J Sched (2018) 21:505–516
https://doi.org/10.1007/s10951-017-0552-y

Models and algorithms for energy-efficient scheduling with immediate start of jobs

Akiyoshi Shioura[1] · **Natalia V. Shakhlevich**[2] · **Vitaly A. Strusevich**[3] · **Bernhard Primas**[2]

Published online: 12 December 2017

Abstract We study a scheduling model with speed scaling for machines and the immediate start requirement for jobs. Speed scaling improves the system performance, but incurs the energy cost. The immediate start condition implies that each job should be started exactly at its release time. Such a condition is typical for modern Cloud computing systems with abundant resources. We consider two cost functions, one that represents the quality of service and the other that corresponds to the cost of running. We demonstrate that the basic scheduling model to minimize the aggregated cost function with n jobs is solvable in $O(n \log n)$ time in the single-machine case and in $O(n^2 m)$ time in the case of m parallel machines. We also address additional features, e.g., the cost of job rejection or the cost of initiating a machine. In the case of a single machine, we present algorithms for minimizing one of the cost functions subject to an upper bound on the value of the other, as well as for finding a Pareto-optimal solution.

✉ Natalia V. Shakhlevich
N.Shakhlevich@leeds.ac.uk

Akiyoshi Shioura
shioura.a.aa@m.titech.ac.jp

Vitaly A. Strusevich
V.Strusevich@greenwich.ac.uk

Bernhard Primas
scbjp@leeds.ac.uk

[1] Department of Industrial Engineering and Economics, School of Engineering, Tokyo Institute of Technology, Tokyo, Japan

[2] School of Computing, University of Leeds, Leeds, UK

[3] Department of Mathematical Sciences, Faculty of Architecture, Computing & Humanities, University of Greenwich, Old Royal Naval College, Park Row, London SE10 9LS, UK

Keywords Speed scaling · Energy minimization · Immediate start · Bicriteria optimization

1 Introduction

In this paper, we study scheduling models that address two important aspects of modern computing systems: machine speed scaling for time and energy optimization and the requirement to start jobs immediately at the time they are submitted to the system. The first aspect, speed scaling, has been the subject of intensive research since the 1990s, see Yao et al. (1995), and has become particularly important recently, with the increased attention to energy-saving demands, see surveys Albers (2009, 2010a), Jing et al. (2013), Gerards et al. (2016). It reflects the ability of modern computing systems to change their clock speeds through the technique known as Dynamic Voltage and Frequency Scaling (DVFS). The higher the speed, the better the performance from users' perspective, but the energy usage and other computation costs do increase. The goal is to select the right speed value from the full spectrum of speed to achieve a desired trade-off between performance and energy. DVFS techniques have been successfully applied in Cloud data centers to reduce the energy usage, see, e.g., VonLaszewski et al. (2009), Wu et al. (2014), DoLago et al. (2011).

The second aspect, the immediate start condition, is motivated by the advancements of modern Cloud computing systems, and it is widely accepted by practitioners. This feature is not typical for the traditional scheduling research dealing with scenarios arising from manufacturing. In such systems, jobs compete for limited resources. They often have to wait until resources become available, and job starting times can be delayed if the system is busy.

In modern computing systems (Clouds and data centers), processing units are no longer scarce resources, but quite opposite, abundant resources; see, e.g., Kushida et al. (2015). Clouds give the illusion of infinite computing resources available *on demand*. Cloud providers agree with customers on a service-level agreement (SLA) and sell computing services to customers as utilities. Special mechanisms allow Cloud providers to ensure that the actual demand for resources is met at practically any point in time (Armbrust et al. 2009, 2010; Jennings and Stadler 2015). In the modern competitive market, Cloud providers achieve *high availability* of resources, promise their customers *instant* access to resources and allow customers to *monitor* how that promise is kept (Aceto et al. 2013). These features are unprecedented in the history of IT and have now become a standard.

The infrastructure for Cloud computing systems is provided by data centers. Data centers execute a large number of computing processes (which we call jobs from now on). In order to guarantee on-demand access, the execution of a job needs to be started immediately upon its submission to the system. Customers experiencing waiting times in order to get their jobs started become unsatisfied with the service and are likely to change the provider next time (Armbrust et al. 2010). It is therefore in the interest of providers to start job execution as soon as jobs are submitted to the system. This phenomenon is our motivation for what we call the *immediate start* condition.

The optimization criteria are typically of two types: those related to the system performance and the quality-of-service (QoS) provision, as well as those related to the operational cost of the processing system. The criteria of the first type may represent the mean flow times of jobs, total (or mean) tardiness or a more general function F defined as the sum of penalty functions of job completion times. The second objective G is the sum of operational costs for using individual resources, each of which depends on the time the resource is used. It can be linear, to model the monetary resource usage cost, or convex, to model the energy consumption cost.

Observing the immediate start condition is one of the key priorities for resource providers, and it is usually included in the QoS protocols. The case study presented by Garg et al. (2013) characterizes a possible waste of execution time due to the resource unavailability. It is estimated as low as 0.5% for customers of Amazon EC2 and 0.1% for Windows Azure customers. The Rackspace web hosting company guarantees 100% network availability and 99.95–99.99% platform availability; see Rackspace (2015); in reality Rackspace often achieves 100% availability of its resources (Garg et al. 2013). Nowadays, special software is being developed in order to strengthen service-level agreements (SLAs) for customers by fixing the maximum response time, which can be as small as a few seconds (Iqbal et al. 2009). Due to a strong competition in the area, the customers choose providers who are prepared to demonstrate that handling the submitted jobs is their top priority.

The immediate start requirement is not a seemingly strong assumption, but a fact of today's life. It is widely accepted in distributed computing, but generally overlooked by the scheduling community, where the traditional perception remains, that of limited resources and acceptable delayed starting times.

In this paper, we initiate the study of the immediate start off-line scheduling models, assuming that accurate job characteristics can be available in advance through historical analysis, predicting techniques or a combination of both; see, e.g., Moreno et al. (2014) for off-line scenarios in Cloud computing. To satisfy the immediate start condition, we recommend a policy of changing the processing speeds, so that a certain measure of the schedule quality and the cost of speeds (normally understood as energy) are both taken into consideration. The owners of submitted tasks and the providers of processing facilities both want an early completion of tasks, as recorded in the SLAs, and this in practice leads to a no-preemption requirement; see Tian and Zhao (2015). We understand that the models that we address in this paper are rather ideal and simple, but we see our work as a necessary step that should be made before more advanced and practically relevant models are investigated.

In the remainder of this section, we provide a formal definition of the model under study and discuss the relevant literature.

1.1 Definitions and notation

Formally, in the models under consideration, we are given a set of jobs $N = \{1, 2, \ldots, n\}$. A job $j \in N$ can be understood as a computational task characterized by its volume or work γ_j, measured in millions of instructions to be performed on a computing device. Each job $j \in N$ is associated with a release date r_j before which it is not available. For completeness, assume that $r_{n+1} = +\infty$.

It is also possible that job j is given a due date d_j, before which it is desired to complete that job, and/or a deadline $\bar{d}_j$. The due dates d_j are seen as "soft", i.e., it is possible to violate them, and usually a certain penalty is associated with such a violation. On the other hand, the deadlines $\bar{d}_j$ are "hard", i.e., in any feasible schedule job j must be completed by time $\bar{d}_j$. If for a job $j \in N$ no deadline is given, we assume that $\bar{d}_j = +\infty$. Additionally, job j can be given weight w_j, which indicates its relative importance.

Each job is processed without preemption either on a single machine or on one of m parallel machines. For processing job $j \in N$, the corresponding machine has to be assigned speed s_j, measured in millions of instructions performed per time unit, so that the actual processing time of job j is defined by

$$p_j = \frac{\gamma_j}{s_j}. \tag{1}$$

The main feature of the models that we study in this paper is the requirement of the immediate start or dispatch of each job, i.e., job $j \in N$ must start its processing immediately upon its arrival at time r_j. Without loss of generality, we assume that all release dates are distinct and the jobs are numbered in increasing order of their release dates, i.e.,

$$r_1 < r_2 < \cdots < r_n. \tag{2}$$

For a fixed schedule, let C_j denote the completion time of job $j \in N$. Provided that job $j \in N$ is processed at speed s_j, we have that

$$C_j = r_j + p_j,$$

where the actual processing time p_j is defined by (1).

