Energy Efficient Scheduling and Routing via Randomized Rounding

Evripidis Bampis[*1], Alexander Kononov[†2], Dimitrios Letsios[*1,3], Giorgio Lucarelli[*1,3], and Maxim Sviridenko[‡4]

1 **LIP6, Université Pierre et Marie Curie, France**
{Evripidis.Bampis,Giorgio.Lucarelli}@lip6.fr
2 **Sobolev Institute of Mathematics, Novosibirsk, Russia**
alvenko@math.nsc.ru
3 **IBISC, Université d' Évry, France**
dimitris.letsios@ibisc.univ-evry.fr
4 **Department of Computer Science, University of Warwick, UK**
M.I.Sviridenko@warwick.ac.uk

Abstract

We propose a unifying framework based on configuration linear programs and randomized rounding, for different energy optimization problems in the dynamic speed-scaling setting. We apply our framework to various scheduling and routing problems in heterogeneous computing and networking environments. We first consider the energy minimization problem of scheduling a set of jobs on a set of parallel speed-scalable processors in a fully heterogeneous setting. For both the preemptive-non-migratory and the preemptive-migratory variants, our approach allows us to obtain solutions of almost the same quality as for the homogeneous environment. By exploiting the result for the preemptive-non-migratory variant, we are able to improve the best known approximation ratio for the single processor non-preemptive problem. Furthermore, we show that our approach allows to obtain a constant-factor approximation algorithm for the power-aware preemptive job shop scheduling problem. Finally, we consider the min-power routing problem where we are given a network modeled by an undirected graph and a set of uniform demands that have to be routed on integral routes from their sources to their destinations so that the energy consumption is minimized. We improve the best known approximation ratio for this problem.

1998 ACM Subject Classification F.2.2 [Nonnumerical Algorithms and Problems]: Sequencing and scheduling

Keywords and phrases randomized rounding, scheduling; approximation, energy-aware; configuration linear program

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2013.449

1 Introduction

We focus on energy minimization problems in heterogeneous computing and networking environments in the dynamic speed-scaling setting. For many years, the exponential increase of processors' frequencies followed Moore's law. This is no more possible because of physical

* Supported by the French Agency for Research under the DEFIS program TODO, ANR-09-EMER-010, and by the project ALGONOW under the program THALES.
† Supported by the RFBR grant No 12-01-00184.
‡ Supported by EPSRC grants EP/J021814/1, EP/D063191/1, FP7 Marie Curie Career Integration Grant and Royal Society Wolfson Research Merit Award.

33rd Int'l Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2013).
Editors: Anil Seth and Nisheeth K. Vishnoi; pp. 449–460

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

(thermal) constraints. Today, for improving the performance of modern computing systems, designers use parallelism, i.e., multiple cores running at lower frequencies but offering better performances than a single core. These systems can be either homogeneous where an identical core is used many times, or heterogeneous combining general-purpose and special-purpose cores. Heterogeneity offers the possibility of further improving the performance of the system by executing each job on the most appropriate type of processors [9]. However in order to exploit the opportunities offered by the heterogeneous systems, it is essential to focus on the design of new efficient power-aware algorithms taking into account the heterogeneity of these architectures. In this direction, Gupta et al. in [12] have studied the impact of the heterogeneity on the difficulty of various power-aware scheduling problems.

In this paper, we show that rounding configuration linear programs helps in handling the heterogeneity of both the jobs and the processors. We adopt one of the main mechanisms for reducing the energy consumption in modern computer systems which is based on the use of speed scalable processors. Starting from the seminal paper of Yao et al. [15], many papers adopted the speed-scaling model in which if a processor runs at speed s, then the rate of the energy consumption, i.e., the power, is $P(s) = s^{\alpha}$ with α a constant close to 3 (new studies show that α is rather smaller: 1.11 for Intel PXA 270, 1.62 for Pentium M770 and 1.66 for a TCP offload engine [14]). Moreover, the energy consumption is the integral of the power over time. This model captures the intuitive idea that the faster a processor works the more energy it consumes.

We first consider a *fully heterogeneous environment* where both, the jobs' characteristics are processor-dependent and every processor has its own power function. Formally, we consider the following problem: we are given a set $\mathcal{J}$ of n jobs and a set $\mathcal{P}$ of m parallel processors. Every processor $i \in \mathcal{P}$ obeys to a different speed-to-power function, i.e., it is associated with a different $\alpha_i \geq 1$ and hence if a job runs at speed s on processor i, then the power is $P(s) = s^{\alpha_i}$. Each job $j \in \mathcal{J}$ has a different release date $r_{i,j}$, deadline $d_{i,j}$ and workload $w_{i,j}$ if job j is executed on processor $i \in \mathcal{P}$. The goal is to find a schedule of minimum energy respecting the release dates and the deadlines of the jobs.

In this paper we propose a unifying framework for minimizing energy in different heterogeneous computing and networking environments. We first consider two variants of the heterogeneous multiprocessor *preemptive* problem. In both cases, the execution of a job may be interrupted and resumed later. In the *non-migratory* case each job has to be entirely executed on a single processor. In the *migratory* case each job may be executed by more than one processors, without allowing parallel execution of a job. We also focus on the *non-preemptive* single processor case. Furthermore, we consider the energy minimization problem in an heterogeneous *job shop* environment where the jobs can be preempted. Finally, we consider the *min-power routing problem*, introduced in [4], where a set of uniform demands have to be routed on integral routes from their sources to their destinations so that the energy consumption to be minimized. We believe that our general techniques will find further applications in energy optimization.

