Scalable Load Balancing in Interference-prone Queueing Systems

Abstract

We study a dispatching problem in large-scale queueing systems, where each server independently alternates between serving jobs at a fast rate (when the server is functioning normally) and at a slow rate (when the server is undergoing performance degradation). This problem arises in cloud computing settings when dispatching jobs to a large set of virtual machines (VMs) located across a variety of physical servers. As not all physical resources are easily partitioned across VMs on the same server, VMs frequently experience a temporary and unpredictable—yet detectable—form of performance degradation known as interference. We address the problem of load balancing in interference-prone VMs, where one must immediately dispatch each incoming job to one of many VMs to minimize average response times. This problem is further complicated because each VM undergoes interference independently of the others. We propose several distributed dispatching policies that dispatch incoming requests based on the interference and busy status of a randomly sampled subset of the servers under the power-of-d-choices paradigm. Using a combination of mean field analysis and recursive renewal reward, we evaluate the performance of these heuristics exactly in a variety of settings while deducing and proving a number of surprising results. In particular, we find that while using interference status information when making dispatching decisions can help reduce the mean response time, making use of this information naively can be very costly.

keywords: Queueing systems; Dispatching policies; Mean field analysis; Interference; Virtual machines

1 Introduction

We study the dispatching problem in many-server systems where the service rate of each server alternates between a fast state and a slow state in a Markov-modulated fashion. This problem arises in cloud computing, for example, wherein servers occasionally experience temporary slowdowns due to a phenomenon known as *virtual machine interference*. Public cloud services are computing services provided by third-party providers—such as Amazon, Dropbox, Salesforce, Slack, and Microsoft Azure—over the public internet. These services function by hosting numerous virtual machines (VMs) on a single physical machine. The most significant advantage of public cloud services is that they are inherently scalable, allowing for allocating additional resources as an organization's computing needs grow. However, this flexibility comes at a cost: the hosted VMs often share the same physical resources. As a result, delays in the job execution time can occur due to waiting on shared resources [16, 18, 29, 38].

We model such *interference* as an unpredictable, yet detectable, drop in the speed (service rate) of a particular

VM (server). At any given time, some VMs may undergo interference (functioning at a lower service rate) while others continue to function normally. Should an incoming job be sent to be queued at an interference-free (fast) server that is already busy working on other jobs? Or, are we better off sending it to an idle server under interference, hoping that the server will begin to function normally soon? Addressing such questions is the primary focus of this paper. Indeed, the policies by which job requests are dispatched to VMs in large-scale network systems—and how these policies take into account the presence of interference—have significant effects on the performance (i.e., response times) of such systems.

We position our paper by contrasting how it stands apart from the existing work across several related literature streams, starting with a brief review of the work on classical dispatching problems. We know that the Join the Shortest Queue (**JSQ**) policy is optimal—with respect to a variety of metrics, including the mean response time—in systems with parallel homogeneous servers with decentralized queues when the service time has a non-decreasing hazard rate [37]. Moreover, **JSQ** is still optimal in homogeneous multi-server systems with exponential service times, even if the dispatcher is informed of the job arrival times in advance [8]. If the job size information is available to the dispatcher, we can also track the total workload at the servers, which is more informative than queue length information alone. This more detailed information allows the dispatcher to implement the Join the Least Workload (**JLW**) policy, where the job is sent to the server with the minimum work. It turns out that **JLW** is optimal for homogeneous server systems when the job size information is available to the dispatcher and when the job size variability is low [13]. By contrast, when job sizes are highly variable, one can outperform **JLW** by grouping servers to serve different job size ranges [13].

Our work departs significantly from the classical work above in several respects, the most important being that we consider servers that can run at different speeds at any given time. Load balancing problems in the presence of heterogeneous servers (with respect to their service rates) are often referred to as *slow server problems* in the literature. There is a significant body of work in this area, with a recurring theme being that optimal dispatching policies are known to exhibit threshold structures in a variety of settings; i.e., the decision of sending the job to a fast server depends on the number of jobs queued at the slower servers [15, 19, 26, 31]. Meanwhile, our work lies in a unique middle ground. While, *at any given time*, our servers run at different speeds (as in slow server problems), they are *stochastically identical* (as in classical dispatching, although classical servers do not change speeds). This middle ground comes with technical challenges that complicate performance analysis. The fact that servers are stochastically identical also has important qualitative consequences, as we show in this paper. In particular, policies that favor routing to fast servers tend to be unstable in slow server problems; they do not exhibit this tendency in our setting.

There also exists a stream of work—lying entirely outside the scope of analytic performance evaluation and operations research in general—dedicated to mitigating the effects of interference *at the hardware level*. However, despite fruitful efforts in this area, interference remains a problem that can frequently cause significant reductions in job processing speeds [6, 27]. This suggests that there is a need for analytical work that informs the design of efficient dispatching policies in the presence of interference; this paper aims to address this need partially. Other work in this area is still scarce. Most notably, [33, 34] study load dispatching policies under interference

in a system with a central queue (a queue forms in front of the dispatcher, and there are no queues in front of the servers). Using a Markov Decision Process (MDP) model, [34] proves that the optimal dispatching policy minimizes response times by using slow servers only when the current queue length exceeds a threshold. While the problem formulation is technically valid for any number of servers, most of the analysis in [34] is dedicated to the two-server setting. Our work is significantly differentiated from the aforementioned papers due to our consideration of immediate dispatching (an important assumption in settings where requests are dropped if they are not immediately sent to a VM) as opposed to the management of a central queue. Our dispatching model also yields a higher state-space dimensionality, which makes the MDP formulation too cumbersome.

Furthermore, unlike all of the work we have mentioned so far, we are interested in scalable policies that are particularly suitable for settings where the number of servers is very large, as is often the case with large cloud computing systems featuring many VMs. In such settings keeping track of the complete state information of all servers tends to be prohibitively difficult. Consequently, policies that leverage complete state information are often impractical. The *power-of-d-choices* approach obviates the need to track complete state information by having the dispatcher query only a small number of d servers uniformly at random whenever a job arrives. The dispatcher then sends the incoming job to one of the queried servers based on the information capturing the state of those (and only those) servers; d is typically a small number (e.g., two or three). Therefore, our work also shares much with the literature on *power-of-d* dispatching policies. It has been shown that the limiting probabilities for the **JSQ-**d policy (where the job is assigned to a server in accordance with the **JSQ** policy among the d servers that were queried) in systems with Poisson arrivals and exponential service times converge as the number of servers tends to infinity and that we can obtain impressive performance results even when d is as low as two [28].

Another useful and easy-to-implement policy is the **JIQ** (Join the idle Queue) policy, whereby the dispatcher knows all idle servers in the system at all times. Since idleness information is much easier to track than complete queue length information, there is little communication overhead in **JIQ**. We find works on different variations of the **JIQ** policy in the literature [24, 36]; however, most works address systems with homogeneous servers. In heterogeneous server systems, recent work shows that the **JIQ** policy is delay-optimal [32]; however, if there are significant differences in the server's speeds, this heterogeneity-unaware policy can result in high response times at low to medium system loads [10]. In systems with heterogeneous servers—including those considered in our work where servers exhibit heterogeneity based on their interference states—the servers' speed information plays a crucial role in dispatching decisions to obtain low response times.

Under homogeneous server settings, when the number of servers tends to infinity, the systems exhibit the *asymptotic independence* property under the **JSQ-**d and **JLL-**d (join the least loaded queue after querying d servers) policies; i.e., servers are asymptotically independent of one another if the state of any given server is independent of the state of any other subset of servers [3, 4]. Crucially, asymptotic independence allows for the application of Mean Field Analysis (**MFA**) to study the performance of many queueing systems [5, 14, 21–23]. Few papers use **MFA** to analyze systems with server speed heterogeneity [1, 2, 10, 12, 17]. However, none of them account for changing server speeds. By contrast, in our model, server speeds change over time, and the

current speed state of a server is only known once it is queried.

Our contribution is fourfold: (i) we introduce a novel and applicable model that—as we have argued above—is uniquely positioned within the literature, (ii) we propose a set of simple dispatching policies as well as a method to evaluate their performance using a novel blend of techniques, (iii) we observe and prove several intriguing results on these simple dispatching policies, and (iv) after studying the performance of the set of simple dispatching policies, we develop a more complex policy by leveraging optimization that outperforms all the simple dispatching policies.

The remainder of this paper is organized as follows: In §2, we present our load balancing problem of interest, and we propose a number of easily implementable (and easily understood) power-of-d dispatching policies that make dispatching decisions based on the information about the interference and/or idleness status of a randomly queried subset of the servers. In §3, we analytically measure the performance of these policies using both **MFA** and an adaptation of the recursive renewal reward (**RRR**) method [9] to Markov chains that exhibit non-unidirectional phase transitions, resulting in a novel combination of analytic techniques. While our analysis requires finding the roots of polynomials numerically, it is otherwise exact. In §4, we analyze the performance of the heuristic policies presented in §2 across a variety of problem parameters and highlight certain interesting results and properties; we provide formal proofs for several of these results. In §5, we introduce a new improved policy that is more complex than the heuristic policies in §2 in that this policy is constructed by optimally choosing policy parameters; we compare the performance of this new policy against the performance of the simpler policies. The motivation for the idea of the new policy is obtained from a recent work in the heterogeneous dispatching literature [10]. We summarize the major results of our paper in §6.

2 Model and Dispatching Policies

We consider a K-server system under the asymptotic regime (i.e., $K \to \infty$). Jobs arrive according to a Poisson process with a rate of $K\lambda$. Each server's service rate alternates independently between a fast rate μ_f (when it is not under interference) and a slow rate μ_s (when it is under interference). Servers work at rate μ_f (resp., μ_s) for an exponentially distributed duration of time with rate α_f (resp., α_s) before switching to working at the slower (resp., faster) rate. Each server has a dedicated queue and serves its jobs according to the FCFS (First Come, First Serve) discipline; our results generalize to any other work-conserving service policies. When a job arrives at the system, a central dispatcher immediately queries $d \ll K$ servers for their current interference and idleness states and uses this information to dispatch the job to one of the queried servers. We do not allow for job migration or jockeying, i.e., once a job is dispatched to a server, it stays there until its service completion. Fig. 1 shows a graphical presentation of the model.

We analyze and evaluate the performance of a set of easy-to-implement and scalable heuristic dispatching policies and compare them with respect to their mean response times $\mathbb{E}[T]$, measured as the end-to-end duration of time jobs spend in the system from arrival until service completion and departure. We consider the following six dispatching policies, one of which uses only the speed information of the queried servers, another of which

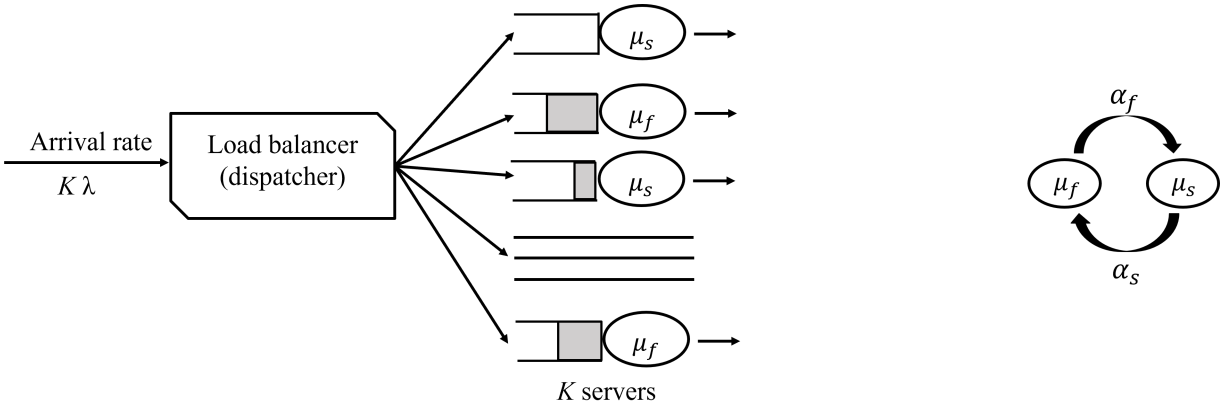


Figure 1: The dispatching model (left); servers interference states (right)

uses only the idleness information of the queried servers, and the remaining four of which use both types of information:

- **JIQ**-d: The dispatcher queries d servers and dispatches an arriving job randomly to an *idle* server, if available; if all d servers are busy, the job is dispatched randomly to one of the d servers.
- **JIQ$^{\mathbf{FP}}$**-d: The dispatcher queries d servers and dispatches an arriving job randomly to an idle server, if available, prioritizing interference-free (fast) servers; if all d servers are busy, the job is dispatched randomly to one of the d servers (speed information is not used among the queried busy servers). Hence, dispatching is done in the following order among the queried servers: (i) fast idle, (ii) slow idle, and (iii) busy.
- **JIQ$^{\mathbf{BR}}$**-d: The dispatcher follows the same procedure as in **JIQ$^{\mathbf{FP}}$**-d, except if all d servers are busy, the job is dispatched to a random busy fast server with probability $\frac{\mu_f\alpha_s}{\mu_f\alpha_s+\mu_s\alpha_f}$ and to a random busy slow server with probability $\frac{\mu_s\alpha_f}{\mu_f\alpha_s+\mu_s\alpha_f}$).
- **JFQ**-d: The dispatcher queries d servers and dispatches an arriving job randomly to a *fast* server, if available; if all d servers are under interference (functioning at the slow rate), the job is dispatched randomly to one of the d servers.
- **JIFQ**-d: The dispatcher queries d servers and dispatches an arriving job randomly to a server using both the idleness and speed statuses, prioritizing idleness over speed. Hence, dispatching is done in the following order among the queried servers: (i) fast idle, (ii) slow idle, (iii) fast busy, and (iv) slow busy.
- **JFIQ**-d: The dispatcher follows the same procedure as in **JIFQ**-d, except it prioritizes speed over idleness. Hence, dispatching is done in the following order among the queried servers: (i) fast idle, (ii) fast busy, (iii) slow idle, and (iv) slow busy.

In §3, we derive the expected response times, as the main performance of interest, under the policies listed above. In §5, we introduce another policy (**JIFQ$^{\mathbf{OPT}}$**) that provides more flexibility in job assignments and

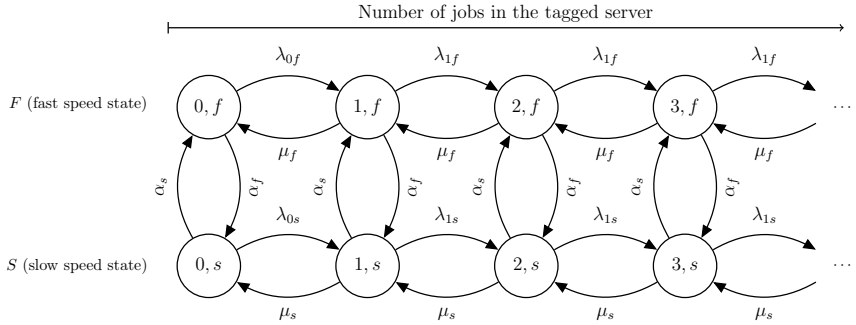


Figure 2: Markov chain for a tagged server

allows optimization over its parameters. Throughout the paper, we also occasionally consider the Random policy (**RND**), which dispatches jobs to one of the K servers randomly (with equal probabilities); **RND** serves as a benchmark policy that does not use any information about the status of the servers.

3 Performance Analysis

In this section, we determine the expected response time $\mathbb{E}[T]$ for the dispatching policies described in §2 via mean-field approximation (**MFA**). Given K is large, we apply **MFA** under the *asymptotic independence* assumption; i.e., as $K \to \infty$, the number of jobs at an arbitrary server becomes independent of the number of jobs at the other servers at any point in time when the system is in steady state (we assess the validity of this assumption via simulation in §4). Therefore, the average response time for the system is the same as the average response time for a tagged-server subsystem. Accordingly, we tag a server and study its stochastic evolution.

Fig. 2 presents the general state transition diagram for a tagged server under any of our policies with the state variable (n, x) where $n \in \{0, 1, 2, \cdots\}$ represents the number of jobs in the tagged server's subsystem (including the job in service, if any) and $x \in \{f, s\}$ represents its speed (or, interference) state (with f and s denoting the *fast* interference-free and *slow* under-interference states, respectively). Jobs are processed at rates μ_f and μ_s in the fast and slow states, respectively. The rate of change of the interference state is α_f when in the fast state and α_s when in the slow state (i.e., interference hits an interference-free server at rate α_f and leaves an under-interference server at rate α_s). Since the policies of interest introduced in §2 exhibit symmetry—in the sense that they treat all servers with the same idleness and interference state in the same way—there must exist arrival rates λ_{0f}, λ_{0s}, λ_{1f}, and λ_{1s}, which respectively correspond to the rate at which jobs arrive to the idle fast, idle slow, busy fast, and busy slow states. Therefore, the Markov chain in Fig. 2 has a repeating structure with identical transition rates for the fast states (n, f), and for the slow states (n, s) for all $n \geq 1$. The rates λ_{0f}, λ_{0s}, λ_{1f}, and λ_{1s} depend on the choice of dispatching policy, so we will derive these rates separately for each policy introduced in §3.1.

Under system stability (sufficient conditions presented in Proposition 2; see §4.1), the evolution of state (n, x) of the tagged server is governed by a positive recurrent Markov chain. This observation allows us to

leverage the Renewal Theorem [30, Chapter 3] to determine the steady-state probabilities distribution over the state space for the tagged server (and hence, for any server), from which we can determine the mean response time at the tagged server (which corresponds to the overall mean response time for the system as a whole). To calculate the exact response times under our policies of interest, we develop a method that is primarily based on the **RRR** approach [9], which allows us to take advantage of the fact that the Markov chain shown in Fig. 2) exhibits the following repeating structure:

- The probability that the tagged-server subsystem visits state $(0, y)$ before $(0, y')$ (where $y' \in \{s, f\} \setminus \{y\}$) when it is currently in state $(1, x)$ equals the probability that it visits state $(n-1, y)$ before $(n-1, y')$ when it is in state (n, x), for all $n \geq 1$ and any $x, y \in \{s, f\}$. We denote this probability by P_{xy}.
- The expected time it takes the tagged-server subsystem to visit level 0 (i.e., state $(0, f)$ or $(0, s)$) when it is in state $(1, x)$ equals the expected time it takes it to visit level $n-1$ (i.e., state $(n-1, f)$ or $(n-1, s)$) when it is in state (n, x), for all $n \geq 1$ and $x \in \{f, s\}$. We denote this expected time by T_x.

One way to interpret P_{xy} and T_x is to think in terms of busy periods: a busy period begins when a subsystem consists of a single job until it next becomes empty (i.e., until the server becomes idle). P_{xy} is the probability that a busy period starting in interference state x ends in interference state y. Meanwhile, T_x is the expected duration of a busy period beginning in interference state x.

We define a *renewal cycle* as a duration that begins when the tagged-server subsystem visits state $(0, f)$ and ends the next time the subsystem re-visits that state; i.e., the tagged-server subsystem *renews* itself each time it visits state $(0, f)$. We let $\mathcal{T}$ denote the expected duration of a renewal cycle. We consider the instantaneous reward rate at time t as $R(t) = i(t)$ where $i(t)$ is the total number of jobs residing in the tagged-server subsystem (including the one in service, if any). Let $\mathcal{R}$ be the expected reward accumulated during a renewal cycle based on the instantaneous reward rate function $R(t)$. By the renewal reward theorem, we have the following expression for the time-average number of jobs in the tagged-server subsystem, $\mathbb{E}[N]$:

$$\mathbb{E}[N] = \lim_{s \to \infty} \frac{\int_0^s R(t)\, dt}{s} = \frac{\mathbb{E}\left[\int_{\text{cycle}} R(t)\, dt\right]}{\text{Expected renewal cycle duration}} = \frac{\mathcal{R}}{\mathcal{T}}. \tag{1}$$

As long as we can determine the arrival rate to the tagged server, Eq. (1) and Little's Law yield the expected response time at the tagged server $\mathbb{E}[T]$. To determine this arrival rate, recall that we are considering a stable system for which the asymptotic independence assumption holds, and we further require that the dispatching policy treats all servers with the same idleness and interference state in the same. Hence, each server must experience the same average long-run arrival rate. Specifically, jobs must arrive at any given server—and hence, the tagged server in particular—at an average rate of λ, as the arrival rate to the entire K-server system is $K\lambda$. It then follows from Eq. (1) and Little's Law that

$$\mathbb{E}[T] = \frac{\mathbb{E}[N]}{\lambda} = \frac{\mathcal{R}}{\lambda \mathcal{T}}. \tag{2}$$

In Proposition 1, we present a system of equations that allows us to obtain $\mathcal{R}$ and $\mathcal{T}$. These equations are valid for all dispatching policies of interest and are in terms of the expected times T_f and T_s, the probabilities P_{ff}, P_{fs}, P_{sf}, and P_{ss}, and the arrival rates λ_{0f}, λ_{0s}, λ_{1f}, and λ_{1s}. To facilitate the presentation of Proposition 1, we define the following auxiliary expected reward values:

- $\mathcal{R}_x$ (for $x \in \{f, s\}$): The expected reward accumulated during a time interval starting with the tagged-server subsystem entering state $(1, x)$ until it first visits either state $(0, f)$ or $(0, s)$ (whichever happens first). Alternatively, $\mathcal{R}_x$ corresponds to the expected reward accumulated in a system that initially has exactly one job and is in interference state x until the system becomes empty (in either interference state).
- $\mathcal{R}_{0s}$: The expected reward accumulated during a time interval starting with the tagged-server subsystem entering state $(0, s)$ and ending when the system next visits state $(0, f)$. Alternatively, $\mathcal{R}_{0s}$ corresponds to the expected reward accumulated in an initially empty and under-interference system (i.e., in interference state s) until that system becomes simultaneously interference-free (i.e., in interference state f) and empty.

As a consequence of the repeating structure of the transition rates, for all $n \geq 1$, the expected reward accumulated in a time interval when the tagged server enters state (n, x) (for $x \in \{f, s\}$) until it first visits either state $(n-1, f)$ or $(n-1, s)$—or equivalently, the reward accumulated in a system that initially has exactly n jobs and is in interference state x until that system has one fewer job—will be $\mathcal{R}_x + (n-1)\,T_x$.

Proposition 1. *The expected reward accumulated during a renewal cycle $\mathcal{R}$ is obtained by solving the following system of equations:*

$$\begin{cases} \mathcal{R}_f = \dfrac{1 + \lambda_{1f}\left(T_f + \mathcal{R}_f + P_{ff}\mathcal{R}_f + P_{fs}\mathcal{R}_s\right) + \alpha_f \mathcal{R}_s}{\lambda_{1f} + \mu_f + \alpha_f} \\ \mathcal{R}_s = \dfrac{1 + \lambda_{1s}\left(T_s + \mathcal{R}_s + P_{sf}\mathcal{R}_f + P_{ss}\mathcal{R}_s\right) + \alpha_s \mathcal{R}_f}{\lambda_{1s} + \mu_s + \alpha_s} \\ \mathcal{R}_{0s} = \dfrac{\lambda_{0s}(\mathcal{R}_s + P_{ss}\mathcal{R}_{0s})}{\lambda_{0s} + \alpha_s} \\ \mathcal{R} = \dfrac{\lambda_{0f}(\mathcal{R}_f + P_{fs}\mathcal{R}_{0s}) + \alpha_f \mathcal{R}_{0s}}{\lambda_{0f} + \alpha_f}, \end{cases} \tag{3}$$

and the expected duration of a renewal cycle $\mathcal{T}$ follows:

$$\mathcal{T} = \frac{\beta_{0f} + \beta_{0s} + \lambda_{0f}\beta_{0s}T_f + \lambda_{0s}\beta_{0f}T_s}{(\alpha_f + \lambda_{0f})\beta_{0s}}, \tag{4}$$

where $\beta_{jx} = \alpha_x + \lambda_{jx}P_{xy}$ and $\gamma_x = \lambda_{1x} - \mu_x, j \in \{0, 1\}, x \in \{f, s\}$, and $y \in \{f, s\} \setminus \{x\}$.

The proofs of all propositions and lemmas are provided in the appendix.

Obtaining $\mathcal{R}$ and $\mathcal{T}$ in Eqs. (3) and (4) requires us to derive the expressions for P_{xy}, T_x, λ_{0x}, and λ_{1x}, for $x, y \in \{f, s\}$; once these values are known, the system of equations given in (3) is linear in the reward variables. We first derive the general equations for P_{xy} and T_x, which can be applied to all dispatching policies of interest, in Lemmas 1 and 2. Subsequently, in §3.1 we derive expressions for the arrival rates λ_{0x} and λ_{1x} (for $x \in \{s, f\}$); unlike the aforementioned P_{xy} and T_x values, these arrival rates depend on the choice of the

dispatching policy. Finally, we conclude this section in §3.2 by presenting Proposition 1, which can be used alongside the other results presented in this section to derive the primary performance metric of interest: the expected response time $\mathbb{E}[T]$.

