Online Algorithms for Non-preemptive Speed Scaling on Power-Heterogeneous Processors

Aeshah Alsughayyir and Thomas Erlebach(✉)

Department of Informatics, University of Leicester, Leicester, England
{ayya1,te17}@leicester.ac.uk

Abstract. In this paper we consider non-preemptive online scheduling of jobs with release times and deadlines on heterogeneous processors with speed scaling. The power needed by processor i to run at speed s is assumed to be s^{α_i}, where the exponent α_i is a constant that can be different for each processor. We require the jobs to have agreeable deadlines, i.e., jobs with later release times also have later deadlines. The aim is to minimize the energy used to complete all jobs by their deadlines. For the case where the densities of the jobs differ only within a factor of two and the same holds for their interval lengths, we present an algorithm with constant competitive ratio. For arbitrary densities and interval lengths, we achieve a competitive ratio that is poly-logarithmic in the ratio of maximum to minimum density and in the ratio of maximum to minimum interval length.

1 Introduction

Efficient use of energy is becoming increasingly important because of energy cost and the need for sustainable use of resources. Modern processors support DVFS (dynamic voltage and frequency scaling), or speed scaling, which means that the speed at which a processor runs can be adjusted dynamically. The rate at which energy is consumed by a processor is called the *power*. It can be represented by a function $f(s) = s^\alpha$, for some constant $\alpha > 1$, that maps the speed s to the rate of energy consumption. In applications in cloud computing where jobs need to be dispatched to servers in a data center, different servers may have different power functions. Motivated by this, we consider *heterogeneous* processors where the exponent α of the power function can be different for each processor. We are interested in non-preemptive scheduling because preemption is undesirable in many application settings, e.g., in high-performance computing applications where jobs require a huge amount of data to be placed in main memory.

We study non-preemptive online scheduling of jobs with release times and deadlines on heterogeneous processors with speed scaling. There are m processors $P_1, \ldots, P_m$. The power function of processor P_i, $1 \leq i \leq m$, is $f_i(s) = s^{\alpha_i}$

A. Alsughayyir—Partially supported by the Department of Computer Science of Taibah University in Medina.

T. Erlebach—Supported by a study leave granted by University of Leicester.

X. Gao et al. (Eds.): COCOA 2017, Part II, LNCS 10628, pp. 457–465, 2017.
https://doi.org/10.1007/978-3-319-71147-8_32

for some constant $\alpha_i > 1$. Without loss of generality, we assume $\alpha_1 \leq \cdots \leq \alpha_m$. There are n jobs $J_1, \ldots, J_n$. Each job J_j has a release time r_j, a deadline d_j, and work (size) w_j. The time period from r_j to d_j is called the *interval* of job J_j, and $d_j - r_j$ is called the *interval length*. The *density* of job J_j is $\delta_j = \frac{w_j}{d_j - r_j}$. Jobs arrive online at their release times. Jobs with the same release time arrive in arbitrary order. Each job must be scheduled non-preemptively on one of the m processors between its release time and deadline. The speed of each processor can be changed at any time, and a processor running at speed s performs s units of work per unit time. Our objective is to find a feasible schedule that minimises the total energy consumption of all m processors. The total energy consumption $E(P_i)$ of processor P_i is the integral, over the duration of the schedule, of the power function of its speed, i.e., $E(P_i) = \int_0^H f_i(s_i(t))dt$, where $s_i(t)$ is the speed of processor P_i at time t and H denotes the time when the schedule ends, i.e., when all jobs are completed. The objective value is the total energy cost, $\sum_{i=1}^{m} E(P_i)$. We refer to the scheduling problem with this objective as *minimum energy scheduling*.

We assume that the jobs have *agreeable* deadlines, i.e., a job with later release time also has a later deadline. Formally, if job J_i arrives before job J_j, then $d_i \leq d_j$ must hold. This assumption is realistic in many scenarios and helps to schedule the jobs assigned to a processor non-preemptively. Let the *density ratio* $D = \frac{\max \delta_j}{\min \delta_j}$ be the ratio between maximum and minimum job density, and let the *interval-length ratio* $T = \frac{\max (d_j - r_j)}{\min (d_j - r_j)}$ be the ratio between maximum and minimum interval length. In this paper, we present an online algorithm with ratio $O(\lceil \log T \rceil^{\alpha_m + 1} \lceil \log D \rceil^{\alpha_m + 1})$ for non-preemptive online scheduling of agreeable jobs on heterogeneous processors. As far as we are aware, this is the first result for non-preemptive online minimum energy scheduling on heterogeneous processors.

