Sādhanā (2021) 46:46
https://doi.org/10.1007/s12046-021-01564-w

Energy-efficient scheduling: classification, bounds, and algorithms

PRAGATI AGRAWAL[1,*] and SHRISHA RAO[2]

[1]Department of Computer Science and Engineering, Maulana Azad National Institute of Technology, Bhopal, India
[2]International Institute of Information Technology Bangalore, Bangalore, India
e-mail: pragati.a.in@ieee.org; shrao@ieee.org

MS received 13 April 2020; revised 31 August 2020; accepted 6 December 2020

Abstract. The problem of attaining energy efficiency in distributed systems is of importance, but a general, non-domain-specific theory of energy-minimal scheduling is far from developed. In this paper, we classify the problems of energy-minimal scheduling and present theoretical foundations of the same. We derive results concerning energy-minimal scheduling of independent jobs in a distributed system with functionally similar machines with different working and idle power ratings. The machines considered in our system can have identical as well as different speeds. If the jobs can be divided into arbitrary parts, we show that the minimum-energy schedule can be generated in linear time and give exact scheduling algorithms. For the cases where jobs are non-divisible, we prove that the scheduling problems are NP-hard and also give approximation algorithms for the same along with their bounds.

Keywords. Scheduling; energy efficiency; approximation algorithms; distributed systems; computational complexity.

1. Introduction

Energy is a precious resource of the current industrial economy. We present a generalised theory for energy-efficient offline scheduling in this paper, which does not go into domain-specific issues or technologies. Our work is applicable to systems whose machines are similar in their capabilities but can have different working and idle power consumptions. The speeds at which machines can execute tasks can be different from one another, but are not time-varying, i.e., each machine operates at the same speed whenever working. All machines are connected to one another, so that a job from one machine can be transferred to any other machine. Machines and jobs to be executed are independent, and hence any job can be executed in any order on any machine. The theorems and algorithms given in the paper are relevant to all applications that satisfy the assumptions and the system model presented, irrespective of the domain or technology used. For example, scheduling in IoT devices, any job shop scheduling problem, any factory where there are several machines of similar capabilities and work is to be assigned to some of those machines such that the job is done using minimum energy, and many more. Given a set of interconnected machines and independent jobs, our results specify the allocation of jobs on the machines so that the total power consumption of the system is minimised. We classify the problems as follows.

1. *Identical speeds, divisible jobs*: This is discussed in Section 5.1a, where we derive results to prove that it can be solved in linear time and give an algorithm for the same.
2. *Identical speeds, non-divisible jobs*: This turns out to be NP-hard, as shown in Section 5.1b. We give an approximation algorithm that gives a solution within a bound of $\frac{4}{3}$.
3. *Different speeds, divisible jobs*: In Section 5.2a we give results and an algorithm to find the minimum-energy schedule.
4. *Different speeds, non-divisible jobs*: This is also NP-hard. We give an approximation algorithm in Section 5.2b that gives a result within a bound of $1 + \sqrt{3}/3 \approx 1.5773$ times the optimal.

We derive results (for divisible jobs) that serve as an upper-bound to maximum achievable efficiency—because a system with divisible jobs can always have higher efficiency as compared with system with non-divisible jobs [1]. In the present paper we deal with theoretical aspects of energy-minimal scheduling, deriving results that can guide the design of scheduling algorithms for systems. On any given schedule, if the conditions indicated in the results are satisfied then that schedule is energy-minimal. Likewise, the

*For correspondence
Published online: 26 February 2021

results can also indicate when a system of machines would be inherently wasteful of energy and can thus guide more energy-efficient system designs. Machines not in use are not assumed to be switched off, but do consume power. We have deliberately allowed that machines do not get switched off when not in use, given the reality of machines in many domains. The situation where a machine does get switched off is merely a special case where the idle power of that machine is zero.

The remainder of the paper is organised as follows. Related work is mentioned in Section 2. The system model along with the notation used and the motivation is presented in Section 3. Section 4 discusses the precedence of machines. The proposed algorithms along with their bounds are given in Section 5. Finally, Section 6 describes the two types of measures commonly used for energy efficiency in systems and shows that they are incompatible.

2. Related work

In energy-minimal scheduling, as opposed to classical makespan scheduling, there are additional parameters (energy specifications) that come into picture and hence it is inherently more complex than makespan scheduling. Agrawal and Rao [2] show that energy-efficient scheduling is strictly harder than makespan scheduling. Though there has been a lot of work done in the field of scheduling, the prime focus of general works on scheduling [1, 3–5] has been to optimise objectives related to time—such as makespan, earliness, and avoidance of tardiness.

Existing literature dealing with scheduling for energy reduction generally focuses on specific domains, such as communication networks, embedded systems, and high performance computing [6–11]. Pouwelse *et al* [12] give a heuristic for energy priority scheduling for variable voltage processors. Bambagini *et al* [13] use the combination of offline-DVFS and online-DPM (dynamic power management) techniques to reduce energy consumption in embedded systems. Some recent anthologies deal with energy efficiency in computing systems at various scales (processors to data centers) [14, 15].

In the domain of microprocessor power management and energy saving, some prior work uses speed scaling and power-down techniques [16, 17]. Irani *et al* [18] have given algorithms for power savings by putting the system on sleep state while idle and varying the speeds at which jobs are run. Augustine *et al* [19] suggest optimal power-down strategies with more than one low-power state. Bansal *et al* [20, 21] use another technique called speed scaling. Bansal *et al* [22] have given a tighter bound for the YDS algorithm given by Yao *et al* [23] and proved that their BKP algorithm is cooling-oblivious. Bansal *et al* [20, 21] consider an arbitrary power function of speed, for speed scaling. Arunarani *et al* [24] present a comprehensive survey of task scheduling strategies and the associated metrics suitable for cloud computing environments. Juarez *et al* [25] have given dynamic energy-aware scheduling for parallel task-based application in cloud computing. Gu *et al* [26] investigate the energy cost minimisation problem in edge computing. They jointly consider the task allocation, service migration, and energy scheduling and give a relaxation-based heuristic algorithm to reduce energy consumption.

There are some studies of scheduling whose theoretical developments can help in energy-efficient scheduling too. One such is divisible job theory, which studies methods involving the continuous and linear modeling of partitionable communication and computation jobs for parallel processing [27–32]. There is also some prior work concerning scheduling for multiple criteria [33–37].

The theoretical framework and classification of energy-minimal scheduling problems as considered here is not discussed in or derivable from any of these papers. The theoretical framework proposed is thus a necessary addition to existing and ongoing work focusing on heuristic approaches to energy-efficient scheduling, and also complements technology-specific approaches to energy reduction such as DVFS [38–40].

3. System model

As in other scheduling problems, we have a set of interconnected machines and a set of jobs that are to be executed using the machines. We presume that all machines are connected in such a way that a job can be transferred from any machine to any other machine without any energy dissipation. Jobs are independent and hence can be executed in any order and on any machine. We consider two types of jobs: divisible and non-divisible. There can be two types of systems as well, based on the speeds of the machines in the system: one in which all machines run at the same speed (even though their power ratings may be different), and the other class in which machines run at different speeds. Thus, based on the type of job and the type of system, we have four classes of scheduling problems. The notation used is indicated in table 1.

The energy consumed by a machine can be calculated given the power consumption by the machine in the working state, and the duration for which the machine works. Since the power rating of a machine is energy consumed per unit time, it is in principle possible to calculate the energy consumed by any machine over a given time duration by integrating the power consumption over time [41]. Machines may continue to consume energy even when not working, i.e., when idle. It is also to be noted that in many systems like data centers it is not a simple matter to shut off machines, even when they are not given any job.

Table 1. Notation.

