utilized.*

Proof Recall that $j_x + a$ is the last job being processed in the region, and hence, it is the last job processed in the partially utilized interval I_k.

Suppose there is a time slot $[t, t+1) \in \left[S_{j_x}, C_{j_x}\right)$ with cost at most $e_{\max}^{j_x}$ that is not utilized. If we utilize $[t, t+1)$ instead of the last utilized slot in I_k and redistribute the processing volume accordingly without changing the processing order, then by Observation 2 and Lemma 2 the difference in cost is $\delta_1 := e(t) - e_k - \sum_{j=j_x}^{j_x+a} w_j$. On the other hand, if we utilize one additional time slot in I_k instead of a time slot in $\left[S_{j_x}, C_{j_x}\right)$ with cost $e_{\max}^{j_x}$ and redistribute the processing volume accordingly without changing the processing order , then by Observation 2 and Lemma 2 the difference in cost is $\delta_2 := e_k - e_{\max}^{j_x} + \sum_{j=j_x}^{j_x+a} w_j$. We consider an optimal schedule $\mathcal{S}^*$, thus $\delta_1 \geq 0$ and $\delta_2 \geq 0$ which implies that $\delta_1 + \delta_2 = e(t) - e_{\max}^{j_x} \geq 0$. This is a contradiction if $e(t) < e_{\max}^{j_x}$. If $e(t) = e_{\max}^{j_x}$, then $\delta_1 = -\delta_2 = 0$, because we consider an optimal schedule $\mathcal{S}^*$. This, however, contradicts our assumption that $\mathcal{S}^*$ minimizes the value $\sum_{t=0}^{d_K-1} t \cdot y(t)$. □

We now show how to construct an optimal partial schedule for a given ordered job set in a given region in polynomial time.

Lemma 7 *Given a region R_x and an ordered job set J_x, we can find in polynomial time an optimal utilization decision for scheduling J_x within the region R_x, which does not contain any other split points than τ_x and τ_{x+1}, the boundaries of R_x.*

Proof Given R_x and J_x, we guess the optimal combination $\left(j_x, e_{\max}^{j_x}\right)$, i.e., we enumerate over all $O(nK)$ combinations and choose eventually the best solution.

We firstly assume that a partially utilized interval exists and it is the last one in R_x (case (i) in Lemma 5). Based on the characterization in Lemma 4 we find in polynomial time the slots to be utilized for the jobs $j_x - b, \ldots, j_x - 1$. This defines $C_{j_x-b}, \ldots, C_{j_x-1}$. Then starting job j_x at time C_{j_x-1}, we check intervals in the order given and utilize as much as needed of each next interval I_h if and only if $e_h \leq e_{\max}^{j_x}$, until a total of p_{j_x} time slots have been utilized for processing j_x. Lemma 6 justifies to do that. This yields a completion time C_{j_x}.

Starting at C_{j_x}, we use again Lemma 4 to find in polynomial time the slots to be utilized for processing the jobs $j_x + 1, \ldots, j_x + a$. This gives $C_{j_x+1}, \ldots, C_{j_x+a}$.

Now we assume that there is no partially utilized interval or we are in case (ii) of Lemma 5. Similar to the case above, we find in polynomial time the slots that $\mathcal{S}^*$ utilizes for the jobs $j_x - b, \ldots, j_x - 1$ based on Lemma 4. This defines $C_{j_x-b}, \ldots, C_{j_x-1}$. To find the slots to be utilized for the jobs $j_x + 1, \ldots, j_x + a$, in this case, we start at the end of R_x and go backwards in time. We can start at the end of R_x because in this case the last interval of R_x is fully utilized. This gives $C_{j_x+1}, \ldots, C_{j_x+a}$. Job j_x is thus to be scheduled in $\big[C_{j_x-1}, S_{j_x+1}\big)$. In order to find the right slots for j_x we solve a makespan problem in the interval $\big[C_{j_x-1}, S_{j_x+1}\big)$, which can be done in polynomial time (Theorem 4) and gives a solution that cannot be worse than what an optimal schedule $\mathcal{S}^*$ does.

If anywhere in both cases the utilized intervals can not be made sufficient for processing the job(s) for which they are intended, or if scheduling the jobs in the utilized intervals creates any intermediate split point,then this $\left(j_x, e_{\max}^{j_x}\right)$-combination is rejected. Hence, we have computed the optimal schedules over all $O(nK)$ combinations of $\left(j_x, e_{\max}^{j_x}\right)$ and over both cases of Lemma 5 concerning the position of the partially utilized interval.We choose the schedule with minimum total cost and return it with its value.This completes the proof. □

Now we are ready to prove our main theorem.

Proof (of Theorem 2) We give a dynamic program. Assume jobs are indexed according to the order given by σ. We define a state (j, t), where t is a potential split point $t \in \mathcal{P}$ and j is a job from the job set J, and a dummy job 0. The value of a state, $Z(j, t)$, is the optimal scheduling cost plus utilization cost for completing jobs $1, \ldots, j$ by time $t \in \mathcal{P}$. We apply the following recursion:

$$\begin{aligned} Z(j,t) &= \min\Big\{ Z(j',t') + z\big(\{j'+1,\ldots,j\},[t',t)\big) \mid t' \in \mathcal{P}, t' < t, j' \in J, j' \leq j \Big\},\\ Z(0,t) &= 0, \qquad \text{for any} t \in \mathcal{P},\\ Z(j,s_1) &= \infty, \qquad \text{for any } j > 0, \end{aligned}$$

where $z\big(\{j'+1,\ldots,j\},[t',t)\big)$ denotes the value of an optimal partial schedule for job set $\{j'+1, j'+2, \ldots, j\}$ in the region $[t', t)$, or ∞ if no such schedule exists. In case $j = j'$ there is no job to be scheduled in the interval $[t', t)$, whence we set $z\big(\{j'+1,\ldots,j\},[t',t)\big) = 0$. This models the option of leaving regions empty.

An optimal partial schedule can be computed in time polynomial in n and K as we have shown in Lemma 7. Hence, we compute $Z(j, t)$ for all $O(nK)$ states in polynomial time, which concludes the proof. □

Remark It is worth mentioning that the characterization of an optimal utilization decision above (Theorem 2) can be used to obtain a simple $(4 + \varepsilon)$-approximation for the *weighted* problem $1 \mid pmtn \mid \sum w_j C_j + E$. For the weighted problem, there may not exist a job sequence that is universally optimal for *all* utilization decisions (Epstein et al. 2012). However, there is a polynomial-time algorithm that computes a universal $(4+\varepsilon)$-approximation (Epstein et al. 2012). More precisely, the algorithm constructs a sequence of jobs which approximates the scheduling cost for any utilization decision within a factor at most $4 + \varepsilon$.

