W. Karl, J. Keller (Eds.): PARS 2017
Proc. 27th PARS-Workshop

Minimizing Energy Cost in Task-Graph Execution on Parallel Platforms

Rainer Gerhards[1], Jörg Keller[1]

Abstract: We investigate minimization of energy cost for execution of statically scheduled task graphs on parallel machines with frequency scaling and given deadlines, assuming that the power profile of the processing elements and the energy price curve over time is known or can be predicted. We present both a mixed integer linear program and a heuristic to solve this problem, using time slots of fixed lengths and discrete frequency levels for both approaches and a fixed budget per time slot for the heuristic. We evaluate the heuristic by comparison to cost-optimal schedules. For price curves occurring in practice, and for deadlines not too close to the minimum makespan, the heuristic produces about 15% more energy cost than the optimal solution.

Keywords: Static scheduling, Taskgraph, Energy Cost

1 Introduction

Task graphs are widely used as models for parallel programs [KA99], and static scheduling of task graphs is used in applications ranging from high-performance computing to real-time systems [MLK+11], e.g. to enable repeated execution of predictable workloads in embedded systems. With the advent of frequency scaling, scheduling for energy efficiency has been widely investigated. In particular, static scheduling of task graphs onto a parallel platform with the goal to meet a deadline and minimize the energy for execution has been investigated with both optimal and heuristic solutions [EK17]. The availability of renewable energy, e.g. solar energy, at certain times has shifted this target, as the cost of solar energy is often considered negligible compared to the cost of brown energy (non-renewable sources) [GLH+11], and storing energy is costly both in terms of machinery and loss of energy [WLLMR14]. Moreover, the price for brown energy is not necessarily fixed: large scale consumers like large computing centres might choose a tariff with their provider that varies over the day [CGK10]. This also influences scheduling decisions if we consider a computing grid with resources in different time zones. These developments call for scheduling strategies that optimize *energy cost* while meeting a certain deadline.

A number of works have investigated this field, but they do not work with task graphs, sometimes focus on trading solar and brown energy, sometimes on predicting energy price, and sometimes focus solely on heuristic solutions like follow-the-sun (exploit as much solar energy as possible) or follow-the-moon (exploit as much cheap brown energy at night time as possible).

[1] FernUniversität in Hagen, Faculty of Mathematics and Computer Science, 58084 Hagen, Germany, firstname.lastname@fernuni-hagen.de

We try to take a broader view and consider the following scenario: we assume that we know the price per energy unit and time slot[2] from now until the deadline, and we know the power and performance characteristic of each processing element (PE) in the computing platform used. We provide a mixed integer linear program (mILP) that schedules the task graph onto the machine such that the deadline is met and and the energy cost is minimized. Additionally, we provide a heuristic for determining the start times of tasks and scaling the task execution frequencies per time slot, given a mapping and ordering of the tasks onto PEs. We evaluate the heuristic by comparing simulated runtimes (and energy cost) with optimal mILP solutions for small task graph instances from a benchmark suite [HS04].

The remainder of this article is structured as follows. In Section 2, we discuss related work. In Section 3, we detail our computational model. In Section 4, we present the mixed integer linear program and the heuristic. Section 5 presents our experiments and discusses the results. In Section 6, we conclude and give an outlook to future work.

2 Related Work

We schedule static tasks graphs on homogeneous[3] multi-PE systems as they model a multitude of parallel applications with known structure and behavior. A *static task graph* is a directed acyclic graph where the nodes represent tasks to be executed and edges represent precedence constraints between tasks. Nodes are annotated with the expected task execution time. Edges are annotated with communication costs. A *homogeneous multi-PE system* consists of multiple processing elements (PE) offering equal capabilities. A *schedule* is a mapping of tasks to PEs. It is feasible iff task executions on a PE never overlap, and if it respects precedence constraints including communication costs. Communications costs need to be applied iff predecessor and successor task run on different PEs. The *makespan* of the schedule is the finish time of the last task(s) inside the task graph. The classic problem that assumes fixed-speed PEs is to minimize makespan, formally classified as $P_m|prec|C_{max}$ in [GLLK79].

