# Beyond Shortest Queue Routing with Heterogeneous Servers and General Cost Functions

Esa Hyytiä
University of Iceland

Rhonda Righter
University of California Berkeley

Sigurður Gauti Samúelsson
Aalto University

## ABSTRACT

Routing jobs to parallel servers is a common and important task in today's computer systems. Join-the-shortest-queue (JSQ) routing minimizes the mean response time under rather general settings as long as the servers are identical and service times are independent and exponentially distributed. Apart from this, surprisingly few optimality results exist, mainly due to the complexities arising from the infinite state spaces. Indeed, it is difficult to analyze the performance of any given routing policy. In this paper, we consider an elementary job routing problem with heterogeneous servers and general cost structures. By a novel approximation, we reduce the state space to finite size, which enables us to estimate the mean performance, and to determine (practically) optimal routing policies, for a large class of cost structures. We demonstrate the approximation and its application to job routing policy optimization in numerical examples.

## CCS CONCEPTS

• **Mathematics of computing** → *Queueing theory*; • **Theory of computation** → *Markov decision processes*; Routing and network design problems; • **Information systems** → Data centers;

## KEYWORDS

routing jobs, JSQ, SED, heterogeneous servers, general cost function

## 1 INTRODUCTION

Routing jobs to parallel servers has been a long standing problem class for queueing theory. The problem was first studied by Haight already in 1958 [6]. Today, the same problem arises in many new contexts. For example, when routing data traffic in the Internet, alternative routes can be modelled as parallel servers. Similarly, in cloud computing, each task needs to be assigned to one of the available servers. In supercomputing, the time scales are longer but the same fundamental question appears. Moreover, the heterogeneity of computing hardware is increasing both in large-scale systems comprising several (thousands of) physical computers, as well as within a single physical device (cf. GPUs vs. CPUs, or new heterogeneous multi-core architectures such as those introduced by ARM for mobile devices, where some cores have higher capacity at the expense of higher energy consumption).


*VALUETOOLS 2017, December 5–7, 2017, Venice, Italy*

ACM ISBN 978-1-4503-6346-4/17/12.
https://doi.org/10.1145/3150928.3150946

In this paper, we study one of the most elementary routing problems, where both job inter-arrival times and service times are exponentially distributed, so the state information is the number of jobs at each server. For clarity, we consider systems with two heterogeneous parallel servers subject to a large class of cost structures. The modelling approach itself generalizes straightforwardly to $K > 2$ servers at the cost of computational complexity. We discuss this later. One of the most popular routing policies is Join-the-Shortest-Queue (JSQ), which chooses the server with the fewest jobs. JSQ has been shown to be optimal in some specific cases, but, especially when the service rates are unequal, the exact analysis of the system becomes surprisingly tedious. The key idea in our approach is to accurately model the system where decisions matter the most, and rely on appropriate approximation elsewhere.

The main contribution of this paper is a novel modification of the system model with arbitrary cost structures, yielding a finite state space, which in turn enables us (i) to estimate the mean performance of arbitrary routing policies that are reasonable when the system is congested (i.e., stabilize the system), and (ii) to determine (near) optimal routing policies. Moreover, we obtain numerical evidence on how quickly policy iteration converges for this type of (modified) routing system. In particular, we observe that the first policy iteration round tends to yield the largest improvement (a phenomenon that has been assumed in numerous papers). These new (practically) optimal policies serve also as benchmarks when evaluating, e.g., simple (yet robust) policies such as JSQ.

### 1.1 Related Work

Routing problems have been studied actively during the last decades in very different contexts. Three classes of results are relevant to us: exact optimality results, approximate performance analysis, and heuristics for approximate optimization.

In terms of optimality results, Winston [17] showed that JSQ minimizes the mean response time under exponential assumptions.[1] JSQ has been further analyzed in [1, 7, 9, 15]. With heterogeneous servers, the natural generalization is the *Shortest-Expected-Delay* (SED) routing[2] which chooses the server with the smallest expected response time. Foschini [4] has shown that SED is asymptotically optimal in the heavy traffic limit.

The most common performance measure is mean response time, which is also non-trivial to compute in general for dynamic routing policies. However, under exponential assumptions good approximations exist for JSQ. For example, Nelson and Philips [11] develop a systematic approach based on the observation that the total number of jobs in the system under JSQ tends to behave similarly to the M/M/k system (with a shared queue). Then conditioning on the total number of jobs, one still needs to estimate the length of the

[1] Poisson arrival process, and exponentially distributed service times.
[2] Sometimes referred to as the Shortest expected Delay Routing (SDR).

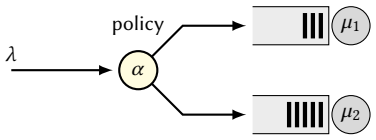


**Figure 1: Two server routing system.**

shortest queue in order to find the steady state distribution. Our approach is based on the same observation, but we model the system accurately for states with a small number of jobs where the routing decision can be critical. See also [3]. Selen et al. [13] show that the steady state distribution for two heterogeneous exponential servers under SED can be expressed as a series of product forms that can be determined recursively, enabling the computation of, e.g., the mean response time, numerically. Analysis of queueing systems tends to become harder when exponential assumptions are relaxed, including systems with JSQ routing. Some results do exist, e.g., Gupta et al. [5] consider JSQ with a general job size distribution and processor sharing (PS).

The third class of results provide good routing heuristics that outperform JSQ and SED for heterogeneous systems with different cost structures. The basic routing problem is a classical Markov decision process (MDP) with an infinite state space. If the state space were finite, the optimal routing policy would be trivially available (at least numerically) by carrying out policy or value iteration until it converges. In our setting, one often resorts to heuristic routing policies obtained by starting from a static policy, where the value function can be computed, and then carrying out *one policy iteration round*. This approach, referred to as *first policy iteration* (FPI), tends to yield an efficient, though generally not optimal, policy. The FPI approach has been utilized in numerous papers [2, 8, 10].

