SOAP Bubbles: Robust Scheduling Under Adversarial Noise

Ziv Scully
Carnegie Mellon University
zscully@cs.cmu.edu

Mor Harchol-Balter
Carnegie Mellon University
harchol@cs.cmu.edu

***Abstract*— A great many scheduling policies for the M/G/1 queue are so-called *SOAP policies* [1], meaning they assign each job a priority based on its *age*, the amount of service it has received so far. Perhaps the most notable example is the *Gittins policy*, which minimizes mean response time when job sizes are unknown. However, in some computer systems even job ages, let alone job sizes, are not precisely known by the scheduler. This can occur when scheduling in a time-shared system or over a network. Given that the Gittins policy relies on knowing exact job ages, it is not clear how to minimize mean response time in such settings.**

In this paper we study scheduling for the M/G/1 when the scheduler knows only approximate job ages. We find that naively using the traditional Gittins policy is not *robust*, meaning that introducing even an infinitesimal amount of noise in job ages can cause a large jump in mean response time. By examining the ways in which this naive policy fails, we construct a simple variation of the Gittins policy, called the *shift-flat Gittins policy*, which is indeed robust to noise and therefore has near-optimal mean response time. Moreover, we show that our shift-flat construction generalizes, yielding a robust variation of any SOAP policy.

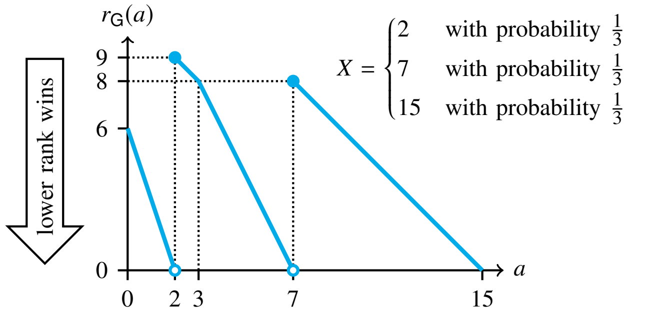
I. Introduction

Scheduling jobs in the M/G/1 queue to minimize mean response time is a classic problem. In a preemptive setting where job sizes are unknown to the scheduler, the optimal scheduling algorithm is the *Gittins policy* [2; 3; 4]. Based on the job size distribution, the Gittins policy assigns each job a number, called the job's *Gittins index*, and serves the job of maximum Gittins index. A job's Gittins index depends only on its *age*, the amount time the job has been served so far. In particular, a job's Gittins index is independent of other jobs.

Although the Gittins policy has been known to be optimal for some time, its mean response time was analyzed only recently, when Scully et al. [1] analyzed all policies in the broad class of *SOAP policies*. A SOAP policy is any policy in which a job's priority, or *rank*, is determined by its age according to a *rank function*. Specifically, a SOAP policy always serves the job of *minimum rank*. Fig. 1 shows the rank function of an example of the Gittins policy.

To implement the Gittins policy or any other SOAP policy, the scheduler needs to know each job's exact age at every moment in time. However, in some computer systems the scheduler only knows each job's *approximate age*. For example, in a time-shared system, it can be hard to precisely track the CPU time allocated to a particular job without incurring significant overhead. As another example, in a networked system, a controller may not be on the same machine as the jobs it is scheduling, in which case updating age information over the network incurs a delay or cost.

This research was supported by NSF-XPS-1629444, NSF-CSR-180341, and a 2018 Faculty Award from Microsoft. Ziv Scully was supported by an ARCS Foundation scholarship and a NSF Graduate Research Fellowship under Grant No. DGE-1745016.



The rank function of the Gittins policy when each job is equally likely to be size 2, 7, or 15. For each age a, the rank function specifies the rank $r_{\mathsf{G}}(a)$ assigned to jobs of age a. The scheduler always serves the job of minimum rank, so we set $r_{\mathsf{G}}(a)$ to be the reciprocal of the job's Gittins index at age a.

Fig. 1. Rank Function of the Gittins Policy

Motivated by such examples, we create a model of M/G/1 scheduling with noisy age information. When a job's exact age is a, the scheduler sees a perturbed age $b \in [a - \Delta, a + \Delta]$, where $\Delta \geq 0$ is the *system noise*. To conservatively model potential correlations between the age errors of different jobs, we assume that b is chosen *adversarially* from $[a - \Delta, a + \Delta]$.

The Gittins policy requires jobs' exact ages, so we ask:

> What scheduling policy has *optimal or near-optimal mean response time* when the job ages reported to the scheduler are adversarially perturbed?

As a first attempt, we might try the *naive Gittins policy*, which simply plugs perturbed ages into the Gittins policy's rank function (Fig. 1) and serves the job of minimum rank. The hope is that introducing a small amount of noise increases mean response time by only a small amount. Unfortunately, we find that the naive Gittins policy performs poorly: introducing *any* noise, even infinitesimal, causes a large upward jump in mean response time. Specifically, we prove $\lim_{\Delta\to 0} \mathbf{E}[T(\Delta)] \neq \mathbf{E}[T(0)]$, where is $\mathbf{E}[T(\Delta)]$ is the mean response time under system noise Δ.

We call a scheduling policy *robust* if $\lim_{\Delta\to 0} \mathbf{E}[T(\Delta)] = \mathbf{E}[T(0)]$. We have seen already that the naive Gittins policy is not robust, so we ask:

> How do we construct a *robust variation* of the Gittins policy?

In this paper we present the *first M/G/1 scheduling policies that are robust to adversarial noise*. We make the following contributions:

- We give a *definition of robustness* for scheduling policies under adversarial age noise (Section III).
- Through a case study, we construct a variation of the Gittins policy called the *shift-flat Gittins policy*, which we prove is robust (Section IV). Remarkably, the shift-flat construction consists of just two simple steps:
  - *Shift:* subtract Δ from each job's perturbed age before applying the rank function (Section IV-A).
  - *Flatten:* turn each local maximum of the rank function into a plateau of length at least 2Δ (Section IV-B).
- As part of evaluating robustness, we develop a new technique for computing an *upper bound on the mean response time* of any SOAP policy in an M/G/1 system with adversarial age noise (Section V).
- We show that the shift-flat construction generalizes beyond the Gittins policy to give a *robust variation of any SOAP policy* (Section VI).

