Learning and Hierarchies in Service Systems

Kostas Bimpikis
Graduate School of Business, Stanford University.

Mihalis G. Markakis
Department of Economics and Business, Universitat Pompeu Fabra and Barcelona School of Management.

Motivated by diverse application areas such as healthcare, call centers, and crowdsourcing, we consider the design and operation of service systems that process tasks with types that are ex ante unknown, and employ servers with different skill sets. Our benchmark model involves two types of tasks, "Easy" and "Hard," and servers that are either "Junior" or "Senior" in their abilities. The service provider determines a resource allocation policy, i.e., how to assign tasks to servers over time, with the goal of maximizing the system's long-term throughput. Information about a task's type can only be obtained while serving it. In particular, the more time a Junior server spends on a task without service completion, the higher her belief that the task is Hard and, thus, needs to be rerouted to a Senior server. This interplay between service time and task-type uncertainty implies that the system's resource allocation policy and staffing levels implicitly determine how the provider prioritizes between learning and actually serving. We show that the performance loss due to the uncertainty in task types can be significant and, interestingly, the system's stability region is largely dependent on the rate at which information about the type of a task is generated. Furthermore, we consider endogenizing the servers' capabilities: assuming that training is costly, we explore the problem of jointly optimizing over the training levels of the system's server pools, the staffing levels, and the resource allocation policy. We find that among optimal designs there always exists one with a "hierarchical" structure, where all tasks are initially routed to the least skilled servers and then progressively move to more skilled ones, if necessary. Comparative statics indicate that uncertainty in task types leads to significantly higher staffing costs and less specialized server pools.

Key words: Service systems, unknown types, learning, stability region, service hierarchies.

1. Introduction

Service systems are typically designed to process tasks that may be heterogeneous in terms of their service requirements. Additionally, the servers themselves may differ in their skills, e.g., the ability to complete a given task's service requirements. Finally, and quite importantly, although the service provider may have a relatively accurate prior belief about the set of attributes describing an arriving task through historical data, these attributes, i.e., the exact *type* of a given task, may be unknown to her ex ante, and information about them can only be obtained gradually, after the task is assigned to service.

An example that falls within this general framework is the design and operation of a healthcare system. Healthcare professionals that are heterogeneous in their abilities, e.g., attending physicians, residents/fellows, and registered nurses/physician assistants/health coaches, are employed to treat patients that may suffer from a variety of conditions, which, originally, are unknown to the healthcare team. In many cases, important information about a given patient's condition arrives gradually, and only after the patient is assigned to a healthcare professional. Finally, while patients suffering from mild conditions can be treated by less skilled staff, e.g., nurse practitioners, acute cases would require consulting with attending physicians. An optimal allocation of the system's resources, which would ensure that complex cases are actually assigned to highly skilled healthcare professionals, is hindered by the ex ante uncertainty in the pathology of incoming patients. This ties together the processes of diagnosis and treatment for a patient. Similar tradeoffs may arise in crowdsourcing systems or call centers, with incoming tasks/customers having service requirements of varying complexity, ex ante unknown, and workers/agents that are heterogeneous in their skill sets.[1]

Our benchmark model concerns a service system with two types of tasks, "Easy" and "Hard," and two types of servers, "Junior" and "Senior." Hard tasks can only be processed by Senior servers, whereas an Easy task can be processed by any server. Although a probabilistic prior is available regarding the type of an arriving task, the actual type of the task is ex ante *unknown*. The service provider's objective is to maximize the system's stability region, i.e., the highest arrival rate of tasks that the system can support, by choosing the system's resource allocation policy, i.e., how to allocate incoming tasks to the two server pools and, potentially, when to reroute tasks from Junior to Senior servers. Note that in this setting, the two main functions of the service system, learning a task's type and fulfilling its service requirements, are intertwined. More specifically, the more time a Junior server spends on a task without service completion, the higher her belief that the task is Hard and, thus, needs to be rerouted to a Senior server.

The main goal of the paper is to understand how ex ante uncertainty in task types affects both the fundamental limits of performance of a service system, in particular its stability region, as well as its optimal operational structure, characterized by its staffing levels and resource allocation policy. Our contributions, in this respect, can be summarized as follows.

(i) We establish that the performance loss, i.e., the reduction in the system's stability region resulting from the uncertainty in task types can be significant and, furthermore, it depends on the rate at which servers generate information about the types. In the benchmark case, where the

[1] Another important feature of crowdsourcing systems is the incentives of (self-interested) workers, which we do not capture in this paper. Our modeling framework can, however, be employed to derive and study the "first-best" solution to the underlying resource allocation problem.

only information available to a server in order to determine a task's type is the time it has been in service, we show that the performance loss is greater for service time distributions with heavy tails. It may be worthwhile to note that this is in contrast to the case of known types, where the system's stability region depends only on the first moments of service times, irrespective of any other characteristic of their distributions, e.g., variance, skewness, tail. This result is particularly relevant, given the ample support for heavy-tailed service time distributions in many settings of interest, e.g., in healthcare (Armory et al. (2015)) and call centers (Brown et al. (2005)).

(ii) We extend our framework to incorporate staffing decisions. We show that the staffing cost required to sustain a given arrival rate is significantly higher in the presence of task-type uncertainty and, as before, it depends on the rate at which servers generate information about the types.

(iii) When the service provider jointly optimizes over staffing levels and the system's resource allocation policy, an optimal system design takes the form of a *service hierarchy*: all tasks are initially routed to the Junior server pool, and a task is rerouted to the Senior server pool only after the time it has spent with a Junior server exceeds an appropriately determined threshold. This threshold is a function of the rate of learning the tasks' types, and of the staffing levels.

The second part of the paper introduces a more general service system model with a continuum of task types, and servers whose skills are not exogenous but, rather, can be determined by the service provider via training. In particular, servers are trained up to a level, which, subsequently, determines the subset of tasks whose service requirements they can fulfill. We show that when training is costly, an optimal system design takes again the form of a hierarchy, with all tasks routed progressively from less to more skilled servers. Uncertainty in task types results in much higher training/staffing costs and a larger number of highly skilled servers relative to the case of known types. Taken together, these results provide a qualitative understanding on how to design, staff, and train service systems in the presence of task-type uncertainty.

We conclude the analysis by providing a number of extensions to our modeling framework. First, we consider a setting where the service provided to an Easy task is fully transferable, i.e., if the task is rerouted, a Senior server only needs to provide the remaining service instead of starting from the beginning. In contrast, any time spent on a Hard task by a Junior server does not contribute towards the task's service completion, as such a task is outside the server's skill set. We show that, qualitatively, the performance of the system in this setting is similar to the benchmark case. Second, we extend our analysis to the case where service takes the form of a sequence of potentially conclusive tests, so that the belief of a server about a task's type is piecewise constant. Third, we incorporate a setup component in the service of every task, in order to capture the distinction between common and type-specific components of service. We establish that if the time required to complete the setup component is short relative to the actual service time, then, as in the main part

of our analysis, some performance loss due to task-type uncertainty is unavoidable. On the other hand, if the setup time is comparable to the mean service time, the service provider can achieve the same performance as in the case of known types by having Junior servers specialize in serving only the setup component, and then rerouting tasks to Senior servers.

Finally, we discuss how uncertainty in task types may affect the average time that tasks spend in the system. Although a rigorous analysis is beyond the scope of the present paper, we provide evidence that it may lead to a considerable increase in the average delay, in part due to the reduction in the system's stability region.

Related Literature. To put our modeling framework in perspective, we note that it does not fall into any of the widely studied classes of queueing systems, e.g., *flexible server systems* (Mandelbaum and Stolyar (2004)); *switched queueing networks* (Tassiulas and Ephremides (1992)); or the very broad family of *stochastic processing networks*, which subsumes the two previous classes (Harrison (2000) and Dai and Lin (2005)). In all these models, the type of a task is known upon its arrival to the system, which ensures that the task is eventually processed only by servers that can fulfill its service requirements. Thus, there is no need for learning, no misallocation of the system's resources and, inevitably, no performance loss due to task-type uncertainty—all of which are central features of our model. As we establish in the paper, the performance and optimal design of service systems in the presence of task-type uncertainty are considerably different than the case where the service provider knows the primitives describing each incoming task.

Closer in spirit to our work is the burgeoning literature that explores the role of diagnostic systems in service environments that feature uncertainty. In an early, influential work, Shumsky and Pinker (2003) focuses on the misalignment of incentives between an informed agent, the *gatekeeper*, and an uninformed principal, who offers a contract to the agent. The gatekeeper assesses an incoming customer's problem complexity, and can either decide to solve the problem herself and earn revenue while incurring the cost of effort, or refer the customer to a specialist. Follow-up work, e.g., Hasija et al. (2005) and Lee et al. (2012), considerably expands our understanding of outsourcing in a service context, addressing questions regarding optimal referral rates and staffing levels. In contrast, we consider the design of a system with a single decision maker, and show how it is affected by both the ex ante uncertainty in task types, and the stochastic variability in service times. Unlike Shumsky and Pinker (2003), we explicitly take the latter into account, as it turns out to have significant implications on the service provider's decision making.

Alizamir et al. (2013) considers a single-server system that is tasked with classifying a stream of arriving customers into one of two groups. Diagnostic tests are costly as they take processing time from the server, but they reduce the probability of misclassification. The paper focuses on the trade-off between classification accuracy and the waiting cost incurred by the incoming customers. Wang

et al. (2010), in a related setting, explores the interplay between the capacity of a service system and the congestion that it induces. In another interesting contribution, Paç and Veeraraghavan (2015) studies how the asymmetry of information between expert and customer may result in overtreatment in a healthcare environment. More recently, Massoulié and Xu (2017) explores the stability region of information processing systems that feature tests which may differ in their classification accuracy, depending on the types of the task and the server.

Sun et al. (2014) and Levi et al. (2016) explore optimal scheduling policies in a single-server setting that can run a (costly) test to determine, conclusively, a customer's type. Unlike Alizamir et al. (2013), they model explicitly the service process that follows the classification. Dobson and Sainathan (2011) and Dobson et al. (2013) also consider interesting variations of this framework, focusing mainly on static design questions; this is in contrast to Sun et al. (2014) and Levi et al. (2016) that provide characterizations of optimal dynamic scheduling policies.

We complement the aforementioned literature by taking the service provider's point of view, and by focusing on the optimal allocation of workflow among servers with different skills. Unlike prior work in which there is an explicit dichotomy between diagnostic testing to determine a customer's type and then scheduling for service, the two functions of learning and serving are intertwined in our model. Furthermore, unlike diagnostic tests that provide a binary label for a task instantaneously, which may or may not be conclusive, information in our setting arrives gradually over time.[2] These features lead to qualitative insights that relate the rate at which information is generated during service to the performance of the system. Finally, having heterogeneous server pools allows us to consider endogenizing the servers' training levels, and jointly optimize over their capabilities, the staffing levels, and the system's resource allocation policy.

Our study also has some high-level similarities to several other papers, although the underlying models are quite different. Netessine et al. (2002) considers a firm that invests in dedicated and flexible capacity before the demand is realized. The authors obtain an analytical solution in the relevant case where services can be upgraded by a class, and derive crisp results on the effects of demand correlation on the optimal mix of flexible and dedicated resources. Bassamboo et al. (2010) considers optimal staffing decisions when the arrival rate of a stream of homogeneous customers is uncertain, but drawn from a known distribution. The paper provides an elegant modeling framework to establish that, depending on whether the uncertainty in the customers' arrival rate or the inherent stochastic variability of service times dominates, staffing decisions need to follow different prescriptions. Gurvich and Van Mieghem (2014) studies stochastic processing networks in

[2] One can view our approach as a continuous-time analog of a sequence of diagnostic tests, each of which has two possible outcomes: if successful, the service comes to an end and the task departs the system; if unsuccessful, the server finds it (infinitesimally) more likely that the task's requirements cannot be fulfilled with her skill set.

the presence of constraints imposed by the requirement of simultaneous collaboration of multiple resources. Such architectures may feature unavoidable idleness, and lead to a discrepancy between the system's theoretical capacity and what is actually achievable by a resource allocation policy. In our model, such performance loss may be the result of the ex ante uncertainty in task types.

Our finding that the optimal design takes the form of a service hierarchy is reminiscent of research on optimal organizational design, e.g., Garicano (2000). Relatedly, Mihm et al. (2010) examines the interaction between the process of search within an organization, e.g., for a new product, with its hierarchical structure. The paper provides compelling analytical and simulation results that shed light on the potential tradeoff between the stability of the search process, which hierarchies tend to favor, and the quality of the end solution. More recently, Acemoglu et al. (2016) explores a related environment motivated by crowdsourcing applications, and provides a pricing mechanism that implements the optimal resource allocation policy when the servers (problem solvers) are self-interested. Given that the authors' main focus is on the servers' incentives, their model assumes that a server can instantaneously determine whether she has the skills required to serve a task. Additionally, Stouras et al. (2016) considers a service setting motivated by two-sided marketplaces and work-from-home call centers, establishing that it is optimal for the service provider to rank agents in a pre-specified number of priority classes based on past performance.

