Algorithms for Dynamic Right-Sizing in Data Centers

Kevin Kappelmann

Chair for Theoretical Computer Science,
Technical University of Munich

May 13, 2017

Contents

1 Introduction

TODO: Hardware prices vs. energy costs in data centres, related work and purpose of this paper (offline algorithm, approximation algorithm,...).

1.1 Model description

We want to address the issue of above-mentioned ever-growing energy consumption by examing a scheduling problem that commonly arises in data centres. More specifically, we consider a model consisting of a fixed amount of homogeneous servers denoted by $m \in \mathbb{N}$ and a fixed amount of time slots denoted by $T \in \mathbb{N}$. In turn, each server possesses two power states, i.e. each server is either powered on (*active state*) or powered off (*sleep state*).

Need to describe what a time slot means?

Better name than sleep state?

For any time slot $t \in [T]$, we have a *mean arrival rate* denoted by λ_t, i.e. the amount of expected load to process in time slot t. We expect the arrival rates to be normalised such that each server can handle a load between 0 and 1 in any time slot. We denote the assigned load for server i in time slot t by $\lambda_{i,t} \in [0, 1]$. Consequently, for any time slot t, we expect an arrival rate between 0 and m, that is $\lambda_t \in [0, m]$; otherwise, the servers would not be able to process the given load in time.

The costs incurred by a single machine are described by the sum of the machine's *operating costs*, specified by $f : [0, 1] \to \mathbb{R}$, as well as its *(power state) switching costs*, specified by $\beta \in \mathbb{R}_{\geq 0}$. The operating costs f may not exclusively consider energy costs. For example, f may also allow for costs incurred by delays, such as lost revenue caused by users waiting for their responses. Similarly, β may also allow for delay costs, wear and tear costs or the like. [1]

Citation needed/appropriate?

We assume that a sleeping server does not cause any costs. Note that $f(0)$ describes the costs incurred by an idle server, not a sleeping one; in particular, $f(0)$ may be non-zero. Further, we assume convexity for f. This may seem like a notable restriction at first, but it indeed captures the behaviour of most modern server models. Since we are dealing with homogeneous servers, f and β are the same for all machines.

For convenience, we assume all machines sleeping at time $t = 0$ and force all machines to sleep after the scheduling process, i.e. at times $t > T$. Consequently, every server must power down exactly as many times as it powers on. This allows us to consolidate power up and power down costs into β and to model both costs as being incurred when powering up a server; that is, a model with power up costs $\beta_\uparrow$ and power down costs $\beta_\downarrow$ can be simply transferred to our model by setting $\beta \coloneqq \beta'_\uparrow \coloneqq \beta_\uparrow + \beta_\downarrow$ and $\beta'_\downarrow = 0$. Similarly, we assume that there are no loads at times $t \notin [T]$, that is $\lambda_t = \lambda_{i,t} = 0$ for $t \notin [T]$.

1.2 Problem statement

Using above definitions, we can define the input of our model by setting $\mathcal{I} \coloneqq (m, T, \Lambda, \beta, f)$ where $\Lambda = (\lambda_1, \dots, \lambda_T)$ is the sequence of arrival rates. We will subsequently identify a problem instance by its input $\mathcal{I}$. Naturally, given a problem instance $\mathcal{I}$, we want to schedule our servers in such a way that we minimise the sum of incurred costs while warranting that we are processing the given loads in time.

For this, consider for each server $i \in [m]$ the sequence of its states S_i and the sequence of its assigned loads L_i; that is

$$\begin{aligned} S_i &:= (s_{i,1}, \dots, s_{i,T}) \in \{0,1\}^T \\ L_i &:= (\lambda_{i,1}, \dots, \lambda_{i,T}) \in [0,1]^T \end{aligned}$$

where $s_{i,t} \in \{0,1\}$ denotes whether server i at time t is sleeping (0) or active (1). Recall that we assume all machines sleeping at times $t \notin [T]$; thus, for $t \notin [T]$ and $i \in [m]$, we have $s_{i,t} = 0$.

We can now define the sequence of all state changes and the sequence of all assigned loads:

$$\begin{aligned} \mathcal{S} &:= (S_1, \dots, S_m) \\ \mathcal{L} &:= (L_1, \dots, L_m) \end{aligned}$$

We will subsequently call a pair $\Sigma := (\mathcal{S}, \mathcal{L})$ a *schedule*. Finally, we are ready to define our problem statement.

Given an input $\mathcal{I}$, our goal is to find a schedule Σ that satisfies the following optimisation:

Definition of c(Σ) too hidden?

$$\text{minimise} \qquad c(\Sigma) := \underbrace{\sum_{t=1}^{T} \sum_{i=1}^{m} \big(f(\lambda_{i,t}) * s_{i,t}\big)}_{\text{operating costs}} + \underbrace{\beta * \sum_{t=1}^{T} \sum_{i=1}^{m} \min\{0, s_{i,t} - s_{i,t-1}\}}_{\text{switching costs}} \tag{1}$$

$$\text{subject to} \qquad \sum_{i=1}^{m} (\lambda_{i,t} * s_{i,t}) = \lambda_t, \quad \forall t \in [T] \tag{2}$$

We call a schedule *feasible* if it satisfies (2) and *optimal* if it satisifes (1) and (2).

2 Preliminaries

In this section, we conduct the prepratory work that will lay the foundations for our algorithms. For this, we analyse the structure of feasible schedules concering their cost efficiency in order to find characteristics of optimal schedules; these characteristics will then allow us to greatly simplify our optimisation conditions.

