

# Algorithms for Dynamic Right-Sizing in Data Centers

Kevin Kappelmann

*Chair for Theoretical Computer Science,  
Technical University of Munich*

May 6, 2017

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Model description . . . . .	1
1.2	Problem statement . . . . .	1
<b>2</b>	<b>Preliminaries</b>	<b>2</b>
<b>3</b>	<b>Optimal scheduling for m homogeneous servers</b>	<b>3</b>
3.1	Graph for an optimal schedule . . . . .	3
3.2	A pseudo-polynomial minimum cost algorithm . . . . .	5
3.2.1	Runtime analysis . . . . .	5
3.2.2	A memory optimized algorithm . . . . .	6
<b>4</b>	<b>A polynomial 4-approximation algorithm for monotonically increasing convex f</b>	<b>6</b>
4.1	Graph for a 4-optimal schedule . . . . .	6

# 1 Introduction

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

## 1.1 Model description

We want to address the issue of the above-mentioned ever-growing energy consumption by examining a scheduling problem commonly arising 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*).

Better name than sleep state?

For any time slot  $t \in [T]$  we have a *mean arrival rate* denoted by  $\lambda_t$ , 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  $i \in [m]$  can handle a load between zero and one in any time step. 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  $m$ , i.e.  $\lambda_t \in [0, m]$ ; otherwise, we would not be able to process the given load in time.

The incurred energy costs of a single machine can be described by the sum of the machine's (*power state*) *switching costs* specified by  $\beta \in \mathbb{R}_{\geq 0}$  as well as its *operating costs* specified by  $f : [0, 1] \rightarrow \mathbb{R}$ . We assume that a sleeping server does not generate any energy costs. Note that  $f(0)$  describes the costs generated by an idle server, not a sleeping one; in particular,  $f(0)$  may be unequal to 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. As we are dealing with homogeneous servers,  $\beta$  and  $f$  are the same for all machines.

We want to stress that  $f$  may not only pose as a depiction of energy costs. For example,  $f$  may also allow for costs incurred by delays, such as lost revenue caused by users that need to wait for their responses. Similarly,  $\beta$  may also allow for delay costs, wear and tear costs or the like. [1]

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  $t > T$ . This justifies the consolidation of power up and power down costs into  $\beta$  because it allows us 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 simply be transferred to our model by setting  $\beta := \beta_{\uparrow} + \beta_{\downarrow}$ . We can now proceed to define our problem statement.

## 1.2 Problem statement

Using above definitions, we can define the input of our model by  $\mathcal{I} := (m, T, \Lambda, \beta, f)$  where  $\Lambda = (\lambda_1, \dots, \lambda_T)$  is the sequence of arrival rates. Naturally, given an input  $\mathcal{I}$ , 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 the two sequences

$$\begin{aligned}\mathcal{S} &:= (s_{1,1}, s_{2,1}, \dots, s_{m,1}, s_{1,2}, \dots, s_{m,T}) \in \{0,1\}^{m \times T} \\ \mathcal{L} &:= (\lambda_{1,1}, \lambda_{2,1}, \dots, \lambda_{m,1}, \lambda_{1,2}, \dots, \lambda_{m,T}) \in [0,1]^{m \times T}\end{aligned}$$

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  $t \notin [T]$ ; consequently, for  $t \notin [T]$  and  $i \in [m]$  we set  $s_{i,t} = 0$ .

Retell what  $\lambda_{i,t}$  means?

We will subsequently call a pair  $(\mathcal{S}, \mathcal{L}) =: \Sigma$  a *schedule*. Our goal is to find for an input  $\mathcal{I}$  a schedule  $\Sigma$  that satisfies the following optimisation:

$$\begin{aligned}\text{minimise} \quad & \underbrace{\sum_{t=1}^T \sum_{i=1}^m f(\lambda_{i,t}) * s_{i,t}}_{\text{operating costs}} + \underbrace{\beta * \sum_{t=1}^T \sum_{i=1}^m \min\{0, s_{i,t} - s_{i,t-1}\}}_{\text{switching costs}} \\ \text{subject to} \quad & \sum_{i=1}^m \lambda_{i,t} * s_{i,t} = \lambda_t, \forall t \in [T]\end{aligned}$$

We call a schedule  $\Sigma$  satisfying this optimisation an *optimal schedule*.