Symbol	Meaning
$E_{m,r}$	Total energy consumption of system with m machines and jobs distributed to r machines
T	Makespan of the system
$\mathcal{C}$	Set of all machines $c_1, c_2, \ldots, c_m$
$\mathcal{R}$	Working set of machines (all others being idle)
r	Number of machines on which jobs are distributed
$\mu(c_i)$	Power consumption in working state by machine c_i
$\gamma(c_i)$	Power consumption in idle state by machine c_i
Γ	The sum of the idle power of all the machines
$\upsilon(c_i)$	Speed (throughput of work per unit time) of machine c_i (fixed throughout its working tenure)
$\tau(c_i)$	Time spent in working state by machine c_i
$\kappa(c_i)$	Time spent in idle state by machine c_i
$w(c_i)$	Amount of work given to machine c_i
$\mathcal{P}$	Set of jobs $p_1, p_2, \ldots, p_n$
$\psi(p_j)$	Weight of job p_j
W	Total working time of all the jobs

The expression for the energy consumed in a system of m machines can be given as the sum of the energies consumed by all machines:

$$E = \sum_{i=1}^{m} [\mu(c_i)\tau(c_i) + \gamma(c_i)(T - \tau(c_i))]. \tag{3.1}$$

The general energy equation can be formed by replacing $\tau(c_i)$ with $\frac{w(c_i)}{\upsilon(c_i)}$, and after simplification

$$E = \sum_{i=1}^{m} \left[\frac{w(c_i)}{\upsilon(c_i)} (\mu(c_i) - \gamma(c_i)) + \gamma(c_i)T \right]. \tag{3.2}$$

We assume the following in our system:

1. All jobs are independent, i.e., there are no precedence constraints between jobs. Hence there is no idle time (gap) for a machine between the execution of two jobs, i.e., $T = \max_i \tau(c_i)$
2. In the cases where jobs are divisible, the divisibility of jobs means that jobs can be arbitrarily divided and assigned to any machine.
3. All machines stay on for the duration of the makespan of the whole set of jobs.
4. For the model in which the speed of all machines is the same, all machines are indexed in increasing order of the differences of their working power and idle power, i.e., $\mu(c_i) - \gamma(c_i) \leq \mu(c_j) - \gamma(c_j) \forall i, j$ where $1 \leq i \leq j \leq m$. For the model where machines can have different speeds the machines are indexed in an order such that the first machine is the one with smallest $\frac{\mu(c_i) - \gamma(c_i) + \Gamma}{\upsilon(c_i)}$ and afterwards machines are indexed in non-decreasing order of $\frac{\mu(c_i) - \gamma(c_i)}{\upsilon(c_i)}$, where $1 \leq i \leq m$.
5. For the model in which the speed of all machines is the same, one unit of work takes one unit of time for execution irrespective of the machine on which it is executed.

The aim is to schedule a set of jobs on the given set of machines such that the energy consumption of the system E given by (3.1) and (3.2) is minimised.

4. Precedence of machines

In Section 4 we first find the precedence of machines, i.e., the order in which they should be assigned work. This precedence helps determine which machines should be allotted work and how much, to reduce the overall energy consumption of the system. With differing power specifications of machines, it stands to reason that it may be advantageous to prefer some machines over others in doing jobs. The problem of finding relative distribution of work among machines can be broken into two parts:

- Given the set of m machines, which subset of machines should be allowed to work and which should remain idle for all times during the makespan?
- What should be the distribution of work among the working machines?

There are no restrictions on the jobs except that the total load is considered to be W. This means that the results derived in this section are applicable to all types of systems. As the case where machines have identical speeds has already been presented elsewhere [42], we derive results for the case where they can have different speeds.

4.1 *Systems with different speed machines*

In this subsection, we use following results from our previous paper [42] to develop a theory for systems in which machines can have different speeds.

Lemma 4.1 *Given $\mu(c_k) - \gamma(c_k) \leq \mu(c_l) - \gamma(c_l)$, then for energy optimality of the system, $\tau(c_k) \geq \tau(c_l)$ $\forall k, l$, where $1 \leq k \leq l \leq m$.*

Theorem 4.2 *When $r - 1$ machines are working, for machine c_r to be given work (i.e., $\tau(c_r) \neq 0$) and result in reduced energy consumption, the following must hold:*

$$\sum_{i=1}^{r-1} [(\mu(c_i) - \gamma(c_i) - \mu(c_r) + \gamma(c_r))s(c_i)] + s(c_1) \sum_{i=1}^{m} \gamma(c_i) > 0.$$

Theorem 4.3 *If we give jobs to $r > 1$ machines (where r is number of working machines), then for minimum energy*

consumption, the distribution of jobs on all r machines should be equal and given by $\frac{W}{r}$.

Let us assume that work is assigned to only a subset $\mathcal{R}$ out of m machines, where $|\mathcal{R}| = r$. Hence

$$\tau(c_i) > 0 \forall i \in \mathcal{R} \quad \text{and} \quad \tau(c_k) = 0 \forall k \notin \mathcal{R}. \tag{4.1}$$

With a little rearrangement of (3.1) and using (4.1) in (3.1), we have

$$E_{m,r} = \sum_{i \in \mathcal{R}} [(\mu(c_i) - \gamma(c_i))]\tau(c_i) + \sum_{i=1}^{m} \gamma(c_i) T. \tag{4.2}$$

Here the makespan T is defined as

$$T = \max(\tau(c_i)) \forall i \in \mathcal{R}. \tag{4.3}$$

Putting the value of $\sum_{i=1}^{m} \gamma(c_i) = \Gamma$ in (4.2) gives

$$E_{m,r} = \sum_{i \in \mathcal{R}} [(\mu(c_i) - \gamma(c_i))]\tau(c_i) + \Gamma T. \tag{4.4}$$

Our aim is to find the set $\mathcal{R}$ for which $E_{m,r}$ is minimal. We first consider the question: if all the work has to be assigned to only one machine, then which one would that be? We present the answer in Lemma 4.4 of Section 4.

Lemma 4.4 *If all the work is assigned to one machine* c_i, *i.e.*, $|\mathcal{R}| = 1$ *and* $\mathcal{R} = \{c_i\}$, *then for energy minimality this should be the machine with minimum* $\frac{\mu(c_i) - \gamma(c_i) + \Gamma}{\upsilon(c_i)}$.

Proof Since all the work is assigned to only one machine c_i, the makespan is $T = \tau(c_i) = \frac{W}{\upsilon(c_i)}$. Substituting these values of $\mathcal{R}$, T, and $\tau(c_i)$ in (4.4), we get

$$E_{m,1} = \frac{W(\mu(c_i) - \gamma(c_i) + \Gamma)}{\upsilon(c_i)}. \tag{4.5}$$

Since W does not depend on machines, to minimise energy consumption $E_{m,1}$, we need to choose c_i for which $\frac{\mu(c_i) - \gamma(c_i) + \Gamma}{\upsilon(c_i)}$ is minimal. □

As mentioned in Assumption 4 in Section 3, we order the machines such that the first machine is with minimum $\frac{\mu(c_i) - \gamma(c_i) + \Gamma}{\upsilon(c_i)}$. Hence here $i = 1$, and the first machine is given work in this case.

Now before moving on to find out which other machines should be included in the working set, we formulate the amount of work that should be given to each machine in the working set.

Lemma 4.5 *If we give jobs to a set* $\mathcal{R}$ *of machines, then for minimum energy consumption, the working time of all r machines should be equal and given by* $\frac{W}{\sum_{i \in \mathcal{R}} \upsilon(c_i)}$.

Proof We prove this by induction. If the working time for all machines in set $\mathcal{R}$ is equal, then

$$\tau(c_i) = \tau(c_j) = \frac{W}{\sum_{i \in \mathcal{R}} \upsilon(c_i)} \quad \forall i, j \in \mathcal{R}. \tag{4.6}$$

From (4.3)

$$T = \frac{W}{\sum_{i \in \mathcal{R}} \upsilon(c_i)}. \tag{4.7}$$

Using (4.6) and (4.7) in (4.4) gives

$$E_{m,r} = W \left[\frac{\sum_{i \in \mathcal{R}} (\mu(c_i) - \gamma(c_i)) + \Gamma}{\sum_{i \in \mathcal{R}} \upsilon(c_i)} \right]. \tag{4.8}$$

Basis: As the basis step of induction, we show (4.8) and hence Lemma 4.5 holds for $|\mathcal{R}| = 1$. Substituting $\mathcal{R} = \{c_i\}$ in (4.8) gives

$$E_{m,1} = \frac{W(\mu(c_i) - \gamma(c_i) + \Gamma)}{\upsilon(c_i)}.$$

This is the same as (4.5). Thus it has been shown that basis step of induction holds.

