Algorithmic Issues in Energy-Efficient Computation

Evripidis Bampis(✉)

Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, Paris, France
evripidis.bampis@lip6.fr

1 Introduction

Energy efficiency has become a crucial issue in Computer Science. New hardware and system-based approaches are explored for saving energy in portable battery-operated devices, personal computers, or large server farms. The main mechanisms that have been developed for saving energy are the ability of transitioning the device among multiple power states, and the use of dynamic voltage scaling (speed scaling). These last years, there is also an increasing interest in the development of algorithmic techniques for finding tradeoff-solutions between energy consumption and performance. In this talk, we will focus on algorithmic techniques with provably good performances for fundamental power management problems. Among the different models that have been developed in the literature, we will focus on the *speed scaling* model, the *power-down* model and the combination of these two models that we will call the *power-down with speed scaling* model. In the *speed scaling* model [54], the speed of the processor (machine) may be dynamically changed over time. When a processor runs at speed s, then the rate with which the energy is consumed (i.e., the power) is $f(s)$ with f a non-decreasing function of the speed. The energy is the integral of the power over time. According to the well-known cube-root rule for CMOS devices, the speed of a device is proportional to the cube-root of the power and hence $f(s) = s^3$, but in the literature, many works consider that the power is $f(s) = s^\alpha$ where $\alpha > 1$ is a constant, or an arbitrary convex function. In the *power-down* model [26], the processors run at a *fixed* speed but are equipped with a *sleep state*. This means that the processor has two states On and Off and may be suspended during its idle time. However, during its wake-up from state Off to state On, there is a *start-up* energy consumption, denoted by L. Hence, suspending the processor is only beneficial when the idle periods are long enough to compensate the consumed start-up energy. The *power-down with speed scaling* model [43] combines the previous two models by considering speed scalable processors with a sleep state. Here, the power function is $g(s) = f(s) + c$ where $f(s)$ is defined as in the speed-scaling model and $c > 0$ is a constant that specifies the power consumed when the processor is in the On state.

We will be interested in some recent developments for scheduling a set of jobs on a (set of) processor(s) focusing in the *offline* context. Most of the problems considered are *deadline-based* problems: we are given a set of n jobs, where each

DOI: 10.1007/978-3-319-44914-2_1

job j is characterized by its release date r_j, its deadline d_j and its processing volume (work) p_j. Two important families of instances that have been studied in the literature are the *agreeable* and the *laminar* instances. We call an instance *agreeable* if earlier released jobs have earlier deadlines, i.e., for each j and j' with $r_j \leq r_{j'}$ then $d_j \leq d_{j'}$. In a *laminar instance*, for any two jobs j and j' with $r_j \leq r_{j'}$ it holds that either $d_j \geq d_{j'}$ or $d_j \leq r_{j'}$. The processing volume of a job is the number of CPU-cycles required by the job. If job j is executed with speed s then its processing time is $\frac{p_j}{s}$. In the case of processors with fixed speed the processing volume equals the processing time of the job. A *feasible schedule* in this context is a schedule in which each job is executed in the interval between its release date and its deadline. The problems in this setting are bi-objective by nature. For instance, we wish to minimize the energy consumption while at the same time we aim to determine a feasible schedule. A lot of other objectives have been studied when we consider a given budget of energy: the *throughput*, i.e. the number of jobs that complete before their deadlines, the *makespan*, i.e. the time at which the last job completes its execution, the *sum of (weighted) completion times*, the *sum of flow times*, ... Finally, in some works, the objective is the minimization of a linear combination of the energy and of some scheduling criterion (e.g. sum of completion times).

2 Speed Scaling

2.1 Single Machine

Energy minimization. Yao et al. [54] considered the problem of scheduling a set of n jobs on a single machine, where the *preemption*, i.e. the possibility to interrupt the execution of a job and resume it later, was allowed. They proposed an optimal $O(n^3)$-time algorithm. Later, Li et al. [47] proposed a faster algorithm with time complexity $O(n^2 \log n)$. Other algorithms with better time complexities than the one of [54] have been proposed in [37] for agreeable instances, and in [36] for general instances. These algorithms exploit the relation of the energy minimization problem with the computation of shortest paths. When the instances are restricted to be laminar, Li et al. [46] showed that the problem can be solved in $O(n)$ time.