1.1 Related Work

Yao et al. [15] proposed an optimal algorithm for finding a feasible preemptive schedule with minimum energy consumption when a single processor is available. The homogeneous multiprocessor case has been solved optimally in polynomial time when both the preemption and the migration of jobs are allowed [2, 5, 7, 8]. Albers et al. [3] considered the homogeneous multiprocessor preemptive problem, where the migration of the jobs is not allowed. They proved that the problem is $\mathcal{NP}$-hard even for instances with common release dates and

common deadlines. Greiner et al. [10] gave a generic reduction transforming an optimal schedule for the homogeneous multiprocessor problem with migration, to a $B_{\lceil\alpha\rceil}$-approximate solution for the homogeneous multiprocessor preemptive problem without migration, where $B_{\lceil\alpha\rceil}$ is the $\lceil\alpha\rceil$-th Bell number. Antoniadis and Huang [6] proved that the single processor non-preemptive problem is $\mathcal{NP}$-hard even for instances in which for any two jobs j and j' with $r_j \leq r_{j'}$ it holds that $d_j \geq d_{j'}$. They also proposed a $2^{5\alpha-4}$-approximation algorithm for general instances. Andrews et al. [4] studied the min-power routing problem and for uniform demands, i.e. for the case where all the demands have the same value, they proposed a γ-approximation algorithm, where $\gamma = \max\{1+\tau 2^{\alpha(\tau+1)\log e}, 2+\tau 2^{\alpha(\tau+1)}\}$, with $\tau = \lceil 2\log(\alpha+4)\rceil$. For non-uniform demands, they proposed a $O(\log^{\alpha-1} D)$-approximation algorithm, where D is the maximum value of the demands. For further results see [1].

1.2 Notation

Given a schedule $\mathcal{S}$ we denote by $E(\mathcal{S})$ the total energy consumed by $\mathcal{S}$. We denote by $\mathcal{S}^*$ an optimal schedule and by OPT the energy consumption of $\mathcal{S}^*$. For each job $j \in \mathcal{J}$, we say that j is *alive* on processor $i \in \mathcal{P}$ during the interval $[r_{i,j}, d_{i,j}]$. Let $\alpha = \max_{i\in\mathcal{P}}\{\alpha_i\}$. The Bell number, B_n, is defined for any integer $n \geq 0$ and corresponds to the number of partitions of a set of n items. It is well known that Bell numbers satisfy the following equality

$$B_n = \sum_{k=0}^{\infty} \frac{k^n e^{-1}}{k!}$$

known as Dobinski's formula. Another way to state this formula is that n-th Bell number is equal to the n-th moment of Poisson random variable with parameter (expected value) 1. That naturally leads to a more general definition. The generalized Bell number, denoted by $\tilde{B}_\alpha = \sum_{k=0}^{\infty} \frac{k^\alpha e^{-1}}{k!}$, is defined for any $\alpha \in \mathbb{R}^+$ and corresponds to the α-th (fractional) moment of Poisson random variable with parameter 1. Note that the ratios of our algorithms depend on the generalized Bell number. Due to space constraints, some results and proofs will be given in the full version of the paper.

1.3 Our Contribution

In this paper we formulate heterogeneous scheduling and routing problems using *configuration linear programs (LPs)* and we apply *randomized rounding*. In Section 3, we consider the heterogeneous multiprocessor speed-scaling problem without migrations and we propose an approximation algorithm of ratio $(1+\varepsilon)\tilde{B}_\alpha$. As this LP has an exponential number of variables, we give an alternative (compact) formulation of the problem using a polynomial number of variables and we prove the equivalence between the two LP relaxations. For real values of α our result improves the $B_{\lceil\alpha\rceil}$ approximation ratio of [10] for the homogeneous case to $(1+\varepsilon)\tilde{B}_\alpha$ for the fully heterogeneous environment that we consider here (see Table 1). In Section 4, using again a configuration LP formulation, we present an algorithm for the heterogeneous multiprocessor speed-scaling problem with migration. This algorithm returns a solution within an additive factor of ε far from the optimal solution and runs in time polynomial to the size of the instance and to $1/\varepsilon$. This result generalizes the results of [2, 5, 7, 8] from an homogeneous environment to a fully heterogeneous environment. In Section 5, we transform the single processor speed-scaling problem without preemptions to the heterogeneous multiprocessor problem without migrations and we give an approximation algorithm of ratio $2^{\alpha-1}(1+\varepsilon)\tilde{B}_\alpha$, improving upon the previous known $2^{5\alpha-4}$-approximation algorithm in [6] for any $\alpha < 114$ (see Table 1). In Section 6, we study the power-aware

Table 1 Comparison of our approximation ratios vs. better previous known ratios for: (i) the preemptive multiprocessor problem without migrations, (ii) the single processor non-preemptive problem, and (iii) the min-power routing problem.

Value of α	Preemptive without migrations		Non-preemptive single processor		Routing uniform demands	
	Homogeneous [10]	Heterogeneous This paper	[6]	This paper	[4]	This paper
1.11	2	$1.07(1+\varepsilon)$	2.93	$1.15(1+\varepsilon)$	375	1.07
1.62	2	$1.49(1+\varepsilon)$	17.15	$2.30(1+\varepsilon)$	2196	1.49
1.66	2	$1.54(1+\varepsilon)$	19.70	$2.43(1+\varepsilon)$	2522	1.54
2	2	$2(1+\varepsilon)$	64	$4(1+\varepsilon)$	8193	2
2.5	5	$3.08(1+\varepsilon)$	362	$8.72(1+\varepsilon)$	46342	3.08
3	5	$5(1+\varepsilon)$	2048	$20(1+\varepsilon)$	262145	5

preemptive job shop scheduling problem and we propose a $((1+\varepsilon)\tilde{B}_\alpha)$-approximation algorithm, where μ is the number of all the operations. Finally, in Section 7, we improve the analysis for the min-power routing problem with uniform demands given in [4], based on the randomized rounding analysis that we propose in this paper. Our approach gives an approximation ratio of $\tilde{B}_\alpha$ significantly improving the analysis given in [4] (see Table 1).