Modern PEs support dynamic frequency scaling from a discrete set of frequencies. With increasing frequency, processable workload grows linearly, but power consumption grows super-linearly. Literature often assumes that the power function grows quadratic [EK17] or cubic [PvSU08]. So it is more energy-efficient to execute the same workload at a lower frequency for a longer period of time. This has been studied intensely. Pruhs et al have shown that minimum energy is consumed if running all PEs at the lowest frequency meeting deadline requirements [PvSU08] ("*constant power schedule*"). Dorronsoro et al propose a two-layer scheduler for large-scale environments where the detail level includes semi-dynamic decisions [DNT+14]. Eitschberger et al also propose a dynamic component, this time used to handle faults during processing flow [EK17]. Most importantly, they also present an mILP for optimal energy-efficient frequency scaling for static task

[2] We consider a local platform for simplification, but our findings can be generalized to scenarios where the price also depends on the place of a processing element.

[3] Our methods can also be applied to heterogeneous systems, but we wanted to restrict the number of variables to avoid interference between different effects.

graphs. There is little literature on energy cost, but a richer field of work on green data center and optimization of renewable vs. brown energy. Prediction of renewable energy plays an important rule. For example Giori et al include prediction processes into the actual scheduling algorithm [GLH+11]. Another idea is workload classification, usually into time-critical and background work, which can be deferred, found for example in Lei et al [LZL+15]. Most importantly, they work with pre-computed predictions, which is a model we consider especially useful for static scheduling. Li et al propose a model where the workload is opportunistically adapted to renewable energy availability [LOM17]. Even if not explicitly spelled out, the core motivation of preferring green energy is a cost argument: either given as actual market price or CO2-emissions that shall be avoided. Green energy depends on time of day, which is obvious for solar energy. The same holds true for brown energy: for example Wierman et al [WLLMR14] describe "time of use", "day ahead" and "peak pricing" models (among others). A cost-aware scheduling model needs to use discrete time to support these pricing models. This is common for dynamic, but not for static schedulers. Also, these price models cannot be described by a continuous function. As such running a PE at a higher frequency during an inexpensive time period can be less costly than running it at a low frequency during an expensive period. This turns the static scheduling problem into a multi-goal optimization problem (makespan and cost minimization). It also means that there is no simple guiding function to explore the solution space, in contrast to work purely focusing on minimization of energy. In our work, we combine ideas from energy minimization with results from dynamic scheduling cost optimization. We further assume that a sufficiently precise energy cost prediction is available.

3 Model

We use discrete time slots $T = (t_0, t_1, \ldots)$, where the duration t of all time slots t_i is equal. W.l.o.g. we use low time resolution. If finer resolution is required, one can always obtain it by scaling both time slot duration and the rest of the model accordingly.

Let P be the set of available processing elements p_i. All p_i are homogeneous but can independently be frequency-scaled. We assume a set of discrete frequencies $F = (f_0, \ldots, f_n)$ with $1 = f_0 > f_1 > \ldots > f_{n-1} > f_n = 0$. Frequency 0 models a turned off PE. Let $\text{work}(f) = f \cdot t$ be the units of work performed by a PE during one time slot when running at frequency f. Let $\text{power}(f) = f^2$ be the power used when a PE is running at frequency f and let $\text{energy}(f) = \text{power}(f) \cdot t$ be the energy consumed when running a PE at frequency f for one time slot of length t. We assume a PE can be turned off and on with negligible energy consumption and does not consume any energy while turned off. At any time t_i, a single frequency f_i can be used. If needed, finer control of frequencies can be achieved by increasing the time resolution. If a workload $w < \text{work}(f_k)$ is scheduled at frequency f_k at a time slot, then $\text{work}(f_k) - w$ processing capability remains unused. This loss can be minimized by selecting the minimal frequency sufficient to perform w.

Let $\mathscr{T} = (D, J, C)$ be the task graph to be executed with deadline D, $J = (j_0, \ldots, j_n)$ the individual task nodes, and C the set of edges (i, i') representing precedence constraints: task i must be completed before i' can be started. Each task j_i is attributed with its work-

load w_i being expressed in number of time slots required to execute the task at execution frequency 1. We ignore communication costs, so edges are *not* annotated with cost.

The energy price schedule $R = (r_0, \ldots, r_m), m \geq |T|$ specifies the energy price rate r_i for one unit of work performed at frequency 1, i.e. for one unit of energy, during time slot t_i. Prices are normalized to $1 = \max(R)$. The energy cost of processing a task on a PE with frequency $f \in F$ during slot t_i is $\text{energy}(f) \cdot r_i$. The energy cost of a task is the cost for processing in all time slots that the task is active. The energy cost of a schedule is the sum of its task costs.

