Dynamic Speed Scaling and Load Balancing of Interconnected Queues

Chiunlin Lim Ao Tang
Cornell University, Ithaca, NY 14853, USA

Abstract— **We consider the problem of joint service rate control and load balancing of a network of servers. The system incurs holding cost, effort cost, and a routing cost whenever a demand is routed to other servers. This formulation is motivated by recent interest on energy efficiency in IT systems where effort cost models power consumption and holding cost represents performance in terms of delay. The aim is to find a stationary policy that minimizes, over an infinite horizon, the long-run average cost rate. Using a dynamic programming formulation, we show that the optimal routing policy is acyclic and bipartite. We prove that the relative cost function is monotonically non-decreasing in queue size while for the case of 2 servers, the optimal service policy is non-decreasing in queue size and the optimal routing policy is a threshold policy. We show how upper and lower bounds of the optimal average cost rate can be efficiently calculated numerically. In particular, based on the monotonicity property, we develop an approximate dynamic programming procedure to efficiently compute a good upper bound. Numerical examples with two networked servers are provided to illustrate our findings.**

I. Introduction

The performance of a queueing network is usually characterized by its level of congestion through either the number of jobs in the system or the waiting time experienced by a job in the queue. To minimize congestion, we should run the system at the highest service rate possible. But in a real world setting, increasing the service rate incurs additional costs such as increased power consumption and a reduction in the lifetime of network components. Consequently, there is an important tradeoff between decreasing congestion and increasing the service rate.

Our paper is motivated by the recent surge of interest to reduce power consumption in datacenter networks. A large datacenter houses tens of thousands of servers and can consume electricity at the order of tens of megawatts [1]. To reduce electricity cost, the hardware approach is to replace the existing infrastructure with more power efficient components. A better cooling system, more efficient servers and power distribution, and improved architectural design of the datacenters are some of the approaches that have been implemented by datacenter owners [2]. On the other hand, better efficiency can also be achieved by optimizing software and deployment at various levels. At the chip level, we have dynamic voltage/frequency scaling (DVFS) or speed scaling [3], [4], [5]. At the machine level, virtualization is a widely deployed method to run multiple computer systems on a single set of computer hardware. At the datacenter level, one proposed method is to power down inactive network elements [6]. At the network level, we have power-demand routing which exploits the the price differentials of electricity prices for different geographical regions [7], [8], and load balancing with a constrained average delay [9], [10].

From a modeling point of view, our work focuses on the network level by considering a joint minimization of energy cost, delay cost and routing cost. We also allow the service rate to be changed dynamically according to the number of demands in the system. From an analytical perspective, our work is a generalization of the optimal service rate control of a queue [11] to an open network of parallel queues with routing costs and no feedback or cascade topologies. In this framework, the costs in consideration are known in the literature as cost of effort and holding cost (or congestion cost), which corresponds to energy cost and delay cost for the datacenter networks, respectively. In the rest of the paper, we'll use terms from the optimal service rate control literature, and whenever possible, a translation to the datacenter networks context is given.

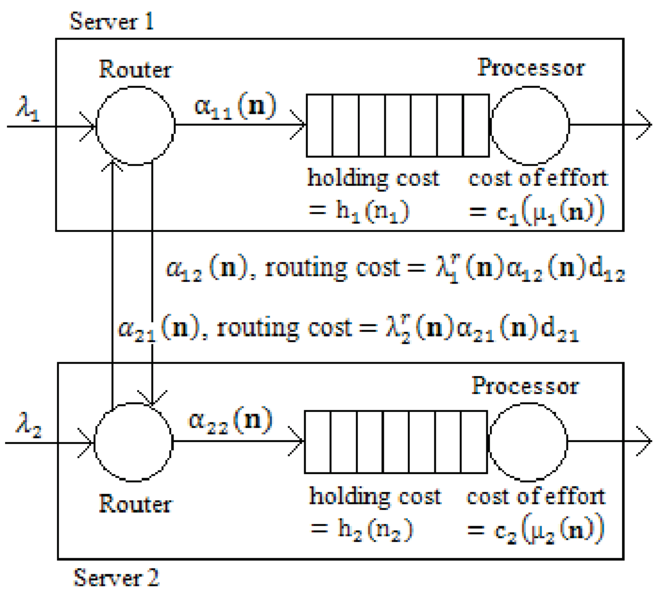
A brief overview of the paper along with a summary of our contributions is as follows. We start by giving a detailed mathematical description of our problem formulation in Section II. Then in Section III, we present the Bellman equation for our model and prove the existence of a stationary optimal policy. From the optimality equation we show that our model can handle any non-convex or discrete effort cost functions in Section IV-A. We show that the routing policy is acyclic and bipartite in Section IV-B. In Section IV-C, we prove that the relative value function is monotonic in queue size and for a network of two servers, the optimal service policy is non-decreasing in queue size and the optimal routing policy is a threshold policy. Based on the structural properties, we propose various techniques, including the use of approximate dynamic programming [12], to calculate the upper and lower bounds of the optimal expected average cost in Section V. Numerical examples that illustrate all our results are presented in Section VI. We conclude the paper in Section VII.

II. Model formulation

Our system consists of a set, $\mathcal{M} = \{1, 2 \ldots, M\}$ of M queues (datacenters). [1] Each queue i has a router r_i and a processor (the whole collection of servers) s_i connected in series (see Figure 1). Demands arrive to each server i as a

[1]Except for two structural properties in IV-C, all main results in this paper are for general M. One can see Section VI with $M = 2$ for concrete graphical illustration.

Fig. 1. Service rate control and load balancing for $M = 2$ servers.


Server 1
Router
Processor
λ_1
$\alpha_{11}(\mathbf{n})$
holding cost $= h_1(n_1)$
cost of effort $= c_1(\mu_1(\mathbf{n}))$
$\alpha_{12}(\mathbf{n})$, routing cost $= \lambda_1^r(\mathbf{n})\alpha_{12}(\mathbf{n})d_{12}$
$\alpha_{21}(\mathbf{n})$, routing cost $= \lambda_2^r(\mathbf{n})\alpha_{21}(\mathbf{n})d_{21}$
Processor
λ_2
$\alpha_{22}(\mathbf{n})$
Router
holding cost $= h_2(n_2)$
cost of effort $= c_2(\mu_2(\mathbf{n}))$
Server 2

Poisson process with rate λ_i. We let $\lambda = \sum_{i \in \mathcal{M}} \lambda_i$ be the total demand arrival rates to all servers. Each demand is assumed to have a workload that is i.i.d. with an exponential distribution of mean 1, and thus departs from server i with rate μ_i when the server is running at speed μ_i. The state of the system is the number of demands at each server, including the demands being served. We assume there is no bounds on the queue size of demands at each server and thus the state space is $\mathcal{N} \triangleq (\mathbb{Z}^+)^M$ and the state vector is $\mathbf{n} = (n_1, n_2, \ldots, n_M)$ where n_i is the number of demands at server i. We define the partial order $\geq$ on $\mathcal{N}$ as follows: we say $\mathbf{n}^+ \geq \mathbf{n}^-$ if $n_i^+ \geq n_i^-$ for all $i \in \mathcal{M}$.

The router serves as the entry point for the demands to the servers. At the arrival of each demand, the router decides, according to a probability distribution, whether to forward the demand to its server or to the routers of other servers. The routing probability is dependent on the current system state. At state $\mathbf{n}$, router i chooses to forward to its server with probability $\alpha_{ii}(\mathbf{n})$ and to router r_j, $j \neq i$ with probability $\alpha_{ij}(\mathbf{n})$. We require that $\forall \mathbf{n} \in \mathcal{N}$, $\sum_{k \in \mathcal{M}} \alpha_{ik}(\mathbf{n}) = 1$, and denote $\alpha(\mathbf{n}) = \{\alpha_{ij}(\mathbf{n}) : i, j \in \mathcal{M}\}$ as the set of routing probabilities when the state vector is $\mathbf{n}$. Due to routing, the total demand arrival rate to router r_i and processor s_i is now state dependent, and are denoted as $\lambda_i^r(\alpha(\mathbf{n}))$ and $\lambda_i^s(\alpha(\mathbf{n}))$, respectively and they satisfy $\lambda_i^s(\alpha(\mathbf{n})) = \alpha_{ii}(\mathbf{n})\lambda_i^r(\alpha(\mathbf{n}))$ as well as

$$\lambda_i^r(\alpha(\mathbf{n})) = \lambda_i + \sum_{j \neq i} \lambda_j^r(\alpha(\mathbf{n}))\alpha_{ji}(\mathbf{n}) \tag{1}$$

The processor serves the demands forwarded to it by the router. The processor has a load-dependent service rate, i.e. the service rate is $\mu_i(\mathbf{n})$ when there are $\mathbf{n}$ demands in the system. The service rate of s_i is constrained to fall in a compact set $A_i \subset [0, \infty)$, which has a least upper bound of $\bar{\mu}_i$ and is assumed to contain the point $\mu_i = 0$ and some point $\mu_i > \lambda_i$. The set of service rates at state $\mathbf{n}$ is denoted as $\mu(\mathbf{n}) = \{\mu_i(\mathbf{n}) : i \in \mathcal{M}\}$. Thus, the action space, which is the same for any state $\mathbf{n}$, is $A \triangleq A_1 \times \ldots \times A_M \times R$, where $R \triangleq \left\{\alpha \in \mathbb{R}^{M \times M} : \alpha_{ij} \geq 0, \sum_j \alpha_{ij} = 1, i \in \mathcal{M}\right\}$ is the set of admissible routing probabilities.

