Controlling the Variability of Capacity Allocations Using Service Deferrals

ANDRES FERRAGUT and FERNANDO PAGANINI, Universidad ORT Uruguay
ADAM WIERMAN, Caltech, USA

Ensuring predictability is a crucial goal for service systems. Traditionally, research has focused on designing systems that ensure predictable performance for service requests. Motivated by applications in cloud computing and electricity markets, this article focuses on a different form of predictability: predictable allocations of service capacity. The focus of the article is a new model where service capacity can be scaled dynamically and service deferrals (subject to deadline constraints) can be used to control the variability of the active service capacity. Four natural policies for the joint problem of scheduling and managing the active service capacity are considered. For each, the variability of service capacity and the likelihood of deadline misses are derived. Further, the paper illustrates how pricing can be used to provide incentives for jobs to reveal deadlines and thus enable the possibility of service deferral in systems where the flexibility of jobs is not known to the system *a priori*.

CCS Concepts: • **Mathematics of computing → Queueing theory;** • **Information systems → Data centers;** • **Hardware → Smart grid;**

Additional Key Words and Phrases: Scheduling, deadlines, service variability, incentives

ACM Reference Format:
Andres Ferragut, Fernando Paganini, and Adam Wierman. 2017. Controlling the variability of capacity allocations using service deferrals. ACM Trans. Model. Perform. Eval. Comput. Syst. 2, 3, Article 15 (August 2017), 27 pages.
DOI: http://dx.doi.org/10.1145/3086506

1. INTRODUCTION

Controlling variability in service systems is of crucial importance when the goal is to provide predictable performance. As such, there is a large literature that seeks to characterize and optimize the variability (and thus predictability) of service systems through the design and analysis of resource allocation policies. This literature has provided analysis of the variability of response time and queue occupancy, for example, see the surveys in Kleinrock [1975] and Harchol-Balter [2013], as well as more detailed analysis of the tail of the response time and queue length distributions, for example, see Boxma and Zwart [2007], Wierman and Zwart [2012], Nuyens et al. [2008], Mandjes and Zwart [2006], and Borst et al. [2000]. These results have had a remarkable impact

The research in this paper has been partially supported by IADB–Ministerio de Industria y Energía–Uruguay ATN/KF 13883 UR (Component 3) and ANII–Uruguay under grant FSE_1_2014_1_102426. Additional funding for this work was provided by the NSF through grants CNS-1319820, CNS-1518941, and CPS-1545096 as part of the NSF/DHS/DOT/NASA/NIH Cyber-Physical Systems Program.
Authors' addresses: A. Ferragut and F. Paganini, Universidad ORT Uruguay. Cuareim 1451, 11100, Montevideo, Uruguay; emails: {ferragut, paganini}@ort.edu.uy; A. Wierman, Computing and Mathematical Sciences, California Institute of Technology, 1200 E. California Boulevard, MC 305-16, Pasadena, CA 91125; email: adamw@caltech.edu.

DOI: http://dx.doi.org/10.1145/3086506

on applications in service systems, communication networks, and cloud computing. A recent example of this impact is the literature on controlling the "tail at scale," which is highlighted in Dean and Barroso [2013] as one of the fundamental challenges for Google's cloud systems.

The focus of this article is on providing predictability in service systems, but our goal is distinct from that of the classic literature mentioned above. Instead of focusing on the predictability of *performance*, we focus on providing acceptable performance (i.e., meeting deadlines), while ensuring predictability of the *active service capacity* in the system.

Concretely, we study a setting where jobs that are potentially *deferrable* (i.e., subject to a deadline constraint larger than its service time) arrive to a service system that has the ability to *dynamically scale the active capacity* quickly and efficiently. The caveat is that the system incurs a cost associated with the variability of the active service capacity. Thus, the system could choose to scale capacity such that every job can be served quickly (immediately upon arrival), but this would incur significant expense due to variability in the active service capacity. Instead, the system would be better off strategically deferring the service of (some) jobs, while still ensuring deadlines are met, to smooth the active service capacity. The task of designing a policy for deferring jobs that minimizes the variability of the active service capacity subject to meeting job deadlines is delicate and challenging and is our focus in this article.

The discussion above highlights an inherent tradeoff between optimizing the performance of jobs and maintaining a flat, predictable active service capacity. This tradeoff is increasingly common in service systems as the ability to dynamically adjust the active service capacity emerges. For example, this tradeoff is fundamental when performing dynamic capacity provisioning in cloud systems. Startups such as Dropbox and Netflix, which run on top of cloud computing providers such as Amazon EC2 and Microsoft Azure, typically have the ability to scale computing resources to match demand, which is highly nonstationary. They contract compute instances through a mixture of long-term and real-time (on-demand) purchases. Long-term contracts are typically much cheaper, and so the bulk of compute resources are purchased there and variability is handled through the (more expensive) on-demand market purchases. Thus, limiting the variability of capacity has a direct impact on the financial bottom-line (see Zhu and Agrawal [2010], Vaquero et al. [2011], Liu et al. [2011], and Lin et al. [2013]). In particular, in recent work (Adnan et al. [2012] and Adnan and Gupta [2014]), the use of deferrals has been proposed to decrease variability in service capacity in data-center environments. Another example where variability is an issue is the case of distributed energy resources, such as electric vehicle charging stations. These resources often participate in electricity markets for ancillary services, where there is a direct financial incentive for controlling variability—participants in ancillary service markets are charged for the deviations of electricity demand (service capacity) around a target operating point. Thus, controlling the active capacity is crucial for distributed energy resources that wish to participate in these markets. See Sortomme and El-Sharkawi [2012], Quinn et al. [2010], Chen et al. [2014], and Tomić and Kempton [2007] for more details.

1.1. Contributions of this Article

In this article, we introduce queueing models that incorporate service deferral as a means to control service capacity variability. Our objective is to devise *simple* and/or *decentralized* policies for managing jobs in such a way that overall service capacity remains as constant as possible, while at the same time guaranteeing job performance in the sense of meeting their deadlines. The complexity we wish to avoid is in keeping centralized detailed information of individual job properties or having scheduling commands that must micro-manage individual jobs.

In Section 2, we set up a queueing model in which service deferral is specified by a scalar parameter, the *service level*. Jobs arrive with an individual demand for capacity as in an M/G/∞ queue, but service is deferred by scaling down capacity to a fraction u of the demand. The main issue is how the choice of this fraction impacts both the variability of active service capacity and the ability of jobs to meet their deadlines.

In Section 3, we analyze two implementations for which the service level u has global meaning. The simplest, *Equal Service* policy distributes the available capacity uniformly among all jobs present, scaling down all job capacities to this fraction. The more complex *Least Laxity First* (LLF) policy provides maximal capacity to a fraction u of jobs, choosing those with smallest laxity, that is, spare time. Interestingly, under exponential service times, the variability of the service capacity is the same under the two policies. However, LLF significantly outperforms Equal Sharing in terms of the probability of keeping job deadlines. The downside to this improved performance is the complexity required to implement LLF: indeed, a centralized scheduler must be aware of all current job deadlines and make individualized decisions on service for each job.

In Section 4, we explore two more alternatives that meet our simplicity and decentralization requirement and, furthermore, are guaranteed to strictly meet all job deadlines. *Expiring Laxity* extends the Equal Service policy to the setting of hard deadlines, applying the deferred service only to jobs with remaining laxity. Jobs aware of the laxity expiration can request maximum capacity at this moment, providing a decentralized means to ensure deadlines. The service variability obtained under this policy is shown to be, not surprisingly, larger than the Equal Service. The other alternative explored is *Exact Scheduling*; here, each job receives upon arrival the capacity necessary to complete the job exactly at its deadline. While this implies service levels are individualized, the policy is also amenable to decentralization, since jobs are aware of their own deadlines. Again, we analyze variability and find that it improves with respect to Expiring Laxity, although it is still worse than Equal Service or LLF.

To achieve such decentralized implementation, the preceding strategies must relay the control at least in part to job owners themselves. The question arises as to what incentives users would have to participate in this deferral program. For instance, a job can extract a higher rate simply by declaring no laxity, or equivalently selecting the maximum capacity in Exact Scheduling. Motivated by this, Section 5 focuses on how to design a pricing mechanism that can provide incentives for jobs to contribute with their laxity to the reduction of aggregate variability. We propose for the system manager to offer a monetary reward (discount) to customers that offer flexibility through revealing deadlines that allow flexibility in allocation of service capacity. Our results in this section provide a characterization of how to design such discounts to minimize a combination of service variability and cost (from the payment of the discounts). Our results provide pricing mechanisms under both the assumption that a model of aggregate customer response is known and in a model-free setting, where the customer responses must be learned.

Partial results leading up to this article were presented in Ferragut and Paganini [2015].

1.2. Related Work

The task of understanding (and minimizing) the variability of job performance in queueing systems is classical. At this point, results are known in very general settings for a wide variety of scheduling policies. However, the goal of this article is different—our focus is on controlling the variability of the active service capacity, rather than the variability of job performance. As explained before, this is motivated by work in applications such as cloud computing and electricity markets, where capacity can be scaled dynamically, but there is a significant cost associated with the resulting variability of the capacity.

Within the literature on service systems, there is a large classic literature on scheduling jobs with deadlines; for example, see Pinedo [1983], Lehoczky [1997], Bhattacharya and Ephremides [1989]. Many of the policies we consider in this article have appeared in various settings. For example, our analysis on LLF builds on the early work of Hong et al. [1989], and our analysis on Exact Scheduling builds on extending classical $M/G/\infty$ results [Robert 2003; Zachary 2007]. In more recent articles, the problem of dynamically controlling service rate or admission in queues with deadlines has been analyzed [Plambeck et al. 2001; Maglaras and Mieghem 2005; Çelik and Maglaras 2008] by using fluid and diffusion approximations of the underlying queueing model [Gromoll and Kruk 2007]. In this regard, we emphasize that the queueing literature has been dominated by the analysis of fixed capacity systems, where deadlines are the main issue. In our article, we focus on dynamically scaled systems, and as such new tradeoffs appear.

In this line of work, it is worth mentioning some direct precedents: for instance, the recent works by Nayyar et al. [2013] and Subramanian et al. [2013], which analyze the use of LLF scheduling for an aggregate of residential power loads with focus on variability control, providing a direct motivation for the analysis here. In the context of cloud computing, Adnan et al. [2012] and Adnan and Gupta [2014] also propose to use job deferrals to control variability of power usage in data center, also with renewable energy considerations. The main contribution of this article over such works is giving a queueing perspective to the analysis.

Additionally, the results focused on the design of pricing mechanisms for extracting information about job deadlines are related to the literature on strategic queues. A classical article in this regard for fixed service capacity is Mendelson and Whang [1990]. Surveys of this literature can be found in Hassin and Haviv [2003] and Hassin [2016]. The relationship between deadlines (lead-time), service delay, and pricing mechanisms has been analyzed in Afèche and Mendelson [2004], Akan et al. [2012], Afèche and Pavlin [2016]. However, the context considered in this article is novel due to the scalable resources.