Schedule Let S be a not necessarily optimal schedule for $\mathscr{T}$, i.e. a mapping of tasks to PEs that assigns each task a start time. In S all PEs are scheduled with maximum frequency and $\text{makespan}(S) \leq D$. Let ε be a permitted deadline extension. Deadline extensions are supported in our model for jobs that have some flexibility in completion time, represented by ε. This flexibility permits to gain cost savings. If the deadline is tight, scaling of tasks is limited by the length of the critical path (CP) and free slack in non-CP tasks. Let $D' = D \cdot (1+\varepsilon)$ be the maximum time permitted for execution. In the following, let i denote a task j_i, j denote a PE p_j, k denote a frequency f_k, and l denote a time slot t_l.

For a schedule S of a job $\mathscr{T} = (D, J, C)$, let

$$x_{i,j,k,l} = \begin{cases} 1 & \text{if } j_i \text{ is executed on } p_j \text{ at frequency } f_k \text{ during time slot } t_l \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

be a binary function that describes the mapping to the target system. Note that this is the actual schedule. Let $M = \text{makespan}(S) = \max\{l \mid (\exists_{i,j,k})[x_{i,j,k,l} = 1]\} + 1$ be the makespan of S (plus one because time slots start at index zero). Let $D' = \lfloor D(1+\varepsilon) \rfloor$ be an extended deadline for some ε. In the following, let l be restricted to $0 \leq l < D'$ (strict due to zero-based indexing).

A schedule S for $\mathscr{T}$ is *feasible* iff $M \leq D'$, all task precedence constraints are obeyed, each task with a non-zero workload is mapped to one PE, a task j_i mapped to a PE p_j must be executed sequentially and not be interrupted by another task or migrated to another PE. Also, at any given time, one task is only mapped to one PE at one frequency, one PE can only execute one task at one frequency, at least one PE must be mapped (note that a turned-off PE is also mapped, but not performing actual work). Let $\text{start}(i) = \min\{l \mid (\exists_{j,k})[x_{i,j,k,l} = 1]\}$ be the first and $\text{stop}(i) = \max\{l \mid (\exists_{j,k})[x_{i,j,k,l} = 1]\}$ the last time slots in which j_i is mapped. The actual workload performed up to and including time t_l is $w_{\text{actual}}(i,l) = \sum_{j,k} \sum_{p=\text{start}(i)}^{l} (x_{i,j,k,p} \cdot \text{work}(k))$. The total workload $w_{\text{total}}(i) = w_{\text{actual}}(i, \text{stop}(i))$ can be slightly greater than w_i because of unused processing time at the end of the last time slot. Let $w_{\text{remain}}(i,l) = w_i - w_{\text{actual}}(i, l-1)$ be the workload remaining to be done at the beginning of time slot t_l. Then $f_{\text{suff}}(i) = \min\{f_k \mid 0 \leq k < |F| \wedge \text{work}(f_k) \geq w_{\text{remain}}(i, \text{stop}(i))\}$ is the minimal sufficient frequency to complete task j_i at $t_l = t_{\text{stop}(i)}$. As t_l is the last time slot, $\text{remain}(i,l) \leq 1$ because otherwise the work could not be completed in one time slot. As such $f_{\text{suff}}(i)$ will always yield a valid frequency. Thus we have

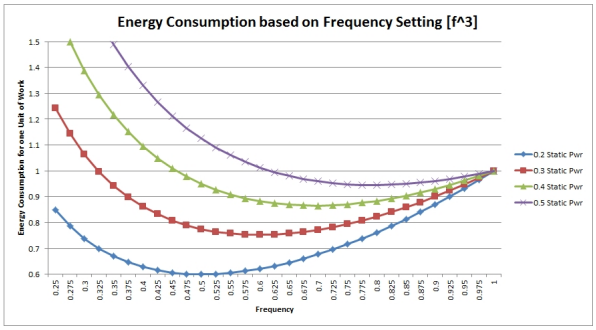

Fig. 1: Effect of PE Static Power Consumption

$\text{loss}(i) = \text{work}(f_{\text{suff}}(i)) - \text{remain}(i, \text{stop}(i))$ units of processing capacity lost by this mapping of S. As we have discrete frequencies, this loss is inevitable except when w_i can be expressed as a sum of products of the discrete frequencies, which is highly unlikely. With these definitions, we can restrict the model to the minimum required workload per task via

$$w_i \leq w_{\text{total}}(i) \leq w_i + f_{\text{suff}}(i) \leq w_i + 1 \tag{2}$$

The slightly relaxed upper bound $w_i + 1$ may be used as approximation inside a mILP in order to keep the model complexity reasonable.

