2021 19th International Symposium on Modeling and Optimization in Mobile, Ad hoc, and Wireless Networks (WiOpt)

The Gittins Policy in the M/G/1 Queue

Ziv Scully*
Carnegie Mellon University
Pittsburgh, PA, USA
zscully@cs.cmu.edu

Mor Harchol-Balter*
Carnegie Mellon University
Pittsburgh, PA, USA
harchol@cs.cmu.edu

***Abstract*—The Gittins policy is a highly general scheduling policy that minimizes a wide variety of mean holding cost metrics in the M/G/1 queue. Perhaps most famously, Gittins minimizes mean response time in the M/G/1 when jobs' service times are unknown to the scheduler. Gittins also minimizes weighted versions of mean response time. For example, the well-known "$c\mu$ rule", which minimizes class-weighted mean response time in the multiclass M/M/1, is a special case of Gittins.**

However, despite the extensive literature on Gittins in the M/G/1, it contains no fully general proof of Gittins's optimality. This is because Gittins was originally developed for the multi-armed bandit problem. Translating arguments from the multi-armed bandit to the M/G/1 is technically demanding, so it has only been done rigorously in some special cases. The extent of Gittins's optimality in the M/G/1 is thus not entirely clear.

In this work we provide the first fully general proof of Gittins's optimality in the M/G/1. The optimality result we obtain is even more general than was previously known. For example, we show that Gittins minimizes mean slowdown in the M/G/1 with unknown or partially known service times, and we show that Gittins's optimality holds under batch arrivals. Our proof uses a novel approach that works directly with the M/G/1, avoiding the difficulties of translating from the multi-armed bandit problem.

I. Introduction

Scheduling to minimize mean holding cost in queueing systems is an important problem. Minimizing metrics such as mean response time, weighted mean response time, and mean slowdown can all be viewed as special cases of minimizing holding cost [1].[1] In single-server queueing systems, specifically the M/G/1 and similar systems, a number of scheduling policies minimize mean holding cost in various special cases. Two famous examples are the *Shortest Remaining Processing Time (SRPT)* policy, which minimizes mean response time when service times are known to the scheduler, and the "$c\mu$ rule", which minimizes weighted mean response time in the multiclass M/M/1 with unknown service times.

It turns out that there is a policy that minimizes mean holding cost in the M/G/1 under very general conditions. This policy, now known as the *Gittins* policy after one of its principal creators [2], has a relatively simple form. Gittins assigns each job an *index*, which is a rating roughly corresponding to how valuable it would be to serve that job. A job's index depends only on its own state, not the state of any other jobs. Gittins then serves the job of maximal index at every decision time. The "magic" of Gittins is in how it determines each job's index, which we describe in detail in Section IV. SRPT and the $c\mu$ rule are both special cases of Gittins.

TABLE I.1
Gittins Optimality Results for M/G/1-Like Queues

Holding Cost	*Model*	*Preemption*	*Service Times*	*Prior Proofs*[a]
All equal	M/G/1	allowed	known	1
	$M/G_{MP}/1$	allowed	see Section III	11
By class	M/G/1	allowed	unknown	4, 8
	M/G/1+fbk[b]	not allowed	unknown	6, 9
	M/M/1+fbk[b]	allowed	unknown	7, 9, 10
By class and service time[c]	M/G/1	not allowed	known	2
	M/G/1	allowed	known	3
	M/G/1	allowed	unknown	5

[a] A list of prior proofs appears in Section II.
[b] Here "fbk" stands for *feedback*, meaning that whenever a job exits the system, it has some probability of being replaced by another job.
[c] This includes minimizing mean *slowdown*, in which a job's holding cost is the reciprocal of its service time.

Given its generality, it is perhaps unsurprising that the Gittins policy has been discovered several times. Similarly, there are several proofs of its optimality under varying conditions. Table I.1 summarizes several M/G/1-like settings in which Gittins has been studied, and Section II gives a more detailed overview. In light of this substantial body of prior work, the optimality of Gittins for minimizing mean holding cost in the M/G/1 is widely accepted in the literature [3–7].

We ourselves are researchers whose work often cites Gittins's optimality in the M/G/1. However, in reviewing the literature, we found that there is *no complete proof of Gittins's optimality in its full generality*. This is in part because Gittins was originally developed not for the M/G/1 but for the Markovian multi-armed bandit problem [8]. There are elegant arguments for Gittins's optimality in the multi-armed bandit problem, but they do not easily translate to the M/G/1. Results for the M/G/1 thus suffer from a variety of limitations (Section II), so the extent of Gittins's optimality in the M/G/1 is not entirely clear.

In this work, we give *a unifying presentation of the Gittins policy* in M/G/1-like systems, resulting in the most general definition of Gittins (Definition IV.2) and optimality theorem (Theorem V.1) to date. Our approach deals directly with the M/G/1, avoiding the difficulties of translating from the multi-armed bandit problem. As a result, we actually *extend the known scope* of Gittins's optimality, such as including systems

*Supported by NSF grant nos. CMMI-1938909 and CSR-1763701 and a Google Faculty Award.

[1] A job's *response time* is the amount of time between its arrival and completion. Jobs may be sorted into *classes* which are weighted by importance. A job's *slowdown* is the ratio between its response time and service time.

