Algorithms for Energy Conservation in Heterogeneous Data Centers

Susanne Albers and Jens Quedenfeld^(✉)

Technical University of Munich, 85748 Garching, Germany
{albers,jens.quedenfeld}@in.tum.de

Abstract. Power consumption is the major cost factor in data centers. It can be reduced by dynamically right-sizing the data center according to the currently arriving jobs. If there is a long period with low load, servers can be powered down to save energy. For identical machines, the problem has already been solved optimally by [25] and [1].

In this paper, we study how a data-center with heterogeneous servers can dynamically be right-sized to minimize the energy consumption. There are d different server types with various operating and switching costs. We present a deterministic online algorithm that achieves a competitive ratio of $2d$ as well as a randomized version that is $1.58d$ -competitive. Furthermore, we show that there is no deterministic online algorithm that attains a competitive ratio smaller than $2d$. Hence our deterministic algorithm is optimal. In contrast to related problems like convex body chasing and convex function chasing [17,30], we investigate the discrete setting where the number of active servers must be an integral, so we gain truly feasible solutions.

1 Introduction

Energy management is an important issue in data centers. A huge amount of a data center's financial budget is spent on electricity that is needed to operate the servers as well as to cool them [12,20]. However, server utilization is typically low. In fact there are data centers where the average server utilization is as low as 12% [16]; only for a few days a year is full processing power needed. Unfortunately, idle servers still consume about half of their peak power [29]. Therefore, right-sizing a data center by powering down idle servers can save a significant amount of energy. However, shutting down a server and powering it up immediately afterwards incurs much more cost than holding the server in the active state during this time period. The cost for powering up and down does not only contain the increased energy consumption but also, for example, wear-and-tear costs or the risk that the server does not work properly after restarting [26]. Consequently, algorithms are needed that manage the number of active servers to minimize the total cost, without knowing when new jobs will arrive in the future. Since about 3% of the global electricity production

---

Work supported by the European Research Council, Grant Agreement No. 691672.

© The Author(s) 2021

T. Calamoneri and F. Corò (Eds.): CIAC 2021, LNCS 12701, pp. 75–89, 2021.

https://doi.org/10.1007/978-3-030-75242-2_5

is consumed by data centers [11], a reduction of their energy consumption can also decrease greenhouse emissions. Thus, right-sizing data centers is not only important for economical but also for ecological reasons.

Modern data centers usually contain heterogeneous servers. If the capacity of a data center is no longer sufficient, it is extended by including new servers. The old servers are still used however. Hence, there are different server types with various operating and switching costs in a data center. Heterogeneous data centers may also include different processing architectures. There can be servers that use GPUs to perform massive parallel calculations. However, GPUs are not suitable for all jobs. For example, tasks with many branches can be computed much faster on common CPUs than on GPUs [31].

Problem Formulation. We consider a data center with d different server types. There are m_j servers of type j . Each server has an active state where it is able to process jobs, and an inactive state where no energy is consumed. Powering up a server of type j (i.e., switching from the inactive into the active state) incurs a cost of β_j (called *switching cost*); powering down does not cost anything. We consider a finite time horizon consisting of the time slots $\{1, \dots, T\}$. For each time slot $t \in \{1, \dots, T\}$, jobs of total volume $\lambda_t \in \mathbb{N}_0$ arrive and have to be processed during the time slot. There must be at least λ_t active servers to process the arriving jobs. We consider a basic setting where the operating cost of a server of type j is load and time independent and denoted by $l_j \in \mathbb{R}_{\geq 0}$. Hence, an active server incurs a constant but type-dependent operating cost per time slot.

A schedule X is a sequence $\mathbf{x}_1, \dots, \mathbf{x}_T$ with $\mathbf{x}_t = (x_{t,1}, \dots, x_{t,d})$ where each $x_{t,j}$ indicates the number of active servers of type j during time slot t . At the beginning and the end of the considered time horizon all servers are shut down, i.e., $\mathbf{x}_0 = \mathbf{x}_{T+1} = (0, \dots, 0)$. A schedule is called *feasible* if there are enough active servers to process the arriving jobs and if there are not more active servers than available, i.e., $\sum_{j=1}^d x_{t,j} \geq \lambda_t$ and $x_{t,j} \in \{0, 1, \dots, m_j\}$ for all $t \in \{1, \dots, T\}$ and $j \in \{1, \dots, d\}$. The cost of a feasible schedule is defined by

$$C(X) := \sum_{t=1}^T \left(\sum_{j=1}^d l_j x_{t,j} + \sum_{j=1}^d \beta_j (x_{t,j} - x_{t-1,j})^+ \right) \quad (1)$$

where $(x)^+ := \max(x, 0)$. The switching cost is only paid for powering up. However, this is not a restriction, since all servers are inactive at the beginning and end of the workload. Thus the cost of powering down can be folded into the cost of powering up. A problem instance is specified by the tuple $\mathcal{I} = (T, d, \mathbf{m}, \boldsymbol{\beta}, \mathbf{l}, \Lambda)$ where $\mathbf{m} = (m_1, \dots, m_d)$, $\boldsymbol{\beta} = (\beta_1, \dots, \beta_d)$, $\mathbf{l} = (l_1, \dots, l_d)$ and $\Lambda = (\lambda_1, \dots, \lambda_T)$. The task is to find a schedule with minimum cost.

We focus on the central case without *inefficient* server types. A server type j is called *inefficient* if there is another server type $j' \neq j$ with both smaller (or equal) operating and switching costs, i.e., $l_j \geq l_{j'}$ and $\beta_j \geq \beta_{j'}$. This assumption is natural because a better server type with a lower operating cost usually has a higher switching cost. An inefficient server of type j is only powered up, if all servers of all types j' with $\beta_{j'} \leq \beta_j$ and $l_{j'} \leq l_j$ are already running. Therefore,

excluding inefficient servers is not a relevant restriction in practice. In related work, Augustine et al. [6] exclude inefficient states when operating a single server.

Our Contribution. We analyze the online setting of this problem where the job volumes λ_t arrive one-by-one. The vector of the active servers $\mathbf{x}_t$ has to be determined without knowledge of future jobs $\lambda_{t'}$ with $t' > t$. A main contribution of our work, compared to previous results, is that we investigate heterogeneous data centers and examine the online setting when truly feasible (integral) solutions are sought.