Lemma 1. *The probabilities P_{ff}, P_{fs}, P_{sf}, and P_{ss} correspond to a solution to the following system of non-linear equations that satisfies P_{ff}, P_{fs}, P_{sf}, $P_{ss} \in [0,1]$:*

$$\begin{cases} P_{ff} = \dfrac{\lambda_{1f}\left(P_{ff}^2 + P_{fs}P_{sf}\right) + \alpha_f P_{sf} + \mu_f}{\lambda_{1f} + \mu_f + \alpha_f}, \\ P_{fs} = 1 - P_{ff}, \\ P_{sf} = \dfrac{\lambda_{1s}\left(P_{sf}P_{ff} + P_{ss}P_{sf}\right) + \alpha_s P_{ff}}{\lambda_{1s} + \mu_s + \alpha_s}, \\ P_{ss} = 1 - P_{sf}. \end{cases}$$

As we discuss in the proof of Lemma 1 (see Appendix A.2), this system of non-linear equations has a unique solution if the system is stable. The condition for a system to be stable in our setting is presented in §4.1.

Lemma 2. *The expected times T_f and T_s follow*

$$T_x = \frac{\gamma_y - \beta_{1f} - \beta_{1s}}{\gamma_f\beta_{1s} + \gamma_s\left(\alpha_f + \mu_f - \lambda_{1f}P_{ff}\right)}, \quad x \in \{f,s\}, y = \{f,s\} \setminus x, \tag{5}$$

where $\beta_{1x} = \alpha_x + \lambda_{1x}P_{xy}$ and $\gamma_x = \lambda_{1x} - \mu_x$.

So far, all the equations we have presented are valid for all the policies we are studying. The only remaining pieces for solving the equations in Proposition 1 and Lemmas 1 and 2 and eventually finding the expected response time $\mathbb{E}[T]$ based on Eq. (2) are the dispatching policy-dependent arrival rates λ_{0f}, λ_{0s}, λ_{1f}, and λ_{1s}. In §3.1, we derive the equations for these arrival rates for each of our policies of interest. Note that the set of equations presented in Lemmas 1 and 2 and those presented in §3.1 for the arrival rates result in a larger system of equations that must be solved simultaneously; we explain the solution procedure for solving this larger system in §3.2, after completing our presentation of the equations that make up this system.

3.1 Deriving the Arrival Rates

In order to determine the arrival rates λ_{0f}, λ_{0s}, λ_{1f}, and λ_{1s}, we need to find the steady-state probabilities of the tagged-server subsystem for states $(0,f)$ and $(0,s)$. Let π_{nx} denote the limiting probability of state (n,x) (i.e., π_{nx} is the steady-state probability that the tagged subsystem has n jobs and is in interference state x, where $x \in \{f,s\}$). Let $\pi_0 \equiv \pi_{0f} + \pi_{0s}$ (resp., $\pi_x \equiv \sum_{n=0}^{\infty}\pi_{nx}$) so that π_0 (resp., π_x) is the steady-state

probability that the tagged server is idle (resp., in interference state x):

$$\pi_f = \sum_{n=0}^{\infty} \pi_{nf} = \frac{\alpha_s}{\alpha_f + \alpha_s},$$
$$\pi_s = \sum_{n=0}^{\infty} \pi_{ns} = \frac{\alpha_f}{\alpha_f + \alpha_s}.$$

Lemma 3 provides a general equation to find π_{0f} and π_{0s}; these two probabilities will then allow us to determine the state-dependent arrival rates of interest (i.e., λ_{0f}, λ_{0s}, λ_{1f}, and λ_{1s}) associated with the various policies that we study in this paper.

Lemma 3. *Let $x \in \{f, s\}$, with $y \in \{f, s\} \setminus \{x\}$ (i.e., x corresponds to one interference state and y to the other). The steady state probabilities π_{0f} and π_{0s} are given by*

$$\pi_{0x} = \frac{\beta_{0y}}{\beta_{0f} + \beta_{0s} + \lambda_{0f}\beta_{0s}T_f + \lambda_{0s}\beta_{0f}T_s}, \tag{6}$$

where $\beta_{jx} = \alpha_x + \lambda_{jx}P_{xy}$ for $j \in \{0, 1\}$ and $\gamma_x = \lambda_{1x} - \mu_x$.

In order to obtain the arrival rates of interest, we leverage the fact that under all of our power-of-d policies, the rate at which the tagged server is queried is given by:

$$\mathbb{P}(\text{Tagged server is among the } d \text{ queried servers}) \times \text{System arrival rate} = \frac{\binom{K-1}{d-1}}{\binom{K}{d}} K\lambda = d\lambda.$$

Arrival rates: JIFQ-d. The state-dependent arrival rates under **JIFQ-d** follow Lemma 4.1:

Lemma 4.1. *The state-dependent arrival rates under* **JIFQ-d** *can be expressed in terms of the limiting probabilities, λ, and d as follows:*

$$\lambda_{0f} = d\lambda \sum_{i=0}^{d-1} \frac{\binom{d-1}{i}}{i+1} (\pi_{0f})^i (1 - \pi_{0f})^{d-1-i},$$
$$\lambda_{1f} = d\lambda \sum_{i=0}^{d-1} \frac{\binom{d-1}{i}}{i+1} (\pi_f - \pi_{0f})^i (\pi_s - \pi_{0s})^{d-1-i},$$
$$\lambda_{0s} = d\lambda \sum_{i=0}^{d-1} \frac{\binom{d-1}{i}}{i+1} (\pi_{0s})^i (1 - \pi_0)^{d-1-i},$$
$$\lambda_{1s} = \lambda(\pi_s - \pi_{0s})^{d-1}.$$

We briefly explain the equation for λ_{0f} (i.e., the job arrival rate to the tagged server when the server is in state $(0, f)$). When the tagged server is in the state $(0, f)$, there can be up to $d-1$ servers in the state $(0, f)$ among the remaining servers in the query. The number of combinations that i out of the $d-1$ servers are in the state $(0, f)$ is $\binom{d-1}{i}$, and the probability of each combination is $(\pi_{0f})^i (1 - \pi_{0f})^{d-1-i}$. Since each of the $(i+1)$ queried servers (including the tagged server sub-system) has an equal chance of receiving the job, we have the division by $i + 1$. Other expressions in Lemma 4.1 follow a similar logic.

Arrival rates: JFIQ-d. The state transition diagram for a tagged server under **JFIQ-d** is the same as that under **JIFQ-d** (Fig. 2). The state-dependent arrival rates under **JFIQ-d** follow Lemma 4.2.

Lemma 4.2. *The state-dependent arrival rates under* **JFIQ-d** *can be expressed in terms of the limiting probabilities, λ, and d as follows:*

$$\lambda_{0f} = d\lambda \sum_{i=0}^{d-1} \frac{\binom{d-1}{i}}{i+1} (\pi_{0f})^i (1-\pi_{0f})^{d-1-i},$$
$$\lambda_{0s} = d\lambda \sum_{i=0}^{d-1} \frac{\binom{d-1}{i}}{i+1} (\pi_{0s})^i (\pi_s - \pi_{0s})^{d-1-i},$$
$$\lambda_{1f} = d\lambda \sum_{i=0}^{d-1} \frac{\binom{d-1}{i}}{i+1} (\pi_f - \pi_{0f})^i (\pi_s)^{d-1-i},$$
$$\lambda_{1s} = \lambda(\pi_s - \pi_{0s})^{d-1}.$$

Arrival rates: JIQ-d. The **JIQ-d** policy uses only the idleness information. We see that the tagged server experiences an arrival rate of λ_0 when the server is idle and λ_1 when busy. Lemma 4.3 gives the expressions for λ_0 and λ_1. The resulting state transition diagram for the tagged server under **JIQ-d** is the same as that under **JIFQ-d** (see Fig. 2) where $\lambda_{0f} = \lambda_{0s}$ and $\lambda_{1f} = \lambda_{1s}$.

Lemma 4.3. *The state-dependent arrival rates under* **JIQ-d** *can be expressed in terms of the limiting probabilities, λ, and d as follows:*

$$\lambda_{0f} = \lambda_{0s} = d\lambda \sum_{i=0}^{d-1} \frac{\binom{d-1}{i}}{i+1} {\pi_0}^i (1-\pi_0)^{d-1-i},$$
$$\lambda_{1f} = \lambda_{1s} = \lambda(1-\pi_0)^{d-1}.$$

Arrival rates: JIQ$^{\mathbf{FP}}$-d. Under this policy, the tagged server will experience an arrival rate depending on its interference state when idle. However, its arrival rate when busy is independent of its interference state. Lemma 4.4 expresses the arrival rates experienced under the **JIQ$^{\mathbf{FP}}$-d** policy.

Lemma 4.4. *The state-dependent arrival rates under* **JIQ$^{\mathbf{FP}}$-d** *can be expressed in terms of the limiting probabilities, λ, and d as follows:*

$$\lambda_{0f} = d\lambda \sum_{i=0}^{d-1} \frac{\binom{d-1}{i}}{i+1} (\pi_{0f})^i (1-\pi_{0f})^{d-1-i},$$
$$\lambda_{0s} = d\lambda \sum_{i=0}^{d-1} \frac{\binom{d-1}{i}}{i+1} (\pi_{0s})^i (1-\pi_0)^{d-1-i},$$
$$\lambda_{1f} = \lambda_{1s} = \lambda(1-\pi_0)^{d-1}.$$

Arrival rates: JIQ$^{\mathbf{BR}}$-d. Under this policy, the tagged server will experience an arrival rate depending on its interference state when idle or busy. When at least one of the queried servers is idle, the job will be sent

to a randomly chosen idle fast server, if at least one such server was queried, and to a randomly chosen idle server, otherwise. If all of the servers are busy and both fast and slow busy servers were queried, the job will be sent to a randomly chosen busy fast (respectively, busy slow) server with probability $\frac{\mu_f\alpha_s}{\mu_f\alpha_s+\mu_s\alpha_f}$ (respectively, $\frac{\mu_s\alpha_f}{\mu_f\alpha_s+\mu_s\alpha_f}$). Lemma 4.5 expresses the arrival rates experienced under the **JIQ$^{\mathbf{BR}}$-d** policy.

Lemma 4.5. *The state-dependent arrival rates under* **JIQ$^{\mathbf{BR}}$-d** *can be expressed in terms of the limiting probabilities, λ, and d as follows:*

$$\lambda_{0f} = d\lambda \sum_{i=0}^{d-1} \frac{\binom{d-1}{i}}{i+1} (\pi_{0f})^i (1-\pi_{0f})^{d-i-1},$$
$$\lambda_{0s} = d\lambda \sum_{i=0}^{d-1} \frac{\binom{d-1}{i}}{i+1} (\pi_{0s})^i (1-\pi_0)^{d-i-1},$$
$$\lambda_{1f} = d\lambda \left(\frac{(\pi_f-\pi_{0f})^{d-1}}{d} + \frac{\mu_f\alpha_s}{\mu_f\alpha_s+\mu_s\alpha_f} \sum_{i=0}^{d-2} \frac{\binom{d-1}{i}}{i+1} (\pi_f-\pi_{0f})^i (\pi_s-\pi_{0s})^{d-i-1} \right),$$
$$\lambda_{1s} = d\lambda \left(\frac{(\pi_s-\pi_{0s})^{d-1}}{d} + \frac{\mu_s\alpha_f}{\mu_f\alpha_s+\mu_s\alpha_f} \sum_{i=0}^{d-2} \frac{\binom{d-1}{i}}{i+1} (\pi_s-\pi_{0s})^i (\pi_f-\pi_{0f})^{d-i-1} \right).$$

Arrival rates: JFQ-d. This policy uses only speed information. We see that the tagged server experiences an arrival rate of $\lambda_{0f} = \lambda_{1f}$ when the server's speed is fast and $\lambda_{0s} = \lambda_{1s}$ when the server's speed is slow; these rates are given by Lemma 4.6. The resulting state transition diagram for the tagged server under **JFQ-d** is the same as that under **JIFQ-d** (see Fig. 2), where $\lambda_{0f} = \lambda_{1f}$ and $\lambda_{0s} = \lambda_{1s}$.

Lemma 4.6. *The state-dependent arrival rates under* **JFQ-d** *can be expressed in terms of the limiting probabilities, λ, and d as follows:*

$$\lambda_{0f} = \lambda_{1f} = d\lambda \sum_{i=0}^{d-1} \frac{\binom{d-1}{i}}{i+1} (\pi_f)^i (\pi_s)^{d-1-i},$$
$$\lambda_{0s} = \lambda_{1s} = \lambda (\pi_s)^{d-1}.$$

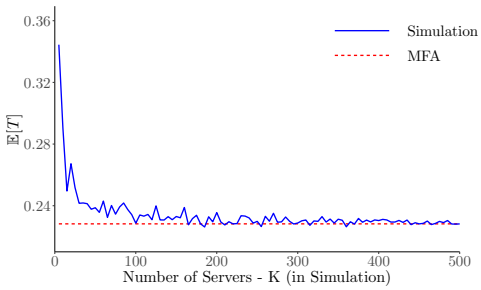
3.2 Finding the Expected Response Time

Below, we list the steps used to numerically determine the expected response time under a dispatching policy given system parameters $\lambda, \mu_f, \mu_s, \alpha_f$, and α_s:

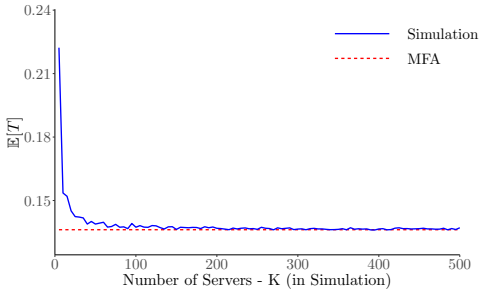
1. We simultaneously solve the set of nonlinear equations given in Lemmas 1-4 (the corresponding lemma for the policy of interest) to obtain a unique feasible solution for arrival rates $\lambda_{0x}, \lambda_{1x} > 0, x \in \{s, f\}$, limiting probabilities π_{0f} and $\pi_{0s} \in [0,1]$, and properties $P_{ff}, P_{fs}, P_{sf}, P_{ss} \in [0,1]$ and $T_f, T_s \geq 0$. We discuss the uniqueness of the solution in Appendix B.

2. We use the above solution as the input to the linear system of equations in Proposition 1 to find $\mathcal{R}$ and $\mathcal{T}$.

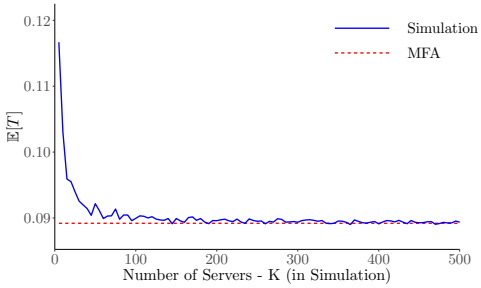
3. We use Eq. (2) to find the expected response time $\mathbb{E}[T]$.



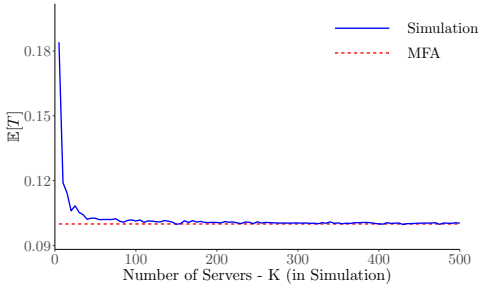
(a) **JIQ-** d Policy



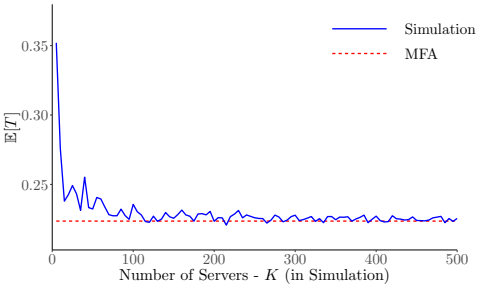
(b) **JFQ-** d Policy



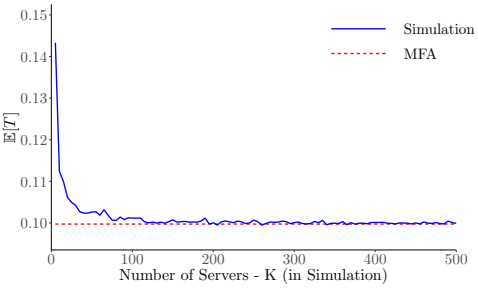
(c) **JIFQ-** d Policy



(d) **JFIQ-** d Policy



(e) **JIQ**$^{\mathbf{FP}}$**-** d Policy



(f) **JIQ**$^{\mathbf{BR}}$**-** d Policy

Figure 3: Simulated expected response times converge to the **MFA** results when the number of servers K is large; parameters: $\lambda = 13 \cdot K, \mu_f = 25, \mu_s = 10, \alpha_f = 0.05, \alpha_s = 0.1, d = 2$; number of jobs run $= 100,000 \cdot K$, warm-up $= 500 \cdot K$ jobs

4 Analytic & Numerical Results

In this section, we first discuss the validity of the asymptotic independence assumption for the dispatching policies described in §2. We do not formally prove this assumption; rather, like several other related papers (e.g., [11,12,36]), we test the assumption's validity by simulating the systems of various sizes (i.e., server counts, K). For example, the plots in Fig. 3 show that once K is sufficiently large, the simulated expected response times match our analytical derivations in §2 based on the asymptotic independence assumption for all policies.

In the remainder of this section, we compare the performance of the dispatching policies analytically and numerically. In §4.1, we present our analytic results: first, we specify the system stability condition, then we prove several results about how some policies outperform others with respect to the expected response time.

In §4.2, we numerically compare the performance of the policies under different parameter settings.

4.1 Stability and Analytical Comparisons

Given the average service rate $\frac{\alpha_s\mu_f + \alpha_f\mu_s}{\alpha_f + \alpha_s}$, we define the system load as in Eq. (7), based on which we establish the condition for system stability for all policies of interest in Proposition 2:

$$\rho = \frac{\lambda(\alpha_f + \alpha_s)}{\alpha_s\mu_f + \alpha_f\mu_s}. \tag{7}$$

Proposition 2. *If the system load, as defined in Eq.* (7)*, is below one (i.e.,* $\rho < 1$*) and asymptotic independence holds, the system is stable (i.e.,* $\mathbb{E}[T] < \infty$*) under all policies of interest, including the random policy. Otherwise, when* $\rho > 1$*, the system is unstable (i.e.,* $\mathbb{E}[T] = \infty$*) under any dispatching policy.*

This result is significant because in models that consider heterogeneous servers without addressing interference (i.e., the servers run at fixed speeds), implementing the policies that favor fast servers (i.e., **JFQ**-d, **JIFQ**-d, and **JFIQ**-d) can result in an unstable system when the dispatching policy queries servers without taking their speeds into consideration. Moreover, even under speed-aware querying in such models, the stability of policies that favor fast servers depends on precisely how this querying is carried out [10]. This hindrance does not exist in our setting as *all servers are stochastically identical*, i.e., all servers are treated the same in the long run because all fast servers will eventually become slow and vice-versa.

In the next proposition, we prove that some of our policies outperform others with respect to mean response time.

Proposition 3. *For any* d*, we have the following weak dominance relationships between our policies:*

(a) **JFIQ**-d *(weakly) outperforms* **JFQ**-d*; i.e.,* $\mathbb{E}[T]^{\textbf{JFIQ-}d} \leq \mathbb{E}[T]^{\textbf{JFQ-}d}$.

(b) **JIQ**-d *(weakly) outperforms* **RND***; i.e.,* $\mathbb{E}[T]^{\textbf{JIQ-}d} \leq \mathbb{E}[T]^{\textbf{RND}}$.

(c) $\textbf{JIQ}^{\textbf{FP}}$-$d$ *(weakly) outperforms* **JIQ**-d*; i.e.,* $\mathbb{E}[T]^{\textbf{JIQ}^{\textbf{FP}}\textbf{-}d} \leq \mathbb{E}[T]^{\textbf{JIQ-}d}$.

We proceed to study the relative performance of those policies that were not dominated in Proposition 3.

4.2 Numerical Experiments

In Proposition 3, we proved that the $\textbf{JIQ}^{\textbf{FP}}$-$d$ and **JFIQ**-d policies outperform **JIQ**-d and **JFQ**-d, respectively. Accordingly, we exclude **JIQ**-d and **JFQ**-d from our analysis in this section, and we run numerical experiments on **JIFQ**-d, **JFIQ**-d, $\textbf{JIQ}^{\textbf{BR}}$-$d$, and $\textbf{JIQ}^{\textbf{FP}}$-$d$ to understand under what parameter settings each policy outperforms the others. Though the **RND** policy is dominated by the **JIQ**-d policy (as we proved in Proposition 3), we use the **RND** policy as an easy-to-implement benchmark throughout this section. We also investigate the consequences of not implementing the best policy and the associated regret.

In our experiments, we set the fast service rate to $\mu_f = 100$ and the number of queried servers to $d = 2$. The results for larger query sizes ($d = 3$ and $d = 4$) are of a similar nature; we discuss these results at the end

Table 1: Optimality proportion and improvement over **RND** when $d = 2$.

Best-performing policy	Optimality proportion	Mean (10^{th}–90^{th} percentile) improvement over **RND**
JFIQ-d	413 (39.3%)	76.3% (39.7%–98.7%)
JIFQ-d	368 (35.0%)	45.0% (11.0%–90.2%)
$\mathbf{JIQ^{BR}}$-d	176 (16.8%)	57.0% (33.6%–95.9%)
$\mathbf{JIQ^{FP}}$-d	93 (8.9%)	45.0% (33.5%–49.9%)

of this subsection. In order to conduct a comprehensive study across all practical parameter settings, we vary the other parameters in the same ranges used in [34], resulting in a total of 1050 experiments:

- $\rho \in \{0.1, 0.25, 0.5, 0.75, 0.9, 0.95\}$,
- $\mu_s/\mu_f \in \{0.01, 0.02, 0.1, 0.2, 0.5, 0.8, 0.95\}$,
- $\alpha_s \in \{0.02, 0.2, 1, 10, 50\}$,
- $\alpha_f/\alpha_s \in \{0.05, 0.1, 0.2, 0.5, 1\}$.

For each experiment, we record the expected response time under each of the **JFIQ**-d, **JFIQ**-d, $\mathbf{JIQ^{BR}}$-d, $\mathbf{JIQ^{FP}}$-d, and **RND** policies. Table 1 provides statistics for the optimality proportion of each policy and the improvement over the **RND** policy when the corresponding policy outperforms the other policies. Based on Table 1, policies that use both the idleness and speed information perform better in most experiments with 39% and 35% optimality proportion for **JFIQ**-d and **JFIQ**-d, respectively. All policies listed in Table 1 result in significant improvement over the easy-to-implement **RND** policy, with an average improvement ranging from 45%–76.3%.

The color-coded Table 2 specifies the best-performing policy (i.e., the one with the minimum expected response time among the policies evaluated) for each experiment. The value recorded in each cell of Table 2 reports the percentage improvement of expected response time under the best-performing policy against the benchmark policy **RND**:

$$\frac{\mathbb{E}[T]^{\mathbf{RND}} - \min\left\{\mathbb{E}[T]^{\mathbf{JFIQ}\text{-}d},\ \mathbb{E}[T]^{\mathbf{JIFQ}\text{-}d},\ \mathbb{E}[T]^{\mathbf{JIQ^{FP}}\text{-}d},\ \mathbb{E}[T]^{\mathbf{JIQ^{BR}}\text{-}d}\right\}}{\mathbb{E}[T]^{\mathbf{RND}}} \times 100$$

Below, we draw insights based on the results presented in Table 2.