Let $c_{k,l} = \text{energy}(f_k) \cdot r_l$ be the cost of executing a task with frequency f_k in time slot t_l. Then $\text{cost}(S) = \sum_{i,j,k,l}(x_{i,j,k,l} \cdot c_{k,l})$ is the total cost incurred for executing S.

Our goal is to create a feasible schedule S' (either from scratch or by transforming a previous schedule S) with $\text{makespan}(S') \leq D$ and minimum cost.

PE Static Power Consumption While not extensively investigated in this work, we would like to mention the effect of static power consumption on optimization. Let s be the power consumed by a PE irrelevant of frequency. Then the frequency-related part of power consumption needs to be considered as cubic [LSK14], resulting in $\text{power}(f) = s + (1-s)f^3$, where $s \in [0,1]$ and dynamic power is scaled by $1-s$ to normalize power consumption in [0,1], and thus $\text{energy}(f) = (s + (1-s)f^3)\frac{1}{f}$ for a unit of workload. As can be seen by differentiation, the energy function is still convex in f, but it is not continuously increasing anymore over the frequency range. As can be seen in Fig. 1 the energy curve decreases to a minimum and after that it increases. In regard to energy consumption, s places a hard lower bound on f: for lower frequencies, we need to prolong the execution by $1/f$. If we ignore dynamic power, power consumption for performing a single unit of work becomes s/f. As such, any frequency below s results in energy consumption larger than one. A pure energy optimization problem thus is severely limited in frequency scaling when a considerable part of PE total energy consumption is static. For example, at $s = 0.5$ the minimum frequency is approximately 0.825.

In cost-optimization, running at lower frequencies may still be useful. This can for example be the case when the pricing schedule has an expensive time slot t_1 followed by an

inexpensive t_2 followed by an even more expensive t_3. Here, it might be useful to run a task at $f = 1$ during t_2, but also run it at a frequency $f < s$ during t_1 if this is the only method to avoid running at t_3.

4 Cost-Optimal Scheduling and Heuristic

Integer Linear Program We present a mILP for computing a cost optimal schedule according to the model description in Section 3. Let $\mathscr{T} = (D, J, C)$ be a task graph and let D' be an extended deadline. Let variables $w_i \in \mathbb{R}$ be the workload of task i and let variables $W_k \in \mathbb{R}$ be the workload that can be processed at frequency k in one time slot. We need $|J| \cdot |P| \cdot |F| \cdot D'$ binary decision variables $x_{i,j,k,l}$ to represent the mapping from Eq. 1. At any time, task i must be mapped on a PE j with one frequency or not be mapped at all:
(1) $\forall i, j, l : \sum_k x_{i,j,k,l} \leq 1$.
The mapping must be sufficient to execute the workload
(2) $\forall i : \sum_{j,k,l} (x_{i,j,k,l} \cdot W_k) \geq w_i$.
A PE must never have more than one task mapped at any time
(3) $\forall j, l : \sum_{i,k} x_{i,j,k,l} \leq 1$.

Let binary decision variable $z_{i,j} = 1$ iff task i is mapped on PE j. If task i is mapped onto PE j it must not be mapped onto any other PE at any frequency anytime:
(4) $\forall i, j : \forall j' \neq j : \sum_{k,l} x_{i,j',k,l} \leq (1 - z_{i,j}) \cdot D'$.
Further, it must be mapped to exactly one PE:
(5) $\forall i : \sum_j z_{i,j} = 1$.
Let binary decision variable $y_{i,l} = 1$ iff task i starts in time l. We need to ensure that each task has only one start time:
(6) $\forall i : \sum_l y_{i,l} = 1$.
Further, task i must not be mapped onto a PE before its start time:
(7) $\forall i, l : \sum_{j,k} \sum_{l' < l} x_{i,j,k,l'} \leq (\sum_{l'' \leq l} y_{i,l''}) \cdot D'$.
Similarly, let binary decision variable $Y_{i,l} = 1$ iff task i stops at time l (this is the last time slot in which it is executed). We need to ensure that each task has only one stop time:
(8) $\forall i : \sum_l Y_{i,l} = 1$.
Further, task i must not be mapped onto a PE after its stop time:
(9) $\forall i, l : \sum_{j,k} \sum_{l' > l} x_{i,j,k,l'} \leq (\sum_{l'' \geq l} Y_{i,l''}) \cdot D'$.
With these definitions, the start time to task i is $\sum_l (l \cdot y_{i,l})$ and the stop time is $\sum_l (l \cdot Y_{i,l})$. We must ensure that a task starts before it ends:
(10) $\forall i : 1 + \sum_l (l \cdot y_{i,l}) \leq \sum_l (l \cdot Y_{i,l})$.
Now we can define the remaining constraints for $x_{i,j,k,l}$: If task i is mapped onto PE j, it must continuously be mapped from start until end:
(11) $\sum_{j,k,l} x_{i,j,k,l} = 1 + \sum_l (l \cdot Y_{i,l}) - \sum_l (l \cdot y_{i,l})$.
We need to enforce task precedence constraints by
(12) $\forall (i, i') \in C : \sum_l (l \cdot y_{i',l}) - \sum_l (l \cdot Y_{i,l}) > 0$.
Note that the relationship must be strict because the stop time is the last time slot in which i is mapped. We must also enforce $0 \leq l \leq D'$, which then enforces that no start time is lower than 0 or end time larger than D'.