In Sect. 2, we present a $2d$ -competitive deterministic online algorithm, i.e., the total cost of the schedule calculated by our algorithm is at most $2d$ times larger than the cost of an optimal offline solution. Roughly, our algorithm works as follows. It calculates an optimal schedule for the jobs received so far and ensures that the operating cost of the active servers is at most as large as the operating cost of the active servers in the optimal schedule. If this is not the case, servers with high operating cost are replaced by servers with low operating cost. If a server is not used for a specific duration depending on its switching and operating costs, it is shut down.

In Sect. 3, we devise a randomized version of our algorithm achieving a competitive ratio of $\frac{e}{e-1}d \approx 1.582d$ against an oblivious adversary.

In Sect. 4, we show that there is no deterministic online algorithm that achieves a competitive ratio smaller than $2d$. Therefore, our algorithm is optimal. Additionally, for a data center that contains m unique servers (that is $m_j = 1$ for all $j \in \{1, \dots, d\}$), we show that the best achievable competitive ratio is $2m$.

Related Work. The design of energy-efficient algorithms has received quite some research interest over the last years, see e.g. [3, 10, 21] and references therein. Specifically, data center right-sizing has attracted considerable attention lately. Lin and Wierman [25, 26] analyzed the data-center right-sizing problem for data centers with identical servers ($d = 1$). The operating cost is load dependent and modeled by a convex function. In contrast to our setting, continuous solutions are allowed, i.e., the number of active server x_t can be fractional. This allows for other techniques in the design and analysis of an algorithm, but the created schedules cannot be used directly in practice. They gave a 3-competitive deterministic online algorithm for this problem. Bansal et al. [9] improved this result by randomization and developed a 2-competitive online algorithm. In our previous paper [1] we showed that 2 is a lower bound for randomized algorithms in the continuous setting; this result was independently shown by [4]. Furthermore, we analyzed the discrete setting of the problem where the number of active servers is integral ($x_t \in \mathbb{N}_0$). We presented a 3-competitive deterministic and a 2-competitive randomized online algorithm. Moreover, we proved that these competitive ratios are optimal.

Data-center right-sizing of heterogeneous data centers is related to convex function chasing, which is also known as smoothed online convex optimization [15]. At each time slot t , a convex function f_t arrives. The algorithm then has to choose a point $\mathbf{x}_t$ and pay the cost $f_t(\mathbf{x}_t)$ as well as the movement cost $\|\mathbf{x}_t - \mathbf{x}_{t-1}\|$ where $\|\cdot\|$ is any metric. The problem described by Eq. (1) is a

special case of convex function chasing if fractional schedules are allowed, i.e., $x_{t,j} \in [0, m_j]$ instead of $x_{t,j} \in \{0, \dots, m_j\}$. The operating cost $\sum_{j=1}^d l_j x_{t,j}$ in Eq. (1) together with the feasibility requirements can be modeled as a convex function that is infinite for $\sum_{j=1}^d x_{t,j} < \lambda_t$ and $x_{t,j} \notin [0, m_j]$. The switching cost equals the Manhattan metric if the number of servers is scaled appropriately. Sellke [30] gave a $(d+1)$ -competitive algorithm for convex function chasing. A similar result was found by Argue et al. [5].

In the discrete setting, convex function chasing has at least an exponential competitive ratio, as the following setting shows. Let $m_j = 1$ and $\beta_j = 1$ for all $j \in \{1, \dots, d\}$, so the possible server configurations are $\{0, 1\}^d$. The arriving convex functions f_t are infinite for the current position $\mathbf{x}_{t-1}$ of the online algorithm and 0 for all other positions $\{0, 1\}^d \setminus \{\mathbf{x}_{t-1}\}$. After $T := 2^d - 1$ functions arrived, the switching cost paid by the algorithm is at least $2^d - 1$ (otherwise it has to pay infinite operating costs), whereas the offline schedule can go directly to a position without any operating cost and only pays a switching cost of at most d .

Already for the 1-dimensional case (i.e. identical machines), it is not trivial to round a fractional schedule without increasing the competitive ratio (see [26] and [2]). In d -dimensional space, it is completely unclear, if continuous solutions can be rounded without arbitrarily increasing the total cost. Simply rounding up can lead to arbitrarily large switching costs, for example if the fractional solution rapidly switches between 1 and $1 + \epsilon$. Using a randomized rounding scheme like in [2] (that was used for homogeneous data centers) independently for each dimension can result in an infeasible schedule (for example, if $\lambda_t = 1$ and $\mathbf{x}_t = (1/d, \dots, 1/d)$ is rounded down to $(0, \dots, 0)$). Therefore, Sellke's result does not help us for analyzing the discrete setting. Other publications handling convex function chasing or convex body chasing are [8, 13, 17].

Goel and Wierman [19] developed a $(3 + \mathcal{O}(1/\mu))$ -competitive algorithm called Online Balanced Descent (OBD) for convex function chasing, where the arriving functions were required to be μ -strongly convex. We remark that the operating cost defined by Eq. (1) is not strongly convex, i.e., $\mu = 0$. Hence their result cannot be used for our problem. A similar result is given by Chen et al. [15] who showed that OBD is $(3 + \mathcal{O}(1/\alpha))$ -competitive if the arriving functions are locally α -polyhedral. In our case, $\alpha = \min_{j \in \{1, \dots, d\}} l_j / \beta_j$, so α can be arbitrarily small depending on the problem instance.

Another similar problem is the Parking Permit Problem by Meyerson [28]. There are d different permits which can be purchased for β_j dollars and have a duration of D_j days. Certain days are driving days where at least one parking permit is needed ($\lambda_t \in \{0, 1\}$). The permit cost corresponds to our switching cost. However, the duration of the permit is fixed to D_j , whereas in our problem the online algorithm can choose for each time slot if it wants to power down a server. Furthermore, there is no operating cost. Even if each server type is replaced by an infinite number of permits with the duration t and the cost $\beta_j + l_j \cdot t$, it is still a different problem, because the algorithm has to choose the time slot for powering down in advance (when the server is powered up).

Data-center right-sizing of heterogeneous data centers is related to geographical load balancing analyzed in [24] and [27]. Other applications are shown in [7, 14, 18, 22, 23, 32, 33].

2 Deterministic Online Algorithm

In this section we present a deterministic $2d$ -competitive online algorithm for the problem described in the preceding section. The basic idea of our algorithm is to calculate an optimal schedule for the problem instance that ends at the current time slot. Based on this schedule, we decide when a server is powered up. If a server is idle for a specific time, it is powered down.