Finally, our paper contributes to the literature that explores settings where the value of service increases with the time a customer spends with a server, and the resulting "quality-speed" tradeoff. Representative contributions include Anand et al. (2011) and Kostami and Rajagopalan (2013).

2. A Service System with Unknown Task Types

We consider service environments with the following features: heterogeneity in the types of incoming tasks, which translates in heterogeneity in their service requirements; servers that differ in their ability to serve tasks of different types; and ex ante uncertainty in the tasks' types. In the context of our modeling framework, serving a task may involve a component that is common among tasks, e.g., setup/preprocessing activities, and a type-specific component. All servers can perform the common component whereas only a subset of the servers, the highly skilled ones, may be able to perform the type-specific component for some types, e.g., the more complex tasks. Our analysis focuses precisely on the interplay between the uncertainty in the service requirements of tasks, i.e., their types, and the heterogeneity in the servers' skills.

2.1. The Benchmark Model

The first part of the paper studies a stylized benchmark model with two server pools and two task types. Server pool J consists of n_j "Junior" servers, while server pool S consists of n_s "Senior" servers. Servers within each pool are homogeneous in their abilities. Tasks waiting to be served by

Junior/Senior servers are queued up in a "first come, first served" manner in queue J/S. As soon as a server becomes available, she picks the task at the head of the line in her respective queue.

Tasks arrive to the system according to a continuous time arrival process with rate λ, and interarrival times that are i.i.d. Arriving tasks can be of two types, "Easy" with probability $p_0 \in (0,1)$ and "Hard" otherwise, independently of the past history of the system. Server pools are heterogeneous in their skills, i.e., Junior servers can only serve Easy tasks whereas Senior servers can serve both types of tasks. In other words, for our benchmark model we assume that there is no common service component, i.e., we allow only for type-specific components; we relax this assumption in Section 4.3. The service times of Easy (Hard) tasks when served by a Senior server are independent random variables, distributed identically to some random variable E (H), with an atomless density $f_E(\cdot)$ $\big(f_H(\cdot)\big)$ supported on $\mathcal{D} \subset [0,\infty)$, respectively. For the model to be meaningful, we assume that $\mathcal{D} \neq \emptyset$ and $\mathbb{E}[E],\ \mathbb{E}[H] < \infty$. Junior servers can serve Easy tasks as efficiently as Senior servers, i.e., the service times of Easy tasks when served by a Junior server are again i.i.d. random variables with density $f_E(\cdot)$, while the service time of a Hard task when served by a Junior server is infinite.[3]

The distinguishing characteristic of our benchmark model is that the type of a task is unknown to the service provider upon its arrival to the system, so the only information available ex ante is the prior p_0. To best illustrate the impact of this uncertainty on the system's performance, we assume that the same prior applies to all incoming tasks, i.e., although tasks may have different types, they look the same upon arrival. As we argue in Section 3, our model can be extended in a straightforward way to incorporate tasks that have ex ante different priors.

An important primitive of our benchmark model, related to the uncertainty in task types, is function $\pi : [0,\infty) \to [0,p_0]$, representing the *posterior belief* of a Junior server[4] that a task is Easy, as a function of the time she has spent on it without service completion. Initially, the belief that the task is Easy and service can be completed by the Junior server is equal to the prior p_0. As time elapses and the service has not been completed yet, the belief that the task is Easy is likely to decrease. It is precisely the fact that it is difficult to distinguish between a Hard task and an Easy task with a long service time that makes the question of optimal system design challenging.

We assume that $\pi(\cdot)$ is deterministic and known to the service provider, with the following properties: (i) $\pi(0) = p_0$; (ii) $\pi(t) > 0$, for every $t \in \mathcal{D}$, and $\lim_{t\to\infty} \pi(t) = 0$; (iii) $\pi(\cdot)$ is continuous; (iv) $\pi(\cdot)$ is monotonically decreasing.

[3] Our qualitative insights remain valid if we allow both types of servers to be able to serve both types of tasks, as long as the following is true: had the service provider known the types ex ante, she would always assign Hard tasks to Senior servers. This would be the case, for instance, if Senior servers were sufficiently faster, e.g., in a stochastic ordering sense, in serving Hard tasks compared to Junior servers.

[4] Senior servers can serve both types of tasks, so they do not *need* to explicitly distinguish between the two. For that reason, and as a way to simplify the exposition, we do not track Senior servers' beliefs.

While the first assumption is just for consistency with the prior on task types, the rest warrant some brief remarks. The second assumption states that the posterior belief function does not become zero anywhere on the support of the service time distributions, but converges to zero eventually. This is the more interesting regime for us, since in that case the learning problem is meaningful, and the functions of learning and serving are really coupled. The last two assumptions, combined, imply that even an infinitesimal amount of additional service provides information about the type of the task at hand. Thus, although our benchmark model can accommodate a variety of learning patterns, it does not capture settings where the posterior belief function may be flat or discontinuous, e.g., piecewise constant. We revisit this point in Section 4.2. Furthermore, we do not allow for the possibility that the posterior belief increases without the service being completed, i.e., the information generated while a task is in service is either conclusive that the task is Easy, in which case its service requirements are fulfilled and the task departs the system, or indicating that the task is more likely to be Hard. In other words, our setting does not capture the possibility that the generated information is indicative, but not conclusive, of the task being Easy.

The posterior belief function may incorporate a wide range of application-specific information that cannot be captured by the service time distributions. For instance, one can imagine a setting where most of the information regarding a task's type can be obtained during the first few units of time that it is in service. This would correspond to a posterior belief function that drops steeply at the beginning, and decreases at a very slow rate from then on. So, while $\pi(\cdot)$ is explicitly only a function of the elapsed service time, implicitly but equally importantly, it is also a function of the application/context. An important special case that we frequently use as a benchmark is when there is no context-specific information, and deciding on a task's type depends only on the service times; more formally, when the distribution of the residual service time of Easy tasks is a sufficient statistic for determining the posterior belief function. Then $\pi(\cdot)$ is not a primitive per se, but can rather be derived by the aforementioned distribution through the *Bayes rule*:

$$\pi(t) = \frac{p_0 \mathbb{P}(E > t)}{p_0 \mathbb{P}(E > t) + (1 - p_0)}, \qquad t \in [0, \infty). \tag{1}$$

The overall performance of the system depends critically on how fast the service provider can distinguish between Easy and Hard tasks, which, intuitively, is related to the rate at which the posterior belief function decreases in time. For example, in the context of Equation (1), the time that it takes for $\pi(\cdot)$ to fall below a given threshold, suggesting that the task was originally misrouted, is increasing in $\mathbb{P}(E > t)$. This implies that the tail of the distribution of random variable E may have a first-order impact on the system's performance. Note, however, that while learning the task types is an integral part of the service provider's decision making process, it is not the end-goal, as

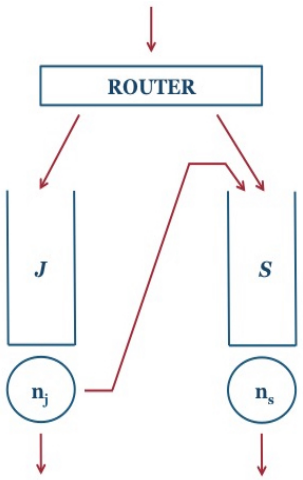

Figure 1 **A graphical representation of the benchmark model.**

her objective is to maximize the system's long-term throughput. We revisit this point in Section 2.3 with a numerical study of the case where the distribution of the residual service time of Easy tasks is a sufficient statistic for determining the posterior belief function. We illustrate that service time distributions with heavy tails, for which distinguishing between Easy and Hard tasks takes relatively longer, lead to smaller stability regions, i.e., greater performance loss.

Let us briefly describe the mechanics of our benchmark model, while a graphical illustration is provided in Figure 1. Upon its arrival to the system, a task reaches the "Router" point, from which it is routed either to queue J or to queue S. If the task is routed to queue S, it waits for the first available Senior server, eventually receives the required service, and then exits the system. If the task is routed to queue J, though, after receiving service for some time, two distinct possibilities arise: either the task exits the system, if it is an Easy task whose service requirements are fulfilled, or the service is preempted and the task is rerouted to queue S. We note that the service does not carry over when a task is rerouted, i.e., its entire service requirements have to be fulfilled by a Senior server. Regardless, the information that the server has generated about the type of the task, which is summarized by her posterior belief, is transferred to Senior servers.

In the above description we have precluded the possibility of rerouting a task from the Senior to the Junior server pool. This seems reasonable given that the posterior belief that a task is Hard is increasing in the time it has spent in service.

As a concluding remark, it may be worthwhile to elaborate on two features of the model described above, and to connect them to the motivating applications of our work, e.g., healthcare, call center, and crowdsourcing operations. The first feature is that the service requirements of a Hard task cannot be fulfilled by a Junior server, so eventually the task has to be rerouted to a Senior server. The second is that, whenever a task is rerouted, the processing steps taken by the Junior server do

not contribute towards the task's service completion by a Senior server, i.e., the service restarts. The rationale behind these assumptions is the following: in practice, different types of servers have different skill sets, and serving a task is equivalent to completing a set of appropriate processing steps. Importantly, the steps required for serving Hard tasks cannot be taken by Junior servers, as they are not part of their skill sets. Therefore, when a Hard task is rerouted from a Junior to a Senior server, a different set of processing steps has to be taken by the Senior server.[5] Consequently, our findings do not rely on information losses during task handoffs, or duplication of processing steps by different servers. Rather, they are driven solely by task-type uncertainty.

2.2. Stability Analysis

In the remainder of the paper we focus on a service provider whose primary objective is to allocate the system's resources in such a way, so as to maximize the system's *stability region*, i.e., the long-term arrival rate of tasks that the system can support.[6] The stability region is a first-order performance metric for any queueing system, and its understanding is necessary before one can proceed to a more refined analysis that could, for example, revolve around minimizing the average delay that tasks experience. Furthermore, the stability region is directly related to the long-term rate at which the service system generates revenue. We provide a characterization of the stability region of the benchmark model as a function of its primitives, i.e., the staffing levels, the distributions of service times, and the posterior belief function. We argue that the performance loss due to the uncertainty in task types can be significant, and it depends on the rate at which servers generate information about the types, as captured by the posterior belief function $\pi(\cdot)$.

Stability Analysis when Task Types are Known. As a benchmark to compare against, we first consider the system described above assuming, however, that task types are *known* upon their arrival. Clearly, Hard tasks are never routed to queue J in this case, and the stability problem reduces to determining how often should Senior servers serve Easy tasks, if at all. It is straightforward to see that the following inequalities are necessary conditions for the system to be stable:

$$\lambda\beta \le n_s, \quad \text{and} \quad \lambda\alpha \le n_j + \big(n_s - \lambda\beta\big),$$

with the shorthand notation $\alpha = p_0\mathbb{E}[E]$ and $\beta = (1-p_0)\mathbb{E}[H]$.

The first inequality states that there should be enough Senior servers to handle the Hard tasks, whereas the second states that the remaining resources should be sufficient to handle the Easy tasks.

[5] Note that in order to simplify the benchmark model and exposition, we also assume that service restarts even when an Easy task is rerouted from a Junior to a Senior server. As we establish in Section 4, this assumption has no bearing in the main findings of the paper.

[6] More formally, the definition of stability that we adopt is what is referred to as "pathwise stability" or "rate stability," i.e., the arrival rate of tasks is less than or equal to their service rate, in the long run, in every queue.

It can be verified that these inequalities are also sufficient for stability, since one can devise either static probabilistic routing policies or dynamic routing policies, e.g., "Join the Shortest Queue," that will stabilize the system as long as these inequalities hold. Thus, we obtain the following result, which enables the comparison to the benchmark model where routing (and rerouting) decisions are made in the presence of uncertainty about task types.

PROPOSITION 1. *The stability region of the benchmark model if task types are known ex-ante, Λ_K, is the set of arrival rates:*

$$\Lambda_K = \left\{ \lambda \geq 0 \;\middle|\; \lambda \leq \min\left\{ \frac{n_s}{\beta}, \frac{n_j + n_s}{\alpha + \beta} \right\} \right\}. \tag{2}$$

Stability Analysis when Task Types are Unknown. Next, we turn our attention to the main focus of the paper, i.e., the case where task types are ex ante unknown. We explore the stability properties of the following parametric class of static policies:

- When a task reaches the Router, it is routed to queue J with probability $q \in [0, 1]$, irrespective of the "state" of the system at that point in time, and to queue S otherwise;

- Each Junior server keeps serving any given task as long as the posterior belief that it is an Easy task is above $p \in [0, p_0]$, irrespective of the "state" of the system during that interval. If the posterior belief falls to p and the service has not been completed, the task is rerouted to queue S.