We also study the routing problem in the MDP framework. Instead of trying to solve the original problem directly, we first develop an approximation for the system that has a finite number of states. Our approach is similar to *the successive lumping method* [14], where the state space is partioned in such a way that the stationary distribution can be computed recursively (at least for finite systems). In contrast to [14], we partition the state space into two sets: the finite primary set includes states with few jobs where routing decisions tend to be most critical, and the secondary infinite set includes states with many jobs. Moreover, we first assume a fixed routing such as JSQ or SED in the secondary set, and then "compress" the infinite subspace to a single super state in a novel manner allowing us to handle also heavy load scenarios accurately. This approach enables us to analyze the system (with any load) and to compute the optimal routing policy exactly for System D. This in turn will provide an accurate and efficient heuristic for the original problem. Our approach also yields a computationally efficient procedure to estimate the mean performance (under any load) with respect to a large class of cost structures and routing policies (including JSQ and SED as special cases).

## 2 MODELLING

For clarity, we first assume $K = 2$ parallel servers and minimize the mean response time. The developments generalize to arbitrary cost functions and $K > 2$ servers, as will be discussed later in Sections 2.5 and 2.6.

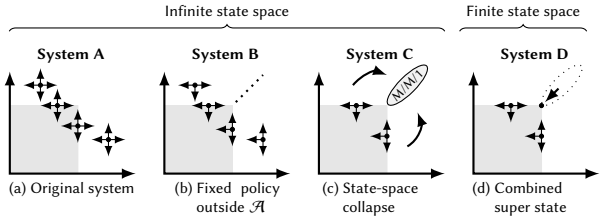


**Figure 2: Steps taken in the approximation.**

### 2.1 System A: Original model

The basic model we consider is illustrated in Figure 1 and is essentially the same as in [2]:

(1) Jobs arrive according to Poisson process with rate $\lambda$ and they are routed immediately upon arrival to one of the available servers.
(2) The system has two parallel servers ($K = 2$), where the service time at server $k$ is exponentially distributed with parameter $\mu_k$. In general, the servers are heterogeneous, $\mu_1 \neq \mu_2$. Let $\mu = \mu_1 + \mu_2$.
(3) We consider the number-aware setting, where in state $\mathbf{n} = (i, j)$ server 1 has $i$ jobs and server 2 $j$ jobs.
(4) Costs are incurred at rate $r_{ij} = i + j$, which according to Little's result corresponds to the response time.

This model, referred to as System A, is a two dimensional MDP. Even though it is elementary, finding the optimal routing policy is a surprisingly difficult problem due to the infinite state space, except when $\mu_1 = \mu_2$ and the objective is response time minimization [17]. With heterogeneous servers and arbitrary cost functions, one typically resorts to heuristics like JSQ/SED, or efficient routing policies based on FPI or Gittin's index [2].

### 2.2 System B: Fixed routing when many jobs

Next we will modify the system one step at a time, eventually obtaining an MDP with a finite state space, as illustrated in Figure 2. In the first step, we limit our focus to those states that we deem to be the most important:

- Routing decisions tend to be most crucial when servers have only few jobs, and therefore we consider optimizing decisions only in a finite number of states near the origin,

$$\mathcal{A} = \{(i, j) \mid i < n,\ j < m\},$$

  where $(n, m)$ are free parameters (eventually defining the size of the final system's state space).
- Elsewhere, a fixed default policy $\alpha_0$ kicks in. Define $\mathcal{S}$ as the union of $\mathcal{A}$ and its boundary $\mathcal{B}$,

$$\mathcal{S} = \{(i, j) \mid i \leq n,\ j \leq m\} \text{ and } \mathcal{B} = \mathcal{S} \setminus \mathcal{A}.$$

  We assume that $\alpha_0$ is such that departures from set $\mathcal{S}$ to states outside it take place only through state $(n, m)$, i.e., the routing decisions along the boundary $\mathcal{B}$ lead towards $(n, m)$.

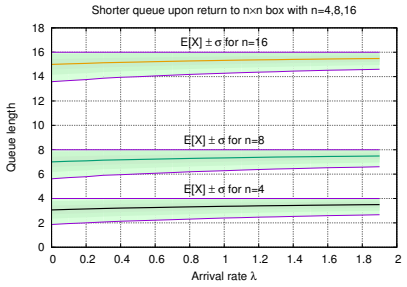


**Figure 3: Shorter queue upon returning to n × n box for n = 4, 8, 16. Typically the return point is close to (n, n), justifying the simplification referred to as the state-space collapse.**

Returns to set $\mathcal{S}$ can happen anywhere on the boundary, as illustrated in Figure 2(b).

Hence, at this point, we have simply fixed the routing decision in states where we believe that SED (or a similar policy) is near optimal (in the sense that use of it does not significantly reduce the achievable mean performance). Heuristically, *"when there is an abundance of jobs, keep all servers busy"*. This system with routing policy fixed outside $\mathcal{A}$ is referred to as System B and is depicted in Figure 2(b). We still have an infinite state space to deal with, even though the routing action is free only in a finite number of states.

Past work analyzing JSQ (with identical servers) has made the important observation that in higher states, when both $i, j \gg 0$, the system tends to stay near the diagonal and $i \approx j$. In particular, e.g., [11] assumes that the total number of jobs, $N = N_1 + N_2$, with JSQ behaves approximately the same way as in the M/M/2 queue, yielding an approximation for the steady state distribution of $N$. In fact, this is exactly the so-called heavy-traffic approximation [4]. We take advantage of the same phenomenon in this paper, but allow any stable load, $0 < \rho < 1$.

**Example 1.** Consider a system with two identical servers, $\mu_1 = \mu_2 = 1$, and $n = m$. The arrival rate $\lambda$ is varied from zero to a heavily-loaded system with $\lambda \approx 2$. Initially, the system is in state $(n+1, n)$ corresponding to the first state after departing the $n \times n$ box. Upon return, the longer queue has $n$ jobs, and the shorter has a random number $X \in \{0, \ldots, n\}$. Figure 3 illustrates the mean and variability of $X$ as a function of $\lambda$. We can see that the variability is highest when $\lambda \to 0$, which is easy to understand as arrivals will push the state closer to the diagonal under JSQ. For $\lambda \to 0$, it is easy to show analytically that the difference between the shorter and longer queue, denoted by $D$, has a truncated geometric distribution,

$$\mathbb{P}\{D = i\} = \begin{cases} q^i(1-q), & i = 0, \ldots, (n-1), \\ q^n, & i = n, \end{cases}$$

where $q = 1/2$, and as $X = n - D$, we have

$$\mathbb{E}[X] = n - 1 + 2^{-n},$$
$$\sigma_X^2 = 2 - 4^{-n} - (1+2n)2^{-n},$$

which rapidly converge to $n - 1$ and 2, respectively, for large $n$. Hence, when $n$ is larger, typically the state in $\mathcal{S}$ where the system returns is one of $(n, k)$ or $(k, n)$ for $k = n, n-1, n-2, n-3$.