II. System Model

We consider an M/G/1 queue with Poisson job arrivals at rate λ and job sizes drawn i.i.d. from distribution X. The *load* of the system is $\rho = \lambda\mathbf{E}[X]$, and we assume $\rho \in (0, 1)$. We use a preempt-resume model, meaning preemption and processor sharing are permitted without penalty or loss of work. At any moment in time, a job's *exact age*, or simply *age*, is the amount of service it has received so far. We are interested in minimizing *mean response time* $\mathbf{E}[T]$, the expected amount of time a job spends in the system from arrival to completion.

Throughout this paper, we discuss *SOAP scheduling policies* [1]. A SOAP policy[1] is specified by a *rank function*

$$r : \mathbb{R}_{\geq 0} \to \mathbb{R}$$

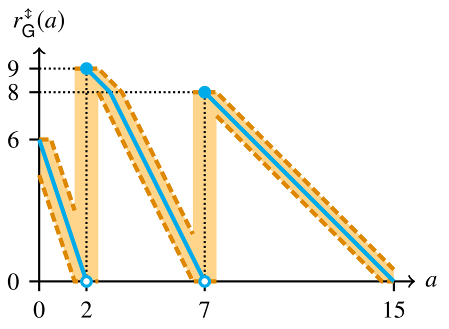
which maps a job's exact age to its *rank*, or priority. At every moment in time, the scheduler serves the job of *minimum rank*, meaning lower rank corresponds to better priority. When multiple jobs have the same rank, the scheduler breaks the tie first-come, first-served.

A. Adversarial Age Noise

To implement a SOAP policy, the scheduler needs to know the exact age of each job at all times. We study a setting where the scheduler does *not* know exact ages. When a job's exact age is a, the scheduler sees only the job's *perturbed age*, an adversarially chosen $b \in [a - \Delta, a + \Delta]$, where $\Delta \geq 0$ is the *system noise*. We typically write a for a job's exact age and b for a job's perturbed age.

For the most part, we work with a very powerful adversary, allowing it complete knowledge of the current system state, including job sizes, but not of the future arrival sequence. While a job is being served, the adversary can at every moment in time choose a new perturbed age for the job. However, while a job is waiting in the queue, its perturbed age must remain constant.

[1]Described below is actually a subclass of SOAP policies. See Scully et al. [1, Section 2] for the fully general definition. The subclass considered herein is SOAP policies with a single descriptor and ranks $\mathbb{R}$. For ease of exposition, we prove results for only this subclass, but the proofs generalize to all SOAP policies.



The rank function (blue line), the lower and upper rank functions (lower and upper dashed orange lines, respectively), and the rank bubble function $r_{\mathsf{G}}^{\updownarrow}$ (orange region) of the naive Gittins policy when each job is equally likely to be size 2, 7, or 15. The system noise is $\Delta = 1/2$.

Fig. 2. Rank Bubble Function of the Naive Gittins Policy

Even though the scheduler knows only perturbed ages, we can still use a rank function r to determine a scheduling policy. At every moment in time, we serve the job of minimal rank $r(b)$, where b is the job's perturbed age. We call this a *perturbed SOAP policy* with rank function r. For example, the *naive Gittins policy* is the perturbed SOAP policy with rank function r_{G} (Fig. 1).

B. SOAP Bubble Policies

A perturbed SOAP policy is a special case of what we call a *SOAP bubble policy*. Instead of a single rank function, a SOAP bubble policy has a *rank bubble function* $r^{\updownarrow}$, which maps each exact age a to an interval of ranks

$$r^{\updownarrow}(a) = [r^{\downarrow}(a), r^{\uparrow}(a)],$$

where[2]

$$\begin{aligned} r^{\downarrow}(a) &= \inf\nolimits_{b \in [a-\Delta, a+\Delta]} r(b) \\ r^{\uparrow}(a) &= \sup\nolimits_{b \in [a-\Delta, a+\Delta]} r(b) \end{aligned} \tag{1}$$

are the *lower and upper rank functions*, respectively. For example, Fig. 2 shows the rank bubble function of the naive Gittins policy. While one could in principle define SOAP bubble policies using arbitrary rank bubble functions, our main focus is on perturbed SOAP policies, which always have lower and upper rank functions given by (1).

At any moment in time, when a job's exact age is a, its rank is adversarially chosen from $r^{\updownarrow}(a)$. A SOAP bubble policy always serves the job of *minimum rank*, but now the rank of a job depends not just on its age but also on the adversary's choice within $r^{\updownarrow}(a)$. The adversary can only change a job's rank while the job is being served.

Strictly speaking, the SOAP bubble formulation of perturbed SOAP policies makes the adversary even more

[2]For the specific rank functions discussed in this paper, the supremum in $r^{\uparrow}(a)$ is always attained. Handling the case where the supremum is not attained requires, roughly speaking, treating ranks where the supremum is not attained as "just below" ranks where the supremum is attained. The changes are similar to those explained by Scully et al. [1, Appendix C].

powerful: we allow the adversary to pick any rank from the interval $r^{\ddagger}(a)$, even though not all ranks in the interior of $r^{\ddagger}(a)$ are necessarily attained as $r(b)$ for some perturbed age $b \in [a - \Delta, a + \Delta]$. Fortunately, giving the adversary this additional power turns out not to weaken our results.

To review, a job's rank in a perturbed SOAP policy is determined as follows:

- Suppose a job J has exact age a.
- The *adversary* reports J's perturbed age $b \in [a-\Delta, a+\Delta]$.
- Based on b, the *scheduler* assigns J's rank $r(b) \in r^{\ddagger}(a)$.

III. Why Naive Policies Can Fail to be Robust

We begin with a case study of the mean response time of the naive Gittins policy. As a running example, in this section and the next we will use job size distribution

$$X = \begin{cases} 2 & \text{with probability } \frac{1}{3} \\ 7 & \text{with probability } \frac{1}{3} \\ 15 & \text{with probability } \frac{1}{3}. \end{cases}$$

The *naive Gittins policy* for job size distribution X is the perturbed SOAP policy, denoted G, with rank function r_{G} (Fig. 1). We can also view it as a SOAP bubble policy[3] with rank bubble function $r_{\mathsf{G}}^{\ddagger}$ (Fig. 2).

Let $\mathbf{E}[T(\Delta)]_{\mathsf{G}}$ be the mean response time of the naive Gittins policy under system noise Δ. We know the naive Gittins policy is optimal when $\Delta = 0$, so we might hope that $\mathbf{E}[T(\Delta)]_{\mathsf{G}}$ is not much greater than $\mathbf{E}[T(0)]_{\mathsf{G}}$ when Δ is small. We call this property *robustness*, formally defined below. Unfortunately, as we will soon show, the naive Gittins policy is not robust.