Previous Work. The problem of minimising the total energy consumption on a single processor using speed scaling was first posed by Weiser et al. [11], who studied different heuristics experimentally. The pioneering work by Yao et al. [12] analyzed algorithms for speed scaling on a single processor so as to minimize the total energy consumption. Each job is characterized by its release time, its deadline, and its work. It must be scheduled during the interval between its release time and its deadline, and preemption is allowed. They presented a polynomial-time optimal algorithm for the offline problem and two online algorithms, Optimal Available (OA) and AVerage Rate (AVR). They showed that the competitive ratio of AVR is at most $\alpha^\alpha 2^{\alpha-1}$. We will use a non-preemptive variation of AVR to schedule the jobs that are assigned to a processor.

Table 1 gives an overview of known results for minimum energy scheduling problems for jobs with release times and deadlines on both homogeneous (S) and heterogeneous (S^*) parallel processors, including our new results (in bold). The problems are identified using an adaptation of the standard three-field notation of Graham et al. [9]. Minimum energy scheduling problems have mostly been studied in the preemptive case where the execution of a job can be interrupted and resumed later on the same processor (no migration) or on an arbitrary

Table 1. Known and new (in bold) results for speed-scaling on parallel processors. S stands for homogeneous and S^* for heterogeneous processors

Type	Problem	Ratio
Online	$S \mid r_j, d_j, w_j = 1, pmtn, no\text{-}mig \mid E$	$\alpha^\alpha 2^{4\alpha}$ [3]
	$S \mid agreeable, pmtn, no\text{-}mig \mid E$	$\alpha^\alpha 2^{4\alpha}$ [3]
	$S \mid r_j, d_j, pmtn, no\text{-}mig \mid E$	$2^{4\alpha}((\log P)^\alpha + \alpha^\alpha 2^{\alpha-1})$ [7]
	$S \mid r_j, d_j, pmtn, no\text{-}mig \mid E$	$2(\frac{\alpha}{\alpha-1})^\alpha e^\alpha B_\alpha$ (randomized) [10]
	$S \mid r_j, d_j, pmtn, mig \mid E$	α^α [1]
	$S^* \mid r_j, d_j, pmtn, mig \mid E$	$(1+\epsilon)(\alpha^\alpha 2^{\alpha-1}+1)$ [2]
	$S^* \mid r_j, d_j - r_j = x, \delta_j = \delta \mid E$	$\mathbf{3^{\alpha_m+1}(\alpha_m^{\alpha_m} 2^{\alpha_m-1}+1)}$
	$S^* \mid agreeable \mid E$	$\mathbf{5^{\alpha_m+1} 2^{\alpha_m} (\alpha_m^{\alpha_m} 2^{\alpha_m-1}+1) \lceil \log D \rceil^{\alpha_m+1} \lceil \log T \rceil^{\alpha_m+1}}$
Offline	$S \mid agreeable, pmtn, no\text{-}mig \mid E$	$\alpha^\alpha 2^{4\alpha}$ [3]
	$S \mid agreeable \mid E$	$(2-\frac{1}{m})^{\alpha-1}$ [5]
	$S \mid r_j, d_j \mid E$	$(m^\alpha(\sqrt[m]{n}))^{\alpha-1}$ [5]
	$S \mid r_j, d_j \mid E$	B_α (randomized) [10]
	$S^* \mid r_j, d_j \mid E$	$\tilde{B}_\alpha((1+\epsilon)(1+\frac{w_{max}}{w_{min}}))^\alpha$ [6]

processor (if migration is allowed). The previously known results do not cover the online problem of non-preemptive speed-scaling on heterogeneous processors, which is the focus of this paper.

