Simple Near-Optimal Scheduling for the M/G/1

Ziv Scully
Computer Science Department
Carnegie Mellon University
zscully@cs.cmu.edu

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

Alan Scheller-Wolf
Tepper School of Business
Carnegie Mellon University
awolf@andrew.cmu.edu

ABSTRACT

We consider the problem of preemptively scheduling jobs to minimize mean response time of an M/G/1 queue. When the scheduler knows each job's size, the *shortest remaining processing time* (SRPT) policy is optimal. Unfortunately, in many settings we do not have access to each job's size. Instead, we know only the job size distribution. In this setting, the *Gittins* policy is known to minimize mean response time, but its complex priority structure can be computationally intractable. A much simpler alternative to Gittins is the *shortest expected remaining processing time* (SERPT) policy. While SERPT is a natural extension of SRPT to unknown job sizes, it is unknown how close SERPT is to optimal.

We present a new variant of SERPT called *monotonic SERPT* (M-SERPT) which is as simple as SERPT but has provably near-optimal mean response time at all loads for any job size distribution. Specifically, we prove the mean response time ratio between M-SERPT and Gittins is at most 3 for load $\rho \leq 8/9$ and at most 5 for any load. This makes M-SERPT the only scheduling policy known to be a constant-factor approximation of Gittins.

1. INTRODUCTION

Scheduling to minimize mean response time in a preemptive M/G/1 queue is a classic problem in queueing theory. When job sizes are known, the *shortest remaining processing time* (SRPT) policy is known to minimize mean response time. Unfortunately, determining or estimating a job's exact size is difficult or impossible in many applications, in which case SRPT is impossible to implement. In such cases we only learn jobs' sizes after they have completed, which can give us a good estimate of the *distribution* of job sizes.

When individual job sizes are unknown but the job size distribution is known, the *Gittins* policy minimizes mean response time [1, 5]. Gittins has a seemingly simple structure:

- Based on the job size distribution, Gittins defines a *rank function* that maps a job's *age*, which is the amount of service it has received so far, to a *rank*, which denotes its priority [12].
- At every instant, Gittins applies the rank function to each job's age and serves the job with the best rank.

This work was supported by NSF-CSR-1763701, NSF-XPS-1629444 and a Microsoft Faculty Award 2018. Ziv Scully was supported by the NSF GRFP under grants DGE-1745016 and DGE-125222 and an ARCS Foundation scholarship.

Unfortunately, hidden in this simple outline is a major obstacle: computing the rank function from the job size distribution requires solving a nonconvex optimization problem for every possible age. Although the optimization can be simplified for specific classes of job size distributions [1], it is intractable in general.

In light of the difficulty of computing the Gittins rank function, practitioners turn to a wide variety of simpler scheduling policies, each of which has good performance in certain settings. Three of the most famous are the following:

- *First-come, first-serve* (FCFS) serves jobs nonpreemptively in the order they arrive.
  - FCFS generally performs well for low-variance job size distributions and is optimal for those with the decreasing mean residual lifetime property [1, 10].
- *Foreground-background* (FB) always serves the job of minimal age, splitting the server evenly in case of ties.
  - FB generally performs well for high-variance job size distributions and is optimal for those with the decreasing hazard rate property [1, 4, 9, 10].
- *Processor sharing* (PS) splits the server evenly between all jobs currently in the system.
  - PS has appealing insensitivity and fairness properties which ensure passable mean response time for all job size distributions, but it is only optimal in the trivial special case of exponential job size distributions.

These are a few of the many scheduling heuristics studied over the past several decades. Unfortunately, there are *no optimality guarantees for any policy* other than Gittins that hold across all job size distributions. In fact, each of FCFS, FB, and PS can have *arbitrarily large* mean response time ratio compared to Gittins.[1] We therefore ask:

> Is there a *simple* scheduling policy that achieves near-optimal mean response time for all job size distributions?

One candidate for such a policy is *shortest expected remaining processing time* (SERPT). Like Gittins, SERPT assigns each job a rank as a function of its age, but SERPT has a much simpler rank function: a job's rank is its *expected remaining size.* That is, if the job size distribution is X, then under SERPT, a job's rank at age a is

$$r_{\mathsf{SERPT}}(a) = \mathbf{E}[X - a \mid X > a],$$

where lower rank means better priority. Intuitively, it seems like SERPT should have low mean response time because it prioritizes jobs that are short in expectation, analogous to

[1] For PS and FB, this happens in the heavy-traffic $\rho \to 1$ limit [7].