2 Technical Probabilistic Propositions

▶ **Proposition 1.** *Consider a set of real numbers $\{Y_1, Y_2, \ldots, Y_n\}$ such that $Y_i \in [0,1]$ for all $i \in \{1,\ldots,n\}$ and a set of non-negative constants $\{e_1, e_2, \ldots, e_n\}$. Assume that we split Y_n to $Y'_n \geq 0$ and $Y'_{n+1} \geq 0$ such that $Y_n = Y'_n + Y'_{n+1}$. Let $e_{n+1} = e_n$ and $Y'_j = Y_j$, $j \in \{1,2,\ldots,n-1\}$. It holds that*

$$\sum_{S\subseteq\{1,2,\ldots,n\}} |S|^{\alpha-1}\left(\sum_{j\in S} e_j\right)\prod_{j\in S} Y_j \prod_{j\notin S}(1-Y_j) \leq \sum_{S\subseteq\{1,2,\ldots,n+1\}} |S|^{\alpha-1}\left(\sum_{j\in S} e_j\right)\prod_{j\in S} Y'_j \prod_{j\notin S}(1-Y'_j)$$

▶ **Proposition 2.** *For any $\alpha \geq 1$, the function $f(x) = x^\alpha$ and parameter $a \in [0,1]$ we have*

$$\mathbb{E}[f(B_a)] \leq \mathbb{E}[f(P_a)]$$

where B_a is a sum of n independent Bernoulli random variables, $\mathbb{E}[B_a] = a$ and P_a is a Poisson random variable with parameter a.

▶ **Proposition 3.** *For any real $\alpha \geq 1$ and a Poisson random variable P_λ with parameter $\lambda \geq 0$, we have:*
(a) *If $0 \leq \lambda \leq 1$, then $\mathbb{E}[P_\lambda^\alpha] \leq \lambda \mathbb{E}[P_1^\alpha]$.*
(b) *If $\lambda > 1$, then $\mathbb{E}[P_\lambda^\alpha] \leq \lambda^\alpha \mathbb{E}[P_1^\alpha]$.*

3 Heterogeneous Multiprocessor without Migrations

In this section we consider the case where the migration of jobs is not permitted, but their preemption is allowed. The corresponding homogeneous problem is known to be $\mathcal{NP}$-hard even if all jobs have common release dates and deadlines [3]. We propose an approximation algorithm by formulating the problem as a configuration integer program (IP) with an

exponential number of variables and a polynomial number of constraints. Given an optimal solution for the configuration LP relaxation, we apply randomized rounding to get a feasible schedule for our problem. In order to get a polynomial-time algorithm, we can give another (compact) formulation of our problem with a polynomial number of variables and constraints and we can show that the relaxations of the two formulations are equivalent.

In order to formulate our problem as a configuration IP we need to discretize the time. In the following lemma we assume that the release dates and the deadlines of all jobs in all processors are integers.

▶ **Lemma 4.** *There is a feasible schedule with energy consumption at most* $((1+\frac{\varepsilon}{1-\varepsilon})(1+\frac{2}{n-2}))^{\alpha} \cdot OPT$ *in which each piece of each job* $j \in \mathcal{J}$ *(*j *is executed on processor* $i \in \mathcal{P}$*) starts and ends at a time point* $r_{i,j} + k\frac{\varepsilon}{n^3}(d_{i,j} - r_{i,j})$*, where* $k \geq 0$ *is an integer.*

Let $\mathcal{S}$ be a schedule that satisfies Lemma 4 and let $j \in \mathcal{J}$ be a job executed on the processor $i \in \mathcal{P}$ in $\mathcal{S}$. The above lemma implies that the interval $(r_{i,j}, d_{i,j}]$ can be partitioned into polynomial, with respect to n and $1/\varepsilon$, number of equal length slots. In each of these slots, j either is executed during the whole slot or is not executed at all. In what follows we consider schedules that satisfy Lemma 4.

3.1 Linear Programming Relaxation

A configuration c is a schedule for a single job on a single processor. Specifically, a configuration determines the slots, with respect to Lemma 4, during which one job is executed. Given a configuration c for a job $j \in \mathcal{J}$, we can define the execution time of j that is equal to the number of slots in c multiplied by the length of the slot. Due to the convexity of the speed-to-power function, in a minimum energy schedule that satisfies Lemma 4, the job j runs at a constant speed s_j. Hence, s_j is equal to the work of j over its execution time. Let $\mathcal{C}_{ij}$ be the set of all possible feasible configurations for all jobs in all processors.

In order to ensure the feasibility of our schedule we need to further partition the time, by merging the slots for all jobs. Given a processor $i \in \mathcal{P}$, consider the time points of all jobs of the form $r_{i,j} + k\frac{\varepsilon}{n^3}(d_{i,j} - r_{i,j})$ as introduced in Lemma 4. Let $t_{i,1}, t_{i,2}, \ldots, t_{i,\ell_i}$ be the ordered sequence of these time points. Consider now the intervals $(t_{i,p}, t_{i,p+1}]$, $1 \leq p \leq \ell_i - 1$. In a schedule that satisfies Lemma 4, in each such interval either there is exactly one job that is executed during the whole interval or the interval is idle. Note also that these intervals might not have the same length. Let $\mathcal{I}$ be the set of all these intervals for all processors.