For *homogeneous parallel processors*, we refer to Table 1 for an overview of known upper bounds on approximation ratios and competitive ratios. For *heterogeneous parallel processors*, Albers et al. [2] study the online version of the problem with migration and propose a $((1+\epsilon)(\alpha^\alpha 2^{\alpha-1}+1))$-competitive algorithm called H-AVR. It aims to assign work in each time interval according to the AVR schedule, and for each interval it creates an offline $(1+\epsilon)$-approximate schedule based on maximum flow computations. Bampis et al. [6] tackle the offline non-preemptive version of the fully heterogeneous speed scaling problem, where the work of a job can be processor-dependent, and propose a $\tilde{B}_\alpha((1+\epsilon)(1+\frac{w_{max}}{w_{min}}))^\alpha$-approximation algorithm, where $\tilde{B}_\alpha$ is the generalised Bell number. We refer to the recent surveys by Bampis [4] and Gerards et al. [8] for further discussion of known results on scheduling algorithms for energy minimization.

Outline. We present the first online algorithms for the non-preemptive scheduling of jobs with agreeable deadlines on heterogeneous parallel processors. In Sect. 2, we observe that a variation of AVR can be used to schedule jobs with agreeable deadlines non-preemptively on a single processor. In Sect. 3, we first show that the non-preemptive speed scaling problem for heterogeneous processors can be solved optimally by a simple greedy algorithm if all jobs are identical (i.e., have the same release time, deadline, and work). From this we obtain a $5^{\alpha_m+1}2^{\alpha_m}(\alpha_m^{\alpha_m}2^{\alpha_m-1}+1)$-competitive algorithm for jobs with agreeable deadlines whose interval lengths and densities differ by a factor of at most 2. For jobs with equal interval lengths and equal densities, the competitive ratio improves to $3^{\alpha_m+1}(\alpha_m^{\alpha_m}2^{\alpha_m-1}+1)$. In Sect. 4, we extend the result to arbitrary jobs with agreeable deadlines and obtain a competitive ratio of

$5^{\alpha_m+1}2^{\alpha_m}(\alpha_m^{\alpha_m}2^{\alpha_m-1}+1)\lceil\log D\rceil^{\alpha_m+1}\lceil\log T\rceil^{\alpha_m+1}$. Our algorithm classifies the jobs based on density and interval length and allocates the jobs in each class to processors by selecting the processor with the smallest energy cost increase.

2 Non-preemptive AVR

Our algorithms decide for each job on which processor it should be run, and then each processor uses an adaptation of the AVR algorithm, which was proposed for online preemptive scheduling by Yao et al. [12], to schedule the allocated jobs. AVR works as follows. We call a job J_j *active* at time t if $r_j \leq t \leq d_j$. At any time t, AVR sets the speed of the processor to the sum of the densities of the active jobs. Conceptually, all active jobs are executed simultaneously, each at a speed equal to its density. On an actual processor, this is implemented using preemption, i.e., each of the active jobs runs repeatedly for a very short period of time and is then preempted to let the next active job execute. To get a non-preemptive schedule for jobs with agreeable deadlines, we modify AVR as follows to obtain NAVR (non-preemptive AVR): The speed of the processor at any time t is set in the same way as for AVR, i.e., it is equal to the sum of the densities of all active jobs (even if some of these jobs have completed already). However, instead of sharing the processor between all active jobs, the jobs are executed non-preemptively in the order in which they arrive, which is the same as earliest deadline first (EDF) order because we have agreeable deadlines. We remark that the idea of a transformation of AVR schedules into non-preemptive schedules for jobs with agreeable deadlines was already mentioned in [5] in the context of offline approximation algorithms.

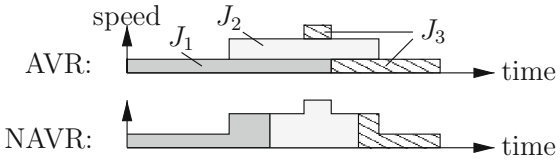

Fig. 1. AVR and NAVR schedules for an example with 3 jobs

An example comparing AVR and NAVR on an instance with 3 jobs is shown in Fig. 1. Each job is shown as a rectangle whose width is its interval length and whose height is its density. AVR shares the processor at each time among all active jobs. NAVR uses the same speed as AVR at any time, but dedicates the whole processor first to J_1, then to J_2, and finally to J_3.