Finally, our focus in this work is on understanding the inherent tradeoff between job performance and service capacity variability, rather than on any specific application. We do, however, take motivation from two specific areas where this issue manifests strongly: cloud computing and regulation services in power systems. The issue of controlling variability in these contexts has been studied in depth in recent years, for example, Zhu and Agrawal [2010], Vaquero et al. [2011], Liu et al. [2011], Adnan et al. [2012], Lin et al. [2013], Adnan and Gupta [2014], and Tomić and Kempton [2007], Quinn et al. [2010], Sortomme and El-Sharkawi [2012] and Chen et al. [2014], respectively.

2. A QUEUEING MODEL WITH DEFERRABLE SERVICE

To study the tradeoff between service capacity variability and job performance, we consider a service system where jobs arrive as a random process. The service system can serve jobs in parallel and directs each job to a server immediately upon arrival. The system controls the service capacity allocated to each job, under the constraint that the allotted service capacity cannot exceed a *nominal* value, p_0, in units of work/second. The *active* capacity may, however, be smaller, corresponding to a fraction $u_k \in (0, 1)$ of the nominal capacity, which we refer to as the *service level* for job k. If $u_k < 1$, then job k is being throttled down to a lower service level, and thus its completion time is *deferred*.

Job requests are assumed to arrive as a Poisson process of intensity λ, requiring a random amount of work that, if served at nominal capacity, would result in a nominal service time denoted by σ_k. Deferral is allowed but is constrained by the *deadline* of the job, d_k, which is assumed to exceed σ_k. We denote by T_k the actual service time attained by job k.

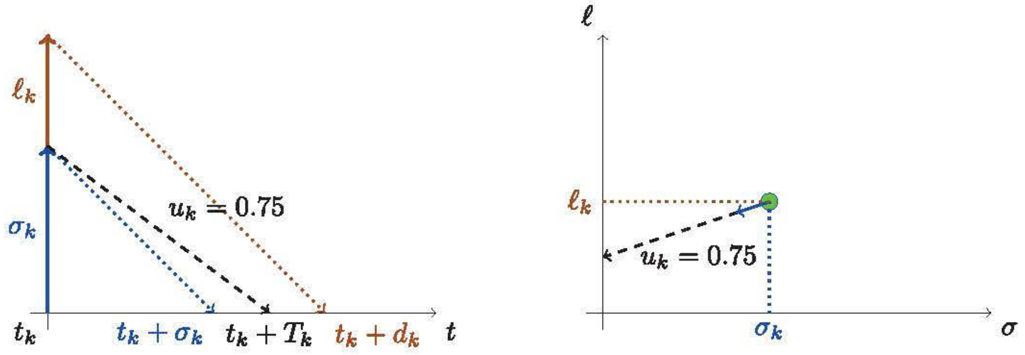


Fig. 1. Service-laxity tradeoff.

For ease of notation, we introduce the concept of the *laxity* of a job, $\ell_k = d_k - \sigma_k$, which is the amount of spare time or slackness the job has upon arrival. The interpretation of ℓ_k is that, if the job were to receive no service at all during this amount of time, then service must be provided at full rate afterward to meet the deadline. Thus, service at partial capacity results in a gradual consumption of laxity and, if the laxity becomes negative at any point in time, the job will miss its deadline. To illustrate the model, consider the following example.

Example 2.1 (*The Evolution of Laxity*). Suppose a job arrives to the system at time t_k with nominal service time σ_k, laxity ℓ_k. Further, suppose that the system allocates to the job a fixed fraction u_k of the nominal capacity.

Figure 1 gives two representations of the evolution. The first uses a dashed line to represent the evolution of the job's residual service time; the lower dotted line represents the lower bound imposed by the nominal (maximum) capacity, and the upper boundary represents the evolution of the deadline horizon. By serving at a fraction of the maximum speed both service time and laxity are consumed, at respective rates u_k and $1 - u_k$; the net service time will be $T_k = \sigma_k/u_k$, and thus the choice of u_k directly determines whether the deadline will be met.

The second graph of Figure 1 plots the trajectory in the (service time, laxity) space. Here, u_k determines the slope of the trajectory, ranging from a horizontal in the case of full service rate, to a vertical in the case of no service. Reaching $\sigma = 0$ indicates service completion, reaching $l = 0$ before that implies a missed deadline.

Another important concept is the individual *relative laxity*, which we define as follows for each job k:

$$\delta_k := \frac{\ell_k}{\sigma_k}. \tag{1}$$

The relative laxity specifies the individual job's tolerance for delay relative to the nominal service time.

An important measure of the system's flexibility is given by the ratio of average laxity versus average service time, that is,

$$\Delta := \frac{E[\ell_k]}{E[\sigma_k]}, \tag{2}$$

and will be called the *deferrability factor* of the job profile. (Note that $\Delta \neq E[\delta_k]$ except when jobs are deterministic). For the initial portions of this article, we assume that σ_k, ℓ_k are independent random variables, with $E[\sigma_k] = 1/\mu$ and $E[\ell_k] = 1/\gamma$. In this case, the deferrability factor is given by $\Delta = \frac{\mu}{\gamma}$. In Section 5, we discuss this assumption further when considering user incentives.

Finally, we use $n(t)$ to denote the number of jobs present at time t. Since each job k receives service capacity $p_0 u_k$, the overall active service capacity is thus

$$p(t) = p_0 \sum_{k=1}^{n(t)} u_k.$$

Note that if all the jobs stay in the system until service is completed (i.e., there are no abandonments or blocking), then the average active capacity of the system should be equal to the average arrival rate times the average work requirement:

$$\bar{p} = \lambda E[p_0 \sigma_k] = p_0 \frac{\lambda}{\mu}, \tag{3}$$

and this quantity is independent of any decision on job deferral or scheduling.

System objectives: We can now formally state the goals of the system operator, which highlight the tradeoff between job performance and variability in service capacity.

(1) Serve all the jobs within deadlines with high probability, that is, keep $P(T_k > d_k)$ small.
(2) Maintain a smooth profile of active service capacity: if $\delta p(t) := p(t) - \bar{p}$, then in steady state one should have $E[(\delta p)^2]$ small.

To highlight that these system objectives are in direct conflict, we consider a simple, motivating example in the following.

2.1. Equal Service Policy

The equal service policy is perhaps the most natural policy to consider first. It offers a constant, *homogeneous* service level $u \in (0, 1]$ (fraction of nominal capacity) applied to all jobs present in the system. To begin the analysis, note that the aggregate active capacity simplifies to

$$p(t) = p_0 n(t) u. \tag{4}$$

We can impose the steady-state condition Equation (3), which then yields

$$\bar{p} = p_0 \frac{\lambda}{\mu} = E[p] = E[p_0 n u].$$

This implies that, for a fixed service level u, the expected number of jobs in the system is

$$\bar{n} = \frac{\lambda}{u\mu}. \tag{5}$$

Next, we can invoke Little's law to obtain the mean time a job spends in the system, $\bar{T} = E[T_k]$:

$$\bar{T} = \frac{\bar{n}}{\lambda} = \frac{1}{u\mu} = \frac{E[\sigma_k]}{u}.$$

To meet deadlines in an average sense, we should have $\bar{T} < E[\sigma_k + \ell_k]$, that is,

$$u > \frac{E[\sigma_k]}{E[\sigma_k] + E[\ell_k]} = \frac{1}{1 + \Delta}. \tag{6}$$

Here, $\eta := \frac{1}{1+\Delta}$ is a minimum service level that jobs should receive to meet their deadlines on average. As deferrability increases, $\eta \to 0$ and the system gains in flexibility, meaning that jobs arrive with larger slack time. In the following sections, we expand on this issue with a more precise measure, the *missed deadline probability* $\alpha := P(T_k > \sigma_k + \ell_k)$.

On the other hand, let us observe the impact of lowering the service level on system variance. Since all requests are served in parallel, the system behaves as an infinite server ($M/G/\infty$) queue, with arrival rate λ and average service time $E[T_k] = 1/(u\mu)$. In particular, the number of jobs in steady state satisfies

$$n \sim \text{Poisson}\left(\frac{\lambda}{u\mu}\right). \tag{7}$$

In particular, this is consistent with the mean value $\bar{n}$ found in Equation (5). However, we can also use the distribution to compute the *variance* of active capacity,

$$E[(\delta p)^2] = E[(p - \bar{p})^2] = p_0^2 u^2 \text{Var}(n) = p_0^2 u^2 \frac{\lambda}{u\mu} = p_0 \bar{p} u. \tag{8}$$

A more normalized way of expressing variability is the *coefficient of variation* $cv^2(p)$ defined by

$$cv^2(p) = \frac{\text{Var}(p)}{\bar{p}^2},$$

which can be readily computed from Equations (3) and (8) to yield

$$cv^2(p) = \frac{p_0}{\bar{p}} u. \tag{9}$$

This illustrates that variability reduces linearly with the service level u. From this perspective there is an incentive toward service deferral, which must be balanced against the likelihood of missed deadlines. In the following, we investigate this tradeoff for a variety of different scheduling and deferral strategies with a strong focus on simple policies that are amenable to implementation and decentralization.

3. TRADING OFF VARIABILITY AND JOB DEADLINES

The Equal Service policy considered above serves as the starting point for our exploration of different strategies to trade off job performance and service variability. In this section, we initially provide with a more precise analysis of this tradeoff under Equal Service, starting with a quantification of the baseline performance of Equal Service in terms of meeting deadlines.

We then move to considering other policies. This exploration is motivated by the observation that the same aggregate service rate $p(t)$ can be obtained if, instead of serving all jobs at a fraction u of nominal rate, we decide to serve a fraction u of jobs at full rate. This poses a crucial question of *scheduling*, that is, which jobs to serve. This question opens the door for different policies that share the variability of Equal Service but may perform better in meeting job deadlines.

A simple scheduling strategy is to pick such a subset at random, which can be seen to perform similarly to Equal Service. Clearly there are better options, however. A more sophisticated alternative is *least-laxity-first (LLF)* scheduling, studied classically in Hong et al. [1989]. LLF serves the jobs with least spare time remaining first, up to the aggregate capacity determined by the service level.

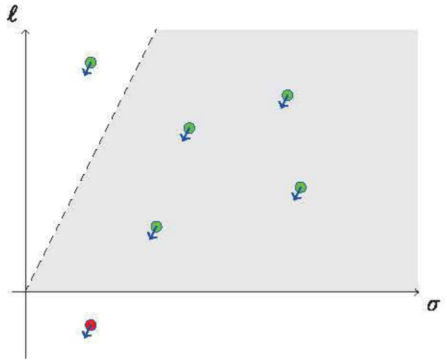


Fig. 2. Equal service scheduling for $u = 1/3$. Arrivals in the shaded area are bound to miss their deadline.

3.1. Missed Deadlines in Equal Service

We now carry out a more precise analysis of the Equal Service policy, for which every job is throttled to a reduced rate $p_0 u$. We have already shown that the number of jobs $n(t)$ in steady state follows the Poisson distribution Equation (7), and its variance is given by Equation (9).