***The impact of parameters on the best-performing policy*:** Figs. 4 and 5 graphically summarize the information in Table 2. The plots of Fig. 4 show the marginal impact of parameters (varying a parameter while fixing other parameters) on the optimality proportion of each policy. Based on Fig. 4a, the **JFIQ**-d policy, which prioritizes speed information over idleness information among queried servers, outperforms the other policies when the impact of service degradation due to interference is more severe (a smaller μ_s/μ_f ratio). As the fast and slow service rates become more comparable, the policies prioritizing idle servers over fast servers

		α_s		0.02					0.2					1					10					50				
		α_f/α_s		0.05	0.1	0.2	0.5	1	0.05	0.1	0.2	0.5	1	0.05	0.1	0.2	0.5	1	0.05	0.1	0.2	0.5	1	0.05	0.1	0.2	0.5	1
ρ	0.1	μ_s/μ_f	0.01	99.5	99.3	94.7	79.7	64.1	95.0	95.5	91.8	78.1	62.9	80.3	85.1	84.0	72.9	58.5	34.7	44.9	51.7	49.8	40.6	15.4	19.4	23.5	24.9	21.4
			0.02	99.4	99.6	99.5	93.9	82.2	94.8	96.4	95.8	88.2	76.2	79.3	85.3	86.1	78.0	65.2	33.7	43.9	51.0	49.8	40.9	15.3	19.2	23.2	24.6	21.3
			0.1	86.6	88.9	88.5	81.8	69.7	73.0	80.8	83.7	79.2	67.9	55.5	67.0	73.5	71.7	61.8	26.5	35.5	43.3	45.4	39.0	14.1	17.4	20.9	22.7	20.4
			0.2	32.0	42.7	51.7	55.1	49.3	31.4	42.0	51.1	54.5	48.8	29.4	39.5	48.4	52.1	46.7	20.2	26.8	33.7	37.4	33.3	12.9	15.5	18.5	20.6	19.0
			0.5	13.0	15.9	19.7	23.7	23.7	13.0	15.9	19.6	23.7	23.6	12.9	15.7	19.4	23.3	23.2	12.2	14.4	17.4	20.4	20.0	10.8	12.1	13.7	15.2	14.7
			0.8	10.0	10.7	11.7	13.0	13.3	10.0	10.7	11.7	13.0	13.3	10.0	10.7	11.6	13.0	13.2	9.9	10.5	11.3	12.4	12.5	9.6	10.0	10.6	11.2	11.1
			0.95	9.3	9.4	9.6	9.9	10.0	9.3	9.4	9.6	9.9	10.0	9.3	9.4	9.6	9.9	10.0	9.3	9.4	9.6	9.8	9.9	9.2	9.3	9.4	9.6	9.6
	0.25	μ_s/μ_f	0.01	99.0	96.0	90.1	76.0	60.4	95.3	94.0	89.0	75.4	60.0	83.2	86.6	84.5	72.9	58.3	42.4	51.2	57.0	54.7	45.4	24.9	27.8	30.7	30.9	28.0
			0.02	99.6	99.1	93.7	79.9	64.7	95.9	96.3	92.2	79.1	64.2	83.3	87.7	86.6	75.8	61.7	41.9	50.8	56.9	55.1	46.2	24.8	27.7	30.5	30.7	28.0
			0.1	99.3	99.6	99.7	99.7	99.3	93.9	96.3	97.4	96.9	93.8	77.0	84.9	88.8	87.7	80.4	37.4	45.9	53.0	54.3	47.7	23.9	26.3	28.7	29.9	28.0
			0.2	97.5	98.4	98.3	91.1	77.5	82.9	88.3	90.4	86.1	75.1	59.8	70.1	76.4	76.3	68.1	31.9	38.6	45.1	47.9	43.3	22.9	24.9	27.2	28.7	27.3
			0.5	24.8	28.2	32.2	36.1	35.6	24.8	28.1	32.1	35.9	35.4	24.5	27.6	31.5	35.1	34.6	22.9	25.1	27.7	30.3	29.7	21.3	22.3	23.5	24.7	24.2
			0.8	20.7	21.3	22.2	23.3	23.5	20.7	21.3	22.1	23.3	23.5	20.7	21.3	22.1	23.2	23.4	20.6	21.1	21.8	22.6	22.6	20.4	20.7	21.1	21.5	21.4
			0.95	20.1	20.2	20.4	20.6	20.6	20.1	20.2	20.4	20.6	20.6	20.1	20.2	20.4	20.6	20.6	20.1	20.2	20.3	20.5	20.5	20.1	20.1	20.2	20.3	20.3
	0.50	μ_s/μ_f	0.01	98.3	96.1	91.7	80.1	65.5	95.6	94.6	90.8	79.6	65.1	86.0	88.8	87.2	77.5	63.6	51.7	59.0	63.8	61.3	51.7	36.7	38.7	40.6	40.3	37.1
			0.02	99.3	97.2	92.9	81.6	67.2	96.4	95.6	92.0	81.0	66.8	86.2	89.4	88.1	78.7	65.1	51.5	58.8	63.8	61.6	52.3	36.6	38.6	40.5	40.4	37.2
			0.1	99.6	99.8	99.8	95.7	85.2	96.1	97.7	98.3	94.0	83.8	84.2	89.9	92.4	88.7	78.8	48.9	56.2	62.1	62.5	55.3	36.2	38.0	39.9	40.3	38.0
			0.2	99.4	99.6	99.8	99.8	99.6	94.2	96.6	97.8	98.0	96.7	78.7	86.0	90.2	91.2	87.1	45.1	51.4	57.3	59.6	54.5	35.7	37.3	39.1	39.8	38.2
			0.5	70.4	73.3	73.0	68.6	62.2	56.9	63.8	67.7	66.4	61.0	46.5	52.9	58.4	60.6	57.0	37.1	39.7	42.7	45.2	44.0	34.4	35.2	36.2	37.0	36.5
			0.8	34.3	35.1	36.1	37.4	37.6	34.3	35.0	36.0	37.3	37.5	34.2	34.9	35.9	37.0	37.1	33.9	34.3	34.9	35.6	35.5	33.6	33.8	34.0	34.3	34.2
			0.95	33.4	33.5	33.5	33.7	33.7	33.4	33.5	33.5	33.7	33.7	33.4	33.5	33.5	33.7	33.7	33.4	33.4	33.5	33.6	33.6	33.4	33.4	33.4	33.5	33.5
	0.75	μ_s/μ_f	0.01	98.8	97.6	94.9	86.1	72.7	96.6	96.3	94.1	85.6	72.3	88.3	91.2	90.8	83.5	70.7	58.5	64.9	69.4	67.5	57.9	45.7	47.5	49.1	48.3	44.4
			0.02	99.2	98.0	95.4	86.9	73.8	96.8	96.7	94.6	86.4	73.4	88.3	91.4	91.2	84.2	71.6	58.3	64.7	69.3	67.6	58.3	45.7	47.4	49.0	48.4	44.5
			0.1	99.7	99.8	99.5	93.1	82.9	96.9	98.2	98.2	92.4	82.2	87.1	91.8	93.6	89.2	79.4	56.5	62.8	67.9	68.1	60.7	45.4	47.0	48.5	48.5	45.6
			0.2	99.6	99.8	99.8	99.8	95.5	95.9	97.6	98.5	98.4	93.8	84.1	89.8	92.9	93.4	88.1	54.2	60.0	65.4	67.2	62.2	44.9	46.3	47.8	48.2	46.2
			0.5	98.5	99.1	99.4	99.5	99.2	87.7	92.2	94.7	95.5	93.7	67.4	75.3	81.2	84.0	81.2	47.4	50.3	53.8	56.6	55.1	43.7	44.4	45.2	45.8	45.2
			0.8	48.4	51.3	54.1	55.9	52.8	46.8	49.3	51.9	54.3	51.6	44.9	46.3	48.3	50.7	48.7	43.1	43.4	44.0	45.2	44.3	42.9	42.9	43.0	43.4	43.2
			0.95	43.0	43.1	43.2	43.3	43.5	43.0	43.0	43.2	43.3	43.4	42.9	43.0	43.1	43.2	43.3	42.9	42.9	42.9	43.0	43.0	42.9	42.9	42.9	42.9	42.9
	0.90	μ_s/μ_f	0.01	99.3	98.7	97.0	90.0	77.5	97.2	97.5	96.2	89.5	77.1	89.5	92.7	93.1	87.4	75.3	61.8	67.9	72.5	71.3	62.2	50.0	51.5	53.0	52.2	48.0
			0.02	99.5	98.9	97.3	90.6	78.4	97.3	97.7	96.5	90.1	78.0	89.5	92.7	93.3	87.9	76.1	61.6	67.7	72.3	71.4	62.5	49.9	51.5	53.0	52.2	48.2
			0.1	99.7	99.8	99.3	94.8	85.6	97.2	98.4	98.3	94.1	84.9	88.3	92.6	94.3	91.3	82.3	60.1	66.1	71.1	71.7	64.9	49.6	51.0	52.5	52.3	49.2
			0.2	99.6	99.8	99.9	99.2	94.0	96.5	98.0	98.7	98.1	92.9	86.0	91.1	94.0	94.1	88.8	58.1	63.6	68.8	71.0	66.4	49.2	50.4	51.7	52.0	49.9
			0.5	98.9	99.4	99.6	99.7	99.6	90.9	94.4	96.4	97.1	96.7	73.1	80.4	85.8	88.2	87.2	51.6	54.5	58.1	60.3	59.3	47.9	48.4	49.2	49.6	48.8
			0.8	90.1	93.4	94.9	95.4	90.6	62.6	68.2	72.6	77.1	71.7	49.8	51.3	53.4	60.2	57.6	47.5	47.5	47.6	49.3	48.7	47.4	47.3	47.3	47.6	47.5
			0.95	48.2	48.7	49.4	49.9	50.3	47.8	48.1	48.5	48.8	49.1	47.5	47.6	47.7	47.8	48.1	47.4	47.4	47.4	47.4	47.5	47.4	47.4	47.4	47.4	47.4
	0.95	μ_s/μ_f	0.01	99.5	99.1	97.7	91.4	79.1	97.4	97.9	97.0	90.9	78.7	89.9	93.2	93.8	88.7	76.9	62.8	68.9	73.5	72.7	63.7	51.2	52.7	54.2	53.4	49.1
			0.02	99.6	99.2	98.0	91.9	80.0	97.5	98.0	97.2	91.4	79.6	89.8	93.2	94.0	89.1	77.7	62.6	68.7	73.3	72.7	64.0	51.1	52.7	54.1	53.4	49.3
			0.1	99.7	99.8	99.5	95.6	86.8	97.3	98.4	98.5	95.0	86.1	88.7	92.8	94.6	92.1	83.5	61.1	67.0	72.0	72.9	66.3	50.8	52.2	53.6	53.4	50.3
			0.2	99.6	99.8	99.9	99.2	94.4	96.6	98.1	98.8	98.2	93.4	86.5	91.4	94.2	94.4	89.5	59.1	64.5	69.6	72.1	67.7	50.4	51.5	52.8	53.0	51.0
			0.5	99.0	99.4	99.6	99.7	99.4	91.3	94.5	96.4	97.4	96.0	73.7	80.7	86.0	89.3	86.9	52.5	55.3	59.0	61.8	59.9	49.1	49.5	50.2	50.8	49.6
			0.8	92.1	94.3	95.0	96.7	95.3	63.9	67.5	69.0	79.5	77.0	50.1	50.8	51.6	60.8	60.3	48.7	48.7	48.7	50.2	50.1	48.7	48.7	48.6	48.8	48.9
			0.95	53.1	55.4	57.5	57.7	57.8	49.4	49.9	50.5	50.8	51.8	48.8	48.9	49.0	49.1	49.6	48.7	48.7	48.7	48.7	48.8	48.7	48.7	48.7	48.7	48.7

JFIQ-d **JIFQ-d** **JIQ$^{\text{BR}}$-d** **JIQ$^{\text{FP}}$-d**

Table 2: The best policy and its percentage improvement compared to **RND** . $d = 2$

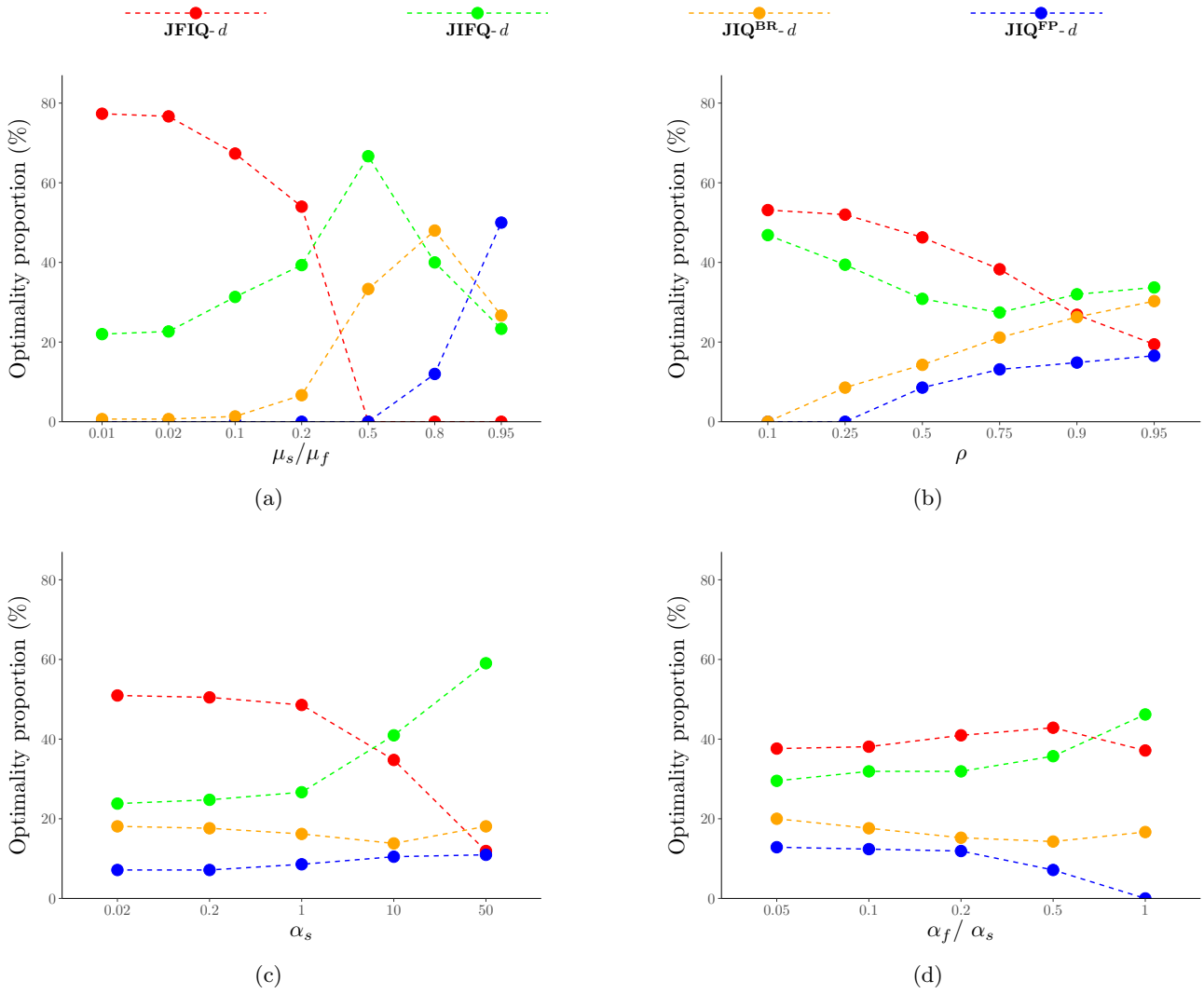


Figure 4: Marginal effect of system parameters on the best policy

(i.e., **JIFQ**-d, **JIQ$^{\mathbf{FP}}$**-d, and **JIQ$^{\mathbf{BR}}$**-d) perform better. Almost the same effect explained for μ_s/μ_f holds for the system utilization, based on Fig. 4b. In under-utilized systems (smaller ρ), prioritizing speed over idleness information results in shorter expected response times (better performance) since most queried servers will be idle when the utilization is low. However, as the system utilization increases and the chance of having idle servers among queried servers decreases, it is better to dispatch jobs to idle servers than fast servers. Fig. 4c shows that as the system spends more time under interference before returning to the normal condition (smaller α_s), prioritizing speed information over idleness is more valuable. On the other hand, when interference periods are shorter (i.e., as α_s increases), the value in speed information begins to decrease in favor of idleness information. We observe little takeaways in the marginal impact of the ratio α_f/α_s (Fig. 4d).

Fig. 5 provides the box plots for the marginal effect of parameters on the percentage improvement of the best dispatching policy over the **RND** policy. As the red lines in the plots show, the mean improvement over the **RND** policy is well above 25% across our parameter space. According to Fig. 5, the improvement over

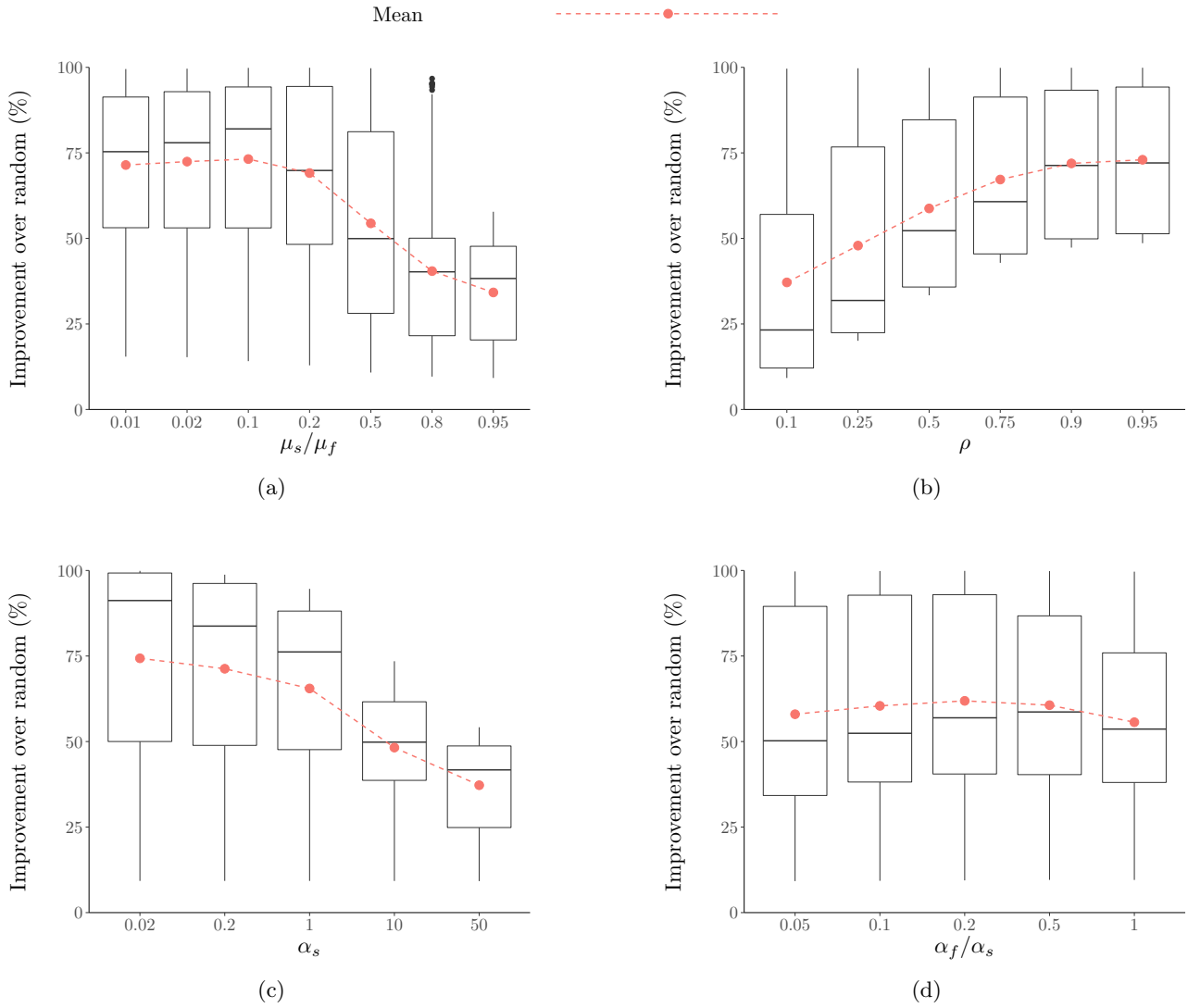


Figure 5: Marginal effect of system parameters in the improvement of the best policy over the random policy

the **RND** policy increases as the difference between the fast and slow speeds increases (Fig. 5a), the system load increases (Fig. 5b), and the system spends longer sojourns under interference before returning to normal (Fig. 5c).

Table 2 also shows how the interaction of the parameters affects the best-performing dispatching policy. We observe:

- As the system load ρ increases, **JFIQ**-d (specified by red cells) is generally more preferable in systems with lower α_s (spending more time under interference before returning to the normal condition) and lower μ_s/μ_f ratio (more severe degradation of the service rate due to interference).
- **JIFQ**-d (specified by green cells) becomes more preferable as the system load ρ and the rate of speed changes become moderate (while the speed disparity continues to be great).

Table 3: % increase in the expected response time due to implementing a non–optimal policy; the cell values report the mean ($10^{\text{th}} - 90^{\text{th}}$ percentile).

Implemented policy	Best–performing policy			
	JFIQ-d	**JIFQ**-d	**JIQ$^{\mathbf{BR}}$**-d	**JIQ$^{\mathbf{FP}}$**-d
JFIQ-d	–	15.7 (0.3 – 21.2)	1929.8 (4.6 – 2795.3)	2947.6 (10.9 – 2271.5)
JIFQ-d	5.0 (0.1 – 13.9)	–	348.0 (0 – 78.0)	306.7 (0.3 – 93.5)
JIQ$^{\mathbf{BR}}$-d	22.2 (1.1 – 33.9)	14.6 (0 – 6.8)	–	16.1 (0.2 – 33.8)
JIQ$^{\mathbf{FP}}$-d	1049.2 (6.0 – 2272.5)	733.8 (0 – 322.9)	759.7 (0 – 785.7)	–
RND	3459.4 (65.8 – 7545.3)	1709.1 (12.3 – 923.8)	1848.0 (50.5 – 2357.2)	84.0 (50.3 – 99.7)

- **JIQ$^{\mathbf{BR}}$**-d (specified by yellow cells) is preferred when the system load ρ is high with a moderate impact of interference on service speeds.
- **JIQ$^{\mathbf{FP}}$**-d (specified by blue cells) is preferred under high loads when the impact of interference on service rates is small.

***The consequence of implementing a non-optimal policy*:** Table 3 provides the regret associated with implementing a non-optimal policy. In each cell, we report the mean percentage increase (along with the 10^{th} and 90^{th} percentiles of the percentage increase) in the expected response time due to implementing a non-optimal policy instead of the best-performing policy. For example, if we employ the **JFIQ**-d policy rather than **JFIQ**-d in experiments in which **JFIQ**-d is optimal, the mean percentage increase in the expected response time will be 15.7%. As expected, implementing the naive **RND** policy increases the expected response time severely when any of the other policies are optimal. The table results show that no one policy is a near-optimal heuristic across all parameter settings. Table 3 shows the least regret of implementing the **JIQ$^{\mathbf{BR}}$**-d policy in all parameter settings. Also, using the **JIFQ**-d policy instead of **JFIQ**-d (and vice versa) causes a relatively mild increase in the mean response time. Motivated by these observations, in §5, we develop a method for optimally choosing a policy across a feasible space of policies that includes **JIQ$^{\mathbf{BR}}$**-d, **JIFQ**-d, and **JFIQ**-d.

***The impact of the query size*:** We also run our numerical experiments for query sizes $d = 3$ and $d = 4$. Appendix C presents the results. The general structure and takeaways are similar to those we discussed for $d = 2$. As the query size increases, we observe that the optimality proportion of **JFIQ**-d decreases while the optimality proportion of **JIFQ**-d increases; the optimality proportions of **JIQ$^{\mathbf{FP}}$**-d and **JIQ$^{\mathbf{BR}}$**-d remain almost unchanged. We can explain the change in the optimality proportions of **JFIQ**-d and **JIFQ**-d with d by noting that **JFIQ**-d prioritizes faster servers over idle servers. As the query size increases, there is a higher chance that there is a fast server among the queried servers. Therefore, arrivals are rarely dispatched to servers that are under interference. Thus, **JFIQ**-d overloads interference-free servers even when they are already busy and even when the query includes idle (but under-interference) servers. Whereas, under **JIFQ**-d and higher query sizes, the issue of overloading busy interference-free servers is better addressed by prioritizing dispatching jobs to queried idle

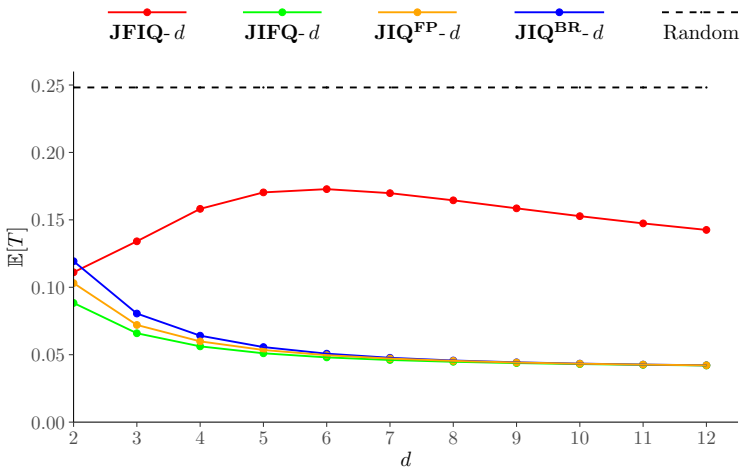


Figure 6: Expected response time as the query size changes; $\lambda = 15.75, \mu_f = 30, \mu_s = 15, \alpha_f = \alpha_s = 1$.

under-interference servers if the other queried interference-free servers are all busy.

We see this illustration in Fig. 6 for one problem set. When $d = 1$, all policies correspond to the **RND** policy whose response time is plotted in Fig. 6 as a dotted line for reference. We observe that the performance of **JFIQ**-d is not monotonic in d—contradicting conventional wisdom that querying more servers can only help. The performance under $d = 2$ is better because we are overloading the currently interference-free servers and ignoring—and hence, underutilizing—servers when they are under interference. Moreover, for **JFIQ**-d, we observe that after an initial increase in response time with d, further increases in d (starting from $d = 6$) yield improvements in the expected response time. Even though the servers are loaded more quickly when in the interference-free state, the probability of querying a fast idle server also increases with d. The performance of all of these policies (except seemingly **JFIQ**-d) tends to converge as d increases because the fraction of time the arriving job does not join a fast idle server under **JIFQ**-d, **JIQ$^{\mathbf{FP}}$**-d, and **JIQ$^{\mathbf{BR}}$**-d vanishes as $d \to \infty$. The same is true even for **JFIQ**-d, but it requires a larger d to converge as servers become overloaded when interference-free, making it difficult to query fast idle servers.