Definition 1. A perturbed SOAP policy P is *robust* for a given arrival rate λ and job size distribution X if its mean response time $\mathbf{E}[T(\Delta)]_P$ satisfies[4]

$$\lim_{\Delta \to 0} \mathbf{E}[T(\Delta)]_P = \mathbf{E}[T(0)]_P.$$

Determining whether or not a perturbed SOAP policy is robust requires analyzing its mean response time. Exactly computing mean response time is challenging because it requires analyzing a system with both stochastic and adversarial features. Fortunately, for our purposes it suffices to bound mean response time.

- To show that a policy is *not robust*, we need a *lower bound*. One way to find a lower bound is to fix a strategy for the adversary. If the strategy is simple enough, we obtain a system which is purely stochastic and can thus be analyzed as an ordinary SOAP policy [1].
- To show that a policy is *robust*, we need an *upper bound*. This is outside the scope of existing techniques. We give a new method for computing such upper bounds in Section V.

[3]Strictly speaking, saying a perturbed SOAP policy is a SOAP bubble policy slightly abuses terminology. A perturbed SOAP policy actually corresponds to many SOAP bubble policies, one for each choice of Δ, because $r^{\ddagger}$ in (1) depends on Δ.

[4]We will later generalize this definition by allowing P to depend on Δ (Section IV) and relaxing the condition by replacing = with $\leq$ (Section VI).

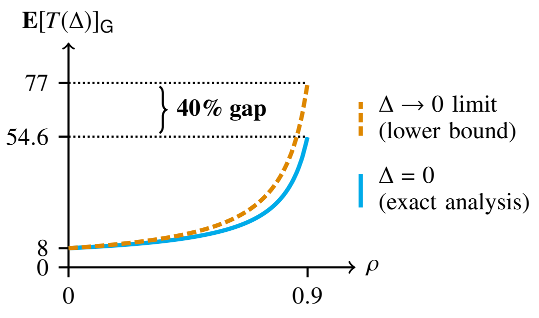


Fig. 3. Response Time of the Naive Gittins Policy

We aim to show that the naive Gittins policy G is not robust, so we must compute a lower bound $\ell(\Delta) \leq \mathbf{E}[T(\Delta)]_{\mathsf{G}}$ such that $\mathbf{E}[T(0)]_{\mathsf{G}} < \lim_{\Delta \to 0} \ell(\Delta)$. To compute the lower bound $\ell(\Delta)$, we have the adversary follow a simple fixed strategy: always give each job the *worst possible rank*. If the adversary follows this strategy, we exactly can analyze mean response time, as the scheduler effectively follows a SOAP policy with rank function $r_{\mathsf{G}}^{\uparrow}$ (Fig. 2, upper orange line). We omit the routine SOAP analysis, which follows the method given by Scully et al. [1, Section 6.4], and simply state the resulting mean response times. For $\Delta = 0$, we obtain

$$\mathbf{E}[T(0)]_{\mathsf{G}} = \ell(0) = \frac{\frac{139}{36}\rho}{(1-\rho)\left(1-\frac{1}{4}\rho\right)} + \frac{\frac{37}{6}}{1-\frac{1}{4}\rho} + \frac{11}{6},$$

and for $\Delta > 0$, we obtain

$$\begin{aligned} \lim_{\Delta \to 0} \mathbf{E}[T(\Delta)]_{\mathsf{G}} &\geq \lim_{\Delta \to 0} \ell(\Delta) \\ &= \frac{\frac{139}{24}\rho}{(1-\rho)\left(1-\frac{1}{4}\rho\right)} + \frac{6}{1-\frac{1}{4}\rho} + 2 \\ &= \mathbf{E}[T(0)]_{\mathsf{G}} + \frac{\frac{17}{9}\rho + \frac{1}{24}\rho^2}{(1-\rho)\left(1-\frac{1}{4}\rho\right)}. \end{aligned} \tag{2}$$

Fig. 3 shows the above mean response times.

From (2) we conclude that *the naive Gittins policy is not robust*. In fact, at load $\rho = 0.9$, introducing even infinitesimal noise increases mean response time by over 40%. This is a lower bound on the true increase because a different adversarial strategy may further increase mean response time.

Why does adding such a small amount of noise cause such a large jump in mean response time? It turns out that the noise has the largest effect on size 2 jobs, so we focus on those. Recall that the adversary's strategy means the scheduler effectively uses a SOAP policy with rank function $r_{\mathsf{G}}^{\uparrow}$. Examining Fig. 2, we see that $r_{\mathsf{G}}^{\uparrow}(a)$ jumps up to rank 9, the worst possible rank, at age $a = 2 - \Delta$.

- When $\Delta = 0$, a size 2 job completes before this jump.
- When $\Delta > 0$, a size 2 job has Δ work left when this jump happens. The job then has the worst possible rank, so it has to wait for *every job that arrived before it* to complete.

IV. Creating a Robust Policy

Our goal in this section is to create a policy P with near-optimal response time, at least for small system noise Δ.

Because the naive Gittins policy G is optimal when $\Delta = 0$, it suffices for P to be both

- *robust*, as defined in Definition 1, and
- *noiseless-optimal*, meaning $\mathbf{E}[T(0)]_P = \mathbf{E}[T(0)]_{\mathsf{G}}$.

To achieve both of these, *our policy must depend on the system noise* Δ.[5] This is because any noiseless-optimal policy must be essentially equivalent to the Gittins policy [3], so the adversary can report perturbed age 2 for jobs of exact age $2 - \Delta$, which prevents robustness.

We have discovered multiple methods for constructing a robust variation of our running example G. We focus our presentation on just one of them: the *shift-flat construction*, summarized in Fig. 4. Although shift-flat will be motivated by the specific running example, we show in Section VI that it generalizes to yield a robust version of any SOAP policy.

A. Shifting the Rank Function

We saw in Section III that one way the adversary can increase mean response time is to make jobs' ranks jump upward earlier than they are supposed to. How can we prevent this? One approach is to delay the rank jump at age 2 to age $2 + \Delta$. When the jump happens at age $2 + \Delta$, the earliest the adversary can make a job's rank jump is age 2, which means size 2 jobs do not reach the jump, as desired.

