Algorithms for Dynamic Right-Sizing in Data Centers

Kevin Kappelmann

Chair for Theoretical Computer Science,
Technical University of Munich

May 6, 2017

Contents

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 examing 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*).

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 $i \in [m]$ 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 $f : [0,1] \to \mathbb{R}$. Further, we assume convexity for f; this may seem like a notable restriction at first, but indeed captures the behaviour of most modern server models. 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, like lost revenue as users need to wait for their responses. Similarly, β may also allow for delay costs, wear and tear costs or the like. [1] As we are dealing with homogeneous servers, β and f 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 $t > T$. This justifies the consolidation of power up and power down costs into β 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 $\beta \coloneqq \beta_\uparrow + \beta_\downarrow$. 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 $\mathcal{I} \coloneqq (m, T, \Lambda, \beta, f)$ where $\Lambda = (\lambda_1, \ldots, \lambda_T)$ 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

$$\mathcal{X} \coloneqq (x_1, \ldots, x_T) \in \{0, \ldots, m\}^T$$
$$\mathcal{L} \coloneqq (\lambda_{1,1}, \lambda_{2,1}, \ldots, \lambda_{m,1}, \lambda_{1,2}, \ldots, \lambda_{m,T}) \in [0,1]^{m*T}$$

where x_t denotes the number of active servers at time t and $\lambda_{i,t}$ the load assigned to server i at time t (as described in section 1.1).

$$\mathcal{S}^* = (s_{1,1}, s_{2,1}, \ldots, s_{m,1}, s_{1,2}, \ldots, s_{m,T}) \in \{0,1\}^{m*T}$$
$$\mathcal{L}^* = (\lambda_{1,1}, \lambda_{2,1}, \ldots, \lambda_{m,1}, \lambda_{1,2}, \ldots, \lambda_{m,T}) \in [0,1]^{m*T}$$

where $\mathcal{S}$ is the series of power states and $\mathcal{L}$ is the series of assigned loads for our servers subject to

$$(\mathcal{S}^*, \mathcal{L}^*) = \underset{(\mathcal{S},\mathcal{L})}{\arg\min}\{| \sum_{t=1}^{T} \lambda_t = \sum_{i=1}^{m} \sum_{t=1}^{T} \lambda_{i,t} * s_{i,t}\}$$

1.3 Overview

Below, we give an overview of just given definitions and conventions:

- $m \in \mathbb{N} \ldots$ number of homogeneous servers
- $T \in \mathbb{N} \ldots$ number of time slots
- $\lambda_1, \ldots, \lambda_T \in [0, m] \ldots$ arrival rates
- $\Lambda := (\lambda_1, \ldots, \lambda_T) \ldots$ sequence of arrival rates
- $\lambda_{i,t} \ldots$ assigned arrival rate for server i at time t
- $\beta \in \mathbb{R}_{\geq 0} \ldots$ switching costs of a server
- $f : [0,1] \to \mathbb{R} \ldots$ convex operating costs function of a server
- $x_t \ldots$ number of active servers at time t
- $\mathcal{X} := (x_1, \ldots, x_T) \ldots$ sequence of active servers
- $\mathcal{I} := (m, T, \Lambda, \beta, f) \ldots$ input of problem instance
- $\lambda_t = 0$ for all $t \notin [T]$
- All servers are powered down at time $t = 0$

2 Preliminaries

2.1 Feasible schedules

2.2 Input

- $m \in \mathbb{N} \ldots$ the number of homogeneous servers
- $\beta \in \mathbb{R}_{\geq 0} \ldots$ the power-up costs of a single server
- $T \in \mathbb{N} \ldots$ the number of time steps
- $\Lambda = (\lambda_1, \ldots, \lambda_T) \in [0, m]^T \ldots$ the sequence of arrival rates

2.3 Definitions and conventions

Definition 2.1. Let $\mathcal{X} = (x_0, \ldots, x_T)$ be the sequence of active servers of a schedule. We call a schedule and its sequence **feasible** if

$$\forall t \in \{0, \ldots, T\} : x_t \geq \lambda_t$$

We call a feasible schedule **optimal** if its incurred costs are minimal under all feasible schedules.