ISBN 978-3-903176-37-0 © 2021 IFIP

with batch arrivals. We make the following contributions:

- We discuss many prior proofs of Gittins's optimality, detailing the limitations of each one (Section II).
- We give a new general definition of the Gittins policy (Section IV). This involves introducing a new generalization of the M/G/1 called the *M_B/G_{MP}/1 queue* (Section III).
- We state (Section V) and prove (Sections VI and VII) Gittins's optimality in the $M_B/G_{MP}/1$.

II. History of the Gittins Policy in the M/G/1

In this section we review prior work on the Gittins policy in M/G/1-like queues. This includes work on special cases of Gittins, such as SRPT in the case of known service times, that are not typically thought of as instances of the Gittins policy. Unfortunately, every prior proof of Gittins's optimality is limited in some way. Most limitations are one of the following:

(i) *Job finiteness.* Most proofs assume some type of "finiteness" of the job model. This manifests as one of
  (i-a) all service times being less than some finite bound,
  (i-b) service time distributions being discrete with finitely many support points, or
  (i-c) finitely many job classes.
(ii) *Simple job model or metric.* Some proof techniques that work for simple job models do not readily generalize. This includes models with
  (ii-a) known service times,
  (ii-b) unknown, exponentially distributed service times, or
  (ii-c) unknown, generally distributed service times with nonpreemptive service.
(iii) *Only considers index policies.* Some proofs only show that Gittins is an optimal *index policy*, as opposed to optimal among all policies. An index policy is one that, like Gittins, assigns each job an index based on the job's state and always serves the job of maximum index.

We now present prior work on the Gittins policy in rough chronological order. Due to space limitations, we only briefly summarize most prior proofs.

A. 1960s and 1970s: Initial Developments in Queueing

Prior Proof 1. Schrage [9].
Model: Preemptive single-server queue, known service times.
Holding costs: Same for all jobs.
Limitations: (ii-a).

Prior Proof 2. Fife [10, Section 4].
Model: Nonpreemptive M/G/1, known service times.
Holding costs: Based on class and service time.
Limitations: (i-b), (i-c), (ii-a), and (ii-c)

Prior Proof 3. Sevcik [11, Theorem 4-1].
Model: Preemptive M/G/1, known service times.
Holding costs: Based on class and service time.
Limitations: (ii-a) and (iii). Sevcik [11, Conjecture 4-1] argues informally that an index policy should be optimal.

Prior Proof 4. Sevcik [11, Theorem 4-2].
Model: Preemptive M/G/1, unknown service times.
Holding costs: Based on class.
Limitations: (i-b), (i-c), and (iii). Sevcik [11, Conjecture 4-3] argues informally that an index policy should be optimal.

Prior Proof 5. Von Olivier [12].
Model: Preemptive M/G/1, unknown service times.
Holding costs: Based on class and service time.
Limitations: (i-b), (i-c), and (iii).

One unique aspect of the von Olivier [12] result deserves highlighting: jobs' holding costs can depend on their unknown service times. This allows minimizing metrics like mean slowdown even when service times are unknown. However, this result is not widely known in the queueing theory community, perhaps in part because it has only been published in German.

Klimov [13] studied a nonpreemptive M/G/1 with *feedback*, denoted M/G/1+fbk. In systems with feedback, whenever a job exits the system, it has some probability of immediately returning as another job, possibly of a different class.

Prior Proof 6. Klimov [13].
Model: Nonpreemptive M/G/1+fbk, unknown service times.
Holding costs: Based on class.
Limitations: (i-c) and (ii-c).

B. 1980s and 1990s: Connection to Multi-Armed Bandits

Prior Proof 7. Lai and Ying [14].
Model: Preemptive M/M/1+fbk, unknown service times.
Holding costs: Based on class.
Limitations: (i-c) and (ii-b).

Prior Proof 8. Gittins [2, Theorem 5.6].
Model: Preemptive M/G/1, unknown service times.
Holding costs: Based on class.
Limitations: (i-a) and (i-c).

Gittins's result [2] is often cited in the literature as proving the Gittins policy's optimality in the M/G/1 [3–7]. As such, it deserves some more detailed discussion.

Prior Proof 8 has two main steps. The first step simplifies the problem by assuming the scheduler can only preempt jobs in a discrete set of states[2] [2, Theorem 3.28]. The set can be countable in principle, but the proof assumes a side condition that is only guaranteed to hold if the set is finite. The condition comes from translating multi-armed bandit results to the M/G/1.

The second step uses a limit argument to allow unrestricted preemption [2, Theorem 5.6]. However, because the first step is limited to finitely many job states, the second step's result is also limited. Specifically, it requires finitely many classes and that all service times be less than some finite bound.

Prior Proof 9. Achievable region approaches. See Bertsimas [15], Dacre et al. [16], and references therein.
Model: Preemptive M/M/1+fbk or nonpreemptive M/G/1+fbk, unknown service times.
Holding costs: Based on class.
Limitations: (i-c), (ii-b), and (ii-c).

[2]In this setting, a job's state is the pair of its class and attained service.

