# Algorithms for Dynamic Right-Sizing in Data Centers

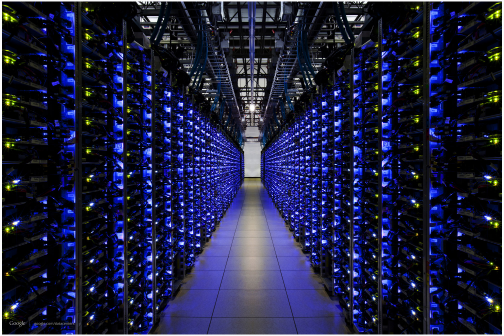
Kevin Kappelmann

August 03, 2017

Technical University of Munich

# Dynamic Right-Sizing

Dynamically adjust the number of active servers and efficiently distribute the varying workloads



# Model and Problem Formulation

# Model Description

Input:

- $m \in \mathbb{N} \ldots$ Number of homogeneous servers

## Model Description

Input:

- $m \in \mathbb{N}$ ... Number of homogeneous servers
- $\beta \in \mathbb{R}_{\geq 0}$ ... Switching costs of a server

# Model Description

Input:

- $m \in \mathbb{N}$ ... Number of homogeneous servers
- $\beta \in \mathbb{R}_{\geq 0}$ ... Switching costs of a server
- $f : [0, 1] \to \mathbb{R}$ ... Convex operating cost function of a server

# Model Description

Input:

- $m \in \mathbb{N}$ ... Number of homogeneous servers
- $\beta \in \mathbb{R}_{\geq 0}$ ... Switching costs of a server
- $f : [0, 1] \to \mathbb{R}$ ... Convex operating cost function of a server
- $T \in \mathbb{N}$ ... Number of time slots

# Model Description

Input:

- $m \in \mathbb{N}$ ... Number of homogeneous servers
- $\beta \in \mathbb{R}_{\geq 0}$ ... Switching costs of a server
- $f : [0, 1] \to \mathbb{R}$ ... Convex operating cost function of a server
- $T \in \mathbb{N}$ ... Number of time slots
- $\lambda_1, \dots, \lambda_T \in [0, m]$ ... Arrival rates (workloads)

# Model Description

Input:

- $m \in \mathbb{N}$ ... Number of homogeneous servers
- $\beta \in \mathbb{R}_{\geq 0}$ ... Switching costs of a server
- $f : [0, 1] \to \mathbb{R}$ ... Convex operating cost function of a server
- $T \in \mathbb{N}$ ... Number of time slots
- $\lambda_1, \dots, \lambda_T \in [0, m]$ ... Arrival rates (workloads)

Conventions:

- $x_1, \dots, x_T \in \{0, \dots, m\}$ ... Numbers of active servers

# Model Description

Input:

- $m \in \mathbb{N}$ ... Number of homogeneous servers
- $\beta \in \mathbb{R}_{\geq 0}$ ... Switching costs of a server
- $f : [0, 1] \to \mathbb{R}$ ... Convex operating cost function of a server
- $T \in \mathbb{N}$ ... Number of time slots
- $\lambda_1, \dots, \lambda_T \in [0, m]$ ... Arrival rates (workloads)

Conventions:

- $x_1, \dots, x_T \in \{0, \dots, m\}$ ... Numbers of active servers
- All servers are shut down before and after the scheduling process.

# Problem Statement

Operating costs for one time step $t$

$$c_{op}(x_t, \lambda_t) := \begin{cases} \end{cases}$$

# Problem Statement

Operating costs for one time step $t$

$$c_{op}(x_t, \lambda_t) := \begin{cases} \infty, & \text{if } \lambda_t > x_t \qquad \text{//too few servers} \\ & \\ & \end{cases}$$

# Problem Statement

Operating costs for one time step $t$

$$c_{op}(x_t, \lambda_t) := \begin{cases} \infty, & \text{if } \lambda_t > x_t & //\text{too few servers} \\ 0, & \text{if } x_t = \lambda_t = 0 & //\text{no active servers} \\ & \end{cases}$$