Lemma 2.2. *Given a convex cost function f, x active servers and an arrival rate λ, the optimal strategy is to assign each server a load of λ/x.*

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

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

□

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

$$
costs(\mathcal{X}, t, t') \coloneqq \beta \max\{0, x_{t'} - x_t\} + x_{t'} f(\lambda_{t'}/x_{t'}) \tag{1}
$$

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

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

and the total costs by

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

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 $i, j \in \{0, \ldots, m\}$ 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 λ with x active servers:

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

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

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

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 $d(i, 0, \lambda_T) = 0$.

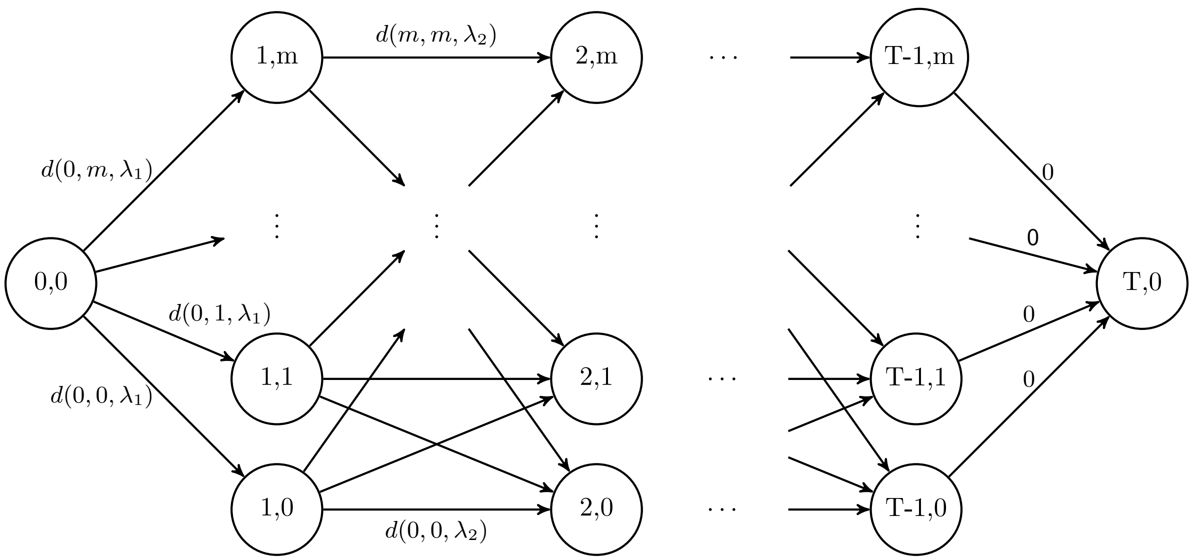

Figure 1: Graph for optimal schedule algorithm.

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 $costs(\mathcal{X}) = costs(P)$ and vice versa.*

Proof.

"⇒": We construct a feasible path in our graph from $\mathcal{X}$ as follows:

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

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

"$\Leftarrow$": Let $P = \big((0,0) = v_0, \dots, v_T = (T,0)\big)$ with $v_t \in \big\{(t,i) \mid 0 \leq i \leq m\big\}$ be a shortest path of the graph.
We can construct an optimal schedule from P by setting $\mathcal{X} := \big(v_0(1), \dots, v_T(1)\big)$
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 $costs(\mathcal{X}) = costs(P)$.

$\square$

3.2 A pseudo-polynomial minimum cost algorithm

Algorithm 1 Calculate costs for m homogeneous servers

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

function SCHEDULE$(m, T, \beta, \lambda_1, \dots, \lambda_{T-1})$
    if $T < 2$ **then return**
    let $p[2 \dots T-1, m]$ and $M[1 \dots T-1, m]$ **be** new arrays
    for $j \leftarrow 0$ to m **do**
        $M[1, j] \leftarrow d(0, j, \lambda_1)$
    for $t \leftarrow 1$ to $T-2$ **do**
        for $j \leftarrow 0$ to m **do**
            $opt \leftarrow \infty$
            for $i \leftarrow 0$ to m **do**
                $M[t+1, j] \leftarrow M[t, i] + d(i, j, \lambda_{t+1})$
                if $M[t+1, j] < opt$ **then**
                    $opt \leftarrow M[t+1, j]$
                    $p[t+1, j] \leftarrow i$
            $M[t+1, j] \leftarrow opt$
    return p and M

Algorithm 2 Extract schedule for m homogeneous servers

function EXTRACT(m, p, M, T)
    let $x[0 \dots T]$ **be** a new array
    $x[0] \leftarrow x[T] \leftarrow 0$
    if $T < 2$ **then return** x ▷ Trivial solution
    $x[T-1] \leftarrow \underset{0 \leq i \leq m}{arg\ min}\{M[T-1, i]\}$
    for $t \leftarrow T-2$ to 1 **do**
        $x[t] \leftarrow p[t+1, x[t+1]]$
    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 $b \coloneqq \lceil \log_2(m) \rceil$. 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.