## 2 Preliminaries

In this section, we conduct the preparatory work needed for our algorithms and proofs.

This sounds strange... What should stand here?

**Proposition 2.1.** *Given a problem instance  $\mathcal{I} = (m, T, \Lambda, \beta, f)$ , there exists an optimal schedule  $\Sigma = (\mathcal{S}, \mathcal{L})$  such that.*

*Proof.*

□

Next, let  $\mathcal{X}$  denote the sequence of sums of active servers at each time slot  $t$ , that is:

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

Recall that we assume all machines sleeping at times  $t \notin [T]$ ; consequently, for  $t \notin [T]$  we set  $x_t = 0$ .

$$\begin{aligned}\text{minimise} \quad & \underbrace{\sum_{t=1}^T \sum_{i=1}^m f(\lambda_{i,t}) * s_{i,t}}_{\text{operating costs}} + \underbrace{\beta * \sum_{t=1}^T \min\{0, x_t - x_{t-1}\}}_{\text{switching costs}} \\ \text{subject to} \quad & \sum_{i=1}^m \lambda_{i,t} * s_{i,t} = \lambda_t, \forall t \in [T]\end{aligned}$$

**Lemma 2.2** (Equal load sharing). *Given  $x \in \mathbb{N}$  active servers, an arrival rate  $\lambda \in [0, m]$ , and a convex cost function  $f$ , the most cost-efficient scheduling strategy is to assign each active server a load of  $\lambda/x$ .*

*Proof.*  $\mu_i \in [0, 1] : \sum_{i=1}^x \mu_i = 1 :$

$$\begin{aligned}
f\left(\frac{\lambda}{x}\right) &= f\left(\sum_{i=1}^x \frac{\mu_i \lambda}{x}\right) \\
\Rightarrow f\left(\frac{\lambda}{x}\right) &\leq \sum_{i=1}^x \frac{1}{x} f(\mu_i \lambda) && \text{by Jensen's inequality} \\
\Longleftrightarrow x f\left(\frac{\lambda}{x}\right) &\leq \sum_{i=1}^x f(\mu_i \lambda)
\end{aligned}$$

□

Lemma 2.2 allows us to uniquely identify an optimal schedule by its sequence of numbers of active servers  $\mathcal{X}$ .

**Definition 2.3.** Define the minimum costs function of a feasible sequence  $\mathcal{X}$  between time steps  $t$  and  $t'$  with  $0 \leq t < t' \leq T$  as

$$\text{costs}(\mathcal{X}, t, t') := \beta \max\{0, x_{t'} - x_t\} + x_{t'} f(\lambda_{t'}/x_{t'}) \quad (1)$$

Then the minimum costs of a feasible sequence  $\mathcal{X}$  at time  $0 < t \leq T$  are given by

$$\text{costs}(\mathcal{X}, t-1, t) := \text{costs}(\mathcal{X}, t) = \underbrace{\beta \max\{0, x_t - x_{t-1}\}}_{\text{power up costs}} + x_t f(\lambda_t/x_t)$$

and the total costs by

$$\text{costs}(\mathcal{X}) := \sum_{t=1}^T \beta \max\{0, x_t - x_{t-1}\} + x_t f(\lambda_t/x_t)$$

### 3 Optimal scheduling for m homogeneous servers

TODO: introduction text

#### 3.1 Graph for an optimal schedule

We construct a directed acyclic graph as follows:

$\forall t \in [T-1]$  and  $i, j \in \{0, \dots, m\}$  we add vertices  $(t, i)$  modelling the number of active servers at time  $t$ . Moreover, we add vertices  $(0, 0)$  and  $(T, 0)$  for our initial and final state respectively.