# Problem Statement

Operating costs for one time step $t$

$$c_{op}(x_t, \lambda_t) := \begin{cases} \infty, & \text{if } \lambda_t > x_t & //\text{too few servers} \\ 0, & \text{if } x_t = \lambda_t = 0 & //\text{no active servers} \\ x_t f(\lambda_t / x_t), & \text{otherwise} & //\text{even distribution} \end{cases}$$

# Problem Statement

Operating costs for one time step $t$

$$c_{op}(x_t, \lambda_t) := \begin{cases} \infty, & \text{if } \lambda_t > x_t & //\text{too few servers} \\ 0, & \text{if } x_t = \lambda_t = 0 & //\text{no active servers} \\ x_t f(\lambda_t / x_t), & \text{otherwise} & //\text{even distribution} \end{cases}$$

**Goal: Minimize total costs**

# Problem Statement

Operating costs for one time step $t$

$$c_{op}(x_t, \lambda_t) := \begin{cases} \infty, & \text{if } \lambda_t > x_t & \text{//too few servers} \\ 0, & \text{if } x_t = \lambda_t = 0 & \text{//no active servers} \\ x_t f(\lambda_t / x_t), & \text{otherwise} & \text{//even distribution} \end{cases}$$

**Goal: Minimize total costs**

$$\text{minimize} \quad \sum_{t=1}^{T} \Big( \underbrace{c_{op}(x_t, \lambda_t) + \beta \max\{0, x_t - x_{t-1}\}}_{c(x_{t-1}, x_t, \lambda_t)} \Big)$$

# Optimal Offline Algorithm

# Pseudo-Linear-Time Algorithm

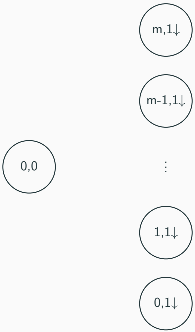
Fundamental idea: reduce problem to shortest path problem