Table 1.1: Scheduling Policy Comparison

Policy	Computation		Performance
	Discrete	*Continuous*	
Gittins	$\Omega(n^2)$	intractable	optimal
SERPT	$O(n)$	tractable	unknown
M-SERPT	$O(n)$	tractable	5-approximation

what SRPT does for known job sizes. SERPT is certainly much simpler than Gittins, as shown in Table 1.1.

- For discrete job size distributions with n support points, Gittins's rank function can be piecewise linear with $\Omega(n^2)$ segments, thus taking $\Omega(n^2)$ time to compute, while SERPT takes $O(n)$ time to compute.
- For continuous job size distributions, computing Gittins's rank function at a single age a requires solving a nonconvex optimization problem, while SERPT needs just numerical integration.

1.1 Challenges

SERPT is intuitively appealing and simple to compute, but does it have near-optimal mean response time? This question is open: there is *no known bound* on the performance gap between SERPT and Gittins. To be precise, letting

$$C_{\mathsf{SERPT}}(X) = \frac{\mathbf{E}[T_{\mathsf{SERPT}}(X)]}{\mathbf{E}[T_{\mathsf{Gittins}}(X)]}$$

be the mean response time ratio between SERPT and Gittins for a given job size distribution X, there is no bound on

$$\text{approximation ratio of SERPT} = \sup_X C_{\mathsf{SERPT}}(X).$$

This approximation ratio is difficult to bound because we have to consider *all* possible job size distributions X.

In fact, until recently it was unknown how to compute $C_{\mathsf{SERPT}}(X)$ even given a *specific* job size distribution X. This changed with the introduction of the *SOAP* technique [12], which can analyze the mean response time of any scheduling policy that can be specified by a rank function. We can use SOAP to *numerically* compute $C_{\mathsf{SERPT}}(X)$ for any given job size distribution X. But SOAP does not bound SERPT's approximation ratio, which requires considering all possible X.

One might hope to derive a general algebraic expression for $C_{\mathsf{SERPT}}(X)$ using SOAP. While this is possible in principle, the resulting expression appears intractable. To deal with this, our strategy is to design a new scheduling policy that captures the essence of SERPT but has a tractable mean response time expression.

1.2 Our Contribution: M-SERPT

In this paper we introduce a new policy called *monotonic SERPT* (M-SERPT) that is simple to compute and has provably near-optimal mean response time. Like Gittins and SERPT, we specify M-SERPT using a rank function. Roughly speaking, M-SERPT has the same rank function as SERPT, except *a job's rank never improves:*

$$r_{\mathsf{M\text{-}SERPT}}(a) = \max_{0 \le b \le a} r_{\mathsf{SERPT}}(b).$$

We prove that M-SERPT is a 5-approximation, meaning its mean response time is at most 5 times that of Gittins. This makes M-SERPT the first scheduling policy known to have a constant-factor approximation ratio. The approximation ratio is even smaller at lower system loads. For example, M-SERPT is a 3-approximation for load $\rho < 8/9$. M-SERPT achieves this near-optimal mean response time with a rank function that is as simple to compute as SERPT's (Table 1.1).

One might wonder how M-SERPT's performance compares to SERPT. In numerical investigations, which we omit for lack of space, we have found them to have comparable mean response time, with each performing better than the other for some job size distributions. In fact, both have mean response time within about 10% of Gittins's outside of a few pathological job size distributions. Of course, our approximation ratio proof only applies to M-SERPT, but we conjecture that SERPT also has a constant-factor approximation ratio.

We state our main result in Theorem 3.3. Its proof, which we omit for lack of space, is in the full version of this work [13].

1.3 Related Work

In this paper we consider minimizing mean response time in the setting of an M/G/1 queue with unknown job sizes. We are not aware of prior work on approximation ratios in this exact setting, but there is prior work in related settings.

We begin with the M/G/1 with *known* job sizes. Wierman et al. [14] prove that all scheduling policies in a class called *SMART* are 2-approximations for mean response time, where the baseline for this setting is SRPT [11]. All SMART policies use job size information, so they cannot be applied to our setting of unknown job sizes. Proving approximation ratios in our setting is significantly more challenging because the scheduling policies involved, namely M-SERPT and Gittins, have much more complicated mean response time formulas than SRPT and the SMART class [12].