Formally, given the original problem instance $\mathcal{I} = (T, d, \mathbf{m}, \beta, \mathbf{l}, \Lambda)$, the shortened problem instance $\mathcal{I}^t$ is defined by $\mathcal{I}^t := (t, d, \mathbf{m}, \beta, \mathbf{l}, \Lambda^t)$ with $\Lambda^t = (\lambda_1, \dots, \lambda_t)$. Let $\hat{X}^t$ denote an optimal schedule for $\mathcal{I}^t$ and let $X^{\mathcal{A}}$ be the schedule calculated by our algorithm $\mathcal{A}$.

W.l.o.g. there are no server types with the same operating and switching costs, i.e., $\beta_j = \beta_{j'}$ and $l_j = l_{j'}$ implies $j = j'$. Furthermore, let $l_1 > \dots > l_d$, i.e., the server types are sorted by their operating costs. Since inefficient server types are excluded, this implies that $\beta_1 < \dots < \beta_d$.

Let $[n] := \{1, \dots, n\}$ where $n \in \mathbb{N}$. We separate a problem instance into $m := \sum_{j=1}^d m_j$ lanes. At time slot t , there is a single job in lane $k \in [m]$, if and only if $k \leq \lambda_t$. We can assume that $\lambda_t \leq m$ holds for all $t \in [T]$, because otherwise there is no feasible schedule for the problem instance. Let X be an arbitrary feasible schedule with $\mathbf{x}_t = (x_{t,1}, \dots, x_{t,d})$. We define

$$y_{t,k} := \begin{cases} \max\{j \in [d] \mid \sum_{j'=j}^d x_{t,j'} \geq k\} & \text{if } k \in \left[\sum_{j=1}^d x_{t,j}\right] \\ 0 & \text{else} \end{cases} \quad (2)$$

to be the server type that handles the k -th lane during time slot t . If $y_{t,k} = 0$, then there is no active server in lane k at time slot t . By definition, the values $y_{t,1}, \dots, y_{t,m}$ are sorted in descending order, i.e., $y_{t,k} \geq y_{t,k'}$ for $k < k'$. Note that $y_{t,k} = 0$ implies $\lambda_t < k$, because otherwise there are not enough active servers to handle the jobs at time t . For the schedule $\hat{X}^t$, the server type used in lane k at time slot t' is denoted by $\hat{y}_{t',k}^t$. Our algorithm calculates $y_{t,k}^{\mathcal{A}}$ directly, the corresponding variables $x_{t,j}^{\mathcal{A}}$ can be determined by $x_{t,j}^{\mathcal{A}} = |\{k \in [m] \mid y_{t,k}^{\mathcal{A}} = j\}|$.

Our algorithm works as follows: First, an optimal solution $\hat{X}^t$ is calculated. If there are several optimal schedules, we choose a schedule that fulfills the inequality $\hat{y}_{t',k}^t \geq \hat{y}_{t',k}^{t-1}$ for all time slots $t' \in [t]$ and lanes $k \in [m]$, so $\hat{X}^t$ never uses smaller server types than the previous schedule $\hat{X}^{t-1}$. We will see in Lemma 2 that such a schedule exists and how to construct it.

If there is a server type j with $l_j = 0$, then in an optimal schedule such a server can be powered up before it is needed, although $\lambda_t = 0$ holds for this time slot. Similarly, such a server can run for more time slots than necessary. W.l.o.g. let $\hat{X}^t$ be a schedule where servers are powered up as late as possible and powered down as early as possible.

Beginning from the lowest lane ($k = 1$), it is ensured that $\mathcal{A}$ uses a server type that is not smaller than the server type used by $\hat{X}^t$, i.e., $y_{t,k}^{\mathcal{A}} \geq \hat{y}_{t,k}^t$ must be fulfilled. If the server type $y_{t-1,k}^{\mathcal{A}}$ used in the previous time slot is smaller than $\hat{y}_{t,k}^t$, it is powered down and server type $\hat{y}_{t,k}^t$ is powered up. A server of type j that is not replaced by a greater server type stays active for $\bar{t}_j := \lfloor \beta_j / l_j \rfloor$ time slots. If $\hat{X}^t$ uses a smaller server type $j' \leq j$ in the meantime, then server type j will run for at least $\bar{t}_{j'}$ further time slots (including time slot t). Formally, a server of type j in lane k is powered down at time slot t , if $\hat{y}_{t',k}^{t'} \neq j'$ holds for all server types $j' \leq j$ and time slots $t' \in [t - \bar{t}_{j'} + 1 : t]$ with $[a : b] := \{a, a + 1, \dots, b\}$.

The pseudocode below clarifies how algorithm $\mathcal{A}$ works. The variables e_k for $k \in [m]$ store the time slot when the server in the corresponding lane will be powered down.

Algorithm 1. Algorithm $\mathcal{A}$

```

1: for  $t := 1$  to  $T$  do
2:   Calculate  $\hat{X}^t$  such that  $\hat{y}_{t',k}^t \geq \hat{y}_{t',k}^{t-1}$  for all  $t' \in [t]$  and  $k \in [m]$ 
3:   for  $k := 1$  to  $m$  do
4:     if  $y_{t-1,k}^{\mathcal{A}} < \hat{y}_{t,k}^t$  or  $t \geq e_k$  then
5:        $y_{t,k}^{\mathcal{A}} := \hat{y}_{t,k}^t$ 
6:        $e_k := t + \bar{t}_{y_{t,k}^{\mathcal{A}}}$ 
7:     else
8:        $y_{t,k}^{\mathcal{A}} := y_{t-1,k}^{\mathcal{A}}$ 
9:        $e_k := \max\{e_k, t + \bar{t}_{\hat{y}_{t,k}^t}\}$  where  $\bar{t}_0 := 0$ 

```

Structure of Optimal Schedules. Before we can analyze the competitiveness of algorithm $\mathcal{A}$, we have to show that an optimal schedule with the desired properties required by line 2 actually exists. First, we will investigate basic properties of optimal schedules. In an optimal schedule $\hat{X}$, a server of type j that runs in lane k does not change the lane while running. Formally, if $\hat{y}_{t-1,k} = j$ and $\hat{y}_{t,k} \neq j$, then there exists no other lane $k' \neq k$ with $\hat{y}_{t-1,k'} \neq j$ and $\hat{y}_{t,k'} = j$. Furthermore, a server is only powered up or powered down if the number of jobs is increased or decreased, respectively. Finally, in a given lane k , the server type does not change immediately, i.e., there must be at least one time slot, where no server is running in lane k . These properties are proven in the full version of this paper.