In order to warrant that there are at least  $\lceil \lambda_t \rceil$  active servers  $\forall t \in [T-1]$ , we define an auxiliary function which calculates the costs for handling an arrival rate  $\lambda$  with  $x$  active servers:

$$c(x, \lambda) := \begin{cases} 0, & \text{if } x = 0 \\ x f(\lambda/x), & \text{if } x \neq 0 \wedge \lambda \leq x \\ \infty, & \text{otherwise} \end{cases} \quad (2)$$

Then,  $\forall t \in [T-2]$  and  $i, j \in \{0, \dots, m\}$ , we add edges from  $(t, i)$  to  $(t+1, j)$  with weight

$$d(i, j, \lambda_{t+1}) := \beta \max\{0, j - i\} + c(j, \lambda_{t+1}) \quad (3)$$

Finally, for  $0 \leq i \leq m$  we add edges from  $(0, 0)$  to  $(1, i)$  with weight  $d(0, i, \lambda_1)$  and from  $(T-1, i)$  to  $(T, 0)$  with weight  $d(i, 0, \lambda_T) = 0$ .

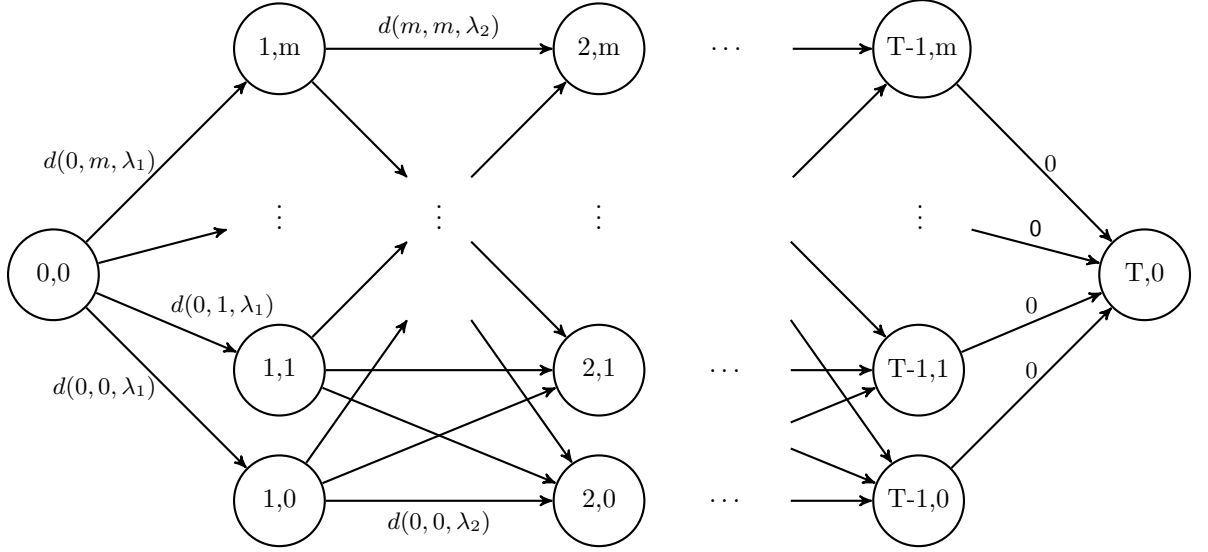


Figure 1: Graph for optimal schedule algorithm.

**Note:** All edges from  $(t, i)$  to  $(t+1, j)$  have weight  $d(i, j, \lambda_{t+1})$

**Proposition 3.1.** *Any given optimal schedule  $\mathcal{X}$  corresponds to a shortest path  $P$  from  $(0, 0)$  to  $(T, 0)$  with  $\text{costs}(\mathcal{X}) = \text{costs}(P)$  and vice versa.*

*Proof.*

“ $\Rightarrow$ ”: We construct a feasible path in our graph from  $\mathcal{X}$  as follows:

$$\begin{aligned} \text{First set} \quad e_t &:= \left( (t, \mathcal{X}(t)), (t+1, \mathcal{X}(t+1)) \right), \quad \forall t \in \{0, \dots, T-1\} \\ \text{then set} \quad P &:= (e_0, \dots, e_{T-1}) \end{aligned}$$

As each edge  $e_t$  in our graph has weight  $d(\mathcal{X}(t-1), \mathcal{X}(t), \lambda_t)$ , it corresponds to the costs of switching from  $\mathcal{X}(t-1)$  to  $\mathcal{X}(t)$  servers and processing  $\lambda_t$  with  $\mathcal{X}(t)$  active servers. Hence, it directly follows that  $P$  is a shortest path of the graph with  $\text{costs}(P) = \text{costs}(\mathcal{X})$ .