Induction step: We show that if (4.8) holds for $|\mathcal{R}| = r$, then it also holds for $|\mathcal{R}'| = r + 1$. This means if we include another machine in the working set $\mathcal{R}$ to minimise the energy consumption, then all the machines in the new working set $\mathcal{R}'$ should work for equal amount of time. Mathematically, the new makespan should be given by

$$T_{r+1} = \frac{W}{\sum_{i \in \mathcal{R}'} \upsilon(c_i)}. \tag{4.9}$$

For ease of representation, let

$$p_r = \sum_{i \in \mathcal{R}} (\mu(c_i) - \gamma(c_i)) + \Gamma \& q_r = \sum_{i \in \mathcal{R}} \upsilon(c_i).$$

Now (4.8) can be written as

$$E_{m,r} = W \left[\frac{p_r}{q_r} \right].$$

Let the working time of machine c_{r+1} be $\tau(c_{r+1})$. The updated working time of all machines in set $\mathcal{R}$ becomes

$$T_{r+1} = \frac{W - \tau(c_{r+1})\upsilon(c_{r+1})}{q_r}. \tag{4.10}$$

We can assume $\tau(c_{r+1}) \leq T_{r+1}$, because if it is not the case then we swap machine c_{r+1} with any other machine in $\mathcal{R}$ that makes the assumption true; since $\tau(c_{r+1}) \leq T_{r+1}$, the new makespan of the system becomes T_{r+1}. Using these values in (4.4) gives

$$E_{m,r+1} = p_r T_{r+1} + (\mu(c_{r+1}) - \gamma(c_{r+1}))\tau(c_{r+1}).$$

Substituting T_{r+1} from (4.10) gives

$$\begin{aligned} E_{m,r+1} =& p_r \left[\frac{W - \tau(c_{r+1})\upsilon(c_{r+1})}{q_r}\right] \\ &+ (\mu(c_{r+1}) - \gamma(c_{r+1}))\tau(c_{r+1}) \\ =& E_{m,r} - \tau(c_{r+1}) \\ &\left[\frac{p_r \upsilon(c_{r+1})}{q_r} - (\mu(c_{r+1}) - \gamma(c_{r+1}))\right] \\ E_{m,r} - E_{m,r+1} =& \tau(c_{r+1}) \\ &\left[\frac{p_r \upsilon(c_{r+1})}{q_r} - (\mu(c_{r+1}) - \gamma(c_{r+1}))\right]. \end{aligned} \tag{4.11}$$

Since the energy consumption should decrease by the addition of machine c_{r+1} to the working set, i.e., $E_{m,r} - E_{m,r+1} > 0$, the R.H.S. of (4.11) has to be positive. Since time cannot be negative, both the sub-terms $\tau(c_{r+1})$ and $\left[\frac{p_r \upsilon(c_{r+1})}{q_r} - (\mu(c_{r+1}) - \gamma(c_{r+1}))\right]$ must be positive. The term $\left[\frac{p_r \upsilon(c_{r+1})}{q_r} - (\mu(c_{r+1}) - \gamma(c_{r+1}))\right]$ is independent of working time. Hence, to maximise $E_{m,r} - E_{m,r+1}$, we need to maximise $\tau(c_{r+1})$. As we know the maximum value of $\tau(c_{r+1}) = T_{r+1}$, using this in (4.10) and after simplifying, we get

$$T_{r+1} \frac{W}{\sum_{i \in \mathcal{R}'} \upsilon(c_i)}.$$

This shows that (4.9) is true. Hence our induction step is also proved. This completes the proof that all the machines in the working set should be working for equal amount of time for minimal energy consumption. □

Using Lemma 4.5, the energy consumed by the system when machines in set $\mathcal{R}$ are working is given by

$$E_{m,r} = W \left[\frac{\sum_{i \in \mathcal{R}} (\mu(c_i) - \gamma(c_i)) + \Gamma}{\sum_{i \in \mathcal{R}} \upsilon(c_i)}\right].$$

Since our goal is to find the relative distribution of work, we can assume with no loss of generality that the total work is one unit to make the representation simpler:

$$E_{m,r} = \frac{\sum_{i \in \mathcal{R}} (\mu(c_i) - \gamma(c_i)) + \Gamma}{\sum_{i \in \mathcal{R}} \upsilon(c_i)}. \tag{4.12}$$

It can be noted here that irrespective of machines being of identical or different speeds, the machines in the working set should be working for equal amount of time for minimal energy consumption.

We already found which machine to give work to if only one machine can be assigned work. Now, if we want to give jobs to two machines, then we do this only if the energy consumed when two machines are used is less than the energy consumption with just one machine. The same principle applies whenever we want to expand the working set of machines. The new expanded set must have lower energy consumption than the previous set. In theorem 4.6, we specify the conditions under which a set can be expanded.

Theorem 4.6 *If a machine c_j has to be included in the working set of machines $\mathcal{R}$, then the following condition must be satisfied*:

$$\frac{\sum_{i \in \mathcal{R}} (\mu(c_i) - \gamma(c_i)) + \Gamma}{\sum_{i \in \mathcal{R}} \upsilon(c_i)} > \frac{\mu(c_j) - \gamma(c_j)}{\upsilon(c_j)}. \tag{4.13}$$

Proof After a little re-arrangement in our condition (4.13), we get

$$\begin{aligned} &\frac{\sum_{i \in \mathcal{R}} (\mu(c_i) - \gamma(c_i)) + \Gamma}{\sum_{i \in \mathcal{R}} \upsilon(c_i)} \\ &> \frac{\sum_{i \in \mathcal{R}} (\mu(c_i) - \gamma(c_i)) + \Gamma + \mu(c_j) - \gamma(c_j)}{\sum_{i \in \mathcal{R}} \upsilon(c_i) + \upsilon(c_j)}. \end{aligned}$$

Let $\mathcal{R}'$ be the new set formed after including c_j in $\mathcal{R}$.

$$\begin{aligned} &\frac{\sum_{i \in \mathcal{R}} (\mu(c_i) - \gamma(c_i)) + \Gamma}{\sum_{i \in \mathcal{R}} \upsilon(c_i)} \\ &> \frac{\sum_{i \in \mathcal{R}'} (\mu(c_i) - \gamma(c_i)) + \Gamma}{\sum_{i \in \mathcal{R}'} \upsilon(c_i)}. \end{aligned} \tag{4.14}$$

The left hand side of (4.14) gives the energy consumption by working set $\mathcal{R}$. The right hand side of (4.14) gives the energy consumption when machine c_j is included in the original working set. Hence the energy consumption of the new set is lower when the condition (4.13) is satisfied. □

Theorem 4.6 gives the condition in which a machine could be included in the working set to reduce energy consumption of the system. However, there can be many machines that satisfy this condition. Theorem 4.7 specifies which machine should be given preference for inclusion in the working set.

Theorem 4.7 *Given any two machines c_j and c_k that qualify to be included in working set $\mathcal{R}$ according to Theorem 4.6, i.e.*

$$\frac{\mu(c_j) - \gamma(c_j)}{\upsilon(c_j)} < \frac{p_r}{q_r} \tag{4.15}$$

and

$$\frac{\mu(c_k) - \gamma(c_k)}{\upsilon(c_k)} < \frac{p_r}{q_r}, \tag{4.16}$$

for minimal energy consumption, machine c_j should be chosen over c_k if $\upsilon(c_j) > \upsilon(c_k)$ and

$$\frac{\mu(c_j) - \gamma(c_j)}{\upsilon(c_j)} < \frac{\mu(c_k) - \gamma(c_k)}{\upsilon(c_k)}. \tag{4.17}$$

Proof Given the condition (4.17), we would like to derive energy equations from it. Multiplying the denominators of both sides in (4.17), we get

$$\begin{aligned}&(\mu(c_j)-\gamma(c_j))\upsilon(c_k)\\&<(\mu(c_k)-\gamma(c_k))\upsilon(c_j).\end{aligned}\tag{4.18}$$

By adding and subtracting $(\mu(c_j)-\gamma(c_j))\upsilon(c_j)$ in (4.18), we get

$$\begin{aligned}&\frac{(\mu(c_j)-\gamma(c_j))-(\mu(c_k)-\gamma(c_k))}{\upsilon(c_j)-\upsilon(c_k)}\\&<\frac{(\mu(c_j)-\gamma(c_j))}{\upsilon(c_j)}.\end{aligned}\tag{4.19}$$

From (4.15) and (4.19), we get

$$\begin{aligned}&(\mu(c_j)-\gamma(c_j))q_r-(\mu(c_k)-\gamma(c_k))q_r\\&-p_r\upsilon(c_j)+p_r\upsilon(c_k)<0.\end{aligned}\tag{4.20}$$