We now turn our attention to deadline misses. The probability of missing a deadline is

$$\alpha = P\left(T_k > \sigma_k + \ell_k\right) = P\left(\frac{\sigma_k}{u} > \sigma_k + \ell_k\right) = P\left(\frac{\sigma_k}{u} > \frac{\ell_k}{1-u}\right). \tag{10}$$

This equation can be interpreted as follows: when a job arrives with service time σ_k, initial laxity ℓ_k, and is served at rate u, then after a time dt the remaining service time will be $\sigma' = \sigma_k - udt$. Since its deadline is $\sigma_k + \ell_k - dt$ time units ahead, the remaining laxity after a time dt is

$$\ell' = \sigma_k + \ell_k - dt - (\sigma_k - udt) = \ell_k - (1-u)\,dt.$$

This means that for service level u, laxity is consumed at rate $1-u$, and Equation (10) simply states that laxity is consumed before service. A depiction of the equal service policy in the service-laxity space is given in Figure 2: all jobs present in the system consume service and laxity in certain fixed proportions, therefore points move following the same vector. Arrivals in the shaded area are bound to miss their deadline.

Assume now that (σ_k, ℓ_k) are exponentially distributed; then α can be readily calculated by observing that $\frac{\sigma_k}{u} \sim \exp(u\mu)$ and $\frac{\ell_k}{1-u} \sim \exp((1-u)\gamma)$. Using the minimum of two exponential random variables, we have

$$\alpha = \frac{\gamma(1-u)}{\mu u + \gamma(1-u)} = \frac{(1-u)}{\Delta u + (1-u)}. \tag{11}$$

Deadline misses are decreasing in u, as expected. In particular, for $u = \eta = \frac{1}{1+\Delta}$, which results from the previous analysis in the mean, we have $\alpha = 1/2$.

Analogous calculations can be performed for any joint distribution in (σ_k, ℓ_k). For comparison purposes, we compute also the probability for deterministic service time $\sigma_k \equiv \frac{1}{\mu}$ and exponential laxity, which yields

$$\alpha^{ds} = P\left(\frac{1}{\mu u} > \frac{\ell_k}{1-u}\right) = 1 - e^{-\frac{1}{\Delta}\frac{1-u}{u}}.$$

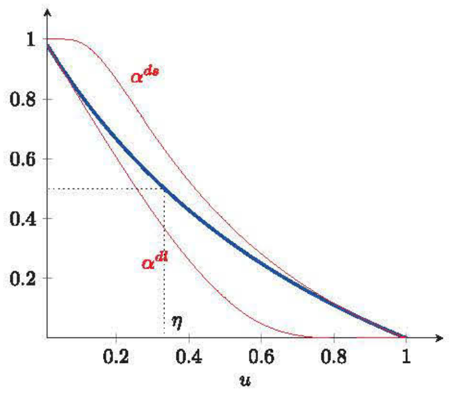


Fig. 3. Missed deadline probability as a function of u for $\Delta = 2$.

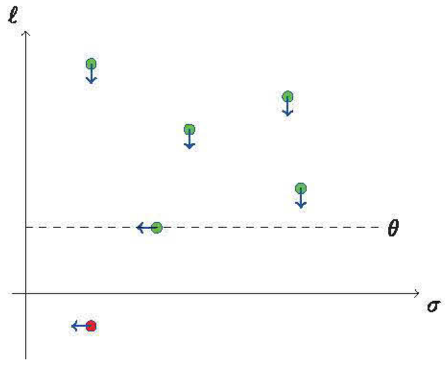


Fig. 4. LLF scheduling for $u = 1/3$ and frontier level θ.

For deterministic laxity $\ell_k \equiv \frac{1}{\gamma}$ and exponential service time, the corresponding expression is

$$\alpha^{dl} = P\left(\frac{\sigma_k}{u} > \frac{1}{\gamma(1-u)}\right) = e^{-\Delta\frac{u}{1-u}}.$$

The three cases are depicted in Figure 3 for a deferrability parameter of $\Delta = 2$. As we can see, deadline misses are rather high even for moderate values of $u > \eta$. This provides a baseline for contrasting with more sophisticated policies such as LLF.

3.2. Least Laxity First (LLF)

Least Laxity First (LLF) can be formally defined as follows: sort the current jobs by increasing laxity and serve the first $k(t) = \lfloor n(t)u \rfloor$ at the nominal rate. The remaining jobs consume laxity until they are scheduled. This policy was introduced in the context of processor time scheduling by Hong et al. [1989], and has been thoroughly analyzed in the case of single-server queues (see Gromoll and Kruk [2007] for a recent treatment). The difference here is that we are dealing with an infinite server system.

A depiction of the LLF policy in service-laxity space is given in Figure 4. It is convenient to define the *frontier process* as follows:

$$\theta(t) := \sup\left\{\ell : \sum_{k=1}^{n(t)} \mathbf{1}_{\{\ell_k \leqslant \ell\}} < n(t)u\right\}. \quad (12)$$

The frontier process, $\theta(t)$, represents the maximum laxity of the jobs currently in service. Jobs with laxity greater than $\theta(t)$ only consume laxity and are not served.

We now move to the analysis of LLF, with focus on the case of exponentially distributed service times σ_k. Our first proposition shows that under this assumption the occupancy process is the same as an equal service policy.

PROPOSITION 3.1. *Under the LLF policy and exponential service times, the total occupation process $n(t)$ is a birth-death process with birth rate λ and death-rate μnu, and thus is equal in distribution to the occupation process of the equal service policy.*

PROOF. Under LLF, at any time t there are $n(t)u$ jobs in service. Due to the memoryless property of the exponential distribution and the fact that laxities are independently chosen, the service process of the system for occupation state $n(t)$ and service level u corresponds to $n(t)u$ exponential servers in parallel. Therefore, the total population evolves as a birth-death process with birth rate λ and death rate μnu, which are the rates of an $M/M/\infty$ with individual service level u as in the equal service policy. □

From this proposition, we can conclude that, in steady state, $n \sim$ Poisson $\left(\lambda/(\mu u)\right)$, the average system occupation is again $\bar{n} = \lambda/(\mu u)$, and the output variance is again given by Equation (9), that is, it is linear in u. This was already observed empirically in Bliman et al. [2015].

To continue our analysis, we focus on the case when the scale is large ($\lambda \to \infty$). To illustrate the system's behavior, we plot in Figure 5 two simulation experiments for a system with $\lambda/\mu = 500$ and $\Delta = 2$ ($\eta = 1/3$). In the first case $u = 0.5 > \eta$ and the frontier process $\theta(t)$ finds a positive equilibrium θ^*. Jobs arriving with laxity greater that θ^* consume laxity down to level θ^* and then they are served, while jobs with laxity less than θ^* are served upon arrival. In practice, most jobs get served before their deadlines expire.

In the second case, with $u = 0.2 < \eta$, the system finds an equilibrium value of $\theta^* < 0$. In this case, jobs consume all their laxity and are further delayed an amount $|\theta^*|$ before receiving service. In particular, this means that deadlines are not being honored.

By applying Little's law to our analysis of the occupancy of LLF, we can characterize the equilibrium value θ^* through a fixed-point analysis. Recall that, from Proposition 3.1, the average number of clients in the system is $\bar{n} = \lambda/(\mu u)$. Therefore, the average time in the system by Little's law is $\bar{T} = 1/(\mu u)$. Also, since jobs are not served up to reaching laxity level θ^*, we can compute the average time in the system as

$$\bar{T} = E[(\ell_k - \theta^*)^+] + E[\sigma_k].$$

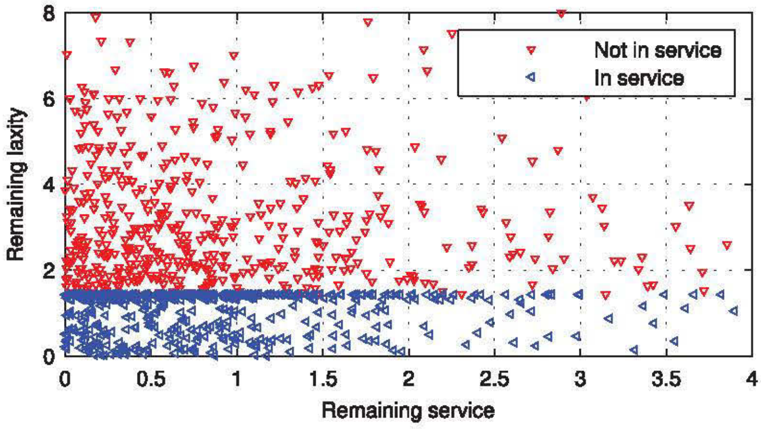
The first term simply states that jobs arriving with laxity greater that θ^* should wait to consume their laxity up to level θ^* before being served. By combining the preceding equations and using that $E[\sigma_k] = 1/\mu$, we have that θ^* should satisfy

$$E[(\ell_k - \theta^*)^+] = \frac{1}{\mu}\frac{1-u}{u}. \tag{13}$$

We have the following characterization:

PROPOSITION 3.2. *For $u \in (0, 1)$, the value of θ^*, solution of Equation (13), is unique and satisfies $\theta^* > 0 \Leftrightarrow u > \eta = \frac{1}{1+\Delta}$.*

PROOF. Consider the function $g(\theta) = E[(\ell_k - \theta)^+]$. For fixed ℓ, the function $(\ell - \theta)^+$ is decreasing in θ and strictly decreasing if $\theta < \ell$. By taking expectations, we conclude that $g(\theta)$ is also decreasing and is strictly decreasing in θ provided $P(\ell_k > \theta) > 0$. If the distribution of ℓ_k has upper bounded support, then $g(\theta) = 0$ for θ beyond this bound.



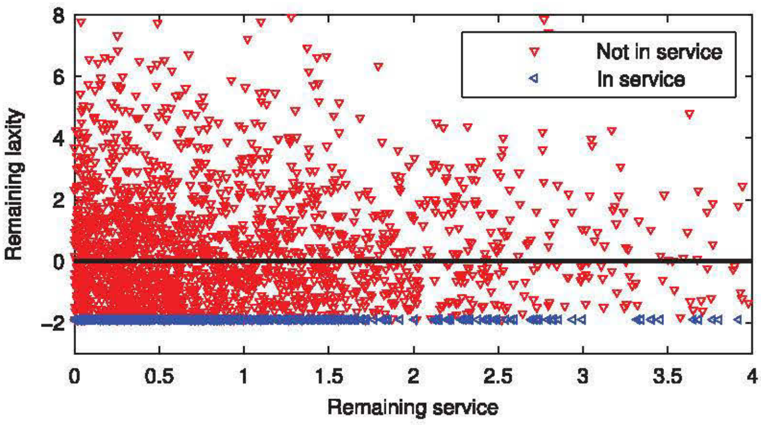


Fig. 5. Remaining service and laxities for LLF when $u > \eta$ (above) and $u < \eta$ (below). The load is $\lambda/\mu = 500$.