“ $\Leftarrow$ ”: Let  $P = ((0, 0) = v_0, \dots, v_T = (T, 0))$  with  $v_t \in \{(t, i) \mid 0 \leq i \leq m\}$  be a shortest path of the graph.

We can construct an optimal schedule from  $P$  by setting  $\mathcal{X} := (v_0(1), \dots, v_T(1))$

By definition (2) it is guaranteed that  $P$  only traverses edges such that there are enough active servers  $\forall t \in [T]$ . Therefore, the created schedule is feasible. Its optimality directly follows from the definition of the edges' weights and so does the equality  $\text{costs}(\mathcal{X}) = \text{costs}(P)$ .

□

### 3.2 A pseudo-polynomial minimum cost algorithm

---

**Algorithm 1** Calculate costs for  $m$  homogeneous servers

---

**Require:** Convex cost function  $f$ ,  $\lambda_0 = \lambda_T = 0$ ,  $\forall t \in [T - 1] : \lambda_t \in [0, m]$

```

1: function SCHEDULE( $m, T, \beta, \lambda_1, \dots, \lambda_{T-1}$ )
2:   if  $T < 2$  then return
3:   let  $p[2 \dots T - 1, m]$  and  $M[1 \dots T - 1, m]$  be new arrays
4:   for  $j \leftarrow 0$  to  $m$  do
5:      $M[1, j] \leftarrow d(0, j, \lambda_1)$ 
6:   for  $t \leftarrow 1$  to  $T - 2$  do
7:     for  $j \leftarrow 0$  to  $m$  do
8:        $opt \leftarrow \infty$ 
9:       for  $i \leftarrow 0$  to  $m$  do
10:         $M[t + 1, j] \leftarrow M[t, i] + d(i, j, \lambda_{t+1})$ 
11:        if  $M[t + 1, j] < opt$  then
12:           $opt \leftarrow M[t + 1, j]$ 
13:           $p[t + 1, j] \leftarrow i$ 
14:         $M[t + 1, j] \leftarrow opt$ 
15:   return  $p$  and  $M$ 

```

---



---

**Algorithm 2** Extract schedule for  $m$  homogeneous servers

---

```

1: function EXTRACT( $m, p, M, T$ )
2:   let  $x[0 \dots T]$  be a new array
3:    $x[0] \leftarrow x[T] \leftarrow 0$ 
4:   if  $T < 2$  then return  $x$  ▷ Trivial solution
5:    $x[T - 1] \leftarrow \arg \min_{0 \leq i \leq m} \{M[T - 1, i]\}$ 
6:   for  $t \leftarrow T - 2$  to  $1$  do
7:      $x[t] \leftarrow p[t + 1, x[t + 1]]$ 
8:   return  $x$ 

```

---

#### 3.2.1 Runtime analysis

The algorithm visits every vertex and every edge of the graph exactly once. As the number of vertices is bounded by  $\mathcal{O}(Tm)$  and the number of edges is bounded by  $\mathcal{O}(Tm^2)$  the running time is given by:

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

As we need  $\log_2(m)$  bits to encode  $m$ , the running time is polynomial in the numeric value of the input but exponential in the length of the input. Hence, the algorithm is pseudo-polynomial.