To conclude this section, we summarize some of our main analytical and numerical findings here: (1) For any given parameter setting, one of the **JIQ$^{\mathbf{FP}}$**-d, **JIQ$^{\mathbf{BR}}$**-d, **JIFQ**-d, or **JFIQ**-d policies is the best-performing among the seven policies, including **RND**, that we study. (2) Policies prioritizing idle servers over fast servers perform better as μ_s/μ_f, ρ, and α_s increase. (3) Though no single policy outperforms the others across all parameters, the regret of implementing **JIQ$^{\mathbf{BR}}$**-d, **JIFQ**-d, and **JFIQ**-d (when they are not the best-performing policy) are the lowest; this motivates us to study an optimization-driven policy in §5 that can outperform all three of these policies.

5 An Optimization-based Dispatching Policy

In this section, we take inspiration from [1] and [10] to develop an optimization-based policy, which we call $\mathbf{JIFQ^{OPT}}$. First, consider the following family of policies, which we denote by $\mathcal{P}$, each member of which is parameterized by $p, p' \in [0, 1]$, where the dispatcher again queries d servers and dispatches the job randomly in the following priority order:

1. If the query includes fast idle servers, a fast idle server.
2. Otherwise, if the query includes fast busy and slow idle servers, a fast busy (resp., a slow idle) server with probability p (resp., $1 - p$).
3. Otherwise, if the query consists of fast busy and fast slow servers, a fast busy (resp., slow busy) server with probability p' (resp., $1 - p'$).
4. Otherwise, if the query includes only slow servers, including at least one idle server, then (one of) the idle server(s).
5. Otherwise, if all queried servers are busy slow servers, then one of those servers.

Note that all policies in the family $\mathcal{P}$ are symmetric in that all servers that are "tied" (with respect to both their current interference state and their idle/busy status) are equally likely to be assigned the new arrival. The **JIFQ**-d, **JFIQ**-d, and $\mathbf{JIQ^{BR}}$-d policies are all members of $\mathcal{P}$:

- When $p = 0$ and $p' = 1$, it is identical to **JIFQ**-d.
- When $p = 1$ and $p' = 1$, it is identical to **JFIQ**-d.
- When $p = 0$ and $p' = \dfrac{\mu_f \alpha_s}{\mu_f \alpha_s + \mu_s \alpha_f}$, it is identical to $\mathbf{JIQ^{BR}}$-d.

Now define $\mathbf{JIFQ^{OPT}}$ as the policy chosen from $\mathcal{P}$ where given a problem instance (set of parameters), p and p' are chosen to minimize the mean response time $\mathbb{E}[T]$.[1] Clearly, $\mathbf{JIFQ^{OPT}}$ (weakly) outperforms **JIFQ**-d, **JFIQ**-d, and $\mathbf{JIQ^{BR}}$-d as it is the optimal policy across a feasible space of policies that includes all three of the aforementioned policies. Proposition 1 and Lemmas 1-3 still hold for $\mathbf{JIFQ^{OPT}}$. However, to explicitly find this policy (i.e., carry out the desired optimization), we still need to derive the state-dependent arrival rates for each policy in $\mathcal{P}$ in terms of the parameters $p, p' \in [0, 1]$.

Lemma 4.7. *The state-dependent arrival rates under the policies in the $\mathcal{P}$ family can be expressed in terms of the limiting probabilities (which are themselves dependent on p and p'), λ, d, and assignment*

[1] We will treat this policy as well-defined, even though we leave open the question of whether such an optimal policy is always unique for parameter settings where $\lambda > 0$.

probability parameters p and p' as follows:

$$\lambda_{0f} = d\lambda \sum_{i=0}^{d-1} \frac{\binom{d-1}{i}}{i+1} (\pi_{0f})^i (1-\pi_{0f})^{d-1-i},$$

$$\lambda_{0s} = d\lambda \Bigg[\sum_{i=0}^{d-2} \frac{\binom{d-1}{i}}{i+1} (\pi_{0s})^i (1-\pi_0)^{d-1-i} \Bigg(\left(\frac{\pi_s - \pi_{0s}}{1-\pi_0} \right)^{d-1-i} + \left(1 - \left(\frac{\pi_s - \pi_{0s}}{1-\pi_0} \right)^{d-1-i} \right) (1-p) \Bigg) + \frac{(\pi_{0s})^{d-1}}{d} \Bigg],$$

$$\lambda_{1f} = d\lambda \Bigg[\sum_{i=0}^{d-2} \frac{\binom{d-1}{i}}{i+1} (\pi_f - \pi_{0f})^i \left((\pi_s - \pi_{0s})^{d-1-i} p' + (\pi_s)^{d-1-i} \left(1 - \left(1 - \frac{\pi_{0s}}{\pi_s} \right)^{d-1-i} \right) p \right) + \frac{(\pi_f - \pi_{0f})^{d-1}}{d} \Bigg],$$

$$\lambda_{1s} = d\lambda \sum_{i=0}^{d-2} \frac{\binom{d-1}{i}}{i+1} (\pi_s - \pi_{0s})^i (\pi_f - \pi_{0f})^{d-1-i} (1-p') + \lambda (\pi_s - \pi_{0s})^{d-1}.$$

Under $\mathbf{JIFQ^{OPT}}$, we set p and p' to the values corresponding to the solution of a non-linear optimization problem where (i) the objective is to minimize the expected response time, (ii) the system parameters $\lambda, \mu_f, \mu_s, \alpha_f, \alpha_s$, and d are taken as given constants and $\lambda_{ij}, \pi_{0j}, T_j, P_{jf}, R_j$, and R ($i \in \{0,1\}, j \in \{f,s\}$) are treated as decision variables alongside p and p', and (iii) the queueing-theoretic relationships between these variables are captured through a set of constraints. Specifically, we have the following problem:

$$
\begin{aligned}
\min \quad & \mathbb{E}[T] = \frac{\mathcal{R}(\alpha_f + \lambda_{0f})\beta_{0s}}{\beta_{0f} + \beta_{0s} + \lambda_{0f}\beta_{0s}T_f + \lambda_{0s}\beta_{0f}T_s} \\
\text{s.t.} \quad & \mathcal{R}_f = \frac{1 + \lambda_{1f}\left(T_f + \mathcal{R}_f + P_{ff}\mathcal{R}_f + P_{fs}\mathcal{R}_s\right) + \alpha_f\mathcal{R}_s}{\lambda_{1f} + \mu_f + \alpha_f} \\
& \mathcal{R}_s = \frac{1 + \lambda_{1s}\left(T_s + \mathcal{R}_s + P_{sf}\mathcal{R}_f + P_{ss}\mathcal{R}_s\right) + \alpha_s\mathcal{R}_f}{\lambda_{1s} + \mu_s + \alpha_s} \\
& \mathcal{R}_{0s} = \frac{\lambda_{0s}(\mathcal{R}_s + P_{ss}\mathcal{R}_{0s})}{\lambda_{0s} + \alpha_s} \\
& \mathcal{R} = \frac{\lambda_{0f}(\mathcal{R}_f + P_{fs}\mathcal{R}_{0s}) + \alpha_f\mathcal{R}_{0s}}{\lambda_{0f} + \alpha_f}, \\
& T_x = \frac{\gamma_y - \beta_{1f} - \beta_{1s}}{\gamma_f\beta_{1s} + \gamma_s\left(\alpha_f + \mu_f - \lambda_{1f}P_{ff}\right)} \qquad x \in \{f,s\}, y \in \{f,s\} \setminus \{x\} \\
& P_{ff} = \frac{\lambda_{1f}\left(P_{ff}^2 + P_{fs}P_{sf}\right) + \alpha_f P_{sf} + \mu_f}{\lambda_{1f} + \mu_f + \alpha_f} \\
& P_{sf} = \frac{\lambda_{1s}\left(P_{sf}P_{ff} + P_{ss}P_{sf}\right) + \alpha_s P_{ff}}{\lambda_{1s} + \mu_s + \alpha_s} \\
& \lambda_{0f} = d\lambda\sum_{i=0}^{d-1}\frac{\binom{d-1}{i}}{i+1}\left(\pi_{0f}\right)^i\left(1-\pi_{0f}\right)^{d-1-i} \\
& \lambda_{1f} = d\lambda\left[\sum_{i=0}^{d-2}\frac{\binom{d-1}{i}}{i+1}\left(\pi_f - \pi_{0f}\right)^i\left(\left(\pi_s - \pi_{0s}\right)^{d-1-i}p' + \left(\pi_s\right)^{d-1-i}\left(1 - \left(1 - \frac{\pi_{0s}}{\pi_s}\right)^{d-1-i}\right)p\right)\right. \\
& \qquad \left. + \frac{\left(\pi_f - \pi_{0f}\right)^{d-1}}{d}\right] \\
& \lambda_{0s} = d\lambda\left[\sum_{i=0}^{d-2}\frac{\binom{d-1}{i}}{i+1}\left(\pi_{0s}\right)^i\left(1-\pi_0\right)^{d-1-i}\left(\left(\frac{\pi_s - \pi_{0s}}{1-\pi_0}\right)^{d-1-i}\right.\right. \\
& \qquad \left.\left. + \left(1 - \left(\frac{\pi_s - \pi_{0s}}{1-\pi_0}\right)^{d-1-i}\right)(1-p)\right) + \frac{\left(\pi_{0s}\right)^{d-1}}{d}\right], \\
& \lambda_{1s} = d\lambda\sum_{i=0}^{d-2}\frac{\binom{d-1}{i}}{i+1}\left(\pi_s - \pi_{0s}\right)^i\left(\pi_f - \pi_{0f}\right)^{d-1-i}\left(1-p'\right) + \lambda(\pi_s - \pi_{0s})^{d-1} \\
& \pi_{0x} = \frac{\beta_{0y}}{\beta_{0f} + \beta_{0s} + \lambda_{0f}\beta_{0s}T_f + \lambda_{0s}\beta_{0f}T_s} \qquad x \in \{f,s\}, y \in \{f,s\} \setminus \{x\} \\
& \mathcal{R}_f, \mathcal{R}_s, \mathcal{R}_{0s}, \mathcal{R}, T_f, T_s, P_{ff}, P_{sf}, \lambda_{0f}, \lambda_{0f}, \lambda_{0s}, \lambda_{1s} \geq 0 \\
& 0 < \pi_{0f}, \pi_{0s} \leq 1 \\
& \beta_{jx} = \alpha_x + \lambda_{jx}P_{xy} \text{ and } \gamma_x = \lambda_{1x} - \mu_x \qquad j \in \{0,1\}, x \in \{f,s\}, y \in \{f,s\} \setminus \{x\}
\end{aligned}
$$

We formulated the optimization problem in Julia using the JuMP package [7]. We used the Interior Point Optimizer optimization package (IPOPT) [25, 35] to solve the optimization problem heuristically and determine the best values of p and p'.[2] IPOPT did not return a solution for 7% of the parameter settings. For these parameter settings, we heuristically solved the optimization model by feeding the solution of the best of the seven (non-optimized) policies studied earlier in this paper (specified in Table 2 for each experiment) as an initial solution.[3]

[2]We note the p and p' values that we find using IPOPT may not correspond to the theoretically optimal global solution as it is not guaranteed that IPOPT will find such a solution. Nevertheless, we call the resulting policy $\mathbf{JIFQ^{OPT}}$.

[3]Using this approach (where we feed an initial solution) on the remaining parameter settings for which it was not necessary did not yield substantially different results: the greatest discrepancy in expected response times across all such parameter settings was 0.07%.

Table 4: Improvement achieved by implementing $\mathbf{JIFQ^{OPT}}$

		Best-performing non-optimized policy			
	Overall	**JFIQ**-d	**JFIQ**-d	$\mathbf{JIQ^{BR}}$-d	$\mathbf{JIQ^{FP}}$-d
Mean improvement	0.83%	0.00%	0.34%	3.58%	1.25%
Median improvement	0.00%	0.00%	0.00%	0.40%	0.25%
Maximum improvement	52.94%	0.41%	52.94%	46.07%	23.61%

Table 4 presents the summary statistics for the (heuristically found) $\mathbf{JIFQ^{OPT}}$ policy's improvement over the (parameter setting-specific) best of the seven (non-optimized) policies studied earlier in this paper across the full range of parameter settings. The overall mean improvement is at 0.83%, and $\mathbf{JIFQ^{OPT}}$ does not achieve noteworthy overall improvement over the best of the non-optimized policies across the majority of the parameter settings (median improvement of 0.00%). However, Table 4 shows that there exist instances when **JFIQ**-d, $\mathbf{JIQ^{BR}}$-d, or $\mathbf{JIQ^{FP}}$-d is the best of the non-optimized policies, where optimization allows for substantial improvements as high as 52.94%. While **JFIQ**-d can perform quite poorly on some parameter settings, optimization only allows for small (if any) benefits when **JFIQ**-d is the best of the seven non-optimized policies; the largest improvement offered by optimization over **JFIQ**-d—across those parameter settings where **JFIQ**-d outperformed the other six policies—was 0.41%.

Table 5 tabulates the details of the $\mathbf{JIFQ^{OPT}}$'s improvement relative to the best-performing non-optimized policy (presented in Table 2) for all experiments. We observe that $\mathbf{JIFQ^{OPT}}$ always performs at least as well as the best non-optimized policy. In Fig. 7, we plot the mean and maximum improvement obtained with $\mathbf{JIFQ^{OPT}}$ over the best non-optimized policy, varying the values across each parameter. We observe, according to Table 5 and the plots in Fig. 7, that the modest improvements achieved using the optimization-based policy $\mathbf{JIFQ^{OPT}}$ occur under the following ranges of the parameter settings:

- When the system load is high.
- When the speed difference between interference states is moderate ($\mu_s/\mu_f \geq 0.5$).
- When the rate of state change in the interference state is much lower than the job processing rates ($\alpha_f \leq \alpha_s \ll \mu_s \leq \mu_f$).

In Appendix C, we recreate Table 5 with query sizes $d = 3$ (Table 12) and $d = 4$ (Table 13). The parameter settings with significant $\mathbf{JIFQ^{OPT}}$ improvement remain as discussed for $d = 2$. However, since the probability of reaching a fast idle server for the best-performing non-optimized heuristic increases as d increases, the benefit of adopting $\mathbf{JIFQ^{OPT}}$ becomes less apparent. However, there exist instances where we see significantly more improvements with $\mathbf{JIFQ^{OPT}}$ at $d = 3$ than $d = 2$. However, we can expect that as d increases, the best-performing non-optimized heuristic would perform the same or very close to $\mathbf{JIFQ^{OPT}}$.

Finally, we briefly examine what these optimized policies look like and compare our policies to the conventional (and heterogeneity/interference-unaware) **JIQ** and **JSQ**-d policies. Recall that under **JIQ**—a policy that is outside of the "power-of-d" paradigm—the dispatcher is aware of all servers that are idle at

		α_s		0.02					0.2					1					10					50				
		α_f/α_s		0.05	0.1	0.2	0.5	1	0.05	0.1	0.2	0.5	1	0.05	0.1	0.2	0.5	1	0.05	0.1	0.2	0.5	1	0.05	0.1	0.2	0.5	1
ρ	0.1	μ_s/μ_f	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.02	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.95	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.25	μ_s/μ_f	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.02	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.95	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.5	μ_s/μ_f	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.02	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.8	0.0	0.0	0.0	0.1	0.1	0.0	0.0	0.0	0.1	0.1	0.0	0.0	0.0	0.1	0.1	0.0	0.0	0.0	0.1	0.1	0.0	0.0	0.0	0.0	0.0
			0.95	0.0	0.0	0.1	0.1	0.1	0.0	0.0	0.1	0.1	0.1	0.0	0.0	0.1	0.1	0.1	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0
	0.75	μ_s/μ_f	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.02	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.2	0.0	0.0	0.3	0.0	0.0	0.0	0.0	0.3	0.0	0.0	0.0	0.0	0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.8	0.9	1.5	1.7	0.1	3.1	0.9	1.5	1.6	0.1	3.1	0.9	1.4	1.6	0.1	2.7	0.6	1.1	1.0	0.0	1.2	0.2	0.4	0.5	0.0	0.3
			0.95	0.1	0.1	0.2	0.3	0.2	0.1	0.1	0.2	0.3	0.2	0.1	0.1	0.2	0.3	0.1	0.0	0.1	0.1	0.2	0.1	0.0	0.0	0.0	0.1	0.0
	0.9	μ_s/μ_f	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.02	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.5	1.7	2.4	1.9	8.0	8.6	1.6	2.3	1.7	7.3	6.9	1.4	1.9	1.2	5.3	3.6	0.7	0.8	0.3	1.5	0.7	0.4	0.4	0.2	0.2	0.5
			0.8	6.4	11.1	16.0	7.2	12.1	6.3	10.9	15.5	6.4	10.1	5.9	10.0	13.5	4.5	6.4	1.2	2.1	3.2	1.1	1.5	0.3	0.5	0.8	0.3	0.3
			0.95	0.4	0.7	1.1	1.7	0.6	0.4	0.6	1.0	1.5	0.5	0.2	0.4	0.7	1.0	0.4	0.1	0.1	0.2	0.3	0.1	0.0	0.0	0.1	0.1	0.0
	0.95	μ_s/μ_f	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.02	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.2	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0
			0.5	5.7	9.3	10.7	7.8	52.9	5.3	8.5	9.2	6.0	32.8	4.3	6.4	5.8	3.6	14.1	1.6	2.1	1.3	0.9	2.1	0.7	0.8	0.6	0.0	1.0
			0.8	16.6	29.7	46.1	32.0	37.3	16.0	28.2	41.7	24.1	21.4	10.2	16.6	23.6	12.7	9.5	1.5	2.5	4.0	2.2	1.7	0.3	0.5	0.8	0.6	0.3
			0.95	2.0	3.4	5.1	6.7	1.9	1.2	2.0	3.2	4.2	1.2	0.5	0.9	1.4	1.9	0.6	0.1	0.2	0.2	0.3	0.1	0.0	0.0	0.1	0.1	0.0

JFIQ-d **JIFQ-d** **JIQ$^{\mathbf{BR}}$-d** **JIQ$^{\mathbf{FP}}$-d**

Table 5: The percentage improvement with **JIFQ$^{\mathbf{OPT}}$** over the best non-optimized policy with $d = 2$

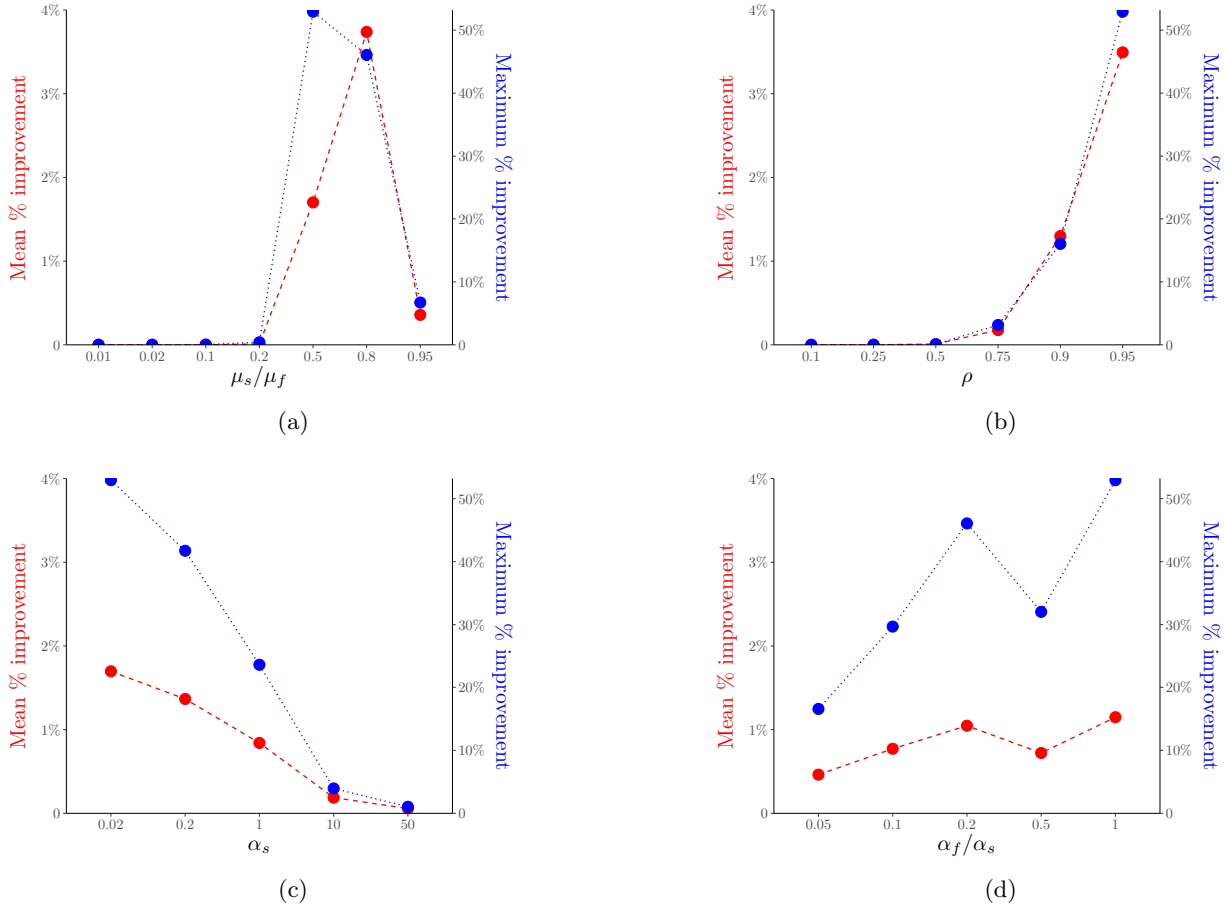


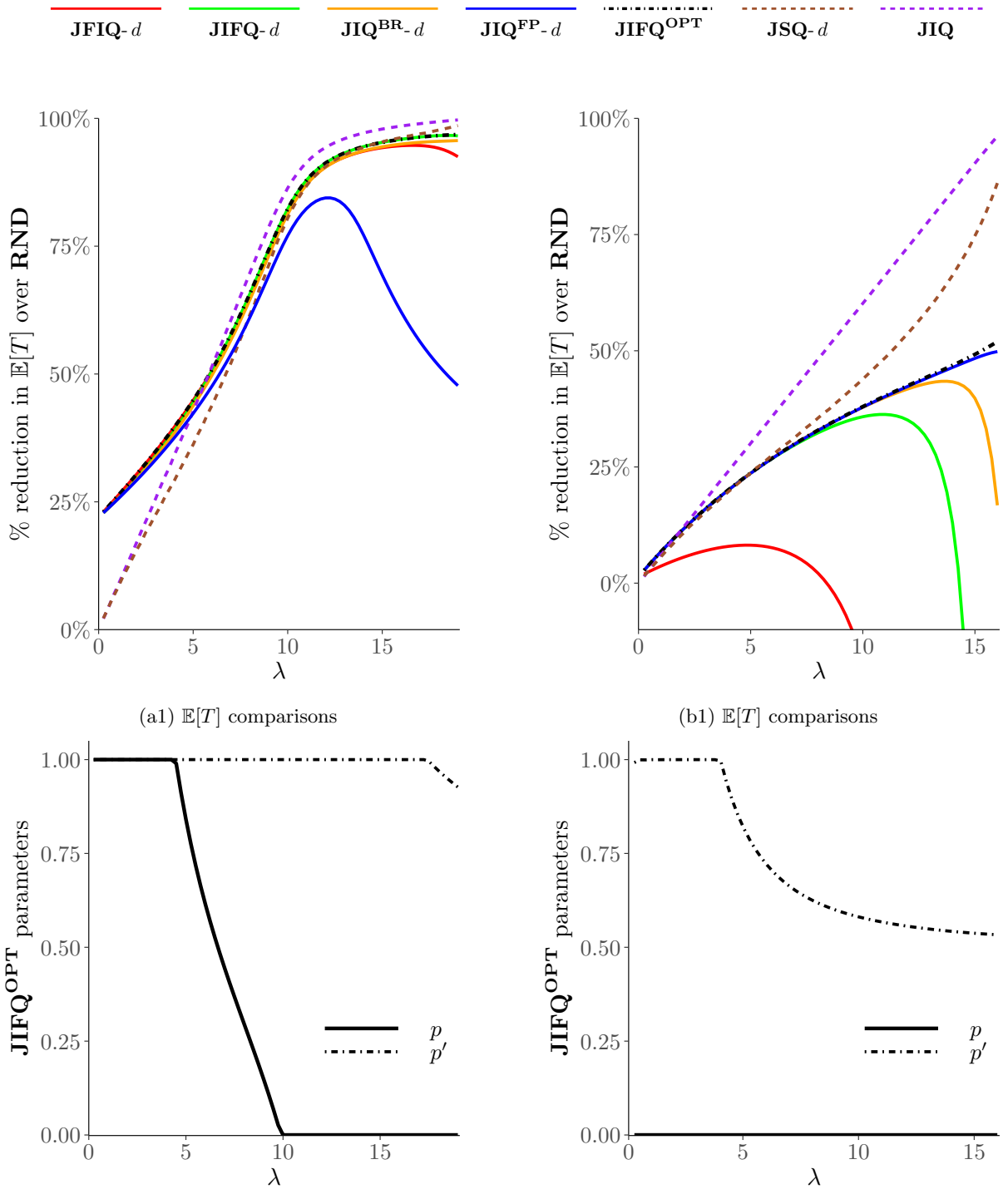
Figure 7: Improvement of $\mathbf{JIFQ^{OPT}}$ over the best non-optimized policy

any given time (if any) without needing to query them and always sends a job to a randomly chosen idle server (should any servers currently be idle); meanwhile, under **JSQ**-d, the dispatcher queries d servers at random and sends the job to whichever queried server has the shortest queue (breaking ties at random). Both policies make dispatching decisions without regard to the interference state information.