# Pseudo-Linear-Time Algorithm

Fundamental idea: reduce problem to shortest path problem

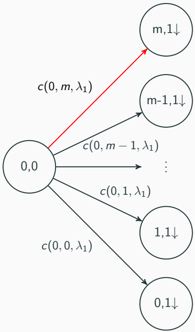
# Pseudo-Linear-Time Algorithm

Fundamental idea: reduce problem to shortest path problem



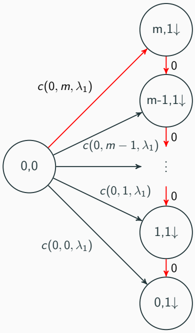
# Pseudo-Linear-Time Algorithm

Fundamental idea: reduce problem to shortest path problem



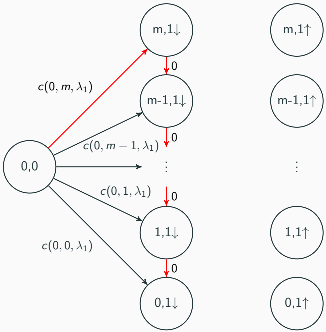
# Pseudo-Linear-Time Algorithm

Fundamental idea: reduce problem to shortest path problem



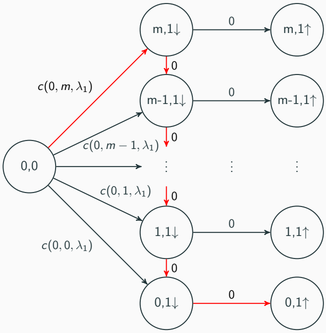
# Pseudo-Linear-Time Algorithm

Fundamental idea: reduce problem to shortest path problem



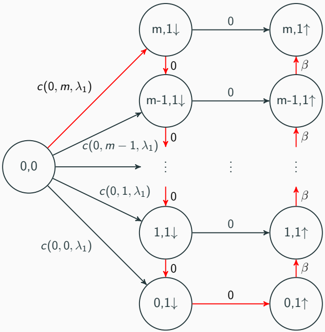
# Pseudo-Linear-Time Algorithm

Fundamental idea: reduce problem to shortest path problem



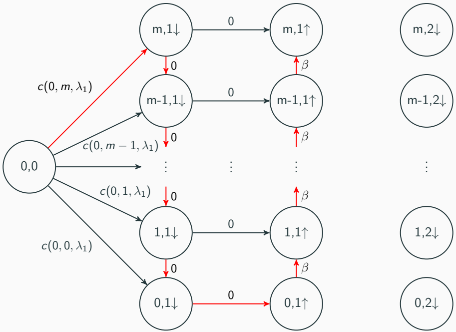
# Pseudo-Linear-Time Algorithm

Fundamental idea: reduce problem to shortest path problem



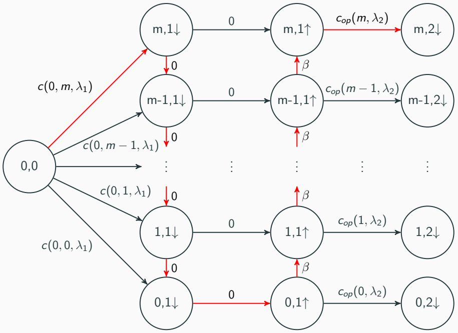
# Pseudo-Linear-Time Algorithm

Fundamental idea: reduce problem to shortest path problem



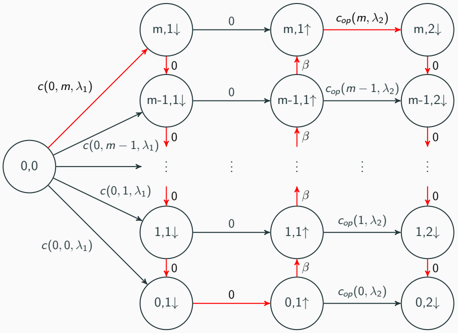