### 3.2.2 A memory optimized algorithm

TODO: use only array with size  $2m$

## 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.

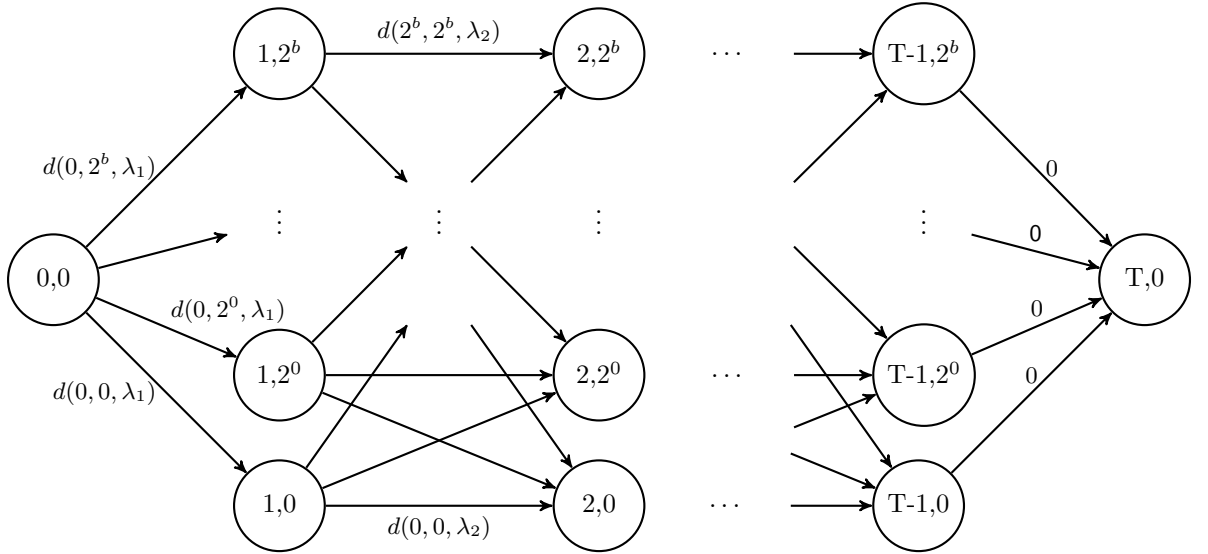


Figure 2: Graph for a 4-approximation algorithm

**Definition 4.1.** Let  $\mathcal{X} = (x_0, \dots, x_T)$  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

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



**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  $\text{costs}(\mathcal{X}') = \text{costs}(P)$ .
2. Any shortest path  $P$  from  $(0, 0)$  to  $(T, 0)$  corresponds to a 4-optimal schedule  $\mathcal{X}$  with  $\text{costs}(P) = \text{costs}(\mathcal{X})$ .

*Proof.*

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

$$x'_0 := 0, \quad x'_{t+1} := \begin{cases} \min\{2^{\lceil \log_2(2x_{t+1}) \rceil}, 2^b\}, & \text{if } 0 < x_t \leq x_{t+1} \\ 2^{\lceil \log_2(2x_{t+1}) \rceil}, & \text{if } 0 < x_{t+1} < x_t \text{ and } x'_t \geq 4x_{t+1} \\ x'_t, & \text{if } 0 < x_{t+1} < x_t \text{ and } x'_t < 4x_{t+1} \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

Then let  $\mathcal{X}' := (x'_0, \dots, x'_T)$  be the modified sequence of active servers. Notice that  $x_t \leq x'_t \leq 4x_t$  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:

$$\text{First set} \quad e_t := \left( (t, \mathcal{X}'(t)), (t+1, \mathcal{X}'(t+1)) \right), \quad \forall t \in \{0, \dots, T-1\}$$

$$\text{then set} \quad P := (e_0, \dots, e_{T-1})$$

By the definition of the edges' weights it follows that  $\text{costs}(\mathcal{X}') = \text{costs}(P)$ .

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  $2x_{t_i} \leq x'_{t_i}$  holds (TODO: only if not discrete but continuous time steps). This can be seen exemplarily in figure 3 by observing that  $\mathcal{X}'$  changes its state only if  $\mathcal{X}$  crosses or touches a bordering power of two.

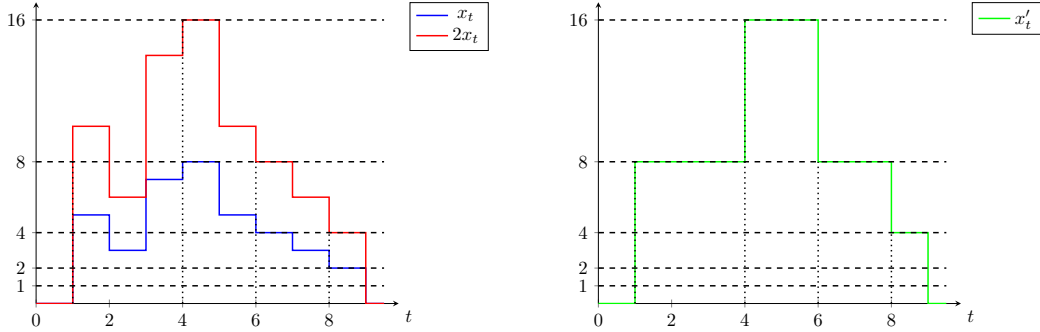


Figure 3: Adaption of an optimal schedule

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  $t_i$

$$\frac{\text{costs}(\mathcal{X}', t_{i-1}, t_i)}{\text{costs}(\mathcal{X}, t_{i-1}, t_i)} \quad (5)$$