Three types of costs are incurred for the system: cost of effort, holding cost, and routing cost. If we run processor s_i at service rate x_i, an effort cost is continuously incurred at a rate of $c_i(x_i)$. For each i, the function $c_i : A_i \to \mathbb{R}$ is assumed to be non-decreasing, and has, as a normalization, $c_i(0) = 0$. We denote $c(\mathbf{x}) = \sum_{i \in \mathcal{M}} c_i(x_i)$ as the total cost of effort. Note that we do not require A_i to be connected or $c_i(\cdot)$ to be convex, although we will show in Section IV-A that we could have assumed, without loss of generality, that $A_i = [0, \bar{\mu}_i]$ and $c_i(\cdot)$ is convex.

When processor s_i has a queue size of n_i, a holding cost is continuously incurred at rate $h_i(n_i)$. The function $h_i : \mathbb{Z}^+ \to \mathbb{R}$ is assumed to be nondecreasing and have less-than geometric growth:

$$\sum_{n=0}^{\infty} h_i(n) u_i^n < \infty \text{ for all } u_i \in [0, 1), \forall i \in \mathcal{M} \tag{2}$$

We denote $h(\mathbf{n}) = \sum_{i \in \mathcal{M}} h_i(n_i)$ as the total holding cost. The holding cost is, in a sense, the performance metric of the individual servers. For instance, in the datacenter networks context, $h_i(n) = n, \forall i \in \mathcal{M}$, and the holding cost is simply the total delay experienced by the demands in all the servers. Note that when the holding cost is bounded by $\lim_{n \to \infty} h_i(n) = h_i^\infty < \infty$ for at least one server i, then it is possible that the optimal policy routes all demands to i, sets all server speeds to 0, and achieves a long-run average cost rate of h_i^∞. Reference [11] calls such a problem *degenerate.*

A fixed routing cost $d_{ij} \geq 0$ is incurred whenever router r_i decides to route an incoming demand to router r_j. By convention, $d_{ii} = 0$. We denote $d(\alpha) = \sum_{i,j \in \mathcal{M}} \lambda_i^r(\alpha)\alpha_{ij}d_{ij}$ as the total routing cost. In the datacenter networks context, the fixed routing cost can be thought of as the cost due to routing delay, bandwidth consumption or energy required to route the demand. Note that unlike cost of effort and holding cost which are incurred continuously, the routing cost is incurred only when a demand is routed. However, since demands arrive at a rate of $\lambda_i^r(\alpha(\mathbf{n}))$, the routing cost can be thought of as being incurred continuously at a rate of $\lambda_i^r(\alpha(\mathbf{n}))\alpha_{ij}(\mathbf{n})d_{ij}$.

Before formulating the optimization problem, we address the key modeling assumptions that we have made so far. Although each datacenter houses thousands of servers, we treat each datacenter as a single cluster of server and represent the cluster as a single server. This representation is in keeping with our focus on the network level instead of the datacenter level. Any datacenter-level optimization can be independently deployed and after all the optimization is done, all we need to perform our analysis is the effort cost function, which in this context, is the relationship between the datacenter utilization level and the power consumption. As for the demands, the Poisson arrivals and the exponential service times are approximations and are assumed for tractability. We further assume there is only one class of demand, and each demand requires only one service and any datacenter is able to service the demands equally well. In practice, datacenters typically have to deal with multiple types of requests, each with its own servicing time and frequently, due to incomplete data replication, a datacenter may have to acquire data from another datacenter before it can service a demand.

The final component of our model is a *policy* that specifies (μ, α), where $\mu = \{\mu(\mathbf{n}) : \mathbf{n} \in \mathcal{N}\}$ and $\alpha = \{\alpha(\mathbf{n}) : \mathbf{n} \in \mathcal{N}\}$. In other words, a policy tells us the routing probabilities and the service rates to adopt at each server for every feasbile state. At the times in between state transitions, the service rates and the transition probabilities are constrained to be constant. This means that our problem is a continuous-time Markov decision process (CTMDP) [13]. Note that we are restricting our attention to stationary policies that choose the same service rate $\mu(\mathbf{n})$ and routing probabilities $\alpha(\mathbf{n})$ whenever the state is $\mathbf{n}$. The justification for this restriction is given in Theorem 1 where we show the existence of an optimal stationary policy.

A stationary policy (μ, α) induces a continuous-time Markov chain $\{I(t) : t \geq 0\}$ with an embedded Markov chain $\{I_u : u = 1, 2, \ldots\}$ that has a transition probability matrix $P(\mu, \alpha)$ on $\mathcal{N}$. Denote $C(\mathbf{n}, \mathbf{x}, Q) = h(\mathbf{n}) + c(\mathbf{x}) + d(Q)$ as the total cost rate when the system is in state $\mathbf{n}$, runs at speed $\mathbf{x}$ and routes demands according to Q. The long-run average cost rate of the policy, given that the starting state is $I(0) = \mathbf{n}$, is defined by

$$z(\mu, \alpha) = \lim_{T \to \infty} \mathbb{E}_{\mathbf{n}} \left\{ \frac{\int_0^T C(I(t), \mu(I(t)), \alpha(I(t)))\, dt}{T} \right\}$$

where $\mathbb{E}(\cdot)$ is the expectation operator. Our problem is to determine the minimum long-run average cost rate and find the policy that achieves it, i.e. determine

$$z^* = \min_{(\mu,\alpha)} z(\mu, \alpha) \tag{3}$$

$$(\mu^*, \alpha^*) = \arg\min_{(\mu,\alpha)} \mathrm{z}(\mu, \alpha) \tag{4}$$

Note that the set of all stationary policies is a closed set since the action space at each state is the compact set A, and the product of a collection of compact sets is compact by Tychonoff's theorem [13]. A policy (μ, α) is said to be *optimal* if $z(\mu, \alpha) = z^*$. Assumption (2) ensures that $z^* < \infty$ since we can specify a policy (μ, α) for which no routing occurs, and for all $\mathbf{n} \in \mathcal{N}$, $i \in \mathcal{M}$, $\mu_i(\mathbf{n}) = x_i > \lambda_i$. Without routing, the demand arrival rate is no longer state-dependent and thus we can determine the average cost through the stationary distribution of the Markov chain by applying Jackson's theorem [14]. The joint stationary distribution is decomposable into the product of M marginal stationary distributions, each of which is geometric with parameter $u_i = \frac{\lambda_i}{x_i}$ and thus $z(\mu, \alpha) < \infty$.

In this section, we have laid down all the components of an infinite horizon continuous-time Markov decision process. To sum it up, our problem has a continuous time parameter, a countably infinite state space, a possibly unbounded one-stage cost, a long-run average cost rate optimality criterion and the set of admissible policies is Markov deterministic, stationary and unichain.

III. Optimality Condition

The standard approach to a Markov decision process with a long-run average cost rate optimality criterion is to start with the Bellman equation or the average cost optimality equation (ACOE) [15], [13]:

$$\begin{aligned} v_{\mathbf{n}} = \min_{\mathbf{x},Q} \Bigg\{ \frac{1}{\tau_{\mathbf{n}}(\mathbf{x})} \Bigg[& h(\mathbf{n}) + c(\mathbf{x}) - z \\ & + \sum_{i \in \mathcal{M}: n_i > 0} x_i v_{\mathbf{n}-\mathbf{e_i}} + \sum_{i \in \mathcal{M}} \lambda_i^r(Q) q_{ii} v_{\mathbf{n}+\mathbf{e_i}} \\ & + \sum_{i \in \mathcal{M}} \sum_{k \in \mathcal{M}} \lambda_i^r(Q) q_{ik} d_{ik} \Bigg] \Bigg\} \end{aligned} \tag{5}$$

where the minimum is taken over all admissible speeds $\{\mathbf{x} = (x_1, \ldots, x_M) : x_i \in A_i, i \in \mathcal{M}\}$ and routing probabilities $\left\{Q \in \mathbb{R}^{M \times M} : q_{ij} \geq 0, \sum_j q_{ij} = 1, i \in \mathcal{M}\right\}$, and $\tau_{\mathbf{n}}(\mathbf{x}) = \lambda + \sum_{i \in \mathcal{M}: n_i > 0} x_i$, is the rate of state changes (observe that the optimal service rate for datacenter i when $n_i = 0$ is $x_i = 0$). z can be interpreted as the minimum average cost rate, z^*. $v_{\mathbf{n}}$ is known as the *relative cost function* and can be interpreted as follows: we define a new cost structure known as the z-revised cost structure where the holding cost rate is $\sum_i h_i(n_i) - z$, then starting from state $\mathbf{n}$, the minimum expected cost incurred until the first entry into an arbitrary reference state $\mathbf{m} \in \mathcal{N}$ is $v_{\mathbf{n}}$.

By moving $v_{\mathbf{n}}$ over to the RHS, multiplying through by $\tau_{\mathbf{n}}(\mathbf{x})$, and grouping the terms together, (5) become

$$\begin{aligned} 0 = \min_{Q} \Bigg[\sum_{i \in \mathcal{M}} \Bigg\{ & \lambda_i^r(Q) q_{ii} (v_{\mathbf{n}+\mathbf{e_i}} - v_{\mathbf{n}}) \\ & + \sum_{j \in \mathcal{M}} \lambda_i^r(Q) q_{ij} d_{ij} \Bigg\} \Bigg] + \sum_{i \in \mathcal{M}} h_i(n_i) - z \\ & - \sum_{i \in \mathcal{M}: n_i > 0} \max_{x_i \in A_i} \left[x_i (v_{\mathbf{n}} - v_{\mathbf{n}-\mathbf{e_i}}) - c_i(x_i) \right] \end{aligned} \tag{6}$$

If a stationary policy achieves the minimum in the ACOE, it is optimal (see Theorem 5.1 of [16]) but the converse does not hold in general.