Antoniadis and Huang [16] were the first to consider the non-preemptive energy minimization problem. They proved that it is strongly $\mathcal{NP}$-hard even for laminar instances. They also presented a $2^{4\alpha-3}$-approximation algorithm for laminar instances and a $2^{5\alpha-4}$-approximation algorithm for general instances. Furthermore, the authors noticed that the problem can be solved optimally in polynomial time when the instances are agreeable by observing that the optimal preemptive schedule produced by the algorithm in [54] executes the jobs non-preemptively. A series of papers improved the approximation ratio of the non-preemptive case. In [21], an approximation algorithm of ratio $2^{\alpha-1}(1+\epsilon)^\alpha \tilde{B}_\alpha$ has been proposed where $\tilde{B}_\alpha = \sum_{k=0}^{\infty} \frac{k^\alpha e^{-1}}{k!}$ is the generalized Bell number which is defined for any $\alpha \in \mathbb{R}^+$ and corresponds to the α-th (fractional) moment of

Poisson's distribution. This algorithm improved the ratio given in [16] for any $\alpha < 114$. Then, an approximation algorithm of ratio $(12(1+\epsilon))^{\alpha-1}$ was given in [34], improving the approximation ratio for any $\alpha > 25$. In [23], an approximation algorithm of ratio $(1+\varepsilon)^{\alpha-1}\tilde{B}_\alpha$ has been presented which became the best algorithm, at that moment, for any $\alpha \leq 77$. Recently, a $(1+\epsilon)$-approximation algorithm which runs in $n^{O(polylog(n))}$ time has been proposed in [42].

Moreover, the relation between preemptive and non-preemptive schedules in the energy-minimization setting has been studied in [20]. The authors showed that starting from the optimal preemptive solution obtained using the algorithm of [54], it is possible to obtain a non-preemptive solution which guarantees an approximation ratio of $(1+\frac{p_{\max}}{p_{\min}})^\alpha$, where $p_{\max}$ and $p_{\min}$ are the maximum and the minimum processing volumes of the jobs. In the special case where all jobs have equal processing volumes this leads to a constant factor approximation of 2^α. For this special case, Angel et al. [9] and Huang and Ott [41], independently, proposed an optimal polynomial-time algorithm based on dynamic programming.

Throughput. Angel et al. studied the throughput maximization problem in the offline setting in [12]. They provided a polynomial time algorithm to solve optimally the single-machine problem for agreeable instances. More recently in [11], they proved that there is a pseudo-polynomial time algorithm for solving optimally the preemptive single-machine problem with arbitrary release dates, deadlines and processing volumes. For the weighted version, the problem is $\mathcal{NP}$-hard even for instances in which all the jobs have common release dates and deadlines. Angel et al. [12] showed that the problem admits a pseudo-polynomial time algorithm for agreeable instances. Furthermore, Antoniadis et al. [18] considered a related problem. More precisely, they studied a generalization of the classical knapsack problem where the objective is to maximize the total profit of the chosen items minus the cost incurred by their total weight. The case where the cost functions are convex can be translated in terms of a weighted throughput problem where the objective is to select the most profitable set of jobs taking into account the energy costs. They presented a fully polynomial time approximation scheme (FPTAS) and a fast 2-approximation algorithm for the non-preemptive problem where the jobs have no release dates or deadlines.

Sum of Completion Times. Pruhs et al. [49] considered the problem of minimizing the average completion time under a budget of energy. They proposed an $O(n^2 \log \frac{E}{\varepsilon})$ polynomial time algorithm for jobs with equal processing volumes, where E is the energy budget and ε the desired accuracy. Albers et al. [6] proposed a simplified algorithm for the problem of minimizing the average completion time plus energy for jobs with equal processing volumes which is based on dynamic programming. Megow et al. [48] considered the weighted version of the average completion time objective. When all the jobs have equal release dates, they established a polynomial time approximation scheme (PTAS). They also showed that the non-preemptive version of the problem is equivalent to the fixed-speed single-machine problem where the objective function is: $\sum w_j (C_j)^{\frac{\alpha-1}{\alpha}}$, where w_j is the weight of job j and C_j its completion time. This result has also been obtained independently by Vásquez [53]. For the preemptive problem where the jobs have arbitrary release dates, Megow et al. [48] proposed a $(2 + \varepsilon)$-approximation algorithm.