Motivated by the need to delay rank jumps, we define the *shift Gittins policy*, denoted SΔ, to be the perturbed SOAP policy with rank function[6]

$$r_{\mathsf{S}\Delta}(b) = r_{\mathsf{G}}(b - \Delta).$$

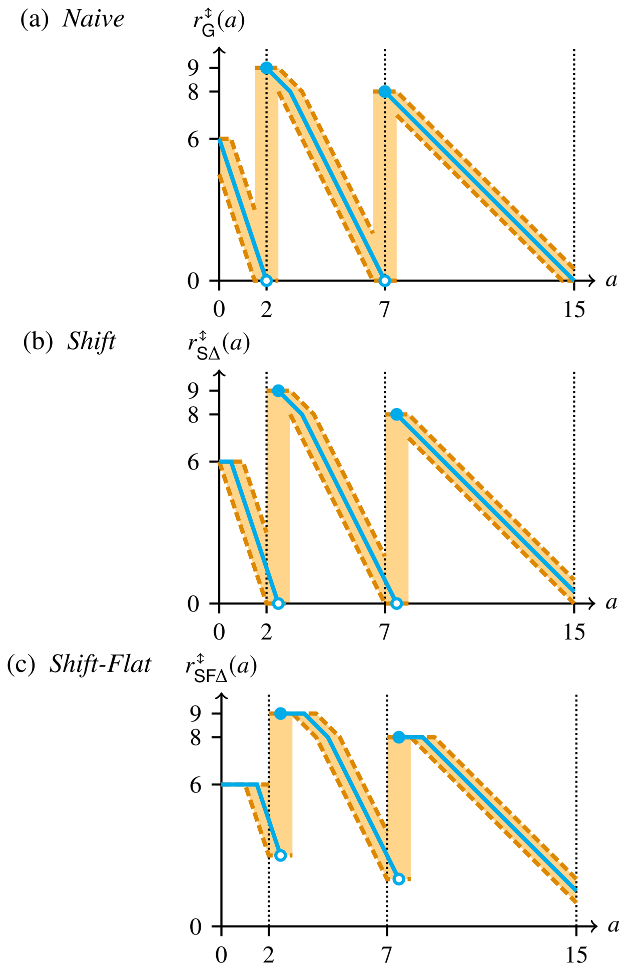
The shift Gittins policy is Δ-parametrized, which is why Δ appears in the notation SΔ. Fig. 4(b) shows the resulting rank bubble function $r^{\ddagger}_{\mathsf{S}\Delta}$.

The shift Gittins policy improves upon the naive Gittins policy. Specifically, the adversarial strategy of always assigning every job the worst possible rank hurts mean response time of the naive Gittins policy but has almost no effect on that of the shift Gittins policy. However, the shift Gittins policy is *still not robust*, because it is vulnerable to a different strategy for the adversary.

Recall from Section II-A that we allow the adversary access to job sizes. We consider the following strategy for the adversary: honestly report the exact ages of all jobs, except give rank 0 to size 15 jobs with exact age $a \in (2, 2 + 2\Delta)$. This forces us to give size 15 jobs priority over size 7 jobs near age 2. In the $\Delta \longrightarrow 0$ limit, the effect of this is that the system behaves as if $\Delta = 0$, except ties between jobs at age 2 are broken in favor of the larger size 15 jobs, which increases mean response time. Fig. 5, which shows the mean response time bound, confirms this. At load $\rho = 0.9$, introducing even infinitesimal noise increases mean response time over 40%.

The adversary knowing job sizes is perhaps too pessimistic in some applications. We conjecture that the shift Gittins policy is robust if the adversary does not know job sizes.

[5]We generalize Definition 1 by allowing P to vary with Δ.
[6]We let $r_{\mathsf{G}}(a - \Delta) = r_{\mathsf{G}}(0)$ for $a < \Delta$.



The rank function (blue line), the lower and upper rank functions (dashed orange lines), and the rank bubble function (orange region) of each of three policies: (a) the naive Gittins policy G, (b) the shift Gittins policy SΔ, (c) the shift-flat Gittins policy SFΔ. Each job is equally likely to be size 2, 7, or 15, and the system noise is $\Delta = 1/2$.

In (a), a job's rank can jump upward as early as age $2 - \Delta$, which greatly increases the mean response time of size 2 jobs. In (b), we solve this problem by shifting the rank function by Δ, but the gap between the lower and upper rank functions at local maxima leaves us vulnerable to a size-aware adversary, who can give size 15 jobs priority over size 7 jobs near age 2. In (c), we flatten the local maxima into plateaus of length 2Δ so that the lower and upper rank functions meet, removing this vulnerability.

Fig. 4. Constructing the Shift-Flat Gittins Policy

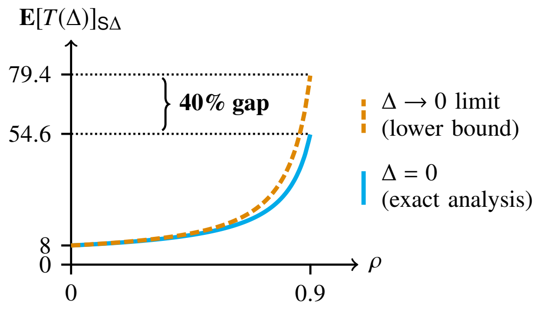


Fig. 5. Response Time of the Shift Gittins Policy

B. Flattening the Rank Function

We saw in Section IV-A that one way the adversary can increase response time is to discriminate based on job sizes, giving larger jobs priority over shorter ones. How can we prevent this? Under the shift Gittins policy $\mathsf{S}\Delta$, the adversary takes advantage of the gap between the lower and upper rank functions near local maxima. For instance, in Fig. 4(b), the maximum rank attained by $r^{\uparrow}_{\mathsf{S}\Delta}$ is 9 but the maximum rank attained by $r^{\downarrow}_{\mathsf{S}\Delta}$ is $9 - 2\Delta$. The adversary can use this gap to let larger jobs to bypass smaller ones. To prevent this, we need to *collapse the rank bubble* near each local maximum, meaning the lower and upper rank functions must meet.

How do we force the lower and upper rank functions to meet? Our idea is to *flatten the rank function* so that each local maximum is attained over an interval of width 2Δ. This forces the rank bubble to collapse at the center of that interval. Specifically, if the rank function of a perturbed SOAP policy P satisfies $r_P(b) = k$ for all $b \in [a - \Delta, a + \Delta]$, then $r^{\downarrow}_P(a) = r^{\uparrow}_P(a) = k$ by (1).

Motivated by the need to flatten the rank function near local maxima, we define the *shift-flat Gittins policy*, denoted $\mathsf{SF}\Delta$, to be the perturbed SOAP policy with rank function

$$r_{\mathsf{SF}\Delta}(b) = \sup_{a \in [b-2\Delta, b]} r_{\mathsf{S}\Delta}(a) = \sup_{a \in [b-3\Delta, b-\Delta]} r_{\mathsf{G}}(a). \quad (3)$$

That is, the shift-flat Gittins policy shifts the rank function by Δ then takes a rolling maximum over an age interval of width 2Δ.