We now justify our restriction to stationary policies by using results from [17], [18] to show that there exists a stationary optimal policy.

Theorem 1: There exists a stationary policy that minimizes the expected average cost for any initial state $\mathbf{n} \in \mathcal{N}$ and the expected average cost z^* is independent of the starting state.

Proof: We prove by checking for the 7 conditions from [18] with the two modifications — condition 7 is dropped while condition 5 becomes: It is possible to go from any state $\mathbf{n}$ to state $\mathbf{0}$ with finite expected cost. The proof of [18] works by first formulating our infinite horizon MDP to have an expected total discounted cost optimality criterion. Next, they utilize the fact that for such an optimality criterion, the MDP has an optimal stationary policy and that the action space is compact to show that we can take the limit as the discount factor β approaches 0 and prove the desired results. Observe that conditions 1-4 and 6 are obviously satisfied for our model. Though we have modified condition 5 and dropped condition 7, the proof of [18] goes through for our model since for an expected total discounted cost optimality criterion, state $\mathbf{0}$ is the most favourable starting state (see Theorem 3), regardless of what the discount factor β is. ■

In the following section, we will show that optimal policies satisfying the ACOE must possess certain structural properties.

IV. Structural Properties

A. Service Rates

From (6), we see that we have two separate optimizations to deal with. Let's first focus on the last term that determines the optimal service rates. Defining

$$\phi_i(y) = \max_{x_i \in A_i} [x_i y - c_i(x_i)] \tag{7}$$

$$\psi_i(y) = \min\{x : xy - c_i(x) = \phi_i(y)\} \tag{8}$$

$\phi_i(\cdot)$ is the convex conjugate [19] of $c_i(\cdot)$ and is therefore convex. $\psi_i(\cdot)$ is the smallest element of the subgradient of $\phi_i(\cdot)$ and since the subgradient is a singleton if $\phi'(\cdot)$ exists, $\psi(\cdot)$ equals $\phi'(\cdot)$ if the derivative exists. $\psi_i(\cdot)$ is left-continuous and non-decreasing. Since for $x_i \in A_i$, $c_i(x_i) \geq c_i(0)$, and $A_i \subset [0,\infty)$, we have that

$$\phi_i(y) = 0, \text{ for all } y \leq 0 \tag{9}$$

As explained in detail in [11], [17], both (7) and (8) remain the same if we replace the effort cost $c_i(\cdot)$ by its convex hull, which by definition, is the largest convex function $\hat{c}_i(\cdot)$ on $[0,\bar{\mu}_i]$such that $c_i(x_i) \geq \hat{c}_i(x_i)$ for all $x_i \in A_i$. To establish equivalency of the optimization problem after the replacement, define $A_i^* \subset A$ to be the set of points $x_i \in A_i$ such that $c_i(x_i) = \hat{c}_i(x_i)$, then we just need to check that for all $y \geq 0$, $\psi_i(y) = \hat{x}_i \in A_i^*$. From (7), for any $x_i \in A$, $c_i(x) \geq c_i(\hat{x}_i) + y(x_i - \hat{x}_i) \triangleq f(x_i)$. $f(x_i)$ is a convex function and since $\hat{c}_i(\cdot)$ is the largest convex function, we have that $c_i(\hat{x}_i) \geq \hat{c}_i(\hat{x}_i) \geq f(\hat{x}_i)$, but $f(\hat{x}_i) = c_i(\hat{x}_i)$ implies that equality holds throughout and thus $c_i(\hat{x}_i) = \hat{c}(\hat{x}_i)$ implying that $\hat{x}_i \in A_i^*$.

Thus, we could replace $c_i(\cdot)$ of any shape, and any compact action set A_i within $[0,\bar{\mu}_i]$, with a convex $c_i(\cdot)$ and a connected action set $A_i = [0,\bar{\mu}_i]$. The previous statement has a few implications for the datacenter networks context. Prior work, for instance [10], has typically assumed that $c_i(\cdot)$ is convex and that the CPU can run at any utilization level from 0 to 1. The justification for the convex power function arises from theoretical consideration or from measured data on how the power usage of the CPU changes as its speed is varied. However, other components of a server, such as the memory, motherboard, hard drive, etc also consumes power. [3] estimated that the CPU consumes only about 40% of a server's power and their empirical fit, which is accurate to within 1% error, of the server power usage as a function of its speed shows a slightly concave function. Moreover, at present time, CPU only has a finite number of frequency-power operating points. Thus, one strength of our model is that we are able to accomodate non-convex power function and any servers with a finite number of frequency-power operating points.

B. Routing Policy

Next we move on to investigate the optimal routing policy. First define, for all $i,j \in \mathcal{M}$, $y_{ij} = \lambda_i^r q_{ij}$, then the first term of (6) becomes

$$\min_Y \left[\sum_{i\in\mathcal{M}} \left(y_{ii}(v_{\mathbf{n}+\mathbf{e_i}} - v_{\mathbf{n}}) + \sum_{j\in\mathcal{M}} y_{ij} d_{ij} \right)\right] \tag{10}$$

where $Y = \{y_{ij} : i,j \in \mathcal{M}\}$. From (1), we obtain the restriction that, for each i

$$\sum_{j\in\mathcal{M}} y_{ij} = \lambda_i + \sum_{k\neq i} y_{ki} \tag{11}$$

This reformulation of the routing policy allows us to show that the optimal routing policy is acyclic (if i routes to j, then j doesn't route to i) and bipartite (each router is either a "sink" or a "source" but not both).

Theorem 2: Denote the optimal routing policy as Y^*. If $\forall i,j,k \in \mathcal{M}, i \neq j, j \neq k, d_{ij} + d_{jk} > d_{ik}$, i.e. triangle inequality holds strictly, then $y_{ij}^* > 0$ implies that $y_{jk}^* = 0$.

Proof: We'll prove by contradiction. Suppose there exists $i,j,k \in \mathcal{M}, i \neq j, j \neq k$ such that $y_{ij}^* > 0$ and $y_{jk}^* > 0$. Now set $\delta = \min\left\{y_{ij}^*, y_{jk}^*\right\}$ and $\bar{y}_{ij} = y_{ij}^* - \delta$, $\bar{y}_{jk} = y_{jk}^* - \delta$, $\bar{y}_{ik} = \begin{cases} y_{ik}^* + \delta & , i \neq k \\ y_{ik}^* & , i = k \end{cases}$. With this modification, (11) is still satisfied but the cost has decreased since $\delta d_{ik} - \delta(d_{ij} + d_{jk}) < 0$, contradicting optimality of y_{ij}^* and y_{jk}^*. ■

From this point onwards, we assume that triangle inequality holds strictly for the routing cost. With this assumption, Theorem 2 tells us that our problem is equivalent to having the routers decide which processors to route to (instead of deciding between its server's processor and the routers of other servers). Hence, reverting back to the original formulation using routing probabilities, the first term of (6) can be rewritten as

$$\begin{aligned} &\min_Q \left[\sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{M}} \lambda_i q_{ij} \left[(v_{\mathbf{n}+e_j} - v_{\mathbf{n}}) + d_{ij} \right]\right] \\ &= \sum_{i\in\mathcal{M}} \lambda_i \left[-v_{\mathbf{n}} + \min_{Q_i} \sum_{j\in\mathcal{M}} q_{ij} \left(v_{\mathbf{n}+e_j} + d_{ij} \right) \right] \end{aligned} \tag{12}$$

where $Q_i = \left\{q_{ij} : q_{ij} \geq 0, \sum_j q_{ij} = 1, j \in \mathcal{M}\right\}$.

We have thus separated the problem into M different subproblems. For the subproblem, for fixed i, it is clear that the optimal routing policy is to route to server j with the lowest value in $v_{\mathbf{n}+e_j} + d_{ij}$.

C. Monotonicity

Our aim is to show that the following structural properties hold for all $i \in \mathcal{M}, j \neq i$ and $\mathbf{n} \in \mathcal{N}$,

(a) (the relative cost function is nondecreasing) $u_{\mathbf{n}+\mathbf{e_i}} - u_{\mathbf{n}} \geq 0$

(b) (the optimal service rate is nondecreasing in $\mathbf{n}$)

   (i) $u_{\mathbf{n}+2\mathbf{e_i}} - u_{\mathbf{n}+\mathbf{e_i}} \geq u_{\mathbf{n}+\mathbf{e_i}} - u_{\mathbf{n}}$

   (ii) $u_{\mathbf{n}+\mathbf{e_i}+\mathbf{e_j}} - u_{\mathbf{n}+\mathbf{e_i}} \geq u_{\mathbf{n}+\mathbf{e_j}} - u_{\mathbf{n}}$

(c) (the routing policy is a threshold policy, see [20], [15]) $u_{\mathbf{n}+2\mathbf{e_i}} - u_{\mathbf{n}+\mathbf{e_j}+\mathbf{e_i}} \geq u_{\mathbf{n}+\mathbf{e_i}} - u_{\mathbf{n}+\mathbf{e_j}}$

At this point, we have the proof for property (a) while for properties (b) and (c), we can prove it for $M = 2$.

The proof to show that the relative cost function is monotonic in $\mathbf{n}$ takes the same route as Theorem 1 by first considering the minimal expected total discounted cost optimality criterion. If we can show that the desired property holds for any discount factor, then it also holds in the limit as the discount factor approaches 0.