Adding (4.18) and (4.20), we get

$$\begin{aligned}&(\mu(c_j)-\gamma(c_j))q_r-(\mu(c_k)-\gamma(c_k))q_r-p_r\upsilon(c_j)+p_r\upsilon(c_k)\\&+(\mu(c_j)-\gamma(c_j))\upsilon(c_k)-(\mu(c_k)-\gamma(c_k))\upsilon(c_j)<0.\end{aligned}\tag{4.21}$$

Now adding and subtracting p_rq_r in (4.21), we get

$$\frac{p_r+\mu(c_j)-\gamma(c_j)}{q_r+\upsilon(c_j)}<\frac{p_r+\mu(c_k)-\gamma(c_k)}{q_r+\upsilon(c_k)}.\tag{4.22}$$

In (4.22) the left hand side gives the energy consumption of the system when machine c_j is included in the working set $\mathcal{R}$, while the right hand side gives the energy consumption of the system when machine c_k is included. The energy consumption is lower with machine c_j in comparison with machine c_k and hence machine c_j would be given preference to be included in working set. (NB: after including c_j in $\mathcal{R}$, the condition given in Theorem 4.6 is checked again). □

In Theorem 4.7, we considered the special condition $\upsilon(c_j)>\upsilon(c_k)$. However, there can be cases where $\frac{\mu(c_j)-\gamma(c_j)}{\upsilon(c_j)}<\frac{\mu(c_k)-\gamma(c_k)}{\upsilon(c_k)}$ but $\upsilon(c_j)\leq\upsilon(c_k)$. In certain cases, it may be possible that choosing machine c_k over c_j would be more energy efficient. These special cases are fairly rare; hence, we would like to use Theorem 4.7 for the rest of our analysis. However, we prove here that even if we choose c_j before c_k while forming our working set, c_k would be definitely included in the working set later on.

Theorem 4.8 *Given any two machines c_j and c_k that qualify to be included in working set $\mathcal{R}$ according to Theorem 4.6, i.e.*

$$\frac{\mu(c_j)-\gamma(c_j)}{\upsilon(c_j)}<\frac{\mu(c_k)-\gamma(c_k)}{\upsilon(c_k)}<\frac{p_r}{q_r}\tag{4.23}$$

where $\upsilon(c_j)<\upsilon(c_k)$ and

$$\frac{p_r+\mu(c_j)-\gamma(c_j)}{q_r+\upsilon(c_j)}>\frac{p_r+\mu(c_k)-\gamma(c_k)}{q_r+\upsilon(c_k)}.\tag{4.24}$$

The optimal working set of machines contain c_k, i.e.

$$\begin{aligned}&\frac{p_r+\mu(c_j)-\gamma(c_j)+\mu(c_k)-\gamma(c_k)}{q_r+\upsilon(c_j)+\upsilon(c_k)}\\&<\frac{p_r+\mu(c_j)-\gamma(c_j)}{q_r+\upsilon(c_j)}.\end{aligned}\tag{4.25}$$

Proof From (4.23), we have

$$\frac{\mu(c_k)-\gamma(c_k)}{\upsilon(c_k)}<\frac{p_r}{q_r}.\tag{4.26}$$

Multiplying with $\upsilon(c_k)(\mu(c_k)-\gamma(c_k))$ on both sides, we get

$$\frac{\mu(c_k)-\gamma(c_k)}{\upsilon(c_k)}<\frac{p_r+\mu(c_k)-\gamma(c_k)}{q_r+\upsilon(c_k)}.\tag{4.27}$$

Combining (4.24) and (4.27) and adding $(p_r+\mu(c_j)-\gamma(c_j))(q_r+\upsilon(c_j)$ to both sides and solving, we have

$$\begin{aligned}&\frac{p_r+\mu(c_j)-\gamma(c_j)+\mu(c_k)-\gamma(c_k)}{q_r+\upsilon(c_j)+\upsilon(c_k)}\\&<\frac{p_r+\mu(c_j)-\gamma(c_j)}{q_r+\upsilon(c_j)}.\end{aligned}$$

Since the energy consumption by including machine c_k is lower than with just machine c_j, machine c_k is always in the working set. □

For the rest of our paper we stick to Theorem 4.7 for finding precedence of machines as the case explained in Theorem 4.8 is rare and does not affect our analysis. When we put Lemma 4.4 and Theorem 4.7 together, we get the order in which machines get preference to be allotted work. This order is the same as mentioned in Assumption 4 in Section 3. Combining Lemma 4.4, Theorem 4.7, and Assumption 4, we can state the following.

Corollary 4.9 *If the machines are indexed in such an order that the first machine is the one with smallest $\frac{\mu(c_i)-\gamma(c_i)+\Gamma}{\upsilon(c_i)}$ and later machines are in increasing order of $\frac{\mu(c_i)-\gamma(c_i)}{\upsilon(c_i)}$, and if the working set contains r machines, then for minimal energy consumption $\mathcal{R}=\{c_1,c_2,\ldots,c_r\}$, where each machine in this set works for an equal amount of time.*

Using the Corollary 4.9, (4.12) can be re-written as

$$E_{m,r} = W\left[\frac{\sum_{i=1}^{r}(\mu(c_i) - \gamma(c_i)) + \Gamma}{\sum_{i=1}^{r} \upsilon(c_i)}\right], \tag{4.28}$$

which gives the energy consumption of a system with total work W.

From (4.28) we see that energy is dependent on work W, but Theorems 4.6 and 4.7 show that finding the number of working machines (the value of r) from the given set $\mathcal{C}$ of machines is independent of work W. Hence the scheduling decisions do not depend upon the quantum of the work given to the system.

In Section 4, we found the precedence among machines for allotting work based on their energy and speed specifications. The results derived in Section 4 are applicable for handling any class of jobs, which means that the precedence of machines remains the same irrespective of whether the jobs are divisible, non-divisible, or whether they have precedence constraints. In the next sections we consider different classes of systems based on types of jobs.

5. Algorithms and complexity

Depending upon the jobs to be executed, systems can be classified into many categories as already mentioned in Section 3. In the current section we analyse the complexity of scheduling problems for these various types of systems and give scheduling algorithms for the same. We first analyse systems with identical speed machines and then move ahead to those with different speeds.

5.1 *Systems with identical speed machines*

Section 4 gives results that govern the scheduling of systems with identical speed machines. Given the total amount of work and energy specifications of machines, we can find the numbers of machines that have to be assigned work using results of the Section 4. Algorithm 1 is an implementation of those results.

Algorithm 1 takes the number of machines, the working power and idle power ratings of the machines, and the total work to be done as inputs, and gives the value of r, i.e., the number of working machines, as output. The machines are indexed in the precedence order given by Assumption 4 in Section 3. Now we apply binary search to find the value of r, such that we get the minimal energy value of the system. Algorithm 1 is based on the previously given results. The computation complexity of Algorithm 1 is $\mathcal{O}(m \log m)$.

Algorithm 1: Find the number of working machines of the system in order to minimise the energy consumption when the speeds of machines are identical.

input : Number of machines (m), working power of machines ($\mu(c_i)$), idle power of machines ($\gamma(c_i)$), total work to be done (W)
output: Number of working machines (r), makespan (T)

for $i = 1$ *to* m **do**
  calculate $\mu(c_i) - \gamma(c_i)$
end
sort $(\mu(c_i) - \gamma(c_i))$;
$low \leftarrow 1$, $high \leftarrow m$;
while $low \leq high$ **do**
  $mid \leftarrow low + \left\lfloor \frac{(high-low)}{2} \right\rfloor$;
  // $E(k) \leftarrow (\frac{W}{k})[\sum_{i=1}^{k} \mu(c_i) + \sum_{i=k+1}^{m} \gamma(c_i)]$;
  Calculate $E(mid-1), E(mid), E(mid+1)$;
  if $(E(mid-1) > E(mid))\&\&(E(mid) \leq E(mid+1))$ **then**
    $r \leftarrow mid$;
    return r;
  else
    if $E(mid-1) > E(mid)$ **then**
      $low \leftarrow mid$;
    else
      $high \leftarrow mid$;
    end
  end
end
$T \leftarrow \frac{W}{r}$.

We now find the complexities of scheduling problems, and give algorithms for divisible and non-divisible jobs.

5.1a *Divisible jobs*: In this case we are given a set $\mathcal{P}$ of jobs, where these jobs can be arbitrarily broken into any number of smaller jobs and these smaller jobs can be executed in parallel on different machines. We make this assumption of divisibility to simplify the analysis, and consider the energy consumption overhead in the division of jobs to be nil. Since the jobs are divisible they can be trivially distributed over r machines with makespan T, where r and T are given by Algorithm 1.