ISBN 978-3-903176-37-0 © 2021 IFIP

C. 2000s and 2010s: Analyzing Gittins and Its Performance

The 2000s and 2010s did not, for the most part, see new proofs of Gittins's optimality. Researchers instead studied properties of the Gittins policy [3] and analyzed its performance [5–7, 17]. One performance analysis based on dynamic programming also gave a new optimality proof [17], but it did not expand the known scope of Gittins's optimality.

Prior Proof 10. Whittle [17].
Model: Preemptive M/M/1+fbk, unknown service times.
Holding costs: Based on class.
Limitations: (i-c) and (ii-b).

D. 2020: Modeling Jobs as General Markov Processes

Prior Proof 11. Scully et al. [18, Theorem 7.3].
Model: Preemptive M/G_{MP}/1, i.e. the preemptive $M_B/G_{MP}/1$ (Section III) without batch arrivals.
Holding costs: Same for all jobs.
Limitations: Assumes equal holding costs and that jobs are preemptible in any state.

Our work can be seen as a significant extension of Prior Proof 11. Specific aspects we address that Scully et al. [18] do not include varying holding costs, nonpreemptible or partially preemptible jobs, and batch arrivals.

III. System Model: the $M_B/G_{MP}/1$ Queue

We study scheduling in a generalization of the M/G/1 queue to minimize a variety of mean holding cost metrics. The average job arrival rate is λ, the service time distribution is S, and the load is $\rho = \lambda \mathbf{E}[S]$. We assume $\rho < 1$ for stability.

We call our model the *$M_B/G_{MP}/1$ queue*. The "M_B" indicates that jobs arrive in batches with Poisson arrival times. The "G_{MP}" indicates generally distributed service times, with each job's service time arising from an underlying Markov process.

The main feature of the $M_B/G_{MP}/1$ is that it models jobs as Markov processes. The key intuition is:

> A job's *state* encodes all information the scheduler knows about the job.

This means that the job Markov process differs depending on what information the scheduler knows. For example, to model the perfect-information case where the scheduler is told every job's service time when it arrives, a job's state might be its remaining service time, and the Markov process dynamics would be deterministic (Example III.1). On the other extreme, if the scheduler knows nothing other than the overall service time distribution S, then a job's state might be the amount of service it has received so far, and the Markov process dynamics would be stochastic (Example III.2). The $M_B/G_{MP}/1$ thus encompasses a wide variety of M/G/1-like queues.

This section explains the $M_B/G_{MP}/1$ queue in more detail. The model's main feature is that the information the scheduler knows about a job may change as the job receives service (Section III-A). A job's preemptibility (Section III-B) and holding cost (Section III-E) may also change during its service.

A. Markov-Process Jobs

We model jobs as *absorbing continuous-time strong Markov processes*. The state of a job encodes all information that the scheduler knows about the job. Without loss of generality, we assume all jobs share a common state space $\mathbb{X}$ and follow the same stochastic Markovian dynamics. However, the realization of the dynamics may be different for each job. In particular, the *initial state* of each job is drawn from a distribution X_{new}, so different jobs may start in different states.

While a job is in service, its state stochastically advances according to the Markovian dynamics. This evolution is independent of the arrival process and the evolution of other jobs. A job's state does not change while waiting in the queue.

In addition to the main job state space $\mathbb{X}$, there is one additional *final state*, denoted x_{done}. When a job enters state x_{done}, it completes and exits the system. One can think of a service time S as the stochastic amount of time it takes for a job to go from its initial state, which is drawn from X_{new}, to the final state x_{done}. Because we assume $\mathbf{E}[S] < \infty$, every job eventually reaches x_{done} with probability 1. For ease of notation, we follow the convention that $x_{\mathsf{done}} \notin \mathbb{X}$.

Example III.1. To model known service times, let a job's state be its remaining service time. The state space is $\mathbb{X} = (0, \infty)$, the initial state distribution X_{new} is the service time distribution S, and the final state is $x_{\mathsf{done}} = 0$. During service, a job's state decreases at rate 1.

Example III.2. To model unknown service times, let a job's state be its attained service, meaning the amount of time it has been served so far. The state space is $\mathbb{X} = [0, \infty)$, all jobs start in initial state $X_{\mathsf{new}} = 0$, and the final state x_{done} is an isolated point. During service, a job's state increases at rate 1, but it also has a chance to jump to x_{done}. The jump probability depends on the service time distribution S: the probability a job jumps while being served from state x to state $y > x$ is $\mathbf{P}[S \le y \mid S > x]$.

B. Preemptible and Nonpreemptible States

Every job state is either preemptible or nonpreemptible. The job in service can only be preempted if it is in a preemptible state. We write $\mathbb{X}_{\mathsf{P}}$ for the set of preemptible states and $\mathbb{X}_{\mathsf{NP}} = \mathbb{X} \setminus \mathbb{X}_{\mathsf{P}}$ for the set of nonpreemptible states. Naturally, we assume the scheduler knows which states are preemptible.

We assume all jobs start in a preemptible state, i.e. $X_{\mathsf{new}} \in \mathbb{X}_{\mathsf{P}}$ with probability 1. This means that all jobs in the queue are in preemptible states, and only the job in service can be in a nonpreemptible state.

We assume preemption occurs with no cost or delay. Because a job's state only changes during service, our model is preempt-resume, meaning that preemption does not cause loss of work.