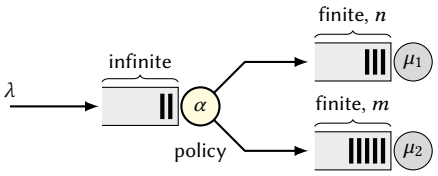


**Figure 4: System C (with two servers).**

## 2.3 System C: (Partial) state-space collapse

With the insight of Example 1 in mind, we next propose that instead of analyzing the original model, we simplify our system by *assuming* that state-space collapse occurs beyond $\mathcal{S}$ [4]. More specifically, we define a new System C that agrees with System B for the states in $\mathcal{S}$, but for states in $\mathcal{S}^c$, reduces to a standard M/M/1 queue with arrival rate $\lambda$ and service rate $\mu = \mu_1 + \mu_2$. When a job arrives at state $(n, m)$ in System C, it starts an M/M/1 mini-busy period during which we can think of jockeying[3] being allowed, or that the two servers collaborate and work on one job at a time until the state $(n, m)$ is reached again. Equivalently, System C corresponds to a system where each server has a finite number of system places, and if all those are full, then a job is held at the dispatcher, as illustrated in Figure 4. The cost rate in the higher states is denoted by $\tilde{r}_i$, where $i$ is the total number of jobs *in the system*, i.e., with the response time metric,

$$\tilde{r}_i = i.$$

In the general case, $\tilde{r}_i$ should resemble the corresponding exact cost rates $r_{ij}$ near the diagonal, $i \approx j$. Note that this approximation tends to underestimate the costs during the mini busy period a bit (depending on the cost structure).

The state space of System C is depicted in Figure 2(c). It is significantly smaller than those of Systems A and B, but still infinite.

## 2.4 System D: Aggregated super state

Let us next consider System C from the moment it enters state $(n, m)$ until it moves to state $(n-1, m)$ or $(n, m-1)$. This corresponds to a mini busy period in the M/M/1 queue initially having $n + m$ jobs, with mean duration

$$\mathbb{E}[B] = \frac{1/\mu}{1 - \lambda/\mu} = \frac{1}{\mu - \lambda}.$$

Moreover, the fraction of time there are $n + m + i$ jobs, $i = 0, 1, \ldots$, is geometrically distributed,

$$\pi^*_{n+m+i} = (1-\rho)\rho^i,$$

and therefore the mean cost rate $r_{n^*}$ during the mini busy period (where the job routing policy is "fixed") is

$$r_{n^*} = (1-\rho)\sum_{i=0}^{\infty} \tilde{r}_{n+m+i}\rho^i = (1-\rho)\sum_{i=0}^{\infty}(n+m+i)\rho^i$$
$$= n + m + \frac{\rho}{1-\rho},$$

where the first two terms corresponds to the baseline of having at least $n + m$ jobs, and the third term adds the mean number of additional jobs present during the mini busy period.

[3] Jockeying refers to moving jobs between queues after the initial assignment.

Next we replace "the M/M/1 queue" in System C with an equivalent super state $n^* = (n, m)$, which has mean duration $\mathbb{E}[B]$ and incurs costs at rate $r_{n^*}$. This model is referred to as System D. The corresponding transition and cost rates of the (modified) Markov process for state $n^*$ are,

$$\begin{aligned} q_{n^*,(n-1,m)} &= \mu_1(1-\rho), \\ q_{n^*,(n,m-1)} &= \mu_2(1-\rho), \\ r_{n^*} &= n + m + \frac{\rho}{1-\rho}. \end{aligned}$$

Elsewhere within $\mathcal{S}$, the transition rates are according to the service rates $\mu_k$ and the arrival rate $\lambda$ with the destination state defined by the routing policy (e.g., JSQ). For the interior points, i.e. states in $\mathcal{A}$, whether to route a new job to Server 1 or Server 2 can be chosen freely, while on the boundary $\mathcal{B}$ a suitable default policy $\alpha_0$ is assumed. The resulting Markov (decision) process of System D has $(n+1)(m+1)$ states, as depicted in Figure 2(d), and well-defined cost rates in each state. Once the routing policy is fixed, we have a finite Markov process for which the steady state distribution $\pi_{ij}$ can be easily computed. The mean cost rate is then given by $r = \sum_{i,j} \pi_{ij} r_{ij}$. Note that in terms of (expected) costs, Systems C and D are equivalent.

## 2.5 General cost functions

Our approach allows for other performance metrics besides response times. For example, we may incur a a unit cost if an arriving job sees more than two jobs ahead of itself upon arrival (at the same server) assuming FCFS service. This cost may better represent how people tend to feel about queueing.

Let $a_{ij}$ denote the probability that an arriving job is routed to server 1 in state $(i, j)$, so with probability $1 - a_{ij}$ it is routed to Server 2. Typically, $a_{ij} = 0$ or $a_{ij} = 1$, but this definition allows also probabilistic routing in every state.[4] Due to PASTA, instead of incurring costs upon arrival, we can define the equivalent cost rate in state $(i, j)$ as

$$r_{ij} = \lambda \left( a_{ij} \mathbf{1}(i > 2) + (1 - a_{ij}) \mathbf{1}(j > 2) \right).$$

The above holds both for the original System A and the modified System D, and $\tilde{r}_i = \lambda$ given $n, m > 2$. For System D we have

$$\begin{aligned} q_{n^*,(n-1,m)} &= \mu_1(1-\rho), \\ q_{n^*,(n,m-1)} &= \mu_2(1-\rho), \\ r_{n^*} &= \lambda. \end{aligned}$$

With $a_{ij}$ fixed, we again have a finite Markov process for which the steady state distribution $\pi_{ij}$ and the mean cost rate $r$ can be easily determined.