# Pseudo-Linear-Time Algorithm

Fundamental idea: reduce problem to shortest path problem



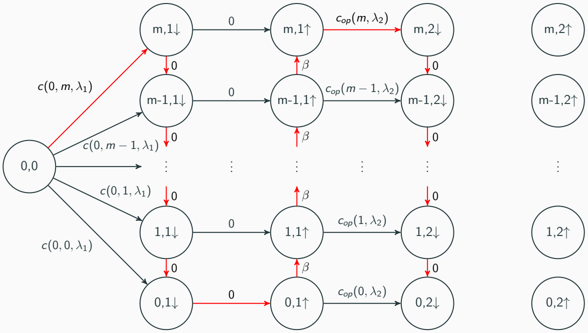
# Pseudo-Linear-Time Algorithm

Fundamental idea: reduce problem to shortest path problem



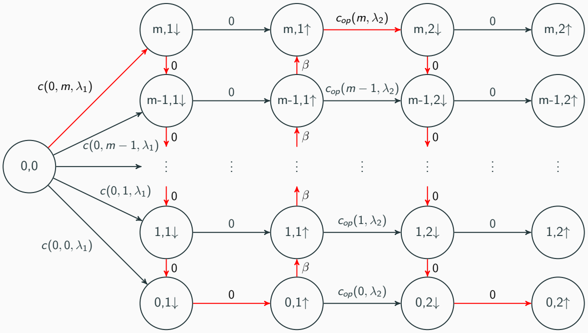
# Pseudo-Linear-Time Algorithm

Fundamental idea: reduce problem to shortest path problem



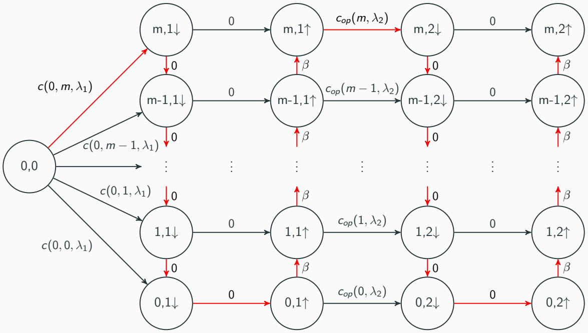
# Pseudo-Linear-Time Algorithm

Fundamental idea: reduce problem to shortest path problem



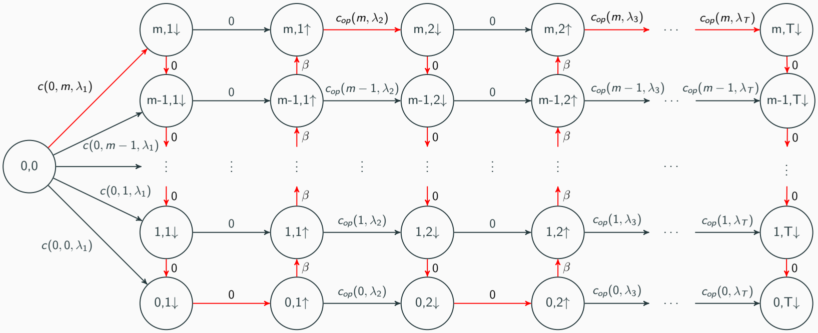
# Pseudo-Linear-Time Algorithm

Fundamental idea: reduce problem to shortest path problem



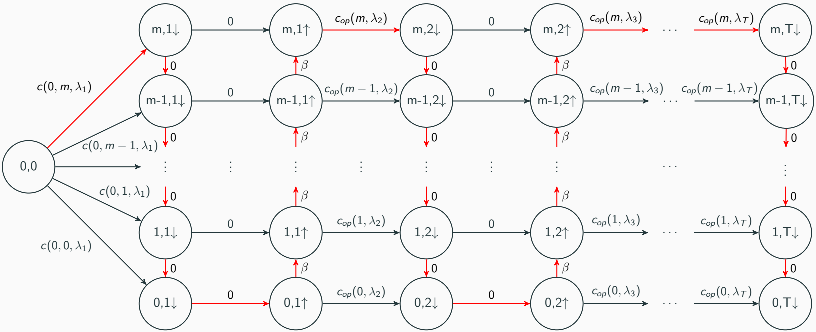
# Pseudo-Linear-Time Algorithm

Fundamental idea: reduce problem to shortest path problem



# Pseudo-Linear-Time Algorithm

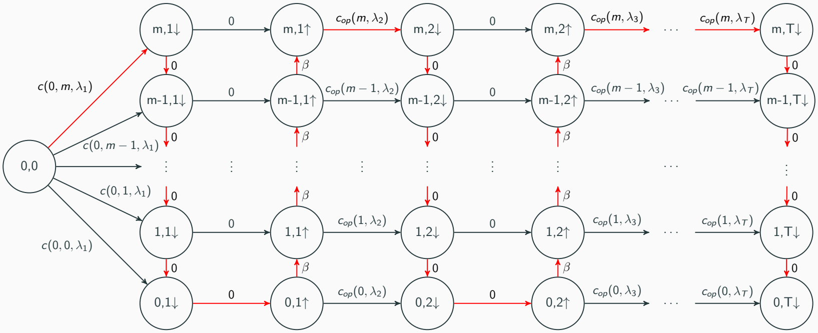
Fundamental idea: reduce problem to shortest path problem




Time complexity: $\Theta(Tm)$

# Pseudo-Linear-Time Algorithm

Fundamental idea: reduce problem to shortest path problem




Time complexity: $\Theta(Tm)$

. . . only $\log_2(m)$ bits required to encode $m$.