C. Batch Poisson Arrival Process

In the $M_B/G_{MP}/1$, jobs arrive in batches. We represent a batch as a list of states, where the ith state is the initial state of the ith job in the batch. The batch vector has distribution $\mathbf{X}_{\mathsf{batch}} = (X_{\mathsf{batch},1}, \ldots, X_{\mathsf{batch},B})$, where B is the distribution

ISBN 978-3-903176-37-0 © 2021 IFIP

of the number of jobs per batch. The batch arrival times are a Poisson process of rate $\lambda/\mathbf{E}[B]$, with each batch drawn independently from $\mathbf{X}_{\mathsf{batch}}$. The initial state distribution X_{new} is an aggregate distribution determined by picking a random element from a length-biased sample of $\mathbf{X}_{\mathsf{batch}}$.

We allow $\mathbf{X}_{\mathsf{batch}}$ to be an arbitrary distribution over lists of preemptible states. That is, the starting states of the jobs within a batch can be correlated with each other or with the size of a batch. However, after arrival, jobs' states evolve independently of each other (Section III-A).

Our $M_B/G_{MP}/1$ model differs from the traditional M/G/1 with batch Poisson arrivals, often denoted $M^X/G/1$, in an important way. In the $M^X/G/1$, service times within a batch are drawn i.i.d. from S. The $M_B/G_{MP}/1$ is more general in that starting states within a batch can be correlated, so service times within a batch can also be correlated.

D. System State

The state of the system can be described by a list $(x_1, \ldots, x_n)$. Here n is the number of jobs in the system, and $x_i \in \mathbb{X}$ is the state of the ith job. We denote the equilibrium distribution of the system state as $(X_1, \ldots, X_N)$, where N is the equilibrium distribution of the number of jobs.

When discussing the equilibrium distribution of quantities under multiple scheduling policies, we use a superscript π, as in N^π, to refer to the distribution under scheduling policy π.

E. Holding Costs and Objective

We assume that there each job incurs a cost for each unit of time it is not complete. Such a cost is called a *holding cost*, and it applies to every job. A job's holding cost depends on its state, so it may change during service. We denote the holding cost of state $x \in \mathbb{X}$ by $\mathsf{hold}(x)$. Holding costs have dimension COST/TIME. We assume that holding costs are deterministic, positive,[3] and known to the scheduler. For ease of notation, we also define $\mathsf{hold}(x_{\mathsf{done}}) = 0$.

Let $H = \sum_{i=1}^{N} \mathsf{hold}(X_i)$ be the equilibrium distribution of the total holding cost of all jobs in the system. Our objective is to schedule to *minimize mean holding cost* $\mathbf{E}[H]$.

F. What Does the Scheduler Know?

The scheduler also knows, at every moment in time, the current state of all jobs in the system. This assumption is natural because the intuition of our model is that a job's state encodes everything the scheduler knows about the job.

We assume the scheduler knows a description of the job model: the state space $\mathbb{X}$, the subset of preemptible states $\mathbb{X}_{\mathsf{P}} \subseteq \mathbb{X}$, and the Markovian dynamics that govern how a job's state evolves. This assumption is necessary for the Gittins policy, as the policy's definition depends on the job model.

Finally, we assume that the scheduler knows the holding cost $\mathsf{hold}(x)$ of each state $x \in \mathbb{X}$. However, it is possible to transform some problems with unknown holding costs into problems with known holding costs. A notable example is minimizing mean slowdown when service times are unknown to the scheduler (Example V.2). After transforming such problems into known-holding-cost form, one can apply our results.

[3]The holding cost of nonpreemptible states does not impact minimizing mean holding cost (Lemma VII.2), so one could have $\mathsf{hold}(x) \leq 0$ for $x \in \mathbb{X}_{\mathsf{NP}}$.

G. Technical Foundations

We have thus far avoided discussing technical measurability conditions that the job model must satisfy. For example, if the job Markov process has uncountable state space $\mathbb{X}$, one should make some topological assumptions on $\mathbb{X}$ and $\mathbb{X}_{\mathsf{P}}$, as well as some continuity assumptions on holding costs. As another example, when discussing subsets $\mathbb{Y} \subseteq \mathbb{X}_{\mathsf{P}}$ (Definitions VI.1 and IV.2), one should restrict attention to measurable subsets. See Scully et al. [18, Appendix D] for additional discussion.

We consider these technicalities outside the scope of this paper. All of our results are predicated on being able to apply basic optimal stopping theory to solve the Gittins game (Section VI). Optimal stopping of general Markov processes is a broad field, and the theory has been developed under many different types of assumptions [19]. Our main result (Theorem V.1) can be understood as proving Gittins's optimality in any setting where optimal stopping theory of the Gittins game has been developed.

IV. The Gittins Policy

We now define the Gittins policy, the scheduling policy that minimizes mean holding cost in the $M_B/G_{MP}/1$ (Section III).

Before defining Gittins, we discuss its intuitive motivation. Suppose we are scheduling with the goal of minimizing mean holding cost. How do we decide which job to serve? Because our objective is minimizing mean holding cost, our aim should be to quickly lower the holding cost of jobs in the system. We can lower a job's holding cost by completing it, in which case its holding cost becomes $\mathsf{hold}(x_{\mathsf{done}}) = 0$, or by serving it until it reaches a state with lower holding cost.