We use the shorthand notation $S(q, p)$ for such a policy, with parameter values q and p, respectively.

Note that considering this class of resource allocation policies is with some loss of generality. In particular, we do not allow for "dynamic" policies that condition routing and rerouting decisions on information such as the number of tasks in queues J and S, or the amount of service that each task in the system, queued or in service, has received so far. That said, focusing on the class of $S(q, p)$ policies is not particularly restrictive: it can be shown that the stability region achieved by a policy with a state-dependent routing probability (and a static threshold) can also be achieved by some $S(q, p)$ policy. This is because the stability of a queueing system depends on long-term arrival and departure rates, not on short-term load fluctuations. Hence, we believe that for the given objective, a $S(q, p)$ policy with appropriately chosen routing and threshold parameters, serves as a good starting point for studying the impact of task-type uncertainty in service systems.

Under the $S(q, p)$ policy, Hard tasks as well as Easy tasks with long service times that are initially routed to queue J, end up being rerouted to queue S. Since their service has to be restarted, such rerouting implies that the system's resources are wastefully utilized, and this has to lead to a reduction in the system's stability region compared to the case where task types are known. On the other hand, the service provider cannot route all tasks to queue S, as this would leave the pool of Junior servers underutilized, resulting, again, in a reduced stability region. The remainder of the section is aimed at quantifying this tradeoff.

Our analysis proceeds in two steps. First, in Proposition 2, we determine the stability region of the $S(q,p)$ policy, for fixed q and p. Then, Proposition 3 provides an expression for the stability region of the $S(q,p)$ class of policies. The latter result, combined with Proposition 1, illustrates the performance loss resulting from the lack of information on the types of arriving tasks, and motivates a series of comparative statics results that we outline subsequently.

We define the inverse of the posterior belief function, $\phi : [0,p_0] \to [0,\infty)$ as $\phi(p) = \pi^{-1}(p)$. In other words, if p is the posterior belief threshold, a Junior server needs to spend $\phi(p)$ units of time with a task for this threshold to be reached, assuming there is no service completion in the meantime. As $\pi(\cdot)$ is continuous and monotonically decreasing, $\phi(\cdot)$ exists and is continuous and monotonically decreasing as well. Also, to simplify the exposition, we introduce the following notation:

(i) $h(p) = p_0 \int_0^{\phi(p)} x f_E(x) dx$, i.e., the expectation of the time that a task spends with a Junior server multiplied by the indicator of the event that its service requirement is fulfilled under policy $S(q,p)$. The latter is equivalent to the task being Easy and having service time at most $\phi(p)$;

(ii) $g(p) = \phi(p)\big(1 - p_0 + p_0 \mathbb{P}\big(E > \phi(p)\big)\big) + h(p)$, i.e., the expected time that a task spends with a Junior server under the $S(q,p)$ policy. The first term on the right-hand side corresponds to the expected time that a task spends with a Junior server under the $S(q,p)$ policy on the event that its service requirement is not fulfilled.

PROPOSITION 2. *The stability region of the benchmark model under the $S(q,p)$ policy, $\Lambda_{S(q,p)}$, is the set of arrival rates:*

$$\Lambda_{S(q,p)} = \left\{ \lambda \geq 0 \;\middle|\; \lambda \leq \min\left\{ \frac{n_j}{q g(p)}, \frac{n_s}{\alpha + \beta - q h(p)} \right\} \right\}. \tag{3}$$

Proposition 2 highlights the fact that the stability region of the benchmark model depends on the entire service time distribution (of Easy tasks), and on the rate of information acquisition regarding task types. Clearly, the maximum arrival rate that can be supported by the system under a $S(q,p)$ policy is obtained by optimizing over the parameter space (q,p). The following result provides a closed-form expression for this rate, when the service time distributions have unbounded support.

PROPOSITION 3. *Let the support of the service times be $\mathcal{D} = [0,\infty)$. The largest stability region that the benchmark model can obtain under a policy in the $S(q,p)$ class, Λ_S, is the set of arrival rates:*

$$\Lambda_S = \left\{ \lambda \geq 0 \;\middle|\; \lambda \leq \frac{n_j + n_s}{(\alpha+\beta)\left(1 + n_j\big(g(p^*) - h(p^*)\big)/\big(n_j h(p^*) + n_s g(p^*)\big)\right)} \right\}, \tag{4}$$

where p^* is a solution to the optimization problem:

$$\begin{aligned} &\textit{minimize} && \frac{g(p) - h(p)}{n_j h(p) + n_s g(p)} \\ &\textit{subject to} && p \leq \{p_0, p_1\}, \end{aligned}$$

and p_1 is (uniquely) determined by the equation: $n_j h(p_1)+n_s g(p_1)=n_j(\alpha+\beta)$. A policy that obtains the largest stability region is $S\big(i(p^*),p^*\big)$, where

$$i(p)\equiv\frac{n_j(\alpha+\beta)}{n_j h(p)+n_s g(p)}. \tag{5}$$

Comparing the expressions in Propositions 1 and 3 allows us to quantify the performance loss attributed to the lack of information regarding the type of arriving tasks. In particular, assuming that Easy tasks are much more frequent than Hard ones, i.e., $1/p_0 \ll 1/(1-p_0)$, we have that the stability region in the case of known types is the set of arrival rates $\Lambda_K = \big\{\lambda\geq 0 \;\big|\; \lambda\leq (n_s+n_j)/(\alpha+\beta)\big\}$. Hence, the reduction in the stability region that the optimal $S(q,p)$ policy has to incur due to the fact that task types are unknown, is succinctly captured by the term

$$(\alpha+\beta)\cdot\frac{n_j g(p^*)-n_j h(p^*)}{n_s g(p^*)+n_j h(p^*)}, \tag{6}$$

in the denominator of Equation (4). In the special case where $p^*=p_1$, the above term reduces to $\phi(p_1)\big(1-p_0+p_0\mathbb{P}\big(E>\phi(p_1)\big)\big)$, which shows the close relation between the loss in performance and the shape of the service time distribution of Easy tasks. We note that this case is quite relevant as it corresponds to a service system with a "hierarchical" structure, i.e., a system where all tasks are initially routed to queue J. As we show in later sections, when the service provider jointly optimizes over both staffing levels and the resource allocation policy, there always exists an optimal design with such a structure.

2.3. Learning vs. Serving

The numerical study that follows sheds additional light on the effect of uncertainty in the performance of the system. Our goal is to relate the hardness of the learning problem, i.e., how challenging it is to distinguish Easy from Hard tasks, to the reduction in the stability region that results from this uncertainty. Throughout the section, we assume that the distribution of the residual service time of Easy tasks is a sufficient statistic for determining the posterior belief function. This allows us to determine directly the rate at which information about the type of a task is generated by the servers. Note that we make no further assumptions about the functional form of $\pi(\cdot)$, as our interest here is primarily in the maximum stability region of the system, and not in the $S(q,p)$ policy that obtains it. Hence, we can optimize directly over the time threshold $\phi(p)$. The benchmark described in Equation (1), i.e., when $\pi(\cdot)$ is derived by applying the Bayes rule on the service time distribution of Easy tasks, is a special case of this setting.

In particular, we consider a system with m servers in total, $n_j=mp_0$ of which belong to the Junior server pool and the remaining $n_s=m(1-p_0)$ belong to the Senior server pool. The expected

service time of both Easy and Hard tasks is normalized to m, i.e., $\mathbb{E}[E] = \mathbb{E}[H] = m$. The simple staffing heuristic that motivates this setting is that in every m tasks that arrive, we expect to see mp_0 Easy and $m(1-p_0)$ Hard ones. Since the mean service time for both task types is the same, it is reasonable to keep the proportion of server types the same. Analytical results incorporating optimal staffing decisions are presented in the following section. To simplify both the analysis and the exposition, we ignore any integrality constraints. We aim at providing an understanding on how the system's stability region may be affected as the rate at which information is generated by the servers scales with m. In light of our assumption on the form of $\pi(\cdot)$, the latter is closely related to the stochastic variability of service times, and specifically to the tail behavior of the corresponding distributions.

First, note that if task types were known ex ante, the maximum arrival rate that the system could support would be equal to $\lambda_{\max} = \min\{(1-p_0)m/(1-p_0)m, m/m\} = 1$, irrespective of the scaling parameter m or the service time distributions; see Equation (2).

We consider two cases regarding the prior probability of Easy tasks in the numerical experiments that follow: (i) $p_0 = 0.9$; and (ii) $p_0 = 2/3$. For each of these cases, we compute the maximum arrival rate over all $S(q,p)$ policies that can be supported by the system, for all $m \in (1, 100]$, and for three probability distributions that have significantly different tail behavior: an exponential distribution; a power-law distribution, whose tail decays at a polynomial rate; and a lognormal distribution, whose tail decays faster than a power law but slower than an exponential. We emphasize the practical relevance of the latter case, as there is ample evidence of lognormal distributions in a variety of service systems, e.g., healthcare (Armory et al. (2015)), call centers (Brown et al. (2005)), and data centers (Ersoz et al. (2007)).

(a) Exponential Service Times: Service times of Easy tasks are exponentially distributed, with probability density function: $f_E(x) = e^{-x/m}/m$, for $x \geq 0$, so that $\mathbb{E}[E] = m$ and $\mathrm{Var}[E] = m^2$. We have that $\mathbb{P}(E > t) = e^{-t/m}$ and $\int_0^t x f_E(x)dx = m - (m+t)e^{-t/m}$. Therefore,

$$h(p) = p_0 m - p_0\big(m + \phi(p)\big)e^{-\phi(p)/m}, \text{ and}$$

$$g(p) = \phi(p)\Big(1 - p_0 + p_0 e^{-\phi(p)/m}\Big) + p_0 m - p_0\big(m + \phi(p)\big)e^{-\phi(p)/m};$$

(b) Lognormal Service Times: Service times of Easy tasks are lognormally distributed, with probability density function:

$$f_E(x) = \frac{1}{x\sqrt{2\pi \log m}} e^{-\frac{(\log x - \log m/2)^2}{2\log m}}, \qquad x > 0,$$

so that $\mathbb{E}[E] = m$ and $\mathrm{Var}[E] = (m-1)m^2$. We have that $\mathbb{P}(E > t) = \frac{1}{2} - \frac{1}{2}\mathrm{erf}\Big(\frac{\log x - \log m/2}{\sqrt{2\log m}}\Big)$, and

$$\int_0^t x f_E(x)dx = \int_0^t \frac{1}{\sqrt{2\pi \log m}} e^{-\frac{(\log x - \log m/2)^2}{2\log m}} dx.$$

Since the above integral cannot be computed in closed form, we approximate it numerically in our experiments and then use the approximation to compute the functions $h(p)$ and $g(p)$;

(c) Power-Law Service Times: Service times of Easy tasks are power-law distributed, with probability density function: $f_E(x) = m/(m-1)x^{-\frac{2m-1}{m-1}}$, for $x \geq 1$, so that $\mathbb{E}[E] = m$ and $\text{Var}[E] = \infty$, if $m \geq 2$. We have that

$$h(p) = p_0 m\left(1 - \phi(p)^{-\frac{1}{m-1}}\right), \text{ and}$$

$$g(p) = \phi(p)\left(1 - p_0 + p_0\phi(p)^{-\frac{m}{m-1}}\right) + p_0 m\left(1 - \phi(p)^{-\frac{1}{m-1}}\right).$$

We report on two types of experiments:

(i) For a given service time distribution and value of prior p_0, we compare the stability regions for different values of the scaling parameter m. On that end, note the connection of the stochastic variability of service times to the scaling parameter: the coefficient of variation is equal to one for the exponential distribution, irrespective of the value of m, and is equal to $\sqrt{m-1}$ for the lognormal distribution. Regarding the power-law distribution where the variance is infinite, the exponent of the pdf is monotonically decreasing in m, in absolute value, implying a progressively heavier tail;

(ii) For a given value of the scaling parameter m, we compare the stability regions for different service time distributions and values of prior p_0. Here, recall that in all cases the mean service time of Easy tasks is equal to m, which implies that if types were known ex ante, the stability regions would be identical. Note also that as one moves from the exponential to the lognormal, and then to the power-law distribution, the stochastic variability of service times increases.