As we will show formally in Sections V and VI, *the shift-flat Gittins policy is robust*. The argument goes roughly as follows. Assume the system noise Δ is small but nonzero. By (3), each local maximum of $r_{\mathsf{SF}\Delta}$ is attained over an interval of width 2Δ, ensuring the rank bubble collapses. Specifically, suppose r_{G} has a local maximum at age a. Then $r_{\mathsf{SF}\Delta}(b) = r_{\mathsf{G}}(a)$ for all $b \in [a + \Delta, a + 3\Delta]$, which implies

$$r^{\downarrow}_{\mathsf{SF}\Delta}(a + 2\Delta) = r^{\uparrow}_{\mathsf{SF}\Delta}(a + 2\Delta) = r_{\mathsf{G}}(a).$$

The shift-flat Gittins policy therefore behaves similarly to the Gittins policy *without system noise*, except upward rank jumps are delayed by at most 2Δ. This means, roughly speaking, from the perspective of any job J:

> While J is in the system, each other job is served for *at most* 2Δ *more time* under the shift-flat Gittins policy than under the Gittins policy without noise.

This increase is continuous in Δ, implying robustness.

Formalizing the above robustness argument takes two steps:

- Theorem 6 gives an upper bound on $\mathbf{E}[T(\Delta)]_{\mathsf{SF}\Delta}$, and
- Theorem 8 proves robustness from the upper bound.

Fig. 6 plots the upper bound we obtain from Theorem 6. We see that at load $\rho = 0.9$, introducing noise $\Delta = 1/2$ increases mean response time by under 11%. This is much better performance under much more noise than the naive and shift Gittins policies, highlighting the importance of robustness.

V. Bounding Response Time

We have carried out a case study on robustness over the course of Sections III and IV, starting with a policy, the naive Gittins policy, that is not robust to noise, and ending with a new variation of it, the shift-flat Gittins policy, that we argued is robust. Our remaining goal is to rigorously prove robustness of the shift-flat Gittins policy.

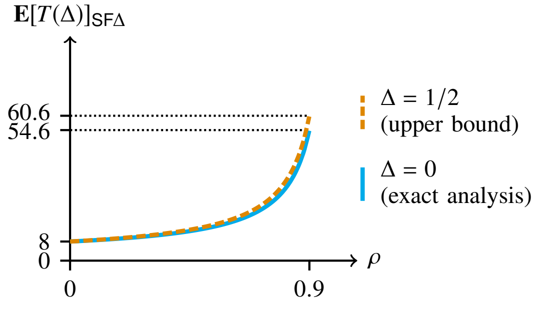


Fig. 6. Response Time of the Shift-Flat Gittins Policy

The first step of the robustness proof is upper bounding the mean response time of the shift-flat Gittins policy. In this section we prove a much more general result, giving an upper bound on the mean response time of *any SOAP bubble policy*. The bound, roughly speaking, is the same formula as the mean response time of an ordinary SOAP policy [1, Theorem 5.5], except every appearance of the rank function r is replaced with either $r^{\downarrow}$ or $r^{\uparrow}$, whichever results in a greater bound. It therefore behooves us to first review the analysis of ordinary SOAP policies.

A. SOAP Analysis Review

We consider an arbitrary SOAP policy with rank function r. Recall that jobs arrive at rate λ and have size distribution X. There is a universal formula giving mean response time under an arbitrary SOAP policy. The notation used in the formula will be explained in the remainder of this section.

Theorem 2 ([1, Theorem 5.5])**.** *Under any SOAP policy, the expected response time of a job of size x is*

$$\mathbf{E}[T_x] = \frac{\lambda \sum_{i=0}^{\infty} \mathbf{E}[(X_i^{\text{old}}[w_x])^2]}{2(1 - \rho_0^{\text{old}}[w_x])(1 - \rho^{\text{new}}[w_x])} + \int_0^x \frac{1}{1 - \rho^{\text{new}}[w_x(a)]}\, da,$$

where $w_x = w_x(0)$ and

$$\rho^{\text{new}}[w] = \lambda \mathbf{E}[X^{\text{new}}[w]] \qquad \rho_i^{\text{old}}[w] = \lambda \mathbf{E}[X_i^{\text{old}}[w]].$$

Theorem 2 is derived by considering a tagged job J of size x arriving to a steady-state system. It uses three pieces of notation we have yet to define:

- $w_x(a)$ is the *worst future rank* J will experience between its current age a and final size x.
- $\mathbf{E}[X^{\text{new}}[w]]$ is related to the amount of time each *new job*, meaning a job that arrives after J, delays J's completion.
- $\mathbf{E}[X_i^{\text{old}}[w]]$ is related to the amount of time each *old job*, meaning a job that was in the system when J arrived, delays J's completion.

We now define these three terms formally. Our presentation slightly simplifies that of Scully et al. [1], avoiding unnecessary generality and omitting some corner cases.

Definition 3. The *worst future rank* of a job of size x at age a is

$$w_x(a) = \sup_{a' \in [a,x)} r(a').$$

The importance of worst future rank lies in the *Pessimism Principle* [1, Section 4.3], which is a key step of the proof of Theorem 2. Roughly speaking, the Pessimism Principle states that we can replace J's current rank $r(a)$ with its worst future rank $w_x(a)$ without changing J's response time. More precisely, the Pessimism Principle gives J's delay due to each other job K, meaning the amount of time K is served before J completes, in terms of J's worst future rank:

> When J has age a, its remaining delay due to K is the remaining service time K needs to either complete or reach rank worse than $w_x(a)$.

To write down how much service a job needs to either complete or reach a rank cutoff, we need two new definitions.

Definition 4. Let w be a rank. The *new w-interval* is the interval of ages $[0, c[w])$ during which a new job has rank better than w. Specifically, we define

$$c[w] = \inf\{a \geq 0 \mid r(a) \geq w\}.$$

The *new w-work* is a random variable $X^{\text{new}}[w]$ representing how long a new job is served while its age is in the new w-interval. Specifically, we define

$$X^{\text{new}}[w] = \min\{X, c[w]\}.$$

Put another way, $X^{\text{new}}[w]$ is how long a new job is served until it either completes or has rank worse than w.