The basic idea of Gittins is to always serve the job whose holding cost we can decrease the fastest. To formalize this description, we need to define what it means for a job's holding cost to decrease at a certain rate.

A. Gittins Index

As a warm-up, consider the setting of Example III.1: the scheduler knows every job's service time, and a job's state is its remaining service time. Suppose that every state is preemptible.

How quickly can we decrease the holding cost of a job in state x, meaning x remaining service time? Serving a job from state x to state y takes $x - y$ time and decreases the job's holding cost by $\mathsf{hold}(x) - \mathsf{hold}(y)$, so the holding cost decreases at rate $(\mathsf{hold}(x) - \mathsf{hold}(y))/(x - y)$. To find the fastest possible decrease, we optimize over y:

$$\begin{pmatrix}\text{maximum holding cost}\\ \text{decrease rate from } x\end{pmatrix} = \sup_{y \in [0,x)} \frac{\mathsf{hold}(x) - \mathsf{hold}(y)}{x - y}.$$

The above quantity is called the *(Gittins) index* of state x. A state's index is the maximum rate at which we can decrease its holding cost by serving it for some amount of time.

To generalize the above discussion to general job models, we need to make two changes. Firstly, because a job's state

ISBN 978-3-903176-37-0 © 2021 IFIP

dynamics can be stochastic, we need to consider serving it until it enters a *set* of states $\mathbb{Y}$. Secondly, because we cannot stop serving a job while it is nonpreemptible, we require $\mathbb{Y} \subseteq \mathbb{X}_{\mathsf{P}}$.

Definition IV.1. For all $x \in \mathbb{X}$ and $\mathbb{Y} \subseteq \mathbb{X}_{\mathsf{P}}$, let

$$\mathsf{Serve}(x, \mathbb{Y}) = \begin{pmatrix}\text{service needed for a job starting in} \\ \text{state } x \text{ to first enter } \mathbb{Y} \cup \{x_{\mathsf{done}}\}\end{pmatrix},$$
$$\mathsf{serve}(x, \mathbb{Y}) = \mathbf{E}[\mathsf{Serve}(x, \mathbb{Y})],$$
$$\mathsf{Hold}(x, \mathbb{Y}) = \begin{pmatrix}\text{holding cost of a job starting in state } x \\ \text{when it first enters } \mathbb{Y} \cup \{x_{\mathsf{done}}\}\end{pmatrix},$$
$$\mathsf{hold}(x, \mathbb{Y}) = \mathbf{E}[\mathsf{Hold}(x, \mathbb{Y})].$$

To clarify, $\mathsf{Serve}(x, \mathbb{Y})$ and $\mathsf{Hold}(x, \mathbb{Y})$ are distributions. If $x \in \mathbb{Y}$, then $\mathsf{Serve}(x, \mathbb{Y}) = 0$ and $\mathsf{Hold}(x, \mathbb{Y}) = \mathsf{hold}(x)$.

If we serve a job from state x until it enters $\mathbb{Y}$, its holding cost decreases at rate $(\mathsf{hold}(x) - \mathsf{hold}(x, \mathbb{Y}))/\,\mathsf{serve}(x, \mathbb{Y})$ on average. We obtain a state's Gittins index by optimizing over $\mathbb{Y}$.

Definition IV.2. The *(Gittins) index* of state $x \in \mathbb{X}$ is

$$\mathsf{index}(x) = \sup_{\mathbb{Y} \subseteq \mathbb{X}_{\mathsf{P}}} \frac{\mathsf{hold}(x) - \mathsf{hold}(x, \mathbb{Y})}{\mathsf{serve}(x, \mathbb{Y})}.$$

When we say that a job has a certain index, we mean that the job's current state has that index.

Given the definition of the Gittins index, the Gittins policy boils down to one rule: at every moment in time, unless the job in service is nonpreemptible, serve the job of *maximal Gittins index*, breaking ties arbitrarily.

Because the Gittins index depends on the job model, it might be more accurate to view Gittins not as one specific policy but rather as a family of policies, with one instance for every job model. When we refer to "the" Gittins policy, we mean the Gittins policy for the current system's job model.

B. Gittins Rank

Some work on the Gittins policy refers to the *(Gittins) rank* of a state [6, 11, 18, 20], which is the reciprocal of its index:

$$\mathsf{rank}(x) = \frac{1}{\mathsf{index}(x)}.$$

Gittins thus always serves the job of minimal rank.

The Gittins rank sometimes has a more intuitive interpretation than the Gittins index. For instance, when jobs have known service times and constant holding cost 1, Gittins reduces to SRPT, and a job's rank is its remaining service time.

We use both the index and rank conventions in this work. This section mostly uses the index convention. Sections VI and VII, which prove Gittins's optimality, use the rank convention because it better matches the authors' intuitions, though this choice is certainly subjective.

V. Scope of Gittins's Optimality

Our main result is that Gittins is optimal in the $\mathrm{M_B/G_{MP}/1}$ with arbitrary state-based holding costs. Specifically, Gittins is optimal among *nonclairvoyant* scheduling policies, which are policies that make scheduling decisions based only on the current and past system states.