We can use this universal sequence for weighted jobs as the input for the algorithm in this section for computing the optimal utilization decision. Given an instance of problem

$1 \mid pmtn \mid \sum w_j C_j + E$, we compute a universal $(4+\varepsilon)$-approximate sequence σ by the algorithm by Epstein et al. (2012). Then, we determine the optimal utilization decision for σ by the algorithm presented in this section (Theorem 2) and obtain a schedule $\mathcal{S}$.

We argue that the schedule $\mathcal{S}$ has cost within a factor $4+\epsilon$ of the cost of an optimal schedule $\mathcal{S}^*$. Let $\mathcal{S}'$ denote the schedule which we obtain by changing the utilization decision of $\mathcal{S}$ to the utilization in an optimal schedule $\mathcal{S}^*$ (but keeping the scheduling sequence σ). The schedule $\mathcal{S}'$ has total cost no less than the original cost of $\mathcal{S}$. Furthermore, given the utilization decision in the optimal solution $\mathcal{S}^*$, the sequence σ approximates the scheduling cost of $\mathcal{S}^*$ within a factor of $4+\varepsilon$. Let $C_j(X)$ and $E(X)$ denote the completion time of job j and the total utilization cost in schedule X, respectively. Then the total cost of schedule $\mathcal{S}$ is

$$\begin{aligned}\sum_j w_j C_j(\mathcal{S}) + E(\mathcal{S}) \;&\leq\; \sum_j w_j C_j(\mathcal{S}') + E(\mathcal{S}') \;\leq\; (4+\varepsilon)\sum_j w_j C_j(\mathcal{S}^*) + E(\mathcal{S}^*)\\ &\leq\; (4+\varepsilon)\Bigg(\sum_j w_j C_j(\mathcal{S}^*) + E(\mathcal{S}^*)\Bigg).\end{aligned}$$

We conclude with the following corollary.

Corollary 2 *There is a* $(4+\varepsilon)$*-approximation algorithm for the scheduling problem* $1 \mid pmtn \mid \sum w_j C_j + E$.

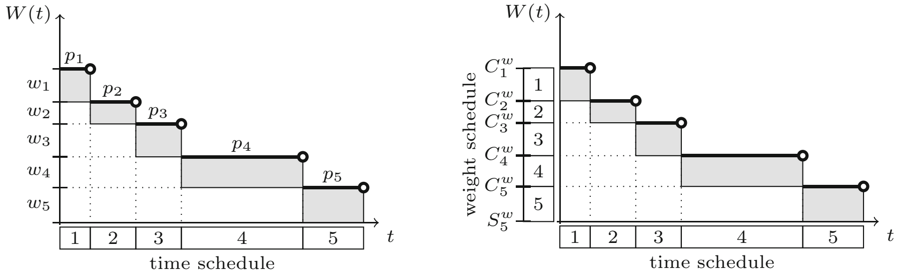
While this corollary for weighted jobs follows directly from the main result in this section on unweighted jobs (Theorem 2) and known results by Epstein et al. (2012), it improves substantially on the exponentially slower algorithm by Kulkarni and Munagala (2013). However, we present a best-possible approximation algorithm for the weighted setting in the following section.

3 A PTAS for minimizing the total weighted completion time

The main result of this section is a polynomial-time approximation scheme (PTAS) for minimizing the total weighted completion time with time-varying utilization cost. Unless P=NP, our PTAS is a best possible approximation algorithm since the problem is strongly NP-hard, even if there are only two tariffs (Wang et al. 2005).

Theorem 3 *There is a polynomial-time approximation scheme for the scheduling problem* $1 \mid pmtn \mid \sum w_j C_j + E$.

The roadmap to prove the theorem is as follows. We firstly describe a well-known two-dimensional representation of schedules in a 2D-Gantt chart and introduce some notation and preliminaries in Sect. 3.1. In Sect. 3.2, we present a dynamic programming (DP) algorithm with exponential running time. It draws inspiration from a known DP for scheduling on a machine of varying speed (Megow and Verschae 2018). As we mentioned earlier, a direct application of their DP does not seem possible. However, we can exploit methods developed by Megow and Verschae (2018) to reduce the number of states of our DP and, thus, reduce the running time to be polynomial in the input. This is explained in Sect. 3.3.



(a) 2D-Gantt chart of a schedule with 5 jobs with the function of remaining weights (bold).

(b) Same figure as (a), additionally depicting the weight schedule and the used notation.

Fig. 1 2D-Gantt chart (figure from Megow and Verschae (2018))

3.1 Preliminaries and scheduling in the weight-dimension

Consider a schedule $\mathcal{S}$ with completion times $C_j(\mathcal{S})$ for jobs $j \in J$. For any time t, let $W^{\mathcal{S}}(t) = \sum_{j:C_j>t} w_j$ denote the total weight of all jobs that complete in $\mathcal{S}$ strictly after t.

Notice that, by definition, $W^{\mathcal{S}}(t)$ is right-continuous, i. e., if $C_j(\mathcal{S}) = t$, the weight of j does not contribute to the remaining weight $W^{\mathcal{S}}(t)$. The scheduling cost for any schedule $\mathcal{S}$ can now be expressed equivalently in terms of completion times and the remaining weight function:

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

This connection can be seen well in the natural representation of a schedule in a classical 2D-Gantt chart using a time-dimension and a weight-dimension. It was shown originally by Eastman et al. (1964) and further used, e.g., in Goemans and Williamson (2000) and Megow and Verschae (2018).

In the following we elaborate more on this connection and introduce further notation. Here we follow closely the notation introduced by Megow and Verschae (2018) and use in Fig. 1 one of the visualizations from their paper. Figure 1a depicts a typical schedule with five jobs horizontally aligned along the time axis (we say, in the time-dimension); we refer to this standard schedule as *time-schedule*. Above it, we show the two-dimensional Gantt chart representation where each job j is represented by a rectangle with a length corresponding to the processing time p_j and a height corresponding to the job weight w_j. The function in bold is the remaining weight function $W(\cdot)$ of the schedule. The total scheduling cost equals the area under the remaining weight function, as also stated in Eq. (1).

Following the idea of Megow and Verschae (2018), we implicitly describe the completion time of a job j by the value of the function $W^{\mathcal{S}}$ at the time that j completes. We refer to this value as the *starting weight* S_j^w of job j. For a visualization consider Fig. 1b. It shows the same 5-job instance with the same time-schedule as Fig. 1a. In addition, it shows on the vertical axis (we say, in the weight dimension) a dual schedule which is obtained by projecting the 2D-Gantt chart to the vertical axis. In this schedule, which we call *weight-schedule*, each job has a length corresponding to its original weight w_j. The scheduling order, from bottom to top, is the reverse order of the time-schedule. In analogy to the time-dimension, the value $C_j^w := S_j^w + w_j$ is called *completion weight* of job j. Observe that in Fig. 1b we have $S_5^w = 0$, $C_5^w = S_4^w = w_5$, $C_4^w = S_3^w = w_5 + w_4$, etc.