Let $\beta > 0$ be any positive discount factor and define $u^\beta_{\mathbf{n}}$ as the minimal expected total discounted cost starting from state $\mathbf{n}$. We first apply the standard technique of uniformization [15] to convert the problem from continuous time to discrete time. Denote $\tau = \lambda + \sum_i \bar{\mu}_i$, then the Bellman equation for the equivalent discrete time problem is

$$u^\beta_{\mathbf{n}} = \min_{\mathbf{x},Q} \frac{1}{\beta+\tau} \sum_{i \in \mathcal{M}} \Bigg\{ h_i(n_i) + c_i(x_i) + x_i u^\beta_{\mathbf{n}-\mathbf{e_i}} + \sum_{j\in\mathcal{M}} \lambda_i q_{ij} \left(d_{ij} + u^\beta_{\mathbf{n}+\mathbf{e_j}}\right) + (\bar{\mu}_i - x_i) u^\beta_{\mathbf{n}} \Bigg\} \quad (13)$$

Denoting $C^\beta(\mu,\alpha) \triangleq \left\{\frac{C(\mathbf{n},\mu(\mathbf{n}),\alpha(\mathbf{n}))}{\beta+\tau}\right\}_{\mathbf{n}\in\mathcal{N}}$ as the vector of one-stage costs, $u^\beta = \left\{u^\beta_{\mathbf{n}}\right\}_{\mathbf{n}\in\mathcal{N}}$ as the vector of minimal expected total discounted cost starting from state $\mathbf{n}$, $\mathbf{n}_a = \{\mathbf{n} \pm \mathbf{e_i} \geq \mathbf{0} : i \in \mathcal{M}\} \cup \{\mathbf{n}\}$ as the set of states adjacent to state $\mathbf{n}$, inclusive of state $\mathbf{n}$ itself, $P(\mu,\alpha) = \{P(\mathbf{n},\mathbf{m}|\mu(\mathbf{n}),\alpha(\mathbf{n}))\}_{\mathbf{n},\mathbf{m}\in\mathcal{N}}$ as the transition probability matrix of going from state $\mathbf{n}$ to $\mathbf{m}$, then Bellman equation can be written compactly as

$$\begin{aligned} u^\beta &= \min_{(\mu,\alpha)} C^\beta(\mu,\alpha) + \gamma \left\{ \sum_{\mathbf{m}\in\mathbf{n}_a} P(\mathbf{n},\mathbf{m}|\mu(\mathbf{n}),\alpha(\mathbf{n})) u^\beta_{\mathbf{m}} \right\}_{\mathbf{n}\in\mathcal{N}} \\ &= \min_{(\mu,\alpha)} C^\beta(\mu,\alpha) + \gamma P(\mu,\alpha) u^\beta \end{aligned}$$

The Bellman operator $\mathcal{T}$ is thus defined as

$$\mathcal{T}u = \min_{(\mu,\alpha)} C^\beta(\mu,\alpha) + \gamma P(\mu,\alpha) u$$

To prove monotonicity, we rely on the value iteration algorithm [15], which works as follows:

1) Initialize with $u^0 = \mathbf{0}$.
2) For iteration k, $k \geq 1$, update u by applying the Bellman operator: $u^k = \mathcal{T}u^{k-1}$.

Since the one-stage cost can be unbounded, in general, value iteration does not converge. However, we will show that in this setup, value iteration does indeed converge by drawing from the results from Chapter 6.10 of [13].

For all $\mathbf{n} \in \mathcal{N}$, let $w^\beta_{\mathbf{n}} = \frac{h(\mathbf{n}) + c(\bar{\mu}) + \sum_i \lambda_i \max_j d_{ij}}{\beta+\tau}$ be the maximum one-stage cost that can be incurred. Define the weighted supremum norm $\|\cdot\|_{w^\beta}$ for real-valued functions on $\mathcal{N}$ by

$$\|u\|_{w^\beta} = \sup_{\mathbf{n}\in\mathcal{N}} \frac{|u_{\mathbf{n}}|}{w^\beta_{\mathbf{n}}}$$

and let U_{w^β} be the space of real-valued functions u on $\mathcal{N}$ satisfying $\|u\|_{w^\beta} < \infty$.

Lemma 1: For any discount factor $\beta > 0$, if there exists a positive constant L^β such that $h(\mathbf{n}+\mathbf{e_j}) - h(\mathbf{n}) \leq L^\beta(\beta+\tau)$ for all $\mathbf{n} \in \mathcal{N}$ and $j \in \mathcal{M}$, then the optimality equation $u^\beta = \mathcal{T}u^\beta$ has a unique solution $\hat{u}^\beta \in U_{w^\beta}$. Moreover, for any $u_0 \in U_{w^\beta}$, $\lim_{n\to\infty} \|\mathcal{T}^n u_0 - \hat{u}^\beta\| = 0$.

Proof: The lemma is modified from Theorem 6.10.4 of [13]. To satisfy the conditions of the Theorem 6.10.4, we apply proposition 6.10.5a of [13] and check for the following two conditions: for all $\mathbf{n} \in \mathcal{N}$,

$$\max_{a\in A} \left| \frac{h(\mathbf{n}) + c(\mu(\mathbf{n})) + \sum_i \sum_j \lambda_i \alpha_{ij}(\mathbf{n}) d_{ij}}{\beta+\tau} \right| \leq w^\beta_{\mathbf{n}} \quad (14)$$

$$\sum_{\mathbf{m}\in\mathbf{n}_a} P(\mathbf{n},\mathbf{m}|a) w^\beta_{\mathbf{m}} \leq w^\beta_{\mathbf{n}} + L^\beta, \text{ for all } a \in A \quad (15)$$

The first condition holds trivially by the definition of $w^\beta_{\mathbf{n}}$ while for the second condition:

$$\begin{aligned} &\sum_{\mathbf{m}\in\mathbf{n}_a} P(\mathbf{n},\mathbf{m}|a) w^\beta_{\mathbf{m}} \\ =& w^\beta_{\mathbf{n}} + \sum_{\mathbf{m}\in\mathbf{n}_a} P(\mathbf{n},\mathbf{m}|a) \frac{h(\mathbf{m}) - h(\mathbf{n})}{\beta+\tau} \\ \leq& w^\beta_{\mathbf{n}} + L^\beta \end{aligned}$$

■

Now that we know value iteration converges, we can use it to inductively show the desired structural property.

Theorem 3: For any $\beta > 0$, if u^β satisfies $u^\beta = \mathcal{T}u^\beta$, and $\mathbf{n}^+ \geq \mathbf{n}^-$, then $u^\beta_{\mathbf{n}^+} - u^\beta_{\mathbf{n}^-} \geq 0$

Proof: We'll show by induction on the value iteration algoritm. For $t = 0$, it clearly holds since $u^0 = \mathbf{0}$. Now suppose it holds for $t = 0, 1, \ldots, k$. For iteration $t = k+1$, denote the policy that minimizes $\mathcal{T}u^k_{\mathbf{n}^+}$ as $(\mathbf{x}^*, Q^*)$. Apply the policy $(\mathbf{x}^*, Q^*)$ to state $\mathbf{n}^-$, then we have

$$\begin{aligned} &\mathcal{T}u^k_{\mathbf{n}^+} \\ \geq& \frac{1}{\beta+\tau} \Bigg\{ h(\mathbf{n}^-) + c(\mathbf{x}^*) + \sum_{i\in\mathcal{M}:n_i>0} x^*_i u^{k-1}_{\mathbf{n}^- - \mathbf{e_i}} \\ &+ \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{M}} \lambda_i q^*_{ij} \left(d_{ij} + u^{k-1}_{\mathbf{n}^- + \mathbf{e_j}}\right) \\ &+ (\tau - \tau_{\mathbf{n}^-}(\mathbf{x}^*)) u^{k-1}_{\mathbf{n}^-} \geq \mathcal{T}u^k_{\mathbf{n}^-} \end{aligned}$$

The first inequality holds because of the induction hypothesis and because$h_i(n_i)$ is nondecreasing in n_i. The second inequality holds by the definition of $\mathcal{T}$. ■

We next prove monotonicity properties (b) and (c) for $M=2$.

Theorem 4: For $M=2$ and for any $\beta>0$, if the holding costs are convex, i.e. $h_i(n+2)-h_i(n+1)\geq h_i(n+1)-h_i(n)$ for $i=1,2$ and $n\geq 0$, then for any u^β satisfying $u^\beta=\mathcal{T}u^\beta$, u^β exhibits monotonicity properties (b) and (c).

Proof: We prove by induction and first prove property (b)(i) before proceeding to property (b)(ii) and finally property (c). The base case $t=0$ is trivially true with $u^0=\mathbf{0}$. Assuming properties (b) and (c) holds for $t=0,1,\ldots,k$. For iteration $t=k+1$, to prove property (b)(i), we need to show that for all $i\in\mathcal{M}, j\neq i$ and $\mathbf{n}\in\mathcal{N}$,

$$\mathcal{T}u^k_{\mathbf{n}+2\mathbf{e_i}}-\mathcal{T}u^k_{\mathbf{n}+\mathbf{e_i}}-\mathcal{T}u^k_{\mathbf{n}+\mathbf{e_i}}+\mathcal{T}u^k_{\mathbf{n}}\geq 0 \qquad (16)$$

All four terms can be expanded using (13). First, without loss of generality, we assume $i=1$. Next, we can separate the terms involving holding cost, routing and service into 3 different groups and show that each group of terms are individually non-negative. The terms involving holding cost are non-negative since the holding costs are convex. For the routing terms, denoting Q^+ and Q^- as the optimal policy for states $\mathbf{n}+2\mathbf{e_1}$ and $\mathbf{n}$, respectively, and applying Q^+ and Q^- once each to state $\mathbf{n}+\mathbf{e_1}$, it is sufficient to show that

$$\sum_{l=1}^{2}\sum_{j=1}^{2}\lambda_l\Big[q^+_{lj}u^k_{\mathbf{n}+2\mathbf{e_1}+\mathbf{e_j}}-\overset{+}{q_{lj}}u^k_{\mathbf{n}+\mathbf{e_1}+\mathbf{e_j}} \\ -\overset{-}{q_{lj}}u^k_{\mathbf{n}+\mathbf{e_1}+\mathbf{e_j}}+q^-_{lj}u^k_{\mathbf{n}+\mathbf{e_j}}\Big]\geq 0 \qquad (17)$$