Definition 5. Let w be a rank. The *i-old w-interval* is the interval of ages $(b_i[w], c_i[w])$ during which an old job has rank better than w for the ith time. Specifically,[7]

$$b_0[w] = 0 \qquad c_0[w] = \inf\{a \geq 0 \mid r(a) > w\}$$

and, for all $i \geq 1$,

$$\begin{aligned} b_i[w] &= \inf\{a > c_i[w] - b_{i-1}[w] \mid r(a) \leq w\} \\ c_i[w] &= \inf\{a > b_i[w] \mid r(a) > w\}. \end{aligned}$$

See Fig. 7 for an illustration. The *i-old w-work* is the random variable $X_i^{\text{old}}[w]$ representing how long an old job is served while its age is in the i-old w-interval. Specifically,

$$X_i^{\text{old}}[w] = \begin{cases} 0 & \text{if } X \leq b_i[w] \\ X - b_i[w] & \text{if } X \in (b_i[w], c_i[w]) \\ c_i[w] - b_i[w] & \text{if } X \geq c_i[w]. \end{cases}$$

The Pessimism Principle immediately implies that the expected delay due to any new job is $\mathbf{E}[X^{\text{new}}[w_x(a)]]$, where a is J's age when the new job arrives. However, the situation for old jobs is more subtle because there may be several old jobs of various ages when J arrives. The details are outside the scope of this paper.

[7]There is a special case for $i = 0$ because the "0th time" is special: we say it occurs only if the old job starts with rank better than w.

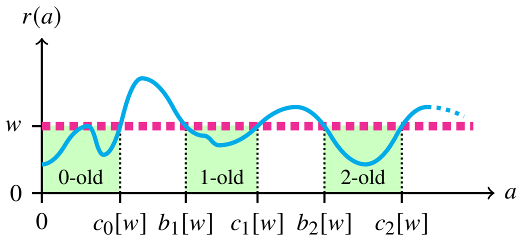


Fig. 7. Illustration of i-Old w-Intervals (Definition 5)

B. SOAP Bubble Response Time Bound

Having reviewed the analysis of ordinary SOAP policies, we now turn to the more difficult task of analyzing SOAP bubble policies. We consider an arbitrary SOAP bubble policy with rank bubble function $r^{\ddagger}$.

Considering a tagged job J of size x arriving to a steady-state system, we ask: what could the adversary do to maximize J's response time? Intuitively, the adversary should always assign J the maximum possible rank and always assign every other job the minimum possible rank. That is, the worst that can happen is that J effectively has rank function $r^{\uparrow}$ while every other job effectively has rank function $r^{\downarrow}$.

It turns out this intuition gives exactly the right answer. The bound in Theorem 6 below is nearly identical to the formula in Theorem 2. It has just two changes:

- Because J effectively has rank function $r^{\uparrow}$, we replace $w_x(a)$ with $w_x^{\uparrow}(a) = \sup_{a' \in [a,x)} r^{\uparrow}(a')$.
- Because all other jobs effectively have rank function $r^{\downarrow}$, we replace $X^{\text{new}}[w]$ and $X_i^{\text{old}}[w]$ with $X^{\text{new}\downarrow}[w]$ and $X_i^{\text{old}\downarrow}[w]$, respectively, which are defined as in Definitions 4 and 5 but using $r^{\downarrow}$ instead of r.

Theorem 6. *Under any SOAP bubble policy, the expected response time of a job of size x is bounded by*

$$\begin{aligned} \mathbf{E}[T_x] \leq{} & \frac{\lambda \sum_{i=0}^{\infty} \mathbf{E}[(X_i^{\text{old}\downarrow}[w_x^{\uparrow}])^2]}{2(1 - \rho_0^{\text{old}\downarrow}[w_x^{\uparrow}])(1 - \rho^{\text{new}\downarrow}[w_x^{\uparrow}])} \\ & + \int_0^x \frac{1}{1 - \rho^{\text{new}\downarrow}[w_x^{\uparrow}(a)]}\, da, \end{aligned}$$

where $w_x^{\uparrow} = w_x^{\uparrow}(0)$ and

$$\rho^{\text{new}\downarrow}[w] = \lambda \mathbf{E}[X^{\text{new}\downarrow}[w]] \qquad \rho_i^{\text{old}\downarrow}[w] = \lambda \mathbf{E}[X_i^{\text{old}\downarrow}[w]].$$

Along the same lines as Scully et al. [1, Theorem 5.4], the proof of Theorem 6 generalizes to give the Laplace-Stieltjes transform of a stochastic upper bound on response time.

Proof of Theorem 6. Let J be a tagged job of size x. Consider two possible strategies for the adversary:

- Arb, an arbitrary strategy; and
- Bad, the strategy that gives J rank function $r^{\uparrow}$ and gives every other job rank function $r^{\downarrow}$.

By definition, J's rank under rank under Arb is no worse than its rank under Bad, and vice versa for all other jobs. We will show that the distribution of J's response time under Arb is stochastically dominated by that under Bad. Given this, the result follows immediately from the fully general version of

Theorem 2 [1, Theorem 5.5], which allows different jobs to have different rank functions.[8]

It is clear that once J has arrived, giving J a worse rank or giving another job a better rank cannot possibly increase J's response time. It thus suffices to prove the following claim:

> The distribution of J's *delay due to old jobs* under Arb is stochastically dominated by that under Bad.

Proving this is nontrivial because the steady-state system under Arb might be different from that under Bad.

By the Pessimism Principle, the delay due to old jobs is the total service time required for each old job present when J arrives to either complete or exceed rank $w_x^\uparrow$. Of course, the age at which an old job might exceed rank $w_x^\uparrow$ depends on the adversary's strategy. The following definition lets us ignore some of the strategy's details.

Suppose the adversary is using strategy Q. At any time t, let the *potential old work*, denoted $W_Q(t)$, be the hypothetical delay due to old jobs if J were to arrive at t and the adversary were to thereafter use the Bad.

- Under Arb, the delay due to old jobs that J would experience if it arrived at time t is *upper bounded* by $W_{\mathsf{Arb}}(t)$, because Arb might sometimes give old jobs worse ranks than Bad would.
- Under Bad, the delay due to old jobs that J would experience if it arrived at time t is *exactly* $W_{\mathsf{Bad}}(t)$.

Consider an arrival sequence that does not include the tagged job J occurring in each of two systems, one using Arb and the other using Bad. For a fixed arrival sequence, both $W_{\mathsf{Arb}}(t)$ and $W_{\mathsf{Bad}}(t)$ are deterministic functions of t. Suppose J arrives at a uniformly random time T. By the above observations, it suffices to show $W_{\mathsf{Arb}}(T) \leq_{\mathrm{st}} W_{\mathsf{Bad}}(T)$, where $\leq_{\mathrm{st}}$ denotes stochastic domination.