Observation 1. *For scheduling jobs with agreeable deadlines on a single processor, the schedule produced by NAVR is non-preemptive and feasible. It has the same energy cost as the schedule produced by AVR.*

To analyze algorithms for minimum energy scheduling, we will compare the schedule produced by an algorithm with the optimal schedule that uses AVR (or equivalently NAVR for jobs with agreeable deadlines) on each processor and does not use migration. By applying Lemma 8 in [2] to NAVR instead of AVR, we get that, for instances with agreeable deadlines, there exists a schedule that uses NAVR on each processor and uses energy at most $(\max_{1\leq i\leq m}\{\rho_i\}+1)OPT$, where ρ_i is the competitive ratio of AVR on processor P_i. Let OPT^A denote the energy cost of the optimal NAVR schedule for a given instance of minimum energy scheduling with agreeable deadlines, and OPT the energy cost of an optimal schedule. As AVR is $\alpha^\alpha 2^{\alpha-1}$-competitive for a single processor with power function s^α [12], we get the following corollary:

Corollary 1. $OPT^A \leq (\alpha_m^{\alpha_m} 2^{\alpha_m - 1} + 1)OPT$.

3 Small Density Ratio and Interval-Length Ratio

Jobs with Equal Release Time, Deadline, and Density. First, consider the special case where all the jobs are identical, i.e., have the same release time, deadline, and density. We show that a simple greedy algorithm for allocating the jobs to processors, combined with NAVR on each processor, produces an optimal schedule. We need the following auxiliary result that shows that the extra power required by increasing the speed of a processor by δ grows with the current speed of the processor.

Lemma 1. *Let $\alpha > 1$, let x, y be real values satisfying $0 \leq x \leq y$, and let $\delta > 0$. Then $(x+\delta)^\alpha - x^\alpha \leq (y+\delta)^\alpha - y^\alpha$.*

For a given instance with identical jobs, we propose Algorithm EQ that assigns the jobs one by one as they arrive, always picking a processor that minimises the increase in power needed to accommodate the extra job.

Lemma 2. *Algorithm* EQ *produces an optimal schedule for identical jobs.*

Proof. Let r be the common release time, d the common deadline, and δ the common density of the jobs. Observe that if k jobs are assigned to a processor P_i, then the optimal schedule for these k jobs will be to run P_i at speed $k\delta$ from time r to time d and complete the jobs one by one in arbitrary order, with a total energy usage of $(d-r)(k\delta)^{\alpha_i}$ for P_i. For $1 \leq i \leq m$, let k_i be the number of jobs allocated to P_i by the algorithm, and let o_i be the number of jobs allocated to P_i by the optimal solution. Let ALG denote the total energy cost of Algorithm EQ, and OPT the total energy cost of the optimal schedule. We have $ALG = (d-r)\sum_{i=1}^m (k_i\delta)^{\alpha_i}$ and $OPT = (d-r)\sum_{i=1}^m (o_i\delta)^{\alpha_i}$.

Assume that $ALG > OPT$. Then there must be at least one P_i with $k_i > o_i$ and at least one P_h with $k_h < o_h$. Consider the last job, say job J_j, that the algorithm allocated to P_i. At the time the algorithm allocated J_j to P_i, the load of P_h was some $k'_h \leq k_h$. As the algorithm allocated J_j to P_i and not to P_h, we know that $(k'_h\delta+\delta)^{\alpha_h} - (k'_h\delta)^{\alpha_h} \geq (k_i\delta)^{\alpha_i} - (k_i\delta-\delta)^{\alpha_i}$. If we change

```
C ← 0 ;                                   /* current time period is [C, C + y/2) */
δ ← x ;                          /* treat all jobs as if their density was x */
while not all jobs allocated do
    for i ← 1 to m do
        L_i ← 0
    while next job J_j has r_j < C + y/2 do
        for i ← 1 to m do
            Z_i ← (L_i + δ)^{α_i} − L_i^{α_i};          /* power increase on P_i */
        i_min ← argmin_i Z_i ;                    /* smallest power increase */
        L_{i_min} ← L_{i_min} + δ ;          /* assign job J_j to processor P_{i_min} */
    C ← C + y/2
```