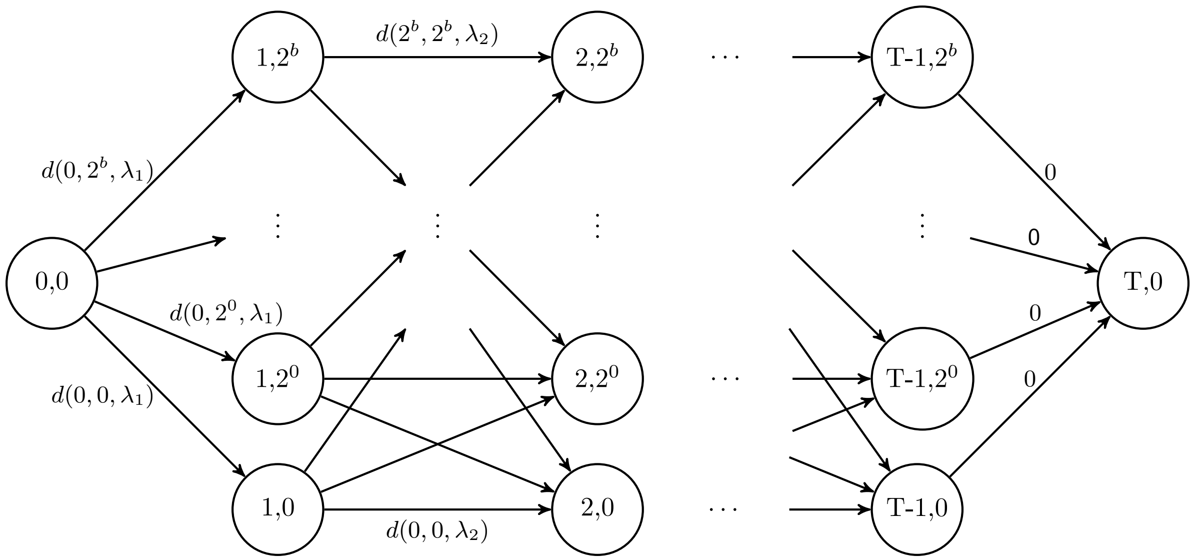

Figure 2: Graph for a 4-approximation algorithm

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

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

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 $0 \leq t < T$ we inductively set:

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

Then let $\mathcal{X}' \coloneqq (x'_0, \dots, x'_T)$ 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:

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

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.

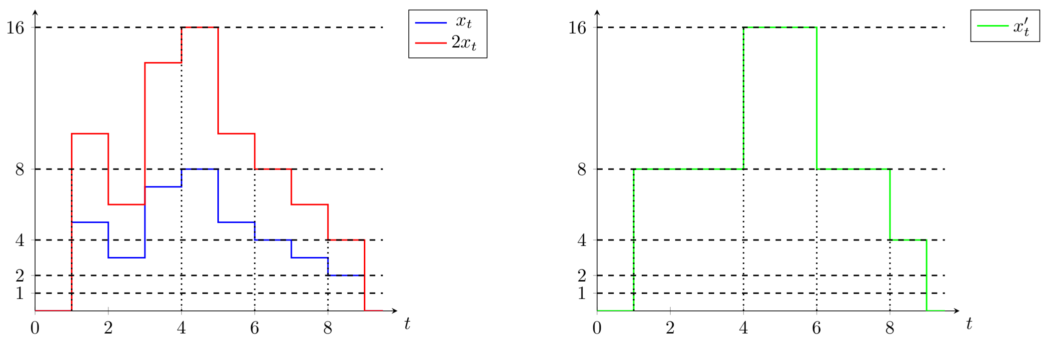

Figure 3: Adaption of an optimal schedule

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

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

For $x_{t_i} = 0$ it follows from (1) 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

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

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

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

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

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

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

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

From (i) and (ii) it follows:

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

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.

□

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.