We introduce the binary variable $x_{i,j,c}$ that is equal to one if the job $j \in \mathcal{J}$ is entirely executed on the processor $i \in \mathcal{P}$ according to the configuration c, and zero otherwise. Note that, given the configuration and the processor i where the job j is executed, we can compute the energy consumption $E_{i,j,c}$ for the execution of j. For ease of notation, we say $I \in (i,j,c)$ if the interval $I \in \mathcal{I}$ is included in the configuration c of processor $i \in \mathcal{P}$ for the job $j \in \mathcal{J}$, that is there is a slot $(r_{i,j} + k\frac{\varepsilon}{n^3}(d_{i,j} - r_{i,j}), r_{i,j} + (k+1)\frac{\varepsilon}{n^3}(d_{i,j} - r_{i,j})]$ in c that contains I.

$$\min \sum_{i,j,c} E_{i,j,c} \cdot x_{i,j,c}$$

$$\sum_{i,c} x_{i,j,c} \geq 1 \qquad \forall j \in \mathcal{J} \tag{1}$$

$$\sum_{(i,j,c): I \in (i,j,c)} x_{i,j,c} \leq 1 \qquad \forall I \in \mathcal{I} \tag{2}$$

$$x_{i,j,c} \in \{0,1\} \qquad \forall i \in \mathcal{P}, j \in \mathcal{J}, c \in \mathcal{C}_{ij} \tag{3}$$

Inequality (1) enforces that each job is entirely executed according to exactly one configuration. Inequality (2) ensures that at most one job is executed in each interval $(t_{i,p}, t_{i,p+1}]$, $1 \leq p \leq \ell_i - 1$. We next relax the constraints (3) such that $x_{i,j,c} \geq 0$. Since the structure of this LP is quite simple we can define an equivalent compact LP relaxation with polynomial number of constraints and variables. We describe how to do it in the full version of the paper. For now we assume that we can find an optimal solution of our configuration LP in polynomial time.

3.2 Randomized Rounding

In this section, we show how to apply randomized rounding to get an approximation algorithm for our problem. Our algorithm follows.

Algorithm 1

1: Solve the configuration LP relaxation.
2: For each job $j \in \mathcal{J}$, choose a configuration at random with probability $x_{i,j,c}$.
3: Let K_I be the number of configurations that contain the interval I. Scale the speeds during I by a factor of K_I.

▶ **Theorem 5.** *Assume that $\alpha_i \geq 1$ for all $i = 1, \ldots, m$. Algorithm 1 achieves an approximation ratio of $((1 + \frac{\varepsilon}{1-\varepsilon})(1 + \frac{2}{n-2}))^{\alpha} \tilde{B}_{\alpha}$ for the heterogeneous multiprocessor preemptive speed-scaling problem without migrations in time polynomial to n and to $1/\varepsilon$, where $\alpha = \max_{i \in \mathcal{P}} \alpha_i$.*

Proof. For each interval $I \in \mathcal{I}$, we estimate its expected energy consumption. By definition, an interval corresponds to a single processor $i \in \mathcal{P}$. Given a job $j \in \mathcal{J}$, let n_j be the number of the non-zero $x_{i,j,c}$ variables such that I belongs to configuration c. Moreover, let $X_{j,k}$ be the k-th, $1 \leq k \leq n_j$, of the above non-zero variables and $s_{j,k}$ be the corresponding speed.

Let Y_j be the probability that the job j is assigned to be processed in the interval I by the randomized rounding procedure, that is $Y_j = \sum_{k=1}^{n_j} X_{j,k}$. By the constraint (2), we know that $\sum_{j=1}^{n} Y_j \leq 1$. The expected energy that the job j consumes on the interval I under the condition that j is assigned to be processed in the interval I without considering the other jobs is $e_j = \frac{\sum_{k=1}^{n_j} |I| s_{j,k}^{\alpha_i} X_{j,k}}{Y_j}$.

The energy consumption in the interval I achieved by the optimal solution of the LP relaxation is $LP^* = \sum_{j=1}^{n} e_j Y_j$. If the randomized rounding assigns a set S of jobs to be processed during the interval I then we need to speed up the execution of all jobs in the intervals I by factor $|S|$. This means that the energy consumption increases by the factor $|S|^{\alpha_i - 1}$. Therefore, the expected energy consumption during the interval I in the final approximate schedule is

$$E_I = \sum_{S \subseteq \{1,2,\ldots,n\}} |S|^{\alpha_i - 1} \left(\sum_{j \in S} e_j \right) P_S(Y'_1, Y'_2, \ldots, Y'_n)$$

where $P_S(Y'_1, Y'_2, \ldots, Y'_n)$ is the probability that exactly the jobs in the set S are selected during I, and $Y'_1, Y'_2, \ldots, Y'_n$ are independent Bernoulli random variables with $Pr(Y'_j = 1) = Y_j$. Therefore, we have that

$$E_I = \sum_{S \subseteq \{1,2,\ldots,n\}} |S|^{\alpha_i - 1} \left(\sum_{j \in S} e_j \right) \prod_{j \in S} Y_j \prod_{j \in \mathcal{J} \setminus S} (1 - Y_j)$$

We can assume that there exists $Q \in \mathbb{N}$ such that $Y_j = \frac{q_j}{Q}$, $1 \leq j \leq n$, for some $q_j \in \mathbb{N}$ (we don't make any assumptions on the encoding length of these numbers, we use them only for analysis purposes) since these numbers come from solving an LP with rational coefficients. Hence we can chop each Y_j into q_j pieces $Y_{j,\ell} = \frac{1}{Q} = Y$. Let $q = \sum_{j=1}^{n} q_j$ be the number of all chopped pieces and $e_{j,\ell} = e_j$, $1 \leq j \leq n$ and $1 \leq \ell \leq q_j$. Note that, $q \leq Q$ since $\sum_{j=1}^{n} Y_j \leq 1$. For the ease of exposition we identify the set $\{1, 2, \ldots, q\}$ with the set of all pairs (j, ℓ) such that $1 \leq j \leq n$ and $1 \leq \ell \leq q_j$. Using Proposition 1 we get

$$\begin{aligned} E_I &\leq \sum_{S \subseteq \{1,2,\ldots,q\}} |S|^{\alpha_i - 1} \left(\sum_{(j,\ell) \in S} e_{j,\ell} \right) Y^{|S|} (1-Y)^{q-|S|} \\ &= \sum_{k=1}^{q} \sum_{S \subseteq \{1,2,\ldots,q\}, |S|=k} \left(\sum_{(j,\ell) \in S} e_{j,\ell} \right) k^{\alpha_i - 1} Y^k (1-Y)^{q-k} \end{aligned}$$