To complete the proof, we will show $W_{\mathsf{Arb}}(T) \leq_{\mathrm{st}} W_{\mathsf{Bad}}(T)$. We divide old jobs into three categories:

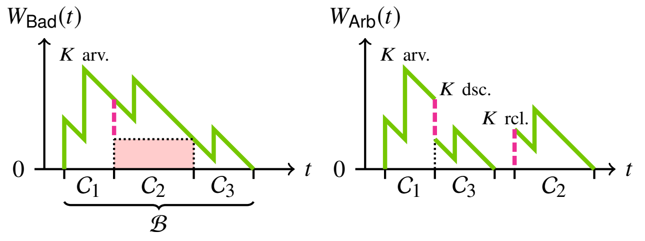
- *Discarded:* current rank greater than w_x.
- *Original:* current rank less than w_x, has never been discarded.
- *Recycled:* current rank less than w_x, has been discarded at some point in the past.

An important fact about these categories is that a job can only become recycled if there are no original or other recycled jobs in the system. This is because a job that is about to be recycled is currently discarded, so any original or recycled job would have better rank.

Consider a potential old work busy period $\mathcal{B}$ in the Bad system, meaning an interval of time during which $W_{\mathsf{Bad}}(t) > 0$, as shown on the left of Fig. 8. We call the interval of ages in which a job is original or recycled during $\mathcal{B}$ the job's *relevance interval*, which is a subset of an i-old w_x-interval.

For concreteness, we focus one job K, supposing that Arb differs from Bad only in its treatment of K. Under Bad, job K remains original for its entire relevance interval, but Arb may discard K in the middle of its relevance interval. Suppose

[8]Specifically, using the terminology of Scully et al. [1], we give the tagged job its own rank function by creating a unique descriptor for it that occurs with probability 0.



The potential old work (green line) in the Bad (left) and Arb (right) systems. Job K is the second job to arrive in the pictured time (K arv.). If Arb discards K, the potential old work jumps down (magenta dashed line at K dsc.) but later jumps back up the same amount when K is recycled (magenta dashed line at K rcl.). This decreases potential old work during C_2 (red region on left is absent on right).

Fig. 8. Potential Old Work in Coupled Systems

that K is discarded for just a single age. The effect of this discarding is shown on the right of Fig. 8. At the moment K is discarded, the Arb system suddenly loses potential old work. Once discarded, K is not served until the busy period ends, because K cannot become recycled while the system has potential old work. When K is finally served again, it is a recycled job starting a new busy period.

Because the Poisson arrival process has independent increments, we may couple the two systems' arrival sequences such that the overall effect of K's discarding is to chop a time interval out of $\mathcal{B}$ and move later, creating a new busy period. This chopped interval is C_2 in Fig. 8. Given a time t in the Bad system, let t' be the corresponding time in the Arb system. Notice that $W_{\mathsf{Arb}}(t') \leq W_{\mathsf{Bad}}(t)$ for $t \in C_2$ because the same arrivals occur but the amount of potential old work at the start of C_2 is higher in the Bad system. We have $W_{\mathsf{Arb}}(t') = W_{\mathsf{Bad}}(t)$ at all other times, so, recalling that T is a uniformly random time, $W_{\mathsf{Arb}}(T) \leq_{\mathrm{st}} W_{\mathsf{Bad}}(T)$.

We have shown $W_{\mathsf{Arb}}(T) \leq_{\mathrm{st}} W_{\mathsf{Bad}}(T)$ in the special case where Arb and Bad differ only in their treatment of a single job K, which Arb discards for a single age. We now address the general case.

- If K is discarded for a non-negligible age interval within its relevance interval, the same situation as shown in Fig. 8 occurs, except that K's return as a recycled job contributes less potential old work because K is not recycled until a later age, further decreasing $W_{\mathsf{Arb}}(T)$.
- Multiple jobs may be discarded during their relevance intervals, each possibly multiple times. Iterating the argument for each such discarding shows $W_{\mathsf{Arb}}(T) \leq_{\mathrm{st}} W_{\mathsf{Bad}}(T)$. □

VI. Proving Robustness

Having given a method for upper bounding the mean response time of any SOAP bubble policy, we now turn to proving robustness of the shift-flat Gittins policy. Once again, we prove a much more general result. The definition of the shift-flat Gittins policy in (3) does not use any specific properties of the naive Gittins policy, which means we can use the same definition for other policies.

Definition 7. Let P be a SOAP policy with rank function r_P. The policy *shift-flat* P, denoted $P\Delta$, is the perturbed SOAP policy with rank function

$$r_{P\Delta}(b) = \sup_{a \in [b-3\Delta, b-\Delta]} r_P(a).$$

Even though we created the shift-flat construction specifically motivated by the failure of the naive Gittins policy, we might ask if it always yields a robust policy. Remarkably, the answer is yes with one small change: we replace $=$ with $\leq$ in Definition 1. This is because if P is suboptimal, $P\Delta$ might "accidentally" improve upon P's mean response time.

Theorem 8. *For any SOAP policy P, shift-flat P is robust and therefore*

$$\lim_{\Delta \to 0} \mathbf{E}[T(\Delta)]_{P\Delta} \leq \mathbf{E}[T(0)]_P.$$

Proof. Throughout, SOAP notation (Section V-A) refers to P and SOAP bubble notation (Section V-B) refers to $P\Delta$. Let

$$f(\Delta) = \int_0^x \frac{1}{1-\rho^{\mathrm{new}\downarrow}[w_x^\uparrow(a)]}\,da - \int_0^x \frac{1}{1-\rho^{\mathrm{new}}[w_x(a)]}\,da$$

$$g(\Delta) = \frac{\lambda \sum_{i=0}^\infty \mathbf{E}[(X_i^{\mathrm{old}\downarrow}[w_x^\uparrow])^2]}{2(1-\rho_0^{\mathrm{old}\downarrow}[w_x^\uparrow])(1-\rho^{\mathrm{new}\downarrow}[w_x^\uparrow])} - \frac{\lambda \sum_{i=0}^\infty \mathbf{E}[(X_i^{\mathrm{old}}[w_x])^2]}{2(1-\rho_0^{\mathrm{old}}[w_x])(1-\rho^{\mathrm{new}}[w_x])},$$

where we use the same notation as in Theorems 2 and 6. The SOAP bubble notation on the right-hand sides implicitly depends on Δ. It suffices to show $\lim_{\Delta\to 0}(f(\Delta)+g(\Delta)) \leq 0$.