Makespan. Bunde [29] proposed an optimal polynomial-time algorithm for the problem of scheduling a set of jobs with arbitrary release dates and deadlines, under a given budget of energy, so that the makespan to be minimized.

Maximum Lateness. In [25], the non-preemptive problem of minimizing the maximum lateness, under a given budget of energy, has been studied. An optimal combinatorial polynomial-time algorithm has been proposed for the case in which the jobs have common release dates. For arbitrary release dates, the problem is shown to be strongly $\mathcal{NP}-$hard. The authors study also the problem where the objective is the minimization of a linear combination of maximum lateness and energy. The results for the budget variant can be adapted to this case. More interestingly, a 2-approximation algorithm is presented when the jobs are subject to release dates.

2.2 Multiple Machines

When more than one machines are available, we distinguish again between two cases: the *preemptive* and the *non-preemptive* cases. In the preemptive case, the execution of the jobs may allow the *migration* of the jobs, i.e. the possibility to execute a job on more than one machines, without allowing its parallel execution. This case is known as the *migratory* case. In the preemptive *non-migratory case*, the execution of a job must be done on the same machine.

We have also to distinguish between *homogeneous* and *heterogeneous* environments. In the homogeneous case, the characteristics of each job (release date, deadline and processing volume) are independent of the machine on which it is executed and the speed-to-power function is the same for all the machines. In the heterogeneous case, we consider the following subcases: In the *fully heterogeneous environment* both, the jobs' characteristics are machine-dependent and every machine has its own power function. Formally, the problem is as follows: we are given a set $\mathcal{J}$ of n jobs and a set $\mathcal{P}$ of m parallel machines. Every machine $i \in \mathcal{P}$ 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 machine i, then the power is $f(s) = s^{\alpha_i}$. Each job $j \in \mathcal{J}$ has a different release date $r_{i,j}$, deadline $d_{i,j}$ and processing volume $p_{i,j}$ in each machine $i \in \mathcal{P}$. In the *power-heterogeneous environment*, the characteristics of each job are independent of the machine on which the job is executed, while every machine has its own speed-to-power function. Finally, in the *unrelated-heterogeneous* environment the processing volumes of the jobs are machine-dependent while all the other characteristics are independent of the machine on which each job is executed.

Energy minimization. Chen et al. [30] were the first to study a multiprocessor energy-efficient scheduling problem involving speed scaling. More specifically, they proposed an $O(n \log n)$-time algorithm for solving optimally the homogeneous-migratory problem when the release dates and deadlines are identical for all the jobs. Later, Bingham et al. [28] constructed an optimal algorithm for the homogeneous-migratory problem when the jobs have arbitrary release dates and deadlines. The algorithm in [28] makes repetitive calls of a black-box algorithm for solving linear programs. Then, independently, Albers et al. [4] and Angel et al. [15] presented combinatorial algorithms based on a series of maximum flow computations that allow the partition of the set of jobs into subsets in which all the jobs are executed at the same speed. The optimality of these algorithms is based on a series of technical lemmas showing that this partition and the corresponding speeds lead to the minimization of the energy consumption. In [24], it has been shown that both the algorithms and their analysis can be greatly simplified. In order to do this, the problem has been formulated as a convex cost flow problem in an appropriate flow network. Furthermore, it has been shown that this approach is useful to solve other problems in the dynamic speed-scaling setting. As an example, the authors consider the *preemptive open-shop* speed-scaling problem and they propose a polynomial-time algorithm for finding an optimal solution based on the computation of convex cost flows. In [52], Shioura et al. consider the same formulation as convex cost flow for the homogeneous-migratory problem and they propose a method for reducing the running time of the algorithm. For the migratory problem in a *fully heterogeneous* environment, an algorithm using a configuration linear programming (LP) formulation, has been proposed in [21]. This algorithm returns a solution which is within an additive factor of ε far from the optimal solution and runs in time polynomial to the size of the instance and to $1/\varepsilon$. However, the algorithm proposed in [21] is based on the solution of a configuration linear program using the Ellipsoid method. Given that this method may not be very efficient in practice, an alternative polynomial-time algorithm based on a compact linear programming formulation which solves the problem within any desired accuracy was proposed in [5]. This algorithm does not need the use of the Ellipsoid method and it applies for more general than convex power functions; it is valid for a large family of continuous non-decreasing power functions. Furthermore, in the same work, a max-flow based algorithm has been proposed for the migratory problem in a *power-heterogeneous* environment, in which jobs' densities are lower bounded by a small constant, producing a solution arbitrarily close to the optimal.