Other terminologies, such as feasibility and idle time, also translate from the time-dimension to the weight-dimension. A weight-schedule is called *feasible* if no two jobs overlap and the machine is called *idle in weight-dimension* if there exists a point w in the weight-dimension with $w \notin [S_j^w, C_j^w]$ for all jobs $j \in J$.

A weight-schedule together with a utilization decision can be translated into a time-schedule by ordering the jobs in decreasing order of completion weights and scheduling them in this order in the time-dimension in the utilized time slots. For a given utilization decision, consider a weight-schedule $\mathcal{S}$ with completion weights $C_1^w > \cdots > C_n^w > C_{n+1}^w := 0$ and the corresponding completion times $0 =: C_0 < C_1 < \cdots < C_n$ for the jobs $j = 1, \ldots, n$. We define the *(scheduling) cost of a weight-schedule* $\mathcal{S}$ as $\sum_{j=1}^n (C_j^w - C_{j+1}^w) C_j$. This value equals $\sum_{j=1}^n \pi_j^{\mathcal{S}} C_j^w$, where $\pi_j^{\mathcal{S}} := C_j - S_j$, if and only if there is no idle weight. If there is idle weight, then the cost of a weight-schedule can only be greater, and we can safely remove idle weight without increasing the scheduling cost (Megow and Verschae 2018).

In summary, a time-schedule implies a corresponding weight-schedule of the same cost. On the other hand, a weight-schedule plus a utilization decision imply a time-schedule with a possibly smaller cost.

3.2 Dynamic programming algorithm

Let $\varepsilon > 0$. Firstly, we scale the input parameters so that all job weights w_j, $j = 1, \ldots, n$, and all tariffs e_k, $k = 1, \ldots, K$, are non-negative integers. Then, we apply standard geometric rounding to the weights to gain more structure on the input, i.e, we round up the weights to the next integer power of $(1+\varepsilon)$, losing at most a factor $(1+\varepsilon)$ in the objective value. Furthermore, we discretize the weight-space into intervals of exponentially increasing size: we define intervals $WI_u := [(1+\varepsilon)^{u-1}, (1+\varepsilon)^u)$ for $u = 1, \ldots, \nu$ with $\nu := \lceil \log_{1+\varepsilon} \sum_{j \in J} w_j \rceil$.

Consider a subset of jobs $J' \subseteq J$ and a partial weight-schedule of J'. In the dynamic program, the set J' represents the set of jobs at the beginning of a corresponding weight-schedule, i.e., if $j \in J'$ and $k \in J \backslash J'$, then $C_j^w < C_k^w$. However, the jobs in J' are scheduled at the end in a corresponding time-schedule. As discussed in Sect. 3.1, a partial weight-schedule $\mathcal{S}$ for the jobs in $J \backslash J'$ together with a utilization decision for these jobs can be translated into a time-schedule.

Let $\mathcal{F}_u := \{J_u \subseteq J : \sum_{j \in J_u} w_j \leq (1+\varepsilon)^u\}$ for $u = 1, \ldots, \nu$. The set $\mathcal{F}_u$ contains all the possible job sets J_u that can be scheduled in WI_u or before. Additionally, we define $\mathcal{F}_0$ to be the set that contains only the set of all zero-weight jobs $J_0 := \{j \in J : w_j = 0\}$. The following straightforward observation allows us to restrict to simplified completion weights.

Observation 3 *Consider an optimal weight-schedule in which the set of jobs with completion weight in* WI_u, $u \in \{1, \ldots, \nu\}$*, is exactly* $J_u \backslash J_{u-1}$ *for some* $J_u \in \mathcal{F}_u$ *and* $J_{u-1} \in \mathcal{F}_{u-1}$*. By losing at most a factor* $(1+\varepsilon)$ *in the objective value, we can assume that for all* $u \in \{1, \ldots, \nu\}$ *the completion weight of the jobs in* $J_u \backslash J_{u-1}$ *is exactly* $(1+\varepsilon)^u$.

The following observation follows from a simple interchange argument.

Observation 4 *There is an optimal time-schedule in which* J_0 *is scheduled completely after all jobs in* $J \backslash J_0$.

The dynamic program recursively constructs states $Z = [J_u, b, avg]$ and computes for every state a time point $t(Z)$ with the following meaning. A state $Z = [J_u, b, avg]$ with time point $t(Z)$ expresses that there is a feasible partial time-schedule $\mathcal{S}$ for the jobs in $J \backslash J_u$

with $J_u \in \mathcal{F}_u$ together with a utilization decision for the time interval $[0, t(Z))$ with total utilization cost at most b and for which the *average scheduling cost* is at most avg , i.e.,

$$\frac{1}{t(Z)} \cdot \int_0^{t(Z)} W^{\mathcal{S}}(t)\,\mathrm{d}t \ \leq avg.$$

We remark that even if partial schedule $\mathcal{S}$ only contains jobs in $J\backslash J_u$, the remaining weight function $W^{\mathcal{S}}$ still considers jobs in J_u with $W^{\mathcal{S}}(t(Z)) = \sum_{j \in J_u} w_j$. Further, $\mathcal{S}$ implies a weight-schedule for jobs in $J\backslash J_u$ with completion weights in the interval $[\sum_{j \in J_u} w_j, \sum_{j \in J} w_j]$. Note that $avg \cdot t(Z)$ is an upper bound on the total scheduling cost of $\mathcal{S}$ and that the average scheduling cost is non-increasing in time, because the remaining weight function $W^{\mathcal{S}}(t)$ is non-increasing in time.

In the *iteration for* u, we only consider states $[J_u, b, avg]$ with $J_u \in \mathcal{F}_u$. The states in the iteration for u are created based on the states from the iteration for $u + 1$. Initially, we only have the state $Z_\nu = [J, 0, 0]$ with $t(Z_\nu) := 0$. We start the dynamic program with $u = \nu - 1$, iteratively reduce u by one, and stop the process after the iteration for $u = 0$. In the iteration for u, the states together with their time points are constructed in the following way. Consider candidate sets $J_{u+1} \in \mathcal{F}_{u+1}$ and $J_u \in \mathcal{F}_u$ with $J_u \subseteq J_{u+1}$, a partial time-schedule $\mathcal{S}$ of $J\backslash J_u$, in which the set of jobs with completion weight (in the corresponding weight-schedule) in WI_{u+1} is exactly $J_{u+1}\backslash J_u$ and the set of jobs later than WI_{u+1} is exactly $J\backslash J_{u+1}$, two budgets b_1, b_2 with $b_1 \leq b_2$, and two bounds on the average scheduling cost avg_1, avg_2. Let $Z_1 = [J_{u+1}, b_1, avg_1]$ and $Z_2 = [J_u, b_2, avg_2]$ be the corresponding states. We know that there is a feasible partial schedule for the job set $J\backslash J_{u+1}$ up to time $t(Z_1)$ having average scheduling cost at most avg_1 and utilization cost at most b_1. By augmenting this schedule, we want to compute a minimum time point $t(Z_1, Z_2)$ that we associate with the link between Z_1 and Z_2 so that there is a feasible partial schedule for $J\backslash J_u$ that processes the jobs from $J_{u+1}\backslash J_u$ in the interval $[t(Z_1), t(Z_1, Z_2))$, has average scheduling cost at most avg_2, and utilization cost at most b_2. That is, $t(Z_1, Z_2)$ is the minimum makespan if we start with Z_1 and want to arrive at Z_2. For the computation of $t(Z_1, Z_2)$, we use the following subroutine.