Since $\ell_k \geqslant 0$, for $\theta < 0$, $E[(\ell_k - \theta)^+] = E[\ell_k - \theta]$ and thus $g(\theta) = \frac{1}{\gamma} - \theta$. As $\theta \to +\infty$, $g(\theta) \to 0$ by the dominated convergence theorem (and is exactly 0 if ℓ_k is bounded). In conclusion, $g(\theta)$ is strictly decreasing from ∞ up to reaching 0, and therefore Equation (13) admits only one solution for $u > 0$. To prove the second statement, note that $g(0) = \frac{1}{\gamma}$, and thus,

$$\theta^* > 0 \Leftrightarrow \frac{1}{\mu}\frac{1-u}{u} < \frac{1}{\gamma} \Leftrightarrow u > \frac{\gamma}{\mu+\gamma} = \eta. \quad \square$$

As an example, consider the case where $\ell_k \sim \exp(\gamma)$. If $u > \eta$, then Equation (13) becomes

$$\frac{1-u}{\mu u} = \frac{1}{\gamma} e^{-\gamma\theta^*},$$

or equivalently,

$$\theta^* = \frac{1}{\gamma} \log\left[\frac{\Delta u}{1-u}\right]. \tag{14}$$

Note that Equation (14) yields a positive solution provided $u > \eta$, and (asymptotically) all deadlines are achieved.

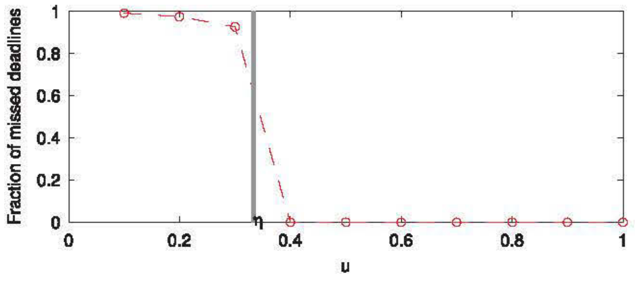


Fig. 6. Empirical missed deadline probability as a function of u for LLF scheduling with $\Delta = 2$ ($\eta = 1/3$) and $\lambda/\mu = 500$.

For the case $u < \eta$, we can solve for

$$\theta^* = \frac{1}{\gamma} - \frac{1}{\mu}\frac{1-u}{u} < 0. \tag{15}$$

Therefore, in steady state, the frontier converges to a negative equilibrium and all deadlines are missed.

This behavior is valid only in the large scale limit, where the frontier process becomes a constant in steady state. However, simulations show that the approximation is indeed good for moderate values of λ, as depicted in Figure 6, where a sharp decline in missed deadline probability is observed around $u = \eta$.

In summary, the main conclusion of this analysis is that, for large-scale systems using LLF, the service level can be reduced up to nearly η (thereby reducing variance), without great impact on deadline misses. Of course, this comes at the cost of using a complex scheduling policy (one that requires knowledge of job laxities). In the next section, we analyze a different class of policies that cope with deadlines in ways that are more amenable to decentralized scheduling.

4. DECENTRALIZED CONTROL OF VARIABILITY SUBJECT TO HARD DEADLINES

The previous section focuses on scheduling policies that allow some deadlines to be missed. Note that the allowance for these soft deadlines provides more flexibility for scheduling and service capacity allocation than if deadlines were strictly enforced (hard deadlines). We now analyze this second case. As we shall see, this still allows the system room to exploit the deferrability of jobs, but clearly there is a more limited ability to defer in this setting. Thus, we should not expect policies here to perform as well as LLF, and indeed they do not. Nevertheless, these policies also require less information from the system state, and are thus more amenable to decentralization, a desirable feature.

In particular, we focus on two specific policies in this setting: *Expiring Laxity* and *Exact Scheduling*. Expiring Laxity is a variation on the Equal Service policy that only throttles jobs that have positive remaining laxity, but ensures that jobs with expired laxity receive the nominal service capacity to meet their deadline. In contrast, Exact Scheduling throttles jobs individually so they complete their jobs exactly when their laxity expires (i.e., exactly at their deadline).

4.1. Expiring Laxity

Expiring Laxity is a simple variation of Equal Service. It operates by applying a fixed service level to jobs with positive remaining laxity. Since this is not an homogeneous service level across all jobs, we denote this quantity by $\tilde{u} \in (0, 1]$. Jobs for which the laxity has already expired are served with the nominal capacity.

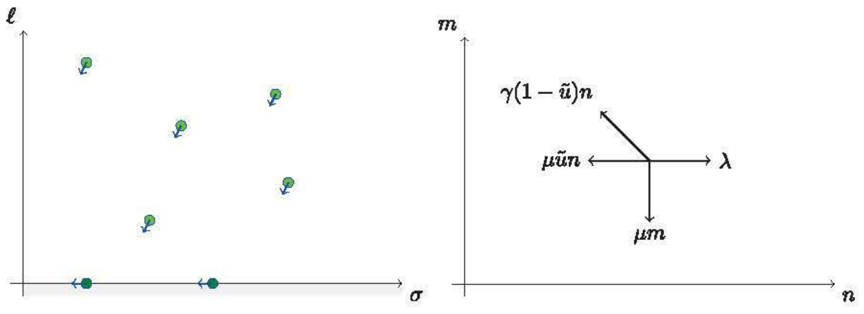


Fig. 7. Laxity expiring scheduling for $u = 1/3$ and Markov chain model.

A depiction of the trajectories of this policy is given on the left in Figure 7. Its main advantage is that it is very easy to decentralize. The system operator fixes a service level $\tilde{u}$ for those jobs that still have laxity and distributes this as a common signal. When a given job reaches the point where it cannot be deferred any longer, it scales up its consumption to the maximum rate.

We again focus on the case of exponential job sizes and laxities, where the analysis is clean. Let $n(t)$ denote the number of jobs with positive laxity and $m(t)$ those whose laxity has expired. Then $(n(t), m(t))$ is a continuous time Markov chain with state space $\mathbb{N}^2$ with the transition rates depicted in Figure 7. The Markov chain has a product form solution given by the following proposition, which can be readily verified by substituting in the global balance equations.

PROPOSITION 4.1. *The equilibrium distribution of the Markov process of the laxity-expiring policy is given by*

$$\pi(n, m) = e^{-\rho_n-\rho_m}\frac{\rho_n^n}{n!}\frac{\rho_m^m}{m!}, \quad n, m \in \mathbb{N}; \tag{16}$$

that is, in steady-state n and m behave as independent Poisson random variables with parameters

$$\rho_n = \frac{\lambda}{\mu\tilde{u} + \gamma(1-\tilde{u})}, \quad \rho_m = \frac{\gamma(1-\tilde{u})}{\mu}\rho_n.$$

To interpret the above, let us now introduce the following parameters:

$$\upsilon := \mu\tilde{u} + \gamma(1-\tilde{u}), \quad \alpha := \frac{\gamma(1-\tilde{u})}{\mu\tilde{u} + \gamma(1-\tilde{u})}.$$

Here, $1/\upsilon$ is the average time before either the laxity or the service of a given job ends, and α is the probability that the laxity expires before the job ends, and thus the job starts service at full rate. Note that α has the same form as the missed deadline probability for the equal sharing policy in the previous section. With this choice of notation, we can rewrite

$$\rho_n = \frac{\lambda}{\upsilon}, \quad \rho_m = \frac{\alpha\lambda}{\mu}.$$

Noting that the total service rate $p(t)$ is $p_0\tilde{u}$ for the first n jobs and the nominal value p_0 for the remaining m jobs, we can compute

$$\bar{p} = E[p] = E[p_0(n\tilde{u} + m)] = p_0\left(\tilde{u}\frac{\lambda}{\upsilon} + \frac{\alpha\lambda}{\mu}\right) = p_0\frac{\lambda}{\mu}\underbrace{\left[\frac{\tilde{u}\mu + \upsilon\alpha}{\upsilon}\right]}_{=1} = p_0\frac{\lambda}{\mu},$$

as expected, since all jobs are eventually served.

We can also quantify the deviations from equilibrium in steady-state as

$$\begin{aligned} E[(\delta p)^2] &= \mathrm{Var}[p_0(n\tilde{u} + m)] \\ &= p_0^2(\tilde{u}^2\mathrm{Var}(n) + \mathrm{Var}(m)) \\ &= p_0^2(\tilde{u}^2\rho_n + \rho_m) \\ &= p_0^2\left(\tilde{u}^2\frac{\lambda}{\upsilon} + \frac{\alpha\lambda}{\mu}\right) \\ &= p_0^2\frac{\lambda}{\mu}\left[1 - \frac{\mu\tilde{u}(1-\tilde{u})}{\upsilon}\right], \end{aligned}$$

where we have used that n and m are independent random variables in steady state due to the product form distribution, and the definitions of υ and α.

Observe that $E[(\delta p)^2] \leqslant p_0^2\frac{\lambda}{\mu}$. This bound is achieved for $\tilde{u} = 0$ or $\tilde{u} = 1$. The case $\tilde{u} = 0$ corresponds to not giving any service until laxity expires, effectively delaying arrival for *all* jobs to the second queue and losing control on deferrability. The case $\tilde{u} = 1$ corresponds to serving the jobs at full rate upon arrival, also abandoning deferrability as in the Equal Service policy with $u = 1$.

Again, it is better to express the variability in normalized units, by computing the coefficient of variation as

$$cv^2(p) = \frac{E[(\delta p)^2]}{\bar{p}^2} = \frac{p_0}{\bar{p}}\left[1 - \frac{\Delta\tilde{u}(1-\tilde{u})}{\Delta\tilde{u} + (1-\tilde{u})}\right]. \tag{17}$$

The above expression is minimized at

$$\tilde{u}^* = \frac{1}{1+\sqrt{\Delta}}, \tag{18}$$

and the minimal value of $cv^2(p)$ for this policy is

$$cv^2(p)\,|_{\tilde{u}=\tilde{u}^*} = \frac{p_0}{\bar{p}}\left[1 - \frac{1}{(1+\sqrt{1/\Delta})^2}\right].$$

Note that both the optimal value of $\tilde{u}$ as well as the ratio between optimal and maximal variance do not depend on the arrival rate and only on the deferrability factor.

4.2. Exact Scheduling

The final policy that we discuss is Exact Scheduling, which aims at finishing all jobs *exactly* at their deadline by tailoring the individual service level. Namely, for a job with service time σ_k and laxity ℓ_k, the service level is chosen to be

$$u_k = \frac{\sigma_k}{\sigma_k + \ell_k}.$$

This choice implies that job k spends a time

$$T_k = \frac{\sigma_k}{u_k} = \sigma_k + \ell_k$$

in the system, that is, it departs exactly at its deadline. A depiction of the policy is given in the first graph of Figure 8. We remark here that this policy is designed with decentralized implementation in mind: provided that jobs can throttle their service level, they can self tune it to the appropriate value, since job requests are already aware of their respective service time and deadline.