## 2.6 General case with K servers

In this section, we illustrate how the approach generalizes to $K > 2$ servers and an arbitrary box defining the boundary for $\mathcal{S}$. As before, we assume that the routing on the boundary ensures a single exit state (see Figure 2(c)), and that the fixed routing policy outside $\mathcal{S}$ is such that a state-space collapse occurs and the return state is (approximately) the same as the exit state, so the M/M/1 model for the mini busy period is justified.

[4] For example, the load balancing random split (RND) is defined by $a_{ij} = \mu_1/(\mu_1 + \mu_2)$.

Let $\mathbf{m} = (m_1, \ldots, m_K)$ denote the dimensions of the finite state space, where $m_k$ is the maximum number of jobs in server $k$ we are tracking. Hence, the number of states is

$$M = \prod_{k=1}^{K} (m_k + 1).$$

An arbitrary state is denoted with $\mathbf{n} = (n_1, \ldots, n_K)$, where $n_k$ is the number of jobs in server $k$. We can easily map the $K$-dimensional state space to one dimension using

$$s(\mathbf{n}) = \sum_{i=1}^{K} \left( n_i \prod_{k=1}^{i-1} (m_k + 1) \right).$$

Then an arbitrary probabilistic routing is defined by an $M \times K$ matrix $\alpha$, where $\alpha_{ik}$ defines the fraction of jobs routed to server $k$ in state $i$. We assume that $\alpha$ honors the boundaries, so that, e.g., $\alpha_{Mk} = 0$ for all $k$ (in the full system, arriving jobs are "blocked"). With the routing policy fixed, the transition rate matrix $\mathbf{Q}$ is easy to obtain.

Consider first the departure rates. For an arbitrary state $\mathbf{n}$, given $n_k > 0$ and server $k$ is busy, the corresponding departure rate shows up in $\mathbf{Q}$ as

$$q_{s(\mathbf{n}),s(\mathbf{n}-\mathbf{e_k})} = \mu_k,$$

where $e_k$ denotes a vector with all elements zero except the $k$th element that is one. For the combined super state $\mathbf{n} = \mathbf{m}$, i.e., for $i = s(\mathbf{m}) = M$, we have

$$q_{s(\mathbf{m}),s(\mathbf{m}-\mathbf{e_k})} = (1-\rho)\mu_k, \qquad \forall\, k.$$

For the arrivals, with $i = s(\mathbf{n})$, given $\alpha_{ik} > 0$, we have

$$q_{s(\mathbf{n}),s(\mathbf{n}+\mathbf{e_k})} = \alpha_{ik}\, \lambda.$$

The cost rates, e.g., with respect to response time, are simply

$$r_{\mathbf{n}} = n_1 + \ldots n_K + \mathbf{1}(\mathbf{n} = \mathbf{m}) \frac{\rho}{1-\rho}.$$

# 3 EVALUATING THE APPROXIMATION

In this section, we apply our approximation to estimate the mean response time with JSQ and SED. This exercise has two purposes: first it validates the use of System D, and second, it yields a sequence of increasingly more accurate estimates for the mean response time. As already mentioned, JSQ minimizes the mean response time with identical servers [17], and hence an analytic expression for its performance, even if an estimate, is valuable.

## 3.1 Two identical servers with JSQ

Suppose we have $K = 2$ identical servers. Let $\mathbb{E}[N]$ denote the mean number of jobs in the system. Recall the two limits:

$$\textit{light-traffic limit}, \qquad \mathbb{E}[N] \approx 2\rho, \qquad \text{as } \rho \to 0, \tag{1}$$

$$\textit{heavy-traffic limit}, \qquad \mathbb{E}[N] \approx \rho/(1-\rho), \qquad \text{as } \rho \to 1. \tag{2}$$

By straightforward analysis of System D with $n = 0, 1, \ldots,$ one obtains a sequence of estimates for the mean number of jobs. The first four are given in Table 1, where $\rho = \lambda/(2\mu)$. Case $n = 0$ is the approximation where the original system is immediately replaced with the M/M/1 queue (equivalently, when two servers can process the same job concurrently). Similarly, $n = 1$ with identical servers reduces to the M/M/K system. Due to the assumed jockeying, we

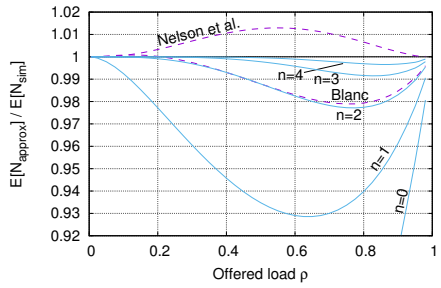


**Figure 5: Numerical evalution of the estimates for the mean response time with two identical servers and JSQ.**

**Table 1: Estimates for the mean number of jobs with two identical servers and JSQ based on System D with n = 0, . . . , 3.**

$$\mathbb{E}[N_0] = \frac{\rho}{1-\rho} \qquad \text{(M/M/1)}$$

$$\mathbb{E}[N_1] = \frac{2\rho}{1-\rho^2} \qquad \text{(M/M/2)}$$

$$\mathbb{E}[N_2] = \frac{2\rho(2+(3-\rho)\rho(1+\rho))}{(1-\rho)(1+2\rho)(2+\rho+\rho^2)}$$

$$\mathbb{E}[N_3] = \frac{2\rho(4+\rho(14+\rho(23+\rho(16+7\rho-4\rho^3))))}{(1-\rho)(1+2\rho)(1+2\rho(1+\rho))(4+\rho(2+\rho+\rho^2))}$$

know that these approximations are strict lower bounds for the mean performance under JSQ.

Figure 5 shows the ratio of the estimate of $\mathbb{E}[N]$ to the simulated value for $n = 0, \ldots, 4$. We see that the estimates quickly become very accurate, thus supporting the assumption that System D serves as a reasonably good model for the original system when $n$ is sufficiently large.