Using Observation 3, we approximate the area under the remaining weight function $W^{\mathcal{S}}(t)$ for the jobs in $J_{u+1}\backslash J_u$ by $(1 + \varepsilon)^{u+1} \cdot (t(Z_1, Z_2) - t(Z_1))$, where $t(Z_1, Z_2)$ is the time point that we want to compute. Approximating this area gives us the flexibility to schedule the jobs in $J_{u+1}\backslash J_u$ in any order. However, we need that $avg_2 \cdot t(Z_1, Z_2)$ is an upper bound on the integral of the remaining weight function by time $t(Z_1, Z_2)$. That is, we want that

$$avg_2 \cdot t(Z_1, Z_2) \geq (1 + \varepsilon)^{u+1} \cdot t(Z_1, Z_2) + t(Z_1) \cdot (avg_1 - (1 + \varepsilon)^{u+1}).$$

Both the left-hand side and the right-hand side of this inequality are linear functions in $t(Z_1, Z_2)$. So, we can compute a smallest time point t^{LB} such that the right-hand side is greater or equal to the left-hand side for all $t(Z_1, Z_2) \geq t^{LB}$. If there is no such t^{LB}, then we set $t(Z_1, Z_2)$ to infinity and stop the subroutine. Otherwise, we know that our average scheduling cost at t^{LB} or later is at most avg_2. Let $E(p, [t_1, t_2))$ denote the total cost of the p cheapest slots in the time-interval $[t_1, t_2)$. We compute the earliest time point $t(Z_1, Z_2) \geq t^{LB}$ such that the set of jobs $J_{u+1}\backslash J_u$ can be feasibly scheduled in $[t(Z_1), t(Z_1, Z_2))$ having utilization cost not more than $b_2 - b_1$. That is, we set

$$t(Z_1, Z_2) = \min\left\{t \geq \max\{t(Z_1), t^{LB}\} : E(p(J_{u+1}\backslash J_u), [t(Z_1), t)) \leq b_2 - b_1\right\},$$

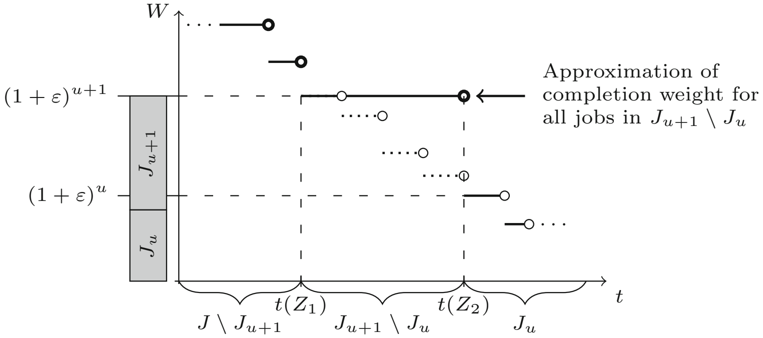


Fig. 2 Dynamic program. Illustration of an iteration for u

where $p(J') := \sum_{j \in J'} p_j$. The time point $t(Z_1, Z_2)$ can be computed in polynomial time by applying binary search in the interval $[\max\{t(Z_1), t^{LB}\}, d_K)$, since $E(p, [t_1, t_2))$ is a monotone function in t_2.

Given all possible states $[J_{u+1}, b_1, avg_1]$ of the iteration for $u+1$, the dynamic program enumerates all possible links for all these states to states $[J_u, b_2, avg_2]$ from the iteration for u fulfilling the above requirement on the candidate sets J_{u+1} and J_u, on the budgets b_1 and b_2, and on the average scheduling costs avg_1 and avg_2. For any such possible link (Z_1, Z_2) between states from the iteration for $u+1$ and u, we apply the above subroutine and associate the time point $t(Z_1, Z_2)$ with this link. Thus, the dynamic program associates several possible time points with a state $Z_2 = [J_u, b_2, avg_2]$ from the iteration for u. However, we only keep the link with the smallest associated time point $t(Z_1, Z_2)$ (ties are broken arbitrarily) and this defines the time point $t(Z_2)$ that we associate with the state Z_2. That is, for a state Z_2 from the iteration for u we define $t(Z_2) := \min\{t(Z_1, Z_2) \mid Z_1 \text{ is a state from the iteration for } u+1\}$. Figure 2 gives an illustration of the described iteration for u.

Let $E_{\max}$ be an upper bound on the total utilization cost in an optimal solution, e.g., the total cost of the first $p(J)$ finite-cost time slots. The dynamic program does not enumerate all possible budgets but only a polynomial number of them, namely budgets with integer powers of $(1+\eta_1)$ with $\eta_1 > 0$ determined later. That is, for the budget on the utilization cost, the dynamic program enumerates all values in

$$B := \{0, 1, (1+\eta_1), (1+\eta_1)^2, \ldots, (1+\eta_1)^{\omega_1}\} \text{ with } \omega_1 = \lceil \log_{1+\eta_1} E_{\max} \rceil.$$

The value η_1 will be chosen so that $(1+\eta_1)^{\omega_1} \leq (1+\varepsilon)$ and ω_1 is polynomial (see proof of Lemma 8 for the exact definition). Similarly, we observe that $(1+\varepsilon)^{\nu}$ is an upper bound on the average scheduling cost. The dynamic program does also only enumerate a polynomial number of possible average scheduling costs, namely integer powers of $(1+\eta_2)$ with $\eta_2 > 0$ also determined later. This means, for the average scheduling cost, the dynamic program enumerates all values in the set

$$AVG := \{0, 1, (1+\eta_2), (1+\eta_2)^2, \ldots, (1+\eta_2)^{\omega_2}\} \text{ with } \omega_2 = \lceil \nu \log_{1+\eta_2}(1+\varepsilon) \rceil.$$

As before, the value η_2 will be chosen so that $(1+\eta_2)^{\omega_2} \leq (1+\varepsilon)$ and ω_2 is polynomial. The dynamic program stops after the iteration for $u = 0$. Now, only the set of zero-weight jobs is not scheduled yet. For any state $Z = [J_0, b, avg]$ constructed in the iteration for $u = 0$, we append the zero-weight jobs starting at time $t(Z)$ and utilizing the cheapest slots, which is justified by Observation 4. We add the additional utilization cost to b. After this, we return the state $Z = [J_0, b, avg]$ and its corresponding schedule, which can be computed by backtracking and following the established links, with minimum total cost $b + avg \cdot t(Z)$. With this, we obtain the following result.