Algorithm 1. Jobs with interval length in $[y, 2y]$ and density in $[x, 2x]$

the optimal schedule by moving one job from P_h to P_i, the energy cost of that schedule increases by $d-r$ multiplied with $(o_i\delta+\delta)^{\alpha_i}-(o_i\delta)^{\alpha_i}-((o_h\delta)^{\alpha_h}-(o_h\delta-\delta)^{\alpha_h})$. By Lemma 1, we have $(o_i\delta+\delta)^{\alpha_i}-(o_i\delta)^{\alpha_i} \leq (k_i\delta)^{\alpha_i}-(k_i\delta-\delta)^{\alpha_i}$ and $(o_h\delta)^{\alpha_h}-(o_h\delta-\delta)^{\alpha_h} \geq (k'_h\delta+\delta)^{\alpha_h}-(k'_h\delta)^{\alpha_h}$. This implies $(o_i\delta+\delta)^{\alpha_i}-(o_i\delta)^{\alpha_i}-((o_h\delta)^{\alpha_h}-(o_h\delta-\delta)^{\alpha_h}) \leq 0$. As we started with the optimal schedule, the change in energy cannot be negative, so the new schedule must have the same energy cost and again be optimal. This operation can be repeated, without increasing the energy cost, until the optimal schedule and the schedule produced by the algorithm are identical. □

Interval Lengths and Densities within a Factor of Two. Assume that the interval lengths of all jobs are in $[y, 2y]$ and the densities of all jobs in $[x, 2x]$. The algorithm, shown as Algorithm 1, assigns each job to one of the m processors. It treats the jobs as if their density was equal to $\delta = x$ and proceeds in time periods of length $\frac{y}{2}$. Jobs arriving in a time period are handled independently of those arriving in other time periods. On each processor, the allocated jobs are scheduled using NAVR.

Algorithm 1 allocates the jobs arriving in the time period $[C, C + \frac{y}{2})$ to machines in the same way as Algorithm EQ would allocate them if they were identical jobs with density δ. Furthermore, all these jobs are active in the whole interval $[C + \frac{y}{2}, C + y)$ because their interval length is at least y.

Lemma 3. *Consider the allocation that Algorithm 1 produces for jobs arriving in the time period $[C, C + \frac{y}{2})$. Then the energy use for those jobs alone in the time period $[C + \frac{y}{2}, C + y)$ is at most 2^{α_m} times the optimal energy cost that any AVR schedule for the same jobs incurs in that period.*

Let ALG_C be the total energy cost of the algorithm in the time interval $[C, C + \frac{y}{2})$, and let OPT_C^A be the total energy cost of an optimal AVR schedule in the time interval $[C, C + \frac{y}{2})$. For the schedule of Algorithm 1, let A_C be the total energy cost incurred during the time period $[C, C + \frac{y}{2})$ for jobs that are released in the time interval $[C - \frac{y}{2}, C)$. Let $K_{C,i}$ be the set of jobs that are

released in $[C - \frac{y}{2}, C)$ and assigned to P_i by the algorithm. We have $A_C = \frac{y}{2}\sum_{i=1}^{m}(\sum_{J_j \in K_{C,i}} \delta_j)^{\alpha_i}$. By Lemma 3 we have that $A_C \leq 2^{\alpha_m} OPT_C^A$.