For  $x_{t_i} = 0$  it follows from (1) that  $\text{costs}(\mathcal{X}', t_{i-1}, t_i) = \text{costs}(\mathcal{X}, t_{i-1}, t_i) = 0$ . 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

$$\begin{aligned}
\text{costs}(\mathcal{X}', t_{i-1}, t_i) &= \beta \max\{0, x'_{t_i} - x'_{t_{i-1}}\} + x'_{t_i} f(\lambda_{t_i}/x'_{t_i}) && \text{by (1)} \\
&\leq \beta \max\{0, x'_{t_i} - x'_{t_{i-1}}\} + 4x_{t_i} f(\lambda_{t_i}/x'_{t_i}) && \text{by (4)} \\
&\leq \beta \max\{0, x'_{t_i} - x'_{t_{i-1}}\} + 4x_{t_i} f(\lambda_{t_i}/x_{t_i}) && \text{f monotonically increasing} \\
\implies \text{costs}(\mathcal{X}', t_{i-1}, t_i) &\leq \beta \max\{0, x'_{t_i} - x'_{t_{i-1}}\} + 4x_{t_i} f(\lambda_{t_i}/x_{t_i}) && (6)
\end{aligned}$$

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

$$\text{costs}(\mathcal{X}, t_{i-1}, t_i) = \beta \max\{0, x_{t_i} - x_{t_{i-1}}\} + x_{t_i} f(\lambda_{t_i}/x_{t_i}) \quad (7)$$

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)  $x_{t_i} \leq x_{t_{i-1}}$ : From (4) it follows that  $x'_{t_i} \leq x'_{t_{i-1}}$ . Thus, we can simplify (5):

$$\begin{aligned}
\frac{\text{costs}(\mathcal{X}', t_{i-1}, t_i)}{\text{costs}(\mathcal{X}, t_{i-1}, t_i)} &\leq \frac{\beta \max\{0, x'_{t_i} - x'_{t_{i-1}}\} + 4x_{t_i} f(\lambda_{t_i}/x_{t_i})}{\beta \max\{0, x_{t_i} - x_{t_{i-1}}\} + x_{t_i} f(\lambda_{t_i}/x_{t_i})} && \text{by (6),(7)} \\
&= \frac{4x_{t_i} f(\lambda_{t_i}/x_{t_i})}{x_{t_i} f(\lambda_{t_i}/x_{t_i})} && (x_{t_i} \leq x_{t_{i-1}} \text{ and } x'_{t_i} \leq x'_{t_{i-1}}) \\
&= 4
\end{aligned}$$

(ii)  $x_{t_i} > x_{t_{i-1}}$ : From (4) it follows that  $x'_{t_i} \geq x'_{t_{i-1}}$ . Thus, we can simplify (5):