Hence, the minimal energy of a system with divisible jobs can be calculated using (5.1):

$$E_{m,r} = T\left[\sum_{i=1}^{r} \mu(c_i) + \sum_{i=r+1}^{m} \gamma(c_i)\right]. \tag{5.1}$$

Using Algorithm 1 and (5.1), we can find the energy of the system.

We now analyse the complexity of scheduling non-divisible jobs.

5.1b *Non-divisible jobs*: Given the set $\mathcal{P}$ of non-divisible jobs and the time required to execute job p_j on a machine with unit speed $\psi(p_j)$, our aim is to distribute the set $\mathcal{P}$ of jobs among the given set $\mathcal{C}$ of machines so that energy consumption is minimised. From Lemma 4.5 it is evident that whichever machines are chosen to work, they should work for an equal amount of time to achieve energy minimality. However, it is not straightforward or sometimes even possible to distribute

work in such a way when jobs are non-divisible. We prove that it is an NP-hard problem to calculate the minimal-energy schedule when the jobs are not divisible.

Proposition 5.1 *In a system where machines have identical speeds and jobs are non-divisible, computing the energy-minimal schedule is* NP*-hard.*

Proof Given a finite set of positive numbers and another positive number called the goal, to find the subset whose sum is closest to the goal is well known as the classical *subset-sum problem* [43].

It can be written mathematically as follows: given n natural numbers a_i and a target number B, we are asked to find an $S \subset \{1, 2, \ldots, n\}$ with

$$\sum_{i \in S} a_i = B. \tag{5.2}$$

This is reduced to our problem. Let r denote the number of machines that are given work and T be the makespan according to Algorithm 1. Since all machines have identical speed, the scheduling problem is to make exclusive subsets P_i from set $\mathcal{P}$ such that the sum of work in set P_i is T, i.e.,

$$\sum_{j \in P_i} \psi(p_j) = T \quad \forall i, 1 \le i \le r. \tag{5.3}$$

In the current scenario, the set of positive numbers $\{\psi(p_j) : j \in \mathcal{P}\}$ has to be distributed in subsets P_i such that sum of each subset is the positive number T. For each subset (P_i ; $i \le r$), this problem of finding the exclusive subsets P_i can be seen as the classical *subset-sum problem*. Since the subset-sum problem is NP-hard [43], and the subset-sum problem is reduced to our problem, energy-minimal scheduling of jobs in a system where jobs are non-divisible is also NP-hard. □

Since finding the energy-minimal schedule is NP-hard, we develop an approximation algorithm to find energy-efficient schedules. Though we strive to achieve the makespan given by Algorithm 1 for divisible jobs, in a system with non-divisible jobs it is not always possible to achieve it. We state the result that specifies the ideal makespan T_o and working set of machines $\mathcal{R}_o$ for energy minimality of a system with non-divisible jobs.

Lemma 5.2 *Given a system with identical speed machines and a set $\mathcal{P}$ of jobs with longest job of length $\psi(p_{max})$, the makespan is given by $T_o = \max(T, \psi(p_{max}))$ and the working set of machines $\mathcal{R}_o = \{c_1, c_2, \ldots, c_{r_o}\}$ where*

$$r_o = \left\lceil \frac{W}{T_o} \right\rceil. \tag{5.4}$$

Proof It is not possible to have a makespan lower than $\psi(p_{max})$, since jobs cannot be divided. Also, if we take T_o to be less than T, we cannot have an energy-minimal schedule. Hence for energy minimality, $T_o = \max(T, \psi(p_{max}))$.

From Theorem 4.3, it is evident that work should be distributed equally if possible to the working machines. Hence, with a possibly increased makespan, the number of working machines might get decreased and given by (5.4). Also, it is evident that the working set of machines is given by $\mathcal{R}_o = \{c_1, c_2, \ldots, c_{r_o}\}$. □

Algorithm 2: Approximation algorithm for energy-efficient scheduling of the system when the speeds of machines are identical.

input : Number of machines (m), working power of machines ($\mu(c_i)$), idle power of machines ($\gamma(c_i)$), total work to be done (W), $\mathcal{P}$, $\psi(p_j)$, r_o, T_o
output: Makespan (T), energy (E)

$S \leftarrow \{\}$;
for $i = 1$ *to* m **do**
  $d_i \leftarrow \mu(c_i) - \gamma(c_i)$;
  $S \leftarrow S \cup \{d_i\}$;
end
sort S ;
$J \leftarrow \{\}$;
for $j = 1$ *to* n **do**
  $J \leftarrow J \cup \{\psi(p_j)\}$;
end
reverse-sort (J);
$r \leftarrow r_o$;
$P_j \leftarrow \{\}$;
for $j = 1$ *to* r **do**
  $P_j \leftarrow P_j \cup \{p_j\}$;
  $b_j \leftarrow \sum_{j \in P_j} \psi(p_j)$;
end
for $l = r + 1$ *to* n **do**
  calculate min(b_j);
  $b_u \leftarrow \min(b_j)$;
  $u \leftarrow j$;
  $P_j \leftarrow P_j \cup \{p_l\}$;
  $b_j \leftarrow \sum_{j \in P_j} \psi(p_j)$;
end
for $i = 1$ *to* r **do**
  assign p_i to c_i $(\forall p_i \in P_i)$;
end
for $i = 1$ *to* r **do**
  $\tau(c_i) \leftarrow \sum_{j \in P_j} \psi(p_j)$;
end
sort $\tau(c_i)$;
$T \leftarrow \max(\tau(c_i))$;
for $i = 1$ *to* r **do**
  $\kappa(c_i) \leftarrow T - \tau(c_i)$;
end
$T \leftarrow \max \tau(c_i)$ $(\forall p_i \in P_i)$;
$E \leftarrow \sum_{i=1}^{r} \mu(c_i)\tau(c_i) + \sum_{i=1}^{r} \gamma(c_i)\kappa(c_i) + T \sum_{i=r+1}^{m} \gamma(c_i)$;

It is not always possible to keep each machine working till time T_o. Some machines might work for more than T_o and some might get less than that because of non-divisibility of jobs. Lemma 4.1 shows that the amount of work should be in decreasing order by the machines' indices. Taking all this information into consideration we have devised Algorithm 2, an approximation algorithm that is an adaptation of list-scheduling, in order of non-decreasing processing times for energy-minimal scheduling of non-divisible jobs.

In Algorithm 2, we index the machines in non-decreasing order of the difference of their working power and idle power. Jobs are indexed in non-increasing order of their weight. Initially, r idle machines are considered for the jobs to be done. First, r jobs are chosen and given to each r machines such that each machine has one job.

Now, to distribute the remaining $n-r$ jobs to the machines, we follow a sequential process. The weighted sum of jobs on each machine is calculated, and the next job is assigned to the machine in which the value is the lowest. The same process is repeated until all jobs are assigned to one or the other machine or sets. Next, we index the machines in non-increasing order of their total weights. We assign the jobs of set i to machine c_i. Now, the working times of all the machines are calculated. Hence we can calculate the makespan, which is the maximum working time of all the machines.

The approximation algorithm 2 is inspired by Graham's algorithm [44], which arranges a set of independent non-divisible jobs on identical speed machines such that the makespan is minimum. We made some changes in the algorithm to mould it for different power specifications of machines. We now calculate the bound on deviation from ideal energy consumption due to approximation.