We associate each job $j \in N$ with two types of penalties:

(i) a traditional scheduling cost function $f_j\left(C_j\right)$, where f_j is a non-decreasing function, so that $f_j\left(C_j\right)$ represents the penalty for completing job j at time C_j ; the total cost is then

$$F = \sum_{j=1}^{n} f_j\left(C_j\right) = \sum_{j=1}^{n} f_j\left(r_j + p_j\right);$$

(ii) the speed cost function $g_j\left(s_j\right)$, which is often interpreted as the energy that is required for running job j for one time unit at speed s_j; the operational cost is then

$$G = \sum_{j=1}^{n} \frac{\gamma_j}{s_j} g_j\left(s_j\right) = \sum_{j=1}^{n} p_j g_j\left(\frac{\gamma_j}{p_j}\right).$$

Notice that our model is formulated for a homogeneous distributed system: all physical machines have the same speed and energy characteristics, and the cost functions g_j are independent of machines.

It is widely accepted in the literature on power-aware scheduling that energy is proportionate to the cube of speed; see, e.g., Brooks et al. (2000) and Pruhs et al. (2008). This is why in most illustrative examples presented in the remainder of this paper we assume that

$$g_j\left(s_j\right) = \beta_j s_j^3, \quad j \in N, \tag{3}$$

so that

$$G = \sum_{j=1}^{n} \beta_j \gamma_j s_j^2 = \sum_{j=1}^{n} \frac{\beta_j \gamma_j^3}{p_j^2}. \tag{4}$$

Depending on the type of the objective function, in this paper we address several different models with immediate start:

- Π_+: it is required to find a feasible schedule that minimizes the aggregated cost $F + G$;
- Π_1: it is required to find a feasible schedule that minimizes one of the cost functions subject to an upper bound on the other function, e.g., to minimize total energy G subject to an upper bound on the value of F;
- Π_2: it is required to find feasible schedules that simultaneously minimize two cost components, e.g., to find the Pareto-optimal solutions for the problem of minimizing total cost F and total energy G.

1.2 Related work

Both features, machine speed scaling and the immediate start condition, have a long history of study. However, so far they have been considered separately and in different contexts. One point of difference is related to preemption, the ability to interrupt and resume job processing at any time. This feature is typically accepted in speed scaling research in order to avoid intractable cases, while it is forbidden in the immediate start model on a single machine and on parallel identical machines. Notice that preemptive version with immediate start should have additional condition on immediate migration and restart, which makes preemption redundant. In what follows, we provide further details about the two streams of research.

The speed scaling research stems from the seminal paper by Yao et al. (1995), who developed an $O(n^3)$-time algorithm for preemptive scheduling of n jobs on a single machine within the time windows $\left[r_j, \bar{d}_j\right]$ given for each job $j \in N$. Note that in that paper time windows are treated in the traditional sense, without the immediate start requirement. Subsequent papers by Li et al. (2006, 2014), Albers et al. (2011, 2014) and Angel et al. (2012) proposed improved algorithms for the single-machine problem and extended this line of research to the multi-machine model. The running times of the current fastest algorithms are $O(n^2)$ and $O(n^4)$ for the single-machine and parallel-machine cases, see Shioura et al. (2015).

Speed scaling problems which involve not only the speed cost function G, but also a scheduling cost function $F = \sum_{j=1}^{n} f_j(C_j)$ have been under study since the paper by Pruhs et al. (2008). The two most popular functions are the total completion time $F_1 = \sum_{j=1}^{n} C_j$ and the total rejection cost $F_2 = \sum_{j=1}^{n} w_j \mathrm{sgn}\left(\max\left\{C_j - \bar{d}_j, 0\right\}\right)$, where w_j is the cost incurred if job j cannot be processed before its deadline and therefore is rejected. Without the immediate start condition, the tractable cases of problems Π_+ and Π_1 with objectives F_1 and F_2 are very limited.

In the case of function F_1, the version of problem Π_+ with equal release dates is solvable in $O\left(n^2m^2(n+\log m)\right)$ time, where m is the number of machines; see Bampis et al. (2015). Notice that preemptions are redundant in that model. If jobs are available at arbitrary release times r_j, then problem Π_1 is NP-hard even if there is only one machine and preemption is allowed, see Barcelo (2015). For problems with arbitrary release dates and equal-work jobs, preemption allowance makes no difference to an optimal solution, and due to a nonlinear nature of the problem an optimal value of the objective can be found within a chosen accuracy ε. For example, for problem Π_1 on a single machine an algorithm by Pruhs et al. (2008) takes $O(n^2 \log \frac{\overline{G}}{\varepsilon})$ time, where $\overline{G}$ is the upper bound on the speed cost function (energy), while for problem Π_+ on parallel machines an algorithm by Albers and Fujiwara (2007) requires $O(n^3 \log \frac{1}{\varepsilon})$ time. The difficulties associated with arbitrary-length jobs are discussed by Pruhs et al. (2008), Bunde (2009), Barcelo et al. (2013). For the problem of preemptive scheduling on a single discretely controllable machine, Antoniadis et al. (2014) provide an algorithm with time complexity $O(n^4k)$, where k is the number of possible speed values of the processor.

In the speed scaling research, the problems of minimizing the total rejection cost F_2 are typically studied as those of maximizing the throughput, defined as the number of jobs that can be processed by their deadlines. Polynomial-time algorithms are known only for special cases, where various conditions are imposed, in addition to the assumption that all jobs have equal weights w_j. Notice that strict assumptions of those models make preemption redundant. The single-machine problem Π_1 with $w_j = 1$ for all $j \in N$ is solvable in $O\left(n^4 \log n \log\left(\sum \gamma_j\right)\right)$ time and in $O\left(n^6 \log n \log\left(\sum \gamma_j\right)\right)$ time, depending on whether the jobs are available simultaneously ($r_j = 0$ for all $j \in N$) or not; in the latter case, it is further required that release dates and deadlines are agreeable, see Angel et al. (2013). The parallel-machine problem Π_1 with the jobs of equal size and equal weight ($\gamma_j = w_j = 1$, $j \in N$) is solvable in $O(n^{12m+9})$ time or in $O(n^{4m+4}m)$ time, if additionally release dates and deadlines are agreeable, see Angel et al. (2016).

Research on speed scaling problems extends to the design of approximation algorithms and the study of their online versions. Without providing a comprehensive list of results of this type, we refer an interested reader to the survey papers by Albers (2009, 2010a, b) and Bampis (2016).

As far as the immediate start condition is concerned, the most relevant problems studied in the literature fall into the category of interval scheduling. In such models, each job is characterized by time intervals where it can be processed (Kovalyov et al. 2007). One of the most well-studied versions of interval scheduling assumes that there is only one interval per job $[r_j, \overline{d}_j]$. In interval scheduling, there is no freedom in selecting job starting times and in making preemption: every job $j \in N$ should start precisely at a given time r_j and complete at a given deadline $\overline{d}_j$. There is also no control over machine speeds, which are fixed and cannot be changed. The decision making consists in (i) selecting a subset of jobs that can be processed within their time intervals and (ii) assigning them to the machines for processing without preemption. The two typical objectives are the job rejection cost, which is defined similarly to the function F_2, and the machine usage cost defined typically as the (weighted) number of machines which are selected to process the jobs. Note that unlike the operational cost function G used in our model, the machine usage cost in interval scheduling does not take into account the actual time of using a machine.

Within the broad range of interval scheduling results (see the survey papers by Kolen et al. (2007) and Kovalyov et al. (2007)), those relevant to our study deal with identical parallel machines or uniform machines. In the case of identical parallel machines, the fastest algorithms for minimizing the job rejection cost have time complexity $O(n \log n)$ if all jobs have equal weights (Carlisle and Lloyd 1995) and $O(mn \log n)$ if job weights are allowed to be different (Bouzina and Emmons 1996); the fastest algorithm for minimizing the machine usage cost is of time complexity $O(n \log n)$ if machine weights are equal (Gupta et al. 1979).