Given the optimal schedules $\hat{X}^u$ and $\hat{X}^v$ with $u < v$, we construct a *minimum* schedule $X^{\min(u,v)}$ with $y_{t,k}^{\min(u,v)} := \min\{\hat{y}_{t,k}^u, \hat{y}_{t,k}^v\}$. Furthermore, we construct a *maximum* schedule $X^{\max(u,v)}$ as follows. Let $z_l(t, k)$ be the last time slot $t' < t$ with $\hat{y}_{t',k}^u = \hat{y}_{t',k}^v = 0$ (no active servers in both schedules) and let $z_r(t, k)$ be the first time slot $t' > t$ with $\hat{y}_{t',k}^u = \hat{y}_{t',k}^v = 0$. The schedule $X^{\max(u,v)}$ is defined by

$$y_{t,k}^{\max(u,v)} := \max_{t' \in [z_l(t,k)+1 : z_r(t,k)-1]} \{\hat{y}_{t',k}^u, \hat{y}_{t',k}^v\}. \quad (3)$$

Another way to construct $X^{\max(u,v)}$ is as follows. First, we take the maximum of both schedules (analogously to $X^{\min(u,v)}$). However, this can lead to situations where the server type changes immediately, so the necessary condition for optimal schedules would not be fulfilled. Therefore, we replace the lower server type by the greater one until there are no more immediate server changes. This construction is equivalent to Eq. (3).

We will see in Lemma 2 that the *maximum* schedule is an optimal schedule for $\mathcal{I}^v$ and fulfills the property required by algorithm $\mathcal{A}$ in line 2, which says that the server type used in lane k at time t never decreases when the considered problem instance is expanded. To prove this property, first we have to show that $X^{\min(u,v)}$ and $X^{\max(u,v)}$ are feasible schedules for the problem instances $\mathcal{I}^u$ and $\mathcal{I}^v$, respectively.

Lemma 1. $X^{\min(u,v)}$ and $X^{\max(u,v)}$ are feasible for $\mathcal{I}^u$ and $\mathcal{I}^v$, respectively.

The proof can be found in the full version of this paper. Now, we are able to show that the maximum schedule is optimal for the problem instance $\mathcal{I}^v$.

Lemma 2. Let $u, v \in [T]$ with $u < v$. $X^{\max(u,v)}$ is optimal for $\mathcal{I}^v$.

The works roughly as follows (the complete proof can be found in the full paper). First, we prove that the sum of the operating costs of $\hat{X}^u$ and $\hat{X}^v$ is greater than or equal to the sum of the operating cost of $X^{\min(u,v)}$ and $X^{\max(u,v)}$. Each server activation in $X^{\min(u,v)}$ and $X^{\max(u,v)}$ can be mapped to exactly one server activation in $\hat{X}^u$ and $\hat{X}^v$ with the same or a greater server type. Therefore, $C(X^{\min(u,v)}) + C(X^{\max(u,v)}) \leq C(\hat{X}^u) + C(\hat{X}^v)$ holds and by using Lemma 1, it is shown that $X^{\max(u,v)}$ is optimal for $\mathcal{I}^v$.

Feasibility. In the following, let $\{\hat{X}^1, \dots, \hat{X}^T\}$ be optimal schedules that fulfill the inequality $\hat{y}_{t',k}^t \geq \hat{y}_{t',k}^{t-1}$ for all $t, t' \in [T]$ and $k \in [m]$ as required by algorithm $\mathcal{A}$. Lemma 2 ensures that such a schedule sequence exists (and also shows how to construct it). Before we can prove that algorithm $\mathcal{A}$ is $2d$ -competitive, we have to show that the computed schedule $X^{\mathcal{A}}$ is feasible. In an optimal schedule $\hat{X}^t$, the values $\hat{y}_{t',1}^t, \dots, \hat{y}_{t',m}^t$ are sorted in descending order by definition. This also holds for schedule calculated by our algorithm.

Lemma 3. For all time slots $t \in [T]$, the values $y_{t,1}^{\mathcal{A}}, \dots, y_{t,m}^{\mathcal{A}}$ are sorted in descending order, i.e., $y_{t,k}^{\mathcal{A}} \geq y_{t,k'}^{\mathcal{A}}$ for $k < k'$.

The proof uses the fact that the running times $\bar{t}_j$ are sorted in ascending order, i.e., $\bar{t}_1 \leq \dots \leq \bar{t}_d$, because $l_1 > \dots > l_d$ and $\beta_1 < \dots < \beta_d$. In other words, the higher the server type is, the longer it stays in the active state. See the full paper for more details. By means of Lemma 3, we are able to prove the feasibility of $X^{\mathcal{A}}$.

Lemma 4. The schedule $X^{\mathcal{A}}$ is feasible.

Proof Idea. A schedule is feasible, if (1) there are enough active servers to handle the incoming jobs (i.e., $\sum_{j=1}^d x_{t,j}^{\mathcal{A}} \geq \lambda_t$) and (2) there are not more active servers

than available (i.e., $x_{t,j}^A \leq m_j$). The first property directly follows from the definition of algorithm $\mathcal{A}$, since $\sum_{j=1}^d x_{t,j}^A \geq \sum_{j=1}^d \hat{x}_{t,j}^t \geq \lambda_t$. Lemma 3 is used to prove that $x_{t,j}^A \leq m_j$ is always fulfilled after setting $y_{t,k}^A$ in line 5 or 8. The complete proof is presented in the full paper. $\square$

Competitiveness. To show the competitiveness of $\mathcal{A}$, we divide the schedule X^A into blocks $A_{t,k}$ with $t \in [T]$ and $k \in [m]$. Each block $A_{t,k}$ is described by its creation time t , its start time $s_{t,k}$, its end time $e_{t,k}$, the used server type $j_{t,k}$ and the corresponding lane k . The start time is the time slot when $j_{t,k}$ is powered up and the end time is the first time slot, when $j_{t,k}$ is inactive, i.e., during the time interval $[s_{t,k} : e_{t,k} - 1]$ the server of type $j_{t,k}$ is in the active state.