For the *homogeneous non-migratory* problem, Albers et al. [7] considered the case of a set of jobs with unit processing volumes. They showed that the problem can be solved optimally in polynomial time if the instance is agreeable. Moreover, they established an $\mathcal{NP}$-hardness proof for the unit-work case when the release dates and the deadlines of the jobs are arbitrary. They proposed an $\alpha^{\alpha}2^{4\alpha}$-approximation algorithm for this special case. They have also presented an algorithm of the same approximation ratio for arbitrary-work instances when the jobs have either equal release dates or equal deadlines. Next, Greiner et al. [39] presented a $B_{\lceil\alpha\rceil}$-approximation algorithm for the problem with jobs having arbitrary processing volumes, release dates and deadlines, where $B_{\lceil\alpha\rceil}$ is the $\lceil\alpha\rceil$-th Bell number. Cohen-Addad et al. [34] proved that the non-migratory problem is APX-hard for the *unrelated-heterogeneous* model even if all the jobs have the same release dates and deadlines. For the non-migratory problem in a *fully heterogeneous* environment, an approximation algorithm of ratio $(1+\varepsilon)\tilde{B}_\alpha$ based on a randomized rounding of a configuration LP relaxation has been presented in [21].

For the non-preemptive problem in a *homogeneous* environment, Albers et al. [7] observed that the problem is $\mathcal{NP}$-hard even in the special case where the jobs have the same release dates and deadlines. Moreover, they showed that, for this special case, there exists a polynomial time approximation scheme (PTAS). For arbitrary release dates, deadlines and processing volumes; an approximation algorithm with ratio $m^\alpha(\sqrt[m]{n})^{\alpha-1}$ has been presented in [20]. Cohen-Addad et al. [34] presented an algorithm of ratio $(\frac{5}{2})^{\alpha-1}\tilde{B}_\alpha((1+\varepsilon)(1+\frac{p_{\max}}{p_{\min}}))^\alpha$. This algorithm leads to an approximation ratio of $2(1+\varepsilon)^\alpha 5^{\alpha-1}\tilde{B}_\alpha$ when all jobs have equal processing volumes. It has to be noticed that he authors in [34] observed that their algorithm can be used for the non-preemptive problem in the unrelated-heterogeneous model by loosing an additional factor of $(\frac{p_{\max}}{p_{\min}})^\alpha$. Finally, a $(1+\epsilon)$-approximation algorithm which runs in $n^{O(polylog(n))}$ time, for the non-preemptive problem in a *homogeneous* environment, has been presented in [42].

Throughput. The throughput maximization problem has been studied in the case of a fully heterogeneous environment in [14]. For the *fully heterogeneous non-migratory* problem, Angel et al. presented a greedy algorithm which is based on the primal-dual scheme that approximates the optimum solution within a factor depending on the speed-to-power functions (the factor is constant for functions of the form $f(s) = s^\alpha$). Then, they focused on the *homogeneous non-preemptive* problem for which they considered a *fixed* number of machines and two important families of instances: (1) instances with equal processing volume jobs; and (2) agreeable instances. For both cases they presented optimal pseudo-polynomial-time algorithms.

Sum of Completion Times. A polynomial-time algorithm for minimizing a linear combination of the sum of the completion times of the jobs and the total energy consumption, for the non-preemptive multiprocessor speed-scaling problem has been proposed in [24]. Instead of using convex cost flows, the proposed algorithm is based on the computation of a minimum weighted maximum matching in an appropriate bipartite graph.