The version of the problem with uniform machines is less studied. For uniform machines, both problems, with job rejection cost and machine usage cost, are strongly NP-hard; see Nakajima et al. (1982) and Bekki and Azizoğlu (2008). Polynomial-time algorithms, all of time complexity $O(n \log n)$, are known for the problem of minimizing the machine usage cost, if there are only two types of machines, slow and fast (Nakajima et al. 1982), and for the problem of minimizing the job rejection cost, in one of the following two cases: if all jobs are available simultaneously and have equal weights, or if all jobs have equal volume and there are only two processing machines (Bekki and Azizoğlu 2008).

One more problem related to our study is a relaxed version of interval scheduling, where the jobs are allowed to start at any time after their release dates r_j, but they are required to complete exactly at their deadlines $\overline{d}_j$. Such a problem can be considered as a counterpart of our problem, where the jobs are required to start at release dates r_j, but they are allowed to complete at any time before deadlines $\overline{d}_j$.

For the model with fixed job completion times, the scheduling cost function $F = \sum f_j\left(C_j\right)$ can be only of type F_2 representing the job rejection cost, since for any accepted job j, $C_j = \overline{d}_j$ and therefore there is no scope for optimizing a function $f_j(C_j)$. Prior study focuses on the model with identical parallel machines, where machine speeds are equal and cannot be changed. Algorithms of time complexities $O(n \log n)$ and $O(n^2m)$ are proposed by Lann and Mosheiov (2003) and Hiraishi et al. (2002) for the case of

equal-weight jobs and for the general case, respectively. The latter result is generalized further to the case of controllable processing times (Leyvand et al. 2010), where a job consuming x_j amount of resources gets a compressed processing time given by an arbitrary decreasing function $p'_j(x_j)$; the associated resource consumption cost is linear,

$$G' = \sum_{j=1}^{n} \beta'_j x_j.$$

The two typical examples of p'_j are

$$p'_j(x_j) = \overline{p}_j - a_j x_j \quad \text{and} \quad p'_j(x_j) = \left(\frac{\theta_j}{x_j}\right)^k,$$

where $\overline{p}_j$ and θ_j are given job-related parameters, while k is a positive constant; see Shabtay and Steiner (2007). The second model is linked to the power-aware model: if $k = 1/2$, $\theta_j = \gamma_j^3$ and $x_j = \gamma_j s_j^2$, then we get $G = G'$ assuming a cubic power function (compare $p'_j(x_j) = \frac{\gamma_j}{s_j}$ with (1) and $G' = \sum_{j=1}^{n} \beta'_j \gamma_j s_j^2$ with (4)). Notice that as observed above, the research with fixed completion times is limited to only one type of scheduling objective: job rejection cost.

As demonstrated in Leyvand et al. (2010), the counterpart of problem Π_+ with fixed completion times can be solved in $O(mn^2)$ time, while the counterparts of problems Π_1 and Π_2 are NP-hard. For discrete versions of NP-hard problems, Leyvand et al. (2010) develop algorithms of time complexity $O(mn^{m+1}X_{\max})$, where $X_{\max}$ is the maximum resource usage cost, $X_{\max} = \sum_{j=1}^{n} \beta_j^{\text{contr}} \max\{x_j\}$, assuming that resource amounts x_j are allowed to take only discrete values from a given range.

We study the most general versions of Π_+, Π_1 and Π_2 with arbitrary functions $f_j(C_j)$, reflecting diverse needs of customer-oriented quality-of-service provisioning in distributed systems. Problem Π_+ is solvable in $O(n)$ time on a single machine (Sect. 2), and in $O(n^2 m)$ on m parallel machines (Sect. 3). The Π_1 model of minimizing energy G on a single machine subject to an upper bound on the total flow time is handled in Sect. 4; we formulate it as a nonlinear resource allocation problem with continuous variables and explain how it can be solved in $O(n \log n)$ time. In Sect. 5, we present a method, also of time complexity $O(n \log n)$, for finding Pareto-optimal solutions for the Π_2 model, in which the functions F and G have to be simultaneously minimized on a single machine. Conclusions are presented in Sect. 6.

2 Problem Π_+ on a single machine

In this section, we consider the problem of minimizing the sum of the performance cost function F and total energy G on a single machine, provided that each job $j \in N$ starts immediately at time r_j.

It is clear that in the single-machine case, in order to guarantee the immediate start of job $j+1$, each job j, $1 \le j \le n-1$, must be completed no later than time r_{j+1}. Taking into account deadlines $\bar{d}_j$, we conclude that in a feasible schedule each job $j \in N$ must be completed by its imposed deadline D_j given by

$$D_j = \min\{\bar{d}_j, r_{j+1}\}, \ 1 \le j \le n.$$

Recall that $r_{n+1} = +\infty$.

Due to the immediate start condition, the actual processing time p_j of job $j \in N$ should not exceed

$$u_j = D_j - r_j.$$

In order to minimize the sum of total cost F and total energy G, we need to solve a problem, which in terms of the decisions variables p_j, $j \in N$, can be formulated as

$$\begin{aligned} &\text{Minimize } Z = \sum_{j=1}^{n} f_j(r_j + p_j) + \sum_{j=1}^{n} p_j g_j\left(\frac{\gamma_j}{p_j}\right) \\ &\text{subject to} \quad 0 < p_j \le u_j, \ j \in N. \end{aligned} \tag{5}$$

Due to a separable structure of the objective in (5), the optimal processing times can be found independently for each job $j \in N$ by solving the following n problems with a single decision variable p:

$$\begin{aligned} &\text{Minimize } Z_j = f_j(r_j + p) + p g_j\left(\frac{\gamma_j}{p}\right) \\ &\text{subject to} \quad 0 < p \le u_j. \end{aligned} \tag{6}$$

For problem (6), let Z_j^* denote the smallest value of the objective function Z_j, and p_j^* be the value of p that minimizes Z_j. In the schedule that minimizes $F + G$, the jobs are processed in the order of their numbering, and the actual processing time of job $j \in N$ is equal to p_j^*.

For most practically relevant cases, we may assume that for each $j \in N$ problem (6) can be solved in constant time. Under this assumption, we obtain the following statement.

Theorem 1 *The problem Π_+ of minimizing the sum of total cost F and total energy G on a single machine is solvable in $O(n)$ time, provided that the jobs are numbered in accordance with (2) and for each $j \in N$ problem (6) can be solved in constant time.*

Below we present several illustrations, taking two popular scheduling performance measures and, as agreed in Sect. 1, a cubic speed cost function (3). Notice that for the latter function, $p g_j\left(\frac{\gamma_j}{p}\right) = \frac{\beta_j \gamma_j^3}{p^2}$, $j \in N$.

For job $j \in N$, suppose that $f_j\left(C_j\right) = w_j C_j$, i.e., F represents the weighted sum of the completion times. Then, problem (6) can be written as

$$\begin{aligned} &\text{Minimize } Z_j = w_j p + w_j r_j + \frac{\beta_j \gamma_j^3}{p^2} \\ &\text{subject to} \qquad 0 < p \le u_j, \end{aligned}$$

so that $p_j^* = \min\left\{\gamma_j \left(\frac{2\beta_j}{w_j}\right)^{1/3}, u_j\right\}$.

For another illustration, assume that job $j \in N$ is given a "soft" due date d_j, but no "hard" deadline $\bar{d}_j$, i.e., $D_j = r_{j+1}$, $1 \le j \le n-1$. Suppose that $f_j\left(C_j\right) = w_j \max\left\{C_j - d_j, 0\right\}$, i.e., F represents total weighted tardiness.