There are two types of blocks: *new* blocks and *extended* blocks. A *new* block starts when a new server is powered up, i.e., lines 5 and 6 of algorithm $\mathcal{A}$ are executed because $y_{t-1,k}^A < \hat{y}_{t,k}^t$ or $t \geq e_k \wedge y_{t-1,k}^A > \hat{y}_{t,k}^t \wedge \hat{y}_{t,k}^t > 0$ (in words: the previous block ends and $\hat{X}^t$ has an active server in lane k , but the server type is smaller than the server type used by $\mathcal{A}$ in the previous time slot). It ends after $\bar{t}_{y_{t,k}^A}$ time slots. Thus $s_{t,k} := t$ and $e_{t,k} := t + \bar{t}_{y_{t,k}^A}$ (i.e., $e_{t,k}$ equals e_k after executing line 6).

An *extended* block is created when the running time of a server is extended, i.e., the value of e_k is updated, but the server type remains the same (that is $y_{t-1,k}^A = y_{t,k}^A$). We have $e_{t,k} := t + \bar{t}_{\hat{y}_{t,k}^t}$ (i.e., the value of e_k after executing line 9 or 6) and $s_{t,k} := e_{t',k}$, where $A_{t',k}$ is the previous block in the same lane. Note that an *extended* block can be created not only in line 9, but also in line 6, if $t = e_k$ and $y_{t-1,k}^A = \hat{y}_{t,k}^t$. If line 8 and 9 are executed, but the value of e_k does not change (because $t + \bar{t}_{\hat{y}_{t,k}^t}$ is smaller than or equal to the previous value of e_k), then the block $A_{t,k}$ does not exist.

Let $d_{t,k} := e_{t,k} - s_{t,k}$ be the duration of the block $A_{t,k}$ and let $C(A_{t,k})$ be the cost caused by $A_{t,k}$ if the block $A_{t,k}$ exists or 0 otherwise. The next lemma describes how the cost of a block can be estimated.

Lemma 5. *The cost of the block $A_{t,k}$ is upper bounded by*

$$C(A_{t,k}) \leq \begin{cases} 2\beta_{j_{t,k}} & \text{if } A_{t,k} \text{ is a new block} \\ l_{j_{t,k}} d_{t,k} & \text{if } A_{t,k} \text{ is an extended block} \end{cases} \quad (4)$$

The lemma follows from the definition of $\bar{t}_j$ (see the full paper for more details). To show the competitiveness of algorithm $\mathcal{A}$, we introduce another variable that will be used in Lemmas 7 and 8. Let

$$\tilde{y}_{t,k}^u := \max_{t' \in [t:u]} \hat{y}_{t',k}^{t'}$$

be the largest server type used in lane k by the schedule $\hat{X}^{t'}$ at time slot t' for $t' \in [t : u]$. The next lemma shows that $\tilde{y}_{t,k}^u$ is monotonically decreasing with respect to t as well as k and increasing with respect to u .

Lemma 6. *Let $u' \geq u$, $t' \leq t$ and $k' \leq k$. It is $\tilde{y}_{t,k}^u \leq \tilde{y}_{t',k'}^{u'}$.*

This lemma follows from the definition of $\tilde{y}_{t,k}^u$. A proof can be found in the full paper. The cost of schedule X in lane k during time slot t is denoted by

$$C_{t,k}(X) := \begin{cases} l_{y_{t,k}} + \beta_{y_{t,k}} & \text{if } y_{t-1,k} \neq y_{t,k} > 0 \\ l_{y_{t,k}} & \text{if } y_{t-1,k} = y_{t,k} > 0 \\ 0 & \text{otherwise.} \end{cases} \quad (5)$$

The total cost of X can be written as $C(X) = \sum_{t=1}^T \sum_{k=1}^m C_{t,k}(X)$. The technical lemma below will be needed for our induction proof in Theorem 1. Given the optimal schedules $\hat{X}^u$ and $\hat{X}^v$ with $u < v$, the inequality $\sum_{k=1}^m \sum_{t=1}^u C_{t,k}(\hat{X}^u) \leq \sum_{k=1}^m \sum_{t=1}^v C_{t,k}(\hat{X}^v)$ is obviously fulfilled (because $\hat{X}^u$ is an optimal schedule for $\mathcal{I}^u$, so $\hat{X}^v$ cannot be better). The lemma below shows that this inequality still holds if the cost $C_{t,k}(\cdot)$ is scaled by $\tilde{y}_{t,k}^u$.

Lemma 7. *Let $u, v \in [T]$ with $u < v$. It holds that*

$$\sum_{k=1}^m \sum_{t=1}^u \tilde{y}_{t,k}^u C_{t,k}(\hat{X}^u) \leq \sum_{k=1}^m \sum_{t=1}^v \tilde{y}_{t,k}^u C_{t,k}(\hat{X}^v). \quad (6)$$

The proof is shown in the full paper. The next lemma shows how the cost of a single block $A_{v,k}$ can be folded into the term $2 \sum_{t=1}^{v-1} \tilde{y}_{t,k}^{v-1} C_{t,k}(\hat{X}^v)$ which is the right hand side of Eq. (6) given in the previous lemma with $u = v - 1$.

Lemma 8. *For all lanes $k \in [m]$ and time slots $v \in [T]$, it is*

$$2 \sum_{t=1}^{v-1} \tilde{y}_{t,k}^{v-1} C_{t,k}(\hat{X}^v) + C(A_{v,k}) \leq 2 \sum_{t=1}^v \tilde{y}_{t,k}^v C_{t,k}(\hat{X}^v). \quad (7)$$

Proof. If the block $A_{v,k}$ does not exist, Eq. (7) holds by Lemma 6 and $C(A_{v,k}) = 0$.