The above results for $\mathbb{E}[N]$ are not new in the sense that compact and accurate approximations can be found from the literature (also for $K > 2$ servers). For two identical servers, Blanc [3] gives,

$$\mathbb{E}[N] \approx \frac{\rho\left(4+10\rho-5\rho^2\right)}{(1-\rho)(7\rho+2)},$$

whereas Nelson's and Philips' approximation [11] in the same case is,

$$\mathbb{E}[N] \approx \frac{2\rho(1+\rho+\rho^2-\rho^3)}{(1-\rho)(1+\rho)^2}.$$

The accuracy of Blanc's expression is approximately the same as that of $\mathbb{E}[N_2]$, while the accuracy of Nelson's and Philips' expression is somewhere between those of $\mathbb{E}[N_2]$ and $\mathbb{E}[N_3]$ (in terms of maximum relative error), see Figure 5. These approximations are easy to evaluate and they all (except $\mathbb{E}[N_0]$) behave correctly at the limits (1) and (2). Additionally, our approximations are lower bounds. However, our main goal is to find (near) optimal routing policies for heterogeneous systems and System D with a finite state space is designed with this goal in mind.

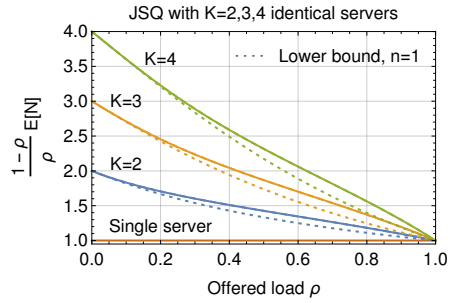


**Figure 6: Mean occupation scaled by $\rho/(1-\rho)$ with JSQ and K = 2, 3, 4 identical servers (solid lines). Lower bounds with n = 1 (i.e., M/M/K system) are depicted with dashed lines.**

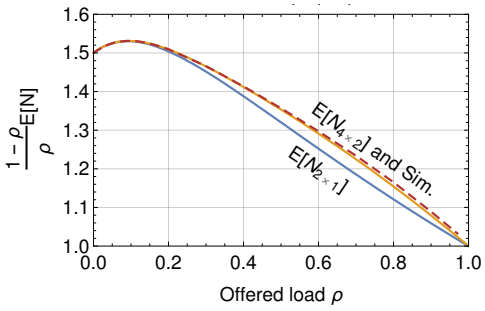


**Figure 7: Numerical results with SED when $(\mu_1, \mu_2) = (2, 1)$.**

## 3.2 Three and four identical servers with JSQ

Let us now use our approach to a bit larger systems of three and four identical servers fed by JSQ. The mean number of jobs $\mathbb{E}[N]$ in the M/M/1 queue is $\rho/(1-\rho)$. With $K$ parallel servers, having equal service rates, and fed by JSQ, $\mathbb{E}[N]$ is obviously higher as sometimes servers can be idle even if there are jobs in the system. Figure 6 illustrates the penalty, based on our approximation, due to having multiple servers instead of a single faster one for $K = 2, 3, 4$. The solid curves correspond to estimates obtained using sufficiently large values of $n$, and they can be easily verified to be surprisingly accurate with simulations. Dashed lines correspond to the lower bounds obtained with $n = 1$, i.e., to the M/M/K system (where routing is replaced with a common queue).

## 3.3 Two heterogeneous servers with SED

Suppose next a heterogeneous system, $\mu = (2, 1)$, with SED(2) where ties are resolved in favor of the slower server 2. For the mean number of jobs, with $(n, m) = (2, 1)$, one obtains

$$\mathbb{E}[N_{2\times1}] = \frac{3\rho\left(6+26\rho+26\rho^2+5\rho^3-3\rho^4\right)}{(1-\rho)\left(12+46\rho+74\rho^2+39\rho^3+9\rho^4\right)}.$$

In Figure 7 we have depicted $\mathbb{E}[N_{2\times1}]$, $\mathbb{E}[N_{4\times2}]$ and simulated results (dashed curve). We can see that our approximation with $(n, m) = (4, 2)$ is already surprisingly accurate.

# 4 NEAR OPTIMAL ROUTING POLICY

In this section, we shift our focus to finding the optimal routing policy. To this end, we consider System D and determine the optimal routing for it. This is then assumed to serve as a (near) optimal routing policy also for the original System A (within $\mathcal{S}$), and therefore we refer to it as the NO policy. We let $n_k$ denote the number of jobs in server $k$, so that the state of the whole system is $\mathbf{n}$.

## 4.1 Admission costs

So far, we have assumed that each state $\mathbf{n}$ has a certain cost rate $r_{\mathbf{n}}$. In particular, for the response time metric we have simply $r_{\mathbf{n}} = \sum_k n_k$. To make routing decisions it is convenient to think of costs incurred upon admission of a customer. Therefore, we define an *admission cost function*, denoted by $c_i^{(k)}$, which is the cost when a job is added to server $k$ currently having $i$ jobs. The admission cost may depend on the server's service rate.

(1) It is easy to see that the expected response time in server $k$,

$$c_i^{(k)} = \frac{i+1}{\mu_k} \quad \text{and} \quad r_{\mathbf{n}} = \sum_k n_k,$$

are equivalent (cf. Little's result).

(2) In general, due to PASTA, an arbitrary $c_i^{(k)}$ is equivalent to

$$r_{\mathbf{n}} = \lambda \sum_k a_{\mathbf{n}}(k) c_{n_k}^{(k)}.$$

where $a_{\mathbf{n}}(k)$ denotes the fraction of jobs routed to server $k$ in state $\mathbf{n}$. Thus, e.g., for a threshold $\theta$ on the queue length,

$$c_i^{(k)} = \mathbf{1}(i > \theta) \quad \text{and} \quad r_{\mathbf{n}} = \lambda \sum_k a_{\mathbf{n}}(k) \mathbf{1}(n_k > \theta),$$

are equivalent.

In general, $c_i^{(k)}$ is some non-negative increasing function of $i$.