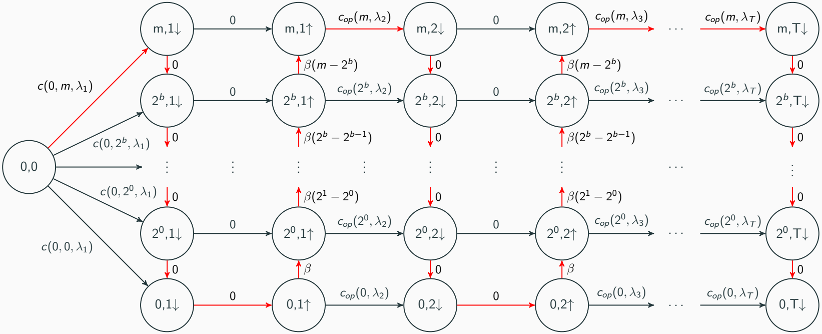
# Offline Approximation Algorithm

# 2-Optimal Linear-Time Algorithm

Use logarithmic steps to reduce number of nodes

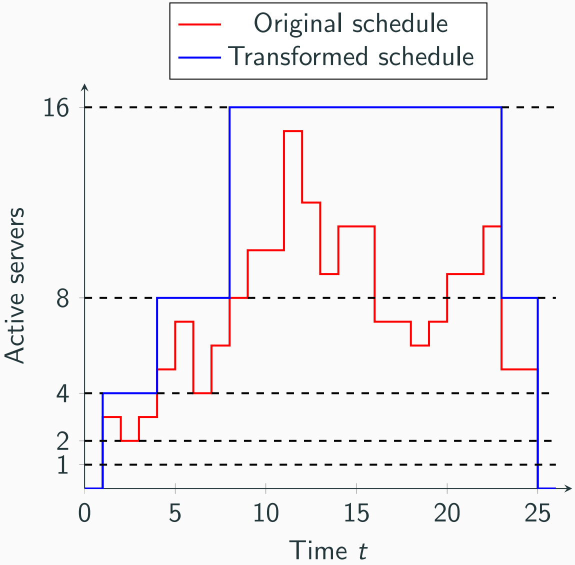
# 2-Optimal Linear-Time Algorithm

Use logarithmic steps to reduce number of nodes




where $b := \lfloor \log_2(m) \rfloor$

# 2-Optimal Linear-Time Algorithm



# Approximative Scheduling

A shortest path in the graph corresponds to a 2-optimal schedule.

# Approximative Scheduling

A shortest path in the graph corresponds to a 2-optimal schedule.

Time complexity: $\Theta\big(T \log_2(m)\big)$

# Approximative Scheduling

A shortest path in the graph corresponds to a 2-optimal schedule.

Time complexity: $\Theta\big(T \log_2(m)\big)$

Can we generalize to allow for arbitrary precisions?

# Approximative Scheduling

A shortest path in the graph corresponds to a 2-optimal schedule.

Time complexity: $\Theta\big(T \log_2(m)\big)$

Can we generalize to allow for arbitrary precisions? Yes!

# Approximative Scheduling

A shortest path in the graph corresponds to a 2-optimal schedule.

Time complexity: $\Theta\big(T \log_2(m)\big)$

Can we generalize to allow for arbitrary precisions? Yes!

Approximation for arbitrary precision $\varepsilon > 0$ with
time complexity $\Theta\big(T \log_{1+\varepsilon}(m)\big)$

# Summary and Prospects

# Summary

We reduced the scheduling problem to the shortest path problem of acyclic graphs.

# Summary

We reduced the scheduling problem to the shortest path problem of acyclic graphs.

Optimal Offline Algorithm with pseudo-linear runtime

$$\Theta(Tm)$$

## Summary

We reduced the scheduling problem to the shortest path problem of acyclic graphs.

Optimal Offline Algorithm with pseudo-linear runtime

$$\Theta(Tm)$$

$(1+\varepsilon)$-Optimal Offline Algorithm with linear runtime

$$\Theta\big(T \log_{1+\varepsilon}(m)\big)$$

# Prospects

Can our approach be modified to . . .

## Prospects

Can our approach be modified to ...

- deal with more than one homogeneous server collection?

# Prospects

Can our approach be modified to . . .

- deal with more than one homogeneous server collection?
- deal with multiple sleep states?

## Prospects

Can our approach be modified to ...