Lemma 8 *The dynamic program computes a* $(1 + O(\varepsilon))$*-approximate solution.*

Proof Consider an arbitrary iteration for u of the dynamic program and let $i = \nu - u$. We consider states $Z = [J_u, b, avg]$ with $J_u \in \mathcal{F}_u$, $b \in B$, and $avg \in AVG$ for which we construct the time points $t(Z)$. Let $Z_1^* = [J_{u+1}^*, b_1^*, avg_1^*]$ and $Z_2^* = [J_u^*, b_2^*, avg_2^*]$ with $J_{u+1}^* \in \mathcal{F}_{u+1}$ and $J_u^* \in \mathcal{F}_u$ be the states that represent an optimal solution $\mathcal{S}^*$ for which the set of jobs with completion weight in WI_{u+1} is exactly $J_{u+1}^* \backslash J_u^*$. Using Observation 3, at a loss of at most $(1 + \varepsilon)$, also in $\mathcal{S}^*$ we can approximate the area under the remaining weight function $W^{\mathcal{S}^*}(t)$ for the jobs in $J_{u+1}^* \backslash J_u^*$ by $(1 + \varepsilon)^{u+1} \cdot (t(Z_2^*) - t(Z_1^*))$. We claim the following invariant . The dynamic program constructs in iteration i a state $Z = [J_u, b, avg]$ with $J_u \in \mathcal{F}_u$, $b \in B$, and $avg \in AVG$ such that

(i) $J_u = J_u^*$,
(ii) $b \le (1 + \eta_1)^i \cdot b_2^*$,
(iii) $avg \le (1 + \eta_2)^i \cdot avg_2^*$, and
(iv) $t(Z) \le t(Z_2^*)$.

We prove this statement by induction on $i \in \{1, \dots, \nu\}$. Consider the first iteration of the dynamic program, in which we consider states with job sets from $\mathcal{F}_{\nu-1}$. Let $Z^* = [J_{\nu-1}^*, b^*, avg^*]$ be the state that corresponds to the optimal solution $\mathcal{S}^*$. The dynamic program also considers the job set $J_{\nu-1}^*$. Suppose we utilize the same slots that $\mathcal{S}^*$ utilizes for the jobs in $J \backslash J_{\nu-1}^*$ in the interval $[0, t(Z^*))$. Let b be the resulting utilization cost after rounding b^* up to the next value in B. With this, we know that $b \le (1 + \eta_1) \cdot b^*$. Furthermore, by our assumption, we know that the average scheduling cost of $\mathcal{S}^*$ up to time $t(Z^*)$ is $(1 + \varepsilon)^\nu$. Let avg be $(1 + \varepsilon)^\nu$ rounded up to the next value in AVG. Then we know that $avg \le (1 + \eta_2) \cdot avg^*$. The dynamic program also considers the state $Z = [J_{\nu-1}^*, b, avg]$. However, the dynamic program computes the *minimum* time point $t(Z_\nu, Z) \ge t^{LB}$ so that the set of jobs $J \backslash J_{\nu-1}^*$ can be feasibly scheduled in $[0, t(Z_\nu, Z))$ having utilization cost not more than b. This implies that $t(Z_\nu, Z) \le t(Z^*)$, which implies that $t(Z) \le t(Z^*)$. Note that $t^{LB} = 0$ for the specified values in Z.

Suppose, the statement is true for the iterations $1, 2, \dots, i - 1$. We prove that it is also true for iteration i, in which we consider job sets from $\mathcal{F}_u$. Again, let $Z_1^* = [J_{u+1}^*, b_1^*, avg_1^*]$ and $Z_2^* = [J_u^*, b_2^*, avg_2^*]$ with $J_{u+1}^* \in \mathcal{F}_{u+1}$ and $J_u^* \in \mathcal{F}_u$ be the states that represent $\mathcal{S}^*$. By our hypothesis, we know that the dynamic program constructs a state $Z_1 = [J_{u+1}, b_1, avg_1]$ with

(i) $J_{u+1} = J_{u+1}^*$,
(ii) $b_1 \le (1 + \eta_1)^{i-1} \cdot b_1^*$,
(iii) $avg_1 \le (1 + \eta_2)^{i-1} \cdot avg_1^*$, and
(iv) $t(Z_1) \le t(Z_1^*)$.

We augment this schedule in the following way. Suppose, we utilize the same slots that $\mathcal{S}^*$ utilizes for the jobs in $J_{u+1}^* \backslash J_u^*$ in the interval $[t(Z_1^*), t(Z_2^*))$. Let b_2 be the resulting total utilization cost after rounding up to the next value in B. Thus, there is a feasible schedule for $J \backslash J_u^*$ having utilization cost of at most

$$\begin{aligned} b_2 &\le (1 + \eta_1) \cdot (b_1 + b_2^* - b_1^*) \\ &\le (1 + \eta_1)^i \cdot (b_1^* + b_2^* - b_1^*) \end{aligned}$$

$$= (1 + \eta_1)^i \cdot b_2^*.$$

The new average scheduling cost after rounding to the next value in AVG is by construction of the schedule

$$\begin{aligned} avg_2 &\leq (1 + \eta_2) \cdot \frac{avg_1 \cdot t(Z_1) + (1+\varepsilon)^{u+1} \cdot \big(t(Z_2^*) - t(Z_1)\big)}{t(Z_2^*)} \\ &\leq (1 + \eta_2)^i \cdot \frac{avg_1^* \cdot t(Z_1) + (1+\varepsilon)^{u+1} \cdot \big(t(Z_2^*) - t(Z_1)\big)}{t(Z_2^*)}, \end{aligned} \tag{2}$$

where the second inequality follows from applying the induction hypothesis (iii). Further, we use induction hypothesis (iv), namely, $t(Z_1^*) - t(Z_1) \geq 0$, to derive an upper bound on the enumerator of the right-hand side in Inequality (2). We use the fact that $avg_1^* \geq (1+\varepsilon)^{u+1}$ and multiply both sides of the inequality with $(t(Z_1^*) - t(Z_1))$ and then add $t(Z_2^*) \cdot (1+\varepsilon)^{u+1}$. Rearranging terms implies

$$\begin{aligned} &avg_1^* \cdot t(Z_1) + (1+\varepsilon)^{u+1} \cdot \big(t(Z_2^*) - t(Z_1)\big) \\ &\quad \leq avg_1^* \cdot t(Z_1^*) + (1+\varepsilon)^{u+1} \cdot \big(t(Z_2^*) - t(Z_1^*)\big). \end{aligned}$$

Applying this inequality to (2) yields

$$\begin{aligned} avg_2 &\leq (1 + \eta_2)^i \cdot \frac{avg_1^* \cdot t(Z_1^*) + (1+\varepsilon)^{u+1} \cdot \big(t(Z_2^*) - t(Z_1^*)\big)}{t(Z_2^*)} \\ &= (1 + \eta_2)^i \cdot avg_2^*. \end{aligned}$$

Here, the equality follows from the definitions of an optimal schedule and avg_2^*, and from Observation 3.