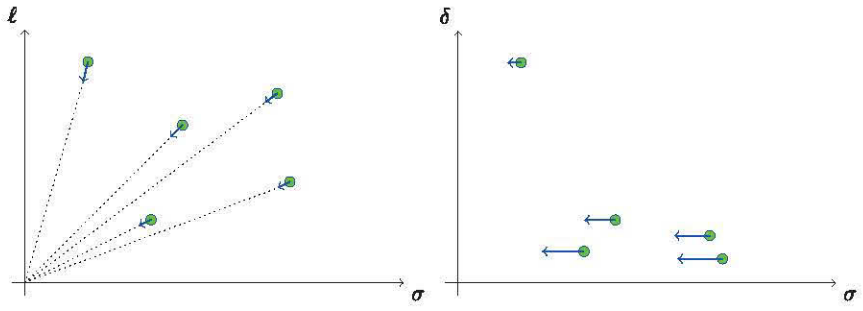


Fig. 8. Exact scheduling depicted in service-laxity space and in the service-relative laxity coordinates.

To analyze this policy it is more convenient to work with the job's *relative* laxity, $\delta_k = \ell_k/\sigma_k$, and to express the service level as

$$u_k = \frac{1}{1+\delta_k}.$$

In the coordinates (σ, δ) the service trajectory follows horizontal lines, with δ fixed throughout service, as depicted in the second graph of Figure 8. This is due to the fact that the job moves toward the origin in the service-laxity space, thus maintaining a constant ratio between remaining service time and laxity. The speed of service (horizontal motion in the second graph) is decreasing with the vertical coordinate δ.

Since deadlines are strictly enforced, our objective is simply to compute the steady-state variance of active capacity $E[(\delta p)^2]$ for this system. In this policy, we can carry out the analysis under more general assumptions. As before, we assume arrivals over time form a Poisson process, but no longer require exponential service times and laxities. Instead, assume arrival k carries a service time-relative laxity (σ_k, δ_k), distributed over the positive orthant with a joint density function $g(\sigma, \delta)$. Note that if the model is specified via a joint density $f(\sigma, l)$ over service time and absolute laxity, the change of variables $\ell = \sigma\delta$ with Jacobian

$$\left|\frac{\partial(\sigma, l)}{\partial(\sigma, \delta)}\right| = \sigma \tag{19}$$

leads to the joint density $g(\sigma, \delta) = \sigma f(\sigma, \sigma\delta)$ in the desired variables.

The state of the system at time t can be expressed through the following counting measure, as in Robert [2003] (Chapter 8):

$$\Phi_t(B) = \sum_{i=1}^{n(t)} \mathbf{1}_{\{(\sigma_i, \delta_i) \in B\}}, \tag{20}$$

where σ_i represents the remaining service of job i and δ_i its relative laxity,

With the above definitions, Φ_t is a measure-valued Markov process with the following dynamics: new points arrive as a Poisson process, and a new point (σ_k, δ_k) is chosen with joint density $g(\sigma, \delta)$. Each point in the system moves to the left at a fixed rate $1/(1+\delta)$, determined by the mark δ_k. Since all jobs are served in parallel, the system behaves as a mixture of infinite server queues of different speeds. Applying results on the steady-state characteristics of the $M/G/\infty$ queue (see Robert [2003] and Zachary [2007]), we obtain:

THEOREM 4.2. *Under the exact scheduling policy with requests distributed as $g(\sigma, \delta)$, the distribution of Φ_t in steady state is a Poisson point process on $\mathbb{R}^+ \times \mathbb{R}^+$ with mean* *measure density*

$$h(\sigma,\delta)=\lambda\int_{\sigma}^{\infty}(1+\delta)g(z,\delta)dz. \tag{21}$$

As a consequence of this result, average characteristics of the steady-state process can be computed by suitable integrals with respect to the above density; we refer to Baccelli and Błaszczyszyn [2009] for these derivations. To summarize, the mean number of jobs present in the system is

$$\bar{n}=E[n]=\int_0^{\infty}\int_0^{\infty}h(\sigma,\delta)d\sigma d\delta. \tag{22}$$

The expected service rate of the jobs is given by

$$E[p]=p_0E\left[\sum_i\tfrac{1}{1+\delta_i}\right]=p_0\int_0^{\infty}\int_0^{\infty}\frac{1}{1+\delta}h(\sigma,\delta)d\sigma d\delta. \tag{23}$$

Finally, the variance of the service rate is given by

$$\mathrm{Var}[p]=p_0^2\mathrm{Var}\left[\sum_i\tfrac{1}{1+\delta_i}\right]=p_0^2\int_0^{\infty}\int_0^{\infty}\frac{1}{(1+\delta)^2}h(\sigma,\delta)d\sigma d\delta. \tag{24}$$

In the case of Equation (23), note that by substitution with Equation (21), we obtain

$$\begin{aligned}\bar{p}=E[p]&=p_0\lambda\int_0^{\infty}d\sigma\int_0^{\infty}d\delta\int_{\sigma}^{\infty}g(z,\delta)dz=p_0\lambda\int_0^{\infty}d\sigma\int_{\sigma}^{\infty}g_1(z)dz\\&=p_0\lambda\int_0^{\infty}[1-G_1(\sigma)]d\sigma=p_0\lambda E[\sigma_k]\\&=p_0\frac{\lambda}{\mu};\end{aligned}$$

here, we have denoted $g_1(\sigma)$ the marginal density of σ_k, $G_1(\sigma)$ its cumulative distribution; thus, our mean service capacity is consistent with Equation (3).

To evaluate the variance, we focus on two special cases: (i) independent exponential service and laxity and (ii) independent service and relative laxity. The first provides us the ability to contrast the variance under all the policies discussed to this point, and the second provides the building blocks for our pricing analysis in Section 5.

4.2.1. Independent, Exponential Service and Laxity. Concretely, in this section, we assume that the joint density of service time and (absolute) laxity is

$$f(\sigma,\ell)=\mu\gamma e^{-\mu\sigma-\gamma\ell},\quad \sigma,\ell>0.$$

By performing the change of variables mentioned in Equation (19), we have

$$g(\sigma,\delta)=\sigma f(\sigma,\sigma\delta)=\mu\gamma\sigma e^{-(\mu+\gamma\delta)\sigma}. \tag{25}$$

Define $\upsilon:=\mu+\gamma\delta$, then the steady-state density is given by computing the integral in Equation (21) to yield

$$h(\sigma,\delta)=\lambda\mu\gamma(1+\delta)\left[\frac{\upsilon\sigma+1}{\upsilon^2}\right]e^{-\upsilon\sigma}.$$

With this density, we can carry out the calculations indicated in Equations (22)–(24). In particular, it is easily checked that

$$\int_0^{\infty}h(\sigma,\delta)d\sigma=\frac{2\lambda\mu\gamma(1+\delta)}{\upsilon^3}=2\lambda\mu\left(\frac{1}{\upsilon^2}+\frac{\gamma-\mu}{\upsilon^3}\right); \tag{26}$$

integrating now over δ and applying the change of variables $v = \mu + \gamma\delta$ yields

$$\bar{n} = 2\lambda\mu \int_{\mu}^{\infty} \left(\frac{1}{v^2} + \frac{\gamma - \mu}{v^3} \right) \frac{dv}{\gamma} = \frac{2\lambda\mu}{\gamma} \left(\frac{1}{\mu} + \frac{\gamma - \mu}{2\mu^2} \right)$$
$$= \lambda \left(\frac{1}{\mu} + \frac{1}{\gamma} \right). \tag{27}$$

Equation (27) simply states that the average number of customers in the system is the arrival rate λ times the expected service time $E[\sigma_k + \ell_k] = \frac{1}{\mu} + \frac{1}{\gamma}$, consistent with the fact that the system behaves as an $M/G/\infty$ queue. Also, as shown above, in general, we have $E[p] = p_0\lambda/\mu = \bar{p}$.

While service level in this system is job dependent, an *effective* service level can be computed as point of comparison as follows:

$$u_{\text{eff}} := \frac{E[p]}{p_0 E[n]} = \frac{\lambda/\mu}{\lambda(\frac{1}{\mu} + \frac{1}{\gamma})} = \frac{\gamma}{\mu + \gamma} = \frac{1}{1 + \Delta} = \eta,$$

that is, the exact scheduling policy works at a service level comparable to taking $u = \eta$ in the soft deadline policies.

More importantly, using Equations (24) and (26) yields after some calculations the following expression for the steady-state variability:

$$E[(\delta p)^2] = p_0^2 \lambda\mu\gamma \left[-\frac{1}{(\mu - \gamma)\mu^2} - \frac{2}{(\mu - \gamma)^2\mu} + \frac{2}{(\mu - \gamma)^3} \log\left(\frac{\mu}{\gamma}\right) \right]. \tag{28}$$

As before, we can compute the coefficient of variation in terms of the deferrability factor:

$$cv^2(p) = \frac{p_0}{\bar{p}} \left[\frac{1}{1 - \Delta} - \frac{2\Delta}{(1 - \Delta)^2} - \frac{2\Delta^2 \log(\Delta)}{(1 - \Delta)^3} \right]. \tag{29}$$

4.2.2. Independent Service and Relative Laxity. We now move to a different scenario, where the independent stochastic primitives are service time and *relative* laxity δ. These primitives facilitate the analysis of pricing in Section 5.

The basic assumption in this section is that $g(\sigma, \delta) = g_1(\sigma)g_2(\delta)$. In this case, from Equation (21), we have

$$h(\sigma, \delta) = \lambda(1 + \delta)(1 - G_1(\sigma))g_2(\delta),$$

and therefore

$$E[(\delta p)^2] = p_0^2 \lambda \int_0^{\infty} \frac{1}{1 + \delta} g_2(\delta) d\delta \int_0^{\infty} (1 - G_1(\sigma)) d\sigma$$
$$= p_0^2 \lambda E\left[\frac{1}{1 + \delta_k} \right] E[\sigma_k]$$
$$= p_0^2 \frac{\lambda}{\mu} E\left[\frac{1}{1 + \delta_k} \right]$$
$$= p_0 \bar{p} E[u_k]. \tag{30}$$

The above expression is very similar to Equation (8); in that case, the service level u was fixed across jobs. Here, we are adapting to the offered, random u_k of each job, yet we find an analogous expression for the variance of service capacity, replacing u by its mean $E[u_k]$.

An alternative interpretation of exact scheduling is to assume that service level u_k is itself under the control of the jobs, and this quantity is chosen independently of

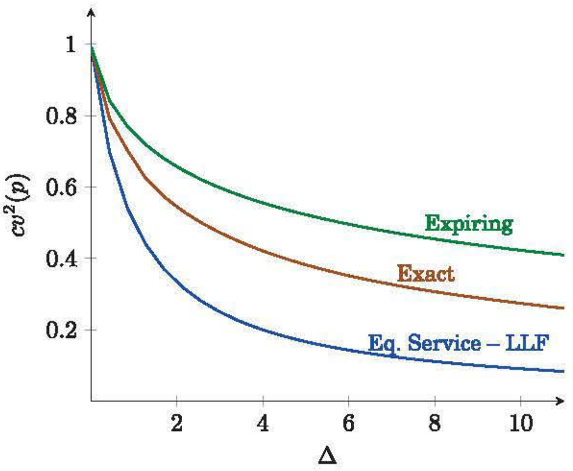


Fig. 9. Normalized coefficient of variation for the different scheduling policies.

service time. What we find is that this offered reduction in nominal capacity translates directly, and linearly, to the variance of the aggregate service capacity.