By changing the order of the sums in the above inequality we get

$$\begin{aligned} E_I &\leq \left(\sum_{j=1}^{n} \sum_{\ell=1}^{q_j} e_{j,\ell} \right) \sum_{k=1}^{q} \binom{q-1}{k-1} k^{\alpha_i - 1} Y^k (1-Y)^{q-k} \\ &= \left(\sum_{j=1}^{n} q_j e_j \right) \sum_{k=1}^{q} \binom{q}{k} \frac{\binom{q-1}{k-1}}{\binom{q}{k}} k^{\alpha_i - 1} Y^k (1-Y)^{q-k} \\ &= \left(\frac{\sum_{j=1}^{n} q_j e_j}{q} \right) \sum_{k=1}^{q} \binom{q}{k} k^{\alpha_i} Y^k (1-Y)^{q-k} = \frac{Q}{q} LP_I^* \sum_{k=1}^{q} \binom{q}{k} k^{\alpha_i} Y^k (1-Y)^{q-k} \\ &\leq \frac{Q}{q} LP_I^* \mathbb{E}[B_{q/Q}^{\alpha_i}] \end{aligned}$$

where $\binom{q-1}{k-1}$ is the number of sets of cardinality k that contain j. Moreover, $B_{q/Q}$ is a random variable with expectation $\frac{q}{Q}$ which corresponds to the sum of q independent Bernoulli random variables. Therefore,

$$E_I \leq \frac{Q}{q} LP_I^* \cdot \mathbb{E}[B_{q/Q}^{\alpha_i}] \leq \frac{Q}{q} LP_I^* \cdot \mathbb{E}[P_{q/Q}^{\alpha_i}] \leq \frac{Q}{q} LP_I^* \cdot \frac{q}{Q} \mathbb{E}[P_1^{\alpha_i}]$$

where the second inequality follows from Proposition 2 and the last inequality follows from Proposition 3(a). Therefore, by summing over all intervals and processors and as $\alpha = \max_{i \in \mathcal{P}} \alpha_i$, we get

$$E \leq LP^* \cdot \mathbb{E}[P_1^{\alpha}] = LP^* \cdot \tilde{B}_{\alpha}$$

and then the theorem follows. ◀

4 Heterogeneous Multiprocessor with Migrations

In this section we present an algorithm for the heterogeneous multiprocessor speed-scaling problem with preemptions and migrations. We assume that, if x units of work for the job j are executed on the processor i, then $x/w_{i,j}$ portion of j is accomplished by i. We first formulate the problem as a configuration LP with an exponential number of variables and a polynomial number of constraints. Then, we consider the dual LP and we show how to apply the Ellipsoid algorithm to it and obtain an $OPT + \varepsilon$ solution for the primal LP.

A configuration c is a one-to-one assignment of n_c, $0 \leq n_c \leq m$, jobs to the m processors as well as an assignment of a speed value for every processor. We denote by $\mathcal{C}$ the set of all possible configurations. A well defined schedule for our problem has to specify exactly one configuration at each time t. The cardinality of $\mathcal{C}$ is unbounded, since the processors' speeds may be real values. Hence, we have to discretize the possible speed values and consider only a finite number of speeds at which the processors can run.

▶ **Lemma 6.** *There is a feasible schedule of energy consumption at most $OPT + \varepsilon$ that uses a finite (exponential to the size of the instance and polynomial to $1/\varepsilon$) number of discrete processors' speeds.*

In what follows in this section, we deal with schedules that satisfy Lemma 6. Let, now, $t_0 < t_1 < \ldots < t_\ell$ be the time instants that correspond to release dates and deadlines of jobs so that there is a time t_i for every possible release date and deadline. We denote by $\mathcal{I}$ the set of all possible intervals of the form $(t_{i-1}, t_i]$, for $1 \leq i \leq \ell$. Let $|I|$ be the length of the interval I.

We introduce a variable $x_{I,c}$, for each $I \in \mathcal{I}$ and $c \in \mathcal{C}$, which corresponds to the total processing time during the interval $I \in \mathcal{I}$ that the processors run according to the configuration $c \in \mathcal{C}$. We denote by $E_{I,c}$ the instantaneous energy consumption of the processors if they run with respect to the configuration c during the interval I. Moreover, let $s_{j,c}$ be the speed of the job j according to the configuration c. For notational convenience, we denote by (I, c) the set of jobs which are alive during the interval I and which are executed on some processor by the configuration c. Finally, let $i(j, c)$ be the processor on which the job j is assigned into configuration c. We propose the following configuration LP:

$$\min \sum_{I \in \mathcal{I}, c \in \mathcal{C}} E_{I,c} \cdot x_{I,c}$$

$$\sum_{c \in \mathcal{C}} x_{I,c} \leq |I| \qquad \forall I \in \mathcal{I} \tag{4}$$

$$\sum_{I,c:\ j \in (I,c)} \frac{s_{j,c}}{w_{i(j,c),j}} x_{I,c} \geq 1 \qquad \forall j \in \mathcal{J} \tag{5}$$

$$x_{I,c} \geq 0 \qquad \forall I \in \mathcal{I},\ c \in \mathcal{C}$$

Consider the schedule for the interval I that occurs by an arbitrary order of the configurations assigned to I. This schedule is feasible, as the processing time of all configurations assigned to I is equal to the length of the interval. Hence, Inequality (4) ensures that for each interval I there is exactly one configuration for each time $t \in I$. Inequality (5) implies that each job j is entirely executed.