In Fig. 8, we plot the expected response times for the policies of interest introduced in this paper (the four non-dominated non-optimized policies and $\mathbf{JIFQ}^{\mathbf{OPT}}$) and **JIQ** and **JSQ**-d across two families of parameter settings where we allow λ to vary in each family. As discussed, we observe that $\mathbf{JIFQ}^{\mathbf{OPT}}$ always performs at least as well as the best non-optimized policy (introduced in this paper), if not better. The flexibility in setting the p and p' values allows for achieving the best non-optimized policy performance or better after optimizing the policy parameters.

Figs. 8a2 and 8b2 plot p and p' for $\mathbf{JIFQ}^{\mathbf{OPT}}$. In Fig. 8a2, we observe that under low to medium loads (particularly in the region where the heuristically optimized parameter values are $p = p' = 1$, which is precisely when $\mathbf{JIFQ}^{\mathbf{OPT}}$ simplifies to **JFIQ**-d), it is beneficial to route jobs to queried fast servers whenever available because of the significant speed differences between the two interference states. As load increases, it becomes advantageous to favor idleness over server speed (e.g., an idle server that is under-interference may become preferable to a busy server that is interference-free) even when under-interference servers are queried and there exist busy interference-free servers in the query; for example, at moderately high loads, we see that heuristic optimization yields the parameter values $p = 0$ and $p' = 1$, in which case $\mathbf{JIFQ}^{\mathbf{OPT}}$ simplifies to **JIFQ**-d. However, In Fig. 8b2, since there is little difference between server speeds, we find p=1 even at very low loads. Under this setting, when the queried servers are all busy, we would assign jobs to only interference-free servers, if possible, at very low loads. As load increases, we would increase the probability of routing to busy under-interference servers (p' decreases).

In the setting for Fig. 8a1 (where the speed difference between the slow and fast servers is significant), $\mathbf{JIFQ}^{\mathbf{OPT}}$ outperforms **JSQ**-d *except* at very high loads because **JSQ**-d does not take into account interference-state information. Even **JIQ**, where the dispatcher is completely aware of all idle servers in the system, is overshadowed by $\mathbf{JIFQ}^{\mathbf{OPT}}$ at low loads for the same reason. The **JIQ** policy will fail to make consistent use of fast idle servers before slow idle servers. Meanwhile, **JIQ** is dominant at high load because it is crucial to dispatch jobs to idle servers whenever possible under high load; **JIQ** maximizes the rate at which jobs are sent to idle servers. Moreover, faster servers will tend to become idle more often under **JIQ**. These observations are consistent with those observed in [10] in the context of systems with heterogeneous servers, each operating at a fixed speed. In Fig. 8b1, we see that **JIQ** and **JSQ**-d outperform $\mathbf{JIFQ}^{\mathbf{OPT}}$ except for a small region at very low loads, as interference has little effect on server speed; hence, there is no value in using interference-state information. Ultimately, $\mathbf{JIFQ}^{\mathbf{OPT}}$ is ideal for low loads in settings where interference causes significant performance degradation. Otherwise, rather than using the power-of-d paradigm, one can use **JIQ** to obtain excellent performance.



(a1) $\mathbb{E}[T]$ comparisons

(b1) $\mathbb{E}[T]$ comparisons

(a2) **JIFQ$^{\text{OPT}}$** optimized parameters

(b2) **JIFQ$^{\text{OPT}}$** optimized parameters

(a) Parameter setting: $\lambda = .25, .5, ...19$; $\mu_f = 25$; $\mu_s = 10$; $\alpha_f = .05$; $\alpha_s = .1$; $d = 2$

(b) Parameter setting: $\lambda = .25, .5, ...16$; $\mu_f = 17$; $\mu_s = 16$; $\alpha_f = .05$; $\alpha_s = .1$; $d = 2$

Figure 8: a1, b1: Percentage improvement compared to the random policy for different λ values for two parameter settings; a2, b2: The optimal **JIFQ$^{\text{OPT}}$** parameters

6 Conclusion

In this paper, we studied the problem of scalable dispatching in the presence of servers that can undergo interference (i.e., undergo speed state changes). We analyzed a set of easy-to-implement heuristic policies based on the power-of-d-choices paradigm, dispatching incoming jobs based on the speed and idleness status of a randomly sampled subset of servers. We presented a technique of exact measuring the performance of these policies analytically using mean field analysis and recursive renewal reward, observing and proving several intriguing results. We proved that power-of-d policies that use both interference and idleness information outperform those that use only one type of information; specifically, $\mathbf{JIQ^{FP}}$-d and **JFIQ**-d outperform **JIQ**-d and **JFQ**-d, respectively. Though the best policy varies depending on the system parameter settings, for each of the four non-dominated policies ($\mathbf{JIQ^{FP}}$-d, $\mathbf{JIQ^{BR}}$-d, **JIFQ**-d, and **JFIQ**-d), we identified (through a comprehensive numerical investigation) parameter settings where that policy outperforms the other three. Moreover, we found that naive use of speed information (as in **JFQ**-d) can result in poor performance and often underperforms pure random dispatching.

We introduced the new sophisticated—yet still simple—$\mathbf{JIFQ^{OPT}}$ policy that can overcome the above-mentioned deficiency. We found that this optimization-based policy is ideal for low loads in settings where interference causes significant performance degradation. Meanwhile, in other settings, the conventional **JIQ** policy outperforms the $\mathbf{JIFQ^{OPT}}$ policy. We note that one could develop a superior speed-aware version of the **JIQ** policy that keeps communications down for systems with heterogeneous servers operating at constant speeds. However, such a speed-aware version of **JIQ** may not allow for keeping communication costs down in the context of interference, as this would require servers to communicate all their speed changes (once detected) to the dispatcher. Future work could focus on the development of such a policy. Furthermore, Figs. 8a1 and 8b1 motivate further work that could formulate load balancing policies that take into account not only idleness information but also queue length information along with interference state information for the job assignment. We are optimistic that such heterogeneity-aware descendants of the **JSQ**-d policy would perform better than conventional policies such as **JIQ** and **JSQ**-d. The analysis of these new policies would greatly differ from that presented here; it would likely necessitate solving differential equations that govern the flow rates into different server states, similar to the concept presented in [10]. Nevertheless, we anticipate that performance analysis will once again allow for finding strong dispatching policies by heuristically optimizing across a broad family of parameterized dispatching policies.

References

[1] Abdul Jaleel, J., Doroudi, S., Gardner, K., Wickeham, A.: A general "power-of-d" dispatching framework for heterogeneous systems. Queueing Systems pp. 1–50 (2022)

[2] Abdul Jaleel, J., Wickeham, A., Doroudi, S., Gardner, K.: A general "power-of-d" dispatching framework for heterogeneous systems. ACM SIGMETRICS Performance Evaluation Review **48**(2), 30–32 (2020)

[3] Bramson, M., Lu, Y., Prabhakar, B.: Randomized load balancing with general service time distributions. ACM SIGMETRICS Performance Evaluation Review **38**(1), 275–286 (2010)

[4] Bramson, M., Lu, Y., Prabhakar, B.: Asymptotic independence of queues under randomized load balancing. Queueing Systems **71**(3), 247–292 (2012)

[5] Bu, Q., Liu, L., Zhao, Y.Q.: Mean field approximations to a queueing system with threshold-based workload control scheme. Communications in Statistics-Theory and Methods pp. 1–22 (2021)

[6] Bu, X., Rao, J., Xu, C.z.: Interference and locality-aware task scheduling for MapReduce applications in virtual clusters. In: Proceedings of the 22nd international symposium on High-performance parallel and distributed computing, pp. 227–238 (2013)

[7] Dunning, I., Huchette, J., Lubin, M.: JuMP: A modeling language for mathematical optimization. SIAM review **59**(2), 295–320 (2017)

[8] Ephremides, A., Varaiya, P., Walrand, J.: A simple dynamic routing problem. IEEE Transactions on Automatic Control **25**(4), 690–693 (1980)

[9] Gandhi, A., Doroudi, S., Harchol-Balter, M., Scheller-Wolf, A.: Exact analysis of the M/M/k/setup class of Markov chains via recursive renewal reward. In: Proceedings of the ACM SIGMETRICS/international conference on Measurement and modeling of computer systems, pp. 153–166 (2013)

[10] Gardner, K., Abdul Jaleel, J., Wickeham, A., Doroudi, S.: Scalable load balancing in the presence of heterogeneous servers. Performance Evaluation **145**, 102151 (2021)

[11] Gardner, K., Harchol-Balter, M., Scheller-Wolf, A., Velednitsky, M., Zbarsky, S.: Redundancy-d: The power of d choices for redundancy. Operations Research **65**(4), 1078–1094 (2017)

[12] Gardner, K., Stephens, C.: Smart dispatching in heterogeneous systems. ACM SIGMETRICS Performance Evaluation Review **47**(2), 12–14 (2019)

[13] Harchol-Balter, M., Crovella, M.E., Murta, C.D.: On choosing a task assignment policy for a distributed server system. Journal of Parallel and Distributed Computing **59**(2), 204–228 (1999)

[14] Hellemans, T., Van Houdt, B.: On the power-of-d-choices with least loaded server selection. Proceedings of the ACM on Measurement and Analysis of Computing Systems **2**(2), 1–22 (2018)

[15] Hyytiä, E.: Optimal routing of fixed size jobs to two parallel servers. INFOR: Information Systems and Operational Research **51**(4), 215–224 (2013)

[16] Javadi, S.A., Gandhi, A.: Dial: Reducing tail latencies for cloud applications via dynamic interference-aware load balancing. In: 2017 IEEE International Conference on Autonomic Computing (ICAC), pp. 135–144. IEEE (2017)

[17] Karthik, A., Mukhopadhyay, A., Mazumdar, R.: Choosing among heterogeneous server clouds. Queueing Systems **85**(1), 1–29 (2017)

[18] Kim, S.g., Eom, H., Yeom, H.Y.: Virtual machine consolidation based on interference modeling. The Journal of Supercomputing **66**(3), 1489–1506 (2013)

[19] Koole, G.: A simple proof of the optimality of a threshold policy in a two-server queueing system. Systems & Control Letters **26**(5), 301–303 (1995)

[20] Latouche, G., Ramaswami, V.: Introduction to matrix analytic methods in stochastic modeling. SIAM (1999)

[21] Li, Q.L., Chen, C., Fan, R.N., Xu, L., Ma, J.Y.: Queueing analysis of a large-scale bike sharing system through mean-field theory. arXiv preprint arXiv:1603.09560 (2016)

[22] Li, Q.L., Dai, G., Lui, J.C.S., Wang, Y.: The mean-field computation in a supermarket model with server multiple vacations. Discrete Event Dynamic Systems **24**(4), 473–522 (2014)

[23] Li, Q.L., Yang, F.: Mean-field analysis for heterogeneous work stealing models. In: International Conference on Information Technologies and Mathematical Modelling, pp. 28–40. Springer (2015)

[24] Lu, Y., Xie, Q., Kliot, G., Geller, A., Larus, J.R., Greenberg, A.: Join-Idle-Queue: A novel load balancing algorithm for dynamically scalable web services. Performance Evaluation **68**(11), 1056–1071 (2011)

[25] Lubin, M., Dunning, I.: Computing in operations research using Julia. INFORMS Journal on Computing **27**(2), 238–248 (2015)

[26] Luh, H.P., Viniotis, I.: Threshold control policies for heterogeneous server systems. Mathematical Methods of Operations Research **55**(1), 121–142 (2002)

[27] Maji, A.K., Mitra, S., Bagchi, S.: ICE: An integrated configuration engine for interference mitigation in cloud services. In: 2015 IEEE International Conference on Autonomic Computing, pp. 91–100 (2015)

[28] Mitzenmacher, M.: The power of two choices in randomized load balancing. IEEE Transactions on Parallel and Distributed Systems **12**(10), 1094–1104 (2001)

[29] Pu, X., Liu, L., Mei, Y., Sivathanu, S., Koh, Y., Pu, C.: Understanding performance interference of I/O workload in virtualized cloud environments. In: 2010 IEEE 3rd International Conference on Cloud Computing, pp. 51–58 (2010)

[30] Ross, S.M., Kelly, J.J., Sullivan, R.J., Perry, W.J., Mercer, D., Davis, R.M., Washburn, T.D., Sager, E.V., Boyce, J.B., Bristow, V.L.: Stochastic processes, vol. 2. Wiley New York (1996)

[31] Shimkin, N., Shwartz, A.: Control of admission and routing in parallel queues operating in a random environment. In: Proceedings of the 28th IEEE Conference on Decision and Control,, pp. 1064–1065 (1989)

[32] Stolyar, A.L.: Pull-based load distribution in large-scale heterogeneous service systems. Queueing Systems **80**(4), 341–361 (2015)

[33] Votke, S., Jaleel, J.A., Suresh, A., Delasay, M., Doroudi, S., Gandhi, A.: Optimal Markovian dynamic control of interference-prone server farms. In: 2019 IEEE 27th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), pp. 295–308 (2019)

[34] Votke, S., Jaleel, J.A., Suresh, A., Delasay, M., Doroudi, S., Gandhi, A.: Optimal Markovian Dynamic Control of Interference-Prone Server Farms. In: 2019 IEEE 27th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), pp. 295–308 (2019). DOI 10.1109/MASCOTS.2019.00041

[35] Wächter, A., Biegler, L.T.: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical Programming **106**(1), 25–57 (2006)

[36] Wang, C., Feng, C., Cheng, J.: Distributed join-the-idle-queue for low latency cloud services. IEEE/ACM Transactions on Networking **26**(5), 2309–2319 (2018)

[37] Weber, R.R.: On the optimal assignment of customers to parallel servers. Journal of Applied Probability **15**(2), 406–413 (1978)

[38] Xu, F., Liu, F., Jin, H., Vasilakos, A.V.: Managing performance overhead of virtual machines in cloud computing: A survey, state of the art, and future directions. Proceedings of the IEEE **102**(1), 11–31 (2013)

A Appendix: Proofs of Results

A.1 Proof of Proposition 1

Deriving $\mathcal{R}$ and $\mathcal{R}_{0s}$:

We justify the expression for $\mathcal{R}$ below:

$$\begin{aligned}\mathcal{R} =& \mathbb{E}[\text{Reward accumulated in state } (0,f) \text{ before exiting it}] \\ &+ \mathbb{P}\left[\text{Move to } (1,f) \text{ from } (0,f)\right] \times \mathbb{E}\left[\text{Reward accumulated starting from } (1,f) \text{until end of cycle}\right] \\ &+ \mathbb{P}\left[\text{Move to } (0,s) \text{ from } (0,f)\right] \times \mathbb{E}\left[\text{Reward accumulated starting from } (0,s) \text{until end of cycle}\right],\end{aligned} \tag{8}$$

in which:

$$\mathbb{E}[\text{Reward accumulated in state } (0,f) \text{ before exiting it}] = 0, \quad \text{by definition of } R(t), \tag{9}$$

$$\mathbb{P}\left[\text{Move to } (1,f) \text{ from } (0,f)\right] = \frac{\lambda_{0f}}{\lambda_{0f}+\alpha_f}, \tag{10}$$

$$\mathbb{P}\left[\text{Move to } (0,s) \text{ from } (0,f)\right] = \frac{\alpha_f}{\lambda_{0f}+\alpha_f}, \tag{11}$$

$$\mathbb{E}\left[\text{Reward accumulated starting from } (1,f) \text{until end of cycle}\right] = \mathcal{R}_f + P_{fs}\mathcal{R}_{0s}, \tag{12}$$

$$\mathbb{E}\left[\text{Reward accumulated starting from } (0,s) \text{until end of cycle}\right] = \mathcal{R}_{0s}. \tag{13}$$

Substituting Eqs. (9)-(13) in Eq. (8) results in the expression for $\mathcal{R}$ in the proposition. The expression for $\mathcal{R}_{0s}$ can be derived in a similar fashion.

Deriving $\mathcal{R}_f$ and $\mathcal{R}_s$:

We justify the expression for $\mathcal{R}_f$ below:

$$\begin{aligned}\mathcal{R}_f =& \mathbb{E}[\text{Reward accumulated during time in state } (1,f) \text{ before leaving it}] \\ &+ \mathbb{P}\left[\text{Move to } (2,f) \text{ from } (1,f)\right] \times \mathbb{E}[\text{Reward accumulated starting from } (2,f) \text{until it reaches state } (0,f) \text{ or } (0,s)] \\ &+ \mathbb{P}\left[\text{Move to } (1,s) \text{ from } (1,f)\right] \times \mathbb{E}[\text{Reward accumulated starting from } (1,s) \text{until it reaches state } (0,f) \text{ or } (0,s)],\end{aligned} \tag{14}$$

in which:

$$\mathbb{E}[\text{Reward accumulated during time in state } (1,f) \text{ before leaving it}] = \frac{1}{\lambda_{1f}+\mu_f+\alpha_f}, \tag{15}$$

$$\mathbb{P}\left[\text{Move to } (2,f) \text{ from } (1,f)\right] = \frac{\lambda_{1f}}{\lambda_{1f}+\alpha_f+\mu_f}, \tag{16}$$

$$\mathbb{P}\left[\text{Move to } (1,s) \text{ from } (1,f)\right] = \frac{\alpha_f}{\lambda_{1f}+\alpha_f+\mu_f}, \tag{17}$$

$$\begin{aligned}
&\mathbb{E}[\text{Reward accumulated starting from } (2,f) \text{until it reaches state } (0,f) \text{ or } (0,s)] = \\
&\quad \mathbb{E}[\text{Reward accumulated starting from state } (2,f) \text{until we reach } (1,f) \text{ or (1,s)}] \\
&\qquad + \mathbb{P}\left[\text{Enter state } (1,f) \text{ before } (1,s) \text{ from } (2,f)\right] \times \\
&\qquad\quad \mathbb{E}[\text{Reward accumulated starting from } (1,f) \text{until it reaches state } (0,f) \text{ or } (0,s)] \\
&\qquad + \mathbb{P}\left[\text{Enter state } (1,s) \text{ before } (1,f) \text{ from } (2,f)\right] \times \\
&\qquad\quad \mathbb{E}[\text{Reward accumulated starting from } (1,s) \text{until it reaches state } (0,f) \text{ or } (0,s)] \\
&= (T_f + \mathcal{R}_f) + P_{ff}\mathcal{R}_f^N + P_{fs}\mathcal{R}_s.
\end{aligned} \tag{18}$$

In Eq. (18), note that we use the earlier defined property that the expected reward accumulated with the instantaneous reward rate $R_N(t)$ when the tagged server begins in state (n,x) until it reaches state $(n-1,f)$ or $(n-1,s)$ for all $n \geq 1$ will be $\mathcal{R}_x + (n-1)\,T_x$.

Substituting Eqs. (15)-(18) in Eq. (14) results in the equation for $\mathcal{R}_f$ in the proposition. The expression for $\mathcal{R}_s$ can be derived in a similar fashion.

Deriving $\mathcal{T}$:

We have:

$$\begin{aligned}
\mathcal{T} = &\mathbb{E}[\text{Residing time in } (0,f) \text{ when renewal cycle begins}] \\
&+ \mathbb{P}[\text{Transition to } (1,f) \text{ from } (0,f)] \cdot \mathbb{E}[\text{Time to reach } (0,f) \text{ from } (1,f)] \\
&+ \mathbb{P}[\text{Transition to } (0,s) \text{ from } (0,f)] \cdot \mathbb{E}[\text{Time to reach } (0,f) \text{ from } (0,s)].
\end{aligned} \tag{19}$$

In Eq. (19):

$$\mathbb{E}[\text{Residing time in } (0,f) \text{ when renewal cycle begins}] = \frac{1}{\lambda_{0f}+\alpha_f} \tag{20}$$

$$\mathbb{P}[\text{Transition to } (1,f) \text{ from } (0,f)] = \frac{\lambda_{0f}}{\lambda_{0f}+\alpha_f}, \tag{21}$$

$$\begin{aligned}\mathbb{E}[\text{Time to reach } (0,f) \text{ from } (0,s)] &= \mathbb{E}[\text{Residing time in } (0,s)] \\ &\quad + \mathbb{P}[\text{Transition to } (1,s) \text{ from } (0,s)] \cdot \mathbb{E}[\text{Time to reach } (0,f) \text{ from } (1,s)] \\ &= \frac{1}{\lambda_{0s}+\alpha_s} + \frac{\lambda_{0s}}{\lambda_{0s}+\alpha_s}\left(T_s + P_{ss}\,\mathbb{E}[\text{Time to reach } (0,f) \text{ from } (0,s)]\right)\end{aligned}$$

Rearranging the terms we get:

$$\mathbb{E}[\text{Time to reach } (0,f) \text{ from } (0,s)] = \frac{1+\lambda_{0s}T_s}{\alpha_s+\lambda_{0s}P_{sf}}. \tag{22}$$

$$\begin{aligned}\mathbb{E}[\text{Time to reach } (0,f) \text{ from } (1,f)] &= \mathbb{E}[\text{Time to reach } (0,f) \text{ or } (0,s) \text{ from } (1,f)] \\ &\quad + \mathbb{P}[\text{Enter } (0,s) \text{ before } (0,f) \text{ from } (1,f)] \cdot \mathbb{E}[\text{Time to reach } (0,f) \text{ from } (0,s)] \\ &= T_f + P_{fs}\frac{1+\lambda_{0s}T_s}{\alpha_s+\lambda_{0s}P_{sf}}\end{aligned} \tag{23}$$

$$\mathbb{P}[\text{Transition to } (0,s) \text{ from } (0,f)] = \frac{\alpha_f}{\lambda_{0f}+\alpha_f}, \tag{24}$$

Substituting Eqs. (20)–(24) in Eq. (19), we obtain:

$$\mathcal{T} = \frac{\beta_{0f}+\beta_{0s}+\lambda_{0f}\beta_{0s}T_f+\lambda_{0s}\beta_{0f}T_s}{(\alpha_f+\lambda_{0f})\beta_{0s}}.$$

where $\beta_{jx} = \alpha_x + \lambda_{jx}P_{xy}$ and $\gamma_x = \lambda_{1x} - \mu_x, j \in \{0,1\}, x \in \{f,s\}$, and $y = \{f,s\} \setminus x$.

A.2 Proof of Lemma 1

We explain the derivation of P_{ff} below. The equations for P_{sf} can be justified similarly. It follows by definition that $P_{fs} = 1 - P_{ff}$ and $P_{ss} = 1 - P_{sf}$.

The tagged server can transition to three states when the server is in any state (n,f), $n \geq 1$:

1. State $(n+1,f)$, with probability $\lambda_{1f}/(\lambda_{1f}+\mu_f+\alpha_f)$, from which the server can experience two sets of paths such that it visits $(n-1,f)$ before $(n-1,s)$: (i) Visit (n,f) before (n,s) and then visit $(n-1,f)$ before $(n-1,s)$; the probability of such a path is P_{ff}^2; (ii) visit (n,s) before (n,f) and then visit $(n-1,f)$ before $(n-1,s)$; the probability of such a path is $P_{fs}P_{sf}$. The contribution of this state to measure P_{ff}:
$$\frac{\lambda_{1f}}{\lambda_{1f}+\mu_f+\alpha_f}\left(P_{ff}^2 + P_{fs}P_{sf}\right)$$
2. State (n,s), with probability $\alpha_f/(\lambda_{1f}+\mu_f+\alpha_f)$, from which the server reaches $(n-1,f)$ before $(n-$

$1, s)$ with probability P_{sf}. The contribution of this state to measure P_{ff}:

$$\frac{\alpha_f}{\lambda_{1f} + \mu_f + \alpha_f} P_{sf}$$

3. State $(n-1, f)$, with probability $\mu_f/(\lambda_{1f} + \mu_f + \alpha_f)$, from which the server reaches $(n-1, f)$ before $(n-1, s)$. The contribution of this state to measure P_{ff}:

$$\frac{\mu_f}{\lambda_{1f} + \mu_f + \alpha_f}$$

Adding up the above three probabilities results in the equation for P_{ff}.