If for a job $j \in N$, the inequality $r_{j+1} \le d_j$ holds, then job j will be completed before its due date and problem (6) can be written as

$$\begin{aligned} &\text{Minimize} \qquad \frac{\beta_j \gamma_j^3}{p^2} \\ &\text{subject to } 0 < p \le r_{j+1} - r_j, \end{aligned}$$

so that $p_j^* = \min\left\{\gamma_j \left(\frac{2\beta_j}{w_j}\right)^{1/3}, r_{j+1} - r_j\right\}$. Otherwise, i.e., if $r_{j+1} > d_j$, in order to solve problem (6), we need to solve two problems:

$$\begin{aligned} &\text{Minimize} \qquad \frac{\beta_j \gamma_j^3}{p^2} \\ &\text{subject to } 0 < p \le d_j - r_j, \end{aligned}$$

which corresponds to an early completion of job j so that no tardiness occurs, and

$$\begin{aligned} &\text{Minimize } w_j(p + r_j - d_j) + \frac{\beta_j \gamma_j^3}{p^2} \\ &\text{subject to } d_j - r_j \le p \le r_{j+1} - r_j, \end{aligned}$$

where job j completes after its due date. For job $j \in N$, the optimal actual processing time p_j^* is equal to the value of p that delivers the lowest value of the objective function in these two problems.

In the presented examples, which can be extended to most traditionally used objective functions, the actual processing time p_j^* of each job is essentially written in closed form, which justifies our assumption that each problem (6) can be solved in constant time.

3 Problem Π_+ on parallel machines

In this section, we study the problem of finding an immediate start schedule for processing the jobs of set N on m parallel machines $M_1, M_2, \ldots, M_m$ that minimizes the sum of the total cost F and total energy G. Adapting the ideas from the work by Hiraishi et al. (2002) and Leyvand et al. (2010), we reduce our problem to a minimum-cost flow problem in a special network (Fig. 1).

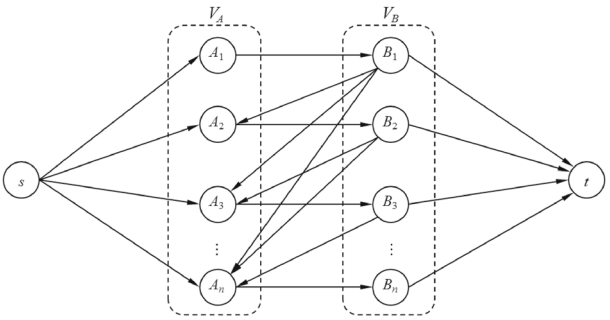

Fig. 1 Network $H = (V, T)$

Introduce a bipartite network $H = (V, T)$. The node set $V = \{s, t\} \cup N_A \cup N_B$ consists of a source s, a sink t, two sets $N_A = \{A_1, A_2, \ldots, A_n\}$ and $N_B = \{B_1, B_2, \ldots, B_n\}$, where each node A_j and B_j is associated with job $j \in N$. The set T of arcs is defined as $T = T_A \cup T_{AB} \cup T_{BA} \cup T_B$, where

$$\begin{aligned} &T_A = \{(s, A_j) \mid A_j \in N_A\}, \\ &T_B = \{\left(B_j, t\right) \mid B_j \in N_B\}, \\ &T_{AB} = \{(A_j, B_j) \mid j \in N\}, \\ &T_{BA} = \{(B_j, A_k) \mid B_j \in N_B,\ A_k \in N_A,\ 1 \le j < k \le n\}. \end{aligned}$$

Each arc $\left(q, q'\right) \in T$ is associated with capacity $\mu(q, q')$ and cost $c(q, q')$. Recall that a feasible flow $x : A \to \mathbb{R}$ satisfies the *capacity constraint*

$$0 \le x(q, q') \le \mu(q, q'), \qquad (q, q') \in T, \tag{7}$$

i.e., the flow on an arc cannot be larger than its capacity, and the *flow balance constraint*

$$\sum_{q:\ (q', q) \in T_{AB} \cup T_{BA}} x(q', q) = \sum_{q:\ (q, q') \in T_{AB} \cup T_{BA}} x(q, q'), \tag{8}$$

for $q \in V \setminus \{s, t\}$, i.e., for each node q other than the source and the sink the flow that enters the node must be equal to the flow that leaves the node.

The value of a flow x is equal to the total flow on the arcs that leave the source (or, equivalently, enter the sink):

$$\text{the value of flow } x = \sum_{j \in N} x(s, A_j) = \sum_{j \in N} x(B_j, t).$$

For network H, let us set all arc capacities to 1. By appropriately defining the costs on the arcs of network H,

we reduce the original problem of minimizing the objective function $F + G$ on m parallel machines to finding the minimum-cost flow of value m in H.

Suppose a flow of value m in network H is found. Since the network is acyclic, the arcs with a flow equal to 1 will form m paths from s to t , and the order of arcs of set T_{BA} in each path defines the sequence of jobs on a machine. A path starts with an arc (s, A_j), proceeds with pairs of arcs of the form (A_j, B_j), (B_j, A_k), and concludes with the final pair (A_ℓ, B_ℓ), (B_ℓ, t). An arc (s, A_j) implies that job j is the first on some machine. A pair (A_j, B_j), (B_j, A_k) corresponds to scheduling two jobs, j and k, one after another on the same machine, while a pair (A_ℓ, B_ℓ), (B_ℓ, t) corresponds to assigning job ℓ as the last job on a machine.

The arc costs reflect the selected sequence of jobs on a machine. If a job $j \in N$ has no "hard" deadline, define $\bar{d}_j = +\infty$. For the final pair of the chain (A_ℓ, B_ℓ), (B_ℓ, t), the cost of scheduling job ℓ as the last job on a machine is equal to the contribution of job $\ell \in N$ to the objective function. It can be found as the optimal value Z^*_ℓ for the problem (6) with $j = \ell$ and $u_j = \bar{d}_j$. Thus, for each $j \in N$, we compute the value Z^*_j and assign this value as a cost of the arc (B_j, t).

If a pair of arcs (A_j, B_j), (B_j, A_k) with $r_j < r_k$ belongs to a certain path from s to t, then job j is sequenced on some machine immediately before job k, and therefore must complete before time $\min\{\bar{d}_j, r_k\}$. The cost associated with the job sequence (j, k) is equal to the smallest value $Z^*_{j,k}$ of the objective function $Z_{j,k}$ for the problem

$$\begin{aligned} &\text{Minimize } Z_{j,k} = f_j(r_j + p) + pg_j\left(\frac{\gamma_j}{p}\right) \\ &\text{subject to } 0 < p \le \min\{\bar{d}_j - r_j, r_k - r_j\}, \end{aligned} \tag{9}$$

which can be seen as problem (6), where $u_j = \min\{\bar{d}_j - r_j, r_k - r_j\}$. For each pair (j, k) where $1 \le j < k \le n$, we compute the value $Z^*_{j,k}$ and assign this value as a cost of the arc (B_j, A_k).

For each arc $(A_j, B_j) \in T_{AB}$, the cost is set equal to $-M$, where M is a large positive number. This guarantees that every arc $(A_j, B_j) \in T_{AB}$ receives a flow of 1, so that each job $j \in N$ will be scheduled. If we ignore the costs of the arcs $(A_j, B_j) \in T_{AB}$, the total cost of the found flow is equal to the optimal value of the function $F + G$.

Thus, if one of the paths from s to t visits the sequence of nodes $(s, A_{j_1}, B_{j_1}, A_{j_2}, B_{j_2}, \ldots, A_{j_y}, B_{j_y}, t)$, then in the associated schedule on some machine the sequence of jobs $(j_1, j_2, \ldots, j_y)$ is processed. The actual processing time $p^*_{j_i}$ of job j_i, $1 \le i \le y - 1$, is equal to the value of p that delivers the smallest value of $Z^*_{j_i, j_{i+1}}$, while for the last job j_y the actual processing $p^*_{j_y}$ is defined by the value of p that delivers the smallest value of $Z^*_{j_y}$.

As in Sect. 2, we may assume that determining the cost of each arc of network H takes constant time, so that all the costs will be found in $O(n^2)$ time. The required flow can be found in $O(n^2 m)$ by applying the successive shortest path algorithm, similar to the Ford–Fulkerson algorithm; see Ahuja et al. (1993).

