1 Optimal scheduling for m homogeneous servers

TODO: introduction text

1.1 Input and conventions

TODO: rewrite in nice paragraph
Input:

- $m \in \mathbb{N}$: Number of homogeneous servers
- $T \in \mathbb{N}$: Number of time steps
- $\beta \in \mathbb{R}_{\geq 0}$: Power up costs
- $\lambda_0, \dots, \lambda_T \in [0, m]$: Arrival rates

Notations:

- Let $\lambda_{i,t}$ be the assigned arrival rate at time t for server i
- Let x_t be the number of active servers at time t
- Let $\mathcal{X} := (x_0, \dots, x_T)$ be the sequence of active servers

Requirements:

- Convex cost function f
- Power down costs are w.l.o.g. equal to 0
- $\forall t \in \{0, \dots, T\} : \sum_{i=1}^{m} \lambda_{i,t} = \lambda_t$
- $\lambda_0 = \lambda_T = 0$
- $\mathcal{X}(0) = \mathcal{X}(T) = 0$, i.e. all servers are powered down at $t = 0$ and $t = T$

1.2 Preliminaries

Lemma 1.1. *Given a convex cost function f, x active servers and an arrival rate λ, the best method is to assign each server a load of λ/x.*

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

$$
\begin{aligned}
f\Big(\frac{\lambda}{x}\Big) = f\Big(\sum_{i=1}^{x} \frac{\mu_i * \lambda}{x}\Big) &\overset{\text{Jensen's inequality}}{\leq} \sum_{i=1}^{x} \frac{1}{x} f(\mu_i * \lambda) \\
\Leftrightarrow \quad x * f\Big(\frac{\lambda}{x}\Big) &\leq \sum_{i=1}^{x} f(\mu_i * \lambda)
\end{aligned}
$$

□

We will use this fact in our following construction for an optimal schedule.

1.3 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. Furthermore, 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:

$$c(x, \lambda) \coloneqq \begin{cases} x * f(\lambda/x), & \text{if } \lambda \leq x \\ \infty, & \text{otherwise} \end{cases} \tag{1}$$

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

$$d(i, j, \lambda_{t+1}) \coloneqq \underbrace{\beta * \min\{0, j-i\}}_{\text{power up costs}} + c(j, \lambda_{t+1}) \tag{2}$$

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

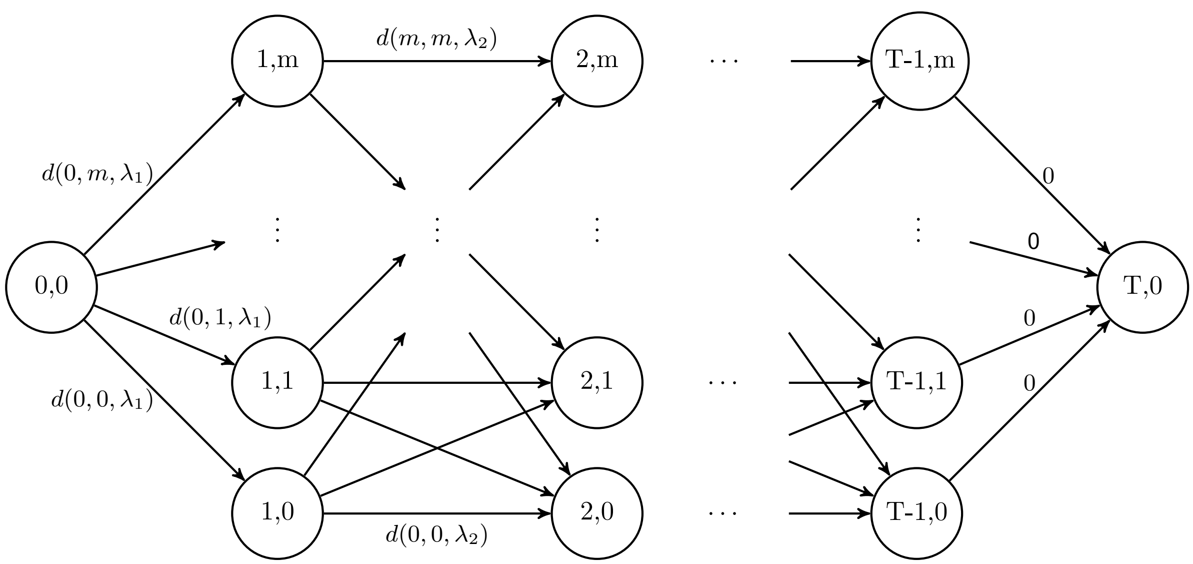

Figure 1: Graph for optimal schedule algorithm.

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

1.4 Proof of correctness

Proposition 1.2. *Any given optimal schedule corresponds to a shortest path from* $(0, 0)$ *to* $(T, 0)$ *in the constructed graph and vice versa.*

Proof.

"$\Rightarrow$": Lemma 1.1 shows that in an optimal schedule each arrival rate λ_t will be shared equally on each active server at time t. Therefore, we can denote an optimal schedule uniquely by the sequence $\mathcal{X}$ of active servers.
We can construct a valid path in our graph from $\mathcal{X}$ as follows:

$\forall t \in [T]$ set $e_t := \Big(\big(t-1, \mathcal{X}(t-1)\big), \big(t, \mathcal{X}(t)\big)\Big)$. Then set $P := (e_1, \ldots, e_T)$.

As each edge e_t in our graph has weight $d\big(\mathcal{X}(t-1), \mathcal{X}(t), \lambda_t\big)$ and hence 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, it directly follows that P is a shortest path of the graph.