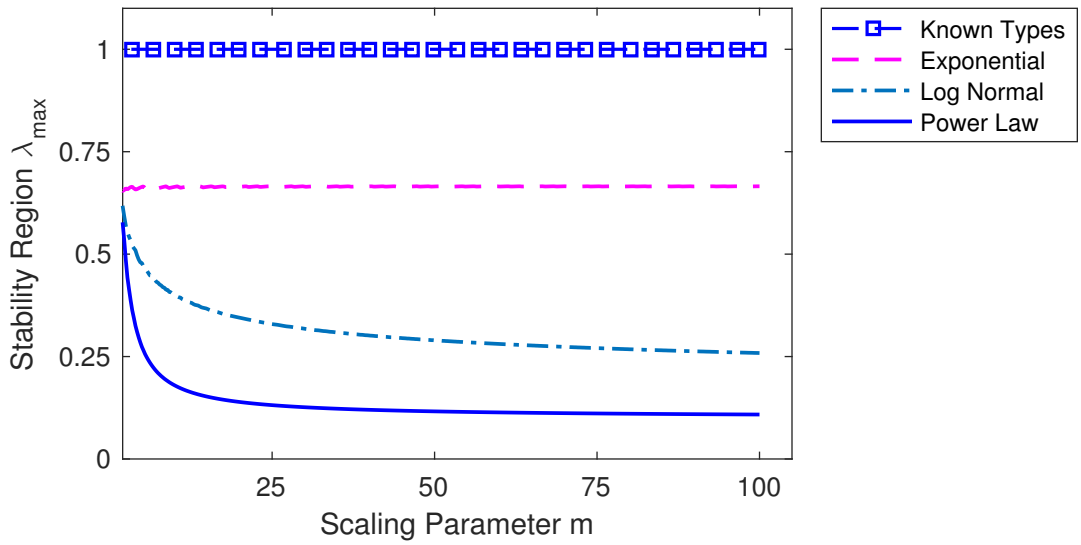

Figure 2 **The impact of the variability in service times on the system's stability region ($p_0 = 0.9$).**

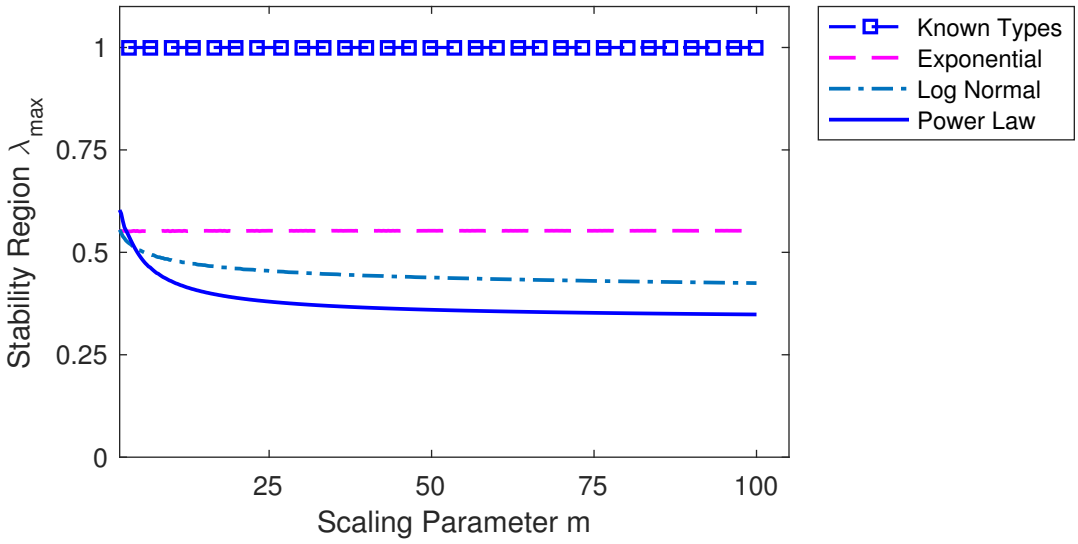

Figure 3 **The impact of the variability in service times on the system's stability region ($p_0 = 2/3$).**

The results of our numerical experiments are presented in Figures 2 and 3, and lead to several conclusions. It is well known that when task types are known ex ante, the service time distributions have no effect on the stability region of the system beyond their mean values. This is no longer true in the presence of task-type uncertainty, in which case the performance loss can be significant compared to the benchmark of known types, ranging from 30% to 90% in our numerical experiments. Moreover, it is noteworthy how the shape of the service time distribution affects the system's stability region: the aforementioned loss is moderate for exponential and increasingly higher for lognormal and power-law distributed service times, i.e., for distributions that feature heavy tails. These findings highlight the fact that uncertainty in task types and the rate of learning are first-order considerations when making system design decisions.[7] Even within the class of lognormal and power-law service time distributions, the higher the stochastic variability of service times, the larger the performance loss.

Finally, Figures 2 and 3 illustrate also the impact of parameter p_0, the fraction of Easy tasks arriving to the system. For service time distributions with heavy tails, the gap between the stability region in the case of known types and the one when task types are not known ex ante becomes larger as p_0 increases. The converse holds for service time distributions with light tails, as this gap seems to be decreasing in p_0. Formalizing these observations and establishing them analytically is challenging: Equation (6) quantifies the aforementioned gap but its behavior with respect to p_0 is hard to disentangle as, apart from the explicit dependence of functions $h(\cdot)$ and $g(\cdot)$, there is an implicit dependence of p^* on p_0 via the solution to the optimization problem in Proposition 3.

[7] Relatedly, Bimpikis and Markakis (2016) establish that the benefits to inventory pooling are substantially lower under heavy-tailed demand uncertainty.

Nevertheless, they are intuitive in the context of our model: information about the type of a given task is generated at a relatively slower rate when the distribution of the service times of Easy tasks has a heavier tail. This, in turn, contributes to the misallocation of resources, e.g., having Easy tasks being rerouted to Senior servers, or having Hard tasks being initially routed to Junior servers; and is particularly critical if the fraction of Easy tasks is higher, in which case the service provider finds it ex ante optimal to have tasks spend more time with Junior servers.

3. Endogenous Training and Service Hierarchies

The first part of the paper focuses on a model that involves a system with two types of servers, heterogeneous in their skills, and two types of tasks, heterogeneous in their service requirements. Junior servers are assumed to be able to serve only Easy tasks, whereas Senior servers are able to serve both types of tasks. The remainder of the paper generalizes this framework, allowing for the possibility of a continuum of task types as well as server skills. A new set of questions arise, which involve determining the optimal skill sets for a given server pool and optimal staffing levels.

More concretely, we consider a setting in which there are two sources of uncertainty regarding a task: its service time, captured by the generic random variable X with support on $[0,\infty)$, and its type, captured by the generic random variable R with support on $[0,1]$. The latter represents the degree of difficulty associated with serving the task, i.e., higher task types correspond to harder tasks. Additionally, there are two types of servers, Junior and Senior, and each of them is associated with a training level. We assume that the training level of Senior servers is equal to one, whereas that of Junior severs, denoted by $\ell \in [0,1]$, is determined by the service provider. A server's training level determines the set of tasks it can serve: servers are not able to serve tasks that have higher type, i.e., difficulty level, than their own training level. We note that this generalization allows us to endogenize the system's staffing cost: training a server to level ℓ—or equivalently hiring a server with training level ℓ—involves cost $w(\ell)$. In summary, the following primitives define the environment that we focus on for the remainder of the section:

(i) *The service time distribution*, conditional on the task type, and parameterized by the training level ℓ of the server, $F^{\ell}_{X|R}(x|r) = \mathbb{P}_{\ell}(X \leq x \mid r \leq R \leq r+dr)$, for $dr \to 0$. We assume that the service time of a task is conditionally independent of the past history of the system, and drawn from the above distribution. We further assume that

(a) if $\ell < r$, then $F^{\ell}_{X|R}(x|r) = 0$ for all x, i.e., if the type of a given task r is larger than the server's training level ℓ, the server cannot complete the task in finite time;

(b) if $r \leq \ell$, then $F^{\ell}_{X|R}(x|r) = F(x)$, i.e., the service time of a task follows the same distribution irrespective of its type r and the server's training level ℓ, as long as $r \leq \ell$.[8]

[8] We make this assumption to simplify the exposition. Our analysis and findings can be extended to service time distributions, and corresponding posterior belief functions, that depend on the servers' training levels in a quite general way.

Finally, we assume that $F(x)$ is continuously differentiable, so we can equivalently consider the corresponding density function $f(x)$, with support $\mathcal{D} \subset [0,\infty)$, as our stochastic primitive.

(ii) *The task type distribution*, $\mathbb{P}(R \le r) = K(r)$. We assume that the type of an arriving task is independent of the past history of the system, and drawn from the above distribution. Moreover, we assume that $K(\cdot)$ is a continuously differentiable function, so we can equivalently consider the density function of task types $k(r)$ as our stochastic primitive.

(iii) *The training cost function*, $w(\ell)$. We assume that $w(\cdot)$ is continuous and monotonically increasing, with $w(0) > 0$ and, without loss of generality, $w(1) = 1$.

(iv) *The inverse posterior belief function*, $\phi(p)$, i.e., the time that a server needs to allocate to a task in order for her belief that she can serve it to reach $p \in [0, p_0]$, provided that there is no service completion in the meantime. Note that $\phi(\cdot)$ does not depend on ℓ, since the service time distribution $F(\cdot)$ is assumed to be independent of ℓ (see Footnote 8). Moreover, we assume that:

(a) $\phi(\cdot)$ is continuous and monotonically decreasing with $\phi(p_0) = 0$;

(b) $\phi^{-1}(t) > 0$, for any $t \in \mathcal{D}$.

We note that in the model of Section 2 we track the posterior belief regarding a task's type explicitly, whereas in the setting considered here we only track the belief of a server about her ability to serve a given task. Similarly to our benchmark model, we let

$$h(\ell,p) = K(\ell)\int_0^{\phi(p)} x f(x)dx,$$
$$g(\ell,p) = \phi(p)\left[1 - K(\ell) + K(\ell)\big(1 - F(\phi(p))\big)\right] + h(\ell,p).$$

Also, we denote by ρ the expected service time of a task if handled by a Senior server, i.e., $\rho = \int_0^\infty x f(x)dx$. With this notation, the stability conditions for the two queues under the $S(q,p)$ policy can be succinctly written as follows:

$$\text{Queue } J: \lambda q g(\ell,p) \le n_j, \text{ and Queue } S: \lambda\big(\rho - q h(\ell,p)\big) \le n_s.$$

3.1. Hierarchies of Service

In this section we provide a structural characterization of the optimal system design when both the Junior servers' training levels, the staffing levels, and the resource allocation policy are decision variables. The service provider's objective is to minimize the total staffing and endogenous training cost, subject to the constraint that both queues are stable for a given arrival rate λ. For simplicity, we express the service provider's optimization problem with respect to the rate-adjusted staffing levels $\bar{n}_j = n_j/\lambda$ and $\bar{n}_s = n_s/\lambda$ as follows:

$$\begin{aligned} \underset{\bar{n}_j,\bar{n}_s,\ell,q,p}{\text{minimize}} \quad & w(\ell)\bar{n}_j + \bar{n}_s \\ \text{subject to} \quad & q g(\ell,p) \le \bar{n}_j \text{ and } \rho - q h(\ell,p) \le \bar{n}_s \\ & \ell, q \le 1, p \le p_0 \text{ and } \bar{n}_j, \bar{n}_s, \ell, q, p \ge 0. \end{aligned} \tag{7}$$

PROPOSITION 4. *An optimal system design either features both Junior and Senior servers and a "hierarchical" structure, i.e., all tasks are initially routed to queue J and then rerouted to queue S, if necessary, or it features only Senior servers.*

Interpreted in the context of the applications that motivate our work, this result justifies the term "hierarchies of service," i.e., the optimal design features either just Senior servers or, when their training/hiring cost is relatively high, a combination of Junior and Senior servers. In the latter case, it is optimal to route all tasks to Junior servers first, and after an appropriately chosen belief threshold to reroute the tasks that are still in service to Senior servers. To gain some intuition on this result, note that the Junior servers' training level and the routing probability are essentially complements. In addition, for a given training level ℓ and posterior belief threshold p, both the objective and the constraints of optimization problem (7) are linear in the remaining decision variables, i.e., the rate-adjusted staffing levels and the routing probability. A direct consequence of this linearity is that an optimal solution can be found at a corner of the constraint set (polytope), which translates to a routing probability of either zero or one.[9]

As a final remark on Proposition 4, we explore the sensitivity of service hierarchies on various features of the model. First, note that the information that the service provider has upon a task's arrival is summarized by the (prior) task type distribution $K(\cdot)$, which we assume to be common for all incoming tasks. One can readily incorporate multiple such priors into the model, as a way of capturing ex ante differences in the information that the provider has at her disposal, and follow a similar analysis, i.e., derive routing probabilities and posterior belief thresholds for tasks with different priors. Given that the posterior belief of a Junior server about her ability to serve a given task incorporates both the prior and the information that is potentially generated by serving the task, it is straightforward to show that (i) there is some performance loss compared to the case where types are known, as long as there is ex ante uncertainty in task types; and (ii) the optimal system design takes again the form of a service hierarchy, with the caveat that tasks that are ex ante different may be initially routed to different queues.