4.3. Contrasting Scheduling Policies

Before leaving the analysis of scheduling policies, it is useful to summarize the variability results so far. We have studied four policies: Equal Service, Least Laxity First, Expiring Laxity, and Exact Scheduling. For each policy, we considered varying assumptions on laxity and service times, but we provided analysis for all policies in the case of independent, exponential service times and laxities. Thus, our comparison here focuses on that case.

We summarize the cv^2 of each policy in the following. For Equal Service and LLF, we use the value for $u = \eta$, which is the minimum value for which they can comply with deadlines on average (note that the missed deadline probability can be much lower in LLF). For Expiring Laxity, we use the optimal value u^* computed in Equation (18). For Exact Scheduling the variability is determined from Equation (29).

$$cv^2(p)_{ES} = cv^2(p)_{LLF} = \frac{p_0}{\bar{p}} \frac{1}{1+\Delta}, \tag{31a}$$

$$cv^2(p)_{ExpL} = \frac{p_0}{\bar{p}} \left[1 - \frac{1}{(1+\sqrt{1/\Delta})^2} \right], \tag{31b}$$

$$cv^2(p)_{ExS} = \frac{p_0}{\bar{p}} \left[\frac{1}{1-\Delta} - \frac{2\Delta}{(1-\Delta)^2} - \frac{2\Delta^2 \log(\Delta)}{(1-\Delta)^3} \right]. \tag{31c}$$

$$\tag{31d}$$

The above expressions are plotted in Figure 9. A first remark is that variability is always lower if $p_0/\bar{p}$ is small, which in fact is the statistical multiplexing effect of the system scale. The ratio $\bar{p}/p_0$ between average service in the system as a whole and individual service at full rate is, in fact, a measure of the size of the system. Thus, variability can be reduced by aggregation, which is the main point behind cloud computing for instance.

We can further improve the variability by using the jobs laxities. We can see in Figure 9 that, as deferrability increases (Δ grows), the variability goes to 0, with the best policy in this regard being LLF. But, LLF does not guarantee that deadlines are met and requires the system scheduler to have detailed information on the current system state, which makes it challenging to decentralize. Among the hard deadline policies, Exact Scheduling is closest to matching the variance of LLF. Further recall that Exact Scheduling is much simpler and easier to decentralize. Further, it does not miss any deadlines.

5. PRICING INCENTIVES TO CONTROL VARIABILITY

Up to this point, the article has focused on a setting where there is a system scheduler that has access to job deadlines (or equivalently, to their corresponding laxities). This is perhaps too optimistic: In many situations, for example, our motivating applications of right-sizing in cloud computing and participating in ancillary service energy markets, jobs may not communicate their deadlines/laxities to the system scheduler.

Our analysis has shown that obtaining this information is crucial to effective scheduling. However, gathering this information from jobs requires paying close attention to the incentives they may have. For example, under LLF or exact scheduling and without proper incentives, it is in the best interest of a job to convey that it has zero laxity to be scheduled in priority or at a higher rate. Thus, the question of how to introduce a pricing mechanism that can extract deadline information from jobs is both crucial and challenging.

In this section, we tackle this question by assuming that a monetary reward is offered to every customer for each time unit of laxity offered to our system. One should think of this reward as a discount from the main charge of providing service. For instance, in a scenario of an electrical vehicle charging participating in an ancillary service market, the customer would pay the full charge for service at the fastest speed (full power), but the fare would be reduced if the customer is willing to allocate some spare time, that is, expose its laxity. Concretely, our analysis assumes the customer k receives the linear reward $r\ell_k$ for offering a laxity ℓ_k to the system.

In what follows, we first provide a model for customers responding strategically to these rewards, based on a private utility they assign to timeliness. Subsequently, we study the optimization by the service system of the offered reward, incorporating a cost for capacity variance, assuming the aggregate customer response is known. Finally, we provide a learning algorithm for the system operator to optimize its global cost while simultaneously estimating the customer response.

5.1. Modeling Customer Responses to Rewards for Deferred Service

We assume that customers assign a utility to their received service that is decreasing and concave with the amount of allocated laxity. Specifically, we work with the form

$$V_k(\ell) = c_k \sigma_k f\left(\frac{\ell}{\bar{\ell}_k}\right). \tag{32}$$

Here, c_k is a proportionality constant that translates utility to units of money and possibly depends on the customer type; σ_k is the requested service, and the utility is assumed to scale linearly with this quantity. The decreasing, concave function $f : \mathbb{R}_+ \to \mathbb{R}$, with $f(0) = 1$, captures the loss of valuation as service time is delayed; the scale parameter $\bar{\ell}_k$ characterizes the customer's value for timeliness. For concreteness, we look at the following instances of the function $f(\cdot)$:

Example 5.1. Let $f(x) = 1 - x^2$, $x \geq 0$. In this case, the utility will remain positive provided $\ell \leq \bar{\ell}_k$.

Example 5.2. Let $f(x) = 1 + \log(1 - x)$, $x \in [0, 1)$. In this case, $\bar{\ell}_k$ represents a hard barrier, the maximum laxity the customer is willing to accept.

When facing a reward $r\ell$, the customer's decision is the convex optimization

$$\max_{\ell} V_k(\ell) + r\ell = \max_{\ell} c_k \sigma_k f\left(\frac{\ell}{\bar{\ell}_k}\right) + r\ell. \tag{33}$$

We denote the maximizing solution by ℓ_k and further assume, for now, that it occurs at an interior point. This assumption means that the optimality condition is

$$\frac{c_k\sigma_k}{\bar{\ell}_k} f'\left(\frac{\ell_k}{\bar{\ell}_k}\right) = -r. \tag{34}$$

From here, we can solve for ℓ_k:

$$\ell_k = \bar{\ell}_k [f']^{-1}\left(-\frac{r\bar{\ell}_k}{c_k\sigma_k}\right).$$

It is more convenient to express the above response in terms of the *relative* offered laxity $\delta_k = \frac{\ell_k}{\sigma_k}$, and also $\bar{\delta}_k = \frac{\bar{\ell}_k}{\sigma_k}$, which represents the "nominal" value for the customer's relative laxity, an individual characteristic that represents the customer's "patience."

With this notation, the preceding expression has the form

$$\delta_k = \bar{\delta}_k [f']^{-1}\left(-\frac{r\bar{\delta}_k}{c_k}\right) =: \varphi\left(\bar{\delta}_k, \frac{r}{c_k}\right). \tag{35}$$

Note that φ represents the offered relative laxity in terms of customer parameters, and the reward, being a increasing function of the latter. We illustrate this offer curve by reconsidering the preceding examples.

Example 5.3. If $f(x) = 1 - x^2$, then Equation (34) becomes

$$\frac{c_k\sigma_k}{\bar{\ell}_k} \cdot 2\frac{\ell_k}{\bar{\ell}_k} = r, \quad \Longrightarrow \quad \ell_k = \frac{r}{2c_k}\frac{\bar{\ell}_k^2}{\sigma_k},$$

or in terms of relative laxities, the expression

$$\delta_k = \frac{r}{2c_k}\bar{\delta}_k^{\,2}.$$

Example 5.4. If $f(x) = 1 + \log(1 - x)$, then Equation (34) leads to

$$\frac{\ell_k}{\bar{\ell}_k} = 1 - \frac{c_k\sigma_k}{r\bar{\ell}_k},$$

provided the right-hand side is positive; otherwise, the optimum is $\ell_k = 0$. The expression

$$\delta_k = \left[\bar{\delta}_k - \frac{c_k}{r}\right]^+,$$

where $[\cdot]^+ = \max\{\cdot, 0\}$, summarizes the resulting offer curve.

We note that with the customer response captured generically by Equation (35), and in particular in the example instances, the mapping between $\bar{\delta}_k$ and δ_k does not involve the amount σ_k of work requested by the customer. Assume now that customers are drawn randomly from a population, with *independent* choice of the parameters $(\sigma_k, \bar{\delta}_k)$ (and, if applicable, the type c_k). This is arguably a natural assumption: when measured in *relative* terms to service time, the willingness to wait is a separate choice. In contrast, the independence of σ_k and $\bar{\ell}_k$ is less natural: if $\sigma_1 \gg \sigma_2$, the first customer should more readily accommodate a small extra service time as compared to the second.

Under the above assumption, we have the following attractive property: the independence will be *preserved* by the consumer response Equation (35), that is, the pairs (σ_k, δ_k) offered to the system will remain independent random variables. This allows us to invoke the theory of Section 4.2.2 regardless of the model for consumer choice.

5.2. Optimizing the Offered Rewards

We now consider the point of view of the service system operator, who has the objective of smoothing out the service capacity profile, and for this purpose offers rewards for service deferral. The offered reward price must be found by balancing the cost of these outlays themselves, against the penalty for service variability.

We assume here that Exact Scheduling is used, which implies that every job exits from the system at exactly its deadline. By Equation (30), the variance of service capacity takes the form

$$E[(\delta p)^2] = p_0^2 \frac{\lambda}{\mu} E\left[\frac{1}{1+\delta}\right],$$

where the expectation depends only on the distribution for the relative laxities offered by the customers. We assume a penalty per unit time proportional to this quantity.

On the other hand, the mean rewards paid per unit time will be equal to the mean reward per customer, times the rate of customers/second, that is,

$$\lambda r E[l] = \lambda r E[\sigma\,\delta] = \frac{\lambda}{\mu} r E[\delta],$$

assuming again independence between δ and σ. The overall cost to the system operator as a function of the price r is, therefore,

$$C(r) = \frac{\lambda}{\mu}\left\{\kappa E\left[\frac{1}{1+\delta}\right] + rE[\delta]\right\}; \tag{36}$$

here, κ is a suitable weighting constant. Note that the above cost depends on r not only in its explicit appearance, but also implicitly through the offered relative laxities δ, which as discussed before are increasing in r. In this way, the first component of the cost decreases in r, and the second increases; its correct tradeoff is the optimization problem in the hands of the system operator.

In general, the operator may not know how this customer responds to its rewards; in the next section, we describe a learning approach for online optimization of the above tradeoff. To gain more insight, we first assume the customer response is known and equal to the one in Example 5.3, namely

$$\delta = \frac{r}{2c}\bar{\delta}^2.$$

For simplicity, we set $c = 1$, which amounts to a choice of units of money. The system optimization is, therefore,

$$\min_{r\geq 0}\left\{\kappa E\left[\frac{1}{1+\frac{r}{2}\bar{\delta}^2}\right] + \frac{r^2}{2}E[\bar{\delta}^2]\right\}. \tag{37}$$

Recall that $\bar{\delta}$ is the parameter that characterizes the consumers' utility functions; as such, it will obey a certain exogenous probability distribution.