The dynamic program also considers the link between the state Z_1 and $Z_2 := [J_u^*, b_2, avg_2]$. We first observe that $t^{LB} \leq t(Z_2^*)$, since

$$avg_2 \cdot t(Z_2^*) \geq avg_1 \cdot t(Z_1) + (1+\varepsilon)^{u+1} \cdot (t(Z_2^*) - t(Z_1))$$

by construction of avg_2. Furthermore, we observe that $b_2 - b_1 \geq b_2^* - b_1^*$ by construction of b_2. These two facts together with $t(Z_1) \leq t(Z_1^*)$ imply that $t(Z_1, Z_2) \leq t(Z_2^*)$, which implies that $t(Z_2) \leq t(Z_2^*)$.

To complete the proof, we need to specify the parameters η_1 and η_2. We want that $(1 + \eta_i)^\nu \leq (1+\varepsilon)$ for $i = 1, 2$. We claim that for a given $\nu \geq 1$ there exists an $\bar{\eta} > 0$ such that for all $\eta \in (0, \bar{\eta}]$ we have $(1+\eta)^\nu \leq 1 + 2\nu\eta$. Consider the function $f(\eta) := (1+\eta)^\nu - 1 - 2\nu\eta$. We have that $f(0) = 0$ and $f'(\eta) < 0$ for $\eta \in [0, 2^{1/(\nu-1)} - 1)$. This shows the claim. Hence, we choose $\eta_i = \min\{\frac{\varepsilon}{2\nu}, 2^{1/(\nu-1)} - 1\}$ for $i = 1, 2$. This shows the statement of the lemma and that the size of B as well as the size of AVG are bounded by a polynomial in the size of the input. □

We remark that the given DP works for more general utilization cost functions $e : \mathbb{N} \to \mathbb{Q}_{\geq 0}$ than considered here in the paper. As argued in the proof, it is sufficient for the DP that there is a function $E(p, [t_1, t_2))$ that outputs in polynomial time for a given time interval $[t_1, t_2)$ and a given $p \in \mathbb{Z}_{\geq 0}$ the total cost of the p cheapest slots in $[t_1, t_2)$.

We also remark that the running time of the presented DP is exponential, because the size of the sets $\mathcal{F}_u$ are exponential in the size of the input. However, in the next section we show that we can trim the sets $\mathcal{F}_u$ down to ones of polynomial size at an arbitrarily small loss in the performance guarantee.

3.3 Trimming the state space

The set $\mathcal{F}_u$, containing all possible job sets J_u, is of exponential size, and so is the DP state space. In the context of scheduling with variable machine speed, it has been shown in Megow and Verschae (2018) how to reduce the set $\mathcal{F}_u$ for a similar DP (without utilization decision, though) to a set $\tilde{\mathcal{F}}_u$ of polynomial size at only a small loss in the objective value. In general, such a procedure is not necessarily applicable to our setting because of the different objective involving additional utilization cost and the different decision space. However, the compactification in Megow and Verschae (2018) holds *independently of the speed of the machine* and, thus, independently of the utilization decision of the DP (by interpreting nonutilization as speed 0 and utilization as speed 1). Hence, we can apply it to our cost-aware scheduling framework and obtain a PTAS. We explain in the following, why the building blocks for the trimming procedure by Megow and Verschae (2018) can be applied to our DP for obtaining the set $\tilde{\mathcal{F}}_u$ of polynomial size.

Light Jobs The first building block for the trimming procedure is a classification of the jobs based on their weights.

Definition 2 Given a weight schedule and a job $j \in J$ with starting weight $S_j^w \in WI_u$, we call job j *light* if $w_j \leq \varepsilon^2 |WI_u|$, otherwise j is called *heavy*.

This classification enables us to structure near-optimal solutions. To impose structure on the set of light jobs, Megow and Verschae (2018) describe the following routine for a given weight schedule $\mathcal{S}$. First, remove all light jobs from $\mathcal{S}$ and move the remaining jobs within each interval WI_u so that the idle weight in WI_u is consecutive. Then, schedule the light jobs according to the *reverse Smith's rule*, that is, for each $u = 1, \ldots, \nu$ and each idle weight $w \in WI_u$, process at w a light job j that maximizes p_j/w_j. Eventually, shift the processing of each interval WI_u to WI_{u+1}, which delays the completion of every job by at most a factor of $(1+\varepsilon)^2$. This delay allows us to completely process every light job in the weight interval where it starts processing. It can be shown that the cost of the resulting schedule is at most a factor of $1 + O(\varepsilon)$ greater than the cost of $\mathcal{S}$, which brings us to the following structural statement.

Lemma 9 (Megow and Verschae 2018) *At a loss of a factor of* $1 + O(\varepsilon)$ *in the scheduling cost, we can assume the following. For a given interval* WI_u*, consider any pair of light jobs* j, k*. If both jobs start in* WI_u *or later and* $p_k/w_k \leq p_j/w_j$*, then* $C_j^w \leq C_k^w$*.*

We remark, that Lemma 9 holds independently of the speed of the machine, as pointed out in Megow and Verschae (2018). This means that at a loss of a factor of $1 + O(\varepsilon)$ in the scheduling cost we can assume also for our problem that light jobs are scheduled according to reverse Smith's rule in the weight-dimension, which holds independently of our actual utilization decision.

Localization We now localize jobs in the weight-dimension to gain more structure. That is, we determine for every job $j \in J$ two values r_j^w and d_j^w such that, independently of our actual utilization decision, j is scheduled in the weight-dimension completely within $\left[r_j^w, d_j^w\right)$ in some $(1 + O(\varepsilon))$-approximate weight-schedule (in terms of the scheduling cost in weight-dimension). We call r_j^w and d_j^w the *release-weight* and the *deadline-weight* of job j, respectively.

Lemma 10 (Megow and Verschae 2018) *We can compute in polynomial time values* r_j^w *and* d_j^w *for each* $j \in J$ *such that:*

(i) *there exists a* $(1 + O(\varepsilon))$*-approximate weight-schedule (in terms of the scheduling cost) that processes each job* j *within* $[r_j^w, d_j^w)$*,*
(ii) *there exists a constant* $s \in O(\log(1/\varepsilon)/\varepsilon)$ *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.*

This lemma enables us to localize all jobs in J in polynomial time and independent of our actual utilization decision, as guaranteed by property (iv).