Do note that by imposing policies Q^+ and Q^- on $\mathbf{n}$, the routing costs of the positive and negative terms cancel each other off. Similar cancelations will occur throughout the rest of the proof. For the proof to be complete, we need to show that (17) is true for all possible routing policies. Due to the acyclic and bipartite nature of the optimal routing policy, at each state, there are 3 possibilities: no routing, route all to server 1 or route all to server 2. Since we have 2 states, $\mathbf{n}+2\mathbf{e_1}$ and $\mathbf{n}$, we have $3\times 3=9$ possible policies to consider. However, our work here is considerably simplified since the first two terms are the smallest when the optimal policy at $\mathbf{n}+2\mathbf{e_1}$ is to route all to to server 2 and the final two terms are the smallest when the optimal policy at $\mathbf{n}$ is to route all to server 1. Non-negativity follows since $u^k_{\mathbf{n}+2\mathbf{e_1}+\mathbf{e_2}}-u^k_{\mathbf{n}+\mathbf{e_1}+\mathbf{e_2}}\geq u^k_{\mathbf{n}+2\mathbf{e_1}}+u^k_{\mathbf{n}+\mathbf{e_1}}$ by property (b)(ii).

For the service terms, we break them up further according to server and show that each is non-negative. Denote x^+ and x^- as the optimal service policy for states $\mathbf{n}+2\mathbf{e_1}$ and $\mathbf{n}$, respectively. At server 1, we have

$$\begin{aligned}&\left(\bar\mu_1-x_1^+\right)u^k_{\mathbf{n}+2\mathbf{e_1}}+x_1^+u^k_{\mathbf{n}+\mathbf{e_1}}-\left(\bar\mu_1-x_1^+\right)u^k_{\mathbf{n}+\mathbf{e_1}}-x_1^+u^k_{\mathbf{n}}\\&-\left(\bar\mu_1-x_1^-\right)u^k_{\mathbf{n}+\mathbf{e_1}}-x_1^-u^k_{\mathbf{n}}+\left(\bar\mu_1-x_1^-\right)u^k_{\mathbf{n}}+x_1^-u^k_{\mathbf{n}-\mathbf{e_1}}\\&=\left(\bar\mu_1-x_1^+\right)\left(u^k_{\mathbf{n}+2\mathbf{e_1}}-u^k_{\mathbf{n}+\mathbf{e_1}}-u^k_{\mathbf{n}+\mathbf{e_1}}+u^k_{\mathbf{n}}\right)\\&\quad+x^-\left(u^k_{\mathbf{n}+\mathbf{e_1}}-u^k_{\mathbf{n}}-u^k_{\mathbf{n}}+u^k_{\mathbf{n}-\mathbf{e_1}}\right)\end{aligned}$$

The sum of terms in the two rows are non-negative by property (b)(i). The inequality holds even if $\mathbf{n}-\mathbf{e_1}\notin\mathcal{N}$ since in that case, $x^-=0$. At server 2, we have

$$\begin{aligned}&\left(\bar\mu_2-x_2^+\right)u^k_{\mathbf{n}+2\mathbf{e_1}}+x_2^+u^k_{\mathbf{n}+2\mathbf{e_1}-\mathbf{e_2}}-\left(\bar\mu_2-x_2^+\right)u^k_{\mathbf{n}+\mathbf{e_2}}\\&-x_2^+u^k_{\mathbf{n}+\mathbf{e_1}-\mathbf{e_2}}-\left(\bar\mu_2-x_2^-\right)u^k_{\mathbf{n}+\mathbf{e_1}}-x_2^-u^k_{\mathbf{n}+\mathbf{e_1}-\mathbf{e_2}}\\&+\left(\bar\mu_2-x_2^-\right)u^k_{\mathbf{n}}+x_2^-u^k_{\mathbf{n}-\mathbf{e_2}}\\&=\left(\bar\mu_2-x_2^+\right)\left(u^k_{\mathbf{n}+2\mathbf{e_1}}-u^k_{\mathbf{n}+\mathbf{e_1}}-u^k_{\mathbf{n}+\mathbf{e_1}}+u^k_{\mathbf{n}}\right)\\&\quad+x_2^+\left(u^k_{\mathbf{n}+2\mathbf{e_1}-\mathbf{e_2}}-u^k_{\mathbf{n}+\mathbf{e_1}-\mathbf{e_2}}-u^k_{\mathbf{n}+\mathbf{e_1}}+u^k_{\mathbf{n}}\right)\\&\quad+x_2^-\left(u^k_{\mathbf{n}+\mathbf{e_1}}-u^k_{\mathbf{n}+\mathbf{e_1}-\mathbf{e_2}}-u^k_{\mathbf{n}}+u^k_{\mathbf{n}-\mathbf{e_2}}\right)\end{aligned}$$

The first row is non-negative by property (b)(i), the second by property (c) and the third by property (b)(ii). If $\mathbf{n}-\mathbf{e_2}\notin\mathcal{N}$, this implies $\mathbf{n}+2\mathbf{e_1}-\mathbf{e_2}\notin\mathcal{N}$ and thus $x_2^+=x_2^-=0$ and the inequality still holds. The induction step is thus complete for property (b)(i).

To prove property (b)(ii) holds, we have to show that for all $\mathbf{n}\in\mathcal{N}$,

$$\mathcal{T}u^k_{\mathbf{n}+\mathbf{e_1}+\mathbf{e_2}}-\mathcal{T}u^k_{\mathbf{n}+\mathbf{e_1}}-\mathcal{T}u^k_{\mathbf{n}+\mathbf{e_2}}+\mathcal{T}u^k_{\mathbf{n}}\geq 0 \qquad (18)$$

We proceed similarly by expanding using (13) and breaking up the terms into groups. The holding cost terms sum to zero. Denote Q^+, and Q^- as the optimal policy for states $\mathbf{n}+\mathbf{e_1}+\mathbf{e_2}$ and $\mathbf{n}$, respectively. We have two options here: impose policy Q^+ on $\mathbf{n}+\mathbf{e_1}$, Q^- on $\mathbf{n}+\mathbf{e_2}$ or Q^+ on $\mathbf{n}+\mathbf{e_2}$, Q^- on $\mathbf{n}+\mathbf{e_1}$. From the 9 possible combination of policies, if the optimal policy at $\mathbf{n}+\mathbf{e_1}+\mathbf{e_j}$ and $\mathbf{n}$ are the same, then by property (b), the routing terms are non-negative. If the optimal policy at $\mathbf{n}+\mathbf{e_1}+\mathbf{e_j}$ is to route all to 2 or the optimal policy at $\mathbf{n}$ is to route all to 1, then we choose to impose policy Q^+ on $\mathbf{n}+\mathbf{e_1}$ and Q^- on $\mathbf{n}+\mathbf{e_2}$ to obtain

$$\sum_{l=1}^{2}\sum_{j=1}^{2}\lambda_l\Big[q^+_{lj}u^k_{\mathbf{n}+\mathbf{e_1}+\mathbf{e_2}+\mathbf{e_j}}-\overset{+}{q_{lj}}u^k_{\mathbf{n}+\mathbf{e_1}+\mathbf{e_j}} \\ -\overset{-}{q_{lj}}u^k_{\mathbf{n}+\mathbf{e_2}+\mathbf{e_j}}+q^-_{lj}u^k_{\mathbf{n}+\mathbf{e_j}}\Big]$$

which is non-negative because if the optimal policy at $\mathbf{n}+\mathbf{e_1}+\mathbf{e_j}$ is to route all to 2, then $u^k_{\mathbf{n}+\mathbf{e_1}+2\mathbf{e_2}}-u^k_{\mathbf{n}+\mathbf{e_1}+\mathbf{e_2}}\geq u^k_{\mathbf{n}+\mathbf{e_2}+\mathbf{e_j}}+u^k_{\mathbf{n}+\mathbf{e_j}}$ for $j=1,2$ by property (b)(i) and (ii) while if the optimal policy at $\mathbf{n}$ is to route all to 1, , then $u^k_{\mathbf{n}+\mathbf{e_2}+\mathbf{e_1}}-u^k_{\mathbf{n}+\mathbf{e_1}}\leq u^k_{\mathbf{n}+\mathbf{e_1}+\mathbf{e_2}+\mathbf{e_j}}-u^k_{\mathbf{n}+\mathbf{e_1}+\mathbf{e_j}}$ for $j=1,2$ by property (b)(i) and (ii). If the optimal policy at $\mathbf{n}+\mathbf{e_1}+\mathbf{e_j}$ is to route all to 1 or the optimal policy at $\mathbf{n}$ is to route all to 2, then we choose to impose policy Q^+ on $\mathbf{n}+\mathbf{e_2}$ and Q^- on $\mathbf{n}+\mathbf{e_1}$ and the argument is the same.

For the service terms, the proof at both servers is analogous to the proof at server 1 for property (b)(i).

To prove property (c) holds, we need to show that for all $i\in\mathcal{M}, j\neq i$ and $\mathbf{n}\in\mathcal{N}$,

$$\mathcal{T}u^k_{\mathbf{n}+2\mathbf{e_i}}-\mathcal{T}u^k_{\mathbf{n}+\mathbf{e_i}}-\mathcal{T}u^k_{\mathbf{n}+\mathbf{e_i}+\mathbf{e_j}}+\mathcal{T}u^k_{\mathbf{n}+\mathbf{e_j}}\geq 0$$

We prove it for $i=1$ and $j=2$ since the proof for $i=2$ and $j=1$ is the same. As before, we expand using (13) and break the terms up into groups. The holding cost terms are non-negative by convexity of the holding costs. The proof for the routing terms are analogous to previous proofs.