The above LP has an exponential number of variables. In order to handle this, we create the dual LP, which has an exponential number of constraints. In the full version of the paper we will show how to efficiently apply the Ellipsoid algorithm to it (see [11]). For this, we provide a separation oracle, i.e., we give a polynomial-time algorithm which given a solution for the dual LP decides if this solution is feasible or otherwise it identifies a violated constraint. As we can compute an optimal solution for the dual LP, we can also find an optimal solution for the primal LP by solving it with the variables corresponding to the constraints that were found to be violated during the run of the ellipsoid method and setting all other primal variables to be zero. The number of these violated constraints is polynomial to the size of the instance and to $1/\varepsilon$. Thus, we can solve the primal LP with a polynomial number of variables.

▶ **Theorem 7.** *A schedule for the heterogeneous multiprocessor speed-scaling problem with migrations of energy consumption $OPT + \varepsilon$ can be found in polynomial time with respect to the size of the instance and to $1/\varepsilon$.*

5 Single processor without Preemptions

In this section we present an approximation algorithm for the single processor speed-scaling problem when the preemption of jobs is not allowed. As a single processor is available, each job $j \in \mathcal{J}$ has a unique release date r_j, deadline d_j and amount of work w_j, while when the processor runs at speed s, it consumes energy with rate s^α. Due to the convexity of the speed-to-power function, j runs at a constant speed s_j in an optimal schedule $\mathcal{S}^*$. Antoniadis and Huang [6] proved that this problem is $\mathcal{NP}$-hard and gave a $2^{5\alpha-4}$-approximation algorithm.

The algorithm in [6] consists of a series of transformations of the initial instance. Our algorithm applies the first of these transformations. Then, we give a transformation to the heterogeneous multiprocessor speed-scaling problem without migrations. For completeness, we describe the first transformation given in [6]. We partition the time as follows: let t_1 be the smallest deadline of any job in $\mathcal{J}$, i.e., $t_1 = \min\{d_j : j \in \mathcal{J}\}$. Let $\mathcal{J}_1 \subseteq \mathcal{J}$ be the set of jobs which are released before t_1, i.e., $\mathcal{J}_1 = \{j \in \mathcal{J} : r_j \leq t_1\}$. Next, we set $t_2 = \min\{d_j : j \in \mathcal{J} \setminus \mathcal{J}_1\}$ and $\mathcal{J}_2 = \{j \in \mathcal{J} : t_1 < r_j \leq t_2\}$, and we continue this procedure until all jobs are assigned into a subset of jobs. Let k be the number of subsets of jobs that have been created. Moreover, let $t_0 = \min\{r_j : j \in \mathcal{J}\}$ and $t_{k+1} = \max\{d_j : j \in \mathcal{J}\}$.

Consider the intervals $(t_{i-1}, t_i]$, $1 \leq i \leq k+1$. Let $\mathcal{I}_j$ be the set of intervals in which the job $j \in \mathcal{J}$ is alive. In some of them j is alive during the whole interval, while in at most two of them it is alive during a part of the interval. Consider now the non-preemptive problem in which the execution of j should take place into exactly one interval $I \in \mathcal{I}_j$. Note that the execution of j should respect its release date and its deadline.

▶ **Proposition 8.** *Let $\mathcal{S}$ be an optimal non-preemptive schedule for the problem in which the execution of each job $j \in \mathcal{J}$ should take place into exactly one interval $I \in \mathcal{I}_j$. It holds that $E(\mathcal{S}) \leq 2^{\alpha-1} OPT$.*

Next, we describe how to pass from the transformed problem to the heterogeneous multiprocessor speed-scaling problem without migrations. For every interval $(t_{i-1}, t_i]$, $1 \leq i \leq k+1$, we create a processor i. For every job $j \in \mathcal{J}$ which is alive during a part or during the whole interval $(t_{i-1}, t_i]$, $1 \leq i \leq k+1$, we set: (i) $r_{i,j} = 0$ if $r_j \leq t_{i-1}$ or $r_{i,j} = r_j - t_{i-1}$ if $r_j > t_{i-1}$, (ii) $d_{i,j} = t_i - t_{i-1}$ if $d_j > t_i$ or $r_{i,j} = d_j - t_{i-1}$ if $d_j \leq t_i$, and (iii) $w_{i,j} = w_j$. For each processor i, $1 \leq i \leq k+1$, we set $\alpha_i = \alpha$.

We next apply the approximation algorithm presented in Section 3. This algorithm will create a preemptive schedule $\mathcal{S}$. However, we can transform S into a non-preemptive schedule $\mathcal{S}'$ of the same energy consumption. To see this, note that in each processor i, $1 \leq i \leq k+1$, each job $j \in \mathcal{J}$ has $r_{i,j} = 0$ or $d_{i,j} = t_i - t_{i-1}$. Hence, by applying the Earliest Deadline First policy to each processor separately we can get the non-preemptive schedule $\mathcal{S}'$.

▶ **Theorem 9.** *The single processor speed-scaling problem without preemptions can be approximated within a factor of $2^{\alpha-1}((1 + \frac{\varepsilon}{1-\varepsilon})(1 + \frac{2}{n-2}))^\alpha \tilde{B}_\alpha$.*

6 Job Shop Scheduling with Preemptions

In this section, we consider the energy minimization problem in a job shop environment. The instance of the problem consists of a set of jobs $\mathcal{J}$, where each job $j \in \mathcal{J}$ consists of μ_j

operations $O_{j,1}, O_{j,2}, \ldots, O_{j,\mu_j}$, which must be executed in this order. That is, $O_{k+1,j}$ can start only once the operation $O_{j,k}$ has finished. Let μ be the number of all the operations, i.e. $\mu = \sum_{j \in \mathcal{J}} \mu_j$. Each operation $O_{j,k}$ has an amount of work $w_{j,k}$. Moreover, we are given a set of m heterogeneous processors $\mathcal{P}$. Each operation $O_{j,k}$, $j \in \mathcal{J}$ and $1 \leq k \leq \mu_j$, is associated with a single processor $i \in \mathcal{P}$ on which it must be entirely executed. Note that more than one operations of the same job may have to be executed on the same processor. Furthermore, for each operation $O_{j,k}$, we are given a release date $r_{j,k}$ and a deadline $d_{j,k}$. For each $j \in \mathcal{J}$, we can assume that $r_{j,1} \leq r_{j,2} \leq \ldots \leq r_{j,\mu_j}$ as well as $d_{j,1} \leq d_{j,2} \leq \ldots \leq d_{j,\mu_j}$. Preemptions of operations are allowed. The objective is to find a feasible schedule of minimum energy consumption.