Compact search space Based on the localization of jobs in weight space, we can cut the number of different possibilities for a candidate set J_u in iteration u of our DP down to a polynomial number. That is, we replace the set $\mathcal{F}_u$ by a polynomially-sized set $\tilde{\mathcal{F}}_u$. Instead of describing all sets $S \in \tilde{\mathcal{F}}_u$ explicitly, we give all possible complements $R = J \backslash S$ and collect them in a set $\mathcal{D}_u$, where a set $R \in \mathcal{D}_u$ represents a possible set of jobs having completion weights in WI_{u+1} or later. Obviously, a set $R \in \mathcal{D}_u$ must contain all jobs $j \in J$ having a release weight $r_j^w \geq (1+\varepsilon)^u$. Furthermore, we know that $d_j^w \geq (1+\varepsilon)^{u+1}$ is necessary for job j to be in a set $R \in \mathcal{D}_u$. Following property (ii) in Lemma 10, we thus only need to decide about the jobs having a release weight $r_j^w = (1+\varepsilon)^i$ with $i \in \{u+1-s, \ldots, u-1\}$. An enumeration over basically all possible job sets for each $i \in \{u+1-s, \ldots, u-1\}$ gives the following desired result.

Lemma 11 (Megow and Verschae 2018) *For each u, we can construct in polynomial time a set* $\tilde{\mathcal{F}}_u$ *that satisfies the following:*

(i) *there exists a* $(1 + O(\varepsilon))$*-approximate weight-schedule (in terms of the scheduling cost) in which the set of jobs with completion weight at most* $(1+\varepsilon)^u$ *belongs to* $\tilde{\mathcal{F}}_u$*,*
(ii) *the set* $\tilde{\mathcal{F}}_u$ *has cardinality at most* $2^{O\left(\log^3(1/\varepsilon)/\varepsilon^2\right)}$*, and*
(iii) *the set* $\tilde{\mathcal{F}}_u$ *is completely independent of the speed of the machine.*

Again, Property (iii) implies that we can construct the set $\tilde{\mathcal{F}}_u$ independently of our utilization decision.

To complete the proof of Theorem 3 it remains to determine the running time of the DP. The DP has ν iterations, where in each iteration for at most $2^{O\left(\log^3(1/\varepsilon)/\varepsilon^2\right)} \cdot |B| \cdot |AVG|$ previous states at most $2^{O\left(\log^3(1/\varepsilon)/\varepsilon^2\right)} \cdot |B| \cdot |AVG|$ many links to new states are considered. Therefore, the running time complexity of our DP is $\nu \cdot (2^{O\left(\log^3(1/\varepsilon)/\varepsilon^2\right)} \cdot |B| \cdot |AVG|)^2$, which is bounded by a polynomial in the size of the input.

4 Minimizing the makespan on unrelated machines

Finally we derive positive results for the problem of minimizing makespan with utilization cost on unrelated machines. The standard scheduling problem without utilization cost $R \mid pmtn \mid C_{\max}$ can be solved optimally in polynomial time by solving a linear program as was shown by Lawler and Labetoulle (1978). We show that the problem complexity does not increase significantly when taking into account time-varying utilization cost.

Consider the preemptive makespan minimization problem with utilization cost. Recall that by our problem definition we can use every machine in a utilized time slot and pay only once. Thus, it is sufficient to find an optimal utilization decision for solving this problem, because we can use the polynomial-time algorithm in Lawler and Labetoulle (1978) to find the optimal schedule within these slots.

Observation 5 *Given the set of time slots utilized in an optimal solution, we can compute an optimal schedule in polynomial time.*

Given an instance of our problem, let Z be the optimal makespan when scheduling *without* utilization cost. Notice that Z is not necessarily integral. To determine an optimal utilization decision, we use the following observation.

Observation 6 *Given an optimal makespan $C^*_{\max}$ for $R \mid pmtn \mid C_{\max} + E$, an optimal schedule utilizes the $\lceil Z \rceil$ cheapest slots before $\lceil C^*_{\max} \rceil$.*

Note that we must pay full tariff for a used time slot, no matter how much it is utilized, and so does an optimal solution. In particular, this holds for the last utilized slot. Hence, it remains to compute an optimal value $C^* := \lceil C^*_{\max} \rceil$.

To do so, we restrict our attention to relevant intervals $I_k = [s_k, d_k), k \in \{1, \dots, K\}$, with $s_k \geq \lceil Z \rceil$ and compute for each such interval an optimal point in time for C^* assuming that $C^* \in I_k$.

Consider a relevant interval I_k. If we knew the precise value of C^* in I_k, then we would utilize the $\lceil Z \rceil$ cheapest time slots before C^*, which is optimal by Observation 6. We cannot afford to enumerate all time points in I_k to determine C^*. However, we can make the following observation.

Observation 7 *Consider a feasible schedule with latest completion time $C \in I_k$ that utilizes the $\lceil Z \rceil$ cheapest time slots before C. Any utilized time slot of cost e with $e > e_k + 1$ can be replaced by a time slot from I_k (if available) leading to a solution of less total cost.*

To determine the optimal makespan within I_k, we let $C^* = s_k$ and determine the schedule utilizing the $\lceil Z \rceil$ cheapest time slots before C^*. If there is no utilized time slot of cost e with $e > e_k + 1$, then C^* is optimal in I_k. If there is such a time slot then do the following. Let U be the set of utilized time points in intervals of cost e with $e > e_k + 1$, and let $a := d_k - s_k$ be the number of available time slots in interval I_k. Select the $u = \min\{a, |U|\}$ most expensive slots from the set U, unutilize them and utilize all time slots in $[s_k, \dots, s_k + u)$ instead, whence C^* is $s_k + u$.

Theorem 4 *The scheduling problem $R \mid pmtn \mid C_{\max} + E$ can be solved in polynomial time in the order of $O(K^2)$ plus the running time for solving $R \mid pmtn \mid C_{\max}$ without utilization cost* (Lawler and Labetoulle 1978).

Proof The algorithm computes Z, the optimal makespan when scheduling without utilization cost, using the algorithm by Lawler and Labetoulle (1978). Then it determines for each interval I_k, $k \in \{1, 2, \dots, K\}$ the best possible makespan within I_k as described above and chooses the one of minimum total cost, i.e., makespan plus utilization cost. The procedure takes at most $O(K)$ operations per interval, hence in total $O(K^2)$.

It remains to argue that for each interval I_k, the choice of C^* is optimal. For $C^* = s_k$ this is obviously true, since all utilized time slots have cost at most $e_k + 1$ and, thus, no change in the utilization decision can decrease the total cost.