A.3 Proof of Lemma 2

Looking at the state transition diagram in Fig. 2, we obtain the following equations for T_f and T_s:

$$\begin{aligned} T_f &= \frac{1 + \lambda_{1f}(T_f + P_{ff}T_f + P_{fs}T_s) + \alpha_f T_s}{(\lambda_{1f} + \mu_f + \alpha_f)}, \\ T_s &= \frac{1 + \lambda_{1s}(T_s + P_{sf}T_f + P_{ss}T_s) + \alpha_s T_f}{(\lambda_{1s} + \mu_s + \alpha_s)}. \end{aligned} \tag{25}$$

We justify the expression for T_f, the expression for T_s then follows suit. Given that the tagged server is in state (n, f) where $n \geq 1$, the expected time it stays in state (n, f) is $1/(\lambda_{1f} + \mu_f + \alpha_f)$. With probability $\lambda_{1f}/(\lambda_{1f} + \mu_f + \alpha_f)$, the next state of the server is $(n+1, f)$ and with probability $\alpha_f/(\lambda_{1f} + \mu_f + \alpha_f)$, the next state of the server is (n, s). Once it enters state $(n+1, f)$, it takes on average T_f time to reach state (n, f) or (n, s). The probability of reaching (n, f) before (n, s) from $(n+1, f)$ is P_{ff}, and the server then takes on average T_f time to reach $(n-1, f)$ or $(n-1, s)$ from (n, f). Similarly, the probability of reaching (n, s) before (n, f) from $(n+1, f)$ is P_{fs} and the server takes on average T_s time to reach $(n-1, f)$ or $(n-1, s)$ from (n, s). Solving the simultaneous equations given in (25) in terms of T_f and T_s we obtain Eq. (5).

A.4 Proof of Lemma 3

As we did for calculating expected response time, we use the **RRR** method that takes advantage of the repeating structure of the state transitions to calculate π_{0f} and π_{0s}. The Markov chain shown in Fig. 2 is ergodic, and hence, we can find expressions for the limiting probabilities π_{0f} and π_{0s} using renewal theory. First let us consider π_{0f}. To calculate π_{0f}, let us denote the instantaneous rate of reward collected at time t by $\bar{R}(t)$. We set $\bar{R}(t) = 1$ if the server is in state $(0, f)$ at time t and $\bar{R}(t) = 0$ if it is in any other state at time t. To facilitate the calculation of π_{0f}, we consider that the tagged server's dynamics renews itself each time it enters state $(0, f)$ - just like we defined the renewal cycle for calculating the

expected response time. Hence $\mathcal{T}$ denotes the expected length of a cycle (renewal), which is the average time taken by the tagged server to exit and return to state $(0, f)$. Using the renewal theorem, it follows:

$$\pi_{0f} = \lim_{s\to\infty} \frac{\int_0^s \bar{R}\,(t)\,dt}{s} = \frac{\mathbb{E}\left[\int_{\text{cycle}} \bar{R}\,(t)\,dt\right]}{\text{Average cycle length}} = \frac{\bar{\mathcal{R}}}{\mathcal{T}}, \tag{26}$$

where $\bar{\mathcal{R}}$ is the expected reward accumulated in one renewal, which is the average amount of time spent in state $(0, f)$ before leaving, i.e.,

$$\bar{\mathcal{R}} = \frac{1}{(\lambda_{0f} + \alpha_f)} \tag{27}$$

We have the relation for $\mathcal{T}$ presented in Proposition 1. Substituting Eqs. (4), (27) in Eq. (26) we get:

$$\pi_{0f} = \frac{\bar{\mathcal{R}}}{\mathcal{T}} = \frac{\beta_{0s}}{\beta_{0f} + \beta_{0s} + \lambda_{0f}\beta_{0s}T_f + \lambda_{0s}\beta_{0f}T_s}. \tag{28}$$

For computing π_{0s}, we similarly use the instantaneous rate of reward $\hat{R}\,(t)$, $\hat{R}\,(t) = 1$ if the server is in state $(0, s)$ at time t and $\hat{R}\,(t) = 0$ otherwise. To facilitate the calculation of π_{0s}, we can use a different definition of the renewal cycle than that used to calculate π_{0f}. To determine π_{0s}, let us consider that the tagged server sub-system's dynamics renews itself each time it enters state $(0, s)$ and the corresponding expected length of a cycle (renewal) which is the average time taken by the tagged server to exit and return to state $(0, s)$ be $\widetilde{\mathcal{T}}$. Again, using the renewal theorem, it follows:

$$\pi_{0s} = \lim_{s\to\infty} \frac{\int_0^s \hat{R}\,(t)\,dt}{s} = \frac{\mathbb{E}\left[\int_{\text{cycle}} \hat{R}\,(t)\,dt\right]}{\text{Average cycle length}} = \frac{\hat{\mathcal{R}}}{\widetilde{\mathcal{T}}} \tag{29}$$

where $\hat{\mathcal{R}}$ is the expected reward accumulated in one renewal which is the average amount of time spent in state $(0, s)$ before leaving, i.e.,

$$\hat{\mathcal{R}} = \frac{1}{(\lambda_{0s} + \alpha_s)} \tag{30}$$

$\widetilde{\mathcal{T}}$ can be determined following the steps similar to determining $\mathcal{T}$ in Proposition 1. We get:

$$\widetilde{\mathcal{T}} = \frac{(\beta_{0f} + \beta_{0s} + \lambda_{0f}\beta_{0s}T_f + \lambda_{0s}\beta_{0f}T_s)}{((\alpha_s + \lambda_{0s})\beta_{0f})} \tag{31}$$

Substituting Eq. (30), (31) in Eq. (29) we get:

$$\pi_{0s} = \frac{\hat{\mathcal{R}}}{\widetilde{\mathcal{T}}} = \frac{\beta_{0f}}{\beta_{0s} + \beta_{0f} + \lambda_{0f}\beta_{0s}T_f + \lambda_{0s}\beta_{0f}T_s}. \tag{32}$$

A.5 Proof of Proposition 2

Given that asymptotic independence holds, we can examine the Markov chain governing any tagged server. A direct application of Theorem 7.2.3 from [20] shows that all such Markov chains are positive recurrent. Moreover, by examining the case of PLCFS (preemptive last come first served) scheduling—which performs the same as FCFS (first come first served) scheduling with respect to $\mathbb{E}[T]$— we find that the T_f, T_s, and $\mathbb{E}[T]$ values associated with a tagged server are finite so long as the average arrival rate (once the chain begins to repeat), $(\lambda_{1f}(\pi_f - \pi_{0f}) + \lambda_{1s}(\pi_s - \pi_{0s}))/(1-\pi_0)$, is less than the average service rate, $(\alpha_s\mu_f + \alpha_f\mu_s)/(\alpha_f + \alpha_s)$. It is easy to show that this average arrival rate is at most λ for all policies under consideration, which completes the proof.

A.6 Proof of Proposition 3

JFIQ-d (weakly) outperforms JFQ-d (i.e., $\mathbb{E}[T]^{\textbf{JFIQ-}d} \leq \mathbb{E}[T]^{\textbf{JFQ-}d}$). We will use a first policy iteration argument to prove this result. First, observe that by Little's Law, the problem of minimizing the average response time is equivalent to minimizing the average number of jobs in the system,

$$E[N] = \mathbb{E}\left[\lim_{s\to\infty} \frac{1}{s}\int_0^s N(t)\,dt\right] = \lim_{s\to\infty}\frac{1}{s}\,\mathbb{E}\left[\int_0^s N(t)\,dt\right], \tag{33}$$

where $N(t)$ is the number of jobs in the system at time t. Next, observe that since (i) the dispatcher does not use job size information when dispatching and (ii) jobs sizes are exponentially distributed, then all work-conserving scheduling policies that—like the dispatcher—do not make use of job size information will result in the same mean response. Furthermore, the stochastic evolution of $N(t)$ is the same under all such policies. Hence, let us assume (without loss of generality) that we are employing the PLCFS (preemptive last come first served) scheduling policy, where all servers serve the job that most recently arrived to their subsystem at all times, interrupting the service of the job currently being served whenever a new job arrives to the subsystem. Interrupted jobs will later be resume service with no loss of work, but since job sizes are exponentially distributed, the remaining service time when a previously interrupted job resumes service must also be exponentially distributed (with the same rate as that of the overall job size distribution).

Our first policy iteration argument works as follows: first, we take a system employing **JFQ-d** policy and consider one particular arrival at any time t where the dispatcher is allowed to choose between **JFQ-d** and **JFIQ-d** policies (the query size d being the same for both), we will show that irrespective of the state of the system the dispatcher can only benefit with the **JIFQ-d** policy in expectation, this will be sufficient to prove the result that **JFIQ-d** weakly outperforms **JFQ-d**.

Let a job arrival occur at time t. The dispatcher queries d servers. The **JFIQ-d** policy can differ from the **JFQ-d** policy only under two scenarios that can arise from the result of this query: (i) if the query result includes both an idle fast server and a busy fast server, and (ii) if none of the servers queried

were fast (i.e., all queried servers are undergoing interference), and among those (slow) servers that were queried, at least one is idle and at least one is busy.

Let us consider the first scenario: **JFIQ**-d will dispatch to an idle fast server while **JFQ**-d can dispatch to either an idle fast server or to a busy fast server. When the **JFQ**-d policy would ultimately send the job to an idle fast server, then again there will be no difference in performance. However, when the **JFQ**-d policy would ultimately send the job to a busy fast server, there may be a difference in performance.

From Eq. (33), the measure of interest to calculate to compare the two decisions will be

$$\mathbb{E}\left[\int_0^s N(x)\,dx \middle| \text{send job arriving at time } t \text{ to a busy fast server with } n \text{ jobs}\right] - \mathbb{E}\left[\int_0^s N(x)\,dx \middle| \text{send job arriving at time } t \text{ to an idle fast server}\right] \tag{34}$$

for a value of s that is sufficiently large compared to the job's arrival time t.

Now observe that since we are assuming PLCFS scheduling and since the new arrival will be sent to a fast server, the distribution of the response time of the new arrival will not depend on the server (and specifically the idle/busy status of the server) to which it is dispatched: the job's response time will be distributed like a busy period duration beginning in the fast (i.e., interference-free state). In particular, the expected response time of that job will be T_x. Moreover, the server to which the job is sent will not affect the response time of future arrivals to these servers (assuming the **JFQ**-d policy will be used for all further dispatching decisions), as all future arrivals will preempt any existing jobs. Therefore, the choice of the server to which we dispatch this arrival impacts $\int_0^s N(x)\,dx$ only insofar as it affects the jobs already present at the server (to which the job is sent) at the time of the job's arrival. Hence, the quantity in expression (34) will be equal to the cumulative increase in response times experienced by the n jobs already present at the busy fast server should the new arrival be sent to that server (rather than the idle fast server).

As a consequence of the new arrival being sent to the busy server, each of the n preexisting jobs will experience an increase in their response time equal to the duration of one busy period (due to the time to process the new arrival and the jobs that preempt its processing, if any). Since these busy periods are of strictly positive duration (and since $n \geq 1$), the value in expression (34) is strictly positive, which allows us to conclude that **JFIQ**-d will result in a lower response time than **JFQ**-d under this scenario.[4]

It remains to show that the same result holds in the second scenario where all queried servers are slow and **JFIQ**-d will dispatch to an idle slow server while **JFQ**-d can dispatch to either an idle fast server or

[4]We note that the increase in response time need not be the same for all n such jobs, as some such busy periods may begin in the fast state and some will begin in the slow state. One can show that the job that is k-th in service order (but the $(n-k)$-th of the n jobs in arrival order) will experience an expected response time increase of

$$B_k \equiv \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} P_{ff} & P_{fs} \\ P_{sf} & P_{ss} \end{pmatrix}^k \begin{pmatrix} T_f \\ T_s \end{pmatrix},$$

and so the quantity in expression (34) is precisely $\sum_{k=1}^{n} B_k$.

a busy fast server. An argument analogous to the one presented for the first scenario also holds for the second scenario. This completes the proof. □

JIQ-d (weakly) outperforms RND (i.e., $\mathbb{E}[T]^{\mathbf{JIQ}\text{-}d} \leq \mathbb{E}[T]^{\mathbf{RND}}$). We facilitate this proof by presenting a method for finding $\mathbb{E}[T]$ under the **JIQ-d** and **RND** policies that is an alternative to that presented in Section 3. Specifically, we compute $\mathbb{E}[T]$ under the assumption that jobs are served in PLCFS order; this assumption is without loss of generality because—as we have discussed in the preceding proof (which shows that $\mathbb{E}[T]$ **JFIQ-d** $\leq \mathbb{E}[T]^{\mathbf{JFQ}\text{-}d}$)—$\mathbb{E}[T]$ is the same under both FCFS and PLCFS scheduling. This enables us to give the response time expressions for **RND** and **JIQ-d** in terms of their respective T_f and T_s (defined before Proposition 1 and expression presented in Lemma 2).

For the **RND** Policy under PLCFS, all job arrivals to the fast state experience an average response time of $T_f^{\mathbf{RND}}$, and all job arrivals to the slow state experience an average response time of $T_s^{\mathbf{RND}}$. So, we have for the **RND** policy:

$$\mathbb{E}[T]^{\mathbf{RND}} = \pi_f T_f^{\mathbf{RND}} + \pi_s T_s^{\mathbf{RND}}. \tag{35}$$

For the **JIQ-d** policy, note that the average arrival rate to any server is still the same as that for **RND** (i.e., λ). However, if we tag any server, its job arrival rate when idle (λ_0) is greater than λ, and its job arrival rate when busy (λ_1) will be less than λ, which is a consequence of the definition of **JIQ-d** (see Lemma 4.3). Next observe that under PLCFS, the jobs that arrive in the fast and slow states will experience average response times of $T_f^{\mathbf{JIQ}\text{-}d}$ and $T_s^{\mathbf{JIQ}\text{-}d}$, respectively; i.e., a

$$\frac{\lambda_0 \pi_{0f}^{\mathbf{JIQ}\text{-}d} + \lambda_1(\pi_f - \pi_{0f}^{\mathbf{JIQ}\text{-}d})}{\lambda} \quad \text{and} \quad \frac{\lambda_0 \pi_{0s}^{\mathbf{JIQ}\text{-}d} + \lambda_1(\pi_s - \pi_{0s}^{\mathbf{JIQ}\text{-}d})}{\lambda}$$

fraction of jobs will experience an average response time of $T_f^{\mathbf{JIQ}\text{-}d}$ and $T_s^{\mathbf{JIQ}\text{-}d}$, respectively, yielding:

$$\mathbb{E}[T]^{\mathbf{JIQ}\text{-}d} = \left(\frac{(\lambda_0 - \lambda_1)\pi_{0f}^{\mathbf{JIQ}\text{-}d} + \lambda_1 \pi_f}{\lambda}\right) T_f^{\mathbf{JIQ}\text{-}d} + \left(\frac{(\lambda_0 - \lambda_1)\pi_{0s}^{\mathbf{JIQ}\text{-}d} + \lambda_1 \pi_s}{\lambda}\right) T_s^{\mathbf{JIQ}\text{-}d}. \tag{36}$$

Note that π_f and π_s only depend on the parameters α_f and α_s, and hence it is needless to distinguish them across policies, unlike T_f and T_s. Now the following four relations can be observed from the state transition Markov chains for the **RND** and **JIQ-d** policies:

$$(1)\ T_f^{\mathbf{JIQ}\text{-}d} \leq T_s^{\mathbf{JIQ}\text{-}d}, \quad (2)\ T_f^{\mathbf{RND}} \leq T_s^{\mathbf{RND}}, \quad (3)\ T_f^{\mathbf{JIQ}\text{-}d} \leq T_f^{\mathbf{RND}}, \quad (4)\ T_s^{\mathbf{JIQ}\text{-}d} \leq T_s^{\mathbf{RND}}$$

Taking account of the above relations along with Eqs. (35) and (36) (both of which are written as weighted averages of T_f and T_s values), if we can show that the coefficient of $T_f^{\mathbf{JIQ}\text{-}d}$ in Eq. (36) is greater than or

equal to that of $T_f^{\mathbf{RND}}$ in Eq. (35), i.e.

$$\frac{(\lambda_0 - \lambda_1)\pi_{0f}^{\mathbf{JIQ}\text{-}d} + \lambda_1 \pi_f}{\lambda} \geq \pi_f, \tag{37}$$

then we must have $\mathbb{E}[T]^{\mathbf{JIQ}\text{-}d} \leq \mathbb{E}[T]^{\mathbf{RND}}$, which proves the claim.

Rearranging terms, we find that Ineq. (37) is equivalent to

$$\pi_{0f}^{\mathbf{JIQ}\text{-}d} \geq \frac{\lambda - \lambda_1}{\lambda_0 - \lambda_1}\pi_f. \tag{38}$$

Next, we show that

$$\frac{\lambda - \lambda_1}{\lambda_0 - \lambda_1} = \pi_0^{\mathbf{JIQ}\text{-}d}, \tag{39}$$

which allows us to rewrite Ineq. (38) (which if true, would complete the proof) as $\pi_{0f}^{\mathbf{JIQ}\text{-}d} \geq \pi_0^{\mathbf{JIQ}\text{-}d}\pi_f$. Recalling that the average arrival rate to a tagged server under any policy is λ, it must be the case that $\pi_0^{\mathbf{JIQ}\text{-}d}\lambda_0 + (1 - \pi_0^{\mathbf{JIQ}\text{-}d})\lambda_1 = \lambda$; rearranging yields the desired result, Eq. (39), from which it follows that in order to complete the proof it is sufficient to show that $\pi_{0f}^{\mathbf{JIQ}\text{-}d} \geq \pi_0^{\mathbf{JIQ}\text{-}d}\pi_f$. Furthermore, since we have $\pi_0^{\mathbf{JIQ}\text{-}d} = \pi_{0f}^{\mathbf{JIQ}\text{-}d} + \pi_{0s}^{\mathbf{JIQ}\text{-}d}$, it is sufficient to show that $\pi_{0f}^{\mathbf{JIQ}\text{-}d} \geq \left(\pi_{0f}^{\mathbf{JIQ}\text{-}d} + \pi_{0s}^{\mathbf{JIQ}\text{-}d}\right)\pi_f$, which (noting that $1 - \pi_f = \pi_s$) is equivalent to showing that

$$\frac{\pi_{0f}^{\mathbf{JIQ}\text{-}d}}{\pi_f} \geq \frac{\pi_{0s}^{\mathbf{JIQ}\text{-}d}}{\pi_s} \tag{40}$$

So, it remains only to show that Ineq. (40), and we can interpret the left-hand side (respectively, the right-hand side) of this inequality as the probability of the tagged-server's subsystem being empty given that the tagged server is in the fast (respectively, slow) interference state. So, if we can show that under the **JIQ**-d policy, the likelihood of being idle is greater when we know that we are in the fast state as compared to when we know we are in the slow state, then **JIQ**-d must outperform **RND** as claimed. To this end, we leverage our expressions for π_{0f} and π_{0s} as presented in Lemma 3 (and let $\lambda_{0f} = \lambda_{0s} = \lambda_0$ $\lambda_{1f} = \lambda_{1s} = \lambda_1$ as is the case under **JIQ**-d), in order to obtain

$$\frac{\pi_{0f}^{\mathbf{JIQ}\text{-}d} \Big/ \pi_f}{\pi_{0s}^{\mathbf{JIQ}\text{-}d} \Big/ \pi_s} = \frac{\alpha_f(\alpha_s + \lambda_0 P_{sf})}{\alpha_s(\alpha_f + \lambda_0 P_{fs})}. \tag{41}$$

Clearly Ineq.(41) holds if and only if Expression (40) is greater than or equal to 1, or equivalently if:

$$\frac{P_{sf}}{P_{fs}} \geq \frac{\alpha_s}{\alpha_f}. \tag{42}$$

Next, we show that Ineq. (42) holds true for the **JIQ**-d policy, which is sufficient to complete the proof. We

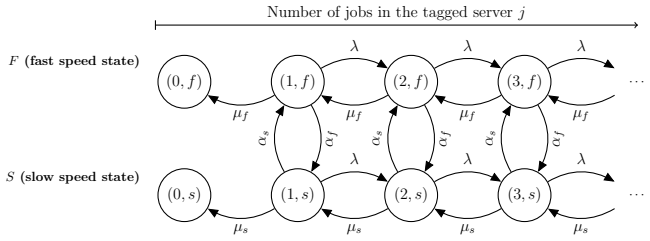


Figure 9: Markov chain to help prove that P_{sf} and P_{ff} increase with decrease in μ_s

do this in the following sequence: first, we show that for any parameter setting with $\mu_s = \mu_f$, Ineq. (42) is satisfied with equality. Second, we will show with the help of an MDP model that given any system parameters λ_1, α_f, α_s, μ_f and μ_s, the ratio P_{sf}/P_{fs} is increasing with decrease in μ_s(as long as the system is stable).

We begin by restricting attention to the speed state transitions in the model, which reduces to the study of the simple two-state Markov chain, where one transitions from the fast state to the slow state with rate α_f and vice-versa with rate α_s.

Let $\bar{P}_{xy}(t)$ denote the probability of being in speed state $y \in \{s, f\}$ at time t, given that the system is in speed state $x \in \{s, f\}$ at time 0. Then by Kolmogrov's Backward and Forward equations we obtain the following two expressions governing the first derivative of $\bar{P}_{fs}(t)$ with respect to t, $\bar{P}'_{fs}(t)$:

$$\bar{P}'_{fs}(t) = \alpha_f \bar{P}_{ss}(t) - \alpha_f \bar{P}_{fs}(t) = \alpha_f(1 - \bar{P}_{sf}(t)) - \alpha_f \bar{P}_{fs}(t) \tag{43}$$

$$\bar{P}'_{fs}(t) = \alpha_f \bar{P}_{ff}(t) - \alpha_s \bar{P}_{fs}(t) = \alpha_f(1 - \bar{P}_{fs}(t)) - \alpha_s \bar{P}_{fs}(t). \tag{44}$$

Equating the right-hand side of Eqs. (43) and (44), we obtain $\bar{P}_{sf}(t)/\bar{P}_{fs}(t) = \alpha_s/\alpha_f$, which is a valid equation for all policies and all times $t \geq 0$.

Now consider the case where $\mu_s = \mu_f$, in which the time taken to reduce the total number in the system by one unit does not depend on the current speed state, and hence we have $P_{sf}/P_{fs} = \alpha_s/\alpha_f$ (i.e., Ineq. (42) holds in the special case where $\mu_s = \mu_f$). Next let us consider any set of problem parameters λ_1, α_f, α_s, μ_f and μ_s. If we can show that the ratio P_{sf}/P_{fs} increases as μ_s decreases, then our claim of **JIQ**-d having an average response time at least as good as the **RND** policy for all problem parameter settings is proved. Moreover, in order to show that P_{sf}/P_{fs} is increasing, it is sufficient to show that both P_{sf} and P_{ff} are increasing with decrease in μ_s, as $P_{fs} = 1 - P_{ff}$. This is our goal in the coming steps. To facilitate showing that both P_{sf} and P_{ff} are increasing with a decrease in μ_s, we analyze the Markov chain in Fig. 9, and we build an MDP model.

Entering (or ending in) the terminal state $(0, f)$ results in a reward of 1 unit. The other terminal state $(0, s)$ and all other states give 0 reward on visits. We use this Markov chain so that the expected reward at each state corresponds to the probability of reaching state $(0, f)$. Let us define the variable $P0F_{(n,x)}$ as the probability of reaching state $(0, f)$ given that the system is currently in state (n, x). Then by

definition, $P0F_{(n,x)}$ will correspond to the expected reward at state (n, x). Let us set $P0F_{(1,f)} = \widetilde{P_{ff}}$ and $P0F_{(1,s)} = \widetilde{P_{sf}}$. Note that if in the Markov chain represented by Fig.9, λ is set to λ_1 of the **JIQ**-d policy, then $\widetilde{P_{ff}}$ and $\widetilde{P_{sf}}$ will correspond to P_{ff} and P_{sf} of the respective **JIQ**-d policy.

Analyzing the Markov chain of Fig. 9 gives the following relations $\forall n \geq 0$:

$$P0F_{(n+1,f)} = \widetilde{P_{ff}} P0F_{(n,f)} + (1 - \widetilde{P_{ff}}) P0F_{(n,s)}, \tag{45}$$

$$P0F_{(n+1,s)} = \widetilde{P_{sf}} P0F_{(n,f)} + (1 - \widetilde{P_{sf}}) P0F_{(n,s)}. \tag{46}$$

Studying Eqs. (45) and (46) gives the following order:

$$P0F_{(0,f)} \geq P0F_{(1,f)} \geq ... \geq P0F_{(n,f)} \geq P0F_{(n,s)} \geq ... \geq P0F_{(1,s)} \geq P0F_{(0,s)}. \tag{47}$$

In the MDP problem, the decision epochs are the instances when an event takes place: this event can be an arrival, a speed state change, or a service completion. The actions available when the server is functioning at a slow speed is to choose service rates μ_s or μ'_s ($\mu'_s \leq \mu_s$ but μ'_s high enough to ensure stability). If the server is functioning in the fast state, jobs are processed at rate μ_f. Let the base policy select μ_s if the server is in the slow speed state. From relations (45)-(47), we observe that the decision of selecting speed μ'_s increases the expected reward for every slow state, and using the notion of policy improvement for MDPs, we have that adopting the μ'_s option at every epoch for all slow speed states improves the expected reward at every state. This proves that decreasing μ_s will improve the ratio $\widetilde{P_{sf}}/\widetilde{P_{fs}}$ and consequently we have the inequality (42) proved for **JIQ**-d. □

JIQ$^{\mathbf{FP}}$-d (weakly) outperforms JIQ-d (i.e., $\mathbb{E}[T]^{\mathbf{JIQ^{FP}}\text{-}d} \leq \mathbb{E}[T]^{\mathbf{JIQ}\text{-}d}$). We use a first policy iteration argument for this proof resembling that used in proving our first dominance result which was that **JFIQ**-d (weakly) outperforms **JFQ**-d. Let us consider a situation where a job arrives and have the option of choosing between **JIQ$^{\mathbf{FP}}$**-d and **JIQ**-d in dispatching this job while assuming that all future job arrivals will be dispatched according to the **JIQ**-d policy. If we can show that in all querying scenarios, **JIQ$^{\mathbf{FP}}$**-d will perform at least as well as **JIQ**-d, then by the first policy iteration argument, we have completed the proof.

