

# Algorithms for Dynamic Right-Sizing in Data Centers

Kevin Kappelmann

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

May 11, 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 offline scheduling</b>	<b>5</b>
3.1	Graph for an optimal schedule . . . . .	5
3.2	A pseudo-polynomial minimum cost algorithm . . . . .	7
3.2.1	Runtime analysis . . . . .	7
3.2.2	A memory optimized algorithm . . . . .	7
<b>4</b>	<b>A polynomial 4-approximation algorithm for monotonically increasing convex <math>f</math></b>	<b>8</b>
4.1	Graph for a 4-optimal schedule . . . . .	8

# 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 above-mentioned ever-growing energy consumption by examining 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  $\lambda_t$ , 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 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  $f : [0, 1] \rightarrow \mathbb{R}$ , 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,  $\beta$  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  $\beta$  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  $\beta$  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  $\beta := \beta'_{\uparrow} := \beta_{\uparrow} + \beta_{\downarrow}$  and  $\beta'_{\downarrow} = 0$ .

## 1.2 Problem statement

Using above definitions, we can define the input of our model by setting  $\mathcal{I} := (m, T, \Lambda, \beta, f)$  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

$$S_i := (s_{i,1}, \dots, s_{i,T}) \in \{0, 1\}^T$$

$$L_i := (\lambda_{i,1}, \dots, \lambda_{i,T}) \in [0, 1]^T$$

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:

$$\mathcal{S} := (S_1, \dots, S_m)$$

$$\mathcal{L} := (L_1, \dots, L_m)$$

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  $\Sigma$  that satisfies the following optimisation:

Definition of  $c(\Sigma)$   
too hidden?

$$\begin{aligned} \text{minimise} \quad & c(\Sigma) := \underbrace{\sum_{t=1}^T \sum_{i=1}^m (f(\lambda_{i,t}) * s_{i,t})}_{\text{operating costs}} + \beta * \underbrace{\sum_{t=1}^T \sum_{i=1}^m \min\{0, s_{i,t} - s_{i,t-1}\}}_{\text{switching costs}} \quad (1) \\ \text{subject to} \quad & \sum_{i=1}^m (\lambda_{i,t} * s_{i,t}) = \lambda_t, \quad \forall t \in [T] \quad (2) \end{aligned}$$

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

## 2 Preliminaries

In this section, we conduct the preparatory work that will lay the foundations for our algorithms. For this, we analyse the structure of feasible schedules concerning 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 examining 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  $\Sigma$ , there exists a feasible schedule  $\Sigma'$  such that*

- (i)  $c(\Sigma') \leq c(\Sigma)$  and

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

$$\forall t \in [T] \left[ \left( \forall i \in [m] (s_{i,t} - s_{i,t-1} \geq 0) \right) \vee \left( \forall i \in [m] (s_{i,t} - s_{i,t-1} \leq 0) \right) \right] \quad (3)$$

*Proof.* Let  $\Sigma = (\mathcal{S}, \mathcal{L})$  be a feasible schedule for  $\mathcal{I}$ . We give a procedure that repeatedly modifies  $\Sigma$  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  $j$ :

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

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

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

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

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

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

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

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

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

Moreover, we decreased the amount of bad spots at time slot  $t$  concerning (3). Hence, by repeating described process on  $\Sigma'$ , we obtain a terminating procedure that returns a schedule satisfying the conditions.  $\square$

bad spots? better description?

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

*Proof.* Let  $\Sigma$  be an optimal schedule for  $\mathcal{I}$ . Applying proposition 2.1 to  $\Sigma$  yields the result.  $\square$

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  $t$ , i.e.

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

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

The next proposition poses the cornerstone of our subsequent works.

**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  $\lambda_t/x_t$ .*

This is sounds awkwardly formulated, doesn't it?

*Proof.* Let  $\Sigma$  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  $A := \{i \in [m] \mid s_{i,t} = 1\}$ . Consider the operating costs of  $\Sigma$  given by

$$\sum_{i=1}^m (f(\lambda_{i,t}) * s_{i,t}) = \sum_{i \in A} f(\lambda_{i,t})$$

Since  $\Sigma$  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  $\lambda_t$  for  $i \in A$  such that

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

In particular, we have  $\sum_{i=1}^{x_t} \mu_i \lambda_t = \lambda_t$ . Using these weights, we now consider the operating costs of a schedule  $\Sigma^*$  that equally distributes  $\lambda_t$  to its  $x_t$  active servers:

$$\sum_{i=1}^{x_t} f\left(\frac{\lambda_t}{x_t}\right) = x_t * f\left(\frac{\lambda_t}{x_t}\right) = x_t * f\left(\sum_{i=1}^{x_t} \frac{\mu_i \lambda_t}{x_t}\right)$$