Now suppose that $C^* > s_k$ and suppose for contradiction that this is not the optimal choice for the makespan in interval I_k. If the optimal makespan in I_k is $C^* - x$ for some appropriate $x > 0$, then, by the way we have determined C^*, there must exist a utilized time slot with cost larger than $e_k + 1$. Replacing it will give a schedule with less total cost, by Observation 7, contradicting optimality. If the optimal makespan in I_k is $C^* + x$, for some appropriate $x > 0$, then in the optimal solution x time slots must be utilized in the interval

I_k, which our algorithm did not utilize. Our algorithm used instead time slots of cost at most $e_k + 1$. Thus, our algorithm has total cost not larger than the optimal schedule since its makespan is less by x and its utilization cost larger by not more than x. This completes the proof. □

5 Conclusion

We investigate basic scheduling problems within the framework of time-varying costs or tariffs, where the processing of jobs causes some time-dependent cost in addition to the usual quality-of-service measure. We presented optimal algorithms and best possible approximation algorithms for the scheduling objectives of minimizing the makespan on unrelated machines and the sum of (weighted) completion times on a single machine.

While our work closes the problems under consideration from an approximation point of view, it leaves open the approximability of multi-machine settings for the min-sum objective. Further research may also ask for the complexity status when assuming that jobs have different release dates and for other natural objective functions such as average and maximum flow-time.

Our unrelated machine model is time-slot based, that is, a utilization decision is made for a time slot and then all machines in this time slot are available. No less relevant appears to be the model with *machine-individual* tariffs, that is, a utilization decision is made for a time slot on each machine individually. It is not difficult to see that a standard LP can be adapted for optimally solving $R \mid pmtn, r_j \mid C_{\max}$ with fractional utilization cost. However, if time slots can be utilized only integrally then the integrality gap for the simple LP is unbounded and the problem seems much harder.

Time-varying costs or tariffs appear in many applications in practice but they have hardly been investigated from a theoretical perspective. With our work we settle the complexity status and approximability status for very classical scheduling problems. We hope to foster further research on this framework of time-varying costs or tariffs. We emphasize that the framework is clearly not restricted to cost-aware scheduling problems. Virtually any problem in which scarce resources are to be rented from some provider lends itself to be modelled in this way, with (vehicle) routing problems as a directly appealing example.

Funding Open Access funding enabled and organized by Projekt DEAL. This research was supported by the German Science Foundation (DFG) under contract ME 3825/1 and by Netherlands Organisation for Scientific Research (NWO) Gravitation Programme Networks 024.002.003.

References

Albers, S. (2010). Energy-efficient algorithms. *Communications of the ACM*, *53*(5), 86–96.

Bansal, N., & Pruhs, K. (2014). The geometry of scheduling. *SIAM Journal on Computing*, *43*(5), 1684–1698.

Chen, B., & Zhang, X. (2019). Scheduling with time-of-use costs. *European Journal of Operational Research*, *274*(3), 900–908.

Chen, L., Megow, N., Rischke, R., Stougie, L., & Verschae, J. (2015). Optimal algorithms and a PTAS for cost-aware scheduling. *Proceedings of the 40th international symposium on mathematical foundations of computer science (MFCS)*, LNCS (Vol. 9235, pp. 211–222). Springer.

Cheung, M., & Shmoys, D. B. (2011). A primal-dual approximation algorithm for min-sum single-machine scheduling problems. In *Proceedings of the 14th international workshop on approximation, randomization, and combinatorial optimization (APPROX),* LNCS (Vol. 6845, pp. 135–146). Springer.

Cheung, M., Mestre, J., Shmoys, D. B., & Verschae, J. (2017). A primal-dual approximation algorithm for min-sum single-machine scheduling problems. *SIAM Journal on Discrete Mathematics*, *31*(2), 825–838.

Eastman, W. L., Even, S., & Isaac, M. (1964). Bounds for the optimal scheduling of n jobs on m processors. *Management Science*, *11*(2), 268–279.

Epstein, L., Levin, A., Marchetti-Spaccamela, A., Megow, N., Mestre, J., Skutella, M., et al. (2012). Universal sequencing on an unreliable machine. *SIAM Journal on Computing*, *41*(3), 565–586.

Fang, K., Uhan, N. A., Zhao, F., & Sutherland, J. W. (2016). Scheduling on a single machine under time-of-use electricity tariffs. *Annals of Operations Research*, *238*(1), 199–227.

Goemans, M. X., & Williamson, D. P. (2000). Two-dimensional gantt charts and a scheduling algorithm of lawler. *SIAM Journal on Discrete Mathematics*, *13*(3), 281–294.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. H. G. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, *5*, 287–326.

Hillier, F. S., & Lieberman, G. J. (2014). *Introduction to operations research* (10th ed.). New York: McGraw-Hill.

Höhn, W., & Jacobs, T. (2015). On the performance of Smith's rule in single-machine scheduling with nonlinear cost. *ACM Transactions on Algorithms*, *11*(4), No. 25.

Höhn, W., Mestre, J., & Wiese, A. (2018). How unsplittable-flow-covering helps scheduling with job-dependent cost functions. *Algorithmica*, *80*(4), 1191–1213.

Kulkarni, J., & Munagala, K. (2013). Algorithms for cost-aware scheduling. In *Proceedings of the 10th international workshop on approximation and online algorithms (WAOA)*, LNCS (Vol. 7846, pp. 201–214). Springer.

Lawler, E. L., & Labetoulle, J. (1978). On preemptive scheduling of unrelated parallel processors by linear programming. *Journal of the ACM*, *25*(4), 612–619.

Lee, C. Y. (2004). Machine scheduling with availability constraints. In J. Y. T. Leung (Ed.), *Handbook of scheduling*. Boca Raton: CRC Press.

Megow, N., & Verschae, J. (2018). Dual techniques for scheduling on a machine with varying speed. *SIAM Journal on Discrete Mathematics*, *32*(3), 1541–1571.

Megow, N., & Verschae, J. (2009). Short note on scheduling on a single machine with one non-availability period. Matheon Preprint 533. urn:nbn:de:0296-matheon-5392.

Rischke, R. (2016). *Deterministic, stochastic, and robust cost-aware scheduling*. Ph.D. thesis, Technical University of Munich.

Taha, H. A. (2007). *Operations research: An introduction*. Upper Saddle River: Pearson/Prentice Hall.

Talluri, K. T., & Van Ryzin, G. J. (2006). *The theory and practice of revenue management* (Vol. 68). New York: Springer.

Wan, G., & Qi, X. (2010). Scheduling with variable time slot costs. *Naval Research Logistics*, *57*, 159–171.

Wang, G., Sun, H., & Chu, C. (2005). Preemptive scheduling with availability constraints to minimize total weighted completion times. *Annals of Operations Research*, *133*, 183–192.

Yuan, J. J., Lin, Y. X., Ng, C. T., & Cheng, T. C. E. (2007). Approximability of single machine scheduling with fixed jobs to minimize total completion time. *European Journal of Operational Research*, *178*(1), 46–56.