We begin by examing the state sequences of feasible schedules. As we are considering homogeneous servers, we do not care which exact servers process the given work loads. Rather we only care about the amount of active servers and the distribution of loads between them. It is in particular unreasonable to power down a machine and to power on a different machine in return; we could just keep the first machine powered on, saving switching costs. This investigation is captured by our first proposition.

Proposition 2.1. *Given a problem instance $\mathcal{I}$ and a feasible schedule Σ, there exists a feasible schedule Σ' such that*

(i) $c(\Sigma') \le c(\Sigma)$ *and*

(ii) Σ' *never powers on and shuts down servers at the same time slot, i.e.* Σ' *satisfies the following formula:*

$$\forall t \in [T]\Big[\big(\forall i \in [m](s_{i,t} - s_{i,t-1} \geq 0)\big) \vee \big(\forall i \in [m](s_{i,t} - s_{i,t-1} \leq 0)\big)\Big] \tag{3}$$

Proof. Let $\Sigma = (\mathcal{S}, \mathcal{L})$ be a feasible schedule for $\mathcal{I}$. We give a procedure that repeatedly modifies Σ such that it satisfies (3) and reduces or retains its costs.

Let $t \in [T]$ be the first time slot falsifying (3). If there does not exist such a time slot, we are done. Otherwise, we can obtain machines $i, j \in [m]$ such that $s_{i,t} - s_{i,t-1} = 1$ and $s_{j,t} - s_{j,t-1} = -1$, i.e. server i powers on at time t and server j powers off. Without loss of generality, we may assume $i < j$.

First, since all servers are sleeping at time $t = 0$, we have

$$s_{k,1} - s_{k,0} = s_{k,1} - 0 = s_{k,1} \geq 0, \quad \forall k \in [m]$$

which satisfies formula (3) for $t = 1$. Thus, we may assume $t > 1$.

Now consider the state sequences of server i and j:

$$\begin{aligned} S_i &= (s_{i,1}, \ldots, s_{i,t-1} = 0, s_{i,t} = 1, \ldots, s_{i,T}) \\ S_j &= (s_{j,1}, \ldots, s_{j,t-1} = 1, s_{j,t} = 0, \ldots, s_{j,T}) \end{aligned}$$

We modify S_i and S_j by swapping their states for time slots $\geq t$, that is we set

$$\begin{aligned} S'_i &\coloneqq (s_{i,1}, \ldots, s_{i,t-1} = 0, s_{j,t} = 0, \ldots, s_{j,T}) \\ S'_j &\coloneqq (s_{j,1}, \ldots, s_{j,t-1} = 1, s_{i,t} = 1, \ldots, s_{i,T}) \end{aligned}$$

Similarly, we need to swap the assigned loads for server i and j:

$$\begin{aligned} L'_i &\coloneqq (\lambda_{i,1}, \ldots, \lambda_{i,t-1}, \lambda_{j,t}, \ldots, \lambda_{j,T}) \\ L'_j &\coloneqq (\lambda_{j,1}, \ldots, \lambda_{j,t-1}, \lambda_{i,t}, \ldots, \lambda_{i,T}) \end{aligned}$$

Finally, we construct a new schedule $\Sigma' \coloneqq (\mathcal{S}', \mathcal{L}')$ given by

$$\begin{aligned} \mathcal{S}' &\coloneqq (S_1, \ldots, S_{i-1}, S'_i, S_{i+1}, \ldots, S_{j-1}, S'_j, S_{j+1}, \ldots, S_T) \\ \mathcal{L}' &\coloneqq (L_1, \ldots, L_{i-1}, L'_i, L_{i+1}, \ldots, L_{j-1}, L'_j, L_{j+1}, \ldots, L_T) \end{aligned}$$

We want to verify that Σ' is a feasible schedule, that is Σ' satisfies (2). For time slots $< t$ the schedules Σ' and Σ still coincide. For time slots $\geq t$ we only changed the order of summation in (2). Thus, Σ' is feasible.

Σ and Σ' coincide in their operating costs; however, their switching costs differ in that there are no switching costs β at time slot t for server i using Σ'. As we assume $\beta \geq 0$, we conclude $c(\Sigma') \leq c(\Sigma)$

Moreover, we decreased the amount of bad spotsat time slot t concerning (3). Hence, by repeating described process on Σ', we obtain a terminating procedure that returns a schedule satisfying the conditions. ∎

bad spots? better description?

As a special case, proposition 2.1 unfolds its power on optimal schedules, yielding the next corrolary.

Corollary 2.2. *Given a problem instance $\mathcal{I}$, there exists an optimal schedule Σ^* satisfying (3).*

Proof. Let Σ be an optimal schedule for $\mathcal{I}$. Applying proposition 2.1 to Σ yields Σ^* and the claim follows. $\square$

Next, we want to consider the sequence of active servers. For this, let $\mathcal{X}$ denote the sequence of sums of active servers at each time slot t, that is

$$\mathcal{X} \coloneqq \big(x_1 = \sum_{i=1}^{m} s_{i,1}, \dots, x_T = \sum_{i=1}^{m} s_{i,T}\big) \in \{0, \dots, m\}^T$$

As we assume all machines sleeping at times $t \notin [T]$, we have $x_t = 0$ for $t \notin [T]$.