We now show that this problem has a well-defined positive optimum. Indeed, differentiation of the cost in Equation (37) with respect to r yields the optimality condition

$$\kappa E\left[\frac{\bar{\delta}^2/2}{(1+\frac{r}{2}\bar{\delta}^2)^2}\right] = rE[\bar{\delta}^2]; \tag{38}$$

the left-hand side is monotone decreasing in r, from $\frac{\kappa}{2}E[\bar{\delta}^2]$ to zero, so it intersects the straight line on the right at a unique point $r^* > 0$. For concreteness, we work out a specific case in the following example.

Example 5.5. Suppose that $\bar{\delta}$ is uniformly distributed in $[0, 1]$ (corresponding to nominal laxity $\bar{\ell}$ of at most 100% of service time). Then, we have

$$E\left[\frac{1}{1+\frac{r}{2}\bar{\delta}^2}\right] = \int_0^1 \frac{dx}{1+\frac{r}{2}x^2} = \frac{\arctan(\sqrt{r/2})}{\sqrt{r/2}}.$$

Also, $E[\bar{\delta}^2] = \frac{1}{3}$, so the optimal reward is the solution of

$$\min_{r>0} \kappa \frac{\arctan(\sqrt{r/2})}{\sqrt{r/2}} + \frac{r^2}{6}.$$

As an example, consider a value of $\kappa = 10^3$, giving more weight to variability costs, returns the optimal value $r^* \approx 21.4$, and a 55% savings with respect to the variability cost when offering no rewards.

5.3. Online Optimization of Rewards

The preceding analysis shows that a suitable tradeoff between variability cost and rewards given to clients can be found by appropriately choosing the reward value r^* as the minimizer of the cost function Equation (36). However, this assumes knowledge of the customer response by the service manager. This final section provides an algorithm for learning the optimal reward value online in a model-free manner. The algorithm uses measurements of the clients' responses to the current reward levels, to drive the system to the optimal value.

Specifically, we consider the following system behavior. Job requests arrive as a Poisson process of intensity λ, and when a job arrives at time t_k, it is offered a reward $r(t_k)$. At this point, the user informs the system its relative laxity δ_k, based on its own optimization for the current reward level, as in Equation (33). The system allocates service to the client using exact scheduling to meet the job deadline, that is, sets $u_k = 1/(1+\delta_k)$, and can now use this measured response to adjust the reward level for future users.

This setup can be represented as an online convex optimization (OCO) problem; see Flaxman et al. [2005] for background. The main challenge beyond classical OCO is that, while the cost function is easy to measure, its gradient is harder to estimate directly, since it depends on the sensitivity of clients to the rewards. Thus, the problem is a "bandit" version of online convex optimization.

There are standard approaches for learning in bandit OCO settings but, as we show below, one can improve upon them by taking advantage of the structure of the problem here. We consider three algorithms in the following: Bandit Gradient Descent from Flaxman et al. [2005], Kiefer-Wolfowitz (KW) stochastic gradient descent from Kiefer and Wolfowitz [1952], and a new algorithm we term Smoothed KW.

5.3.1. Algorithms. To begin, we consider a standard algorithm for bandit OCO problems: **Bandit Gradient Descent (BGD)** from Flaxman et al. [2005]. The algorithm works as follows: at any given arrival time t_k the algorithm has a current estimate of the reward r_k. Instead of using this reward, the system operator chooses a random search direction $\xi_k = \pm 1$ uniformly and offers the reward $\tilde{r}_k = r_k + \xi_k dr$ accordingly, where dr is a perturbation value. The system then obtains a response δ_k and evaluates the current cost as a one-sample estimate of the real cost in Equation (36):

$$C_k := \frac{\lambda}{\mu}\left[\kappa\frac{1}{1+\delta_k} + r_k\delta_k\right]. \tag{39}$$

Here, we assumed the average workload λ/μ as known, since it can be easily estimated from historical measurements, and focus only on the laxity response.

Using this measurement, the system updates its current reward estimate as

$$r_{k+1} = (r_k - \nu C_k \xi_k)^+,$$

where ν is a step size and the projection is to ensure that rewards remain positive. The successive perturbations enable the algorithm to estimate the gradient on average over successive steps, and with appropriate choices of the perturbation dr and step size ν, it can be shown it converges to the optimal value.

The second algorithm we study is an adaptation of the well-known **Kiefer-Wolfowitz (KW) stochastic gradient descent** proposed by Kiefer and Wolfowitz [1952]. At any given arrival time t_k, the algorithm has a current estimate of the reward r_k, but offers the arriving job a reward $r_k^{hi} = r_k + dr$, with dr again a perturbation value. The next job arriving at time T_{k+1} is offered a reward $r_k^{lo} = r_k - dr$. By using these two rewards, the system operator is able to obtain two offered laxities, δ_k^{hi} and $\delta_{k+1} = \delta_k^{lo}$, which can then be used to obtain one sample estimates of the cost, C_k^{hi} and C_k^{lo}. Combining these estimates, we can update the offered reward as

$$r_{k+2} = \left(r_k - \nu\frac{C_k^{hi} - C_k^{lo}}{r_k^{hi} - r_k^{lo}}\right)^+,$$

where again, ν is a step size. The above procedure provides a simple gradient estimate every two arrivals that on average lead to a suitable gradient descent optimization.

The main drawback of the above algorithms is that they rely on one-sample estimates and long-term averaging to reach the optimum, which can induce rapid variations on the offered rewards. Such variabilities are not well suited to a pricing and reward scheme where short-term unfairness and variability can be introduced between successive jobs.

Our third approach is based on the KW iterative procedure, but instead extending the averaging period to provide a better estimate and search direction; hence, we refer to it as **Smoothed KW**. It works as follows. Fix a time ΔT over which rewards are kept constant. Half of the arriving users are offered a high reward $r_k^{hi} = r_k + dr$ and the other half a low reward $r_k^{lo} = r_k - dr$. Successive responses from jobs are stored and averaged over all arrivals to obtain $\bar{C}_k^{hi}$ and $\bar{C}_k^{lo}$, replacing the expectation by empirical averages in Equation (36). Using these averages, we perform a gradient step $r_{k+1} = (r_k - \nu\frac{\bar{C}_k^{hi} - \bar{C}_k^{lo}}{r_k^{hi} - r_k^{lo}})^+$, as before. When ΔT is small, averaging is removed and it resembles the KW algorithm.

5.3.2. Numerical Experiments. To highlight the improvements that come from Smoothed KW, we compare the three approaches by using the framework of Example 5.5. To do this, we use a discrete-event simulation environment where jobs, arriving as a Poisson

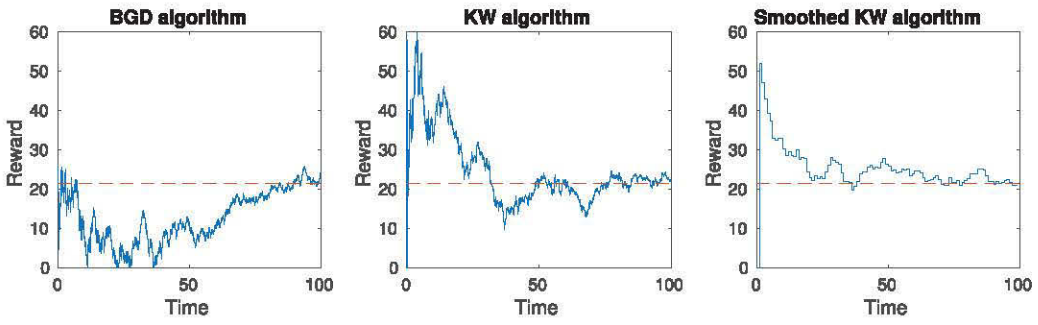


Fig. 10. Reward evolution vs. time for the online optimization algorithms analyzed.

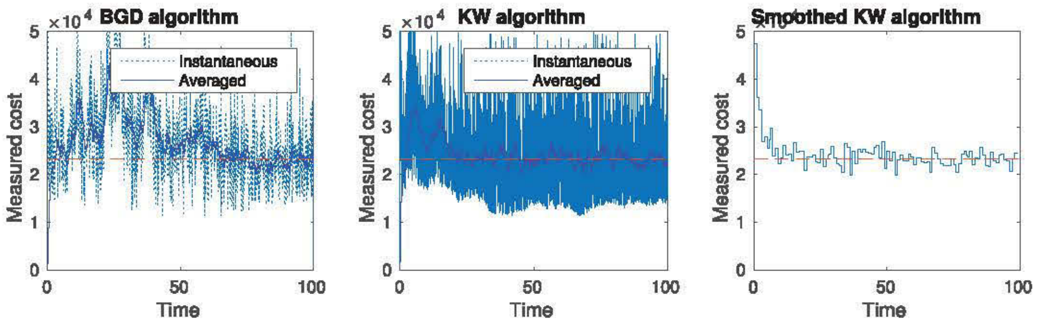


Fig. 11. Cost evolution vs. time for the online optimization algorithms analyzed.

process, are offered rewards $r(t_k)$, which are then adjusted following the preceding algorithms. For this example, we chose $\lambda = 50$, $\mu = 1$, and $\kappa = 10^3$ as before, leading to an average occupation of $\bar{n} = 50$ and variance $Var(\sum_i u_i) = Var(n) = 50$, when $r = 0$ and an average cost $C(0) = \kappa\lambda/\mu = 5 \times 10^4$.

We start the system in the state corresponding to the steady state when no rewards are offered and then let the algorithms find the optimal reward, which in this case is $r^* \approx 21.4$, with an average occupation of $n^* \approx 233$ due to the increased laxity. The optimal cost is $C^* \approx 2.3 \times 10^4$ and the optimal variance is $Var(\sum_i u_i) \approx 19.5$.

In Figure 10, we plot the reward evolution for the three algorithms, using the same step sizes. The figures highlight that all three online algorithms approximately find the correct reward level, with Smoothed KW offering a more predictable evolution of rewards, as expected.

This result is more evident in Figure 11, where the cost measurements are compared against each other. The cost estimation of the Smoothed KW makes a better prediction of the current cost, and drives the system toward the optimum, while BGD and KW have very noisy estimations of the current cost, albeit when averaged they show convergence to the optimal value.

Since the system operator objective was to reduce variability in the service rate, it is worth showing how this quantity evolves over time when using Smoothed KW. In Figure 12, we compare the active service rate, when there are no rewards, with the one obtained using Smoothed KW. After an initial learning phase, we can see that variability is reduced when using rewards. This is achieved at the expense of having a larger system occupation by extending the jobs' service time, purchasing laxity from users. For this example, the variance in steady state is $E[(\delta p)^2] \approx 20$, down from an original value of $\lambda/\mu = 50$ when no rewards are present.