Makespan. Shabtay and Kaspi [51] proved that the problem is NP-hard even if all the jobs have the same release dates. Pruhs et al. [50] observed that when all the jobs have the same release dates then a PTAS can be obtained using the load balancing algorithm of Alon et al. [8] for the minimization of the L_α norm of loads. Pruhs et al. considered in [50] the problem of scheduling a set of jobs on a set of speed scalable machines subject to precedence constraints among the jobs. The goal is to minimize the makespan of the schedule without exceeding a given energy budget. The approach in [50] is based on *constant power schedules*, which are schedules that keep the total power of all processors constant over time. Based on this property and by performing a binary search to determine the value of the power, they transformed the problem to the classical problem of minimizing the makespan for scheduling a set of jobs with precedence constraints on related parallel processors, in which each processor runs at a single predefined speed. The proposed algorithm has an approximation ratio of $O(\log^{1+2/\alpha} m)$, where m is the number of the machines. This ratio has been improved in [22] where a simple $(2-\frac{1}{m})$-approximation algorithm has been presented. The idea of this algorithm is the following: first, a convex programming relaxation for the speed scaling problem is given. The solution of this convex program defines a speed and hence a processing time for each job. Given that the obtained processing times respect the energy budget, it is then sufficient to use the classical list scheduling algorithm. This approach may be used for a more general problem where in addition to the precedence constraints the jobs are subject to release dates and/or precedence delays. For these generalizations, the approximation ratio of the algorithm remains asymptotically smaller than 2.

3 Power down

3.1 Single Machine

Chrétienne [33] proved that it is possible to decide in polynomial time whether there is a schedule with no idle time. Baptiste [26] proposed an $O(n^7)$-time dynamic programming algorithm for unit-time jobs and general L. For that, he proved a dominance property showing that there are only a few relevant starting points for the jobs in some optimal schedule and he proposed a clever decomposition of the problem. Then, Baptiste et al. [27] proposed an $O(n^5)$-time dynamic programming algorithm for the preemptive case with jobs of arbitrary processing times. They also proposed an $O(n^4)$ algorithm for unit-time jobs. A simpler dynamic programming with the same time-complexity for unit-time jobs has been proposed in [32]. Given the high time complexity of the algorithms in the general case, Gururaj et al. [40] improved the time-complexity by restricting their attention to agreeable instances. They proposed an $O(n \log n)$ algorithm for jobs with arbitrary lengths and with unit start-up energy consumption, i.e. $L = 1$. For arbitrary L and unit-time jobs, they proposed an $O(n^3)$ algorithm. In [10], this result has been improved by providing an $O(n^2)$ algorithm for arbitrary L and arbitrary processing times. In [31], a simple greedy algorithm has been presented that approximates the optimum solution within a factor of 2 and it has been shown that its analysis is tight. The algorithm runs in time $O(n^2 \log n)$ and needs only $O(n)$ memory. More recently in [32], different variants of the minimum-gap scheduling problem have been studied. These variants include the maximization of the throughput given a budget for gaps or the minimization of the number of gaps given a throughput requirement. Other objective functions are also studied. For instance, maximizing the number of gaps. For the model without deadlines, the authors focus on the tradeoff between the number of gaps and flow time.

3.2 Multiple Machines

The algorithm of [26] has been generalized for the multiple machines case in [35]. The time complexity becomes $O(n^7m^5)$, where n is the number of jobs and m is the number of machines. For agreeable instances, Gururaj et al. [40] proposed an $O(n^3m^2)$ algorithm for unit-time jobs and unit start-up energy consumption, $L = 1$. This result has been improved in [10], where an $O(n^2m)$ algorithm has been proposed.

4 Power-Down with Speed Scaling

While in the speed scaling model, it is always beneficial for the energy consumption to lower the speed of a job as far as the schedule remains feasible, this is not the case for the power-down with speed scaling model. Indeed, by increasing the speed of a job we may increase the length of some idle period and in that way be able to gain in energy consumption by turning off the machine. A central notion in this model is the notion of *critical speed* which, roughly speaking, is the speed minimizing the energy consumption while jobs are processed.

Irani et al. [43] proposed a 2-approximation algorithm for general convex power functions. The rough idea of the algorithm is the following: first, a schedule is produced using the algorithm of Yao et al. [54] for the speed scaling model. Given this schedule, the set of jobs is partitioned into two subsets: the first subset contains all the jobs that are executed with a speed higher than the critical speed, while the second subset contains the jobs that are executed with a speed smaller than the critical one. The schedule returned by the algorithm of Irani et al. [43] executes all the jobs of the first subset using the algorithm of Yao et al. [54], while all the jobs of the second subset are executed with the critical speed. Only recently, Albers and Antoniadis [3] and Kumar and Shannigrahi [45] proved that the problem is $\mathcal{NP}$-hard. For agreeable instances, an $O(n^3)$-time algorithm has been provided in [19]. This algorithm is based in a combination of the algorithm of Yao et al. and the use of dynamic programming for the jobs that are executed with a speed smaller than the critical speed. For general convex power functions, Albers and Antoniadis [3] derived a $\frac{4}{3}$-approximation algorithm. Their algorithm is also a combination of the algorithm of [54] and the use of dynamic programming. Here the partition of the jobs is not based on the critical speed, but on some appropriate value s_0. All the jobs executed in the schedule produced by the algorithm of [54] with a speed lower than s_0 are scheduled with speed s_0. The schedule of these jobs is derived by the dynamic program for the power-down model of Baptiste et al. [27]. All the other jobs are scheduled using the algorithm of [54]. Albers and Antoniadis have also obtained an approximation factor of $\frac{137}{117} < 1.171$ for power functions of the form $g(s) = \beta s^{\alpha} + c$, where s