The next proposition poses the cornerstone of our subsequent works. We want to establish an optimal scheduling strategy given a fixed amount of active servers. It turns out that equal load-sharing seems a very desirable strategy.

Proposition 2.3 (Equal load sharing)**.** *Given $x_t \in \mathbb{N}$ active servers at time slot t, an arrival rate $\lambda_t \in [0, x_t]$, and a convex cost function f, a most cost-efficient and feasible scheduling strategy is to assign each active server a load of λ_t / x_t.*

This is sounds awkwardly formulated, doesn't it?

Proof. Let Σ be an arbitrary, feasible schedule using x_t servers at time slot t, and let A be its set of active servers at time slot t, that is $A \coloneqq \{i \in [m] \mid s_{i,t} = 1\}$. Consider the operating costs of Σ at time t given by

$$\sum_{i=1}^{m} \big(f(\lambda_{i,t}) * s_{i,t}\big) = \sum_{i \in A} \big(f(\lambda_{i,t}) * 1\big) + \sum_{i \in [m] \setminus A} \big(f(\lambda_{i,t}) * 0\big) = \sum_{i \in A} f(\lambda_{i,t})$$

Since Σ is feasible (see constraint (2)), we have

$$\sum_{i \in A} \lambda_{i,t} = \lambda_t$$

Hence, we can obtain weights $\mu_1, \dots, \mu_{x_t} \in [0, 1]$ that relate $\lambda_{i,t}$ and λ_t for $i \in A$ such that

$$\sum_{i=1}^{x_t} \mu_i = 1 \quad \text{and} \quad \sum_{i \in A} f(\lambda_{i,t}) = \sum_{i=1}^{x_t} f(\mu_i \lambda_t) \tag{4}$$

In particular, we have

$$\sum_{i=1}^{x_t} \mu_i \lambda_t = \lambda_t \tag{5}$$

Using these weights, we now consider the operating costs of a schedule Σ^* that equally distributes λ_t to its x_t active servers:

$$\sum_{i=1}^{x_t} f\Big(\frac{\lambda_t}{x_t}\Big) = x_t * f\Big(\frac{\lambda_t}{x_t}\Big) \overset{(5)}{=} x_t * f\Big(\sum_{i=1}^{x_t} \frac{\mu_i \lambda_t}{x_t}\Big)$$

With the use of Jensen's inequality[1] and the fact that $\sum_{i=1}^{x_t}(1/x_t) = 1$, we can give an upper bound for the costs : style (footnote) okay?

$$x_t * f\Big(\frac{\lambda_t}{x_t}\Big) \le x_t \sum_{i=1}^{x_t} \frac{1}{x_t} f(\mu_i \lambda_t) = \frac{x_t}{x_t} \sum_{i=1}^{x_t} f(\mu_i \lambda_t) = \sum_{i=1}^{x_t} f(\mu_i \lambda_t) \overset{(4)}{=} \sum_{i \in A} f(\lambda_{i,t})$$

Thus, the operating costs of Σ^* give a lower bound for the operatings costs of Σ, and the claim follows. $\square$

As a special case, we can again apply our just derived proposition to optimal schedules.

Corollary 2.4. *Given a problem instance $\mathcal{I}$, there exists an optimal schedule Σ^* that equally distributes its arrival rates to its active servers in each time slot.*

Proof. Let $\Sigma = (\mathcal{S}, \mathcal{L})$ be an optimal schedule for $\mathcal{I}$. We exchange $\mathcal{L}$ with a new strategy $\mathcal{L}^*$ that equally distributes the arrival rates to all active servers of Σ in each time slot. Setting $\Sigma^* := (\mathcal{S}, \mathcal{L}^*)$ we have $c(\Sigma^*) \le c(\Sigma)$ by proposition 2.3 and the claim follows. $\square$

As a result of corrolary 2.4, we can restrict ourselves in finding an optimal schedule that equally distributes its arrival rates to its active servers. Together with corollary 2.2, this allows us to subsequently identify an optimal schedule by its sequence of active servers $\mathcal{X}$.

We are now able to simplify our optimisation conditions (1) and (2). For this, given a problem instance $\mathcal{I}$, we define the operating costs function $c_{op}(x, \lambda)$ that describes the costs incurred by equally distributing λ on x active servers using f:

$$c_{op} : \{0, \dots, m\} \times [0, m] \to \mathbb{R} \cup \{\infty\}, \quad c_{op}(x, \lambda) = \begin{cases} 0, & \text{if } x = 0 \\ x * f(\lambda/x), & \text{if } x \neq 0 \wedge \lambda \le x \\ \infty, & \text{if } x \neq 0 \wedge \lambda > x \end{cases}$$

We assign infinite costs in case $\lambda > x$ as there would be too few active servers to process the arrival rate. Next, we define the switching costs function $c_{sw}(x_{t-1}, x_t)$ that describes the incurred costs when changing the amount of active server from x_{t-1} to x_t:

$$c_{sw}(x_{t-1}, x_t) := \beta * \max\{0, x_t - x_{t-1}\}$$

Lastly, we can define the costs function $c(x_{t-1}, x_t, \lambda_t)$ that describes the incurring costs for a single time step using an equal distribution of loads:

$$c(x_{t-1}, x_t, \lambda_t) := c_{op}(x_t, \lambda_t) + c_{sw}(x_{t-1}, x_t)$$

[1]For convex $f : \mathbb{R} \to \mathbb{R}$, arbitrary $\lambda_1, \dots, \lambda_n \in \mathbb{R}$, and $x_1, \dots, x_n \in [0, 1]$ satisfying $\sum_{i=1}^{n} x_i = 1$ we have:

$$f\Big(\sum_{i=1}^{n} x_i \lambda_i\Big) \le \sum_{i=1}^{n} x_i f(\lambda_i)$$

The optimisation conditions for a schedule now simplify to one single minimalisation:

$$\text{minimise} \quad c(\mathcal{X}) \coloneqq \sum_{t=1}^{T} c(x_{t-1}, x_t, \lambda_t) \tag{6}$$

We subsequently call a schedule $\mathcal{X}$ *optimal* if it satisfies (6).