## 4.2 Policy iteration

Recall that we managed to reduce the infinite state space of the original system to a classical MDP problem with a finite number of states and well-defined cost rates in each state. Such problems are commonly solved by using policy or value iteration methods [12, 16]. The former involves solving Howard's equations yielding relative values,

$$r_{\mathbf{n}} - r + \sum_{\mathbf{n}' \neq \mathbf{n}} q_{\mathbf{n}'\mathbf{n}} (v_{\mathbf{n}'} - v_{\mathbf{n}}) = 0,$$

where $r_{\mathbf{n}} = r_{\mathbf{n}}(\alpha)$ is the cost rate in state $\mathbf{n}$ (with policy $\alpha$), and $r = r(\alpha)$ is the mean cost rate. Fixing, e.g., $v_0 = 0$, the above set of linear equations can be easily solved, yielding both the value function $v_{\mathbf{n}}$ and the mean cost rate $r$. Next the policy iteration step is carried out,

$$\alpha^*(\mathbf{n}) \triangleq \operatorname*{argmin}_k \left( c_{n_k}^{(k)} + v(\mathbf{n} + \mathbf{u}_k) - v(\mathbf{n}) \right),$$

where $c_{n_k}^{(k)}$ is the admission cost to server $k$ at state $n_k$, and $\mathbf{u}_k$ denotes a vector with 1 at position $k$ and otherwise zero. This is repeated until the procedure converges ($r$ remains the same). Typically, policy iteration converges rapidly, and later in the examples we see that this is the case also here.

## 4.3 Difference between models

Let us next compare any two systems (a) and (b) that make the same decisions within $\mathcal{S}$ and honor the boundary $\mathcal{B}$, so differ only outside $\mathcal{B}$. This includes System B with arbitrary, but stable, routing decisions outside $\mathcal{S}$, as well as Systems C and D. The long-run mean cost rates are $r^{(a)}$ and $r^{(b)}$, where the superscripts indicate the system, and in general $r^{(a)} \neq r^{(b)}$.

Then consider an arbitrary state $\mathbf{n} \in \mathcal{S}$. As the two systems make the same decisions until reaching the corner point $n^*$, their sample paths during this time interval are identical. It follows that for the value functions, $v_{\mathbf{n}}^{(a)}$ and $v_{\mathbf{n}}^{(b)}$,

$$v_{\mathbf{n}}^{(a)} - v_{n^*}^{(a)} = \mathbb{E}[C(\mathbf{n}, n^*) - r^{(a)} T(\mathbf{n}, n^*)],$$
$$v_{\mathbf{n}}^{(b)} - v_{n^*}^{(b)} = \mathbb{E}[C(\mathbf{n}, n^*) - r^{(b)} T(\mathbf{n}, n^*)],$$

where $C(\mathbf{n}_1, \mathbf{n}_2)$ and $T(\mathbf{n}_1, \mathbf{n}_2)$ denote the costs incurred and the duration of time before a system initially in state $\mathbf{n}_1$ reaches state $\mathbf{n}_2$) for the first time.[5] Due to the identical routing decisions within $\mathcal{S}$, the only difference on the right-hand side is in the mean cost rates, $r^{(a)} \neq r^{(b)}$. However, given that for $\mathbf{m}$ sufficiently large, $r^{(a)} \approx r^{(b)}$, the corresponding relative values are also practically identical within $\mathcal{S}$.

If (a) is System B with parallel queues with *any*, including the optimal, stable routing, except for on the boundary, and (b) is a system where outside $\mathcal{S}$ the system is reduced to an M/M/1 queue (or an equivalent super state), then costs such as the mean response time are clearly smaller, $r^{(b)} < r^{(a)}$, because (a) could have idling in states outside $\mathcal{S}$. Later we show numerically that, with respect to mean response time, when two servers are identical and the optimal routing policy JSQ is applied in every state in both (a) and (b), $r^{(a)} \approx r^{(b)}$ for $n = m > 2$, Hence, also the corresponding value functions are practically equivalent, and the adverse effects of simplifying the system to a finite Markov process are negligible.

## 4.4 Numerical examples

Next we will illustrate the procedure and the NO policies for heterogeneous two server systems with $\mu_1 \geq \mu_2$. As a reference, we consider the following three heuristic policies: (i) Load balancing random split (RND) that chooses the server $k$ with probability of $\mu_k/(\mu_1 + \mu_2)$; (ii) JSQ; and (iii) Shortest Expected Delay (SED) that chooses the server that minimizes the expected response time [13, 14]. Ties with JSQ and SED are resolved in favor of the faster or slower server. We indicate the tie breaking rule in parentheses, e.g., JSQ(1) resolves ties in favor of the faster Server 1.

**Example 2.** Suppose $(\mu_1, \mu_2) = (h\mu, \mu)$, where $h \geq 1$ measures the asymmetry in the service rates. System D with JSQ and $n = m = 2$ is the smallest system with a non-trivial routing decision. That is, which server should be chosen in state $(1, 0)$? When $\rho \to 0$, the greedy SED is optimal and chooses the faster server only if $h > 2$.

The system has 9 states and both steady-state distribution and value function can be easily computed analytically, and the threshold for routing also the second job to Server 1 can be computed. It

[5] Considering sample paths until reaching state $n^*$ instead of, e.g., the origin has the benefit that no state outside $\mathcal{S}$ is visited before termination. This holds as the default policy $\alpha_0$ applied on the boundary $\mathcal{B}$ forms a "surface" that ensures that the corner point $n^*$ is the only exit gate from $\mathcal{S}$ to outside.

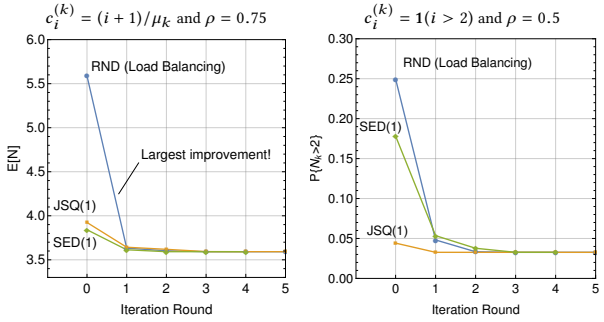


**Figure 8: Convergence of the policy iteration from different basic policies.**

turns out that the threshold increases almost linearly from 2 to 3.74 as $\rho$ increases from zero to one. That is, when the load is higher the secondary server is taken into use earlier. This observation holds also when $n > 2$ with slightly changed numerical values.