If the query result does not contain both fast and slow idle servers, then **JIQ$^{\mathbf{FP}}$**-d and **JIQ**-d will perform identically. In the case of the query result containing both fast and slow idle servers, the **JIQ$^{\mathbf{FP}}$**-d policy will dispatch the server to the idle fast server while the **JIQ**-d policy can dispatch to either the fast or slow idle server. If the **JIQ**-d policy sends the job to an idle fast server, then it imitates the **JIQ$^{\mathbf{FP}}$**-d policy. However in the event that the **JIQ**-d policy dispatches the job to a slow idle server, **JIQ**-d will result in an increased average number in system because the expected length of a busy period associated with a server starting in the fast interference state under **JIQ**-d policy (T_f) is less than the expected length of a busy period associated with a server starting in the in the slow interference state(T_s). The

proof can be completed by following steps very similar to those in our proof establishing that **JFIQ**-d (weakly) outperforms **JFQ**-d.

B Appendix: Uniqueness of P_{ff}, P_{fs}, P_{sf} and P_{ss}

In this appendix, we address the uniqueness of the solution to the system of equations given in Lemma 1 when solving for values $P_{ff}, P_{fs}, P_{sf}, P_{ss} \in [0, 1]$. See that we could also write equations for P_{fs} and P_{ss} conditioning on the next transition rather than exploiting complementarity; in fact these equations would be identical to the equations for P_{sf} and P_{ff}, respectively, except we would need to switch all f and s symbols in the subscripts. For example, we would have

$$P_{fs} = \left(\lambda_{1f}\left(P_{fs}P_{ss} + P_{ff}P_{fs}\right) + \alpha_s P_{ss}\right)/(\lambda_{1f} + \mu_f + \alpha_f).$$

Any solutions to our original system would also be solutions to an alternative system of four equations with the same equations with P_{ff} and P_{sf} on the left-hand side, but with the equations with P_{fs} and P_{ss} on the left-hand side replaced by those described above. This alternative system is equivalent to a matrix quadratic equation given in Eq. (8.15) of [20], where the authors prove in Theorem 8.1.4 that the "correct" probabilities (corresponding to our P_{ff}, P_{fs}, P_{sf} and P_{ss}) are those in the minimal nonnegative solution, meaning that any extraneous solution to this matrix quadratic equation (and hence the alternative system we have constructed) would either have a negative value corresponding to one or more of these variables, or would give a greater value for each variable than that of the correct solution. However, if all such values of an extraneous solution are greater, then this violates the $P_{fs} = 1 - P_{ff}$ and/or $P_{ss} = 1 - P_{sf}$.

C Appendix: Impact Of Higher Query Sizes

Table 6: Optimality proportion and improvement over **RND** when $d = 3$

Best-performing policy	Optimality proportion	Mean (10^{th}–90^{th} percentile) improvement over **RND**
JFIQ-d	390 (37.1%)	81.4% (44.6%–99.8%)
JIFQ-d	418 (39.8%)	53.8% (13.3%–98.3%)
JIQ$^{\mathbf{BR}}$-d	158 (15.0%)	72.0% (52.8%–97.9%)
JIQ$^{\mathbf{FP}}$-d	84 (8.0%)	60.7% (47.3%–66.5%)

Table 7: Optimality proportion and improvement over **RND** when $d = 4$

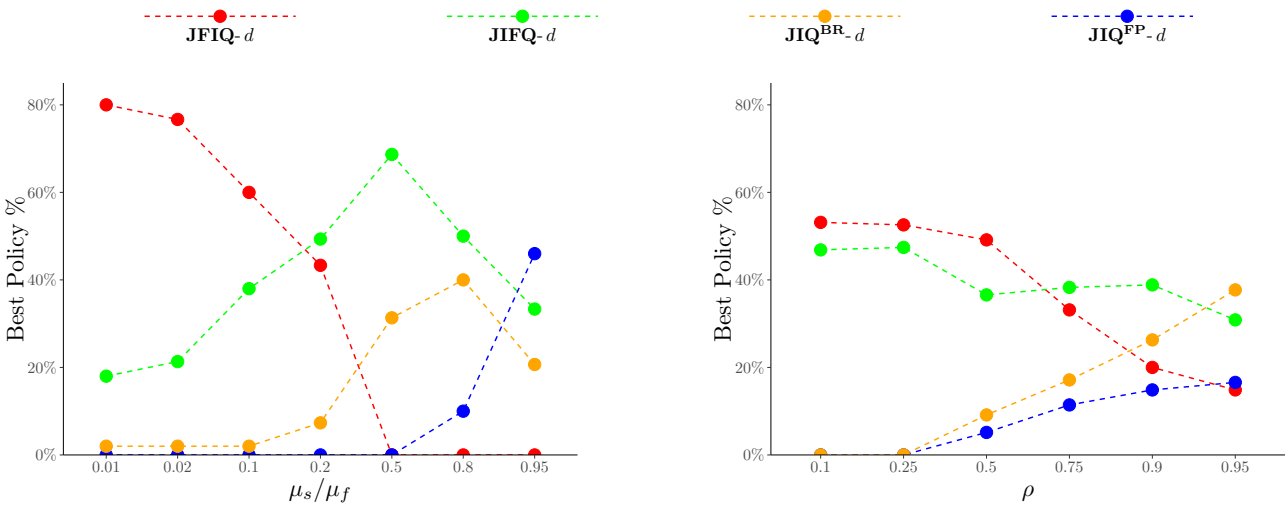
Best-performing policy	Optimality proportion	Mean (10^{th}–90^{th} percentile) improvement over **RND**
JFIQ- d	374 (35.6%)	82.2% (44.0%–99.8%)
JIFQ- d	451 (43.0%)	58.9% (14.6%–99.3%)
$\mathbf{JIQ^{BR}}$- d	153 (14.6%)	76.5% (63.5%–98.3%)
$\mathbf{JIQ^{FP}}$- d	72 (6.9%)	69.1 % (63.5%–74.5%)

Table 8: % increase in the expected response time due to implementing a non-optimal policy; the cell values report the mean (10^{th}–90^{th} percentile) of the % increase ($d = 3$)

Implemented policy	Best-performing policy			
	JFIQ- d	**JIFQ**- d	$\mathbf{JIQ^{BR}}$- d	$\mathbf{JIQ^{FP}}$- d
JFIQ- d	–	90.9 (0.3–35.9)	9680.2 (8.3–16270.1)	9638.5 (16.2–11325.7)
JIFQ- d	8.9 (0.3–25.8)	–	802.7 (0–145.3)	432.7 (0.4–133.5)
$\mathbf{JIQ^{BR}}$- d	23 (0.7–34.5)	34.1 (0–3.9)	–	8.6 (0–23.2)
$\mathbf{JIQ^{FP}}$- d	2350.6 (1.7–4898.2)	2017.4 (0–1264.7)	1042.8 (0–1125)	–
RND	11652.3 (80.5–41384.9)	6340.2 (15.3–5750.8)	3892.8 (115.1–4734.9)	161.7 (92.9–198.4)

Table 9: % increase in the expected response time due to implementing a non-optimal policy; the cell values report the mean (10^{th}–90^{th} percentile) of the % increase ($d = 4$)

Implemented policy	Best-performing policy			
	JFIQ- d	**JIFQ**- d	$\mathbf{JIQ^{BR}}$- d	$\mathbf{JIQ^{FP}}$- d
JFIQ- d	–	102.0 (0.1–45.8)	21177.9 (10.2–43261.2)	16612.1 (19.6–17870.5)
JIFQ- d	10.6 (0.3–30.3)	–	837.4 (0–195.4)	406.1 (0.4–96.6)
$\mathbf{JIQ^{BR}}$- d	16.7 (0.4–38.7)	19.2 (0–4.4)	–	4.7 (0.1–12.7)
$\mathbf{JIQ^{FP}}$- d	2478.6 (0.5–4627.2)	2893.8 (0–2452.9)	590.6 (0–634.9)	–
RND	18005.5 (78.7–61465.6)	11707.7 (17.1–15137.1)	4123 (174.2–5880.3)	237.4 (173.8–292)



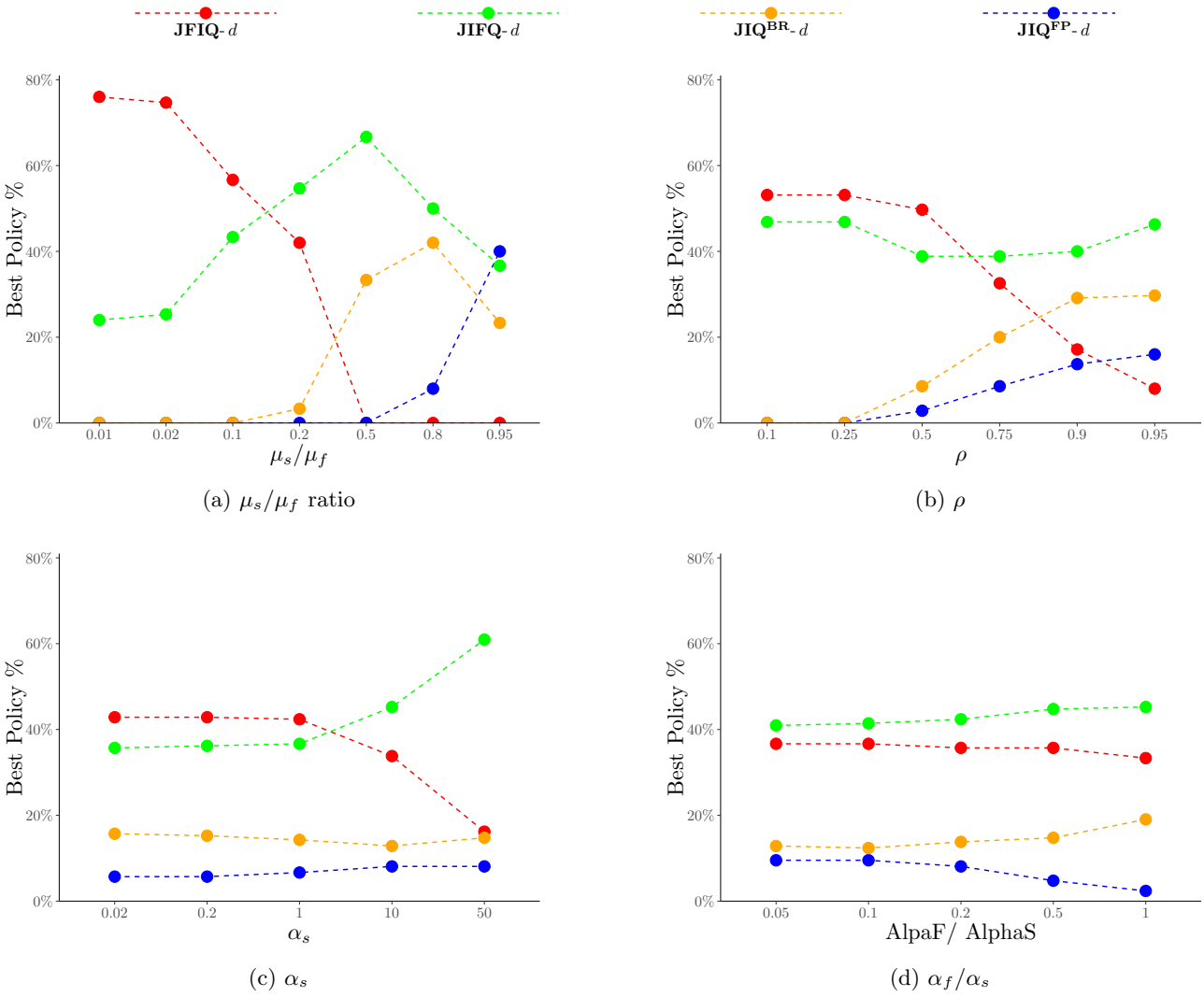
(a) μ_s/μ_f ratio

(b) ρ

(c) α_s

(d) α_f/α_s

Figure 10: Marginal effect of system parameters on the best policy $d = 3$



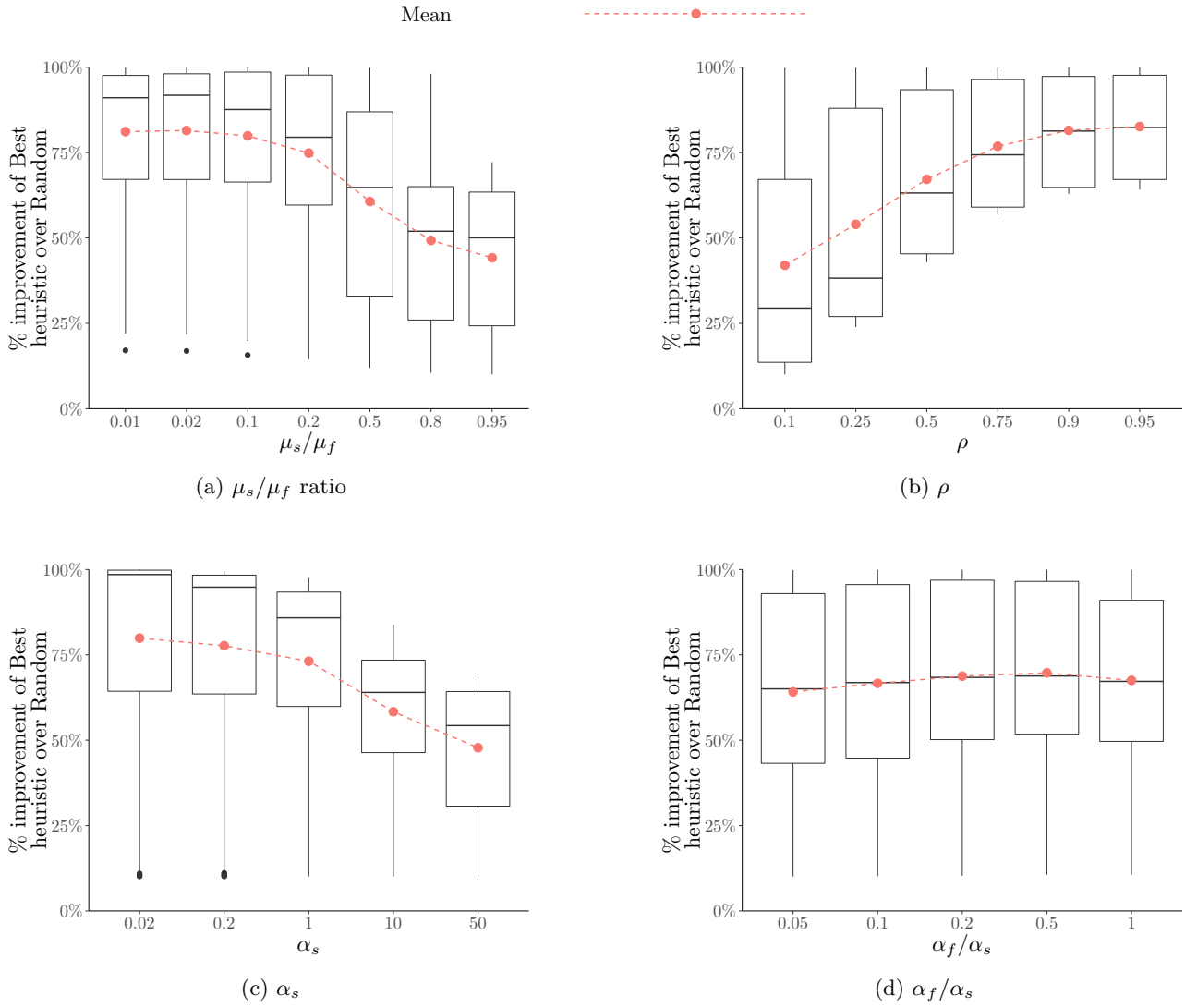
(a) μ_s/μ_f ratio

(b) ρ

(c) α_s

(d) α_f/α_s

Figure 11: Marginal effect of system parameters on the best policy $d = 4$



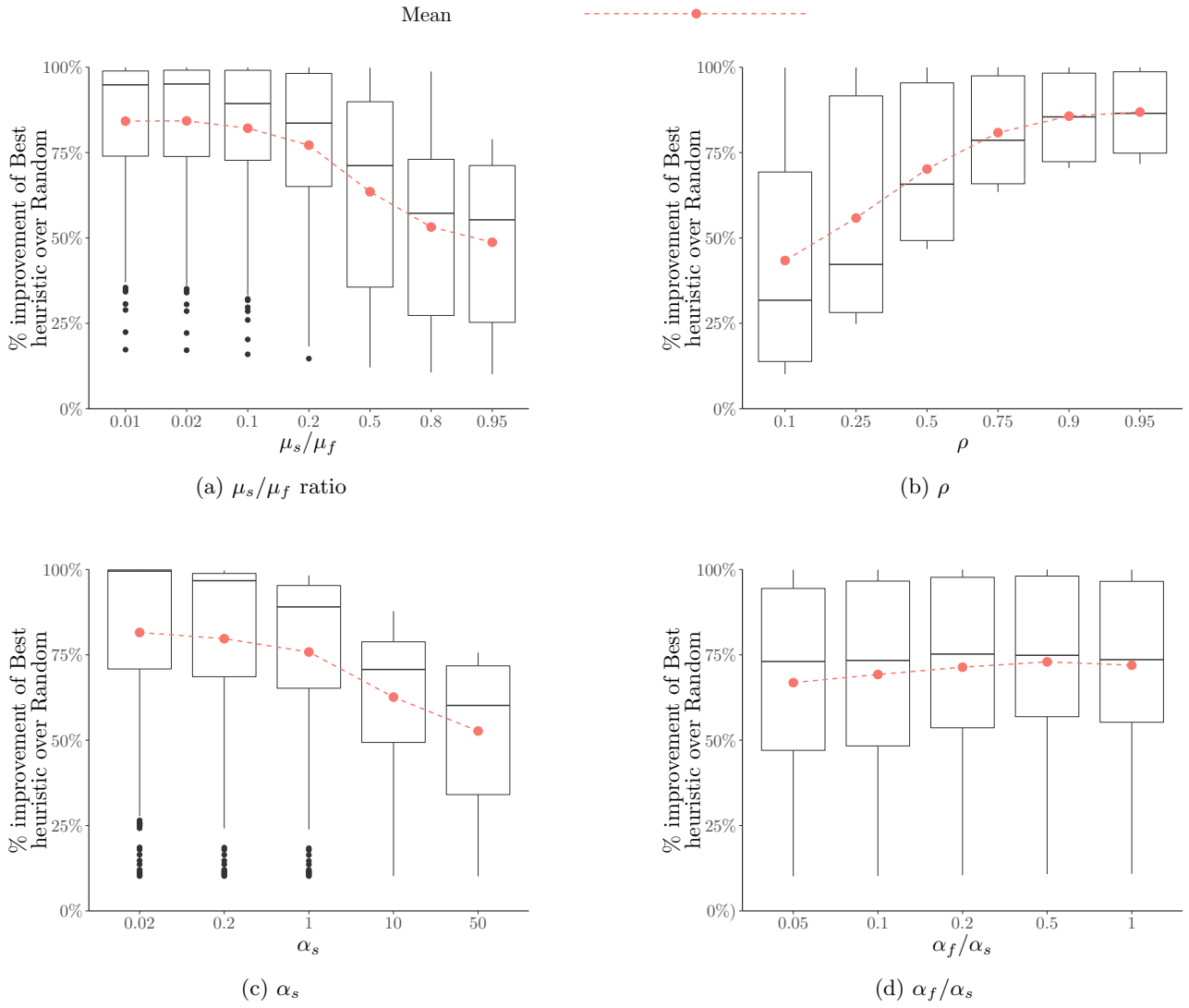
(a) μ_s/μ_f ratio

(b) ρ

(c) α_s

(d) α_f/α_s

Figure 12: Box Plot for the percentage improvement of the best performing heuristic over **RND** ($d = 3$)



(a) μ_s/μ_f ratio

(b) ρ

(c) α_s

(d) α_f/α_s

Figure 13: Box Plot for the percentage improvement of the best performing heuristic over Random when varying each of the four parameter descriptions that define the problem setting. $d = 4$