3 Optimal offline scheduling

In this section, we derive an optimal offline algorithm based on our preliminary work. We reduce our problem specified $\mathcal{I}$ to a shortest path problem of a level structured graph G. We then use a dynamic programming approach to find a shortest path of G and thereby an optimal schedule for $\mathcal{I}$ in pseudo-polynomial time.

3.1 Graph for an optimal schedule

Let $\mathcal{I}$ be a problem instance. Thanks to our preliminary work, we know that there exists an optimal schedule which is identifiable by its sequence of active servers $\mathcal{X}$. In order to find this sequence $\mathcal{X}$, consider the weighted, level structured graph G defined as follows:

$$\begin{aligned}
&V \coloneqq \big\{v_{x,t} \mid x \in \{0,\dots,m\}, t \in \{0,\dots,T+1\}\big\}\\
&E \coloneqq \big\{(v_{x,t}, v_{x',t+1}) \mid x, x' \in \{0,\dots,m\}, t \in \{0,\dots,T\}, v_{x,t}, v_{x',t+1} \in V\big\}\\
&c_G(v_{x,t}, v_{x',t+1}) \coloneqq c(x, x', \lambda_{t+1})\\
&G \coloneqq (V, E, c_G)
\end{aligned}$$

For any possible amount of active servers x and any time slot t we add a node $v_{x,t}$. Moreover, we add a start node $v_{0,0}$ as well as an end node $v_{0,T+1}$. Next, we connect all nodes to their successors with respect to time. Semantically, $v_{x,t}$ denotes the state of scheduling the arrival rate λ_t equally to x servers. For any edge connecting $v_{x,t}$ with $v_{x',t+1}$ we assign costs $c(x, x', \lambda_{t+1})$, i.e. the costs of taking the edge correspond to switching from x to x' machines and processing the load λ_{t+1} with x' machines.

Maybe something different than c_G?

The costs of a path $P = (v_{x_0,T_0}, v_{x_1,T_0+1}, \dots, v_{x_n,T_0+n})$ with length n in our graph (where $x_t \in \{0,\dots,m\}, T_0 \in \{0,\dots,T\}$ and $T_0 + n \leq T+1$) are thus given by

Is this well-written?

$$c(P) \coloneqq \sum_{t=1}^{n} c(x_{t-1}, x_t, \lambda_{T_0+t})$$

In particular, for a path $P = (v_{0,0}, v_{x_1,1}, \dots, v_{x_T,T}, v_{0,T+1})$ from our start node $v_{0,0}$ to our end node $v_{0,T+1}$ we have

$$c(P) = c(0, x_1, \lambda_1) + \sum_{t=2}^{T} c(x_{t-1}, x_t, \lambda_t) + \underbrace{c(x_T, 0, 0)}_{=0} = c(0, x_1, \lambda_1) + \sum_{t=2}^{T} c(x_{t-1}, x_t, \lambda_t) \tag{7}$$

Note that the costs of such a path directly correspond to those of a schedule $\mathcal{X}$ (see (6)). Any shortest path from $v_{0,0}$ to $v_{0,T+1}$ is thus forced to minimise the costs of the corresponding schedule. Needless to say, this demands for a proof of correctness.

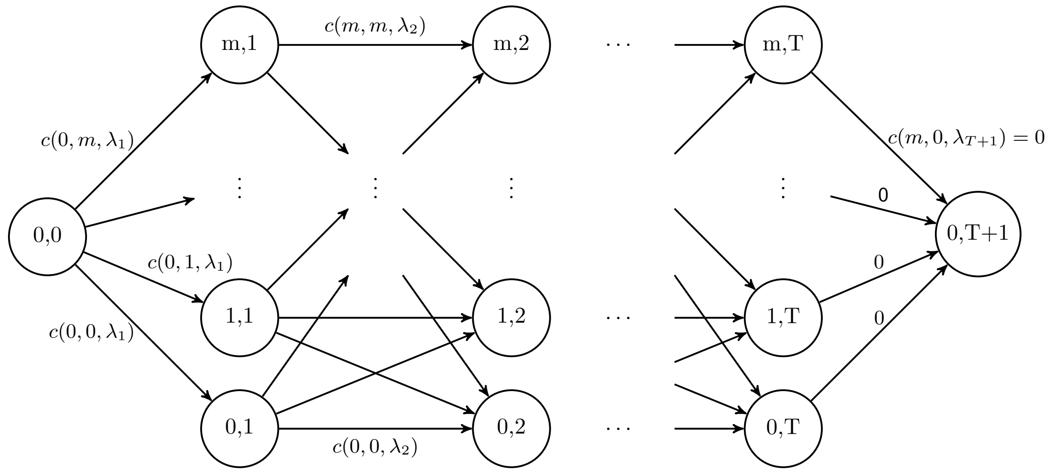


Figure 1: Level structured graph for an optimal offline algorithm. Note that $c(i, 0, \lambda_{T+1}) = 0$ for all $i \in \{0, \dots, m\}$ because $\lambda_{T+1} = 0$ by our conventions.