Using Jensen's inequality<sup>1</sup> 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?

$$x_t * f\left(\frac{\lambda_t}{x_t}\right) \leq x_t \sum_{i=1}^{x_t} \frac{1}{x_t} f(\mu_i \lambda_t) = \frac{x_t}{x_t} \sum_{i=1}^{x_t} f(\mu_i \lambda_t) = \sum_{i=1}^{x_t} f(\mu_i \lambda_t) = \sum_{i \in A} f(\lambda_{i,t})$$

<sup>1</sup>For convex  $f : \mathbb{R} \rightarrow \mathbb{R}$ , 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:

$$f\left(\sum_{i=1}^n x_i \lambda_i\right) \leq \sum_{i=1}^n x_i f(\lambda_i)$$

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

Proposition 2.3 tells us that there always exists an optimal schedule that equally distributes its arrival rate to its active servers at any time slot. As a result, we can restrict ourselves in finding such an optimal schedule. Together with corollary 2.2, this allows us to subsequently identify an optimal schedule by its sequence of active servers  $\mathcal{X}$ . Moreover, we are 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  $\lambda$  on  $x$  active servers using  $f$ :

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

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

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

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

$$\text{minimise } c(\Sigma) = \sum_{t=1}^T c(x_{t-1}, x_t, \lambda_t)$$

### 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:

$$\begin{aligned} V &:= \{v_{i,t} \mid i \in \{0, \dots, m\}, t \in \{0, \dots, T+1\}\} \\ E &:= \{(v_{i,t}, v_{j,t+1}) \mid i, j \in \{0, \dots, m\}, t \in \{0, \dots, T\}, v_{i,t}, v_{j,t+1} \in V\} \\ c_G(v_{i,t}, v_{j,t+1}) &:= c(i, j, \lambda_{t+1}) \\ G &:= (V, E, c_G) \end{aligned}$$

For any possible amount of active servers  $i$  and any time slot  $t$  we add a node  $v_{i,t}$ . 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,  $v_{i,t}$  denotes the state of scheduling the arrival rate  $\lambda_t$  equally on  $i$  servers.

Maybe something different than  $c_G$ ?

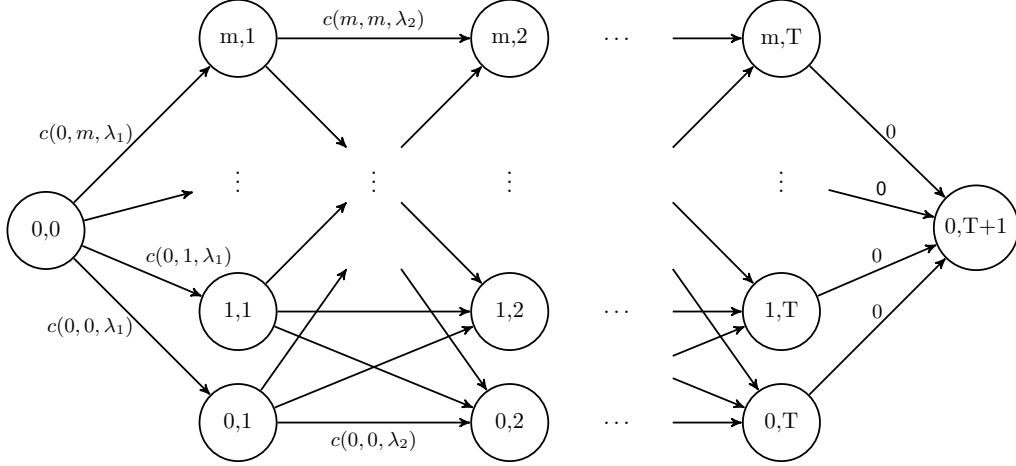


Figure 1: Level structured graph for an optimal offline algorithm.

We now want to verify the correctness of our construction.

**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  $x$   $\triangleright$  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.