As a first step, note that $r_{P\Delta}^\uparrow(a) = \sup_{a\in[b-4\Delta,b]} r_P(a)$, implying $w_x^\uparrow(a) = w_x(a-4\Delta)$, In particular, $w_x^\uparrow(a) = w_x$ for $a \leq 4\Delta$, so

$$f(\Delta) \leq \frac{\Delta}{1-\rho^{\mathrm{new}\downarrow}[w_x]} + \int_0^{x-4\Delta} \left(\frac{1}{1-\rho^{\mathrm{new}\downarrow}[w_x(a)]} - \frac{1}{1-\rho^{\mathrm{new}}[w_x(a)]} \right) da.$$

In the $\Delta \to 0$ limit, the first term is 0, and the second term is nonpositive if $\lim_{\Delta\to 0} \mathbf{E}[X^{\mathrm{new}\downarrow}[w]] - \mathbf{E}[X^{\mathrm{new}}[w]] \leq 0$ for any rank w. We show this by relating $c^\downarrow[w]$ to $c[w]$ (Definition 4).

By (1) and Definition 7, for any age a,

$$r_{P\Delta}^\downarrow(a+2\Delta) \geq r_P(a). \tag{4}$$

Let w be a rank. We know $r_{P\Delta}^\downarrow(c[w]+2\Delta) \geq w$ by (4), so $c^\downarrow[w] \leq c[w] + 2\Delta$. This means $X^{\mathrm{new}\downarrow}[w] \leq_{\mathrm{st}} X^{\mathrm{new}}[w] + 2\Delta$, from which the desired bound follows in the $\Delta \to 0$ limit.

We now bound $g(\Delta)$ in the $\Delta \to 0$ limit. Writing $w = w_x$ for brevity, it suffices to show

- $\lim_{\Delta\to 0} \mathbf{E}[X_0^{\mathrm{old}\downarrow}[w]] - \mathbf{E}[X_0^{\mathrm{old}}[w]] \leq 0$ and
- $\lim_{\Delta\to 0} \mathbf{E}[(X_i^{\mathrm{old}\downarrow}[w])^2] - \mathbf{E}[(X_{\sigma_\Delta(i)}^{\mathrm{old}}[w])^2] \leq 0$ for all i, where $\sigma_\Delta : \mathbb{N} \to \mathbb{N}$ is an injection varying with Δ.

Nearly the same argument as for $X^{\mathrm{new}}[w]$ and $X^{\mathrm{new}\downarrow}[w]$ above shows that $X_0^{\mathrm{old}\downarrow}[w] \leq_{\mathrm{st}} X_0^{\mathrm{old}}[w] + 2\Delta$, which implies the first item above and, setting $\sigma_\Delta(0) = 0$, also implies the second item for $i = 0$.

Let $i \geq 1$. The i-old w-interval of $P\Delta$ is $(b_i^\downarrow[w], c_i^\downarrow[w])$ (Definition 5). By (1), every local minimum of $r_{P\Delta}^\downarrow$ is attained for an interval of length greater than 2Δ, which means $c_i^\downarrow[w] - b_i^\downarrow[w] > 2\Delta$.[9] Using (1) again with Definitions 4 and 7, we have

$$\begin{aligned} & r_{P\Delta}^\downarrow(a) \leq w \text{ for all } a \in (b_i^\downarrow[w], c_i^\downarrow[w]) \\ \Leftrightarrow\quad & r_{P\Delta}(a) \leq w \text{ for all } a \in (b_i^\downarrow[w]+\Delta, c_i^\downarrow[w]-\Delta) \\ \Leftrightarrow\quad & r_P(a) \leq w \text{ for all } a \in (b_i^\downarrow[w]-2\Delta, c_i^\downarrow[w]-2\Delta), \end{aligned}$$

so $(b_i^\downarrow[w]-2\Delta, c_i^\downarrow[w]-2\Delta)$ is, for some j, the j-old w-interval of P. Jobs are less likely to reach greater ages, so $X_i^{\mathrm{old}\downarrow}[w] \leq_{\mathrm{st}} X_j^{\mathrm{old}}[w]$. Thus, defining $\sigma_\Delta(i) = j$ suffices.[10] □

VII. Related Work

The work with the closest goals to ours is the study of so-called ε-SMART policies by Wierman and Nuyens [5]. The SMART class [6] consists of size-based policies that achieve small response time. The ε-SMART class consists of SMART policies applied in a setting where job sizes are not exactly known, with a function ε determining the uncertainty. The ε-SMART work differs from ours in that it deals with size uncertainty rather than age uncertainty.

There has been work on variants of the Gittins policy which, instead of scheduling the job of maximal Gittins index $g_{\max}$, schedule a job with Gittins index $g \geq g_{\max} - \delta$ for some $\delta > 0$. See Gittins et al. [3, Section 4.10] and the references therein for a summary of results. Unfortunately, the bounds do not apply to mean response time because they require discounting future costs. Moreover, age uncertainty can lead to scheduling decisions where there is an arbitrarily large difference between g and $g_{\max}$.

References

[1] Z. Scully, M. Harchol-Balter, and A. Scheller-Wolf, "SOAP: One clean analysis of all age-based scheduling policies," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 2, no. 1, pp. 16:1–16:30, Apr. 2018.

[2] J. C. Gittins and D. M. Jones, "A dynamic allocation index for the sequential design of experiments," in *Progress in Statistics*, J. Gani, Ed. Amsterdam, NL: North-Holland, 1974, pp. 241–266.

[3] J. C. Gittins, K. D. Glazebrook, and R. Weber, *Multi-armed Bandit Allocation Indices*. John Wiley & Sons, 2011.

[4] S. Aalto, U. Ayesta, and R. Righter, "On the Gittins index in the M/G/1 queue," *Queueing Systems*, vol. 63, no. 1, pp. 437–458, 2009.

[5] A. Wierman and M. Nuyens, "Scheduling despite inexact job-size information," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 36, no. 1. ACM, 2008, pp. 25–36.

[6] A. Wierman, M. Harchol-Balter, and T. Osogami, "Nearly insensitive bounds on SMART scheduling," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 33, no. 1. ACM, 2005, pp. 205–216.

[9] Actually, it is possible to have $c_i^\downarrow[w] - b_i^\downarrow[w] = 2\Delta$. It turns out in this case that $r_{P\Delta}^\downarrow(a) \leq w$ holds for all $a \in [b_i^\downarrow[w], c_i^\downarrow[w]]$, so we simply use closed intervals in the implication chain below.

[10] One can show by a similar chain of implications that if $j = 0$, then $r_{P\Delta}^\downarrow(a) \leq w$ for all $a < c_i^\downarrow[w]$ and thus $i = 0$.