Theorem V.1. *The Gittins policy minimizes mean holding cost in the $M_B/G_{MP}/1$. That is, for all nonclairvoyant policies π,*

$$\mathbf{E}[H^{\mathrm{Gittins}}] \leq \mathbf{E}[H^{\pi}].$$

All of the prior optimality results discussed in Section II are special cases of Theorem V.1. This makes Theorem V.1 a *unifying theorem for Gittins's optimality* in M/G/1-like systems. Theorem V.1 also holds in scenarios not covered by any prior result. For instance, no prior result handles batch arrivals or holding costs that change during service.

A. Mean Slowdown and Unknown Holding Costs

Recall from Section III-E that we assume that the holding cost of every job state is known to the scheduler. However, some scheduling problems involve unknown holding costs. An important example is minimizing mean slowdown, in which a job's holding cost is the reciprocal of its service time. Unless all service times are known to the scheduler, this involves unknown holding costs.

Fortunately, we can transform many problems with unknown holding costs into problems with known holding costs. Suppose a job's current unknown holding cost depends only on its current and future states. Then for all job states $x \in \mathbb{X}$, let

$$\mathsf{hold}(x) = \mathbf{E}\left[\begin{matrix}\text{unknown holding cost} \\ \text{of a job in state } x\end{matrix} \;\middle|\; \begin{matrix}\text{job reached} \\ \text{state } x\end{matrix}\right], \quad \text{(V.1)}$$

where the expectation is taken over a random realization of a job's path through the state space. The mean holding cost of nonclairvoyant policies is unaffected by this transformation.

Example V.2 (Gittins for mean slowdown)**.** Consider the system from Example III.2. It has unknown service times, and a job's state x is its attained service. Suppose all states are preemptible. To minimize mean slowdown, we give a job with service time s holding cost s^{-1}. This turns (V.1) into $\mathsf{hold}(x) = \mathbf{E}[S^{-1} \mid S > x]$, and the Gittins index becomes

$$\mathsf{index}(x) = \sup_{y > x} \frac{\mathbf{E}[S^{-1}\mathbb{1}(S \leq y) \mid S > x]}{\mathbf{E}[\min\{S, y\} - x \mid S > x]}.$$

VI. The Gittins Game

In this section we introduce the *Gittins game*, which is an optimization problem concerning a single job. The Gittins game serves two purposes. Firstly, it gives an alternative intuition for the Gittins rank. Secondly, its properties are important for proving Gittins's optimality. We define the Gittins game (Section VI-A), study its properties, (Sections VI-B–VI-D), and explain its relationship to the Gittins rank (Section VI-E).

A. Defining the Gittins Game

The Gittins game is an optimal stopping problem concerning a single job. We are given a job in some starting state $x \in \mathbb{X}$ and a *penalty parameter* $r \geq 0$, which has dimension $\mathrm{TIME}^2/\mathrm{COST}$.

The goal of the Gittins game is to end the game as soon as possible. The game proceeds as follows. We begin by

ISBN 978-3-903176-37-0 © 2021 IFIP

serving the job. The job's state evolves as usual during service (Section III-A). If the job completes, namely by reaching state x_{done}, the game ends immediately. Whenever the job's state is preemptible, we may *give up*. If we do so, we stop serving the job, and the game ends after deterministic delay $r\,\mathsf{hold}(y)$, where $y \in \mathbb{X}_{\mathsf{P}}$ is the job's state when we give up.

We assume the job's current state is always visible. Playing the Gittins game thus boils down to deciding whether or not to give up based on the job's current state.

Because the job's state evolution is Markovian, the Gittins game is a Markovian optimal stopping problem. This means there is an optimal policy of the following form: for some *give-up set* $\mathbb{Y} \subseteq \mathbb{X}_{\mathsf{P}}$, give up when the job's state first enters $\mathbb{Y}$. The strong Markov property implies that this set $\mathbb{Y}$ need not depend on the starting state, though it may depend on the penalty parameter. We use this observation and Definition IV.1 to formally define the Gittins game.

Definition VI.1. The *Gittins game* is the following optimization problem. The parameters are a starting state $x \in \mathbb{X}$ and penalty parameter r, and the control is a give-up set $\mathbb{Y} \subseteq \mathbb{X}_{\mathsf{P}}$. The *cost of give-up set* $\mathbb{Y}$ is

$$\mathsf{game}(x, r, \mathbb{Y}) = \mathsf{serve}(x, \mathbb{Y}) + r\,\mathsf{hold}(x, \mathbb{Y}).$$

The objective is to choose $\mathbb{Y}$ to minimize $\mathsf{game}(x, r, \mathbb{Y})$. The *optimal cost* or *cost-to-go function* of the Gittins game is

$$\mathsf{game}(x, r) = \inf_{\mathbb{Y} \subseteq \mathbb{X}_{\mathsf{P}}} \mathsf{game}(x, r, \mathbb{Y}). \tag{VI.1}$$

B. Shape of the Cost-To-Go Function

To gain some intuition for the Gittins game, we begin by proving some properties of the cost-to-go function, focusing on its behavior as the penalty parameter varies.