is the speed and $\beta, c > 0$ as well as α are constants. More recently, in [17] a fully polynomial-time approximation scheme (FPTAS) for the problem has been proposed.

Finally, the single-machine non-preemptive throughput maximization problem has been studied in [13]. More precisely, optimal polynomial-time algorithms have been presented for two types of instances: (1) agreeable instances and (2) instances with arbitrary release dates and deadlines, but equal processing volumes. Both algorithms are based on dynamic programming.

To the best of our knowledge, no results are known for multiple machines.

5 Concluding Remarks

We gave a quick overview of some recent developments in the context of energy-efficient scheduling focusing on the offline setting. A huge literature exists for the online setting. For more results in this area, the interested reader is invited to consult the recent surveys in [1,2,38,44].

Acknowledgments. This work has been partially supported by the COFECUB project Choosing (n. 828/15).

References

1. Albers, S.: Energy efficient algorithms. Commun. ACM **53**(5), 86–96 (2010)
2. Albers, S.: Algorithms for dynamic speed scaling. In: International Symposium of Theoretical Aspects of Computer Science (STACS 2011), LIPIcs, vol. 9, pp. 1–11. Schloss Dagstuhl (2011)
3. Albers, S., Antoniadis, A.: Race to idle: new algorithms for speed scaling with a sleep state. In: Symposium on Discrete Algorithms (SODA), pp. 1266–1285. ACM-SIAM (2012)
4. Albers, S., Antoniadis, A., Greiner, G.: On multi-processor speed scaling with migration. In: Symposium on Parallelism in Algorithms and Architectures (SPAA), pp. 279–288. ACM (2011)
5. 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)
6. Albers, S., Fujiwara, H.: Energy efficient algorithms for flow time minimization. ACM Trans. Algorithms **3**(4), 49 (2007)
7. Albers, S., Muller, F., Schmelzer, S.: Speed scaling on parallel processors. In: Symposium on Parallelism in Algorithms and Architectures (SPAA), pp. 289–298. ACM (2007)
8. Alon, N., Azar, Y., Woeginger, G.J., Yadid, T.: Approximation schemes for scheduling. In: Proceedings of 8th Annual ACM-SIAM Symposium on Discrete Algorithms, New Orleans, Louisiana, 5–7 January 1997, pp. 493–500 (1997)
9. Angel, E., Bampis, E., Chau, V.: Throughput maximization in the speed-scaling setting. CoRR, abs/1309.1732 (2013)
10. Angel, E., Bampis, E., Chau, V.: Low complexity scheduling algorithms minimizing the energy for tasks with agreeable deadlines. Discret. Appl. Math. **175**, 1–10 (2014)