From the model, let constants $r_l \in R \subset \mathbb{R}$ be the energy price rate in time slot l, and let constants $E_k \in \mathbb{R}$ be the energy consumed in one time slot at frequency k. Our goal then is

$$\min \text{cost} = \sum_{i,j,k,l} (x_{i,j,k,l} \cdot E_k \cdot r_l)$$

.

Heuristic The heuristics presented here use a pre-computed makespan-optimal schedule S with all PEs running at maximum frequency, makespan M and permitted deadline extension $\varepsilon \geq 0$ as basis and transforms it into a cost-minimized schedule S' with deadline $D' := M(1+\varepsilon)$, makespan$(S') \leq D'$, PE frequencies scaled to minimize energy cost, but PE assignments otherwise unaltered, and cost$(S') \leq$ cost(S). A feasible solution always exists: S is in the solution set and may be selected if no other feasible S' can be obtained.

We use a common heuristic framework to conduct experiments with different algorithm details. The framework uses the following variables:

1. B, overall cost budget for the schedule,
2. b represents overall cost budget under consideration,
3. b_l represents remaining budget in time slot l,
4. r_l represents energy price in time slot l,
5. $x_{i,j,k,l}$ represents the schedule (see equation 1),
6. D' represents the extended deadline,
7. $J_r \subseteq J$ tasks ready to run,
8. $J_w \subseteq J$ tasks not yet ready to run because of precedence constraints.

An important detail problem is to ensure B is sufficiently large to create a feasible schedule S'. Let C_m be the cost incurred by the makespan-optimal schedule S. In S, we have no budget restrictions on each time slot. In particular, the budget per time slot can differ greatly. Assume we enforce $b_l := {}^{C_m}/_{D}$ as maximum budget per time slot. In this case, we may be unable to re-compute S, because it may require a budget larger than b_l in some time slots. This is even more the case if we use $b_l := {}^{C_m}/_{D'}$, because the deadline extension decreases budget per time slot. To avoid these kinds of problems, we heuristically use $B := \sigma C_m$ and choose a sufficiently large scale factor σ. In most of our experiments $\sigma := 2$ worked well. Choosing B too large may increase experiment runtime but will not affect result optimality as we minimize cost and thus B.

The major steps in the framework are:

1. initialize all data structures
2. search the solution space
    a) create mapping
        i. pick task $i \in J_r$ with earliest start time (based on S)
        ii. assign i to the same PE it is assigned to in S

iii. scale frequencies k_l, so that for all time slots l during execution of i, power$(k_l) \cdot r_l \leq b_l$ holds
iv. adjust earliest start time of all tasks in J_w, J_r so that they reflect the new stop time of i due to frequency scaling
v. for all tasks $i' \in J_w$ now ready to run: $J_w := J_w \setminus \{i'\}, J_r := J_r \cup \{i'\}$
vi. $J_r := J_r \setminus \{i\}$
vii. continue until $J_r = \emptyset$

b) evaluate mapping, continue until goal is reached

3. output the best solution

Note that the heuristic only scales frequencies but keeps the structure of original schedule S. This is because we use the earliest start times (EST) of S and the same PE assignment. We modify the EST of S only to adapt for the consistent execution time changes due to frequency scaling. The PE assignment is only altered in respect to frequencies, something not even considered in S. As such, the order of task execution and task-to-PE assignment is kept unmodified.