Secondly, throughout the section we assume that the nature of the information generated while a task is in service is about the server's ability to serve the task, i.e., whether the task type is above or below the server's training level, but not on the exact type of the task, i.e., precisely where it lies in the $[0, 1]$ interval. In the latter case service hierarchies may not be optimal. For instance, in

[9] An analogous result, i.e., the optimality of a system design with a hierarchical structure, can be established for a service system with three types of servers, each with potentially different training levels. The proof is very similar to Proposition 4, so we omit the details for brevity. We conjecture that the optimal solution to the joint staffing, training, and (static) resource allocation problem has a hierarchical structure irrespective of the number of levels: for fixed training levels and posterior belief thresholds, it reduces to a network flow-type problem, and such problems are known to admit a linear programming formulation as showcased above.

a service system with three or more distinct types of servers, it may be beneficial for the service provider not to reroute a task to the server pool with the immediately higher training level, if the task is believed to be particularly complex. Relatedly, the optimality of service hierarchies is sensitive to the assumption that the inverse posterior belief function is monotonically decreasing: if a server's posterior belief about a task being within her capabilities can move upwards, then there may be occasions where it is optimal to reroute a task back to a less skilled server, e.g., after the arrival of news indicating that it may be easier than previously thought.

Finally, service hierarchies may not be optimal if the service provider's objective is different than minimizing the total staffing/training cost, e.g., if it involves a component related to the average time that tasks spend in the system.

3.2. Comparative Statics

Although there is always an optimal system design with a simple, hierarchical structure irrespective of the nature of the uncertainty regarding task types and service times, the actual configuration may differ considerably depending on the modeling primitives. In the remainder of the section we consider a number of special cases that aim to highlight how the optimal design, total cost, and overall performance of a service system may differ depending on whether task types are known ex ante or not. Similarly to Section 2.3, we assume that the distribution of the residual service time of Easy tasks is a sufficient statistic for determining the posterior belief function.

First, we consider a setting that involves deterministic service times, affine training cost, and uniformly distributed task types. In particular:

(i) Service times are deterministic, i.e., for fixed $s>0$ and $r\leq \ell$ we have $f(x)=\delta(x-s)$, where $\delta(\cdot)$ denotes the delta function. In other words, if the task is within the server's capabilities, the service time is equal to s irrespective of the task's type and the server's training level. If the task is not within the capabilities of the server, then the service time is infinite;

(ii) Affine training cost, i.e., for fixed $c\in(0,1/2)$, $w(\ell)=c+(1-c)\ell$, for $\ell\in[0,1]$.

The following proposition provides a characterization of the optimal service system design.

PROPOSITION 5. *Consider the service system described above, where service times are deterministic, training cost is affine, and task types are uniformly distributed. Then, the optimal system design takes the following form (all quantities are rate-adjusted):*

(i) if types are known, the optimal system design has $0.5s$ *Junior and* $0.5s$ *Senior servers. Junior servers are trained to serve tasks up to 0.5, and the total staffing and training cost is* $(0.75+0.25c)s$*;*

(ii) if task types are unknown ex ante, the optimal system design consists of s *Senior and no Junior servers. The total staffing and training cost is* s*.*

Setting	Unknown Types/Known Types		
(Service, Training, Types)	Total Servers	Senior Servers	Total Cost
Deterministic, Affine, Uniform	1	2	1.33
Deterministic, Quadratic, Uniform	1.5	1.18	1.21
Stochastic, Affine, Uniform	1	2	1.33
Deterministic, Affine, Nonuniform	1	1.87	1.25

Table 1 Ratio of the total number of servers (second column), Senior servers (third column), and total staffing and training cost (fourth column) for the optimal system designs when types are unknown over when they are known, respectively (assuming $c \to 0$).

Proposition 5 illustrates that the impact of uncertainty in task types can be significant: when the variable cost is the dominant component of the training cost, uncertainty in task types leads to more than 30% increase in the total staffing and training cost associated with the optimal system design. This difference is due to the fact that the service provider finds it optimal to employ more Senior servers as a way to hedge against the uncertainty in task types.

Table 1 summarizes our findings for the case where the fixed cost of training, c, is negligible. Apart from the setting discussed in Proposition 5, we also report on the following settings (the corresponding derivations can be found in the online companion to the paper)::

(i) Quadratic training costs: the training cost is quadratic, i.e., we let $w(\ell) = c + (1-c)\ell^2$, for $\ell \in [0,1]$. Task types are uniformly distributed, and service times are deterministic and equal to s (as long as the task is within the server's capabilities);

(ii) Stochastic service times: service times are uniformly distributed with mean s (as long as the task is within the server's capabilities), i.e., if $r \leq \ell$ we have $f(x) = 1/2s^{-1}$, for $0 \leq x \leq 2s$. The training cost is affine, and task types are uniformly distributed;

(iii) Nonuniform task types: task types are distributed according to the triangular distribution $k(r) = 4r$ for $0 \leq r \leq 1/2$, and $k(r) = 2 - 4(r - 1/2)$ otherwise. This specification aims to capture the fact that, often, task types are concentrated around certain "typical" values, i.e., modes of the distribution. The training cost is affine, and service times are deterministic and equal to s (as long as the task is within the server's capabilities).

Although the specifics of these special cases differ, they all illustrate that the optimal cost in the presence of uncertainty about task types is significantly higher compared to the respective no-uncertainty benchmarks. This is mainly due to the fact that minimizing the total cost subject to system stability commands a design that features a larger number of Senior servers, often eliminating Junior servers altogether.

4. Extensions

The present section discusses three extensions to our benchmark model. First, we consider a variation in which the service of Easy tasks is fully transferable in case they are rerouted. This may

correspond to settings where all servers have a common way of dealing with Easy tasks, so, in the case of a task being rerouted from a Junior server to a Senior one, the latter would not repeat the service already performed by the former. Second, in order to capture settings where service takes the form of a series of discrete steps, we show how our results can be extended to the case where the posterior belief function $\pi(\cdot)$ is piecewise constant. Third, motivated by applications in which the service includes a component that is common to all tasks, e.g., taking the medical history and vital signs of a patient, we consider an alternative formulation where the service requirement of every task includes a deterministic setup time that is transferable if the service is preempted.

4.1. Transferable Service

Consider a variation of our benchmark model, where the service provided to an Easy task is fully transferable, i.e., if the task is rerouted, a Senior server only needs to provide the remaining service instead of starting from the beginning. In contrast, any time spent on a Hard task by a Junior server does not contribute towards service completion, as such a task is not in the server's skill set.

Although this setting is different than the one in Section 2, the insights that we obtain are similar. This is not surprising given that the service provider faces, essentially, the same tradeoff: if a Junior server spends too little time with a task, then she may end up rerouting too many Easy tasks to Senior servers, effectively wasting their resources; if she spends too much time with a task, then she is likely wasting her own resources as the task may be Hard.

In particular, all our main results hold, verbatim, by redefining the functions that we use in order to characterize the stability region of the $S(q,p)$ policy as follows:

$$h_{\rm tr}(p) = p_0 \int_0^{\phi(p)} x f_E(x)dx + p_0 t \mathbb{P}\big(E > \phi(p)\big), \quad \text{and} \quad g_{\rm tr}(p) = \phi(p)(1-p_0) + h_{\rm tr}(p).$$

4.2. Service as a Sequence of Tests

Another feature of interest in the benchmark model of Section 2 is that the posterior belief function $\pi(\cdot)$ is continuous and monotonically decreasing, implying that even an infinitesimal amount of service provides extra information about the type of a task. In certain applications, however, it may be more appropriate to assume that service consists of a series of tests. Each such test, if conclusive, leads to the fulfillment of the task's service requirements. Hence, information about the type of the task arrives only when the outcome of a test is revealed, i.e., information is generated at discrete times. Then, the posterior belief remains constant while a given test is in progress, and jumps to one, if conclusive, or downwards, if not conclusive, upon its completion.

We model such a setting by assuming that function $\pi(\cdot)$ is piecewise constant and we show that, for the most part, the results and insights derived in the context of our benchmark model remain intact. More concretely, let N denote the number of tests that are within a Junior server's skill

set to perform. Also, let nonnegative real numbers $t_0, \ldots, t_N$, with $t_0 = 0 < t_1 < t_2 < \ldots < t_N < \infty$, and $p_1, \ldots, p_N$, with $p_0 > p_1 > \ldots > p_N = 0$. The k^{th} test is administered at time t_{k-1} and its results come back at time t_k. We assume the following posterior belief function:

$$\pi(t) = p_k, \qquad t \in [t_k, t_{k+1}), \qquad k = 0, \ldots, N-1, \tag{8}$$

and $\pi(t) = 0$, for all $t \geq t_N$. In other words, p_k is the posterior belief of a Junior server about the task at hand being Easy, if the first k tests have not been conclusive. Moreover, at least one of the N tests leads to a task's service completion, if the task is Easy.

Let $c_1, \ldots, c_N$ be nonnegative real numbers, with $c_1 + \ldots + c_N = 1$. The service time of an Easy task when served by either a Junior or a Senior server is a discrete random variable with probability mass function $\mathbb{P}(E = t_k) = c_k$, for $k = 1, \ldots, N$, i.e., c_k is the probability that the k^{th} test leads to the task's service completion. If the task is rerouted to a Senior server before service completion, the elapsed *service is transferred*, i.e., the Senior server performs only the remaining tests. Hard tasks cannot be completed by a Junior server, since the necessary tests are beyond the skill set of a Junior server. On the other hand, if a Hard task is assigned to a Senior server, it requires service time that follows probability density function $f_H(\cdot)$. Let

$$\hat{h}(k) = p_0 \sum_{i=1}^{k} t_i c_i + p_0 t_k \sum_{i=k+1}^{n} c_i \qquad \text{and} \qquad \hat{g}(k) = t_k(1 - p_0) + \hat{h}(k), \qquad k = 1, \ldots, N,$$

and $\hat{h}(0) = \hat{g}(0) = 0$. It is straightforward to verify that under the $S(q, p_k)$ policy, with $q \in [0,1]$ and $k \in \{0, \ldots, N\}$, queue J is stable if $\lambda q \hat{g}(k) \leq n_j$. Similarly, queue S is stable under the $S(q, p_k)$ policy, if $\lambda\big(\alpha + \beta - q\hat{h}(k)\big) \leq n_s$. Consequently, the stability region of the $S(q, p_k)$ policy, $\Lambda_{S(q,p_k)}$, is the set of arrival rates: $\Lambda_{S(q,p_k)} = \left\{ \lambda \geq 0 \;\middle|\; \lambda \leq \min\left\{ \frac{n_j}{q\hat{g}(k)}, \frac{n_s}{\alpha + \beta - q\hat{h}(k)} \right\} \right\}$, which is in line with Proposition 2. Clearly, in order to find the (static) resource allocation policy with the largest stability region, one simply needs to optimize over the parameter space (q, p_k). The discrete nature of the posterior belief function (8) does not allow for further analysis in this case, e.g., deriving a result analogous to Proposition 3, but a numerical solution is straightforward.

As a final note, consider the optimal staffing and (static) resource allocation problem, i.e., when the service provider simultaneously optimizes over the staffing levels and the parameters of the $S(q, p_k)$ policy. It is easy to verify that the main insight of Proposition 4, i.e., that there exists an optimal solution with a hierarchical structure, extends to this case as well.

4.3. Setup Time and Specialization

In this section we consider an extension to our benchmark model, where the service of all tasks consists of two components: the setup time that is deterministic and transferable upon rerouting,

and the actual service time that is stochastic and non-transferable. Our results indicate that if the setup time is sufficiently large, then it is optimal to have Junior servers specialize in completing just the setup component, whereas Senior servers are exclusively reserved for the actual service component of all tasks. In that case, there is no performance loss compared to the case where task types are known ex ante. In contrast, if the setup time is relatively small, then some loss in performance is unavoidable. To some extent, this result provides an additional justification for the typical organizational structure in healthcare systems, where nurses or physician assistants collect basic information from patients and then pass it on to the physicians, who are tasked with providing a diagnosis and a treatment plan.

Specifically, the service times of Easy tasks when served by Senior servers are i.i.d and take the form $(\tau_s + E)$, where τ_s is the deterministic setup time and E a random variable with density $f_E(x)$. Similarly, the service times of Hard tasks when served by Senior servers are i.i.d and take the form $(\tau_s + H)$, where H a random variable with density $f_H(x)$. Again, for simplicity we assume that Junior servers can serve Easy tasks at the same rate as Senior servers. Let

$$\tilde{h}(t) = p_0 \int_0^t x f_E(x) dx \qquad \text{and} \qquad \tilde{g}(t) = t\big(1 - p_0 + p_0 \mathbb{P}(E > t)\big) + \tilde{h}(t).$$

Following arguments similar to the proofs of Propositions 2 and 3, we establish the following result (whose proof we omit for brevity).