We now move to settings with unknown job sizes. For such settings, Kalyanasundaram and Pruhs [6] propose the *randomized multilevel feedback* (RMLF) policy to achieve low mean response time. RMLF has been studied in two settings:

- In the *worst-case* setting in which job sizes and arrival times are chosen adversarially, RMLF has mean response time $\Theta(\log n)$ times that of SRPT, where n is the number of jobs in the arrival sequence [3, 6], which is the best possible performance in this setting [8].
- In the *stochastic* GI/GI/1 setting, Bansal et al. [2] prove that as the system load ρ approaches 1,

$$\frac{\mathbf{E}[T_{\mathsf{RMLF}}]}{\mathbf{E}[T_{\mathsf{SRPT}}]} = O\left(\log \frac{1}{1-\rho}\right). \tag{1.1}$$

These results are different from ours in two important ways. First, the results do not prove constant approximation ratios: they give asymptotic ratios that become arbitrarily large in the $n \to \infty$ and $\rho \to 1$ limits, respectively. We show that M-SERPT is a 5-approximation at all loads ρ, even in the $\rho \to 1$ limit. Second, the results compare RMLF with SRPT, not Gittins, even though job sizes are unknown. This is because the optimal policies for the worst-case and GI/GI/1 settings are not known, so SRPT is the only baseline available for comparison. In contrast, in the M/G/1 setting we know Gittins is optimal, so we directly compare M-SERPT to Gittins.

2. SYSTEM MODEL AND NOTATION

We consider scheduling in an M/G/1 queue where jobs have unknown size. We write λ for the arrival rate and X for the job size distribution, so the system load is $\rho = \lambda \mathbf{E}[X]$. We assume $\rho < 1$. Jobs may be preempted at any time without delay or loss of work. We write $\mathbf{E}[T]$ for the mean response time of the system.

The scheduling policies we study in this paper are all *SOAP policies* [12]. A SOAP policy is one that is specified by a *rank function*

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

which maps a job's *age*, the amount of time it has been served, to its *rank*, or priority. All SOAP policies have the same core scheduling rule: always serve the job of *minimum rank*, breaking ties in first-come, first served order.

Definition 2.1. The *Gittins* policy is the SOAP policy with rank function

$$r_{\mathsf{Gittins}}(a) = \inf_{b>a} \frac{\int_a^b \overline{F}(t)\,\mathrm{d}t}{\overline{F}(a) - \overline{F}(b)}.$$

Definition 2.2. The *monotonic SERPT* (M-SERPT) policy is the SOAP policy with nondecreasing rank function

$$r_{\mathsf{M\text{-}SERPT}}(a) = \max_{0 \leq b \leq a} r_{\mathsf{SERPT}}(b) = \max_{0 \leq b \leq a} \mathbf{E}[X - b \mid X > b].$$

3. APPROXIMATION RATIO OF M-SERPT

In this section we outline our analysis of M-SERPT. See the full version of this work for proofs [13], which we omit here for lack of space.

In their analysis of SOAP policies, Scully et al. [12] split mean response time into two parts: *mean waiting time* $\mathbf{E}[Q]$ and *mean residence time* $\mathbf{E}[R]$. Roughly speaking, a job's waiting time is the time from arrival until it is first served, and a job's residence time is the time from that first service until completion.[2]

Our approach is to bound M-SERPT's mean waiting and residence times separately. The first step is to compare the mean waiting times of M-SERPT and Gittins.

Lemma 3.1. *M-SERPT's mean waiting time is bounded by*

$$\mathbf{E}[Q_{\mathsf{M\text{-}SERPT}}] \leq \frac{2}{1 + \sqrt{1-\rho}} \mathbf{E}[Q_{\mathsf{Gittins}}].$$

The second step is to prove an upper bound on the mean residence time under M-SERPT.

Lemma 3.2. *M-SERPT's mean residence time is bounded by*

$$\mathbf{E}[R_{\mathsf{M\text{-}SERPT}}] \leq \mathbf{E}[Q_{\mathsf{M\text{-}SERPT}}] + \left(\frac{1}{\rho} \log \frac{1}{1-\rho}\right) \mathbf{E}[X].$$

The last step is to bound the mean residence time under Gittins. Two bounds based on prior work suffice:

- A job's residence time is at least its size, so

$$\mathbf{E}[R_{\mathsf{Gittins}}] \geq \mathbf{E}[X].$$

- Wierman et al. [14] give the following lower bound on the mean response time of SRPT, which in turn is a lower bound on the mean response time of Gittins:

$$\mathbf{E}[T_{\mathsf{Gittins}}] \geq \mathbf{E}[T_{\mathsf{SRPT}}] \geq \left(\frac{1}{\rho} \log \frac{1}{1-\rho}\right) \mathbf{E}[X].$$

Combining these bounds with Lemmas 3.1 and 3.2 bounds the approximation ratio of M-SERPT compared to Gittins.