$$\begin{aligned}
\frac{\text{costs}(\mathcal{X}', t_{i-1}, t_i)}{\text{costs}(\mathcal{X}, t_{i-1}, t_i)} &\leq \frac{\beta \max\{0, x'_{t_i} - x'_{t_{i-1}}\} + 4x_{t_i} f(\lambda_{t_i}/x_{t_i})}{\beta \max\{0, x_{t_i} - x_{t_{i-1}}\} + x_{t_i} f(\lambda_{t_i}/x_{t_i})} && \text{by (6),(7)} \\
&= \frac{\beta(x'_{t_i} - x'_{t_{i-1}}) + 4x_{t_i} f(\lambda_{t_i}/x_{t_i})}{\beta(x_{t_i} - x_{t_{i-1}}) + x_{t_i} f(\lambda_{t_i}/x_{t_i})} && (x_{t_i} > x_{t_{i-1}} \text{ and } x'_{t_i} \geq x'_{t_{i-1}}) \\
&= \frac{\beta(\min\{2^{\lfloor \log_2(2x_{t_i}) \rfloor}, 2^b\} - x'_{t_{i-1}}) + 4x_{t_i} f(\lambda_{t_i}/x_{t_i})}{\beta(x_{t_i} - x_{t_{i-1}}) + x_{t_i} f(\lambda_{t_i}/x_{t_i})} && \text{by (4)} \\
&\leq \frac{\beta(2^{\lfloor \log_2(2x_{t_i}) \rfloor} - x'_{t_{i-1}}) + 4x_{t_i} f(\lambda_{t_i}/x_{t_i})}{\beta(x_{t_i} - x_{t_{i-1}}) + x_{t_i} f(\lambda_{t_i}/x_{t_i})} \\
&\leq \frac{\beta(2x_{t_i} - x'_{t_{i-1}}) + 4x_{t_i} f(\lambda_{t_i}/x_{t_i})}{\beta(x_{t_i} - x_{t_{i-1}}) + x_{t_i} f(\lambda_{t_i}/x_{t_i})} \\
&\leq \frac{\beta(2x_{t_i} - 2x_{t_{i-1}}) + 4x_{t_i} f(\lambda_{t_i}/x_{t_i})}{\beta(x_{t_i} - x_{t_{i-1}}) + x_{t_i} f(\lambda_{t_i}/x_{t_i})} && \text{by } (2x_{t_{i-1}} \leq x'_{t_{i-1}}) \\
&\leq 4 \frac{\frac{1}{2}\beta(x_{t_i} - x_{t_{i-1}}) + x_{t_i} f(\lambda_{t_i}/x_{t_i})}{\beta(x_{t_i} - x_{t_{i-1}}) + x_{t_i} f(\lambda_{t_i}/x_{t_i})} \\
&\leq 4
\end{aligned}$$

From (i) and (ii) it follows:

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

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  $\text{costs}(P) \leq \text{costs}(P') < \infty$ , and since every path  $P$  with  $\text{costs}(P) < \infty$  corresponds to a feasible schedule  $\mathcal{X}$  with  $\text{costs}(P) = \text{costs}(\mathcal{X})$ ,  $\mathcal{X}$  must also be at least 4-optimal.

□

## References

- [1] Minghong Lin, Adam Wierman, Lachlan L. H. Andrew, and Eno Thereska. Dynamic right-sizing for power-proportional 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:

- $m \in \mathbb{N}$ ... number of homogeneous servers
- $T \in \mathbb{N}$ ... number of time slots
- $\lambda_1, \dots, \lambda_T \in [0, m]$ ... arrival rates
- $\Lambda := (\lambda_1, \dots, \lambda_T)$ ... sequence of arrival rates
- $\beta \in \mathbb{R}_{\geq 0}$ ... switching costs of a server
- $f : [0, 1] \rightarrow \mathbb{R}$ ... convex operating costs function of a server
- $\mathcal{I} := (m, T, \Lambda, \beta, f)$ ... input of a problem instance
- $\lambda_{i,t} \in [0, 1]$ ... assigned load for server  $i$  at time  $t$
- $\mathcal{L} := (\lambda_{0,1}, \lambda_{2,1}, \dots, \lambda_{m,1}, \lambda_{1,2}, \dots, \lambda_{m,T})$ ... sequence of assigned loads
- $s_{i,t}$ ... state of server  $i$  at time  $t$ , i.e. sleeping (0) or active(1)
- $\mathcal{S} := (s_{1,1}, s_{2,1}, \dots, s_{m,1}, s_{1,2}, \dots, s_{m,T}) \in \{0, 1\}^{m \times T}$ ... sequence of states
- $\Sigma = (\mathcal{S}, \mathcal{L})$ ... a schedule for a problem instance  $\mathcal{I}$
- $x_t$ ... number of active servers at time  $t$
- $\mathcal{X} := (x_1, \dots, x_T)$ ... sequence of number of active servers
- $\lambda_t = 0$  for all  $t \notin [T]$
- All servers are powered down at time  $t = 0$