PROPOSITION 6. *The stability region of the benchmark model if service times include a setup component, τ_s , which is transferable, is the set of arrival rates:*

$$\Lambda_T = \left\{ \lambda \geq 0 \;\middle|\; \lambda \leq \frac{n_s + n_j}{\tau_s + \alpha + \beta + q^*\big(\tilde{g}(t^* - \tau_s) - \tilde{h}(t^* - \tau_s)\big)} \right\},$$

where (q^*, t^*) is a solution to the optimization problem:

$$\begin{aligned} &\textit{minimize} && q\big(\tilde{g}(t - \tau_s) - \tilde{h}(t - \tau_s)\big) \\ &\textit{subject to} && q = I(t), q \in [0,1], \ \textit{and } t \geq \tau_s, \end{aligned}$$

with $I(t) \equiv \big[n_j(\tau_s + \alpha + \beta)\big] / \big[n_s\big(\tau_s + \tilde{g}(t - \tau_s)\big) + n_j\big(\tau_s + \tilde{h}(t - \tau_s)\big)\big]$.

We note that the function $I(\cdot)$ is continuous and monotonically decreasing in $[\tau_s, \infty)$, so the largest value it can take in this interval is $I(\tau_s) = \big[n_j(\tau_s + \alpha + \beta)\big] / \big[\tau_s(n_s + n_j)\big]$. If $n_j(\alpha + \beta)/n_s \leq \tau_s$, i.e., if $t = \tau_s$ is a feasible solution, then clearly $\big(I(\tau_s), \tau_s\big)$ is an optimal solution to the above optimization problem. In that case, Proposition 6 implies that there is no performance loss compared to the case where task types are known ex ante. This is because it is optimal for Junior servers to specialize in serving only the setup time of tasks, potentially not for the entire population, while the actual service is left to Senior servers. On the other hand, if $n_j(\alpha + \beta)/n_s > \tau_s$, i.e., the setup time is relatively short, then both Junior and Senior servers are tasked with completing part of the actual service time, which implies that there is performance loss, unavoidably.

5. Delay Considerations

Our analysis so far has adopted the long-term throughput as the performance metric of interest. Although the throughput is a first-order consideration for system design, exploring more refined performance metrics such as average delay, which take into account the short-term workload fluctuations in the system, would certainly add to our understanding of optimal operational decisions in this context.

A thorough investigation of the delay performance of service systems with task-type uncertainty is beyond the scope of the present paper.[10] Nevertheless, we perform an approximate delay analysis in a simple instance of our benchmark model, confirming one of the main findings of the paper: heavier tails in service times lead to greater performance loss compared to the case where task types are known, at least when the rate of learning is driven solely by the service time distributions, e.g., as in Equation (1). On the other hand, while we do not consider staffing decisions explicitly, we argue that our second main finding regarding "service hierarchies" is, in general, not robust when the service provider takes the average delay into account.

Specifically, we consider the benchmark model with one Junior and one Senior server, with tasks arriving to the system according to a homogeneous Poisson process, and the service times of both Hard and Easy tasks being exponentially distributed, i.e., $f_H(x) = f_E(x) = \mu e^{-\mu x}$, for all $x \geq 0$. In that setting, and under a $S(q,p)$ policy, queue J is a $M/G/1$ queue with arrival rate $\lambda_j = \lambda q$, service rate $\mu_j = 1/g(p)$, and load factor $\rho_j = \lambda_j/\mu_j$. The variance of the service time, $\mathrm{Var}(S_j)$, can also be computed in closed form, and the average delay in queue J can thus be obtained by the Pollaczek-Khinchine (P-K) formula:

$$W_j = \frac{1}{\mu_j} + \frac{\lambda_j \mathrm{Var}(S_j)}{2(1-\rho_j)}.$$

In contrast, while the aggregate arrivals in queue S are not Poisson, we treat them as such, with rate $\lambda_s = \lambda\big(1 - qp\mathbb{P}\big(E \leq \phi(p)\big)\big)$, load factor $\rho_s = \lambda\big(\mathbb{E}[E] - qh(p)\big)$, service rate $\mu_s = \mu$, and variance of service times $\mathrm{Var}(S_s) = (1/\mu_s)^2$.[11] Under this approximation, the average delay in queue S, W_s, can be obtained, again, by the P-K formula, and the (approximate) average delay in the system is

$$W = \big(1 - qp_0\mathbb{P}\big(E \leq \phi(p)\big)\big)W_s + qW_j.$$

[10] To put this into context, note that when we have one Junior and one Senior server and task types are *known*, the system is equivalent to the widely studied N-model, for which only policies that are asymptotically delay optimal in heavy traffic are known, e.g., Harrison (1998), Bell and Williams (2001), and Tezcan and Dai (2010).

[11] This is tantamount to the *Kleinrock independence approximation*, often made in the analysis of queueing networks with exogenous Poisson arrivals and near-exponential service times, e.g., see Section 3.6 of Bertsekas and Gallager (1992). The rationale behind this approximation is that by merging the traffic that comes to queue S from queue J with an exogenous stream of Poisson traffic, the dependence between interarrival and service times induced at queue J weakens considerably, and the aggregate traffic can be thought as approximately Poisson.

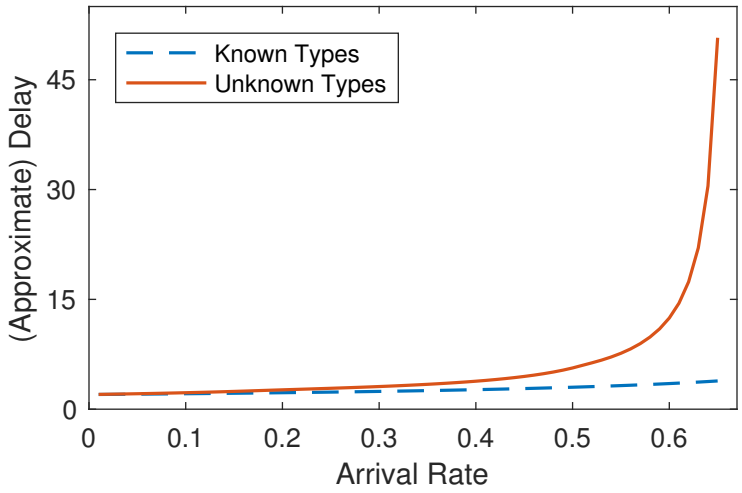

Figure 4 The (approximate) throughput-delay curve of the benchmark model with one Junior server, one Senior server, exponential service times with rate $\mu = 0.5$, and proportion of Easy tasks $p_0 = 0.666$. We compare the performance of the (approximately) optimal $S(q,p)$ policy in the cases of unknown task types to the performance of the optimal static policy in the case of known types.

We use this closed-form expression to compute the optimal $S(q,p)$ policy for any given arrival rate in the stability region, i.e., to determine the (approximate) *throughput-delay curve* of the system. Understanding the latter curve is critical for optimal system design under both measures: minimize the average delay under a lower bound on the arrival rate supported, maximize the supported rate subject to an upper bound on the average delay, or optimize a convex combination of the two.

In Figure 4 we compare the (approximate) throughput-delay curve of the system when task types are known to the one when they are not known ex ante. In the former case, for any given arrival rate in the stability region, we compute the optimal average delay within the class of static policies, i.e., all Hard tasks are routed to queue S and Easy tasks are split probabilistically between the two queues. In the latter case, we compute the optimal average delay within the class of $S(q,p)$ policies. At light loads the delay performance in the two cases is comparable, as the service provider minimizes the average delay by sending tasks directly to Senior servers. At high loads the picture is quite different: the average delay when task types are unknown is much larger, which is attributed to the fact that the stability region of the system in that case is smaller, so the given arrival rates are much closer to the boundary. We note, however, that under critical loads the average delay seems to scale in the familiar $1/(1-\rho)$ fashion, similarly to the single-server queue.

We conclude the section by commenting on how the main findings of the paper regarding the system's stability region may relate to optimizing with respect to a delay criterion. We have shown that heavier tails in service times imply a greater reduction in the stability region compared to the case where task types are known, at least when the rate of learning is driven solely by the service time distributions. We expect the same insight to hold when the service provider's objective

function relates to the average delay: heavier tails result in higher variance of service times and also higher load factors for a given arrival rate, as the stability region shrinks. Both facts point to a higher average delay according to the P-K formula, which dictates (approximately) W_j and W_s.

The other main insight of the paper relates to service hierarchies, i.e., there always exists a hierarchical system structure that minimizes the staffing/training cost subject to queue stability. In general, this insight is not robust under delay considerations. The hierarchy result is based on the fact that, for fixed ℓ and p, the joint training, staffing, and resource allocation problem (7) has constraints relating to queue stability that are linear in the remaining decision variables. As the average delay is nonlinear in these variables, the new constraint set is not a polyhedron, so the existence of an optimal solution in one of its vertices, i.e., with routing probability either zero or one, is not guaranteed any more. That said, there may be scenarios where a hierarchical design is still a reasonable choice. For instance, if the system is heavily loaded and the service times have limited variability, then the P-K formula suggests that the $1/(1-\rho)$ factor is the dominant one in determining the average delay. In that case, the delay performance of a maximally stable hierarchical system design (under a budget constraint) should be reasonably good.

6. Concluding Remarks

Service systems typically deal with tasks (or customers/patients) of varying complexity, and important information about the type of a given task is obtained only after it has been assigned to a server. Our analysis aims to provide an understanding on the design and operation of service systems in the presence of uncertainty regarding task types, by illustrating its impact on the service provider's staffing and resource allocation decisions. We show that the performance loss directly attributed to this uncertainty can be quite significant. In addition, we establish that it is optimal for the service provider to design the system in a hierarchical structure, where all tasks are initially routed to the least skilled servers and then, in case some tasks are still in service when an appropriately determined belief threshold is reached, they are rerouted to higher skilled servers.

Our modeling framework leads to several directions for future research. One could focus on endogenizing the servers' training levels and jointly optimizing over staffing and resource allocation decisions. In the present contribution we consider tasks and servers that can be ranked in terms of their complexity and skills, respectively. However, for some environments, a multi-dimensional representation of a task's complexity and servers whose skills are not necessarily nested may be more appropriate. Then, the optimal design may feature servers with some training in a broad set of skills, assisted by few highly trained, specialized servers. Another potential direction to explore would be to consider a setting where servers act in a decentralized manner, aiming to maximize their (expected) payoff. Taking the incentives of servers into account leads to a host of questions

involving the design of compensation structures that induce the optimal system design, or explicitly having servers determine their training levels and then compete for providing service to customers.

Appendix : Proofs

Proof of Proposition 2

The time that a task spends with a Junior server under the $S(q,p)$ policy is:

- equal to $\phi(p)$, with probability $(1-p_0)+p_0\mathbb{P}\big(E>\phi(p)\big)$;
- up to $\tau\in\big[0,\phi(p)\big]$, with probability $p_0\int_0^\tau f_E(x)dx$.

Therefore, the expected time that a task spends with a Junior server is equal to

$$\phi(p)\left(1-p_0+p_0\mathbb{P}\big(E>\phi(p)\big)\right)+p_0\int_0^{\phi(p)}xf_E(x)dx=\phi(p)\left(1-p_0+p_0\mathbb{P}\big(E>\phi(p)\big)\right)+h(p)=g(p).$$

Thus, queue J is stable if and only if $\lambda qg(p)\le n_j$.

On the other hand, queue S ends up receiving all tasks, except those that are initially routed to queue J and have service time no more than $\phi(p)$. Hence, the (long-term) rate at which workload arrives to queue S is equal to

$$\begin{aligned}&\lambda(1-q)p_0\mathbb{E}[E]+\lambda(1-q)(1-p_0)\mathbb{E}[H]+\lambda q(1-p_0)\mathbb{E}[H]+\lambda qp_0\int_{\phi(p)}^{\infty}xf_E(x)dx\\&\quad=\lambda\left(p_0\mathbb{E}[E]+(1-p_0)\mathbb{E}[H]-qp_0\int_0^{\phi(p)}xf_E(x)dx\right)=\lambda\big(\alpha+\beta-qh(p)\big).\end{aligned}$$

Thus, queue S is stable if and only if $\lambda\big(\alpha+\beta-qh(p)\big)\le n_s$.