[2]The formal definitions of waiting and residence times are more complicated than this rough explanation, but this suffices to explain our method. See Scully et al. [12] for details.

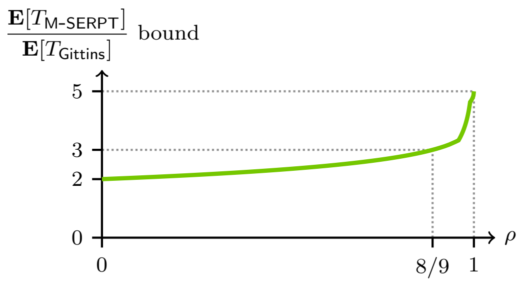

Figure 3.1: Bound on Mean Response Time Ratio

Theorem 3.3. *M-SERPT's approximation ratio is at most*

$$\frac{\mathbf{E}[T_{\mathsf{M\text{-}SERPT}}]}{\mathbf{E}[T_{\mathsf{Gittins}}]} \leq \begin{cases} \dfrac{4}{1+\sqrt{1-\rho}} & 0 \leq \rho < 0.9587 \\ \dfrac{1}{\rho} \log \dfrac{1}{1-\rho} & 0.9587 \leq \rho < 0.9898 \\ 1 + \dfrac{4}{1+\sqrt{1-\rho}} & 0.9898 \leq \rho < 1. \end{cases}$$

Figure 3.1 illustrates the bound as a function of load ρ. Concretely, M-SERPT is a 3-approximation for load $\rho \leq 8/9$ and a 5-approximation for all loads.

References

[1] S. Aalto, U. Ayesta, and R. Righter. On the Gittins index in the M/G/1 queue. *Queueing Systems*, 63(1):437–458, 2009.

[2] N. Bansal, B. Kamphorst, and B. Zwart. Achievable performance of blind policies in heavy traffic. *Mathematics of Operations Research*, 43(3):949–964, 2018.

[3] L. Becchetti and S. Leonardi. Nonclairvoyant scheduling to minimize the total flow time on single and parallel machines. *Journal of the ACM (JACM)*, 51(4):517–539, 2004.

[4] H. Feng and V. Misra. Mixed scheduling disciplines for network flows. In *ACM SIGMETRICS Performance Evaluation Review*, volume 31, 36–39. ACM, 2003.

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

[6] B. Kalyanasundaram and K. R. Pruhs. Minimizing flow time nonclairvoyantly. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, 345–352. IEEE, 1997.

[7] B. Kamphorst and B. Zwart. Heavy-traffic analysis of sojourn time under the foreground-background scheduling policy. *arXiv preprint arXiv:1712.03853*, 2017.

[8] R. Motwani, S. Phillips, and E. Torng. Nonclairvoyant scheduling. *Theor. Comput. Sci.*, 130(1):17–47, 1994.

[9] R. Righter and J. G. Shanthikumar. Scheduling multiclass single server queueing systems to stochastically maximize the number of successful departures. *Probability in the Engineering and Informational Sciences*, 3(3):323–333, 1989.

[10] R. Righter, J. G. Shanthikumar, and G. Yamazaki. On extremal service disciplines in single-stage queueing systems. *Journal of Applied Probability*, 27(2):409–416, 1990.

[11] L. Schrage. A proof of the optimality of the shortest remaining processing time discipline. *Operations Research*, 16(3):687–690, 1968.

[12] 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.*, 2(1):16:1–16:30, Apr. 2018.

[13] Z. Scully, M. Harchol-Balter, and A. Scheller-Wolf. Simple near-optimal scheduling for the M/G/1. *arXiv preprint arXiv:1907.10792*, 2019.

[14] A. Wierman, M. Harchol-Balter, and T. Osogami. Nearly insensitive bounds on SMART scheduling. In *ACM SIGMETRICS Performance Evaluation Review*, volume 33, 205–216. ACM, 2005.