Steps one and three of this algorithm are self-explanatory. They are identical for all experiments. In step two, different methods are used:

The *sweep search* mode linearly explores the solution space in an exhaustive and stepwise manner. It creates a number of equally-spread samples I of the solution space. It uses an ordered set $(b_1, b_2, \ldots, b_I) := (\frac{B}{I}, 2\frac{B}{I}, \ldots, I\frac{B}{I})$ of individual candidate mappings. For each $b \in (b_i)$ a candidate solution is computed and the best one selected as final solution. As such, this mode offers a thorough evaluation of the solution space. Assuming a sufficiently large I, it provides good solutions also for very rough search landscapes. It requires $O(I)$ runtime. This mode is also useful to investigate the structure of the solution space.

The *binary search* mode performs a classical binary search within $0 \leq b \leq B$. Starting from the highest and lowest bound, always a middle budget b is selected. For this budget, a mapping is generated. This is compared to previous result and upper or lower budget bound adjusted accordingly. The search is terminated when it converges towards a solution. This method provides an $O(\log B)$ exploration of the solution space, but does not work well on rough solutions spaces.

Different methods can be used to create the time slot assignment in step 2. The current approach only implements a greedy mapper: When task i has been obtained, it is mapped starting from the PE's first available time slot and is mapped until completed. This means that it receives priority over all other tasks which are mapped after it. Each task can only use the remaining budget, maintained in b_l. In an extreme, a single task may block all other tasks from execution by leaving insufficient remaining budget.

5 Experiments

For our experiments, we used the 32 different task graphs from [EK17], each comprising up to 32 tasks. They are a subset from [HS04]. For them, we have pre-computed makespan-

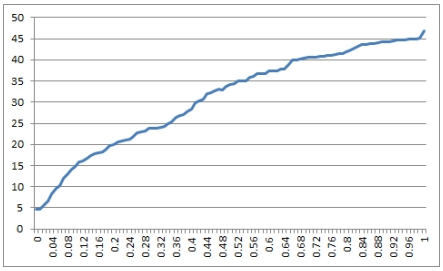

Fig. 2: Cost vs. Deadline

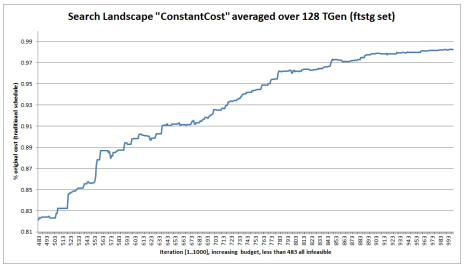

Fig. 3: Search Landscape

optimal schedules for systems with 2, 4, 8, and 16 PEs, leading to a total of 128 different test cases. All makespans were below 100 time slots. We call these schedules the set of *original schedules*. In all experiments involving the *heuristic*, we derive a new set of schedules from the original schedule set. As described in Sect. 4, these schedules have the same structure as the original schedule, but are frequency scaled. We set the deadline D of the original schedules to their respective (minimal) makespan. We further assume that D is not necessarily tight and may be extended which permits us to obtain additional cost savings. As in the rest of the paper, this results in a new deadline $D' \geq D$, which is then used in the frequency scaling process. We call the result of this process a *derived schedule*. In *mILP* calculation, we take the properties of the original schedule (task graph and number of PEs) and calculate a new *energy cost-optimal schedule* from them[4]. If we compare heuristic and mILP results, we also use the same extended deadline D' as deadline for the mILP. Otherwise the results were not comparable, because the tighter mILP schedule would force the mILP to use higher frequencies. We used actual energy market data taken from the European Power Exchange's web site [Exc16] for October 19, 2016. A 100 time slot price schedule was created by repeating that day's data prices. For some experiments, we used other pricing schedules. They are detailed in the experiment description.

We first investigated the *cost reduction* provided by the heuristic over the original schedule. We computed the cost of the original schedule, then created the derived schedule, and computed its cost. We used binary search and sweep mode. The heuristic usually required less than one second to find a solution. We computed the average reduction over all 128 test cases. We experimented with deadline extensions between 0 and 100% (Fig. 2, x axis is deadline extension, y cost reduction). Even when keeping the original deadline, we saved 4.6% cost. This is due to non-critical path tasks which often can be frequency-scaled down without affecting the makespan. Not surprisingly, the cost reduction curve is steeper for the first 15% to 20% of deadline extension, but keeps growing even for large extensions. Detail review showed that most of the saving was caused by using more time slots at slower frequency, but a notable amount was also attributable to optimizing each task's last time slot based on Eq. 3.

Then, we investigated the *search landscape*. We used the heuristic's sweep method to compute schedules with increasing budget per time slot. Let B_o be the original schedule's

[4] Note that in case of a solver timeout, the result is not necessarily optimal, but for reasons given in this section we assume it is very close to the optimum.