Theorem 2 *The problem Π_+ of minimizing the sum of total cost F and total energy G on m parallel machines is solvable in $O(n^2 m)$ time by finding the minimum-cost flow of value m in network H, provided that the cost of each arc of H can be computed in constant time.*

The described approach can be extended to the problem of determining the optimal number of parallel machines to be used. This aspect is particularly important in modern computing systems, as there are overheads related to initialization of virtual machines in Clouds, and overheads for activating the machines which are in the sleep mode.

Suppose that using v parallel machines incurs cost σ_v, $1 \le v \le m$, and we are interested in minimizing $F + G$ plus additionally the cost σ_v of all used machines. This can be done by solving the sequence of flow problems in network H, trying flow values 1, then 2, etc. up to an upper bound m on the machine number. For each tried value of v, $1 \le v \le m$, the function $F + G + \sigma_v$ is evaluated and the best option is taken. The running time for solving the resulting problem remains $O(n^2 m)$, since the successive shortest path algorithm for finding the min-cost flow of value m will iteratively find the min-cost flows with all required intermediate values $1, 2, \ldots, m - 1$.

Theorem 3 *The problem Π_+ of minimizing the sum of total cost F, total energy G and the cost σ_v for using $v \le m$ machines, where v is a decision variable, is solvable in $O(n^2 m)$ time, under the assumptions of Theorem 2.*

A drawback of the model with the aggregated objective function is that it schedules all arrived jobs. In the case of a rather short interval available for processing a job, this can only be achieved if a very high speed is applied, which may be unacceptably expensive. It may appear to be beneficial not to accept certain jobs and to pay an agreed rejection fee.

Suppose that the cost of rejecting job $j \in N$ is δ_j. Let N_A be the set of accepted jobs, while $N_R = N \backslash N_A$ be the set of rejected jobs. If we want to minimize the sum of the performance function for the accepted jobs, total energy used and total rejection penalty, we need to solve the problem with the objective function

$$Z = \sum_{j \in N_A} f_j(p_j + r_j) + \sum_{j \in N_A} p_j g_j\left(\frac{\gamma_j}{p_j}\right) + \sum_{j \in N_R} \delta_j,$$

which is equivalent (up to the additive constant $\sum_{j\in N}\delta_j$) to

$$Z' = \sum_{j\in N_A} f_j\left(p_j + r_j\right) + \sum_{j\in N_A} p_j g_j\left(\frac{\gamma_j}{p_j}\right) - \sum_{j\in N_A}\delta_j.$$

In the network model, we replace the cost on an arc $\left(A_j, B_j\right) \in T_{AB}$ by $-\delta_j$, keeping the cost of an arc $\left(B_j, A_k\right) \in T_{BA}$ the same as in the basic model. Recall that the latter cost is found by solving problem (9). Since in an optimal solution less than m machines can be used, we add an extra arc (s, t) of capacity m and zero cost.

For example, suppose that the minimum-cost flow of value ℓ, $\ell \le m$, in the modified network is found, and one of the paths from s to t visits the sequence of nodes $(s, A_{j_1}, B_{j_1}, A_{j_2}, B_{j_2}, \ldots, A_{j_y}, B_{j_y}, t)$. Then, the sequence of accepted jobs $\left(j_1, j_2, \ldots, j_y\right)$ is processed on some machine, and the contribution of job j_i is equal to the cost of the arc that leaves node B_{j_i}, found by solving problem (9), plus the cost $-\delta_{j_i}$ of the arc that enters node B_{j_i}, $1 \le i \le y$. The described adjustments do not change the time complexity of the approach.

Theorem 4 *The problem Π_+ in which it is required to determine the set N_R of rejected jobs to minimize the sum of total cost F, total energy G and the cost $\sum_{j\in N_R}\delta_j$ is solvable in $O\left(n^2 m\right)$ time, under the assumptions of Theorem 2.*

4 Problem Π_1 on a single machine

In this section, we consider the problem of minimizing total energy G subject to a constraint on total cost F on a single machine. The presented solution approach is based on Karush–Kuhn–Tucker (KKT) reasoning in relation to the associated Lagrange function. This approach works for a wide range of functions G and F; however, below for simplicity it is presented for the case that $F = \sum_{j\in N}\left(C_j - r_j\right)$, i.e., F represents total flow time. Moreover, a natural interpretation of the obtained results occurs if for each $j \in N$ the energy function g_j is polynomial, strictly convex, decreasing in p_j and job-independent, e.g., satisfies (3) with $\beta_j = 1$.

Due to the immediate start condition, we see that $C_j - r_j = p_j$, and let P be a given upper bound on $F = \sum_{j\in N} p_j$. Let u_j be an upper bound on the actual processing time p_j, defined as in Sect. 2. Denote

$$G_j\left(p_j\right) = p_j g_j\left(\frac{\gamma_j}{p_j}\right),\ j \in N.$$

Then, the problem we study in this section can be formulated as

$$\begin{aligned} &\text{Minimize} \quad G = \sum_{j=1}^{n} G_j\left(p_j\right) \\ &\text{subject to} \quad \sum_{j=1}^{n} p_j \le P, \\ &\qquad 0 < p_j \le u_j,\ j \in N. \end{aligned} \tag{10}$$

Such a problem can be classified as a nonlinear resource allocation problem with continuous decision variables; see the survey by Patriksson (2008). Note that we can limit our consideration to the case of $P < \sum_{j\in N} u_j$; otherwise, in an optimal solution $p_j = u_j$ for all $j \in N$.

For a vector $\mathbf{p} = (p_1, p_2, \ldots, p_n)$ and a non-negative Lagrange multiplier λ, introduce the Lagrangian function

$$L\left(\mathbf{p}, \lambda\right) = \sum_{j=1}^{n} G_j\left(p_j\right) + \lambda\left(\sum_{j=1}^{n} p_j - P\right)$$

and its dual

$$Q\left(\lambda\right) = -\lambda P + \sum_{j=1}^{n} \min_{0<p_j\le u_j}\left\{G_j\left(p_j\right) + \lambda p_j\right\},$$

see Patriksson (2008). The derivative of the Lagrangian dual $Q\left(\lambda\right)$ is equal to

$$Q'\left(\lambda\right) = -P + \sum_{j=1}^{n} p_j\left(\lambda\right),$$

where vector $\mathbf{p}\left(\lambda\right) = \left(p_1\left(\lambda\right), p_2\left(\lambda\right), \ldots, p_n\left(\lambda\right)\right)$ delivers the unique minimum to the Lagrangian function for a given λ.

The KKT conditions guarantee that there exists a value λ^* such that $Q'\left(\lambda^*\right) = 0$. Such a multiplier λ^* and vector $\mathbf{p}\left(\lambda^*\right)$ deliver the minimum to the Lagrangian function, so that vector $\mathbf{p}\left(\lambda^*\right)$ is a solution to problem (10), i.e., defines the optimal values of the actual processing times.

Differentiating functions $G_j\left(p_j\right)$ denote

$$\lambda_j = -G_j'\left(u_j\right),\ 1 \le j \le n.$$

For a polynomial energy function, e.g., $G_j\left(p_j\right) = p_j\frac{\gamma_j^3}{p_j^3} = \frac{\gamma_j^3}{p_j^2}$, $j \in N$, the values λ_j admit a natural interpretation. Indeed, $\lambda_j = -G_j'\left(u_j\right) = \frac{2\gamma_j^3}{u_j^3} = 2s_j^3$, i.e., if for a job j the actual processing time is equal u_j, then this job is processed at speed $\sqrt[3]{\lambda_j/2}$.