Theorem 5.3 *The maximum possible ratio of energy consumption using Algorithm* 2 *to the ideal energy consumption is given by*

$$\frac{E^*{}_{max}}{E_o} = 1 + \frac{(\frac{4}{3} - \frac{1}{3r_o} - 1)\Gamma}{\sum_{i=1}^{r_o}(\mu(c_i) - \gamma(c_i)) + \Gamma}. \tag{5.5}$$

Proof From (3.1), if all the r_o machines work for an equal amount of time T_o, the energy consumption is given by

$$E_o = T_o[\sum_{i=1}^{r_o}(\mu(c_i) - \gamma(c_i)) + \Gamma]. \tag{5.6}$$

From Algorithm 2, the energy consumed is given by

$$E^* = \sum_{i=1}^{r_o}(\mu(c_i) - \gamma(c_i))\tau^*(c_i) + \Gamma T^*. \tag{5.7}$$

Assume that the algorithm gives such $\tau^*{}_i$ that all machines other than c_j and c_k are allotted work equal to T_o. Mathematically

$$\tau^*(c_j) = T_o + \epsilon, \quad \tau^*(c_k) = T_o - \epsilon \tag{5.8}$$

where $j<k\leq r_o$, and

$$\tau^*(c_i) = T_o \quad \forall i \neq j,k, \quad i \leq r_o. \tag{5.9}$$

Since $j<k$, according to our algorithm $\tau^*(c_j) > \tau^*(c_k)$ and hence $\epsilon \geq 0$. Using these values from (5.8) and (5.9) in (5.7) and simplifying, we get

$$\begin{aligned} E^* =& (\mu(c_j) - \gamma(c_j) - (\mu(c_k) - \gamma(c_k)))\epsilon \\ &+ \sum_{i=1}^{r_o}(\mu(c_i) - \gamma(c_i))T_o + \Gamma T^*. \end{aligned} \tag{5.10}$$

Now since $j<k$, we have $\mu(c_j) - \gamma(c_j) - (\mu(c_k) - \gamma(c_k)) \leq 0$. To compute the bound we look for the case where E^* is maximum. This occurs when

$$\mu(c_j) - \gamma(c_j) - (\mu(c_k) - \gamma(c_k)) = 0. \tag{5.11}$$

Using (5.11) in (5.10), we get

$$E^*{}_{max} = \sum_{i=1}^{r_o}(\mu(c_i) - \gamma(c_i))T_o + \Gamma T^*. \tag{5.12}$$

Now according to [44], the bound on makespan is given by

$$\frac{T^*}{T_o} = \frac{4}{3} - \frac{1}{3r_o}. \tag{5.13}$$

Using this bound in (5.12), and after simplifying, we get

$$\begin{aligned} E^*{}_{max} =& T_o\left(\sum_{i=1}^{r_o}(\mu(c_i) - \gamma(c_i)) + \Gamma\right) \\ &+ T_o\Gamma\left(\frac{4}{3} - \frac{1}{3r_o} - 1\right). \end{aligned} \tag{5.14}$$

Substituting the values, we get

$$\frac{E^*{}_{max}}{E_o} = \frac{T_o(\sum_{i=1}^{r_o}(\mu(c_i) - \gamma(c_i)) + \Gamma]) + T_o\Gamma\left(\frac{4}{3} - \frac{1}{3r_o} - 1\right)}{T_o\left[\sum_{i=1}^{r_o}(\mu(c_i) - \gamma(c_i)) + \Gamma\right]} \tag{5.15}$$

$$\frac{E^*{}_{max}}{E_o} = 1 + \frac{\Gamma(\frac{4}{3} - \frac{1}{3r_o} - 1)}{\sum_{i=1}^{r_o}(\mu(c_i) - \gamma(c_i)) + \Gamma}. \tag{5.16}$$

In the worst case the values of working power and idle power are equal, i.e., $\mu(c_i) = \gamma(c_i) \forall i$, so the bound is given by

$$\frac{E^*{}_{max}}{E_o} = 1 + \frac{\Gamma(\frac{4}{3} - \frac{1}{3r_o} - 1)}{\sum_{i=1}^{r_o}(\mu(c_i) - \gamma(c_i)) + \Gamma} = \frac{4}{3} - \frac{1}{3r_o}.$$

If r_o is very large, i.e., when $r_o \to \infty$, then

$$\frac{E^*{}_{max}}{E_o} = \frac{4}{3}. \tag{5.17}$$

Equation (5.16) gives the lower bound of inefficiency of our algorithm and (5.17) gives the upper bound.

We now move ahead to analyse the class of scheduling problems in which machines can have different working speeds.

5.2 *Systems with different speed machines*

In Section 5.2, along with power specifications of machines we also consider the speeds of machines to be different and give algorithms for both divisible jobs and non-divisible jobs. In Section 4.1, we proved various results that govern the scheduling of systems with different speed machines. Using these results Algorithm 3 finds the set of machines that should be assigned work for the energy-minimal scheduling when the machines of the system have different speeds.

Algorithm 3 takes the number of machines, the working power, idle power, and speeds of the machines, and the total work to be done as inputs and gives the value of r, i.e., the number of working machines, as output. In the algorithm we first index the machines in their precedence order, as given in Assumption 4 in Section 3. Then to calculate the value of r, we check the condition in (4.13) from Theorem 4.6. Then from the given values of total work to be done and the computed value of number of machines, we calculate the makespan. The algorithm also computes the amount of work that has to be given to the respective machines. The complexity of Algorithm 3 is $\mathcal{O}(m \log m)$.

Algorithm 3: Find the working machines and makespan of the system when the speeds of machines are different.

input : Number of machines (m), working power of machines ($\mu(c_i)$), idle power of machines ($\gamma(c_i)$), sum of idle power of all the machines (Γ), speed of machines ($\upsilon(c_i)$), total work to be done (W)

output: Number of working machines (r), makespan (T)

for $i = 1$ *to* m **do**
  calculate $\frac{\mu(c_i)-\gamma(c_i)+\Gamma}{\upsilon(c_i)}$
end
$\frac{\mu(c_1)-\gamma(c_1)+\Gamma}{\upsilon(c_1)} \leftarrow min(\frac{\mu(c_i)-\gamma(c_i)+\Gamma}{\upsilon(c_i)})$;
for $i = 2$ *to* m **do**
  calculate $\frac{\mu(c_i)-\gamma(c_i)}{\upsilon(c_i)}$
end
sort $(\mu(c_i) - \gamma(c_i))$;
for $r = 1$ *to* m **do**
  $E(r) \leftarrow [\frac{\sum_{i=1}^{r}(\mu(c_i)-\gamma(c_i))+\Gamma}{\sum_{i=1}^{r}\upsilon(c_i)}]W$;
  if $\frac{\mu(c_{r+1})-\gamma(c_{r+1})}{\upsilon(c_{r+1})} < E(r)$ **then**
    $r \leftarrow r + 1$;
    return r;
  else
    stop;
  end
end
$T \leftarrow \frac{W}{r}$;
$w(c_i) \leftarrow T * \upsilon(c_i)$.

We now find the complexity of the scheduling problems, and give algorithms for divisible and non-divisible jobs.

5.2a *Divisible jobs*: In this case, jobs can be arbitrarily broken into any number of smaller jobs and these smaller jobs can be executed in parallel on different machines. Also we consider the energy consumption overhead due to division/breaking of jobs overhead to be nil. In this scenario the jobs can be trivially distributed over r machines such that machine c_i gets $w(c_i)$ amount of work (as specified by Algorithm 3). The energy consumption of such a system with divisible jobs is given by

$$E = \sum_{i=1}^{m}\left[\mu(c_i)\frac{w(c_i)}{\upsilon(c_i)} + \gamma(c_i)\left(T - \frac{w(c_i)}{\upsilon(c_i)}\right)\right]. \tag{5.18}$$

Clearly the scheduling of divisible jobs can be done using Algorithm 3, which is a linear-time algorithm.

5.2b *Non-divisible jobs*: We first prove that scheduling non-divisible jobs for energy optimality on a system with machines having different speeds is an NP-hard problem. We then give an approximation algorithm for the same.

Proposition 5.4 *In a system where machines have different speeds and jobs are non-divisible, computing the energy-minimal schedule is* NP *hard.*

Proof Given the subset-sum problem

$$\sum_{i\in S} a_i = B. \tag{5.19}$$

If the amount of work $w(c_i)$ that has to be assigned to machine c_i is given by Algorithm 3 we need to make exclusive subsets P_i from set $\mathcal{P}$ such that the sum of work in set P_i is $w(c_i)$, i.e.,

$$\sum_{j\in P_i}\psi(p_j) = w(c_i) \quad \forall i, 1 \le i \le r. \tag{5.20}$$

As also discussed with Proposition 5.1 the subset-sum problem, which is NP-hard [43], is in the form of this problem.

Since finding the energy-minimal schedule is NP-hard, we develop an approximation algorithm to find energy-efficient schedules. It is possible that due to size of the jobs, the ideal makespan specified by Algorithm 3 cannot be achieved. In Lemma 5.5 we specify the best achievable makespan.

Lemma 5.5 *Given a system of machines with different speeds and a set $\mathcal{P}$ of jobs with the longest job of length $\psi(p_{max})$ and speed of the fastest machine given by υ_{max}, the best possible achievable makespan is given by*

$$T_o = \max\left(T, \frac{\psi_{max}}{\upsilon_{max}}\right) \tag{5.21}$$

where T is given by Algorithm 3.