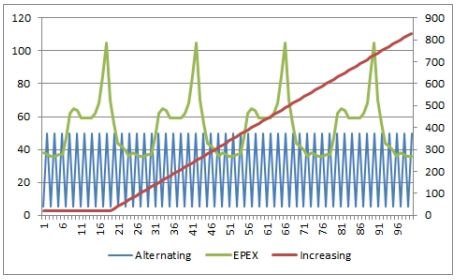

Fig. 4: Some Price Schedules

budget requirement. Then we derived 1000 schedules with increasing budget, where in step s we gave a budget permission of $\frac{s}{500}B_o$. For all feasible derived schedules, we computed the overall cost. Note that for the reasons outlined in Sect. 4 this cost was lower than B_o, even though the maximal budget permission was higher for the second half of the sweep. Fig. 3 plots the average landscape of all experiment runs. As can be seen, it is quite smooth. We also reviewed individual experiment data to verify the pattern and found them to follow the same pattern. This experiment was initially carried out with 20% deadline extension ($D' := 1.2D$). We repeated it with zero and 5 percent deadline extension and found the same structure. From that, we conclude that the landscape pattern is not sensitive to the amount of deadline extension. As a related sub-experiment, we have compared the heuristic's sweep and binary search modes. Both did always return results very close to each other. This also points towards a smooth landscape, as otherwise binary search would probably have selected a local minima in a high valley. Based on both of these results, we conclude that a heuristic can concentrate on finding an area close to the first local minimum and then focus on exploring its local neighborhood.

We compared the heuristic with the *mILP* implementation. We derived schedules with the heuristic with a 20% deadline extension and compared them to optimal cost schedules created with the same parameters by the mILP. We used IBM CPLEX 12.7 for the mILP implementation. The mILP had strongly varying run times depending on the task graph. Some experiments reached optimality within less than a minute, but most experiments timed out. We crafted an experiment that compared solution quality after 8 minute and 4 hour timeouts. Even after 4 hours about half of the experiments timed out. However, we only saw marginal improvements between the 8 minute and the 4 hour experiments (17.1% more cost for the heuristic vs. 17.6%). We suspect that this result is due to the smooth search landscape. In order to be able to cover more experiment scenarios, we limited our follow-on experiments to 8 minute timeout. We then carried out the same experiment with 5 and 50 percent deadline extension to check sensitivity to deadline extension. At 50% extension, we noticed that the mILP did perform notably worse than in the other two cases. This was caused by the model growth due to the larger number of time slots, which lead to fewer optimal solutions after 8 minutes. This also means that it will be practically impossible to increase the time resolution or compute longer running schedules, which strengthens the point that a heuristic solution is required. If we compare the experiments at 5 and 20 percent deadline extension, the heuristic requires a 15 respective 17 percent

higher cost than the mILP, which points to limited sensitivity to the amount of deadline extension. We also did a brief manual *review of mILP detail results* and got the impression that the solutions did considerably differ from the constant cost paradigm used inside the heuristic. An in-depth review will be done as future work to improve the heuristic.

To investigate the *power pricing schedule* effect, we did several experiments at 5% deadline extension with 8 minutes timeout. Some pricing schedules are depicted in Fig. 4. Again, we compared the heuristic with cost optimal solutions from the MILP. First of all, we assumed a traditional flat pricing model ($R = (1,1,\ldots)$). Here, the heuristic required 10% more cost than the mILP. Then we used a schedule ("Increasing") that stayed at flat pricing (absolute value 10) for the first $n = 20$ time slots, and then increased the price by 10 for each following time slots i ($R = (10,\ldots,10,20,30,\ldots)$). Here, the heuristic required 13% more cost. In one experiment, we used a pricing schedule ("Alternating") $R = (5,50,5,50,\ldots)$, specifically crafted to not work well with a constant cost approach. As expected, the heuristic performed notably worse with a 30% increase over mILP cost. In conclusion, the pricing schedule, except for very extreme cases, has moderate effect on optimality. We assume this is related to the precedence constraints which limit our ability to defer work, but need to investigate this in more depth.

Finally, we did one experiment in regard to static power consumption. We used $\text{power}(f) = 0.4 + 0.6f^3$ for the heuristic. We investigated the cost reduction provided by the heuristic over the original schedule. At 20% deadline extension, we only gained 4% reduction. We assume this is due to the limited ability to scale frequencies, as outlined in section 2. In future work, we will investigate static power consumption in more depth.