Let $\pi = \left(\pi\left(1\right), \pi\left(2\right), \ldots, \pi\left(n\right)\right)$ be a permutation of the numbers $1, 2, \ldots, n$ such that

$$\lambda_{\pi(1)} \ge \lambda_{\pi(2)} \ge \cdots \ge \lambda_{\pi(n)}.$$

For a k, $1 \le k \le n$, in accordance with the KKT reasoning for the resource allocation problem (Patriksson 2008), define

$$p_{\pi(j)}\left(\lambda_{\pi(k)}\right) = \begin{cases} u_{\pi(j)}, \text{ if } 1 \le j \le k, \\ p^0_{\pi(j)}, \text{ if } k+1 \le j \le n, \end{cases} \tag{11}$$

where the values $p^0_{\pi(j)}$ for $k+1 \le j \le n$ are solutions of the system of equations

$$G'_{\pi(j)}\left(p_{\pi(j)}\right) = -\lambda_{\pi(k)}, \ k+1 \le j \le n. \tag{12}$$

By applying binary search with respect to k, we find a value k^* such that either

$$\sum_{j=1}^{n} p_j\left(\lambda_{\pi(k^*)}\right) = P$$

or

$$\sum_{j=1}^{n} p_j\left(\lambda_{\pi(k^*)}\right) > P > \sum_{j=1}^{n} p_j\left(\lambda_{\pi(k^*+1)}\right).$$

In the former case, we define $\lambda^* = \lambda_{\pi(k^*)}$ and $\mathbf{p}(\lambda^*) = \mathbf{p}\left(\lambda_{\pi(k^*)}\right)$; otherwise, solve the system of equations

$$\begin{aligned} &G'_{\pi(j)}\left(p^*_{\pi(j)}\right) = -\lambda^*, \quad k^*+1 \le j \le n, \\ &\textstyle\sum_{j=1}^{k^*} u_{\pi(j)} + \sum_{j=k^*+1}^{n} p^*_{\pi(j)} = P. \end{aligned} \tag{13}$$

Having solved the latter system, we determine λ^* and the values $p^*_{\pi(j)}$, $k^*+1 \le j \le n$. The components of the solution vector $\mathbf{p}(\lambda^*)$ are defined by

$$p_{\pi(j)}\left(\lambda^*\right) = \begin{cases} u_{\pi(j)}, \text{ if } 1 \le j \le k^*, \\ p^*_{\pi(j)} \text{ if } k^*+1 \le j \le n. \end{cases}$$

The search for the value k^* takes at most $\log n$ iterations, and system (12) has to be solved in each iteration. Additionally, system (13) has to be solved at most once. If energy functions are cubic, we may assume that solving systems (12) and (13) requires time that is linear with respect to the number of decision variables. Indeed, the solution to (12) is given by

$$p^0_{\pi(j)} = \sqrt[3]{\frac{2}{\lambda_{\pi(j)}}}\gamma_{\pi(j)} = \frac{u_{\pi(k)}}{\gamma_{\pi(k)}} \times \gamma_{\pi(j)}, \quad k+1 \le j \le n. \tag{14}$$

The solution to (13) is given by

$$\begin{aligned} &\lambda^* = \frac{2\left(\sum_{j=k^*+1}^{n}\gamma_{\pi(j)}\right)^3}{\left(P-\sum_{j=1}^{k^*}u_{\pi(j)}\right)^3}; \\ &p_{\pi(j)}\left(\lambda^*\right) = \sqrt[3]{\frac{2}{\lambda^*}}\gamma_{\pi(j)}, \quad k^*+1 \le j \le n. \end{aligned}$$

Thus, we have proved the following statement.

Theorem 5 *The problem Π_1 of minimizing total energy G on a single machine, subject to the bounded total flow time $F \le P$, reduces to the nonlinear resource allocation problem and can be solved in $O(n \log n)$ time, provided that energy functions g_j are polynomial, strictly convex, decreasing in p_j and job-independent.*

The following remark is useful for justifying the solution method for the bicriteria problem, presented in the next section. Simultaneous equations (13) imply that in an optimal solution for each job $\pi(j)$, $1 \le j \le k^*$, the equality $p_{\pi(j)}(\lambda^*) = u_{\pi(j)}$ holds, i.e., each of these jobs fully uses the interval $\left[r_{\pi(j)}, r_{\pi(j)} + u_{\pi(j)}\right]$ available for its processing. The processing speed of job $\pi(j)$, $1 \le j \le k^*$, is $\frac{\gamma_{\pi(j)}}{u_{\pi(j)}} = \sqrt[3]{\lambda_{\pi(j)}/2}$. Besides, for $k^*+1 \le j \le n$, due to (13), it follows that $G'_{\pi(j)}\left(p^*_{\pi(j)}\right) = -\lambda^*$, so that all jobs $\pi(j)$, $k^*+1 \le j \le n$, are processed at the same speed $\sqrt[3]{\lambda^*/2}$ and none of these jobs fully uses the available interval. Moreover, since $\lambda_{\pi(1)} \ge \cdots \ge \lambda_{\pi(k^*)} > \lambda^* > \lambda_{\pi(k^*+1)} \ge \cdots \ge \lambda_{\pi(n)}$, we conclude that the common speed at which each job $\pi(j)$, $k^*+1 \le j \le n$, is processed is less than the processing speed of the jobs $\pi(j)$, $1 \le j \le k^*$.

5 Problem Π_2 on a single machine

In this section, we describe an approach to solving the bicriteria problem, in which it is required to simultaneously minimize total cost F and total energy G on a single machine. Recall that a schedule S' is called *Pareto-optimal* if there exists no schedule S'' such that $F(S'') \le F(S')$ and $G(S'') \le G(S')$, where at least one of these inequalities is strict.

Although the outlined approach can be extended to deal with rather general cost functions, below we present it for $F = \sum_{j=1}^{n}\left(C_j - r_j\right)$ and $G = \sum_{j=1}^{n} p_j g_j\left(\frac{\gamma_j}{p_j}\right) = \sum_{j=1}^{n}\frac{\gamma_j^3}{p_j^2}$. The solution of the problem of finding the Pareto optimum is given in the space of variables F and G by (i) a sequence of break-points $F_0, F_1, F_2, \ldots, F_\nu$ of the variable F and (ii) an explicit formula that expresses variable G as a function of variable $F \in \left[F_k, F_{k+1}\right]$ for all $k = 0, 1, \ldots, \nu-1$. As we show below, $\nu = n$.

In line with the reasoning presented in Sect. 4, compute

$$s_j = \frac{\gamma_j}{u_j}, \ j \in N.$$

The value s_j represents the speed at which job $j \in N$ has to be processed to get the actual processing time u_j. Determine a permutation $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$ of the numbers

$1, 2, \ldots, n$ such that

$$s_{\pi(1)} \geq s_{\pi(2)} \geq \cdots \geq s_{\pi(n)}.$$

For completeness, define $s_{\pi(0)} = +\infty$.

Define

$$U_k(\pi) = \sum_{j=1}^{k} u_{\pi(j)}, \ \Gamma_k(\pi) = \sum_{j=1}^{k} \gamma_{\pi(j)},$$
$$R_k(\pi) = \sum_{j=1}^{k} \frac{\gamma_{\pi(j)}^3}{u_{\pi(j)}^2}, \ 1 \leq k \leq n.$$

Additionally, for completeness, define $U_0(\pi) = \Gamma_0(\pi) = R_0(\pi) = 0$.

Denote $\Gamma = \sum_{j \in N} \gamma_j$. Introduce $F_0 = 0$ and

$$\begin{aligned} F_k &= \sum_{j=1}^{k} u_{\pi(j)} + \frac{u_{\pi(k)}}{\gamma_{\pi(k)}} \sum_{j=k+1}^{n} \gamma_{\pi(j)} \\ &= U_k(\pi) + \frac{\Gamma - \Gamma_k(\pi)}{s_{\pi(k)}}, \quad 1 \leq k \leq n. \end{aligned} \tag{15}$$

Theorem 6 *For the bicriteria problem Π_2 of minimizing total flow time F and total energy G on a single machine, the values F_k, $0 \leq k \leq n$, defined by (15) correspond to the break-points of the variable F, and the variable G can be expressed as*

$$G = R_k(\pi) + \frac{(\Gamma - \Gamma_k(\pi))^3}{(F - U_k(\pi))^2}, \ F \in (F_k, F_{k+1}], \tag{16}$$

for $0 \leq k \leq n-1$. Problem Π_2 is solvable in $O(n \log n)$ time.