Proof If there is a job that cannot be completed within T, then irrespective of the arrangement of jobs, we increase the best achievable makespan T_o to accommodate that biggest job. The biggest job cannot be completed any earlier than $\frac{\psi_{max}}{\upsilon_{max}}$. Also, any schedule with makespan lower than T should be suboptimal. Hence

$$T_o = \max\left(T, \frac{\psi_{max}}{\upsilon_{max}}\right).$$

□

Algorithm 4: Approximation algorithm for energy-efficient scheduling of the system when the speeds of machines are different.

input : Number of machines (m), working power of machines ($\mu(c_i)$), idle power of machines ($\gamma(c_i)$), speed of machines ($\upsilon(c_i)$), total work to be done (W), $\mathcal{P}$, $\psi(p_j)$, r_o, T_o

output: Makespan (T), energy (E)

```
for i = 1 to m do
    A(i) ← (μ(c_i) − γ(c_i)) / υ(c_i) ;
end
sort A(i) in non-decreasing order;
for j = 1 to n do
    B(j) ← ψ(p_j) ;
end
sort B(j) in non-increasing order ;
temp_i ← 0 ;
for j = 1 to n do
    for i = 1 to r_o do
        temp_i ← ψ(p_j)/υ_i + temp_i ;
    end
    assign p_j to c_i, where c_i has min(temp_i) ;
end
T ← max τ(c_i);
E ← Σ_{i=1}^{m} [μ(c_i) w(c_i)/υ(c_i) + γ(c_i)(T − w(c_i)/υ(c_i))].
```

If the makespan is increased then there can be possible reduction in number of working machines. The ideal number of working machines r_o is given by the minimum number of machines that satisfy

$$\frac{W}{\sum_{i=1}^{r_o} \upsilon(c_i)} \geq T_o. \tag{5.22}$$

Using this r_o as input we present Algorithm 4, which seeks to distribute jobs amongst machines with different speeds such that the machines will work for an equal amount of time.

In Algorithm 4 we first index the machines in non-decreasing order of the ratio of the difference of working power and idle power, to the speed of the machine. We also index the jobs in non-increasing order of their weights. Now we take a job at a time and find which machine it would finish first, based on the speed of the machine; then we assign the job to that particular machine. Similarly the process is repeated till all the jobs are assigned to machines. Now the working time of all machines is calculated, and the maximum working time of all machines is the makespan.

Algorithm 4, being an approximation algorithm, may not achieve the exact T_o, but it is within certain bounds.

The bound for the makespan as given by Gonzalez *et al* [45] of such LPT (largest processing time) schedules is

$$\frac{2m}{m+1}. \tag{5.23}$$

Given this bound on makespan, we derive the bound on the energy consumption of our algorithm.

Theorem 5.6 *When the speeds as well as the power specifications of the machines are different, and when jobs are non-divisible, then the maximum possible ratio of energy consumption using Algorithm* 4 *and ideal energy consumption is given by*

$$\frac{E^*_{max}}{E_o} \leq \frac{2r_o}{r_o+1}. \tag{5.24}$$

Proof The energy consumption for an ideal schedule is given by $E_o = T_o(\sum_{i=1}^{r_o} \mu(c_i) + \Gamma)$.

Since we do not know which machines are working for time more than T_o and which ones less than T_o, for the upper bound we assume that all machines are working for T^*_{max}. Hence, $E^*_{max} = T^*_{max}(\sum_{i=1}^{r_o} \mu(c_i) + \Gamma)$.

$$\frac{E^*_{max}}{E_o} = \frac{T^*_{max}\left(\sum_{i=1}^{r_o} \mu(c_i) + \Gamma\right)}{T_o\left(\sum_{i=1}^{r_o} \mu(c_i) + \Gamma\right)} \tag{5.25}$$

$$\frac{E^*_{max}}{E_o} = \frac{T^*_{max}}{T_o}. \tag{5.26}$$

By (5.23)

$$\frac{T^*_{max}}{T_o} \leq \frac{2r_o}{r_o+1}. \tag{5.27}$$

From (5.26) and (5.27), we get

$$\frac{E^*_{max}}{E_o} \leq \frac{2r_o}{r_o+1}. \tag{5.28}$$

□

This bound increases with number of machines in system and reaches 2 asymptotically. Kovács [46] gives a tighter bound for the LPT makespan of $1 + \sqrt{3}/3 \approx 1.5773$. As the bound on energy depends on makespan in our algorithm, the worst-case bound for our algorithm is also $1 + \sqrt{3}/3 \approx 1.5773$.

6. Energy efficiency: incompatible measures

There can be two measures for assessing the energy efficiency of a system of machines: one, to consider the total energy consumed by the system to complete some work; and the second, to consider the fraction of the energy consumed by machines in the system to do work over the total energy consumed in the system.

The first measure gives a sense of the energy required by the system per unit work produced, and the second measure indicates how much of the energy consumed by the system

actually goes into work and how much is idle (non-working) consumption by the system. We hold that the first measure, energy per unit work, is the more meaningful one (as it can lead to lowered overall energy consumption while completing some amount of work).

Our analysis clearly indicates that these two measures of energy efficiency are incompatible, in the sense that they cannot be simultaneously optimised for arbitrary systems. To see why, we may consider (4.28) for the total energy of the system. In (4.28), if we put $\gamma(c_i) = 0\ \forall i, 1 \leq i \leq m$, then the working energy is given by

$$\sum_{i=1}^{r} \mu(c_i) \left[\frac{W}{\sum_{i=1}^{r} \upsilon(c_i)} \right]. \tag{6.1}$$

Dividing (6.1) by (4.28), we get the following for the ratio of the working energy to total energy:

$$\frac{\sum_{i=1}^{r} \mu(c_i)}{\sum_{i=1}^{r} (\mu(c_i) - \gamma(c_i)) + \Gamma}. \tag{6.2}$$

This in turn simplifies to

$$\frac{\sum_{i=1}^{r} \mu(c_i)}{\sum_{i=1}^{r} \mu(c_i) + \sum_{i=r+1}^{m} \gamma(c_i)}. \tag{6.3}$$

Considering (6.3) shows that the ratio of the working energy to total energy can be optimised only by increasing the value of r, i.e., when $r = m$, but this violates Theorem 4.2, which tells us that for minimum total energy consumption, the value of r may not be necessarily be equal to m.

Thus a system running in such a way as to consume the least possible energy while completing some work will not always be most efficient in terms of the second measure, as the energy consumption by the idle machines can be significant. Conversely, if we seek to optimise the system performance by the second measure and reduce the idle consumption of machines, the overall energy consumption to do the work is not always minimised.

The data center energy efficiency metric called Power Usage Effectiveness (PUE) [47], an industry standard recommended by the U.S. Environmental Protection Agency under its Energy Star program,[1] is an energy efficiency measure of the second kind, as it considers only the ratio of the total energy to the working energy. It is thus not a surprise that the PUE comes with its share of controversy and criticisms [48].

References

[1] Pinedo M L 2012 *Scheduling: theory, algorithms, and systems*. Springer

[2] Pragati Agrawal and Shrisha Rao 2014 Energy-aware scheduling of distributed systems. *IEEE Transactions on Automation Science and Engineering* 11(4): 1163–1175

[3] Pinedo M L 2009 *Planning and scheduling in manufacturing and services*, 2nd ed. Springer

[4] Herrmann J W (Ed.) (2006) *Handbook of production scheduling, International Series in Operations Research & Management Science*, vol. 89. Springer, Berlin

[5] Jacek Blazewicz, Klaus Ecker, Erwin Pesch, Günter Schmidt, and Weglarz J 2019 *Handbook on scheduling*. Springer

[6] Yun W and Manas S 1999 Scheduling fixed-priority tasks with preemption threshold. In: *Proceedings of the Sixth International Conference on Real-Time Computing Systems and Applications (RTCSA '99)*. Hong Kong, China, pp. 328–335

[7] Sonia Y, Rachid C, Hubert K, and Bertrand G 2013 Multi-objective approach for energy-aware workflow scheduling in cloud computing environments. *Sci. World J.*.https://doi.org/10.1155/2013/350934

[8] Jia H, Christian B, Andreas R and Alois K 2011 Energy-aware task allocation for network-on-chip based heterogeneous multiprocessor systems. In: *Proceedings of the 19th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP 2011)*, Cyprus, pp. 447–454