If $A_{v,k}$ is a *new* block, then $C(A_{v,k}) \leq 2\beta_j$ with $j := j_{v,k} = \hat{y}_{v,k}^v$ by Lemma 5. Since $A_{v,k}$ is a *new* block, server type j was not used in the last time slot of the last $\bar{t}_j$ schedules, i.e., $\hat{y}_{t,k}^t \leq j - 1$ for $t \in [v - \bar{t}_j : v - 1]$. If $\hat{y}_{v-\bar{t}_j,k}^{v-\bar{t}_j} = j$ would hold, then $y_{v-1,k}^A = j$ and there would be an *extended* block at time slot v . By using the facts above and the definition of $\tilde{t}_{t,k}^v$, for $t \in [v - \bar{t}_j : v - 1]$, we get

$$\tilde{y}_{t,k}^{v-1} = \max_{t' \in [t:v-1]} \hat{y}_{t',k}^{t'} \leq j - 1 = \hat{y}_{v,k}^v - 1 \leq \max_{t' \in [t:v]} \hat{y}_{t',k}^{t'} - 1 = \tilde{y}_{t,k}^v - 1. \quad (8)$$

By using Lemma 6 and Eq. (8), we can estimate the first sum in (7):

$$\begin{aligned} \sum_{t=1}^{v-1} \tilde{y}_{t,k}^{v-1} C_{t,k}(\hat{X}^v) &\stackrel{L6,(8)}{\leq} \sum_{t=1}^{v-\bar{t}_j-1} \tilde{y}_{t,k}^v C_{t,k}(\hat{X}^v) + \sum_{t=v-\bar{t}_j}^{v-1} (\tilde{y}_{t,k}^v - 1) C_{t,k}(\hat{X}^v) \\ &\leq \sum_{t=1}^v \tilde{y}_{t,k}^v C_{t,k}(\hat{X}^v) - \beta_j. \end{aligned} \quad (9)$$

For the second inequality, we add $(\tilde{y}_{v,k}^v - 1) \cdot C_{v,k}(\hat{X}^v) \geq 0$ and use $\sum_{t=v-\bar{t}_j}^v C_{t,k}(\hat{X}^v) \geq \beta_j$ which holds because either j was powered up in $\hat{X}^v$ during $[v - \bar{t}_j : v]$ (then there is the switching cost of β_j) or j runs for $\bar{t}_j + 1$ time slots resulting in an operating cost of $l_j \cdot (\bar{t}_j + 1) = l_j \cdot (\lfloor \beta_j / l_j \rfloor + 1) \geq \beta_j$. Altogether, we get (beginning from the left hand side of Eq. (7) that has to be shown)

$$\begin{aligned} 2 \sum_{t=1}^{v-1} \tilde{y}_{t,k}^{v-1} C_{t,k}(\hat{X}^v) + C(A_{v,k}) &\stackrel{(9), L5}{\leq} 2 \sum_{t=1}^v \tilde{y}_{t,k}^v C_{t,k}(\hat{X}^v) - 2\beta_j + 2\beta_j \\ &\leq 2 \sum_{t=1}^v \tilde{y}_{t,k}^v C_{t,k}(\hat{X}^v). \end{aligned}$$

If $A_{v,k}$ is an *extended* block, the proof of Eq. (7) is quite similar (see the full version of this paper for more details). $\square$

Theorem 1. *Algorithm $\mathcal{A}$ is $2d$ -competitive.*

Proof. The feasibility of $X^{\mathcal{A}}$ was already proven in Lemma 4, so we have to show that $C(X^{\mathcal{A}}) \leq 2d \cdot C(\hat{X}^T)$. Let $C_v(X^{\mathcal{A}}) := \sum_{t=1}^v \sum_{k=1}^m C(A_{t,k})$ denote the cost of algorithm $\mathcal{A}$ up to time slot v . We will show by induction that

$$C_v(X^{\mathcal{A}}) \leq 2 \sum_{k=1}^m \sum_{t=1}^v \tilde{y}_{t,k}^v C_{t,k}(\hat{X}^v) \quad (10)$$

holds for all $v \in [T]_0$.

For $v = 0$, we have no costs for both $X^{\mathcal{A}}$ and $\hat{X}^v$, so inequality (10) is fulfilled. Assume that inequality (10) holds for $v - 1$. By using the induction hypothesis as well as Lemmas 7 and 8, we get

$$\begin{aligned} C_v(X^{\mathcal{A}}) &= C_{v-1}(X^{\mathcal{A}}) + \sum_{k=1}^m C(A_{v,k}) \\ &\stackrel{\text{I.H.}}{\leq} 2 \sum_{k=1}^m \sum_{t=1}^{v-1} \tilde{y}_{t,k}^{v-1} C_{t,k}(\hat{X}^{v-1}) + \sum_{k=1}^m C(A_{v,k}) \\ &\stackrel{L7, L8}{\leq} 2 \sum_{k=1}^m \sum_{t=1}^v \tilde{y}_{t,k}^v C_{t,k}(\hat{X}^v). \end{aligned} \quad (11)$$

Since $\tilde{y}_{t,k}^v \leq d$, we get

$$C_T(X^{\mathcal{A}}) \stackrel{(11)}{\leq} 2 \sum_{k=1}^m \sum_{t=1}^T \tilde{y}_{t,k}^T C_{t,k}(\hat{X}^T) \leq 2d \sum_{k=1}^m \sum_{t=1}^T C_{t,k}(\hat{X}^T) \leq 2d \cdot C(\hat{X}^T).$$

The schedule $\hat{X}^T$ is optimal for the problem instance $\mathcal{I}$, so algorithm $\mathcal{A}$ is $2d$ -competitive. $\square$

3 Randomized Online Algorithm

The $2d$ -competitive algorithm can be randomized to achieve a competitive ratio of $\frac{e}{e-1}d \approx 1.582d$ against an oblivious adversary. The randomized algorithm $\mathcal{B}$ chooses $\gamma \in [0, 1]$ according to the probability density function $f_\gamma(x) = e^x/(e-1)$ for $x \in [0, 1]$. The variables $\bar{t}_j$ are set to $\lfloor \gamma \cdot \beta_j / l_j \rfloor$, so the running time of a server is randomized. Then, algorithm $\mathcal{A}$ is executed. Note that γ is determined at the beginning of the algorithm and not for each block.

Theorem 2. *Algorithm $\mathcal{B}$ is $\frac{e}{e-1}d$ -competitive against an oblivious adversary.*

The complete proof of this theorem is shown in the full paper. Most lemmas introduced in the previous section still hold, because they do not depend on the exact value of $\bar{t}_j$, only Lemmas 5 and 8 have to be adapted. For the proof of Theorem 2, we first give an upper bound for the expected cost of block $A_{t,k}$ (replacing Lemma 5). This bound is used to show that

$$\frac{e}{e-1} \cdot \sum_{t=1}^{v-1} \bar{y}_{t,k}^{v-1} C_{t,k}(\hat{X}^v) + \mathbb{E}[C(A_{v,k})] \leq \frac{e}{e-1} \cdot \sum_{t=1}^v \bar{y}_{t,k}^v C_{t,k}(\hat{X}^v)$$

holds for all lanes $k \in [m]$ and time slots $v \in [T]$ (similar to Lemma 8). Finally, Theorem 2 is proven by induction.