As all jobs have interval length in $[y, 2y]$, the jobs that are executed by the algorithm at some point in the time period $[C, C + \frac{y}{2})$ are released in one of the five intervals $[C - 2y, C - \frac{3y}{2})$, $[C - \frac{3y}{2}, C - y)$, $[C - y, C - \frac{y}{2})$, $[C - \frac{y}{2}, C)$, or $[C, C + \frac{y}{2})$. With $U_{C,i} = K_{C-\frac{3y}{2},i} \cup K_{C-y,i} \cup K_{C-\frac{y}{2},i} \cup K_{C,i} \cup K_{C+\frac{y}{2},i}$, the speed of the processor P_i in the interval $[C + \frac{y}{2}, C + y)$ is at most $\sum_{J_j \in U_{C,i}} \delta_j$. Therefore, we have $ALG_C \leq \frac{y}{2}\sum_{i=1}^{m}(\sum_{J_j \in U_{C,i}} \delta_j)^{\alpha_i} \leq \frac{y}{2}\sum_{i=1}^{m}(5\max\{\sum_{J_j \in K_{C-\frac{3y}{2},i}} \delta_j, \ldots, \sum_{J_j \in K_{C+\frac{y}{2},i}} \delta_j\})^{\alpha_i}$. This is at most $\frac{y}{2}5^{\alpha_m}\sum_{i=1}^{m}\Big(\max\{(\sum_{J_j \in K_{C-\frac{3y}{2},i}} \delta_j)^{\alpha_i}, \ldots, (\sum_{J_j \in K_{C+\frac{y}{2},i}} \delta_j)^{\alpha_i}\}\Big)$, which can be bounded by $5^{\alpha_m}(A_{C-\frac{3y}{2}} + A_{C-y} + A_{C-\frac{y}{2}} + A_C + A_{C+\frac{y}{2}})$. The total energy cost ALG of Algorithm 1 can then be bounded by $ALG = \sum_{C \geq 0} ALG_C \leq \sum_{C \geq 0} 5^{\alpha_m}(A_{C-\frac{3y}{2}} + A_{C-y} + A_{C-\frac{y}{2}} + A_C + A_{C+\frac{y}{2}}) \leq 5^{\alpha_m+1}\sum_{C \geq 0} A_C \leq 5^{\alpha_m+1}\sum_{C \geq 0} 2^{\alpha_m} OPT_C^A = 5^{\alpha_m+1}2^{\alpha_m} OPT^A \leq 5^{\alpha_m+1}2^{\alpha_m}(\alpha_m^{\alpha_m}2^{\alpha_m-1}+1)OPT$. Here, the third inequality follows from Lemma 3 and the last inequality holds by Corollary 1. Thus, we get the following theorem.

Theorem 1. *Algorithm 1 is* $5^{\alpha_m+1}2^{\alpha_m}(\alpha_m^{\alpha_m}2^{\alpha_m-1}+1)$*-competitive for jobs with agreeable deadlines and density ratio at most two and interval-length ratio at most two.*

For the special case where all jobs have the same interval length and the same density, the analysis can be improved, because the factor 2^{α_m} of Lemma 3 can be avoided and only jobs arriving in the three time periods $[C - y, C - \frac{y}{2})$, $[C - \frac{y}{2}, C)$ and $[C, C + \frac{y}{2})$ have intervals that overlap $[C, C + \frac{y}{2})$.

Corollary 2. *For minimum energy scheduling of jobs with equal interval lengths and equal densities, there is an online algorithm that achieves competitive ratio* $3^{\alpha_m+1}(\alpha_m^{\alpha_m}2^{\alpha_m-1} + 1)$.

4 Arbitrary Interval Lengths and Densities

We now consider jobs with arbitrary interval lengths and densities, only requiring that the jobs have agreeable deadlines. Recall that D denotes the density ratio and T the interval-length ratio. Let $\Delta = \max_j \delta_j$ denote the maximum job density, and let $\Lambda = \max_j(d_j - r_j)$ be the maximum interval length. For ease of presentation, we assume that the algorithm knows Δ and Λ, but it is not difficult to adapt the algorithm so that it can work without this assumption.

The interval lengths of all jobs are in $[\Lambda/T, \Lambda]$ and their densities are in $[\Delta/D, \Delta]$. We classify the jobs into groups such that within each group the interval lengths and densities vary only within a factor of two. Each group is scheduled independently of the others using a separate copy of Algorithm 1, but of course all the jobs run on the same set of processors. A job is classified into group $g_{t,d}$ if its interval length is in $[\Lambda/2^t, \Lambda/2^{t-1}]$ and its density in $[\Delta/2^d, \Delta/2^{d-1}]$,

where $t \in \{1, \ldots, \lceil \log T \rceil\}$ and $d \in \{1, \ldots, \lceil \log D \rceil\}$. Jobs that lie at group boundaries can be allocated to one of the two relevant groups arbitrarily. We refer to this algorithm as Algorithm CA.