We propose a configuration IP for the job shop problem. A configuration is a schedule for a job, i.e., a schedule for all its operations. To define the configurations, we discretize the time into a number of length slots, which is polynomial to the size of the instance and to $1/\varepsilon$. As the configuration LP relaxation has an exponential number of variables and a polynomial number of constraints, we consider its dual and we propose a separation oracle for it. Thus, we can solve our problem by applying the Ellipsoid algorithm. Then we apply the same randomized rounding as in Section 3.2 and the following theorem holds (we will give a formal proof in the full version).

▶ **Theorem 10.** *There is an algorithm of complexity polynomial to μ and to $1/\varepsilon$ with approximation ratio $(1+\varepsilon)^\alpha (1+\frac{2}{\mu-2})^\alpha (1+\frac{\varepsilon}{1-\varepsilon})^\alpha \tilde{B}_\alpha$ for the preemptive job shop scheduling problem with the energy objective.*

7 Routing

We are given a directed graph $G = (V, E)$. We are also given a set of demands $\mathcal{D}$. The demand $i \in \mathcal{D}$ is associated with a source node s_i and a destination node t_i and it requests d_i integer units of bandwidth. We consider the special case where all the demands request the same bandwidth, i.e. $d_i = d$ for all $i \in \mathcal{D}$. Each edge $e \in E$ is associated with a constant α_e such that if f units of demand cross e, then there is an energy consumption equal to $c_e f^{\alpha_e}$. The objective is to route all the demands from their sources to their destinations so that the total energy consumption is minimized. We consider the unsplittable version of the problem where each demand has to be routed through a single path.

The min-power routing problem can be formulated as an integer convex program (see [4]). Specifically, we introduce a variable x_e, for all $e \in E$, which corresponds to the number of demands that cross the edge e and a binary variable $y_{i,e}$ which indicates if the demand $i \in \mathcal{D}$ crosses the edge e. Then, we obtain the following integer convex program suggested in [4].

$$\min \sum_{e \in E} c_e d^{\alpha_e} \max\{x_e, x_e^{\alpha_e}\}$$

$$x_e = \sum_i y_{i,e} \qquad \forall e \in E \tag{6}$$

$$\sum_{e \in \Gamma^+(u)} y_{i,e} - \sum_{e \in \Gamma^-(u)} y_{i,e} = 0 \qquad \forall i \in \mathcal{D}, u \in V \setminus \{s_i, t_i\} \tag{7}$$

$$\sum_{e \in \Gamma^+(s_i)} y_{i,e} = 1 \qquad \forall i \in \mathcal{D} \tag{8}$$

$$\sum_{e \in \Gamma^-(t_i)} y_{i,e} = 1 \qquad \forall i \in \mathcal{D} \tag{9}$$

$$y_{i,e} \in \{0,1\} \qquad \forall i \in \mathcal{D}, e \in E \tag{10}$$

The above integer convex program is a valid formulation for our problem. Note first that our original goal is to minimize the total energy consumption for all edges, i.e., $\sum_{e\in E} c_e d^{\alpha_e} x_e^{\alpha_e}$. Since in an optimal integral solution using this objective all variables x_e are integers, the above program has the same optimal integral solution as if we have used as objective the $\sum_{e\in E} c_e d^{\alpha_e} x_e^{\alpha_e}$. However, the use of this objective leads to an integer program with large integrality gap [4]. For this reason, we modify the objective to be $\sum_{e\in E} c_e d^{\alpha_e} \max\{x_e, x_e^{\alpha_e}\}$ obtaining a program with smaller integrality gap. Equation (6) relates the variables x_e and $y_{i,e}$, while Equations (7)-(9) ensure the flow conservation.

In order to obtain a feasible integral solution for our problem, we solve the relaxation of the above convex program, where the constraints $y_{i,e} \in \{0,1\}$ are relaxed so that $y_{i,e} \geq 0$, and we obtain a fractional solution. Then, we apply a randomized rounding procedure, introduced by Raghavan and Thompson [13], in order to select a path for each demand. Specifically, for each demand $i \in \mathcal{D}$, we consider the subgraph of G that contains only the edges with $y_{i,e} > 0$ and define the standard flow decomposition. We compute a (s_i, t_i)-path p on this graph and we set $z_{i,p} = \min_{e\in p}\{y_{i,e}\}$. Then, we subtract $z_{i,p}$ from the variables $y_{i,e}$ which correspond to the edges of the path p. We continue this procedure until there are no (s_i, t_i)-paths. Due to the flow conservation, at this point there are no edges with $y_{i,e} > 0$.

The randomized rounding algorithm chooses a path p for the demand i with probability $z_{i,p}$. Note that $\sum_p z_{i,p} = 1$.