4 Lower Bound

In this section, we show that there is no deterministic online algorithm that achieves a competitive ratio that is better than $2d$.

We consider the following problem instance: Let $\beta_j := N^{2j}$ and $l_j := 1/N^{2j}$ where N is a sufficiently large number that depends on the number of servers types d . The value of N will be determined later. The adversary will send a job for the current time slot if and only if the online algorithm has no active server during the previous time slot. This implies that the online algorithm has to power up a server immediately after powering down any server. Note that $\lambda_t \in \{0, 1\}$, i.e., it is never necessary to power up more than one server. The optimal schedule is denoted by X^* . Let $\mathcal{A}$ be an arbitrary deterministic online algorithm and let $X^{\mathcal{A}}$ be the schedule computed by $\mathcal{A}$.

W.l.o.g. in $X^{\mathcal{A}}$ there is no time slot with more than one active server. If this were not the case, we could easily convert the schedule into one where the assumption holds without increasing the cost. Assume that at time slot t a new server of type k is powered up such that there are (at least) two active servers at time t . If we power up the server at $t+1$, the schedule is still feasible, but the total costs are reduced by l_k . We can repeat this procedure until there is at most one active server for each time slot.

Lemma 9. *Let $k \in [d]$. If $X^{\mathcal{A}}$ only uses servers of type lower than or equal to k and if the cost of $\mathcal{A}$ is at least $C(X^{\mathcal{A}}) \geq N\beta_k$, then the cost of $\mathcal{A}$ is at least*

$$C(X^{\mathcal{A}}) \geq (2k - \epsilon_k) \cdot C(X^*) \quad (12)$$

with $\epsilon_k = 9k^2/N$ and $N \geq 6k$.

Proof Idea. We will prove the lemma by induction. The base case $k = 1$ is shown in the full version of this paper, so we assume that Lemma 9 holds for $k - 1$.

We divide the schedule $X^{\mathcal{A}}$ into phases $L_0, K_1, L_1, K_2, \dots, L_n$ such that in the phases $K_1, \dots, K_n$ server type k is used exactly once, while in the intermediate phases $L_0, \dots, L_n$ the other server types $1, \dots, k - 1$ are used. A phase K_i begins when a server of type k is powered up and ends when it is powered down. The phases L_i can have zero length (if the server type k is powered up immediately after it is powered down, so between K_i and K_{i+1} an empty phase L_i is inserted).

The operating cost during phase K_i is denoted by $\delta_i \beta_k$. The operating and switching costs during phase L_i are denoted by $p_i \beta_k$. We divide the intermediate phases L_i into long phases where $p_i > 1/N$ holds and short phases where $p_i \leq 1/N$. Note that we can use the induction hypothesis only for long phases. The index sets of the long and short phases are denoted by $\mathcal{L}$ and $\mathcal{S}$, respectively.

To estimate the cost of an optimal schedule we consider two strategies: In the first strategy, a server of type k is powered up at the first time slot and runs for the whole time except for phases K_i with $\delta_i > 1$, then powering down and powering up are cheaper than keeping the server in the active state (β_k vs. $\delta_i \beta_k$). The operating cost for the phases K_i is $\delta_i^* \beta_k$ with $\delta_i^* := \min\{1, \delta_i\}$ and the operating cost for the phases L_i is at most $\frac{1}{N^2} p_i \beta_k$, because algorithm $\mathcal{A}$ uses servers whose types are lower than k and therefore the operating cost of $\mathcal{A}$ is at least N^2 times larger. Thus, the total cost of this strategy is at most

$$\beta_k \left(1 + \sum_{i=1}^n \delta_i^* + \sum_{i \in \mathcal{L} \cup \mathcal{S}} \frac{1}{N^2} p_i \right) \geq C(X^*).$$

In the second strategy, for the long phases L we use the strategy given by our induction hypothesis, while for the short phases S we behave like algorithm $\mathcal{A}$ and in the phases K_i we run the server type 1 for exactly one time slot (note that in K_i we only have $\lambda_t = 1$ in the first time slot of the phase). Therefore the total cost is upper bounded by

$$\beta_k \left(\sum_{i \in \mathcal{L}} \frac{1}{\alpha} p_i + \sum_{i \in \mathcal{S}} p_i + 2n \beta_1 / \beta_k \right) \geq C(X^*)$$

with $\alpha := 2k - 2 - \epsilon_{k-1}$.

The total cost of $\mathcal{A}$ is equal to $\beta_k (\sum_{i=1}^n (1 + \delta_i) + \sum_{i \in \mathcal{L} \cup \mathcal{S}} p_i)$, so the competitive ratio is given by

$$\frac{C(X^{\mathcal{A}})}{C(X^*)} \geq \frac{\sum_{i=1}^n (1 + \delta_i) + \sum_{i \in \mathcal{L} \cup \mathcal{S}} p_i}{C(X^*) / \beta_k}.$$

By cleverly separating the nominator into two terms and by estimating $C(X^*)$ with strategy 1 and 2, respectively, it can be shown that $\frac{C(X^{\mathcal{A}})}{C(X^*)} \geq 2 + \alpha - \frac{16k}{N} \geq 2k - \epsilon_k$. The complete calculation including all intermediate steps is shown in the full paper. $\square$

Theorem 3. *There is no deterministic online algorithm for the data-center optimization problem with heterogeneous servers and time and load independent operating costs whose competitive ratio is smaller than $2d$.*

Proof Idea. Assume that there is an $(2d - \epsilon)$ -competitive deterministic online algorithm $\mathcal{A}$. We construct a workload as described at the beginning of this section until the cost of $\mathcal{A}$ is greater than $N\beta_d$ (note that $l_j > 0$ for all $j \in [d]$, so the cost of $\mathcal{A}$ can be arbitrarily large). By using Lemma 9 with $k = d$ and $N := \max\{6d, \lceil 9k^2/\epsilon + 1 \rceil\}$, we get $C(X^{\mathcal{A}}) > (2d - \epsilon) \cdot C(X^*)$ which is a contradiction to our assumption. See the full paper for more details. $\square$

The schedule constructed for the lower bound only uses at most one job in each time slot, so there is no reason for an online algorithm to utilize more than one server of a specific type. Thus, for a data center with m unique servers (i.e. $m_j = 1$ for all $j \in [d]$), the best achievable competitive ratio is $2d = 2m$.