Let $\ell(g_{t,d}, i, t')$ be the load (sum of densities of active jobs) of group $g_{t,d}$ on processor P_i at time t', and let $A_{g_{t,d}}$ be the total energy cost of Algorithm CA for group $g_{t,d}$, assuming that it is the only group running. Let H denote the time when the schedule ends, i.e., the deadline of the last job, and let $OPT(g_{t,d})$ denote the energy cost of the optimal schedule for $g_{t,d}$. From Theorem 1 we get $A_{g_{t,d}} \leq 5^{\alpha_m+1} 2^{\alpha_m} (\alpha_m^{\alpha_m} 2^{\alpha_m - 1} + 1) OPT(g_{t,d})$. We have $OPT(g_{t,d}) \leq OPT$ and thus $\sum_{t,d} OPT(g_{t,d}) \leq \lceil \log T \rceil \lceil \log D \rceil OPT$. Using that the total energy cost ALG of Algorithm CA is $ALG = \sum_{i=1}^{m} \int_0^H \left(\sum_{t,d} \ell(g_{t,d}, i, t') \right)^{\alpha_i} dt'$, we can complete the analysis and show the following theorem.

Theorem 2. *For non-preemptive minimum energy scheduling of jobs with agreeable deadlines on heterogeneous processors, the competitive ratio of Algorithm* CA *is at most* $5^{\alpha_m+1} 2^{\alpha_m} (\alpha_m^{\alpha_m} 2^{\alpha_m - 1} + 1) \lceil \log D \rceil^{\alpha_m+1} \lceil \log T \rceil^{\alpha_m+1}$.

References

1. Albers, S., Antoniadis, A., Greiner, G.: On multi-processor speed scaling with migration. J. Comput. Syst. Sci. **81**(7), 1194–1209 (2015). https://doi.org/10.1016/j.jcss.2015.03.001
2. Albers, S., Bampis, E., Letsios, D., Lucarelli, G., Stotz, R.: Scheduling on power-heterogeneous processors. In: Kranakis, E., Navarro, G., Chávez, E. (eds.) LATIN 2016. LNCS, vol. 9644, pp. 41–54. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49529-2_4
3. Albers, S., Müller, F., Schmelzer, S.: Speed scaling on parallel processors. Algorithmica **68**(2), 404–425 (2014). https://doi.org/10.1007/s00453-012-9678-7
4. Bampis, E.: Algorithmic issues in energy-efficient computation. In: Kochetov, Y., Khachay, M., Beresnev, V., Nurminski, E., Pardalos, P. (eds.) DOOR 2016. LNCS, vol. 9869, pp. 3–14. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-44914-2_1
5. Bampis, E., Kononov, A.V., Letsios, D., Lucarelli, G., Nemparis, I.: From preemptive to non-preemptive speed-scaling scheduling. Discrete Appl. Math. **181**, 11–20 (2015). https://doi.org/10.1016/j.dam.2014.10.007
6. Bampis, E., Letsios, D., Lucarelli, G.: Speed-scaling with no preemptions. In: Ahn, H.-K., Shin, C.-S. (eds.) ISAAC 2014. LNCS, vol. 8889, pp. 259–269. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-13075-0_21
7. Bell, P.C., Wong, P.W.H.: Multiprocessor speed scaling for jobs with arbitrary sizes and deadlines. J. Comb. Optim. **29**(4), 739–749 (2015). https://doi.org/10.1007/s10878-013-9618-8
8. Gerards, M.E.T., Hurink, J.L., Hölzenspies, P.K.F.: A survey of offline algorithms for energy minimization under deadline constraints. J. Sched. **19**(1), 3–19 (2016). https://doi.org/10.1007/s10951-015-0463-8
9. Graham, R.L., Lawler, E.L., Lenstra, J.K., Kan, A.R.: Optimization and approximation in deterministic sequencing and scheduling: a survey. Ann. Discrete Math. **5**, 287–326 (1979)

10. Greiner, G., Nonner, T., Souza, A.: The bell is ringing in speed-scaled multiprocessor scheduling. Theor. Comput. Syst. **54**(1), 24–44 (2014). https://doi.org/10.1007/s00224-013-9477-9
11. Weiser, M., Welch, B.B., Demers, A.J., Shenker, S.: Scheduling for reduced CPU energy. In: Proceedings of the First USENIX Symposium on Operating Systems Design and Implementation (OSDI 1994), pp. 13–23. USENIX Association (1994)
12. Yao, F., Demers, A., Shenker, S.: A scheduling model for reduced CPU energy. In: 36th Annual Symposium on Foundations of Computer Science (FOCS 1995), pp. 374–382. IEEE Computer Society (1995). https://doi.org/10.1109/SFCS.1995.492493