Lemma 3.1. *Any given schedule $\mathcal{X}$ for $\mathcal{I}$ corresponds to a path P from $v_{0,0}$ to $v_{0,T+1}$ with $c(\mathcal{X}) = c(P)$ and vice versa.*

Proof.

"⇒": Let $\mathcal{X} = (x_1, \dots, x_T)$ be a schedule for $\mathcal{I}$. We construct a path in our graph by setting

$$P \coloneqq (v_{0,0}, v_{x_1,1}, v_{x_2,2}, \dots, v_{x_T,T}, v_{0,T+1})$$

We examine the costs and conclude

$$c(\mathcal{X}) \overset{(6)}{=} \sum_{t=1}^{T} c(x_{t-1}, x_t, \lambda_t) = \underbrace{c(x_0, x_1, \lambda_1)}_{=c(0,x_1,\lambda_1)} + \sum_{t=2}^{T} c(x_{t-1}, x_t, \lambda_t) \overset{(7)}{=} c(P)$$

"⇐": Let $P = (v_{0,0}, v_{x_1,1}, v_{x_2,2}, \dots, v_{x_T,T}, v_{0,T+1})$ be a path from $v_{0,0}$ to $v_{0,T+1}$. We construct a schedule for $\mathcal{I}$ by setting

$$\mathcal{X} \coloneqq (x_1, \dots, x_T)$$

Again, by examing the costs we conclude

$$c(P) \overset{(7)}{=} \underbrace{c(0, x_1, \lambda_1)}_{=c(x_0,x_1,\lambda_1)} + \sum_{t=2}^{T} c(x_{t-1}, x_t, \lambda_t) = \sum_{t=1}^{T} c(x_{t-1}, x_t, \lambda_t) \overset{(6)}{=} c(\mathcal{X})$$

Thus, the claim follows. □

Theorem 3.2. *Any given optimal schedule $\mathcal{X}$ for $\mathcal{I}$ corresponds to a shortest path P from $v_{0,0}$ to $v_{0,T+1}$ with $c(\mathcal{X}) = c(P)$ and vice versa.*

Proof. By lemma 3.1 we have a one-to-one correspondence between schedules $\mathcal{X}$ and paths P obeying $c(\mathcal{X}) = c(P)$. Thus, we have

$$c(\mathcal{X}) \text{ minimal} \iff c(P) \text{ minimal}$$

and the claim follows. □

3.2 A pseudo-polynomial minimum cost algorithm

In the following, we give an algorithm based on our results from the preceding section. We use a dynamic programming approach similar to the well-known Bellman-Ford algorithm.

Algorithm 1 Costs and parents for optimal offline scheduling

function OPTIMAL_OFFLINE$(m, T, \Lambda, \beta, f)$
  let $C[T+1, m]$ and $p[T+1, m]$ **be** new arrays ▷ Costs and parents
  for $x \leftarrow 0$ to m **do** ▷ Initialisation
    $C[1, x] \leftarrow c(0, x, \lambda_1)$
    $p[1, x] \leftarrow 0$
  for $t \leftarrow 2$ to T **do** ▷ Iterative calculation of costs and parents
    for $x' \leftarrow 0$ to m **do**
      $p[t, x'] \leftarrow \operatorname*{arg\,min}_{x \in \{0,\ldots,m\}} \left\{C[t-1, x] + c(x, x', \lambda_t)\right\}$ ▷ Find best preceding choice
      $C[t, x'] \leftarrow C\big[t-1, p[t, x']\big] + c\big(p[t, x'], x', \lambda_t\big)$
  $p[T+1, 0] \leftarrow \operatorname*{arg\,min}_{x \in \{0,\ldots,m\}} \left\{C[T, x]\right\}$ ▷ Find best choice at last time slot
  $C[T+1, 0] \leftarrow C\big[T, p[T+1, 0]\big]$
  return C and p

Algorithm 2 Extract schedule from result of algorithm 1

function EXTRACT_OPTIMAL_OFFLINE(C, p, m, T)
  let $\mathcal{X}[T]$ **be** a new array
  $\mathcal{X}[T] \leftarrow p[T+1, 0]$ ▷ Get minimum at last time slot
  for $t \leftarrow T-1$ to 1 **do** ▷ Iteratively obtain schedule from parents array
    $\mathcal{X}[t] \leftarrow p\big[t+1, \mathcal{X}[t+1]\big]$
  return $\mathcal{X}$

Algorithm 1 needs $\mathcal{O}(m)$ iterations for its initialisation, $\mathcal{O}(Tm^2)$ steps for the iterative calculation, and $\mathcal{O}(m)$ steps for its final minimisation search. Algorithm 2 needs $\mathcal{O}(T)$ iterations for its schedule calculation. Thus, we receive a run time of

$$\mathcal{O}(m + Tm^2 + m + T) = \mathcal{O}(Tm^2)$$

The running time is polynomial in the numeric value of the input; however, as we just need $\log_2(m)$ bits to encode m, it is exponential in the length of the input. Hence, the algorithm is pseudo-polynomial.