6 Conclusions

We have proposed to investigate cost-optimal static scheduling of task graphs on parallel machines with frequency scaling when the energy price varies over time. We have presented a mixed integer linear program and a heuristic for this problem, where the heuristic takes a classic, i.e. makespan-optimized, schedule as input and adapts frequencies and starting times. We have evaluated the heuristic with a set of small task graphs for two price curves and some corner cases and find that its solutions are on average 15% more expensive in energy cost than the optimal solution.

In our future work, we plan to extend both the linear program and the heuristic to cover more variations: the price curves could differ for different PEs (e.g. if the PEs are in different time zones), there might be several sources of energy with different cost, but some sources (like solar energy) might be restricted in volume per time slot, and the PEs could be heterogeneous in power consumption and performance. Also, the possibility to use time slots of different lengths to reduce the number of variables shall be tested. Also, we will investigate PE power curves which include static power, so that some power budget cannot run all PEs in a time slot (not even at the lowest frequency). Finally, we want to explore the possibility of a single heuristic to map tasks and scale frequencies.

References

[CGK10] Peter Cappers, Charles Goldman, and David Kathan. Demand response in US electricity markets: Empirical evidence. *Energy*, 35(4):1526–1535, 2010.

[DNT+14] Bernabé Dorronsoro, Sergio Nesmachnow, Javid Taheri, Albert Y Zomaya, El-Ghazali Talbi, and Pascal Bouvry. A hierarchical approach for energy-efficient scheduling of large workloads in multicore distributed systems. *Sustainable Computing: Informatics and Systems*, 4(4):252–261, 2014.

[EK17] Patrick Eitschberger and Jörg Keller. Fault-Tolerant Parallel Execution of Workflows with Deadlines. In *Proc. 25th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP 2017)*, 2017.

[Exc16] European Power Exchange. Market Data, Day Ahead Auction. `http://www.epexspot.com/en/market-data/dayaheadauction/chart/auction-chart/2016-10-19/DE/1d/200d`, 2016. Online, last access Nov 4, 2016.

[GLH+11] Íñigo Goiri, Kien Le, Md E Haque, Ryan Beauchea, Thu D Nguyen, Jordi Guitart, Jordi Torres, and Ricardo Bianchini. Greenslot: scheduling energy consumption in green datacenters. In *Proc. 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 20:1–20:11. ACM, 2011.

[GLLK79] Ronald L Graham, Eugene L Lawler, Jan Karel Lenstra, and AHG Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of discrete mathematics*, 5:287–326, 1979.

[HS04] Udo Hönig and Wolfram Schiffmann. A comprehensive test bench for the evaluation of scheduling heuristics. In *Proc. 16th IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS'04)*, pages 437–442, 2004.

[KA99] Yu-Kwong Kwok and Ishfaq Ahmad. Static Scheduling Algorithms for Allocating Directed Task Graphs to Multiprocessors. *ACM Comput. Surv.*, 31(4):406–471, December 1999.

[LOM17] Yunbo Li, Anne-Cécile Orgerie, and Jean-Marc Menaud. Balancing the use of batteries and opportunistic scheduling policies for maximizing renewable energy consumption in a Cloud data center. In *Proc. 25th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP 2017)*, 2017.

[LSK14] Jörg Lenhardt, Wolfram Schiffmann, and Jörg Keller. Trends in Static Power Consumption. In *Proc. 7th Swedish Workshop Multicore Computing*, 2014.

[LZL+15] Hongtao Lei, Tao Zhang, Yajie Liu, Yabing Zha, and Xiaomin Zhu. SGEESS: Smart green energy-efficient scheduling strategy with dynamic electricity price for data center. *Journal of Systems and Software*, 108:23 – 38, 2015.

[MLK+11] Yosef Masoudi, Shahriar Lotfi, Davod Karimzadgan, Farhad Fathy, and Kiomars Abdi. Static Task Graph Scheduling in Real Time Homogenous Multiprocessor Systems Using Learning Automata. In *Proc. 2011 International Conference on Communication Systems and Network Technologies (CSNT)*, pages 423–429. IEEE, 2011.

[PvSU08] Kirk Pruhs, Rob van Stee, and Patchrawat Uthaisombut. Speed scaling of tasks with precedence constraints. *Theory of Computing Systems*, 43(1):67–80, 2008.

[WLLMR14] Adam Wierman, Zhenhua Liu, Iris Liu, and Hamed Mohsenian-Rad. Opportunities and challenges for data center demand response. In *Proc. 2014 International Green Computing Conference (IGCC)*, pages 1–10. IEEE, 2014.