**Example 3.** Suppose $(\mu_1, \mu_2) = (3, 1)$, i.e., the secondary server is three times slower than the primary server. In the first case, the offered load is relatively high, $\rho = 0.75$, and the objective is to minimize the mean response time, $c_i^{(k)} = (i+1)/\mu_k$. In the second case, the offered load is medium, $\rho = 0.5$, and the cost function is the unit step function $c_i^{(k)} = \mathbf{1}(i > 2)$. The parameters $(n, m)$ we set to $(30, 10)$.

Figure 8 illustrates the convergence of policy iteration when starting from three different basic policies for both cases. On the $x$-axis is the iteration round (zero corresponds to the basic policy), and on the $y$-axis is the performance. We observe that it takes 3-5 rounds before the NO policy is found, and that the largest improvement from every basic policy indeed takes place in the first step, which supports the claim that FPI policies are often "near-optimal", and the basic policy does not matter so much.[6]

**Example 4.** Now we take a closer look at when SED and NO make different decisions. Let us assume that $(\mu_1, \mu_2) = (3, 1)$, the offered load $\rho$ is varied and the objective is to minimize the mean response time. The corresponding policies are illustrated in Figure 9. First we observe that SED(2) appears to be near optimal when load is low. As the load increases, NO routes the first job "earlier" to Server 2 in anticipation of new jobs arriving soon. The higher the load, the more pronounced the proactive action is. Otherwise the switch-over curves show a *"three jobs to faster server, and then one to slower"* pattern, as expected.

Figure 10 depicts the relative increase in mean response time when JSQ and SED with different tie breaking rules are used instead of NO. As expected, JSQ and SED are good and robust routing policies but not optimal (in this case). With JSQ it is clearly important to favor the faster Server 1 so that the first job goes there. SED routes the first job automatically to the faster server, and it is actually better to route a job to slower Server 2 in case of ties. We can observe that JSQ(1) increases the mean response time by about 10% and SED(2) up to 2%, except when $\rho$ is (very) low or high.

**Example 5.** Next we compare a single fast server with $\mu = 4$ to (i) two identical servers with $(\mu_1, \mu_2) = (2, 2)$, (ii) two heterogeneous servers with $(\mu_1, \mu_2) = (3, 1)$, and (iii) two heterogeneous servers with $(\mu_1, \mu_2) = (3.5, 0.5)$. With identical jobs and the mean response time metric, the single fast server is obviously the optimal configuration. For the two server systems, we consider JSQ(1), SED(2) and NO. As some systems are highly asymmetric, it is important to use an appropriate $n \times m$ box instead of an $n \times n$ square for SED and NO. The numerical results are depicted in Figure 11. Note that especially JSQ(1) *suffers* from heterogeneity in the sense that a lower mean response time can be achieved with two identical servers even though JSQ(1) favors the faster server.[7] In the homogeneous case, JSQ and SED coincide with NO. With NO, the mean response time decreases as the heterogeneity increases, as expected.

**Example 6.** Let us next consider the unit step function $c_i^{(k)} = \mathbf{1}(i > 2)$ as the admission cost. The service rates are again $(\mu_1, \mu_2) = (3, 1)$ and the offered load is $\rho = 0.5$. For policy iteration, the immediate (admission) cost $c_i = \mathbf{1}(i > 2)$ must be taken explicitly into account. NO is depicted in Figure 12 (left). We note that when both queues are too long, NO routes the new job to the *slower server*, thus minimizing the time until a job can be admitted to the system without a penalty. However, overloading (usually) the slowest server can lead to instability issues when $\rho$ is sufficiently high. This is actually an artifact of the cost model as it may well be beneficial to overload one queue in order to keep the others sufficiently short. However, the assumed JSQ/SED beyond $\mathcal{S}$ ensures stability as long as $\rho < 1$.

**Example 7.** Finally, let us consider a non-linear cost structure, where we combine the response time metric and the unit cost if queue length exceeds the chosen threshold of 2 (see Section 2.5). In this case, the cost rates in each state are simply summed. For example, in the super state $n^*$ we have

$$r_{n^*} = 2n + \frac{\rho}{1-\rho} + \lambda.$$

As before, for policy iteration, the immediate (admission) cost $c_i = \mathbf{1}(i > 2)$ must be taken into account, whereas for response time this cost is included in the state-specific cost rates.

NO is depicted in Figure 12 (right) for $\rho = 0.5$. Interestingly, in this case jobs are routed to the slow secondary server only when its queue length is below the threshold 2. Having the response time component in the cost structure discourages overloading one queue and thus prevents instability issues. In our case, the assumed JSQ outside $\mathcal{S}$ also ensures stability as long as $\rho < 1$.

## 5 CONCLUSIONS

In this paper, we have studied the classical routing problem to $K$ parallel heterogeneous servers with Poisson arrivals, exponential services, and arbitrary cost structures. Although JSQ is a widely used dynamic routing policy for such systems, it is not generally optimal as it neglects both the service rates and the cost structure. Its generalization, SED, takes the service rates into account, but neglects the cost structure.

[6] When RND is applied in every state, $\mathbb{E}[N] = 6$. However, we assumed JSQ/SED outside $\mathcal{A}$, and this is why RND (within $\mathcal{A}$) has a bit better performance in Figure 8 (left). For us, this discrepancy is irrelevant as our focus is on the optimal dynamic policies, for which JSQ/SED at higher states is a fair choice.

[7] If the asymmetry increases a bit more, JSQ(1) is worse than the load balancing RND.