4 A polynomial 4-approximation algorithm for monotonically increasing convex f

We consider a modification of the problem discussed in chapter 3. Assuming that f is convex and monotonically increasing, we can modify our algorithm to obtain a polynomial time 4-approximation algorithm.

4.1 Graph for a 4-optimal schedule

We modify our graph from chapter 3.1 to the reduce the number of vertices. For this, we stop adding m vertices for each timestep, but use vertices that approximate the number of active servers instead. First, let $b := \lceil \log_2(m) \rceil$. We add vertices $(t, 0)$ and $(t, 2^i), \forall t \in [T-1], 0 \leq i \leq b$. All edges and weights are added analogous to chapter 3.1.

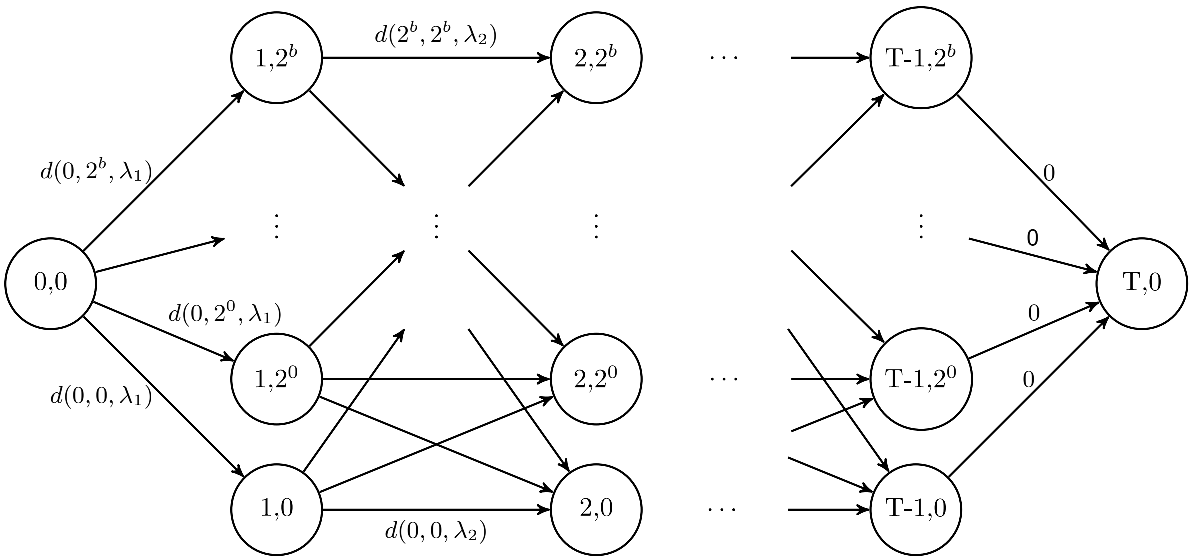


Figure 2: Graph for a 4-approximation algorithm

Definition 4.1. Let $\mathcal{X} = (x_0, \ldots, x_T)$ be a schedule and $t > 0$.
We say that $\mathcal{X}$ changes its **state** at time t if

$$x_t \neq x_{t-1}$$

and that $\mathcal{X}$ changes its **2-state** at time t if

$$x_t = 0 \quad \text{or} \quad x_t \notin \left(2^{\lfloor \log_2(x_{t-1}) \rfloor}, 2^{\lceil \log_2(x_{t-1}) \rceil}\right)$$

Proposition 4.2.

1. *Any given optimal schedule $\mathcal{X}$ can be transformed to a 4-optimal schedule $\mathcal{X}'$ which corresponds to a path P from $(0,0)$ to $(T,0)$ with costs$(\mathcal{X}') =$ costs(P).*

2. *Any shortest path P from $(0,0)$ to $(T,0)$ corresponds to a 4-optimal schedule $\mathcal{X}$ with $costs(P) = costs(\mathcal{X})$.*

Proof.

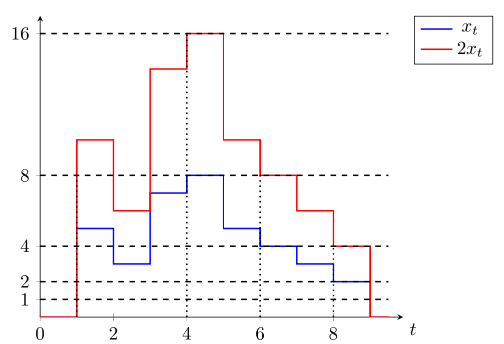
1. Assume we have an optimal schedule identified by $\mathcal{X} = (x_0, \dots, x_T)$. For $0 \le t < T$ we inductively set:

$$x'_0 \coloneqq 0, \qquad x'_{t+1} \coloneqq \begin{cases} \min\{2^{\lfloor \log_2(2x_{t+1}) \rfloor}, 2^b\}, & \text{if } 0 < x_t \le x_{t+1} \\ 2^{\lceil \log_2(2x_{t+1}) \rceil}, & \text{if } 0 < x_{t+1} < x_t \text{ and } x'_t \ge 4x_{t+1} \\ x'_t, & \text{if } 0 < x_{t+1} < x_t \text{ and } x'_t < 4x_{t+1} \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