Combined, these imply that the stability region of policy $S(q,p)$ is the set of arrival rates

$$\Lambda_{S(q,p)}=\left\{\lambda\ge0\;\middle|\;\lambda\le\min\left\{\frac{n_j}{qg(p)},\frac{n_s}{\alpha+\beta-qh(p)}\right\}\right\}.$$

Proof of Proposition 3

Note that the denominator of the first term in Equation (3) is continuous and monotonically increasing in q, whereas the denominator of the second term is continuous and monotonically decreasing in q. Similarly, the denominator of the first term in Equation (3) is continuous and monotonically decreasing in p, whereas the denominator of the second term is continuous and monotonically increasing in p. This is due to the fact that functions $h(\cdot)$ and $g(\cdot)$ are continuous and monotonically decreasing in p, since $\phi(\cdot)$ is a continuous and monotonically decreasing function and the probability density function of random variable E is atomless. Finally, the first term in Equation (3) can take any value in $(0,\infty)$. This is due to the fact that $\mathcal{D}=[0,\infty)$, which implies that for any $t\in[0,\infty)$, there exists $p_t\in(0,p_0]$ such that $t=\phi(p_t)$.

These imply that the stability region of the $S(q,p)$ policy is maximized if the two terms are equal, i.e., $q=n_j(\alpha+\beta)/\big(n_sg(p)+n_jh(p)\big)\equiv i(p)$, or equivalently, $qh(p)=\big[n_j(\alpha+\beta)-n_sq\big(g(p)-h(p)\big)\big]/\big(n_s+n_j\big)$, subject to the constraint that $q\in[0,1]$. The last equation, combined with Equation (3), implies that for a given $q\in[0,1]$, the $S(q,p)$ policy with the largest stability region is $S\big(q,i^{-1}(q)\big)$, and

$$\Lambda_{S(q,p)}\subseteq\Lambda_{S(q,i^{-1}(q))}=\left\{\lambda\ge0\;\middle|\;\lambda\le\frac{n_s+n_j}{\alpha+\beta+q\big(g(p)-h(p)\big)},\;p=i^{-1}(q)\right\}.$$

So, a maximally stable $S(q,p)$ policy is a solution to the optimization problem:

$$\begin{aligned} \text{minimize} \quad & q\big(g(p)-h(p)\big) \\ \text{subject to} \quad & q=i(p), q\in[0,1], \text{ and } p\in[0,p_0]. \end{aligned}$$

Note that $i(p)$ is continuous and monotonically increasing in p, with domain $(0,\infty)$, so there exists a unique p_1 such that $i(p_1)=1$, or equivalently, $n_s g(p_1)+n_j h(p_1)=n_j(\alpha+\beta)$. Thus, a maximally stable $S(q,p)$ policy is $S\big(i(p^*),p^*\big)$, where p^* is a solution to the following optimization problem:

$$\begin{aligned} \text{minimize} \quad & \frac{g(p)-h(p)}{n_s g(p)+n_j h(p)} \\ \text{subject to} \quad & p\le\{p_0,p_1\}. \end{aligned}$$

After substitution, we derive Equation (4).

Proof of Proposition 4

The problem can be equivalently written in the following form:

$$\begin{aligned} \text{minimize} \quad & \alpha(\ell,p) \\ \text{subject to} \quad & \ell\le 1, p\le p_0, \text{ and } \ell,p\ge 0, \end{aligned}$$

where for a given (ℓ,p) pair, the value of the function $\alpha(\ell,p)$ is defined implicitly as the solution to:

$$\begin{aligned} \alpha(\ell,p)\equiv \text{minimize} \quad & w(\ell)\bar{n}_j+\bar{n}_s \\ \text{subject to} \quad & qg(\ell,p)\le\bar{n}_j \text{ and } \rho-qh(\ell,p)\le\bar{n}_s \\ & q\le 1 \text{ and } \bar{n}_j,\bar{n}_s,q\ge 0. \end{aligned}$$

The latter is a linear program with decision variables $\bar{n}_j$, $\bar{n}_s$, and q, and it is straightforward to show that it has a finite solution. Thus, for any (ℓ,p) pair, an optimal solution lies on a vertex of the constraint polyhedron. On the other hand, function $w(\cdot)$ is continuous in ℓ, and functions $h(\cdot)$ and $g(\cdot)$ are continuous in both arguments. Therefore, function $\alpha(\cdot)$ is continuous in both arguments, and defined over a compact set. Hence, the Extreme Value Theorem implies that the former optimization problem has a solution. In sum, there exists an optimal solution with routing probability either zero or one.

Proof of Proposition 5

First, we consider the case where task types are known upon their arrival. The stability conditions for the two queues, assuming a training level ℓ for the Junior servers, are as follows:

- Queue J: $\lambda\mathbb{P}(R\le\ell)s\le n_j+\big(n_s-\lambda\mathbb{P}(R>\ell)s\big) \implies \lambda s\le n_s+n_j$,
- Queue S: $\lambda\mathbb{P}(R>\ell)s\le n_s$.

It is straightforward to verify that there exists a resource allocation policy that stabilizes the system subject to the stability conditions above, so we will not optimize explicitly over the policy parameters. The (rate-adjusted) joint staffing, training, and resource allocation problem takes the form:

$$\begin{aligned} \text{minimize} \quad & w(\ell)\bar{n}_j+\bar{n}_s \\ \text{subject to} \quad & s\le\bar{n}_j+\bar{n}_s \text{ and } \mathbb{P}(R>\ell)s\le\bar{n}_s \\ & \ell\le 1 \text{ and } \bar{n}_j,\bar{n}_s,\ell\ge 0. \end{aligned} \tag{9}$$

The Lagrangian of the problem is

$$L(\bar{n}_j,\bar{n}_s,\ell)=w(\ell)\bar{n}_j+\bar{n}_s+\mu_1(s-\bar{n}_j-\bar{n}_s)+\mu_2\big(\mathbb{P}(R>\ell)s-\bar{n}_s\big)+\mu_3(\ell-1),$$

so the Karush-Kuhn-Tucker optimality conditions imply (i) $\mu_1=w(\ell)>0$; (ii) $\mu_2=1-w(\ell)\geq 0$; and (iii) $(1-c)\bar{n}_j-\big(1-c-(1-c)\ell\big)s+\mu_3=0$. On the other hand, complementary slackness implies that (i) $\bar{n}_j+\bar{n}_s=s$; (ii) $\mu_2\big((1-l)s-\bar{n}_s\big)=0$; and (iii) $\mu_3(\ell-1)=0$. Next, we distinguish between two cases.

Case (i): $\ell<1$. This implies that $\mu_3=0$ and $\mu_2>0$. So, a candidate optimal solution satisfies:

$$\text{(i) } \bar{n}_j+\bar{n}_s=s,\text{(ii) } (1-\ell)s=\bar{n}_s,\text{ and }\ \text{(iii) } \bar{n}_j=(1-\ell)s.$$

We have that $\bar{n}_j=\bar{n}_s=s/2$, and $\ell=1/2$. So, a candidate optimal solution is $(\bar{n}_j,\bar{n}_s,\ell)=\big(0.5s,0.5s,0.5\big)$, with corresponding cost $(0.75+0.25c)s$;

Case (ii): $\ell=1$. This implies that the system "degenerates" into a system with just Senior servers, and we have a continuum of candidate optimal solutions of the form $(\bar{n}_j,\bar{n}_s,\ell)=(\bar{n}_j,s-\bar{n}_j,1),\ \bar{n}_j\in[0,s]$, with associated cost equal to s. Note that for any $c\in(0,1/2)$, the associated cost of the candidate solution in Case (i) is strictly lower than that of any candidate solution in Case (ii). By arguing similarly to Proposition 4 we ensure that an optimal solution to (9) exists, so $(\bar{n}_j,\bar{n}_s,\ell)=\big(0.5s,0.5s,0.5\big)$ is the unique global minimum.

Next, we consider the case of unknown types. Since service times are deterministic and equal to s, clearly, it is optimal to have a threshold p^*, such that $\phi(p^*)=s$. Thus, we restrict our search within the class of $S(q,p^*)$ policies. Assuming that Junior servers have training level equal to ℓ, we have that $h(\ell,p^*)=s\mathbb{P}(R\leq\ell)$ and $g(\ell,p^*)=s\mathbb{P}(R>\ell)+s\mathbb{P}(R\leq\ell)=s$. In turn, the stability conditions for the two queues are as follows:

$$\text{Queue } J:\lambda qs\leq n_j \text{ and Queue } S:\lambda\big(s-qs\mathbb{P}(R\leq\ell)\big)\leq n_s.$$

Therefore, the (rate-adjusted) joint staffing, training, and resource allocation problem takes the form:

$$\begin{aligned}\text{minimize}\quad & w(\ell)\bar{n}_j+\bar{n}_s\\ \text{subject to}\quad & qs\leq\bar{n}_j \text{ and } s-qs\mathbb{P}(R\leq\ell)\leq\bar{n}_s\\ & q,\ell\leq 1 \text{ and } \bar{n}_j,\bar{n}_s,q,\ell\geq 0.\end{aligned}\tag{10}$$

The Lagrangian of the problem is

$$L(\bar{n}_j,\bar{n}_s,q,\ell)=w(\ell)\bar{n}_j+\bar{n}_s+\mu_1(qs-\bar{n}_j)+\mu_2\big(s-qs\mathbb{P}(R\leq\ell)-\bar{n}_s\big)+\mu_3(q-1)+\mu_4(\ell-1).$$

By arguing similarly to Proposition 4 we ensure that an optimal solution to (10) exists, has a hierarchical structure, i.e., $q=1$. Combining this with the Karush-Kuhn-Tucker first-order optimality and complementary slackness conditions, we conclude that we need to study just the following two cases:

Case (i): $q=1$ and $\ell<1$. In that case $\mu_4=0$, and the KKT system of equations implies that $\bar{n}_j=s/(1-c)$ and $\bar{n}_j=s$. The two are inconsistent since $c>0$, so we have no candidate optimal solutions;

Case (ii): $q=1$ and $\ell=1$. Here, we have that $\bar{n}_s=0$, $\bar{n}_j=s$, and the associated cost is equal to s.

Hence, $(\bar{n}_j,\bar{n}_s,\ell,q,p)=(0,s,1,1,p^*)$, where $\phi(p^*)=s$, is a global minimum of (10).

Quadratic Training Costs, Stochastic Service Times and Non-Uniform Task Types

Quadratic Training Costs. We consider the case where training cost is convex (quadratic), motivated by the fact that the marginal training cost may be increasing. In particular, we let $w(\ell) = c + (1-c)\ell^2$, for $\ell \in [0,1]$. As we show in Proposition 7, unlike the case of affine training cost, now it is optimal to employ both Junior and Senior servers even when task types are unknown ex-ante. However, both the total staffing and training cost and the number of Senior servers employed are still significantly higher compared to the no-uncertainty case.

PROPOSITION 7. *Consider an environment where service times are deterministic, training cost is quadratic, and task types are uniformly distributed. Then, the optimal system design takes the following form (all quantities are rate-adjusted):*

(i) if task types are known, the optimal system design consists of $0.577s$ *Junior servers and* $0.423s$ *Senior servers. Junior servers are trained to serve tasks up to 0.577, and the total staffing and training cost is* $(0.615 + 0.384c)s$*;*

(ii) if task types are unknown, the optimal system design consists of s *Junior servers and* $\frac{1-2c}{2(1-c)}s$ *Senior servers. Junior servers are trained to serve tasks up to* $\frac{1}{2(1-c)}$*, and the total staffing and training cost is* $\left[1 + c - \frac{1}{4(1-c)}\right]s$.

Proof: We follow a similar proof strategy to Proposition 5. In particular, note that for the case of known task types, both the optimization problem (9) and its Lagrangian function have been written assuming a general training cost function. Specifying the corresponding Karush-Kuhn-Tucker first-order optimality and complementary slackness conditions for the case of quadratic training cost, we have that a candidate optimal solution must satisfy the system of equations:

$$\begin{aligned}
&\bar{n}_j + \bar{n}_s = s,\\
&\mu_2 = 1 - c - (1-c)\ell^2,\\
&\mu_2\big(\mathbb{P}(R > \ell)s - \bar{n}_s\big) = 0,\\
&-\big(1 - c - (1-c)\ell^2\big)s + 2(1-c)\ell\bar{n}_j + \mu_3 = 0,\\
&\mu_3(\ell - 1) = 0.
\end{aligned}$$

Again, we need to distinguish between the following two cases:

Case (i): $\ell < 1$. This implies that $\mu_3 = 0$ and $\mu_2 > 0$, so that $(1-\ell)s = \bar{n}_s$. We have that

$$-\big(1 - c - (1-c)\ell^2\big)s + 2(1-c)\ell\bar{n}_j = 0 \implies \bar{n}_j = \frac{s}{2}\Big(\frac{1}{\ell} - \ell\Big).$$

Since $\bar{n}_j + \bar{n}_s = s$, we have that $\ell = \frac{1}{\sqrt{3}}$. Therefore, $\bar{n}_j = \Big(\sqrt{3} - \frac{1}{\sqrt{3}}\Big)\frac{s}{2} \approx 0.758s$, $\bar{n}_s = \Big(1 - \frac{1}{\sqrt{3}}\Big)s \approx 0.423s$, and the associated cost is equal to $s\Big(1 - \frac{1}{\sqrt{3}}\Big) + \frac{s(1+2c)}{6}\Big(\sqrt{3} - \frac{1}{\sqrt{3}}\Big) \approx (0.615 + 0.384c)s$. Summarizing, the candidate optimal solution that we obtain in this case is

$$(\bar{n}_j, \bar{n}_s, \ell) = \big((0.615 + 0.384c)s, 0.423s, 0.577s, 0.577\big). \tag{11}$$

It can be verified that for every $c \in (0, 1/2)$, the value of the objective function of the candidate solution is always strictly less than $\left(1 - \frac{1}{3\sqrt{3}}\right)s \approx 0.807s$;

Case (ii): $\ell = 1$. This implies that the system "degenerates" into a system with just Senior servers, and every candidate optimal solution has associated cost equal to s, i.e., higher than that of the previous case. Hence, the unique optimal solution is the vector in Equation (11).