▶ **Theorem 11.** *There is a $\tilde{B}_\alpha$-approximation algorithm for the min-power routing problem with uniform demands.*

Proof. Consider an edge $e \in E$ and let $\lambda_e = \sum_{i\in\mathcal{D}} y_{i,e}$ be the expected value of the number of demands that cross e. The expected energy consumption on the edge e is

$$E_e = c_e d^{\alpha_e} \sum_{S\subseteq\mathcal{D}} |S|^{\alpha_e} Pr(S)$$

where $Pr(S)$ is the probability that exactly the demands in S are routed through (cross) the edge e. Hence, we have

$$E_e = c_e d^{\alpha_e} \sum_{S\subseteq\mathcal{D}} |S|^{\alpha_e} \prod_{i\in S} y_{i,e} \prod_{i\notin S}(1 - y_{i,e}).$$

Since $y_{i,e}$ come from a mathematical programming solver we can assume that that there exist $N \in \mathbb{N}$ such that $y_{ie} = \lambda_e \cdot \frac{q_{i,e}}{N}$ for some $q_{i,e} \in \mathbb{N}$. Similarly with the proof of Theorem 5, we can chop each $y_{i,e}$ into $q_{i,e}$ pieces $z_{i,e,\ell} = \frac{\lambda_e}{N}$. Note that, $N = \sum_{i\in\mathcal{D}} q_{i,e}$ since $\sum_{i\in\mathcal{D}} \frac{y_{i,e}}{\lambda_e} = 1$. For the ease of exposition we identify the set $\{1, 2, \ldots, N\}$ with the set of all pairs $((i, e), \ell)$ such that $i \in \mathcal{D}$ and $1 \leq \ell \leq q_{i,e}$. Iteratively applying Proposition 1 we get

$$\begin{aligned}
E_e &\leq c_e d^{\alpha_e} \sum_{S\subseteq\{1,2,\ldots,N\}} |S|^{\alpha_e} \left(\frac{\lambda_e}{N}\right)^{|S|} \left(1 - \frac{\lambda_e}{N}\right)^{N-|S|} \\
&= c_e d^{\alpha_e} \sum_{k=0}^{N} \sum_{S\in\{1,2,\ldots,N\},|S|=k} k^{\alpha_e} \left(\frac{\lambda_e}{N}\right)^{k} \left(1 - \frac{\lambda_e}{N}\right)^{N-k} \\
&= c_e d^{\alpha_e} \sum_{k=0}^{N} k^{\alpha_e} \binom{N}{k} \left(\frac{\lambda_e}{N}\right)^{k} \left(1 - \frac{\lambda_e}{N}\right)^{N-k}
\end{aligned}$$

as there are $\binom{N}{k}$ subsets of $\{1, 2, \ldots, N\}$ with k elements. The sum in the last expression is the α_e-th moment of Binomial random variable (sum of independent Bernoulli trials) with

expectation λ_e and hence by using Propositions 2 and 3 we get

$$E_e \leq c_e d^{\alpha_e} \mathbb{E}[P_{\lambda_e}^{\alpha_e}] \leq c_e d^{\alpha_e} \max\{\lambda_e, \lambda_e^{\alpha_e}\} \mathbb{E}[P_1^{\alpha_e}] = LP_e^* \tilde{B}_{\alpha_e}$$

where P_{λ_e} is a Poisson random variable with parameter λ_e. By summing up over all edges and setting $\alpha = \max_{e \in E}\{\alpha_e\}$, the theorem follows. ◀

In Table 1, we show that our analysis for the algorithm presented in [4] leads to a significantly better approximation ratio.

Acknowledgements. We would like to thank Oleg Pikhurko for providing the original proof of Part (b) of the Proposition 3.

References

1 S. Albers. Algorithms for dynamic speed scaling. In *STACS*, LIPIcs, Vol. 9, pages 1–11, 2011. doi: 10.4230/LIPIcs.STACS.2011.1

2 S. Albers, A. Antoniadis, and G. Greiner. On multi-processor speed scaling with migration: extended abstract. In *SPAA*, pages 279–288. ACM, 2011.

3 S. Albers, F. Müller, and S. Schmelzer. Speed scaling on parallel processors. In *SPAA*, pages 289–298. ACM, 2007.

4 M. Andrews, A. F. Anta, L. Zhang, and W. Zhao. Routing for power minimization in the speed scaling model. *IEEE/ACM Trans. on Networking*, 20:285–294, 2012.

5 E. Angel, E. Bampis, F. Kacem, and D. Letsios. Speed scaling on parallel processors with migration. In *Euro-Par*, volume 7484 of *LNCS*, pages 128–140, 2012.

6 A. Antoniadis and C.-C. Huang. Non-preemptive speed scaling. In *SWAT*, volume 7357 of *LNCS*, pages 249–260. Springer, 2012.

7 E. Bampis, D. Letsios, and G. Lucarelli. Green scheduling, flows and matchings. In *ISAAC*, volume 7676 of *LNCS*, pages 106–115. Springer, 2012.

8 B. D. Bingham and M. R. Greenstreet. Energy optimal scheduling on multiprocessors with migration. In *ISPA*, pages 153–161. IEEE, 2008.

9 A. R. Brodtkorb, C. Dyken, T. R. Hagen, J. M. Hjelmervik, and O. O. Storaasli. State-of-the-art in heterogeneous computing. *Sci. Program.*, 18:1–33, 2010.

10 G. Greiner, T. Nonner, and A. Souza. The bell is ringing in speed-scaled multiprocessor scheduling. In *SPAA*, pages 11–18. ACM, 2009.

11 M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimizations, 2nd corrected edition.* Springer-Verlag, 1993.

12 A. Gupta, S. Im, R. Krishnaswamy, B. Moseley, and K. Pruhs. Scheduling heterogeneous processors isn't as easy as you think. In *SODA*, pages 1242–1253, 2012.

13 P. Raghavan and C. D. Thompson. Randomized rounding: A technique for provably good algorithms and algorithmic proofs. *Comninatorica*, 7:365–374, 1991.

14 A. Wierman, L. L. H. Andrew, and A. Tang. Power-aware speed scaling in processor sharing systems. In *INFOCOM*, pages 2007–2015, 2009.

15 F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced CPU energy. In *FOCS*, pages 374–382, 1995.