Then let $\mathcal{X}' \coloneqq (x'_0, \dots, x'_T)$ be the modified sequence of active servers. Notice that $x_t \le x'_t \le 4x_t$ holds as x'_t is at most the smallest power of two larger than $2x_t$ which implies that $\mathcal{X}'$ is feasible.
We can now construct a feasible path in our graph from $\mathcal{X}'$ as follows:

$$\begin{aligned} &\text{First set} && e_t \coloneqq \Big(\big(t, \mathcal{X}'(t)\big), \big(t+1, \mathcal{X}'(t+1)\big)\Big), && \forall t \in \{0, \dots, T-1\} \\ &\text{then set} && P \coloneqq (e_0, \dots, e_{T-1}) \end{aligned}$$

By the definition of the edges' weights it follows that $costs(\mathcal{X}') = costs(P)$.
Next, let $(t_0 = 0, t_1, \dots, t_n = 0)$ be the sequence of times where the optimal schedule $\mathcal{X}$ changes its 2-state. Notice that the modified schedule $\mathcal{X}'$ changes its state only at times t_i and that $2x_{t_i} \le x'_{t_i}$ holds (TODO: only if not discrete but continous time steps). This can be seen exemplarily in figure 3 by obvserving that $\mathcal{X}'$ changes its state only if $\mathcal{X}$ crosses or touches a bordering power of two.



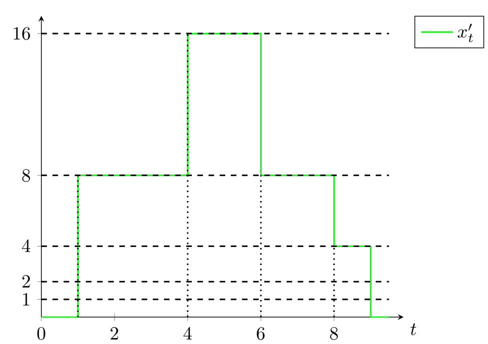


Figure 3: Adaption of an optimal schedule

For this reason, we now only have to consider the fraction of costs of $\mathcal{X}'$ and $\mathcal{X}$ between time steps t_{i-1} and t_i

$$\frac{costs(\mathcal{X}', t_{i-1}, t_i)}{costs(\mathcal{X}, t_{i-1}, t_i)} \tag{9}$$

For $x_{t_i} = 0$ it follows from (**??**) that $costs(\mathcal{X}', t_{i-1}, t_i) = costs(\mathcal{X}, t_{i-1}, t_i) = 0$. Hence, we can restrict ourselves to $0 < t_i < T$ with $x_{t_i} \neq 0$. The costs incurred by $\mathcal{X}'$ are given by

$$\begin{aligned}
costs(\mathcal{X}', t_{i-1}, t_i) &= \beta \max\{0, x'_{t_i} - x'_{t_{i-1}}\} + x'_{t_i} f(\lambda_{t_i}/x'_{t_i}) && \text{by (\textbf{??})} \\
&\leq \beta \max\{0, x'_{t_i} - x'_{t_{i-1}}\} + 4x_{t_i} f(\lambda_{t_i}/x'_{t_i}) && \text{by (8)} \\
&\leq \beta \max\{0, x'_{t_i} - x'_{t_{i-1}}\} + 4x_{t_i} f(\lambda_{t_i}/x_{t_i}) && \text{f monotonically increasing} \\
\implies \quad costs(\mathcal{X}', t_{i-1}, t_i) &\leq \beta \max\{0, x'_{t_i} - x'_{t_{i-1}}\} + 4x_{t_i} f(\lambda_{t_i}/x_{t_i}) && (10)
\end{aligned}$$

and the costs of $\mathcal{X}$ by

$$costs(\mathcal{X}, t_{i-1}, t_i) = \beta \max\{0, x_{t_i} - x_{t_{i-1}}\} + x_{t_i} f(\lambda_{t_i}/x_{t_i}) \tag{11}$$

W.l.o.g. we may assume $x_{t_i} f(\lambda_{t_i}/x_{t_i}) > 0$, otherwise the claim follows trivially. (TODO: is it really trivial?)

(i) $\underline{x_{t_i} \leq x_{t_{i-1}}}$: From (8) it follows that $x'_{t_i} \leq x'_{t_{i-1}}$. Thus, we can simplify (9):

$$\begin{aligned}
\frac{costs(\mathcal{X}', t_{i-1}, t_i)}{costs(\mathcal{X}, t_{i-1}, t_i)} &\leq \frac{\beta \max\{0, x'_{t_i} - x'_{t_{i-1}}\} + 4x_{t_i} f(\lambda_{t_i}/x_{t_i})}{\beta \max\{0, x_{t_i} - x_{t_{i-1}}\} + x_{t_i} f(\lambda_{t_i}/x_{t_i})} && \text{by (10),(11)} \\
&= \frac{4x_{t_i} f(\lambda_{t_i}/x_{t_i})}{x_{t_i} f(\lambda_{t_i}/x_{t_i})} && (x_{t_i} \leq x_{t_{i-1}} \text{ and } x'_{t_i} \leq x'_{t_{i-1}}) \\
&= 4
\end{aligned}$$