		α_s		0.02					0.2					1					10					50				
		α_f/α_s		0.05	0.1	0.2	0.5	1	0.05	0.1	0.2	0.5	1	0.05	0.1	0.2	0.5	1	0.05	0.1	0.2	0.5	1	0.05	0.1	0.2	0.5	1
ρ	0.1	μ_s/μ_f	0.01	99.6	99.8	99.8	99.6	93.4	95.9	97.6	98.3	96.9	89.9	82.5	89.3	92.4	90.5	82.1	36.8	49.0	59.5	64.0	57.2	17.1	22.0	27.9	32.5	29.8
			0.02	99.5	99.7	99.8	99.8	99.4	95.3	97.4	98.3	97.9	95.4	81.0	88.4	92.1	91.5	85.3	35.8	47.9	58.5	63.5	57.1	16.9	21.7	27.6	32.2	29.6
			0.1	87.0	89.8	90.6	88.3	81.2	73.9	82.3	86.7	86.5	79.9	56.9	69.5	78.0	81.0	75.2	28.4	38.7	49.3	56.6	52.5	15.7	19.9	25.0	29.5	27.9
			0.2	33.6	45.4	56.7	64.8	62.3	33.0	44.7	56.1	64.2	61.8	31.0	42.3	53.5	62.0	59.6	21.8	29.6	38.7	46.7	44.7	14.5	17.8	22.1	26.5	25.5
			0.5	14.5	18.2	23.3	30.0	31.4	14.5	18.1	23.2	29.9	31.3	14.4	17.9	22.9	29.5	30.8	13.6	16.5	20.6	25.8	26.5	12.0	13.6	15.9	18.7	18.6
			0.8	11.0	11.9	13.3	15.6	16.4	11.0	11.9	13.3	15.5	16.4	11.0	11.9	13.3	15.5	16.3	10.8	11.6	12.9	14.7	15.2	10.5	11.1	11.8	12.9	13.0
			0.95	10.1	10.3	10.6	11.1	11.3	10.1	10.3	10.6	11.1	11.3	10.1	10.3	10.6	11.1	11.3	10.1	10.3	10.5	11.0	11.1	10.0	10.2	10.3	10.6	10.6
	0.25	μ_s/μ_f	0.01	99.7	99.8	99.8	95.7	85.0	96.7	98.1	98.6	94.8	84.4	85.8	91.3	93.8	91.2	81.6	46.6	56.8	65.5	68.8	62.4	29.5	33.3	37.6	40.2	37.5
			0.02	99.6	99.8	99.9	99.6	91.0	96.5	98.1	98.7	97.8	89.8	85.3	91.1	94.0	93.1	85.5	46.1	56.3	65.1	68.8	62.9	29.4	33.1	37.4	40.0	37.4
			0.1	99.4	99.7	99.8	99.8	99.7	94.3	96.8	98.0	98.3	97.3	78.7	86.6	91.2	92.8	90.0	41.6	50.9	59.8	65.2	61.2	28.5	31.7	35.5	38.2	36.7
			0.2	97.7	98.5	98.6	93.2	84.2	84.0	89.4	91.7	89.5	82.4	62.4	72.7	79.6	81.8	77.0	36.2	43.6	51.5	57.2	54.2	27.4	30.0	33.3	36.2	35.2
			0.5	29.1	32.9	37.6	42.7	42.9	29.1	32.8	37.5	42.5	42.7	28.8	32.4	36.9	41.8	42.0	27.2	29.8	33.2	36.9	36.8	25.5	26.7	28.5	30.4	30.1
			0.8	24.7	25.4	26.5	28.0	28.5	24.7	25.4	26.5	28.0	28.4	24.7	25.4	26.5	27.9	28.3	24.6	25.2	26.1	27.3	27.5	24.3	24.7	25.2	25.9	25.9
			0.95	24.0	24.1	24.3	24.6	24.7	24.0	24.1	24.3	24.6	24.7	24.0	24.1	24.3	24.6	24.7	24.0	24.1	24.3	24.5	24.6	23.9	24.0	24.1	24.3	24.3
	0.25	μ_s/μ_f	0.01	99.7	99.9	99.7	95.4	86.9	97.5	98.6	98.7	94.9	86.5	89.4	93.5	95.2	92.6	84.5	59.6	67.0	73.1	75.0	69.0	46.4	48.6	51.0	51.8	49.3
			0.02	99.7	99.9	99.9	97.0	89.0	97.5	98.6	99.0	96.4	88.4	89.2	93.4	95.5	93.7	86.2	59.3	66.7	73.0	75.1	69.5	46.3	48.5	50.8	51.8	49.4
			0.1	99.7	99.8	99.9	99.9	99.9	96.7	98.2	98.9	99.2	98.8	86.7	91.8	94.7	96.0	94.7	57.0	63.9	70.4	73.9	70.3	45.7	47.7	49.9	51.5	49.9
			0.2	99.5	99.7	99.8	99.9	99.8	95.1	97.2	98.2	98.6	98.3	82.1	88.4	92.1	93.9	92.6	53.6	59.5	65.3	68.8	66.0	45.2	46.9	49.0	50.6	49.6
			0.5	74.7	77.2	77.1	73.9	69.1	63.2	69.1	72.6	72.1	68.1	54.3	59.9	64.8	67.2	64.8	46.3	48.7	51.6	54.4	53.8	44.0	44.8	45.9	47.1	46.8
			0.8	43.8	44.5	45.6	46.8	47.1	43.8	44.5	45.5	46.8	47.0	43.7	44.4	45.4	46.5	46.7	43.4	43.9	44.5	45.3	45.3	43.2	43.4	43.7	44.1	44.0
			0.95	43.0	43.0	43.1	43.3	43.3	43.0	43.0	43.1	43.3	43.3	43.0	43.0	43.1	43.3	43.3	42.9	43.0	43.1	43.2	43.2	42.9	43.0	43.0	43.1	43.1
	0.25	μ_s/μ_f	0.01	99.8	99.9	99.6	97.2	91.4	98.1	98.9	99.0	96.8	90.9	91.9	94.9	96.3	94.8	89.1	69.0	74.4	78.8	80.0	74.8	59.0	60.5	62.0	61.9	58.7
			0.02	99.8	99.9	99.9	97.8	92.3	98.1	98.9	99.2	97.3	91.8	91.8	94.9	96.5	95.3	89.9	68.8	74.2	78.7	80.1	75.0	59.0	60.5	61.9	61.9	58.9
			0.1	99.8	99.9	99.9	99.9	99.5	97.7	98.7	99.2	99.4	98.5	90.3	94.0	96.1	97.1	95.3	67.3	72.4	77.0	79.3	75.9	58.8	60.1	61.6	62.0	59.9
			0.2	99.7	99.8	99.9	99.9	99.9	96.9	98.2	98.9	99.2	99.1	88.0	92.4	94.9	96.2	95.6	65.5	70.1	74.7	77.7	75.7	58.5	59.6	61.0	61.7	60.4
			0.5	98.8	99.3	99.6	99.6	99.4	90.7	94.1	96.0	96.7	95.4	75.4	81.4	85.9	88.1	86.2	60.3	62.7	65.5	67.9	67.0	57.6	58.1	58.9	59.5	59.1
			0.8	61.3	63.6	65.8	66.8	65.1	60.1	62.1	64.2	65.6	64.2	58.6	59.9	61.5	62.9	62.0	57.2	57.6	58.1	58.7	58.3	56.9	57.0	57.2	57.4	57.2
			0.95	56.9	56.9	57.1	57.2	57.3	56.9	56.9	57.1	57.2	57.2	56.8	56.9	57.0	57.1	57.1	56.8	56.8	56.9	56.9	56.9	56.8	56.8	56.8	56.8	56.8
	0.25	μ_s/μ_f	0.01	99.8	99.9	99.8	98.5	94.4	98.4	99.0	99.2	98.1	94.0	93.0	95.6	96.9	96.4	92.2	73.4	77.9	81.6	82.8	78.4	64.9	66.0	67.0	66.4	62.9
			0.02	99.8	99.9	99.9	98.8	95.0	98.3	99.1	99.3	98.4	94.6	92.9	95.6	96.9	96.6	92.7	73.2	77.7	81.5	82.8	78.6	64.9	66.0	67.0	66.4	63.1
			0.1	99.8	99.9	99.9	99.9	99.0	98.1	98.9	99.3	99.5	98.4	91.9	95.0	96.7	97.4	95.9	72.2	76.5	80.4	82.4	79.5	64.7	65.6	66.6	66.5	64.1
			0.2	99.7	99.9	99.9	99.9	99.9	97.5	98.6	99.1	99.4	99.4	90.3	93.9	96.0	97.1	96.7	70.7	74.7	78.6	81.1	79.3	64.4	65.2	66.1	66.3	64.7
			0.5	99.3	99.6	99.7	99.8	99.7	93.7	96.2	97.5	98.1	97.3	81.3	86.4	90.1	92.2	89.6	66.3	68.4	70.9	72.8	70.2	63.6	64.0	64.5	64.7	63.9
			0.8	93.3	95.6	96.8	96.9	94.0	74.5	78.9	82.4	84.5	81.7	65.8	67.4	69.5	72.7	71.7	63.3	63.4	63.5	64.6	64.4	63.1	63.1	63.1	63.3	63.3
			0.95	63.7	64.2	64.7	65.2	65.3	63.5	63.7	64.1	64.4	64.5	63.2	63.3	63.5	63.6	63.7	63.1	63.1	63.1	63.2	63.2	63.1	63.1	63.1	63.1	63.1
	0.25	μ_s/μ_f	0.01	99.8	99.9	99.9	99.0	95.5	98.4	99.1	99.3	98.6	95.1	93.3	95.8	97.1	96.9	93.3	74.6	78.9	82.5	83.8	79.6	66.6	67.5	68.4	67.7	64.1
			0.02	99.8	99.9	99.9	99.2	96.0	98.4	99.1	99.4	98.8	95.6	93.2	95.8	97.1	97.0	93.7	74.5	78.8	82.3	83.7	79.8	66.5	67.5	68.4	67.7	64.3
			0.1	99.8	99.9	99.9	100.0	99.2	98.2	99.0	99.4	99.5	98.6	92.3	95.2	96.9	97.5	96.2	73.5	77.5	81.2	83.1	80.5	66.3	67.2	68.0	67.7	65.2
			0.2	99.8	99.9	99.9	99.9	99.9	97.7	98.7	99.2	99.4	99.4	90.8	94.2	96.1	97.2	96.9	72.1	75.8	79.3	81.6	80.1	66.1	66.8	67.6	67.4	65.8
			0.5	99.3	99.6	99.7	99.8	99.4	94.2	96.4	97.6	98.4	96.8	82.4	87.1	90.6	93.0	90.1	67.7	69.6	72.0	74.2	71.8	65.3	65.5	65.9	66.3	65.5
			0.8	95.0	96.6	97.4	98.1	97.9	76.9	80.5	83.1	87.2	87.0	66.7	67.7	68.8	73.8	74.7	65.0	65.0	65.0	65.9	66.2	64.9	64.9	64.9	65.0	65.1
			0.95	68.3	70.2	71.8	72.2	71.6	65.6	66.1	66.7	67.0	67.4	65.1	65.2	65.3	65.4	65.7	64.9	65.0	65.0	65.0	65.0	64.9	64.9	64.9	64.9	65.0

JFIQ-d **JIFQ-d** **JIQ$^{\text{BR}}$-d** **JIQ$^{\text{FP}}$-d**

Table 10: The best policy and its percentage improvement compared to **RND** . $d = 3$

		α_s		0.02					0.2					1					10					50				
		α_f/α_s		0.05	0.1	0.2	0.5	1	0.05	0.1	0.2	0.5	1	0.05	0.1	0.2	0.5	1	0.05	0.1	0.2	0.5	1	0.05	0.1	0.2	0.5	1
ρ	0.1	μ_s/μ_f	0.01	99.6	99.8	99.9	99.9	99.6	95.9	97.8	98.7	98.7	97.0	82.6	89.7	93.5	94.4	90.3	37.0	49.5	61.0	68.8	65.3	17.3	22.4	28.9	35.5	34.3
			0.02	99.5	99.7	99.8	99.9	99.7	95.4	97.5	98.5	98.8	97.7	81.1	88.7	93.0	94.5	91.3	36.0	48.4	59.9	68.1	65.0	17.1	22.2	28.6	35.1	34.0
			0.1	87.1	89.9	91.0	90.3	86.2	74.0	82.5	87.2	88.8	85.1	57.1	69.8	78.9	84.0	81.3	28.6	39.2	50.5	60.5	59.2	15.9	20.3	26.0	32.1	31.8
			0.2	33.8	45.8	57.7	68.2	68.6	33.2	45.1	57.1	67.7	68.1	31.2	42.7	54.6	65.5	66.1	22.1	30.1	39.8	50.2	50.6	14.7	18.2	23.0	28.8	28.8
			0.5	14.8	18.6	24.2	32.6	35.7	14.7	18.5	24.2	32.5	35.6	14.6	18.4	23.9	32.0	35.0	13.8	16.8	21.3	27.9	29.9	12.1	13.8	16.4	20.0	20.6
			0.8	11.1	12.1	13.6	16.5	18.0	11.1	12.1	13.6	16.5	17.9	11.1	12.0	13.6	16.4	17.8	11.0	11.8	13.2	15.5	16.5	10.6	11.2	12.1	13.4	13.8
			0.95	10.2	10.4	10.8	11.4	11.8	10.2	10.4	10.8	11.4	11.8	10.2	10.4	10.7	11.4	11.7	10.2	10.4	10.7	11.2	11.5	10.1	10.3	10.5	10.8	10.9
	0.25	μ_s/μ_f	0.01	99.7	99.8	99.9	99.9	96.0	96.8	98.2	98.9	98.9	95.0	86.1	91.8	94.9	95.5	91.4	47.6	58.0	67.4	73.5	70.0	30.7	34.8	39.9	44.2	42.3
			0.02	99.6	99.8	99.9	99.9	99.8	96.6	98.2	98.9	99.1	98.2	85.6	91.5	94.8	95.9	93.5	47.1	57.5	67.0	73.3	70.2	30.6	34.6	39.6	44.0	42.2
			0.1	99.4	99.7	99.8	99.9	99.8	94.4	96.9	98.1	98.6	98.0	79.1	87.0	91.7	94.1	92.5	42.6	52.1	61.6	69.0	67.1	29.7	33.2	37.7	41.8	41.0
			0.2	97.7	98.5	98.6	94.0	86.9	84.3	89.6	92.1	90.7	85.4	63.1	73.4	80.5	83.8	80.8	37.3	44.9	53.4	60.9	59.5	28.7	31.6	35.4	39.4	38.9
			0.5	30.3	34.4	39.7	45.7	46.6	30.3	34.3	39.5	45.6	46.4	30.0	33.9	38.9	44.8	45.7	28.4	31.2	35.1	39.8	40.2	26.5	27.9	30.0	32.5	32.5
			0.8	25.7	26.5	27.8	29.7	30.3	25.7	26.5	27.8	29.7	30.3	25.7	26.5	27.7	29.6	30.2	25.5	26.2	27.3	28.8	29.2	25.2	25.7	26.3	27.3	27.4
			0.95	24.9	25.0	25.3	25.7	25.8	24.9	25.0	25.3	25.7	25.8	24.9	25.0	25.3	25.7	25.8	24.9	25.0	25.2	25.6	25.7	24.8	24.9	25.1	25.3	25.3
	0.50	μ_s/μ_f	0.01	99.8	99.9	99.9	99.5	95.1	97.7	98.8	99.2	98.9	94.6	90.2	94.2	96.3	96.5	92.5	62.6	69.8	76.1	79.7	76.3	50.4	52.8	55.6	56.9	55.1
			0.02	99.8	99.9	99.9	99.9	97.2	97.7	98.7	99.2	99.4	96.6	90.0	94.1	96.3	97.0	94.1	62.4	69.6	75.9	79.8	76.6	50.3	52.7	55.5	56.9	55.2
			0.1	99.7	99.8	99.9	99.9	99.9	97.0	98.3	99.0	99.3	99.3	87.7	92.5	95.3	96.8	96.4	60.2	67.0	73.5	77.8	75.6	49.8	51.9	54.2	56.4	55.3
			0.2	99.5	99.7	99.8	99.9	99.8	95.5	97.4	98.4	98.8	98.5	83.4	89.3	92.9	94.7	93.7	57.1	62.9	68.8	72.8	70.6	49.2	50.9	53.2	55.3	54.6
			0.5	76.5	78.9	78.9	76.0	71.7	65.7	71.4	74.7	74.4	70.8	57.5	62.8	67.5	69.9	67.8	50.0	52.4	55.3	58.1	57.8	47.9	48.7	50.0	51.3	51.1
			0.8	47.6	48.3	49.4	50.6	50.9	47.6	48.3	49.3	50.6	50.8	47.5	48.2	49.2	50.4	50.5	47.3	47.7	48.4	49.2	49.3	47.0	47.3	47.6	48.1	48.0
			0.95	46.8	46.9	47.0	47.2	47.2	46.8	46.9	47.0	47.2	47.2	46.8	46.9	47.0	47.1	47.2	46.8	46.8	46.9	47.1	47.1	46.7	46.8	46.9	46.9	46.9
	0.75	μ_s/μ_f	0.01	99.8	99.9	99.9	99.5	96.9	98.4	99.1	99.5	99.1	96.5	93.2	95.9	97.3	97.4	94.8	73.9	78.6	82.6	84.6	81.4	65.5	66.8	68.2	68.4	65.8
			0.02	99.8	99.9	99.9	99.9	97.7	98.4	99.1	99.5	99.5	97.3	93.1	95.8	97.3	97.7	95.5	73.8	78.5	82.5	84.5	81.6	65.4	66.7	68.1	68.5	66.0
			0.1	99.8	99.9	99.9	100.0	100.0	98.0	98.9	99.3	99.5	99.5	91.9	95.0	96.8	97.7	97.5	72.5	76.9	80.9	83.5	81.7	65.2	66.4	67.8	68.5	66.9
			0.2	99.7	99.8	99.9	99.9	99.9	97.4	98.5	99.1	99.3	99.3	89.9	93.5	95.7	96.9	96.7	70.9	74.8	78.8	81.8	80.8	64.9	66.0	67.3	68.2	67.2
			0.5	99.0	99.4	99.6	99.7	99.5	92.2	95.1	96.6	97.2	96.1	79.2	84.3	88.1	90.1	88.5	66.5	68.5	71.0	73.2	72.7	64.2	64.7	65.5	66.2	65.9
			0.8	67.3	69.4	71.3	72.0	70.7	66.4	68.1	69.9	71.0	69.9	65.1	66.3	67.6	68.7	68.1	63.9	64.3	64.8	65.2	64.9	63.6	63.7	63.9	64.1	63.9
			0.95	63.5	63.6	63.7	63.9	63.9	63.5	63.6	63.7	63.9	63.9	63.5	63.6	63.7	63.8	63.8	63.5	63.5	63.5	63.6	63.6	63.5	63.5	63.5	63.5	63.5
	0.90	μ_s/μ_f	0.01	99.9	99.9	100.0	99.7	98.4	98.7	99.3	99.5	99.4	98.0	94.5	96.6	97.7	98.0	96.4	79.0	82.6	85.7	87.0	84.4	72.3	73.2	74.0	73.5	70.4
			0.02	99.9	99.9	100.0	99.9	98.7	98.7	99.3	99.5	99.6	98.4	94.4	96.6	97.7	98.1	96.7	78.9	82.5	85.6	87.0	84.5	72.3	73.2	74.0	73.5	70.6
			0.1	99.8	99.9	99.9	100.0	100.0	98.5	99.1	99.5	99.6	99.6	93.6	96.1	97.5	98.2	97.9	78.1	81.6	84.7	86.6	84.9	72.2	73.0	73.8	73.6	71.5
			0.2	99.8	99.9	99.9	100.0	100.0	98.1	98.9	99.3	99.5	99.5	92.3	95.2	96.9	97.8	97.7	77.0	80.2	83.3	85.5	84.4	72.0	72.7	73.4	73.5	72.1
			0.5	99.4	99.7	99.8	99.8	99.8	95.1	97.0	98.0	98.5	98.1	85.4	89.4	92.3	93.9	92.6	73.5	75.3	77.3	78.7	77.2	71.4	71.7	72.1	72.3	71.7
			0.8	94.8	96.6	97.6	97.6	95.4	80.3	83.9	86.8	88.1	85.8	73.4	75.1	77.1	79.0	77.9	71.1	71.3	71.6	72.4	72.1	70.9	71.0	71.0	71.2	71.1
			0.95	71.5	71.8	72.3	72.7	72.7	71.2	71.5	71.8	72.0	72.1	71.0	71.1	71.3	71.4	71.4	70.9	71.0	71.0	71.0	71.0	70.9	70.9	70.9	70.9	71.0
	0.95	μ_s/μ_f	0.01	99.9	99.9	100.0	99.8	98.9	98.8	99.3	99.6	99.5	98.5	94.9	96.8	97.8	98.2	97.0	80.5	83.8	86.6	87.8	85.4	74.3	75.0	75.6	74.8	71.6
			0.02	99.9	99.9	100.0	99.9	99.1	98.8	99.3	99.6	99.6	98.8	94.8	96.8	97.8	98.2	97.2	80.4	83.7	86.5	87.8	85.4	74.3	75.0	75.6	74.8	71.8
			0.1	99.9	99.9	100.0	100.0	100.0	98.6	99.2	99.5	99.7	99.6	94.1	96.4	97.6	98.3	98.0	79.6	82.8	85.6	87.2	85.6	74.1	74.8	75.4	74.9	72.7
			0.2	99.8	99.9	99.9	100.0	100.0	98.2	99.0	99.4	99.6	99.6	92.9	95.6	97.1	97.9	97.8	78.6	81.5	84.3	86.0	84.8	73.9	74.5	75.0	74.8	73.2
			0.5	99.5	99.7	99.8	99.9	99.8	95.6	97.3	98.3	98.7	98.3	86.6	90.3	93.0	94.7	93.4	75.3	76.9	78.8	80.3	78.8	73.3	73.6	73.9	74.1	73.5
			0.8	96.3	97.6	98.3	98.7	98.4	82.9	86.2	88.9	91.3	90.4	74.9	76.0	77.9	81.3	80.9	73.1	73.2	73.2	74.2	74.1	73.0	73.0	73.0	73.2	73.2
			0.95	75.7	77.2	78.5	78.9	78.2	73.6	74.0	74.5	74.9	75.0	73.2	73.3	73.4	73.5	73.7	73.1	73.1	73.1	73.1	73.1	73.0	73.0	73.0	73.0	73.1

JFIQ-d **JIFQ-d** **JIQ$^{\text{BR}}$-d** **JIQ$^{\text{FP}}$-d**

Table 11: The best policy and its percentage improvement compared to **RND** . $d = 4$

		α_s		0.02					0.2					1					10					50				
		α_f/α_s		0.05	0.1	0.2	0.5	1	0.05	0.1	0.2	0.5	1	0.05	0.1	0.2	0.5	1	0.05	0.1	0.2	0.5	1	0.05	0.1	0.2	0.5	1
ρ	0.1	μ_s/μ_f	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.02	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.95	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.25	μ_s/μ_f	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.02	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.95	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.5	μ_s/μ_f	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.02	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	1.0	0.0	0.0	0.0	0.0	0.0
			0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.95	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0
	0.75	μ_s/μ_f	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.02	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0
			0.2	0.0	0.0	0.0	0.3	3.6	0.0	0.0	0.0	0.2	3.5	0.0	0.0	0.0	0.0	3.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.8	0.3	0.4	0.3	0.0	1.5	0.3	0.4	0.3	0.0	1.4	0.2	0.4	0.3	0.0	1.3	0.2	0.3	0.2	0.0	0.8	0.1	0.2	0.1	0.0	0.3
			0.95	0.0	0.1	0.1	0.2	0.1	0.0	0.1	0.1	0.2	0.1	0.0	0.1	0.1	0.1	0.1	0.0	0.0	0.1	0.1	0.1	0.0	0.0	0.0	0.1	0.0
	0.9	μ_s/μ_f	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.02	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.1	0.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.9	0.0	0.0	0.0	0.0	0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.5	0.7	1.0	0.8	0.8	24.9	0.6	0.9	0.8	0.8	23.0	0.6	0.8	0.6	0.9	17.9	0.3	0.4	0.2	0.5	5.8	0.2	0.2	0.2	0.1	1.1
			0.8	3.6	6.0	8.2	3.4	3.0	3.5	6.0	8.0	3.3	2.8	3.4	5.6	7.4	2.7	2.2	1.0	1.7	2.8	0.9	0.8	0.3	0.5	0.8	0.3	0.2
			0.95	0.2	0.3	0.5	0.8	0.2	0.2	0.3	0.5	0.8	0.2	0.1	0.3	0.4	0.6	0.1	0.1	0.1	0.2	0.2	0.1	0.0	0.0	0.1	0.1	0.0
	0.95	μ_s/μ_f	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.02	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.1	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.2	0.1	0.0	0.1	0.5	0.0	0.0	0.0	0.3	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.2	0.1
			0.5	3.3	5.6	7.3	0.4	70.4	3.2	5.3	6.7	0.2	44.1	2.7	4.3	5.0	0.0	23.4	1.2	1.7	1.6	0.0	5.3	0.5	0.7	0.6	0.0	0.9
			0.8	10.7	18.9	28.5	19.7	6.6	10.5	18.3	27.0	17.0	5.2	8.0	13.2	19.1	11.1	3.1	1.4	2.4	3.7	2.5	0.8	0.3	0.6	0.9	0.7	0.2
			0.95	0.8	1.4	2.2	3.1	0.4	0.6	1.1	1.7	2.4	0.3	0.4	0.6	1.0	1.4	0.2	0.1	0.1	0.2	0.3	0.1	0.0	0.0	0.1	0.1	0.0

JFIQ-d **JIFQ-d** **JIQ$^{\mathbf{BR}}$-d** **JIQ$^{\mathbf{FP}}$-d**

Table 12: The percentage improvement with **JIFQ$^{\mathbf{OPT}}$** over the best non-optimization heuristic policy $d = 3$

		α_s		0.02					0.2					1					10					50				
		α_f/α_s		0.05	0.1	0.2	0.5	1	0.05	0.1	0.2	0.5	1	0.05	0.1	0.2	0.5	1	0.05	0.1	0.2	0.5	1	0.05	0.1	0.2	0.5	1
ρ	0.1	μ_s/μ_f	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.02	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.95	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.25	μ_s/μ_f	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.02	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.95	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.5	μ_s/μ_f	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0
			0.02	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.9	0.0	0.0	0.0	0.0	0.0
			0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.95	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.75	μ_s/μ_f	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.02	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.2	0.0	0.0	0.0	0.0	0.8	0.0	0.0	0.0	0.0	0.7	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.8	0.0	0.0	0.0	0.2	1.1	0.0	0.0	0.0	0.2	1.1	0.0	0.0	0.0	0.2	1.1	0.0	0.0	0.0	0.2	0.7	0.0	0.0	0.0	0.1	0.3
			0.95	0.0	0.0	0.1	0.0	0.1	0.0	0.0	0.1	0.0	0.1	0.0	0.0	0.1	0.0	0.1	0.0	0.0	0.1	0.0	0.1	0.0	0.0	0.0	0.0	0.0
	0.9	μ_s/μ_f	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.02	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.1	0.0	0.0	0.0	0.0	4.7	0.0	0.0	0.0	0.0	3.4	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.5	0.2	0.2	0.0	1.7	13.5	0.2	0.2	0.0	1.7	12.9	0.2	0.1	0.0	1.7	10.7	0.1	0.0	0.0	1.0	4.3	0.1	0.0	0.0	0.2	1.0
			0.8	2.0	3.0	3.3	0.6	1.6	2.0	3.0	3.2	0.5	1.5	1.9	2.8	3.0	0.5	1.3	0.8	1.5	1.7	0.2	0.6	0.3	0.5	0.7	0.1	0.2
			0.95	0.1	0.2	0.3	0.5	0.1	0.1	0.2	0.3	0.5	0.1	0.1	0.2	0.3	0.4	0.1	0.0	0.1	0.1	0.2	0.1	0.0	0.0	0.1	0.1	0.0
	0.95	μ_s/μ_f	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.02	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
			0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.1
			0.2	0.0	0.0	0.2	0.7	1.2	0.0	0.0	0.1	0.4	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.2	0.0	0.0	0.0	0.2	0.6
			0.5	1.8	0.0	2.8	0.0	22.3	1.8	2.6	2.6	0.0	18.4	1.5	2.2	1.9	0.1	12.0	0.7	0.9	0.6	0.1	3.5	0.3	0.4	0.2	0.0	0.6
			0.8	7.1	11.6	15.1	6.6	2.4	7.0	11.3	14.5	6.0	2.1	6.1	10.2	12.4	4.3	1.5	1.2	2.2	3.4	1.1	0.5	0.3	0.6	0.9	0.3	0.1
			0.95	0.4	0.7	1.2	1.8	0.1	0.4	0.7	1.1	1.5	0.1	0.3	0.5	0.7	1.0	0.1	0.1	0.1	0.2	0.3	0.0	0.0	0.0	0.1	0.1	0.0

JFIQ-d **JIFQ-d** **JIQ$^{\text{BR}}$-d** **JIQ$^{\text{FP}}$-d**

Table 13: The percentage improvement with **JIFQ$^{\text{OPT}}$** over the best non-optimization heuristic policy $d = 4$