11. Angel, E., Bampis, E., Chau, V.: Throughput maximization in the speed-scaling setting. In: 31st International Symposium on Theoretical Aspects of Computer Science (STACS 2014), Lyon, France, 5–8 March 2014, pp. 53–62 (2014)
12. Angel, E., Bampis, E., Chau, V., Letsios, D.: Throughput maximization for speed-scaling with agreeable deadlines. In: Chan, T.H.H., Lau, L.C., Trevisan, L. (eds.) TAMC 2013. LNCS, vol. 7876, pp. 10–19. Springer, Heidelberg (2013)
13. Angel, E., Bampis, E., Chau, V., Thang, N.K.: Nonpreemptive throughput maximization for speed-scaling with power-down. In: Proceedings of Euro-Par: Parallel Processing 21st International Conference on Parallel and Dis-tributed Computing, Vienna, Austria, 24–28 August 2015, pp. 171–182 (2015)
14. Angel, E., Bampis, E., Chau, V., Thang, N.K.: Throughput maximization in multiprocessor speed-scaling. Theor. Comput. Sci. **630**, 1–12 (2016)
15. Angel, E., Bampis, E., Kacem, F., Letsios, D.: Speed scaling on parallel processors with migration. In: Kaklamanis, C., Papatheodorou, T., Spirakis, P.G. (eds.) Euro-Par 2012. LNCS, vol. 7484, pp. 128–140. Springer, Heidelberg (2012)
16. Antoniadis, A., Huang, C.C.: Non-preemptive speed scaling. J. Sched. **16**(4), 385–394 (2013)
17. Antoniadis, A., Huang, C.C., Ott, S.: A fully polynomialtime approximation scheme for speed scaling with sleep state. In: Proceedings of 26th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA, San Diego, CA, USA, 4–6 January 2015, pp. 1102–1113 (2015)
18. Antoniadis, A., Huang, C.-C., Ott, S., Verschae, J.: How to pack your items when you have to buy your knapsack. In: Chatterjee, K., Sgall, J. (eds.) MFCS 2013. LNCS, vol. 8087, pp. 62–73. Springer, Heidelberg (2013)
19. Bampis, E., Dürr, C., Kacem, F., Milis, I.: Speed scaling with power down scheduling for agreeable deadlines. Sustain. Comput.: Inform. Syst. **2**(4), 184–189 (2012)
20. Bampis, E., Kononov, A.V., Letsios, D., Lucarelli, G., Nemparis, I.: From preemptive to non-preemptive speed-scaling scheduling. Discret. Appl. Math. **181**, 11–20 (2015)
21. Bampis, E., Kononov, A.V., Letsios, D., Lucarelli, G., Sviridenko, M.: Energy efficient scheduling and routing via randomized rounding. In: IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS, Guwahati, India, 12–14 December 2013, pp. 449–460 (2013)
22. Bampis, E., Letsios, D., Lucarelli, G.: A note on multiprocessor speed scaling with precedence constraints. In: 26th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2014, Prague, Czech Republic, 23–25 June 2014, pp. 138–142 (2014)
23. 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, Heidelberg (2014)
24. Bampis, E., Letsios, D., Lucarelli, G.: Green scheduling, flows and matchings. Theoret. Comput. Sci. **579**, 126–136 (2015)
25. Bampis, E., Letsios, D., Milis, I., Zois, G.: Speed scaling for maximum lateness. Theor. Comput. Syst. **58**(2), 304–321 (2016)
26. Baptiste, P.: Scheduling unit tasks to minimize the number of idle periods: a polynomial time algorithm for offline dynamic power management. In: Symposium on Discrete Algorithms (SODA), pp. 364–367. ACM-SIAM (2006)
27. Baptiste, P., Chrobak, M., Dürr, C.: Polynomial-time algorithms for minimum energy scheduling. ACM Trans. Algorithms **8**(3), 26 (2012)