(ii) $\underline{x_{t_i} > x_{t_{i-1}}}$: From (8) it follows that $x'_{t_i} \geq x'_{t_{i-1}}$. Thus, we can simplify (9):

$$\begin{aligned}
\frac{costs(\mathcal{X}', t_{i-1}, t_i)}{costs(\mathcal{X}, t_{i-1}, t_i)} &\leq \frac{\beta \max\{0, x'_{t_i} - x'_{t_{i-1}}\} + 4x_{t_i} f(\lambda_{t_i}/x_{t_i})}{\beta \max\{0, x_{t_i} - x_{t_{i-1}}\} + x_{t_i} f(\lambda_{t_i}/x_{t_i})} && \text{by (10),(11)} \\
&= \frac{\beta (x'_{t_i} - x'_{t_{i-1}}) + 4x_{t_i} f(\lambda_{t_i}/x_{t_i})}{\beta (x_{t_i} - x_{t_{i-1}}) + x_{t_i} f(\lambda_{t_i}/x_{t_i})} && (x_{t_i} > x_{t_{i-1}} \text{ and } x'_{t_i} \geq x'_{t_{i-1}}) \\
&= \frac{\beta (\min\{2^{\lfloor \log_2(2x_{t_i}) \rfloor}, 2^b\} - x'_{t_{i-1}}) + 4x_{t_i} f(\lambda_{t_i}/x_{t_i})}{\beta (x_{t_i} - x_{t_{i-1}}) + x_{t_i} f(\lambda_{t_i}/x_{t_i})} && \text{by (8)} \\
&\leq \frac{\beta (2^{\lfloor \log_2(2x_{t_i}) \rfloor} - x'_{t_{i-1}}) + 4x_{t_i} f(\lambda_{t_i}/x_{t_i})}{\beta (x_{t_i} - x_{t_{i-1}}) + x_{t_i} f(\lambda_{t_i}/x_{t_i})} \\
&\leq \frac{\beta (2x_{t_i} - x'_{t_{i-1}}) + 4x_{t_i} f(\lambda_{t_i}/x_{t_i})}{\beta (x_{t_i} - x_{t_{i-1}}) + x_{t_i} f(\lambda_{t_i}/x_{t_i})} \\
&\leq \frac{\beta (2x_{t_i} - 2x_{t_{i-1}}) + 4x_{t_i} f(\lambda_{t_i}/x_{t_i})}{\beta (x_{t_i} - x_{t_{i-1}}) + x_{t_i} f(\lambda_{t_i}/x_{t_i})} && \text{by } (2x_{t_{i-1}} \leq x'_{t_{i-1}}) \\
&\leq 4 \, \frac{\frac{1}{2}\beta (x_{t_i} - x_{t_{i-1}}) + x_{t_i} f(\lambda_{t_i}/x_{t_i})}{\beta (x_{t_i} - x_{t_{i-1}}) + x_{t_i} f(\lambda_{t_i}/x_{t_i})} \\
&\leq 4
\end{aligned}$$

From (i) and (ii) it follows:

$$costs(\mathcal{X}') \leq 4 costs(\mathcal{X})$$

2. From 1 we obtain that we can construct a 4-optimal path P' from any optimal schedule. Now, let P be a shortest path. We have $costs(P) \leq costs(P') < \infty$, and since every path P with $costs(P) < \infty$ corresponds to a feasible schedule $\mathcal{X}$ with $costs(P) = costs(\mathcal{X})$, $\mathcal{X}$ must also be at least 4-optimal.

$\square$

References

[1] Minghong Lin, Adam Wierman, Lachlan L. H. Andrew, and Eno Thereska. Dynamic right-sizing for power-propotional data centers. *IEEE/ACM Transactions on Networking (TON)*, 21:1378–1391, 2013.

Appendix

Below, we give an overview of just given definitions and conventions commonly referred to in our paper:

Good idea to have an appendix?

- Input:
  - $m \in \mathbb{N} \ldots$ number of homogeneous servers
  - $T \in \mathbb{N} \ldots$ number of time slots
  - $\lambda_1, \ldots, \lambda_T \in [0, m] \ldots$ arrival rates
  - $\Lambda \coloneqq (\lambda_1, \ldots, \lambda_T) \ldots$ sequence of arrival rates
  - $\beta \in \mathbb{R}_{\geq 0} \ldots$ switching costs of a server
  - $f : [0, 1] \to \mathbb{R} \ldots$ convex operating costs function of a server
  - $\mathcal{I} \coloneqq (m, T, \Lambda, \beta, f) \ldots$ input of a problem instance
- Problem statement:
  - $s_{i,t} \in \{0, 1\} \ldots$ state of server i at time t, i.e. sleeping (0) or active(1)
  - $S_i \coloneqq (s_{i,1}, \ldots, s_{i,T}) \ldots$ sequence of states for server i
  - $\lambda_{i,t} \in [0, 1] \ldots$ assigned load for server i at time t
  - $L_i \coloneqq (\lambda_{i,1}, \ldots, \lambda_{i,T}) \ldots$ sequence of assigned loads for server i
  - $\mathcal{S} \coloneqq (S_1, \ldots, S_m) \ldots$ sequence of all state changes
  - $\mathcal{L} \coloneqq (L_1, \ldots, L_m) \ldots$ sequence of all assigned loads
  - $\Sigma \coloneqq (\mathcal{S}, \mathcal{L}) \ldots$ schedule for a problem instance $\mathcal{I}$
- Miscellaneous:
  - $x_t \in \{0, \ldots, m\} \ldots$ number of active servers at time t
  - $\mathcal{X} \coloneqq (x_1, \ldots, x_T) \ldots$ sequence of number of active servers
- Conventions:
  - $\lambda_t = 0$ for all $t \notin [T]$, i.e. there is no load before and after the scheduling process
  - $s_{i,t} = 0$ for all $t \notin [T]$, i.e. all servers are powered down before and after the scheduling process