For the service terms, we break them up further according to server and show that each is non-negative. Denote x^+ and x^- as the optimal service policy for states $\mathbf{n}+2\mathbf{e_1}$ and $\mathbf{n}+\mathbf{e_2}$, respectively. The proof for server 1 applies policy x_1^+ to $\mathbf{n}+\mathbf{e_1}+\mathbf{e_2}$ and policy x_1^- to $\mathbf{n}+\mathbf{e_1}$ and is analogous to the proof at server 1 for property (b)(i). At server 2, we have two cases to consider here: $x_2^+ \geq x_2^-$ or $x_2^- \geq x_2^+$. If $x_2^+ \geq x_2^-$, then

$$\begin{aligned}
&\left(\bar{\mu}_2 - x_2^+\right) u^k_{\mathbf{n}+2\mathbf{e_1}} + x_2^+ u^k_{\mathbf{n}+2\mathbf{e_1}-\mathbf{e_2}} - \left(\bar{\mu}_2 - x_2^+\right) u^k_{\mathbf{n}+\mathbf{e_1}+\mathbf{e_2}} \\
&- x_2^+ u^k_{\mathbf{n}+\mathbf{e_1}} - \left(\bar{\mu}_2 - x_2^-\right) u^k_{\mathbf{n}+\mathbf{e_1}} - x_2^- u^k_{\mathbf{n}+\mathbf{e_1}-\mathbf{e_2}} \\
&+ \left(\bar{\mu}_2 - x_2^-\right) u^k_{\mathbf{n}+\mathbf{e_2}} + x_2^- u^k_{\mathbf{n}} \\
=& \left(\bar{\mu}_2 - x_2^+\right) \left(u^k_{\mathbf{n}+2\mathbf{e_1}} - u^k_{\mathbf{n}+\mathbf{e_1}+\mathbf{e_2}} - u^k_{\mathbf{n}+\mathbf{e_1}} + u^k_{\mathbf{n}+\mathbf{e_2}}\right) \\
&+ \left(x_2^+ - x_2^-\right) \left(u^k_{\mathbf{n}+\mathbf{e_2}} - u^k_{\mathbf{n}} - u^k_{\mathbf{n}+\mathbf{e_1}} + u^k_{\mathbf{n}+\mathbf{e_1}-\mathbf{e_2}}\right) \\
& x_2^+ \left(u^k_{\mathbf{n}+2\mathbf{e_1}-\mathbf{e_2}} - u^k_{\mathbf{n}+\mathbf{e_1}} - u^k_{\mathbf{n}+\mathbf{e_1}-\mathbf{e_2}} + u^k_{\mathbf{n}}\right)
\end{aligned}$$

All three rows are non-negative by property (c). The proof for the case of $x_2^- \geq x_2^+$ is analogous with policy x_2^+ imposed on state $\mathbf{n}+\mathbf{e_1}$ and policy x_2^- imposed on $\mathbf{n}+\mathbf{e_1}+\mathbf{e_2}$. ■

V. Bounds and Approximation

We know from [11] that if we only have 1 server, the optimal average cost rate z^* can be efficiently calculated. Hence to derive bounds for our problem, a reasonable approach would be to find ways to approximate our general M servers case with 1 server problems. The key difference between the 1 server problem and the M servers problem is the existence of routing. Bounds can thus be obtained if we can do away with routing and it turns out that there are two ways to do it.

One way is done by letting $d_{ij} = 0$ for all i and j. The problem is now equivalent to having a centralized router where all demands arrive at, and the one router will decide which server each demand should go to. This is still a non-trivial problem to solve. We thus further simplify by treating the whole M-server system as one queue with arrival rate $\lambda = \sum_{i \in \mathcal{M}} \lambda_i$. The next step is to determine the appropriate holding cost rate, $\underline{\mathrm{h}}$ and processing cost rate, $\underline{\mathrm{c}}$. To be a lower bound, the cost rates have to satisfy $\forall \mathbf{n} \in \mathcal{N}, \sum_i h_i(n_i) \geq \underline{\mathrm{h}}\left(\sum_i n_i\right)$ and $\forall x_i \in A_i, \sum_i c_i(x_i) \geq \underline{\mathrm{c}}\left(\sum_i x_i\right)$. We construct $\underline{\mathrm{h}}$ and $\underline{c}$ in the same manner and the construction for $\underline{\mathrm{h}}$ is as follows: First define $h_{min}(n) = \min_i \{h_i(n)\}$. Then let $\hat{h}_{min}$ be the lower convex envelope of h_{min}. Finally we obtain $\underline{h}(n) = M\hat{h}_{min}(n/M)$. This is a lower bound due to convexity and Jensen's inequality:

$$\sum_{i=1}^{M} h_i(n_i) \geq M \frac{1}{M} \sum_{i=1}^{M} \hat{h}_{min}(n_i) \geq \underline{h}\left(\sum_{i=1}^{M} n_i\right) \tag{19}$$

As an example, if the holding cost is the total delay, i.e. $\forall i, h_i(n_i) = n_i$, equality holds throughout in (19) and $\underline{h}(n) = n$.

To obtain the upper bound, we set $d_{ij} = \infty$ for all $i \neq j$. Routers no longer route demands to other routers and without routing, the problem decouples into M separate 1-server problem. The sum of the optimal average cost rate of the M 1-server problem thus gives us a simple upper bound on z^*. A better upper bound can be obtained by determining the optimal stationary fraction of demand to route. Note that since the results of [11] are numerical, such a routing fraction can only be numerically computed.

TABLE I

Algorithm for piecewise linear approximations to the relative value function subject to monotonicity constraint

Step 0	Initialize $v^{0,0}$ and set $j = 1$
Step 1	Choose a starting state $\mathbf{n} \in \mathcal{N}$ and set $k = 1$
Step 2	Calculate the new relative value function $\hat{v}_{\mathbf{n}}$
Step 3	Calculate the vector $y_{\mathbf{m}} = \begin{cases} (1-b_j)v^{j,k-1}_{\mathbf{m}} + b_j \hat{v}_{\mathbf{n}} & \text{, if } \mathbf{m} = \mathbf{n} \\ v^{j,k-1}_{\mathbf{m}} & \text{, otherwise} \end{cases}$
Step 4	Project the updated estimate to the space of monotone relative value functions. $v^{j,k} = \Pi(y)$
Step 5	If $k = \bar{k}$ and $j = \bar{j}$, terminate the algorithm. Else if $k = \bar{k}$, set $j = j+1$, $v^{j,0} = v^{j-1,\bar{k}}$ and go to step 1. Else set $k = k+1$, determine the next state $\mathbf{n}$, and go to step 2.

The tightest upper bound can be obtained through the undiscounted version of value iteration [15]. The value iteration here is performed on a truncated state space where the truncation is achieved by choosing a queue capacity $\bar{n}_i$ for each server i and then setting $h_i(n_i) = \infty, \forall n_i > \bar{n}_i$. The new holding cost defines a new problem and with a finite number of state space, value iteration converges to the optimal solution of this new problem. Since the holding cost is replaced with one that is uniformly larger, the final result of value iteration is an upper bound on the optimal cost.

Another upper bound is obtained through an approximate dynamic programming approach. Dynamic programming approaches such as value iteration and policy iteration is often criticized for its scalability issue, also known as the curse of dimensionality. Take value iteration as an example. Since we set a queue capacity of 50 demands for the two servers, for each iteration, we need to perform $50 \times 50 = 2500$ relative value function updates per iteration. Now suppose instead that $M = 10$, each iteration would now require $50^{10} \approx 9.8 \times 10^{17}$ relative value function updates. Assuming you have a 1 GHz CPU that can perform 10^9 updates per second, it would have taken about 31 years for the CPU to perform just one iteration.

Approximate dynamic programming (ADP) [12] is proposed as an efficient method to find near-optimal solutions. ADP encompasses a wide array of heuristics and the algorithm that we are using here is piecewise linear approximations to the relative value function subject to monotonicity constraint (proved in Theorem 3). The algorithm is given in Table I.

We first give a basic explanation of the algorithm before going into the details. At the most basic level, the algorithm is a random walk on the state space. Each random walk is of length $\bar{k}$ and at the end of each walk, the random walk is restarted with a new starting state. In total, we have $\bar{j}$ random walks. At each step of the random walk j, we calculate a new value to the relative value function of the state that we are visiting. The relative value function is then updated according to a stepsize b_j which varies with each walk. After the update, since we know from Theorem 3 that the relative value function is monotonic in $\mathcal{N}$, we project the updated relative value function to the space of monotonic functions.

We will now explain the implementation details of the algorithm. First note that similar to value iteration, we choose a server capacity and truncate the state space at $\bar{n}_i$ for each server. At step 0, we initialize with an initial estimate of the relative value functions. Unlike value iteration, not all initial estimate will work. In order for the algorithm to work, we find that the policy calculated from the initial relative value function must induce a Markov chain with a stationary distribution that is concentrated on the states close to 0. Without such a policy, the random walk has a tendency to drift to the higher states and combined with the monotonicity projection (more on this later), the random walks will stay at the higher states and thus induce a stationary distribution that is concentrated on states far from 0.

At step 1, we choose a starting state for a new random walk. The reason we need to restart the random walk instead of having one long random walk is because the random walk may get trapped in certain states due to the monotonicity projection and thus never explore other states. Also, since we know states close to 0 affect the cost much more than states that are far away, the starting state should be chosen to be state 0 or states that are close to it.