"$\Leftarrow$": Let $P = \big((0,0) = v_0, \ldots, v_T = (T,0)\big)$ with $v_t \in \big\{(t,i) \mid 0 \le i \le m\big\}$ be a shortest path of the graph. Again, it follows from lemma 1.1 that an optimal schedule is uniquely identified by the sequence $\mathcal{X}$ of active servers.

We can construct a schedule from P by setting $\mathcal{X} = \big(v_0(1), \ldots, v_T(1)\big)$ where $v_t(1)$ is the second component of the t-th tuple in P.

By definition (1) 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. It's optimality directly follows from the definition of the edges' weights.

$\square$

1.5 A 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, β, λ_1, ..., λ_{T-1})
    if T < 2 then return
    let p[2 ... T-1, m] and M[1 ... T-1, m] be new arrays
    for j ← 0 to m do
        M[1, j] ← d(0, j, λ_1)
    for t ← 1 to T-2 do
        for j ← 0 to m do
            opt ← ∞
            for i ← 0 to m do
                M[t+1, j] ← M[t, i] + d(i, j, λ_{t+1})
                if M[t+1, j] < opt then
                    opt ← M[t+1, j]
                    p[t+1, j] ← i
            M[t+1, j] ← opt
    return p and M
```

Algorithm 2 Extract schedule for n homogeneous servers

```
function EXTRACT(m, p, M, T)
    let x[0 ... T] be a new array
    x[0] ← x[T] ← 0
    if T < 2 then return x          ▷ Trivial solution
    x[T-1] ← arg min_{0≤i≤m} {M[T-1, i]}
    for t ← T-2 to 1 do
        x[t] ← p[t+1, x[t+1]]
    return x
```

1.5.1 Runtime analysis

Schedule: Loop 5,8 and 10 run $m+1$ times, loop 7 runs $T-2$ times
Extract: Loop 5 runs $T-2$ times, argmin 4 takes time $m+1$.
For $T, n \to \infty$ it holds:

$$\mathcal{O}(m+1+(T-2)*(m+1)^2+T-2+m+1) = \mathcal{O}(2*m+T+(T-2)*(m+1)^2) = \mathcal{O}(T*m^2) \quad (3)$$

As we need $\log_2(m)$ bits to encode m, the algorithm is exponential in the number of servers.

1.6 A memory optimized algorithm

TODO

2 A $3/2$-approximation algorithm for monotonically increasing convex f

We consider a modification of the problem discussed in chapter 1. Assuming that f is convex and monotonically increasing, we can modify our algorithm to obtain a polynomial time 3/2-approximation algorithm.

2.1 Graph for a $3/2$-optimal schedule

The idea of the construction is to reduce the number of vertices by stop adding m vertices for each timestep but using vertices that approximate the number of active servers.
First, let $b := \lceil \log_2(m) \rceil$. We add vertices $(t, 0)$ and $(t, 2^i), \forall t \in [T-1], i \in \{0, \dots, \lceil \log_2(m) \rceil\}$.

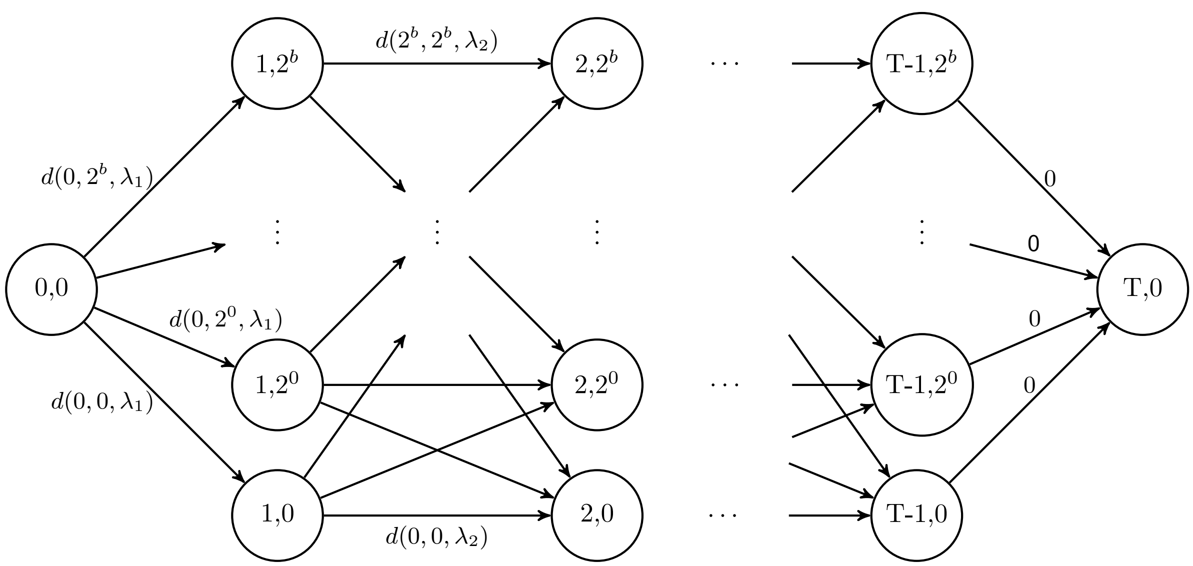

Figure 2: Graph for 3/2-approximation algorithm

Proposition 2.1. *A shortest path from $(0, 0)$ to $(T, 0)$ in the constructed graph delivers a $3/2$-optimal schedule.*

Proof. Assume we have an optimal schedule identified by $\mathcal{X} = (x_0, \dots, x_T)$. □