28. Bingham, B.D., Greenstreet, M.R.: Energy optimal scheduling on multiprocessors with migration. In: International Symposium on Parallel and Distributed Processing with Applications (ISPA), pp. 153–161. IEEE (2008)
29. Bunde, D.P.: Power-aware scheduling for makespan and flow. J. Sched. **12**(5), 489–500 (2009)
30. Chen, J.J., Hsu, H.R., Chuang, K.H., Yang, C.L., Pang, A.C., Kuo, T.W.: Multiprocessor energy efficient scheduling with task migration considerations. In: Euromicro Conference on Real-Time Systems (ECRTS), pp. 101–108. IEEE (2004)
31. Chrobak, M., Feige, U., Taghi Hajiaghayi, M., Khanna, S., Li, F., Naor, S.: A Greedy approximation algorithm for minimum-gap scheduling. In: Spirakis, P.G., Serna, M. (eds.) CIAC 2013. LNCS, vol. 7878, pp. 97–109. Springer, Heidelberg (2013)
32. Chrobak, M., Golin, M., Lam, T.-W., Nogneng, D.: Scheduling with gaps: new models and algorithms. In: Paschos, V.T., Widmayer, P. (eds.) CIAC 2015. LNCS, vol. 9079, pp. 114–126. Springer, Heidelberg (2015)
33. Chrétienne, P.: On single-machine scheduling without intermediate delays. Discret. Appl. Math. **156**(13), 2543–2550 (2008). In: 5th Conference, Honour of Peter Hammer's and Jakob Krarup's 70th Birthday, Graphs and Optimization, Fifth International Conference on Graphs and Optimization (GOV 2006)
34. Cohen-Addad, V., Li, Z., Mathieu, C., Milis, I.: Energy-efficient algorithms for non-preemptive speed-scaling. In: Bampis, E., Svensson, O. (eds.) WAOA 2014. LNCS, vol. 8952, pp. 107–118. Springer, Heidelberg (2015)
35. Demaine, E.D., Ghodsi, M., Hajiaghayi, M.T., Sayedi-Roshkhar, A.S., Zadimoghaddam, M.: Scheduling to minimize gaps and power consumption. In: Symposium on Parallelism in Algorithms and Architectures (SPAA), pp. 46–54. ACM (2007)
36. Gaujal, B., Navet, N.: Dynamic voltage scaling under EDF revisited. Real-Time Syst. **37**(1), 77–97 (2007)
37. Gaujal, B., Navet, N., Walsh, C.: Shortest-path algorithms for real-time scheduling of FIFO tasks with minimal energy use. ACM Trans. Embed. Comput. Syst. **4**(4), 907–933 (2005)
38. 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)
39. Greiner, G., Nonner, T., Souza, A.: The bell is ringing in speed scaled multiprocessor scheduling. In: Symposium on Parallelism in Algorithms and Architectures (SPAA), pp. 11–18. ACM (2009)
40. Gururaj, Jalan, and Stein. Unpublished work, see survey of m. chrobak
41. Huang, C.-C., Ott, S.: New results for non-preemptive speed scaling. In: Csuhaj-Varjú, E., Dietzfelbinger, M., Ésik, Z. (eds.) MFCS 2014, Part II. LNCS, vol. 8635, pp. 360–371. Springer, Heidelberg (2014)
42. Im, S., Shadloo, M.: Brief announcement: a QPTAS for non-preemptive speed-scaling. In: Proceedings of ACM SPAA (2016)
43. Irani, S., Gupta, R.K., Shukla, S.K.: Competitive analysis of dynamic power management strategies for systems with multiple power savings states. In: Conference on Design, Automation and Test in Europe (DATE), pp. 117–123. IEEE (2002)
44. Irani, S., Pruhs, K.: Algorithmic problems in power management. ACM SIGACT News **36**(2), 63–76 (2005)
45. Kumar, G., Shannigrahi, S.: On the NP-hardness of speed scaling with sleep state. Theor. Comput. Sci. **600**, 1–10 (2015)
46. Li, M., Liu, B.J., Yao, F.F.: Min-energy voltage allocation for tree-structured tasks. J. Comb. Optim. **11**(3), 305–319 (2006)

47. Li, M., Yao, A.C., Yao, F.F.: Discrete and continuous min-energy schedules for variable voltage processors. Proc. Nat. Acad. Sci. U.S.A. **103**(11), 3983–3987 (2006)
48. Megow, N., Verschae, J.: Dual techniques for scheduling on a machine with varying speed. In: Fomin, F.V., Freivalds, R., Kwiatkowska, M., Peleg, D. (eds.) ICALP 2013, Part I. LNCS, vol. 7965, pp. 745–756. Springer, Heidelberg (2013)
49. Pruhs, K., Uthaisombut, P., Woeginger, G.J.: Getting the best response for your erg. ACM Trans. Algorithms **4**(3), 38 (2008)
50. Pruhs, K., van Stee, R., Uthaisombut, P.: Speed scaling of tasks with precedence constraints. Theor. Comput. Syst. **43**(1), 67–80 (2008)
51. Shabtay, D., Kaspi, M.: Parallel machine scheduling with a convex resource consumption function. Eur. J. Oper. Res. **173**(1), 92–107 (2006)
52. Shioura, A., Shakhlevich, N., Strusevich, V.: Energy optimization in speed scaling models via submodular optimization. In: 12th Workshop on Models and Algorithms for Planning and Scheduling Problems (MAPSP) (2015)
53. Vásquez, O.C.: Energy in computing systems with speed scaling: optimization and mechanisms design (2012). arXiv:1212.6375
54. Yao, F.F., Demers, A.J., Shenker, S.: A scheduling model for reduced cpu energy. In: Symposium on Foundations of Computer Science (FOCS), pp. 374–382. IEEE (1995)