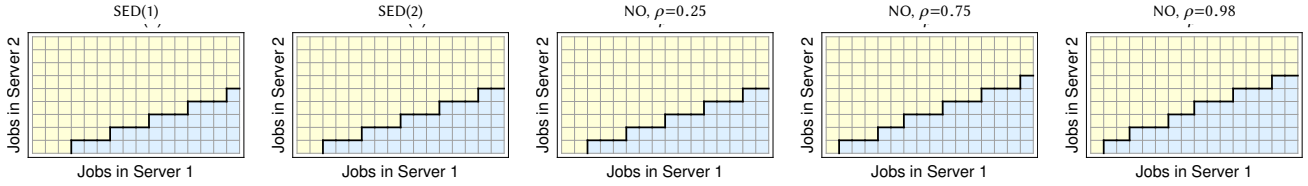


**Figure 9: Routing policies for $(\mu_1, \mu_2) = (3, 1)$ system with n = 16, where lighter (yellow) states are those in which jobs are routed to Server 1. From left to right, SED(1), SED(2), and then NO for $\rho = \{0.25, 0.75, 0.98\}$. The higher the load, the more aggressively NO utilizes the slower secondary server.**

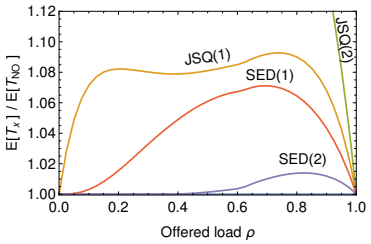


**Figure 10: Sub-optimal routing policies JSQ and SED with different tie breaking rules compared to NO when $(\mu_1, \mu_2) = (3, 1)$.**

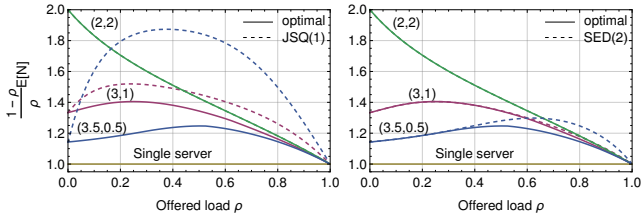


**Figure 11: Performance of different heterogeneous two server systems with JSQ(1) (dashed line on left), SED(2) (dashed line on right), and NO (solid lines) compared to a single fast server.**

We proposed an approach where the infinite $K$ dimensional state space is "compressed" to a finite $K$ dimensional box, where one corner state is a super state corresponding to collapsed version of higher states. For small systems, we obtain closed-form results and policies in symbolic form (with arbitrary $\lambda$ and $\mu_k$). Numerically the proposed approach is very efficient for a large class of routing problems with arbitrary admission costs and levels of offered load. Our numerical examples support the common claim that the first policy iteration round tends to yield the highest performance improvement. The near-optimal NO policy required a few more iteration rounds.

## ACKNOWLEDGEMENTS

This work was supported by the Academy of Finland in the FQ4BD project (grant no. 296206).

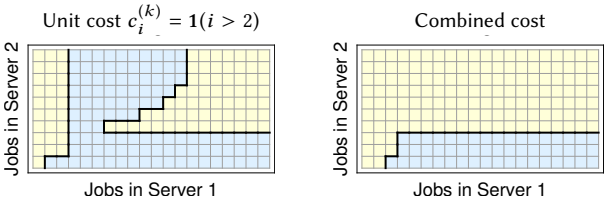


**Figure 12: The NO policy for the heterogeneous system with $(\mu_1, \mu_2) = (3, 1)$ and $\rho = 0.5$ when the admission cost is the unit step function $1(i > 2)$ (left), and a linear combination of the response time and the unit step function (right).**

## REFERENCES


[1] O. Akgun, R. Righter, and R. Wolff. 2011. Multiple Server System with Flexible Arrivals. *Advances in Applied Probability* 43 (2011), 985–1004.

[2] N.T. Argon, L. Ding, K.D. Glazebrook, and S. Ziya. 2009. Dynamic routing of customers with general delay costs in a multiserver queuing system. *Probability in the Engineering and Informational Sciences* 23 (2009), 175–203.

[3] J. P. C. Blanc. 1987. A Note on Waiting Times in Systems with Queues in Parallel. *Journal of Applied Probability* 24, 2 (1987), 540–546.

[4] G. J. Foschini. 1977. On heavy traffic diffusion analysis and dynamic routing in packet switched networks. *Computer Performance* 10 (1977), 499–513.

[5] Varun Gupta, Mor Harchol-Balter, Karl Sigman, and Ward Whitt. 2007. Analysis of Join-the-Shortest-Queue Routing for Web server Farms. *Performance Evaluation* 64, 9-12 (Oct. 2007), 1062–1081.

[6] Frank A. Haight. 1958. Two queues in parallel. *Biometrika* 45, 3-4 (1958), 401–410.

[7] Arie Hordijk and Ger Koole. 1990. On the optimality of the generalized shortest queue policy. *Probability in the Engineering and Informational Sciences* 4, 4 (1990), 477–487.

[8] Esa Hyytiä, Samuli Aalto, and Aleksi Penttinen. 2012. Minimizing Slowdown in Heterogeneous Size-Aware Dispatching Systems. *ACM SIGMETRICS Performance Evaluation Review* 40 (June 2012), 29–40. Issue 1.

[9] Pravin K. Johri. 1989. Optimality of the shortest line discipline with state-dependent service rates. *European Journal of Operational Research* 41, 2 (1989), 157–161.

[10] K. R. Krishnan. 1987. Joining the right queue: a Markov decision rule. In *Proc. of the 28th Conference on Decision and Control*. 1863–1868.

[11] R. D. Nelson and T. K. Philips. 1989. An approximation to the response time for shortest queue routing. *SIGMETRICS Perform. Eval. Rev.* 17, 1 (April 1989), 181–189.

[12] Martin L. Puterman. 2005. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley.

[13] Jori Selen, Ivo Adan, Stella Kapodistria, and Johan van Leeuwaarden. 2016. Steady-state analysis of shortest expected delay routing. *Queueing Systems* 84, 3 (Dec. 2016), 309–354.

[14] L.C. Smit. 2016. *Steady-state analysis of large scale systems : the successive lumping method*. Ph.D. Dissertation. Leiden University.

[15] D. Towsley, P. D. Sparaggis, and C. G. Cassandras. 1992. Optimal routing and buffer allocation for a class of finite capacity queueing systems. *IEEE Trans. Automat. Control* 37, 9 (Sept. 1992), 1446–1451.

[16] Peter Whittle. 1996. *Optimal Control: Basics and Beyond*. Wiley.

[17] W. Winston. 1977. Optimality of the shortest line discipline. *Journal of Applied Probability* 14 (1977), 181–189.