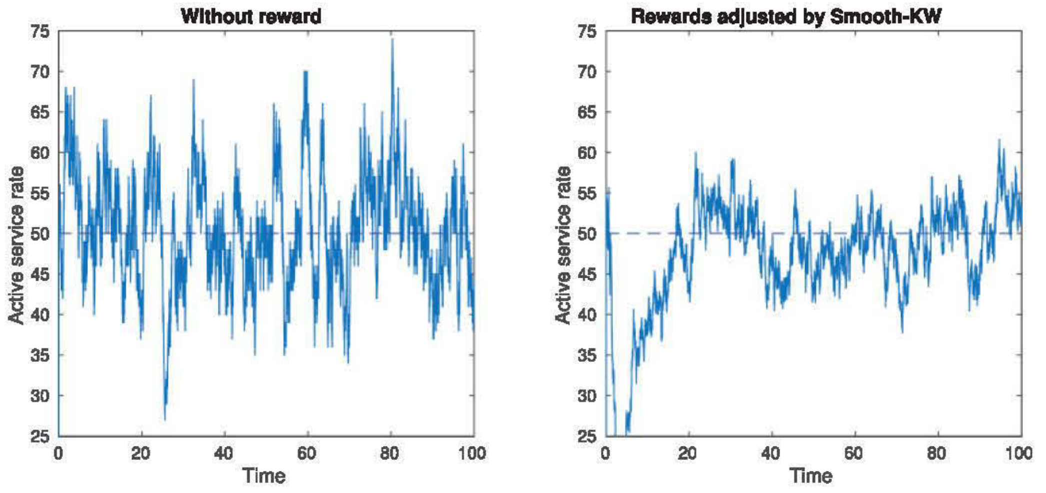


Fig. 12. Service rate variability reduction by using rewards: service rate evolution for a system with $\bar{p} = 50$.

6. CONCLUDING REMARKS

In this article, we have studied service systems that can adjust the capacity offered to each job, with the objective of reducing the variability of the aggregate service capacity. We are motivated by applications, such as cloud computing or smart energy grids, for which there is a cost penalty associated from deviations in the required service rate from the provisioned amount.

Our results highlight the tradeoffs between the variability of service capacity and the timeliness of the service offered to jobs, in particular meeting their deadlines. We studied four policies in this regard, computing the capacity variance through a stochastic queueing analysis, as a function of load deferrability parameters. In two of these policies deadline misses are allowed, and their probability must be quantified; the other two strictly enforce deadlines. The alternatives were compared in their performance and implementation simplicity, in particular, the ability to operate in a decentralized manner.

Since deferral decisions must be based on information about the job deadlines, an important question for the system operator is how to ensure their truthful declaration. We explored the possibility of offering rewards to jobs for their spare time and posed an economic tradeoff between the cost penalty for variability and the reward payments. We investigated decentralized ways to optimize this tradeoff through stochastic gradient algorithms that effectively probe the job population's willingness to defer service.

The goal of this article is to initiate analysis of this new model and the tradeoffs therein, and thus there are many avenues for future study building on the results in this article. For example, our study of service deferral has covered four representative policies, with an emphasis on those with simple implementation. These policies are far from exhaustive and a natural question task for future research is to develop policies that outperform the four studied here. Similarly, our study of incentives focused on Exact Scheduling and it would be very interesting to understand the interaction of other policies with offered rewards and to optimize the tradeoff between performance and the cost of providing incentives.

REFERENCES

Muhammad A. Adnan and Rajesh K. Gupta. 2014. Workload shaping to mitigate variability in renewable power use by data centers. In *Proceedings of the IEEE 7th International Conference on Cloud Computing*. 96–103.

Muhammad A. Adnan, Ryo Sugihara, and Rajesh K. Gupta. 2012. Energy efficient geographical load balancing via dynamic deferral of workload. In *Proceedings of the IEEE 5th International Conference on Cloud Computing (CLOUD'12)*. 188–195.

Philipp Afèche and Haim Mendelson. 2004. Pricing and priority auctions in queueing systems with a generalized delay cost structure. *Manage. Sci.* 50, 7 (2004), 869–882.

Phlippe Afèche and J. Michael Pavlin. 2016. Optimal price/lead-time menus for queues with customer choice: Segmentation, pooling, and strategic delay. *Management Science* 62, 8 (2016), 2412–2436.

Mustafa Akan, Baris Ata, and Tava Olsen. 2012. Congestion-based lead-time quotation for heterogenous customers with convex-concave delay costs: Optimality of a cost-balancing policy based on convex hull functions. *Operat. Res.* 60, 6 (2012), 1505–1519.

François Baccelli and Bartlomiej Błaszczyszyn. 2009. *Stochastic Geometry and Wireless Networks, Volume I—Theory*. Now Publishers.

Partha P. Bhattacharya and Anthony Ephremides. 1989. Optimal scheduling with strict deadlines. *IEEE Trans. Automat. Control* 34, 7 (1989), 721–728.

Federico Bliman, Andres Ferragut, and Fernando Paganini. 2015. Controlling aggregates of deferrable loads for power system regulation. In *Proceedings of the 2015 American Control Conference, Chicago, IL.*

Sem Borst, Onno Boxma, and Predrag Jelenkovic. 2000. Asymptotic behavior of generalized processor sharing with long-tailed traffic sources. In *Proceedings of the IEEE/Infocom 2000*, Vol. 2. IEEE, 912–921.

Onno Boxma and Bert Zwart. 2007. Tails in scheduling. *ACM SIGMETRICS Perform. Eval. Rev.* 34, 4 (2007), 13–20.

Sabri Çelik and Constantinos Maglaras. 2008. Dynamic pricing and lead-time quotation for a multiclass make-to-order queue. *Manage. Sci.* 54, 6 (2008), 1132–1146.

Niangjun Chen, Lingwen Gan, Steven H. Low, and Adam Wierman. 2014. Distributional analysis for model predictive deferrable load control. In *Proceedings of the IEEE 53rd Annual Conference on Decision and Control*.

Jeffrey Dean and Luiz André Barroso. 2013. The tail at scale. *Commun. ACM* 56, 2 (2013), 74–80.

Andres Ferragut and Fernando Paganini. 2015. Queueing analysis of service deferrals for load management in power systems. In *Proceedings of the 53rd Annual Allerton Conference on Communication, Control and Computing*.

Abraham D. Flaxman, Adam Tauman Kalai, and H. Brendan McMahan. 2005. Online convex optimization in the bandit setting: Gradient descent without a gradient. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*. 385–394.

Hans C. Gromoll and Łukasz Kruk. 2007. Heavy traffic limit for a processor sharing queue with soft deadlines. *Ann. Appl. Probabil.* 17, 3 (2007), 1049–1101.

Mor Harchol-Balter. 2013. *Performance Modeling and Design of Computer Systems: Queueing Theory in Action*. Cambridge University Press.

Refael Hassin. 2016. *Rational Queueing*. CRC Press.

Refael Hassin and Moshe Haviv. 2003. *To Queue or not to Queue: Equilibrium Behavior in Queueing Systems*. Vol. 59. Springer Science & Business Media.

J. Hong, X. Tan, and D. Towsley. 1989. A performance analysis of minimum laxity and earliest deadline scheduling in a real-time system. *IEEE Trans. Comput.* 38 (1989), 1736–1744.

Jack Kiefer and Jacob Wolfowitz. 1952. Stochastic estimation of the maximum of a regression function. *The Ann. Math. Stat.* 23, 3 (1952), 462–466.

Leonard Kleinrock. 1975. *Queueing Systems, Volume I: Theory*. Wiley Interscience.

John P. Lehoczky. 1997. Real-time queueing network theory. In *Proceedings of the 18th IEEE Real-Time Systems Symposium*. 58–67.

Minghong Lin, Adam Wierman, Lachlan L. H. Andrew, and Eno Thereska. 2013. Dynamic right-sizing for power-proportional data centers. *IEEE/ACM Trans. Network.* 21, 5 (2013), 1378–1391.

Zhenhua Liu, Minghong Lin, Adam Wierman, Steven H. Low, and Lachlan L. H. Andrew. 2011. Geographical load balancing with renewables. *ACM SIGMETRICS Perform. Eval. Rev.* 39, 3 (2011), 62–66.

Constantinos Maglaras and Jan A. Van Mieghem. 2005. Queueing systems with leadtime constraints: A fluid-model approach for admission and sequencing control. *Eur. J. Operat. Res.* 167, 1 (2005), 179–207.

Michel Mandjes and Bert Zwart. 2006. Large deviations of sojourn times in processor sharing queues. *Queueing Syst.* 52, 4 (2006), 237–250.

Haim Mendelson and Seungjin Whang. 1990. Optimal incentive-compatible priority pricing for the M/M/1 queue. *Operat. Res.* 38, 5 (1990), 870–883.

A. Nayyar, J. Taylor, A. Subramanian, K. Poolla, and P. Varaiya. 2013. Aggregate flexibility of a collection of loads. In *Proceedings of the 52nd IEEE Conference on Decision and Control*.

Misja Nuyens, Adam Wierman, and Bert Zwart. 2008. Preventing large sojourn times using SMART scheduling. *Operat. Res.* 56, 1 (2008), 88–101.

Michael Pinedo. 1983. Stochastic scheduling with release dates and due dates. *Operat. Res.* 31, 3 (1983), 559–572.

Erica Plambeck, Sunil Kumar, and Michael J. Harrison. 2001. A multiclass queue in heavy traffic with throughput time constraints: Asymptotically optimal dynamic controls. *Queueing Syst.* 39, 1 (2001), 23–54.

Casey Quinn, Daniel Zimmerle, and Thomas H. Bradley. 2010. The effect of communication architecture on the availability, reliability, and economics of plug-in hybrid electric vehicle-to-grid ancillary services. *J. Power Sources* 195, 5 (2010), 1500–1509.

Philippe Robert. 2003. *Stochastic Networks and Queues*. Springer.

Eric Sortomme and Mohamed A. El-Sharkawi. 2012. Optimal scheduling of vehicle-to-grid energy and ancillary services. *IEEE Trans. Smart Grid* 3, 1 (2012), 351–359.

A. Subramanian, M. J. Garcia, D. S. Callaway, K. Poolla, and P. Varaiya. 2013. Real-time scheduling of distributed resources. *IEEE Trans. Smart Grid* 4 (2013), 2122–2130.

Jasna Tomić and Willett Kempton. 2007. Using fleets of electric-drive vehicles for grid support. *J. Power Sources* 168, 2 (2007), 459–468.

Luis M. Vaquero, Luis Rodero-Merino, and Rajkumar Buyya. 2011. Dynamically scaling applications in the cloud. *ACM SIGCOMM Comput. Commun. Rev.* 41, 1 (2011), 45–52.

Adam Wierman and Bert Zwart. 2012. Is tail-optimal scheduling possible? *Operat. Res.* 60, 5 (2012), 1249–1257.

Stan Zachary. 2007. A note on insensitivity in stochastic networks. *J. Appl. Probabil.* 44 (2007), 238–248.

Qian Zhu and Gagan Agrawal. 2010. Resource provisioning with budget constraints for adaptive applications in cloud environments. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*. ACM, 304–307.

Received November 2016; accepted April 2017