- deal with more than one homogeneous server collection?
- deal with multiple sleep states?
- work as an online algorithm?

# Prospects

Can our approach be modified to ...

- deal with more than one homogeneous server collection?
- deal with multiple sleep states?
- work as an online algorithm?

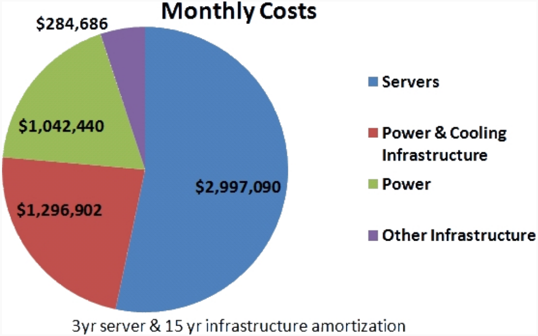
Open question: Is there a polynomial-time optimal algorithm or is it an NP-hard problem?

**Thanks for your attention!**

**Any questions?**

# Data Centers' Costs

Fast-growing demand for data collection, processing, and storage

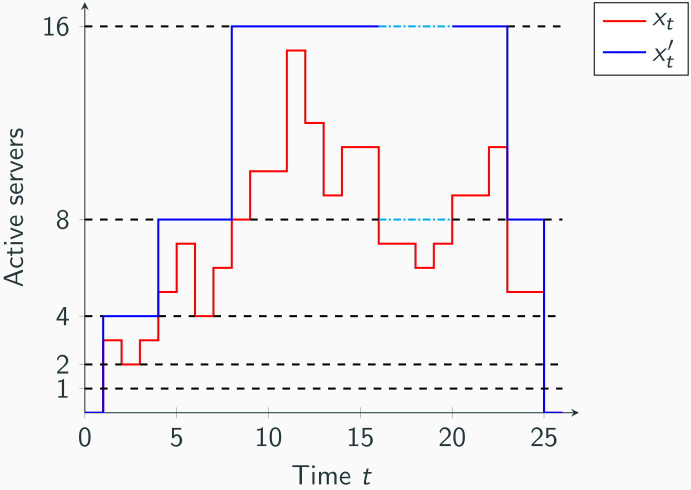


# Proof Idea

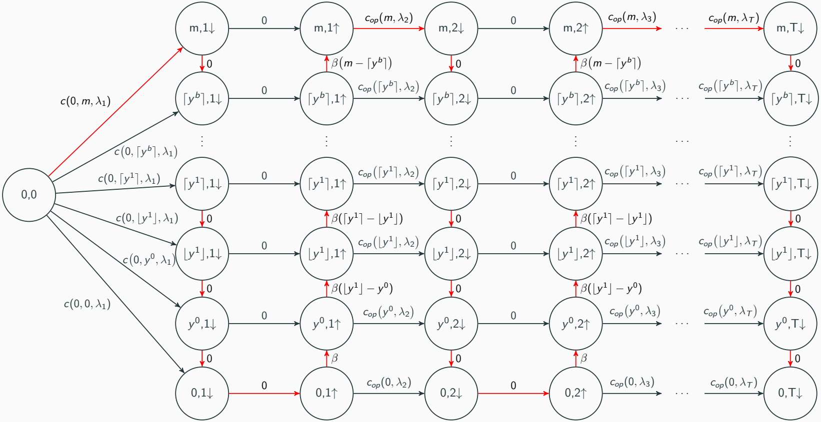
Take a schedule and transform periods between two powers of 2

# Proof Idea

Take a schedule and transform periods between two powers of 2



# $(1+\varepsilon)$-Optimal Offline Algorithm




where $y := 1+\varepsilon$, $b := \lfloor \log_y(m) \rfloor$

# Image Sources I

- Data center: `datacentervoice.com/wp-content/uploads/2015/12/data-center.jpg`
- Data center costs: `perspectives.mvdirona.com/2008/11/cost-of-power-in-large-scale-data-centers/`