Proof The fact that the values F_k, $0 \leq k \leq \nu$, are indeed break-points and that $\nu = n$ follows from the structure of an optimal solution of the problem of minimizing total energy G subject to an upper bound on the sum of actual processing times; see (11) and (14) from Sect. 4. For $F \in (F_k, F_{k+1}]$ considering the jobs in accordance with the permutation π, the actual processing times of the first k jobs are fixed to their upper bounds, while the actual processing times of the remaining jobs are obtained by running these jobs at a common speed s, that decreases starting from $s_{\pi(k)}$. The next break-point F_{k+1} occurs when s becomes equal to $s_{\pi(k+1)}$. Note that break-points F_k and F_{k+1} coincide if $s_{\pi(k)} = s_{\pi(k+1)}$, but we count them separately so that indeed $\nu = n$. The last break-point F_n corresponds to the situation that the actual processing time of job $\pi(n)$ is equal to its largest possible value $u_{\pi(n)}$.

Consider the interval $(F_0, F_1]$. In this interval, the jobs are run with a speed $s \geq s_{\pi(1)}$, so that for $F \in (F_0, F_1]$, $F = \sum_{j \in N} p_j = \sum_{j \in N} \gamma_j / s = \Gamma / s$. We deduce that

$$\begin{aligned} G &= \sum_{j=1}^{n} \frac{\gamma_j^3}{p_j^2} = s^2 \sum_{j=1}^{n} \gamma_j = \frac{\Gamma^3}{F^2} \\ &= R_0(\pi) + \frac{(\Gamma - \Gamma_0(\pi))^3}{(F - U_0(\pi))^2}, \quad F \in (F_0, F_1], \end{aligned}$$

which complies with (16) for $k = 0$.

Now consider the next interval $(F_1, F_2]$. It follows that $F \in (F_1, F_2]$ can be written as

$$F = u_{\pi(1)} + \frac{1}{s} \sum_{j=2}^{n} \gamma_{\pi(j)} = U_1(\pi) + \frac{\Gamma - \Gamma_1(\pi)}{s},$$

as a function of speed s, where s decreases from $s_{\pi(1)}$ to $s_{\pi(2)}$, so that

$$s = \frac{\Gamma - \Gamma_1(\pi)}{F - U_1(\pi)}$$

and

$$G = \frac{\gamma_{\pi(1)}^3}{u_{\pi(1)}^2} + s^2 \sum_{j=2}^{n} \gamma_{\pi(j)} = R_1(\pi) + \frac{(\Gamma - \Gamma_1(\pi))^3}{(F - U_1(\pi))^2},$$

as (16) for $k = 1$.

Consider an interval $(F_k, F_{k+1}]$ for some $k, 0 \leq k \leq n-1$. It follows that $F \in (F_k, F_{k+1}]$ can be written as

$$F = \sum_{j=1}^{k} u_{\pi(j)} + \frac{1}{s} \sum_{j=k+1}^{n} \gamma_{\pi(j)} = U_k(\pi) + \frac{\Gamma - \Gamma_k(\pi)}{s},$$

where s decreases from $s_{\pi(k)}$ to $s_{\pi(k+1)}$, so that

$$s = \frac{\Gamma - \Gamma_k(\pi)}{F - U_k(\pi)} \text{ and}$$
$$G = \sum_{j=1}^{k} \frac{\gamma_{\pi(j)}^3}{u_{\pi(j)}^2} + s^2 \sum_{j=k+1}^{n} \gamma_j = R_k(\pi) + \frac{(\Gamma - \Gamma_k(\pi))^3}{(F - U_k(\pi))^2}.$$

Computing G for all values of k, $0 \leq k \leq n-1$, takes $O(n \log n)$ time. This proves the theorem. □

6 Conclusions

In this paper, we address several versions of the scheduling model that combines a well-established feature of speed scaling and a requirement of immediate job starting times, that is typical for modern Cloud computing systems. Both objectives are of the min-sum type, one depending on the job completion times, and another one on the machine usage cost. We show that the single-machine model with n jobs can be solved in $O(n \log n)$ time for two single criterion versions of our problem, Π_+ and Π_1, or for the most general bicriteria version Π_2. The single criterion version Π_+ of the multi-machine model with n jobs and m machines is solvable in $O(n^2 m)$ time.

Presented results for immediate start models can be naturally generalized to handle problems that combine a max-type

scheduling objective $F^{\max} = \max_{j \in N} \{f_j(C_j)\}$ and the energy component G. For example, for $f_j(C_j) = C_j$ or $f_j(C_j) = C_j - d_j$ the objective $F_{\max}$ becomes either the makespan $C_{\max}$ or the maximum lateness $L_{\max}$, respectively.

- For problem $\Pi_1^{\max}$ (minimizing energy G subject to an upper bound $\overline{F}$ on the value of $F^{\max}$), define deadlines induced by a given value of $\overline{F}$, eliminate $F^{\max}$ from consideration by setting $f_j(C_j) = 0$, $j \in N$, and solve problem Π_+ to minimize $G + 0$ using the techniques from Sects. 2, 3.
- As far as problem $\Pi_+^{\max}$ is concerned, function $F_{\max}$ is convex in p_j for the most popular min-max scheduling objectives, such as $F_{\max} \in \{C_{\max}, L_{\max}\}$. Since the energy component G is also convex in p_j, it follows that the objective $F^{\max} + G$ is convex and its minimum can be found by a numerical method of convex optimization.

To summarize, our study can be considered as the first attempt to explore fundamental properties of the new model with the immediate start condition. Future research may elaborate further the applied aspects of our study: the basic system model can be enhanced to address a range of issues typical for modern Cloud computing systems, such as heterogeneous physical machines having different speed characteristics and energy usage functions, introduction of virtual machines with possible allocation of several virtual machines to one physical machine, the possibility of migrating virtual machines and associated tasks. Our study may also serve as a basis for the development of the online algorithms for problems with the immediate start condition. Notice that the online versions of the traditional models of power-aware scheduling, without immediate start, are proposed by Albers and Fujiwara (2007), Bansal et al. (2010), Chan et al. (2013), Lam et al. (2008, 2012).

Acknowledgements This research was supported by the EPSRC funded project EP/J019755/1 "Submodular Optimisation Techniques for Scheduling with Controllable Parameters". The first author was partially supported by JSPS KAKENHI Grant Numbers 15K00030 and 15H00848.

References

Aceto, G., Botta, A., de Donato, W., & Pescapè, A. (2013). Cloud monitoring: A survey. *Computer Networks*, *57*, 2093–2115.

Ahuja, R. K., Magnanti, T. L., & Orlin, J. B. (1993). *Network flows: Theory, algorithms and applications*. Englewood Cliffs: Prentice Hall.

Albers, S. (2009). Algorithms for energy saving. *Lecture Notes in Computer Science*, *5760*, 173–186.

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

Albers, S. (2010b). Algorithms for energy management. *Lecture Notes in Computer Science*, *6072*, 1–11.

Albers, S., & Fujiwara, H. (2007). Energy-efficient algorithms for flow time minimization. *ACM Transactions on Algorithms*, *3*, 49:1–49:17.

Albers, S., Antoniadis, A., & Geiner, G. (2011). On multiprocessor speed scaling with migration. In *Proceedings of the symposium on parallelism in algorithms and architectures (SPAA)* (pp. 279–288).

Albers, S., Müller, F., & Schmelzer, S. (2014). Speed scaling on parallel processors. *Algorithmica*, *68*, 404–425.

Angel, E., Bampis, E., Kacem, F., & Letsios, D. (2012). Speed scaling on parallel processors with migration. *Lecture Notes in Computer Science*, *7484*, 128–140.

Angel, E., Bampis, E., Chau, V., & Letsios, D. (2013). Throughput maximization for speed-scaling with agreeable deadlines. *Lecture Notes in Computer Science*, *7876*, 10–19.

Angel, E., Bampis, E., Chau, V., & Thang, N. K. (2016). Throughput maximization in multiprocessor speed-scaling. *Theoretical Computer Science*, *630*, 1–12.

Antoniadis, A., Barcelo, N., Consuegra, M., Kling, P., Nugen, M., Pruhs, K., & Scquizzatok, M. (2014). Efficient computation of optimal energy and fractional weighted flow trade-off schedules. In *Proceedings of the 31st international symposium on theoretical aspects of computer science (STACS '14)* (pp. 63–74).