At the second step, similar to value iteration, for step k of random walk j, we calculate the new relative value function via the ACOE:

$$\hat{v}_{\mathbf{n}} = \min_{\mathbf{x},Q}\left\{ \frac{1}{\tau_{\mathbf{n}}(\mathbf{x})}\left[h(\mathbf{n}) + c(\mathbf{x}) + \sum_{i\in\mathcal{M}:n_i>0} x_i v^{j,k-1}_{\mathbf{n}-\mathbf{e_i}} + \sum_{i\in\mathcal{M}}\sum_{g\in\mathcal{M}} \lambda_i q_{ig}\left(v^{j,k-1}_{\mathbf{n}+\mathbf{e_g}} + d_{ig}\right) - z_{est}\right]\right\} \tag{20}$$

where z_{est} is our current estimate of the expected average cost. An estimate that we have found to work well is obtained by solving for z_{est} for the ACOE for state 0:

$$z_{est} = h(0) + \min_{Q} \sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{M}} \lambda_i q_{ig}\left(v^{j,k-1}_{\mathbf{e_g}} - v^{j,k-1}_{0} + d_{ig}\right) \tag{21}$$

Once we have the new relative value function, we update the relative value function of the visited state using weighted average with a stepsize $b_j = \frac{1}{(\gamma+j)}$, where $\gamma > 0$. We remark here that there are many other stepsizes which works well. To get a good feel of viable stepsizes, see [12].

After updating, we project the relative value function to the space of monotonic functions. The projection is required for the algorithm to yield consistently good results. Our projection operation is a minimum Euclidean norm projection and its details and proof can be found at section VIII-A of the appendix. We illustrate our scheme with an example where $M = 2$ and the state space is truncated to only 3×3. The example here show how the projection is actually implemented and is different from the tree structure construction in the appendix since we do not have to worry about dual variables.

Lastly, the next step of the random walk is chosen probabilistically according to the optimal service and routing policy of the current state. Such a policy is given by the argument that achieves optimality in (20). Similarly, once the algorithm terminates, the optimal policy is determined from the final relative value function for each state by finding the argument that achieves optimality in (20) . With the optimal policy, we can compute the stationary distribution of the Markov chain and hence the expected average cost.

VI. Numerical Example with $M = 2$

In this section, we specialize to the case of $M = 2$ and perform simulations to verify our previous findings. The effort cost function used in the simulation is $c_i(x) = \frac{1}{2}x^2, x \in [0,100], \forall i \in \mathcal{M}$ while the holding cost is assumed to be $h_i(n) = n, \forall i \in \mathcal{M}$. Hence, we have that $\phi_i(y) = \frac{1}{2}y^2$,$\psi_i(y) = y$, $\underline{h}(n) = n$, and $\underline{c}(x) = \frac{1}{4}x^2$. The arrival rate to the first server is held fixed at $\lambda_1 = 1$ while for the second server, the arrival rate can take on the values $\lambda_2 = 1, 2, 5$ and 10. The routing cost between the two servers are assumed to be symmetric, i.e. $d_{12} = d_{21} \triangleq d$ and has values $d = 0.01, 2, 5$ and 100. For value iteration, the state is restricted to 50×50 since we find no significant improvement in the optimal average cost rate if the state space is expanded further. This implies that we can treat the relative value function and policy derived from value iteration as the optimal solutions. The state space of the ADP algorithm is similarly truncated to 50×50 so that the results can be compared against value iteration's. The ADP algorithm runs for 1000 random walks which restart at state 0 and has length 200 each. The stepsize rule that we use is $b_j = 1/(3 + \lfloor j/20 \rfloor)$.

The numerical results are summarized in Table II. $\underline{z}$ is the lower bound, $\bar{z}_{val}$ is obtained from value iteration, $\bar{z}_{ADP}$ is the value from the ADP algorithm, $\bar{z}_r$ is the upper bound with optimal stationary routing and $\bar{z}$ is the upper bound with no routing. For the case where $\lambda_1 = 1, \lambda_2 = 5$ and $d = 2$, the graphs of the relative cost functions, the routing policy and the service rate of servers 1 and 2 for different states are plotted at Figure 2, Figure 3, Figure 4(a) and Figure 4(b) respectively.

There are a few things to note from the table. The table shows that ADP performs slightly worse than value iteration. It is however better than the upper bound derived with optimal stationary routing, except when the routing cost is large, since there is minimal routing in this case and $\bar{z}_r$ is already close to optimal. It is worth pointing out that for the cases when $\lambda_2 = 2$ or 5, ADP seems to perform better when the routing cost increases from 2 to 5 to 100. We can explain this phenomenon by first noticing that $\bar{z}$ and $\bar{z}_{val}$ are very close for these cases, implying that the optimal policy does not involve much routing. When the routing cost is low, ADP is more likely to produce a policy with more routing involved than when the cost is high, and are thus further from the optimal solution.

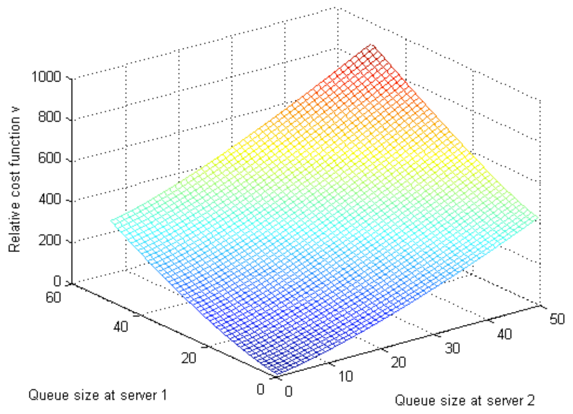
The figures verify that the relative cost function is monotonically nondecreasing, the optimal routing policy is a threshold

TABLE II

BOUNDS AND ESTIMATES FOR THE OPTIMAL AVERAGE COST RATE FOR FIXED $\lambda_1 = 1$

λ_2	d	$\underline{z}$	$\bar{z}_{val}$	$\bar{z}_{ADP}$	$\bar{z}_r$	$\bar{z}$
1	0.01	2.8658	3.3328	3.3754	3.6782	3.6782
1	2	2.8658	3.6772	3.7353	3.6782	3.6782
1	5	2.8658	3.6781	3.7282	3.6782	3.6782
1	100	2.8658	3.6781	3.7175	3.6782	3.6782
2	0.01	4.9514	5.5635	5.5680	6.1786	6.3988
2	2	4.9514	6.3587	6.4801	6.3987	6.3988
2	5	4.9514	6.3987	6.4053	6.3987	6.3988
2	100	4.9514	6.3987	6.4061	6.3987	6.3988
5	0.01	14.012	14.894	15.278	16.408	20.114
5	2	14.012	18.154	19.138	19.316	20.114
5	5	14.012	19.985	20.477	20.114	20.114
5	100	14.012	20.114	20.481	20.114	20.114
10	0.01	38.630	39.757	40.091	42.825	62.135
10	2	38.630	46.790	48.227	50.738	62.135
10	5	38.630	56.312	57.581	58.759	62.135
10	100	38.630	62.135	62.731	62.135	62.135

Fig. 2. Relative cost function v for $\lambda_1 = 1, \lambda_2 = 5, d = 2$



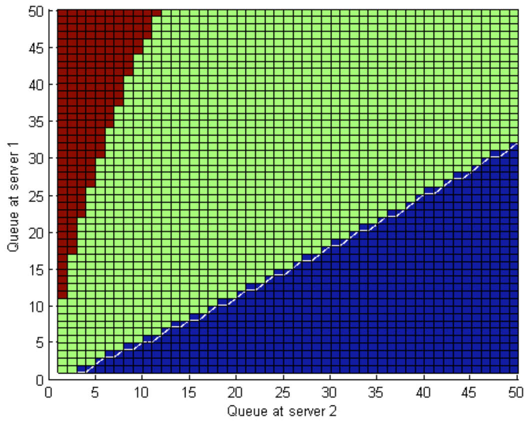
policy and also that the optimal service rate is monotonically nondecreasing if the queue size of other servers are held fixed.

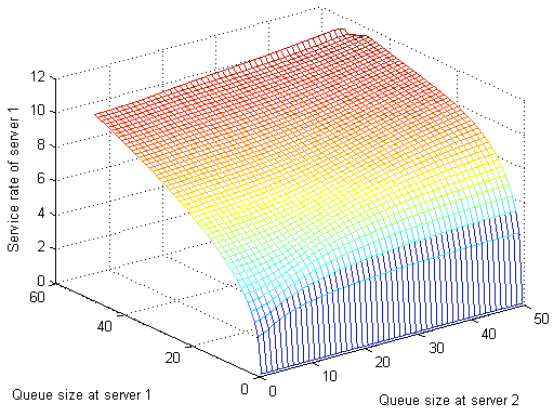
VII. CONCLUSION

This paper discusses optimal joint speed scaling and load balancing for a network of servers. Assuming an $M/M/1$ model for each queue, we show the existence of an optimal stationary policy. We further characterize that the optimal routing policy is acyclic and bipartite. The relative cost function is demonstrated to be monotonically non-increasing with each queue size. For $M = 2$, we show that the optimal service policy is non-decreasing in queue size and the optimal routing policy is a threshold policy.