[9] Hafiz Fahad Sheikh, Hengxing Tan, Ishfaq Ahmad, Sanjay Ranka, and Phanisekhar B V 2012 Energy- and performance-aware scheduling of tasks on parallel and distributed systems. *Journal on Emerging Technologies in Computing Systems* 8(4): 32:1–32:37

[10] Jiming Yao, Zhifeng Li, Yintao Li, Jie Bai, Jue Wang, and Peng Lin 2019 Cost-efficient tasks scheduling for smart grid communication network with edge computing system. In: *Proceedings of the 2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, IEEE, pp. 272–277

[11] Yuping Fan, Zhiling Lan, Paul Rich, Allcock W E, Papka M E, Brian Austin, and David Paul 2019 Scheduling beyond CPUs for HPC. In: *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing*, pp. 97–108

[12] Johan Pouwelse, Koen Langendoen, and Henk Sips 2001 Energy priority scheduling for variable voltage processors. In: *Proceedings of the International Symposium on Low-Power Electronics and Design*, pp. 28–33

[13] Mario Bambagini, Marko Bertogna, Mauro Marinoni, and Giorgio Buttazzo 2013 An energy-aware algorithm exploiting limited preemptive scheduling under fixed priorities. In: *Proceedings of the Eighth IEEE International Symposium on Industrial Embedded Systems (SIES 2013)*, pp. 3–12

[14] Zomaya A Y and Young Choon Lee (Eds.) 2012 *Energy-efficient distributed computing systems*. Wiley–IEEE Computer Society

[15] Jean-Marc Pierson (Ed.) Large-scale distributed systems and energy efficiency. In: *Wiley Series on Parallel and Distributed Computing*. John Wiley and Sons

[16] Sandy Irani and Pruhs K R 2005 Algorithmic problems in power management. *ACM Sigact News* 36(2): 63–76

[17] Susanne Albers 2009 Algorithms for energy saving. In: *Efficient Algorithms*. Springer, pp. 173–186

[1] See https://www.energystar.gov/ia/partners/prod_development/downloads/DataCenterRating_General.pdf.

[18] Sandy Irani, Sandeep Shukla, and Rajesh Gupta 2007 Algorithms for power savings. *ACM Transactions on Algorithms* 3(4): 41

[19] John Augustine, Sandy Irani, and Chaitanya Swamy 2004 Optimal power-down strategies. In: *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, IEEE, pp. 530–539

[20] Nikhil Bansal, Ho-Leung Chan, and Kirk Pruhs 2009 Speed scaling with an arbitrary power function. In: *Proceedings of the Twentieth annual ACM–SIAM Symposium on discrete algorithms*, Society for Industrial and Applied Mathematics, pp. 693–701

[21] Nikhil Bansal, Ho-Leung Chan, and Kirk Pruhs 2013 Speed scaling with an arbitrary power function. *ACM Transactions on Algorithms* 9(2): 18

[22] Nikhil Bansal, Tracy Kimbrel, and Kirk Pruhs 2007 Speed scaling to manage energy and temperature. *Journal of the ACM* 54(1): 3

[23] Frances Yao, Alan Demers, and Scott Shenker 1995 A scheduling model for reduced CPU energy. In: *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, IEEE, pp. 374–382

[24] Arunarani A R, Manjula D, and Vijayan Sugumaran 2019 Task scheduling techniques in cloud computing: a literature survey. *Future Generation Computer Systems* 91: 407–415

[25] Fredy Juarez, Jorge Ejarque, and Badia R M 2018 Dynamic energy-aware scheduling for parallel task-based application in cloud computing. *Future Generation Computer Systems* 78: 257–271

[26] Lin Gu, Jingjing Cai, Deze Zeng, Yu Zhang, Hai Jin, and Weiqi Dai 2019 Energy efficient task allocation and energy scheduling in green energy powered edge computing. *Future Generation Computer Systems* 95: 89–99

[27] Bharadwaj Veeravalli, Debasish Ghose, and Robertazzi T G 2003 Divisible load theory: a new paradigm for load scheduling in distributed systems. *Cluster Computing* 6(1): 7–17

[28] Dantong Yu and Robertazzi T G 2003 Divisible load scheduling for grid computing. In: *Proceedings of the Fifteenth IASTED International Conference on Parallel and Distributed Computing and Systems*, vol. 1, pp. 1–6

[29] Ashish Kumar Singh and Sandeep Sahu 2014 Environment conscious public cloud scheduling algorithm with load balancing. *International Journal of Computer Applications* 87(13): 24–27

[30] Monir Abdullah and Mohamed Othman 2013 Cost-based multi-QoS job scheduling using divisible load theory in cloud computing. *Procedia Computer Science* 18: 928–935

[31] Robertazzi T G, Moges M A, and Dantong Yu 2005 Divisible load scheduling with multiple sources: closed form solutions. In: *Proceedings of the Conference on Information Sciences and Systems*, March 2005

[32] Haiyan Shi, Wanliang Wang, and Ngaiming Kwok 2012 Energy dependent divisible load theory for wireless sensor network workload allocation. *Mathematical Problems in Engineering* https://doi.org/10.1155/2012.235289

[33] Leung J Y T, Kangbok Lee, and Pinedo M L 2012 Bi-criteria scheduling with machine assignment costs. *International Journal of Production Economics* 139(1): 321–329

[34] Kangbok Lee, Leung J Y T, Zhao-hong Jia, Wenhua Li, Pinedo M L, and Lin B M T 2014 Fast approximation algorithms for bi-criteria scheduling with machine assignment costs. *European Journal of Operational Research* 238(1): 54–64

[35] Kangbok Lee, Leung J Y T, and Pinedo M L 2012 Coordination mechanisms for parallel machine scheduling. *European Journal of Operational Research* 220(2): 305–313

[36] Maciej Drozdowski, Jedrzej Marszałkowski, and Jakub Marszałkowski 2014 Energy trade-offs analysis using equal-energy maps. *Future Generation Computer Systems* 36: 311–321

[37] Feng Li, Warren Liao T, Wentong Cai and Lin Zhang 2020 Multitask scheduling in consideration of fuzzy uncertainty of multiple criteria in service-oriented manufacturing. *IEEE Transactions on Fuzzy Systems* 28(11): 2759–2771

[38] Etienne Le Sueur and Gernot Heiser 2010 Dynamic voltage and frequency scaling: the laws of diminishing returns. In: HotPower'10: Proceedings of the 2010 International Conference on Power Aware Computing and Systems, Vancouver, Canada

[39] Asfa Toor, Saif ul Islam, Nimra Sohail, Adnan Akhunzada, Jalil Boudjadar, Hasan Ali Khattak, Ikram Ud Din, and Rodrigues JJPC 2019 Energy and performance aware fog computing: a case of DVFS and green renewable energy. *Future Generation Computer Systems* 101: 1112–1121

[40] Stavrinides G L and Karatza H D 2019 An energy-efficient, QoS-aware and cost-effective scheduling approach for real-time workflow applications in cloud computing systems utilizing DVFS and approximate computations. *Future Generation Computer Systems* 96: 216–226

[41] David Halliday, Robert Resnick, and Jearl Walker 2010 *Fundamentals of physics extended*, vol. 1 Wiley

[42] Pragati Agrawal and Shrisha Rao 2015 Energy-minimal scheduling of divisible loads. In: *Proceedings of the Fourth International Workshop on Energy-Efficient Data Centres (E2DC 2015), co-located with ACM E-Energy 2015*

[43] Johnson D S 1973 Approximation algorithms for combinatorial problems. In: *Proceedings of the Fifth Annual ACM Symposium on Theory of Computing*, ACM, pp. 38–49

[44] Graham R L 1969 Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics* 17(2): 416–429

[45] Teofilo Gonzalez, Ibarra O H, and Sartaj Sahni 1977 Bounds for LPT schedules on uniform processors. *SIAM Journal on Computing* 6(1): 155–166

[46] Annamária Kovács 2010 New approximation bounds for LPT scheduling. *Algorithmica* 57(2): 413–433

[47] Victor Avelar, Dan Azevedo, and Alan French 2012 *PUE: a comprehensive examination of the metric*. White Paper #49, The Green Grid, October 2012

[48] Brady G A, Nikil Kapur, Summers J L, and Thompson H M 2013 A case study and critical assessment in calculating power usage effectiveness for a data centre. *Energy Conversion and Management* 76: 155–161