References

1. Albers, S., Quedenfeld, J.: Optimal algorithms for right-sizing data centers. In: Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures, pp. 363–372. ACM (2018)
2. Albers, S., Quedenfeld, J.: Optimal algorithms for right-sizing data centers—extended version. arXiv preprint [arXiv:1807.05112](https://arxiv.org/abs/1807.05112) (2018)
3. Antoniadis, A., Garg, N., Kumar, G., Kumar, N.: Parallel machine scheduling to minimize energy consumption. In: Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 2758–2769. SIAM (2020)
4. Antoniadis, A., Schewior, K.: A tight lower bound for online convex optimization with switching costs. In: Solis-Oba, R., Fleischer, R. (eds.) WAOA 2017. LNCS, vol. 10787, pp. 164–175. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-89441-6_13
5. Argue, C., Gupta, A., Guruganesh, G., Tang, Z.: Chasing convex bodies with linear competitive ratio. In: Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1519–1524. SIAM (2020)
6. Augustine, J., Irani, S., Swamy, C.: Optimal power-down strategies. SIAM J. Comput. **37**(5), 1499–1516 (2008)
7. Badiei, M., Li, N., Wierman, A.: Online convex optimization with ramp constraints. In: 2015 54th IEEE Conference on Decision and Control (CDC), pp. 6730–6736. IEEE (2015)
8. Bansal, N., Böhm, M., Eliáš, M., Koumoutsos, G., Umboh, S.W.: Nested convex bodies are chaseable. In: Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1253–1260. SIAM (2018)

9. Bansal, N., Gupta, A., Krishnaswamy, R., Pruhs, K., Schewior, K., Stein, C.: A 2-competitive algorithm for online convex optimization with switching costs. In: LIPIcs-Leibniz International Proceedings in Informatics, vol. 40. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2015)
10. Bansal, N., Kimbrel, T., Pruhs, K.: Speed scaling to manage energy and temperature. *J. ACM (JACM)* **54**(1), 1–39 (2007)
11. Bawden, T.: Global warming: data centres to consume three times as much energy in next decade, experts warn (2016). <http://www.independent.co.uk/environment/global-warming-data-centres-to-consume-three-times-as-much-energy-in-next-decade-experts-warn-a6830086.html>
12. Brill, K.G.: The invisible crisis in the data center: the economic meltdown of Moore’s law. White paper, Uptime Institute, pp. 2–5 (2007)
13. Bubeck, S., Klartag, B., Lee, Y.T., Li, Y., Sellke, M.: Chasing nested convex bodies nearly optimally. In: Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1496–1508. SIAM (2020)
14. Chen, N., Agarwal, A., Wierman, A., Barman, S., Andrew, L.L.: Online convex optimization using predictions. In: ACM SIGMETRICS Performance Evaluation Review, vol. 43, pp. 191–204. ACM (2015)
15. Chen, N., Goel, G., Wierman, A.: Smoothed online convex optimization in high dimensions via online balanced descent. *Proc. Mach. Learn. Res.* **75**, 1574–1594 (2018)
16. Delforge, P., et al.: Data center efficiency assessment (2014). <https://www.nrdc.org/sites/default/files/data-center-efficiency-assessment-IP.pdf>
17. Friedman, J., Linial, N.: On convex body chasing. *Discrete Comput. Geom.* **9**(1), 293–321 (1993). <https://doi.org/10.1007/BF02189324>
18. Goel, G., Chen, N., Wierman, A.: Thinking fast and slow: optimization decomposition across timescales. In: 2017 IEEE 56th Annual Conference on Decision and Control (CDC), pp. 1291–1298. IEEE (2017)
19. Goel, G., Wierman, A.: An online algorithm for smoothed regression and LQR control. *Proc. Mach. Learn. Res.* **89**, 2504–2513 (2019)
20. Hamilton, J.: Cost of power in large-scale data centers (2008). <http://perspectives.mvdirona.com/2008/11/cost-of-power-in-large-scale-data-centers/>
21. Irani, S., Pruhs, K.R.: Algorithmic problems in power management. *ACM SIGACT News* **36**(2), 63–76 (2005)
22. Kim, S.J., Giannakis, G.B.: Real-time electricity pricing for demand response using online convex optimization. In: ISGT 2014, pp. 1–5. IEEE (2014)
23. Kim, T., Yue, Y., Taylor, S., Matthews, I.: A decision tree framework for spatiotemporal sequence prediction. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 577–586. ACM (2015)
24. Lin, M., Liu, Z., Wierman, A., Andrew, L.L.: Online algorithms for geographical load balancing. In: Green Computing Conference (IGCC), pp. 1–10. IEEE (2012)
25. Lin, M., Wierman, A., Andrew, L.L., Thereska, E.: Dynamic right-sizing for power-proportional data centers. *IEEE/ACM Trans. Netw. (TON)* **21**(5), 1378–1391 (2013)
26. Lin, M., Wierman, A., Andrew, L.L., Thereska, E.: Dynamic right-sizing for power-proportional data centers – extended version (2013)
27. Liu, Z., Lin, M., Wierman, A., Low, S.H., Andrew, L.L.: Greening geographical load balancing. In: Proceedings of the ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems, pp. 233–244. ACM (2011)

28. Meyerson, A.: The parking permit problem. In: 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), pp. 274–282. IEEE (2005)
29. Schmid, P., Roos, A.: Overclocking core i7: power versus performance (2009). www.tomshardware.com/reviews/overclock-core-i7,2268-10.html
30. Sellke, M.: Chasing convex bodies optimally. In: Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1509–1518. SIAM (2020)
31. Shan, A.: Heterogeneous processing: a strategy for augmenting Moore’s law. *Linux J.* **2006**(142), 7 (2006)
32. Wang, H., Huang, J., Lin, X., Mohsenian-Rad, H.: Exploring smart grid and data center interactions for electric power load balancing. *ACM SIGMETRICS Perform. Eval. Rev.* **41**(3), 89–94 (2014)
33. Zhang, M., Zheng, Z., Shroff, N.B.: An online algorithm for power-proportional data centers with switching cost. In: 2018 IEEE Conference on Decision and Control (CDC), pp. 6025–6032. IEEE (2018)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter’s Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter’s Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.