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R. H., Konwinski, A., et al. (2009). *Above the clouds: A Berkeley view of cloud computing* (p. 28). UCB/EECS, vol: Technical Report, EECS Department, University of California, Berkeley.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R. H., Konwinski, A., et al. (2010). A view of cloud computing. *Communications of the ACM*, *53*, 55–58.

Bampis, E. (2016). Algorithmic issues in energy-efficient computation. *Lecture Notes in Computer Science*, *9869*, 3–14.

Bampis, E., Letsios, D., & Lucarelli, G. (2015). Green scheduling, flows and matchings. *Theoretical Computer Science*, *579*, 126–136.

Bansal, N., Pruhs, K., & Stein, C. (2010). Speed scaling for weighted flow time. *SIAM Journal on Computing*, *39*, 1294–1308.

Barcelo, N. (2015). The complexity of speed-scaling. Ph.D. thesis, University of Pittsburgh.

Barcelo, N., Cole, D., Letsios, D., Nugent, M., & Pruhs, K. (2013). Optimal energy trade-off schedules. *Sustainable Computing: Informatics and Systems*, *3*, 207–217.

Bekki, Ö. B., & Azizoğlu, M. (2008). Operational fixed interval scheduling problem on uniform parallel machines. *International Journal of Production Economics*, *112*, 756–768.

Bouzina, K. I., & Emmons, H. (1996). Interval scheduling on identical machines. *Journal of Global Optimization*, *9*, 379–393.

Brooks, D. M., Bose, P., Schuster, S. E., Jacobson, H., Kudva, P. N., Buyuktosunoglu, A., et al. (2000). Power-aware microarchitecture: Design and modeling challenges for next-generation microprocessors. *IEEE Micro*, *20*, 26–44.

Bunde, D. P. (2009). Power-aware scheduling for makespan and flow. *Journal of Scheduling*, *12*, 489–500.

Carlisle, M. C., & Lloyd, E. L. (1995). On the k-coloring of intervals. *Discrete Applied Mathematics*, *59*, 225–235.

Chan, S.-H., Lam, T.-W., & Lee, L.-K. (2013). Scheduling for weighted flow time and energy with rejection penalty. *Theoretical Computer Science*, *470*, 93–104.

Do Lago, D.G., Madeira, E.R.M., & Bittencourt, L.F. (2011). Power-aware virtual machine scheduling on clouds using active cooling control and DVFS. In *Proceedings of the 9th international workshop on middleware for grids, clouds and e-science (MGC '11)* (pp. 1–6).

Garg, S. K., Versteeg, S., & Buyya, R. (2013). A framework for ranking of cloud computing services. *Future Generation Computer Systems*, *29*, 1012–1023.

Gerards, M. E. T., Hurink, J. L., & Hölzenspies, P. K. F. (2016). A survey of offline algorithms for energy minimization under deadline constraints. *Journal of Scheduling*, *19*, 3–19.

Gupta, U. I., Lee, D. T., & Leung, J. Y.-T. (1979). An optimal solution for the channel-assignment problem. *IEEE Transactions on Computers*, *28*, 807–810.

Hiraishi, K., Levner, E., & Vlach, M. (2002). Scheduling of parallel identical machines to maximize the weighted number of just-in-time jobs. *Computers and Operations Research*, *29*, 841–848.

Iqbal, W., Dailey, M., & Carrera, D. (2009). SLA-driven adaptive resource management for web applications on a heterogeneous compute cloud. *Lecture Notes in Computer Science*, *5931*, 243–253.

Jennings, B., & Stadler, R. (2015). Resource management in clouds: Survey and research challenges. *Journal of Network and Systems Management*, *23*, 567–619.

Jing, S.-Y., Ali, S., She, K., & Zhong, Y. (2013). State-of-the-art research study for green cloud computing. *Journal of Supercomputing*, *65*, 445–468.

Kolen, A. W. J., Lenstra, J. K., Papadimitriou, C. H., & Spieksma, F. C. R. (2007). Interval scheduling: A survey. *Naval Research Logistics*, *54*, 530–543.

Kovalyov, M. Y., Ng, C. T., & Cheng, T. C. E. (2007). Fixed interval scheduling: Models, applications, computational complexity and algorithms. *European Journal of Operational Research*, *178*, 331–342.

Kushida, K. E., Murray, J., & Zysman, J. (2015). Cloud computing: From scarcity to abundance. *Journal of Industry, Competition and Trade*, *15*, 5–19.

Lam, T.-W., Lee, L.-K., To, I. K. K., & Wong, P. W. H. (2008). Speed scaling functions for flow time scheduling based on active job count. *Lecture Notes in Computer Science*, *5193*, 647–659.

Lam, T. W., Lee, L. K., To, I. K. K., & Wong, P. W. H. (2012). Improved multi-processor scheduling for flow time and energy. *Journal of Scheduling*, *15*, 105–116.

Lann, A., & Mosheiov, G. (2003). A note on the maximum number of on-time jobs on parallel identical machines. *Computers and Operations Research*, *30*, 1745–1749.

Leyvand, Y., Shabtay, D., Steiner, G., & Yedidsion, L. (2010). Just-in-time scheduling with controllable processing times on parallel machines. *Journal of Combinatorial Optimization*, *19*, 347–368.

Li, M., Yao, A. C., & Yao, F. F. (2006). Discrete and continuous min-energy schedules for variable voltage processors. *Proceedings of the National Academy of Sciences of the United States of America*, *103*, 3983–3987.

Li, M., Yao, F. F., & Yuan, H. (2014). An $O(n^2)$ algorithm for computing optimal continuous voltage schedules. arxiv:1408.5995v1.

Moreno, I. S., Garraghan, P., Townend, P., & Xu, J. (2014). Analysis, modeling and simulation of workload patterns in a large-scale utility cloud. *IEEE Transactions on Cloud Computing*, *2*, 208–221.

Nakajima, K., Hakimi, S. L., & Lenstra, J. K. (1982). Complexity results for scheduling tasks in fixed intervals on two types of machines. *SIAM Journal on Computing*, *11*, 512–520.

Patriksson, M. (2008). A survey on the continuous nonlinear resource allocation. *European Journal of Operational Research*, *185*, 1–46.

Pruhs, K., Uthaisombut, P., & Woeginger, G. (2008). Getting the best response for your erg. *ACM Transactions on Algorithms*, *4*, 38:1–38:17.

Rackspace: Our 100% Network Uptime Guarantee, Resource Document. Rackspace US Inc. http://www.rackspace.co.uk/about-us/data-centres. Accessed December 17, 2015.

Shabtay, D., Bensoussan, Y., & Kaspi, M. (2012). A bicriteria approach to maximize the weighted number of just-in-time jobs and to minimize the total resource consumption cost in a two-machine flow-shop scheduling system. *International Journal of Production Economics*, *136*, 67–74.

Shabtay, D., & Steiner, G. (2007). A survey of scheduling with controllable processing times. *Discrete Applied Mathematics*, *155*, 1643–1666.

Shioura, A., Shakhlevich, N.V., & Strusevich, V.A. (2015). Energy saving computational models with speed scaling via submodular optimization. In *Proceedings of the 3rd international conference on green computing, technology and innovation (ICGCTI2015)*

Tian, W., & Zhao, Y. (2015). *Optimized cloud resource management and scheduling: Theory and practices*. Los Altos: Morgan Kaufmann.

Von Laszewski, G., Wang, L., Younge, A. J., & He, X. (2009). Power-aware scheduling of virtual machines in DVFS-enabled clusters. In *IEEE international conference on cluster computing and workshops (CLUSTER '09)* (pp. 1–10).

Wu, C.-M., Chang, R.-S., & Chan, H.-Y. (2014). A green energy-efficient scheduling algorithm using the DVFS technique for cloud datacenters. *Future Generation Computer Systems*, *37*, 141–147.

Yao, F. F., Demers, A. J., & Shenker, S. (1995). A scheduling model for reduced CPU energy. In *Proceedings of the 36th IEEE symposium on foundations of computer science (FOCS '95)* (pp. 374–382).