Similarly, for the case of unknown task types, a candidate optimal solution must satisfy the system of equations:

$$\begin{aligned}
&\bar{n}_j = qs,\\
&\bar{n}_s = (1 - q\ell)s,\\
&-qs + 2(1-c)\ell\bar{n}_j + \mu_4 = 0,\\
&\left(c - \ell + (1-c)\ell^2\right)s + \mu_3 = 0,\\
&\mu_3(q-1) = 0,\\
&\mu_4(\ell - 1) = 0.
\end{aligned}$$

Since we know that there exists an optimal solution to this problem that has a hierarchical structure, i.e., $q = 1$, we need to consider only the following cases:

Case (i): $q = 1$ and $\ell < 1$: Then, $\mu_4 = 0$. This implies that $\bar{n}_j = \frac{s}{2(1-c)\ell}$, which combined with the fact that $\bar{n}_j = s$, implies that $\ell = \frac{1}{2(1-c)} \in (1/2, 1)$. From this we have that $\bar{n}_s = \frac{1-2c}{2(1-c)}s$, and the associated cost of the candidate solution is equal to $\left[1 + c - \frac{1}{4(1-c)}\right]s < s$, for all $c \in (0, 1/2)$;

Case (ii): $q = 1$ and $\ell = 1$. Here, again the system "degenerates" into one that has only Senior servers, and every candidate solution has associated cost equal to s, hence is discarded.

Therefore, the optimal solution in the case of unknown task types is

$$(\bar{n}_j, \bar{n}_s, \ell, q, p) = \left(s, \frac{1-2c}{2(1-c)}s, \frac{1}{2(1-c)}, 1, p^*\right), \tag{12}$$

where p^* satisfies $\phi(p^*) = s$. Q.E.D.

Although we do not explicitly state any results for the cases where service times are stochastic or task types are triangularly distributed, we do report on both these settings in Table 1. To provide a characterization of the optimal system design, we follow a similar reasoning to the proofs of Propositions 5 and 7. While for brevity we omit detailed proofs of these findings, below we provide proof sketches.

Stochastic Service Times. When task types are known, the corresponding optimization problem is again given by (9). This is due to the fact that when task types are known, the distribution of service times comes into the constraints of the problem only through its mean. Thus, the optimal solution is given as in the case of deterministic service times.

However, when task types are unknown, we need to explicitly optimize over the entire space of $S(q, p)$ policies since service times are stochastic. To ease the notation we let $t = \phi(p)$, and optimize over time thresholds. More specifically, we have

$$h(\ell, t) = \mathbb{P}(R \le \ell)\int_0^t x\frac{1}{2s}dx = \mathbb{P}(R \le \ell)\frac{t^2}{2}\frac{1}{2s} = \frac{\ell t^2}{4s}, \qquad t \in [0, 2s],$$

and

$$g(\ell,t)=t\Big[\mathbb{P}(R>\ell)+\mathbb{P}(R\leq\ell)\frac{2s-t}{2s}\Big]+\frac{\ell t^2}{4s}=t-\frac{\ell t^2}{4s},\qquad t\in[0,2s].$$

Also $\rho=s$, which implies that the stability conditions for the two queues are as follows:

- Queue J: $\lambda q\Big(t-\frac{lt^2}{4s}\Big)\leq n_j$,
- Queue S: $\lambda\Big(s-q\frac{lt^2}{4s}\Big)\leq n_s$.

Similarly to the proofs of Propositions 5 and 7, using the Karush-Kuhn-Tucker first-order optimality and complementary slackness conditions we obtain a system of equations that any solution to the (rate-adjusted) staffing, training, and resource allocation problem has to satisfy. After some algebra and exploiting the fact that there exists an optimal solution that has a hierarchical structure, i.e., $q=1$, we conclude that an optimal solution in the case of uniformly distributed service times is $(\bar{n}_j,\bar{n}_s,\ell,q,t)=(s,0,1,1,2s)$, or equivalently

$$(\bar{n}_j,\bar{n}_s,\ell,q,p)=(s,0,1,1,\hat{p}),$$

where $\hat{p}$ satisfies $\phi(\hat{p})=2s$.

Non-Uniform Task Types. Note that optimization problem (9) and consequently its Lagrangian function are written assuming a general task type distribution. Specifying the corresponding Karush-Kuhn-Tucker first-order optimality and complementary slackness conditions for the case where task types are triangularly distributed, we obtain the following system of equations that any candidate optimal solution has to satisfy:

$$\begin{aligned}
&\bar{n}_j+\bar{n}_s=s,\\
&-\big(1-c-(1-c)\ell\big)k(\ell)s+(1-c)\gamma+\mu_3=0,\\
&\mu_2=1-c-(1-c)\ell,\\
&\mu_2\big(\mathbb{P}(R>\ell)s-\beta\big)=0,\\
&\mu_3(\ell-1)=0.
\end{aligned}$$

We distinguish between three cases: (i) $0\leq\ell\leq 1/2$; (ii) $1/2<\ell<1$; and (iii) $\ell=1$. We conclude that when task types are known the optimal solution is

$$(\bar{n}_j,\bar{n}_s,\ell)=(0.536s,0.464s,0.634),$$

with associated cost equal to $(0.804+0.196c)s$.

Similarly, the optimization problem and its Lagrangian function corresponding to the case where types are unknown have been written assuming a general task type distribution. We follow exactly the same steps as before: first, we obtain a system of equations that a candidate optimal solution has satisfy by writing down the corresponding KKT first-order optimality and complementary slackness conditions, where task types are distributed according to the particular triangular distribution. Then, we consider separately the cases: (i) $0\leq\ell\leq 1/2$; $1/2<\ell<1$; and $\ell=1$. Exploiting the fact that there exists an optimal solution with $q=1$, we conclude that the optimal solution is

$$(\bar{n}_j,\bar{n}_s,\ell,q,p)=(s,0,1,1,p^*),$$

where p^* satisfies $\phi(p^*) = s$, and the associated cost is equal to s.

Acknowledgments

We thank the Department Editor, Serguei Netessine, an Associate Editor, and two anonymous reviewers for very constructive remarks and suggestions. We are also grateful to Kuang Xu and seminar participants at the London Business School and the 2016 INFORMS conference.

References

Acemoglu, Daron, Mohamed Mostagir, Asuman Ozdaglar. 2016. Managing innovation in a crowd. *Working paper* .

Alizamir, Saed, Francis de Véricourt, Peng Sun. 2013. Diagnostic accuracy under congestion. *Management Science* **59**(1) 157–171.

Anand, Krishnan S., M. Fazıl Paç, Senthil Veeraraghavan. 2011. Quality-speed conundrum: trade-offs in customer-intensive services. *Management Science* **57**(1) 40–56.

Armory, Mor, Shlomo Israelit, Avishai Mandelbaum, Yariv N. Marmor, Yulia Tseytlin, Galit B. Yom-Tov. 2015. On patient flow in hospitals: a data-based queueing-science perspective. *Stochastic Systems* **5**(1) 146–194.

Bassamboo, Achal, Ramandeep S. Randhawa, Assaf Zeevi. 2010. Capacity sizing under parameter uncertainty: safety staffing principles revisited. *Management Science* **56**(10) 1668–1686.

Bell, S.L., Ruth Williams. 2001. Dynamic scheduling of a system with two parallel servers in heavy traffic with resource pooling: asymptotic optimality of a threshold policy. *The Annals of Applied Probability* **11**(3) 608–649.

Bertsekas, Dimitri, Robert Gallager. 1992. *Data Networks*. Prentice-Hall.

Bimpikis, Kostas, Mihalis G. Markakis. 2016. Inventory pooling under heavy-tailed demand. *Management Science* **62**(6) 1800–1813.

Brown, Lawrence, Noah Gans, Avishai Mandelbaum, Anat Sakov, Haipeng Shen, Sergey Zeltyn, Linda Zhao. 2005. Statistical analysis of a telephone call center: a queueing-science perspective. *Journal of the American Statistical Association* **100**(469) 36–50.

Dai, J.G., Wuqin Lin. 2005. Maximum pressure policies in stochastic processing networks. *Operations Research* **53**(2) 197–218.

Dobson, Gregory, Arvind Sainathan. 2011. On the impact of analyzing customer information and prioritizing in a service system. *Decision Support Systems* **51**(4) 875–883.

Dobson, Gregory, Tolga Tezcan, Vera Tilson. 2013. Optimal workflow decisions for investigators in systems with interruptions. *Management Science* **59**(5) 1125–1141.

Ersoz, D., M.S. Yousif, C.R. Das. 2007. Characterizing network traffic in a cluster-based, multi-tier data center. *Proceedings of the* 27^{th} *International Conference on Distributed Computing Systems* .

Garicano, Luis. 2000. Hierarchies and the organization of knowledge in production. *Journal of Political Economy* **108**(5) 874–904.

Gurvich, Itai, Jan A. Van Mieghem. 2014. Collaboration and multitasking in networks: architectures, bottlenecks, and capacity. *Manufacturing & Service Operations Management* **17**(1) 16–33.

Harrison, J. Michael. 1998. Heavy traffic analysis of a system with parallel servers: asymptotic optimality of discrete-review policies. *The Annals of Applied Probability* **8**(3) 822–848.

Harrison, J. Michael. 2000. Brownian models of open processing networks: canonical representation of workload. *The Annals of Applied Probability* **10**(1) 75–103.

Hasija, Sameer, Edieal J. Pinker, Robert A. Shumsky. 2005. Staffing and routing in a two-tier call centre. *International Journal of Operational Research* **1**(1) 8–29.

Kostami, Vasiliki, Sampath Rajagopalan. 2013. Speed-quality trade-offs in a dynamic model. *Manufacturing & Service Operations Management* **16**(1) 104–118.

Lee, Hsiao-Hui, Edieal J. Pinker, Robert A. Shumsky. 2012. Outsourcing a two-level service process. *Management Science* **58**(8) 1569–1584.

Levi, Retsef, Thomas Magnanti, Yaron Shaposhnik. 2016. Scheduling with testing. *Working paper* .

Mandelbaum, Avishai, Alexander L. Stolyar. 2004. Scheduling flexible servers with convex delay costs: heavy-traffic optimality of the generalized $c\mu$-rule. *Operations Research* **52**(6) 836–855.

Massoulié, Laurent, Kuang Xu. 2017. On the capacity of information processing systems. *Forthcoming in Operations Research* .

Mihm, Jürgen, Christoph H. Loch, Dennis Wilkinson, Bernardo A. Huberman. 2010. Hierarchical structure and search in complex organizations. *Management Science* **56**(5) 831–848.

Netessine, Serguei, Gregory Dobson, Robert A. Shumsky. 2002. Flexible service capacity: Optimal investment and the impact of demand correlation. *Operations Research* **50**(2) 375–388.

Paç, M. Fazıl, Senthil Veeraraghavan. 2015. False diagnosis and overtreatment in services. *Working paper* .

Shumsky, Robert A., Edieal J. Pinker. 2003. Gatekeepers and referrals in services. *Management Science* **49**(7) 839–856.

Stouras, Konstantinos I., Karan Girotra, Serguei Netessine. 2016. First ranked first to serve: Strategic agents in a service contest. *Working paper* .

Sun, Zhankun, Nilay Tanık Argon, Serhan Ziya. 2014. Priority scheduling of jobs with unknown types. *Working paper* .

Tassiulas, Leandros, Anthony Ephremides. 1992. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multi hop radio networks. *IEEE Transactions on Automatic Control* **37**(12) 1936–1948.

Tezcan, Tolga, J.G. Dai. 2010. Dynamic control of n-systems with many servers: asymptotic optimality of a static priority policy in heavy traffic. *Operations Research* **58**(1) 94–110.

Wang, Xiaofang, Laurens G. Debo, Alan Scheller-Wolf, Stephen F. Smith. 2010. Design and analysis of diagnostic service centers. *Management Science* **56**(11) 1873–1890.