Lemma VI.2. *For all $x \in \mathbb{X}$ and $r \geq 0$, the cost-to-go function $\mathsf{game}(x, r)$ is (i) nondecreasing in r, (ii) concave in r, (iii) bounded by $\mathsf{game}(x, r) \leq \mathsf{serve}(x, \mathbb{X}_{\mathsf{P}}) + r\,\mathsf{hold}(x, \mathbb{X}_{\mathsf{P}})$, (iv) bounded by $\mathsf{game}(x, r) \leq \mathsf{serve}(x, \varnothing)$. When $x \in \mathbb{X}_{\mathsf{P}}$, property (iii) becomes $\mathsf{game}(x, r) \leq r\,\mathsf{hold}(x)$.*

Proof. Properties (i) and (ii) follow from (VI.1), which expresses $\mathsf{game}(x, r)$ as an infimum of nondecreasing concave functions of r. Properties (iii) and (iv) follow from the fact that two possible give-up sets are $\mathbb{X}_{\mathsf{P}}$, meaning giving up as soon as possible, and $\varnothing$, meaning never giving up. The simplification when $x \in \mathbb{X}_{\mathsf{P}}$ is due to Definition IV.1. □

C. Optimal Give-Up Set

We now characterize one possible solution to the Gittins game. Because the Gittins game is a Markovian optimal stopping problem, we never need to look back at past states when deciding when to give up. This means we can find an optimal give-up set that depends only on the penalty parameter r. We ask for each preemptible state: is it optimal to give up immediately if we start in this state? The set of states for which we answer yes is an optimal give-up set.

Definition VI.3. The *optimal give-up set* for the Gittins game with penalty parameter r is

$$\mathbb{Y}^*(r) = \{x \in \mathbb{X}_{\mathsf{P}} \mid \mathsf{game}(x, r) = r\,\mathsf{hold}(x)\}.$$

Noe that $\mathbb{Y}^*(0) = \mathbb{X}_{\mathsf{P}}$. We also let $\mathbb{Y}^*(\infty) = \varnothing$. For simplicity of language, we call $\mathbb{Y}^*(r)$ "the" optimal give-up set, even though there may be other optimal give-up sets.

Basic results in optimal stopping theory [19] imply that $\mathsf{game}(x, r) = \mathsf{game}(x, r, \mathbb{Y}^*(r))$, so the infimum in (VI.1) is always attained, namely by $\mathbb{Y}^*(r)$.

The sets $\mathbb{Y}^*(r)$ are monotonic in r, i.e. $\mathbb{Y}^*(r) \supseteq \mathbb{Y}^*(r')$ for all $r \leq r'$. This is because increasing the penalty makes giving up less attractive, so giving up is optimal in fewer states.

For most of the rest of this paper, when we discuss the Gittins game, we consider strategies that use optimal give-up sets, so we simplify the notation for that case.

Definition VI.4. For all $x \in \mathbb{X}$ and $r \geq 0$, let

$$\mathsf{Serve}(x, r) = \mathsf{Serve}(x, \mathbb{Y}^*(r))$$

and similarly for $\mathsf{serve}(x, r)$, $\mathsf{Hold}(x, r)$, and $\mathsf{hold}(x, r)$.

D. Derivative of the Cost-To-Go Function

Suppose we solve the Gittins game for penalty parameter r, then change the penalty parameter to $r \pm \varepsilon$ for some small $\varepsilon > 0$. One would expect that the give-up set $\mathbb{Y}^*(r)$ is nearly optimal for the new penalty parameter $r \pm \varepsilon$, which would imply $\mathsf{game}(x, r \pm \varepsilon) \approx \mathsf{serve}(x, r) + (r \pm \varepsilon)\,\mathsf{hold}(x, r)$. One can use Lemma VI.2 and a classic envelope theorem [21, Theorem 1] to formalize this argument. For brevity, we omit the proof. See Scully et al. [18, Lemma 5.3] for a similar proof.

Lemma VI.5. *For all $x \in \mathbb{X}_{\mathsf{P}}$, the function $r \mapsto \mathsf{game}(x, r)$ is differentiable almost everywhere with derivative*

$$\frac{\mathrm{d}}{\mathrm{d}r}\,\mathsf{game}(x, r) = \mathsf{hold}(x, r).$$

E. Relationship to the Gittins Rank

The Gittins game and the optimal give-up set are closely related to the Gittins rank. In fact, we can use the Gittins game to give an alternative definition of a state's rank. For brevity, we simply state the connection below.

Lemma VI.6.

(i) For all $r \geq 0$, we can write the optimal give-up set as $\mathbb{Y}^*(r) = \{x \in \mathbb{X}_{\mathsf{P}} \mid \mathsf{rank}(x) \geq r\}$.

(ii) For all $x \in \mathbb{X}_{\mathsf{P}}$, we can write the Gittins rank of x as $\mathsf{rank}(x) = \max\{r \geq 0 \mid x \in \mathbb{Y}^*(r)\}$.

VII. Proving Gittins's Optimality

We now prove Theorem V.1, namely that Gittins minimizes mean holding cost in the $\mathrm{M_B/G_{MP}/1}$. Our proof has four steps. We begin by showing that minimizing mean holding cost $\mathbf{E}[H]$ is equivalent to minimizing the mean *preemptible* holding cost $\mathbf{E}[H_{\mathsf{P}}]$, which only counts the holding costs of jobs in preemptible states (Section VII-A). We define a new quantity called *r-work*, the amount of work in the system "below rank r"

ISBN 978-3-903176-37-0 © 2021 IFIP

(Section VII-B). We show how to relate an integral of r-work to the preemptible holding cost H_{P}, (Section VII-C) with more r-work implying higher holding cost. We show that Gittins minimizes mean r-work for all $r \geq 0$, so it also minimizes $\mathbf{E}[H]$ (Section VII-D).