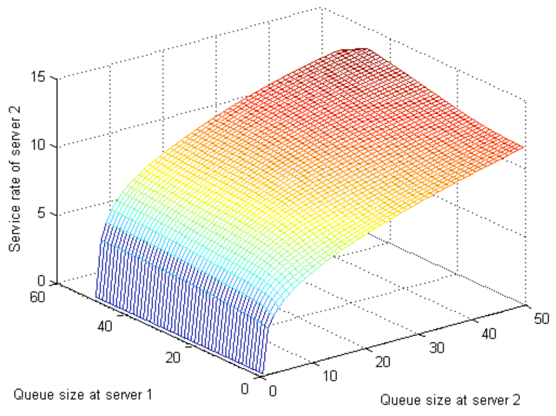
There are a number of ways this preliminary work can be extended. First, we can establish the structural properties for general M. Second, although the ADP algorithm provided is faster than value iteration, it still suffers from the curse of dimensionality as it has to store the relative cost function of all states. A significant improvement would be to represent the whole state space with just a reasonable amount of parameters and modify the ADP algorithm to update the parameters at

Fig. 3. Routing policy for for $\lambda_1 = 1, \lambda_2 = 5, d = 2$. In the red region, server 1 routes to server 2, in the green region, no routing occurs and in the blue region, server 2 routes to server 1





(a) Service rate policy of server 1



(b) Service rate policy of server 2

Fig. 4. Service rate policy for $\lambda_1 = 1, \lambda_2 = 5, d = 2$

each step. Finally, it will be of great interest to see to what extent this joint optimal solution can be decentralized. That will also bring the theoretical results closer to practice.

References

[1] R. H. Katz, "Tech titans building boom," *IEEE Spectrum*, February 2009.

[2] G. Inc., "Efficient computing: Data centers." http://www.google.com/corporate/green/datacenters/.

[3] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in *ISCA '07: Proceedings of the 34th annual international symposium on Computer architecture*, (New York, NY, USA), pp. 13–23, 2007.

[4] A. Wierman, L. Andrew, and A. Tang, "Power-aware speed scaling in processor sharing systems," in *INFOCOM 2009, IEEE*, pp. 2007 –2015, 19-25 2009.

[5] M. Andrews, A. Anta, L. Zhang, and W. Zhao, "Routing for energy minimization in the speed scaling model," in *INFOCOM, 2010 Proceedings IEEE*, pp. 1 –9, 14-19 2010.

[6] M. Andrews, A. Anta, L. Zhang, and W. Zhao, "Routing and scheduling for energy and delay minimization in the powerdown model," in *INFOCOM, 2010 Proceedings IEEE*, pp. 1 –5, 14-19 2010.

[7] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs, "Cutting the electric bill for internet-scale systems," in *Proc. ACM SIGCOMM*, (New York, NY, USA), pp. 123–134, 2009.

[8] A. Qureshi, *Power-Demand Routing in Massive Geo-Distributed Systems*. PhD thesis, Massachusetts Institute of Technology, 2010.

[9] L. Rao, X. Liu, L. Xie, and W. Liu, "Minimizing electricity cost: Optimization of distributed internet data centers in a multi-electricity-market environment," in *INFOCOM, 2010 Proceedings IEEE*, pp. 1 –9, 14-19 2010.

[10] L. Rao, X. Liu, M. Ilic, and J. Liu, "Mec-idc: joint load balancing and power control for distributed internet data centers," in *ICCPS '10: Proceedings of the 1st ACM/IEEE International Conference on Cyber-Physical Systems*, (New York, NY, USA), pp. 188–197, ACM, 2010.

[11] J. M. George and J. M. Harrison, "Dynamic control of a queue with adjustable service rate," *Oper. Res.*, vol. 49, no. 5, pp. 720–731, 2001.

[12] W. B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2007.

[13] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York, NY, USA: John Wiley & Sons, Inc., 1994.

[14] H. Kobayashi and B. L. Mark, *System Modeling and Analysis: Foundations of System Performance Evaluation*. Upper Saddle River, NJ, USA: Prentice Hall Press, 2008.

[15] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Athena Scientific, 2001.

[16] A. Arapostathis, V. S. Borkar, E. Fernández-Gaucherand, M. K. Ghosh, and S. I. Marcus, "Discrete-time controlled markov processes with average cost criterion: a survey," *SIAM J. Control Optim.*, vol. 31, pp. 282–344, March 1993.

[17] S. Stidham Jr. and R. R. Weber, "Monotonic and insensitive optimal policies for control of queues with undiscounted costs," *Oper. Res.*, vol. 37, no. 4, pp. 611–625, 1989.

[18] R. R. Weber and S. Stidham Jr., "Optimal control of service rates in networks of queues," *Adv. Appl. Prob.*, vol. 19, pp. 202–218, 1987.

[19] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004.

[20] B. Hajek, "Optimal control of two interacting service stations," *Automatic Control, IEEE Transactions on*, vol. 29, pp. 491 – 499, jun. 1984.

VIII. Appendix

A. Monotonicity Projection

Denote $\bar{n}_i$ as the queue capacity for server i. The state space is thus $\mathcal{N} = [0, \bar{n}_1] \times \ldots \times [0, \bar{n}_M]$. The convex optimization problem that we want to solve is

$$\min_v \frac{1}{2} \sum_{\mathbf{n} \in \mathcal{N}} (v_\mathbf{n} - y_\mathbf{n})^2 \tag{22}$$
$$\text{s.t. } v_{\mathbf{n}+\mathbf{e_i}} - v_\mathbf{n} \geq 0, \forall \mathbf{n}, \mathbf{n} + \mathbf{e_i} \in \mathcal{N}, i \in \mathcal{M}$$

Denote $\omega^i_{\mathbf{n}+\mathbf{e_i}}$ as the dual variable associated with the inequality $v_{\mathbf{n}+\mathbf{e_i}} - v_\mathbf{n} \geq 0$, the Karush-Kuhn-Tucker (KKT) conditions [19] are

$$v_\mathbf{n} - y_\mathbf{n} = \sum_{i: n_i > 0} \omega^i_\mathbf{n} - \sum_{i: n_i < \bar{n}_i} \omega^i_{\mathbf{n}+\mathbf{e}_i} \tag{23}$$
$$\omega^i_\mathbf{n} \geq 0, \forall 0 < n_i \leq \bar{n}_i, \mathbf{n} \in \mathcal{N}, i \in \mathcal{M} \tag{24}$$
$$\omega^i_{\mathbf{n}+\mathbf{e_i}} (v_{\mathbf{n}+\mathbf{e_i}} - v_\mathbf{n}) = 0, \forall \mathbf{n}, \mathbf{n} + \mathbf{e_i} \in \mathcal{N}, i \in \mathcal{M} \tag{25}$$
$$v_{\mathbf{n}+\mathbf{e_i}} - v_\mathbf{n} \geq 0, \forall \mathbf{n}, \mathbf{n} + \mathbf{e_i} \in \mathcal{N}, i \in \mathcal{M} \tag{26}$$

Recall that after each update of the ADP algorithm, only one value changes. Suppose state $\mathbf{m}$ is the updated state, then after the update, there are three possibilities

a. y is already monotonic in the state space. set $v = y$ and we are done.
b. there exists $i \in \mathcal{M}$ such that $y_\mathbf{m} > y_{\mathbf{m}+\mathbf{e_i}}$.
c. there exists $i \in \mathcal{M}$ such that $y_\mathbf{m} < y_{\mathbf{m}-\mathbf{e_i}}$.

We will look at case b and show that our projection scheme, as mentioned in Section V, is an optimal solution of (22) by checking the KKT conditions. All the steps that follow can be done analogously for case c.

To start with, we construct a tree structure with state $\mathbf{m}$ as the root node and associate it with the value of $y_\mathbf{m}$. We also construct a set L of nodes which initially has $\mathbf{m}$ as its only element. $^{(\dagger)}$Next, we check all adjacent states of $\mathbf{m}$ for monotonicity violation. All such adjacent states are added to the tree as child nodes of $\mathbf{m}$. If the node is $\mathbf{m} + \mathbf{e_i}$, associate the node with value $y_{\mathbf{m}+\mathbf{e_i}}$ and the link connecting $\mathbf{m} + \mathbf{e_i}$ and $\mathbf{m}$ with $\omega^i_{\mathbf{n}+\mathbf{e_i}}$. Once we are done, we check all the child nodes of nodes in L , find the node $\mathbf{m}'$ with the minimum value. If the value of $\mathbf{m}'$ is not strictly lower than the value of any node in L, we can proceed to update the value of the links. If not, we add $\mathbf{m}'$ to L. Following that, we average the values of all nodes in L and update their values with the average. Finally, we repeat the process from $^{(\dagger)}$ onwards with $\mathbf{m}'$ instead of $\mathbf{m}$ until we have no more states to add to L. Do note that any state can only be added to the tree once.

Before updating the link values, we remove all nodes that are not in L from the tree. The final average value of all nodes in L is denoted $\bar{v}$. The link value update proceeds in a bottom-up manner from the leaf nodes. Suppose the node being updated is $\mathbf{l}$, its parent node is $\mathbf{l} - \mathbf{e_i}$ and its g child nodes are $\mathbf{l} + \mathbf{e_{j_1}}, \ldots, \mathbf{l} + \mathbf{e_{j_g}}$. We update by setting $\omega^i_\mathbf{l} = \bar{v} - y_\mathbf{l} + \sum_{k=1}^{g} \omega^{j_k}_{\mathbf{l}+\mathbf{e_k}}$ and thus the dual variable $\omega^i_\mathbf{n}$ represents the aggregate increment due to averaging of state $\mathbf{n}$ and all its descendants.

Setting all other dual variables that are not updated to zero, $v_\mathbf{n} = y_\mathbf{n}$ if $\mathbf{n} \notin L$ and $v_\mathbf{n} = \bar{v}$ if $\mathbf{n} \in L$, we can easily check that our ω and v satisfy the KKT conditions and are hence optimal.