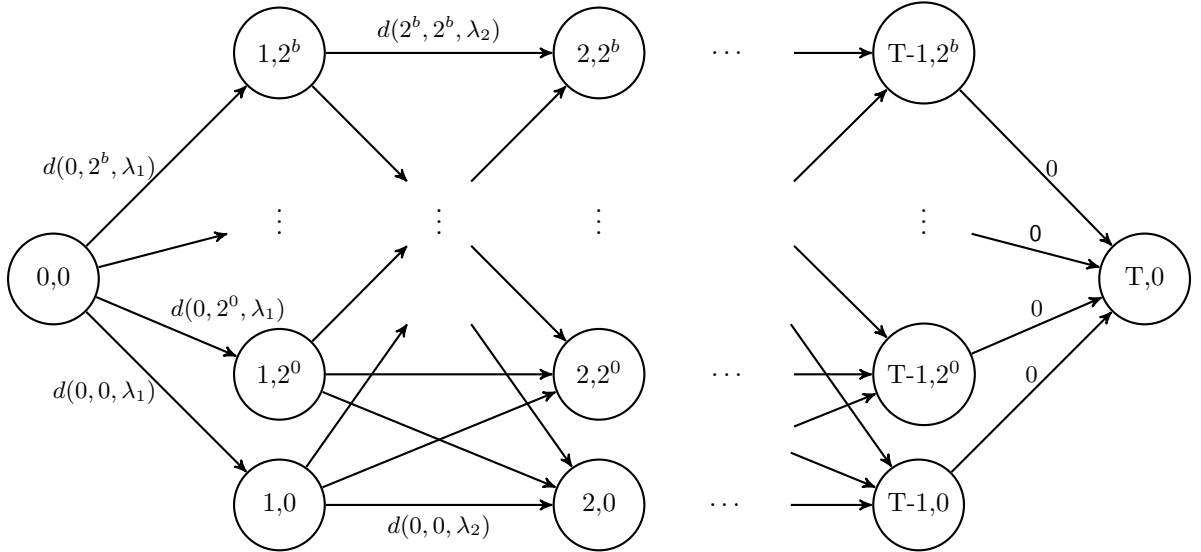


Figure 2: Graph for a 4-approximation algorithm

**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:

$$\begin{aligned} \text{First set} \quad e_t &:= \left( (t, \mathcal{X}'(t)), (t+1, \mathcal{X}'(t+1)) \right), \quad \forall t \in \{0, \dots, T-1\} \\ \text{then set} \quad P &:= (e_0, \dots, e_{T-1}) \end{aligned}$$

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.

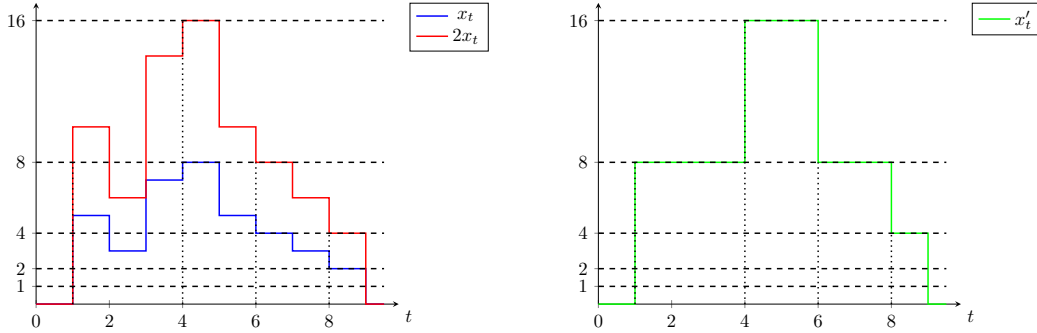


Figure 3: Adaption of an optimal schedule

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 (??) 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

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

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:

Good idea to have an appendix?

- Input:
  - $m \in \mathbb{N}$ ... 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
- Problem statement:
  - $s_{i,t}$ ... state of server  $i$  at time  $t$ , i.e. sleeping (0) or active(1)
  - $S_i := (s_{i,1}, \dots, s_{i,T}) \in \{0, 1\}^T$ ... sequence of states for server  $i$
  - $\lambda_{i,t} \in [0, 1]$ ... assigned load for server  $i$  at time  $t$
  - $L_i := (\lambda_{i,1}, \dots, \lambda_{i,T}) \in [0, 1]^T$ ... sequence of assigned loads for server  $i$
  - $\mathcal{S} := (S_1, \dots, S_m)$ ... sequence of all state changes
  - $\mathcal{L} := (L_1, \dots, L_m)$ ... sequence of all assigned loads
  - $\Sigma := (\mathcal{S}, \mathcal{L})$ ... schedule for a problem instance  $\mathcal{I}$
- Miscellaneous:
  - $x_t$ ... number of active servers at time  $t$
  - $\mathcal{X} := (x_1, \dots, x_T)$ ... 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