A. Preemptible and Nonpreemptible Holding Costs

Definition VII.1. The system's *preemptible holding cost* is the total holding cost of all jobs in the system whose states are preemptible. It has equilibrium distribution $H_{\mathsf{P}} = \sum_{i=1}^{N} \mathbb{1}(X_i \in \mathbb{X}_{\mathsf{P}})\, \mathsf{hold}(X_i)$,, where $\mathbb{1}$ is the indicator function. The *nonpreemptible holding cost* is defined analogously as $H_{\mathsf{NP}} = \sum_{i=1}^{N} \mathbb{1}(X_i \in \mathbb{X}_{\mathsf{NP}})\, \mathsf{hold}(X_i)$.

Our goal is to show that Gittins minimizes mean holding cost $\mathbf{E}[H] = \mathbf{E}[H_{\mathsf{P}}] + \mathbf{E}[H_{\mathsf{NP}}]$. The lemma below shows that $\mathbf{E}[H_{\mathsf{NP}}]$ is unaffected by the scheduling policy. Minimizing $\mathbf{E}[H]$ thus amounts to minimizing $\mathbf{E}[H_{\mathsf{P}}]$.

Lemma VII.2. *In the $M_B/G_{MP}/1$, the mean nonpreemptible holding cost has the same value under all scheduling policies:*

$$\mathbf{E}[H_{\mathsf{NP}}] = \lambda \mathbf{E}\begin{bmatrix}\text{total cost a job accrues while in a}\\ \text{nonpreemptible state during service}\end{bmatrix}.$$

Proof. By a generalization of Little's law [1],

$$\mathbf{E}[H_{\mathsf{NP}}] = \lambda \mathbf{E}\begin{bmatrix}\text{total cost a job accrues while}\\ \text{in a nonpreemptible state}\end{bmatrix}.$$

The desired statement follows from the fact that if a job's state is nonpreemptible state, it must be in service (Section III-B). □

B. Defining r-Work

Definition VII.3. The *(job) r-work of state x* is $\mathsf{Serve}(x, r)$, namely the amount of service it requires to either complete or enter a preemptible state of rank at least r.[4] The *(system) r-work* is the total r-work of all jobs in the system. Its equilibrium distribution, denoted $W(r)$, is

$$W(r) = \sum_{i=1}^{N} \mathsf{Serve}(X_i, r),$$

where $(X_1, \ldots, X_N)$ is the equilibrium system state (Section III-D). In particular, we can think of $W(0)$ as the amount of nonpreemptible work in the system.

Lemma VII.4. *For all $r \geq 0$,*

$$\mathbf{E}[W(r)] = \mathbf{E}\left[\sum_{i=1}^{N} \mathsf{serve}(X_i, r)\right].$$

Proof. This follows from the law of total expectation and the fact that $\mathbf{E}[\mathsf{Serve}(X_i, r) \mid X_i] = \mathsf{serve}(X_i, r)$. □

[4]Strictly speaking, Definitions IV.1 and VI.4 introduce $\mathsf{Serve}(x, r)$ as a distribution, so the r-work of a job in state x is not $\mathsf{Serve}(x, r)$ itself but rather a random variable with distribution $\mathsf{Serve}(x, r)$.

C. Relating r-Work to Holding Cost

Theorem VII.5. *In the $M_B/G_{MP}/1$, under all nonclairvoyant policies,*

$$\mathbf{E}[H_{\mathsf{P}}] = \int_0^\infty \frac{\mathbf{E}[W(r)] - \mathbf{E}[W(0)]}{r^2}\, \mathrm{d}r.$$

Proof. By Lemma VII.4 and the definition of H_{P} it suffices to show that for all $x \in \mathbb{X}_{\mathsf{P}}$,

$$\mathsf{hold}(x) = \int_0^\infty \frac{\mathsf{serve}(x, r) - \mathsf{serve}(x, 0)}{r^2}\, \mathrm{d}r. \qquad \text{(VII.1)}$$

Because $x \in \mathbb{X}_{\mathsf{P}}$, it is optimal to give up in state x when playing the Gittins game with penalty parameter 0, so

$$\mathsf{serve}(x, 0) = 0, \qquad \mathsf{hold}(x, 0) = \mathsf{hold}(x).$$

Using Lemma VI.5, we compute

$$\frac{\mathrm{d}}{\mathrm{d}r}\frac{\mathsf{game}(x, r)}{r} = \frac{r\,\mathsf{hold}(x, r) - \mathsf{game}(x, r)}{r^2} = \frac{-\mathsf{serve}(x, r)}{r^2}.$$

This means the integral in (VII.1) becomes a difference between two limits. Using Lemmas VI.2 and VI.5, we compute

$$\begin{aligned}\int_0^\infty \frac{\mathsf{serve}(x, r)}{r^2}\, \mathrm{d}r &= \lim_{r\to 0}\frac{\mathsf{game}(x, r)}{r} - \lim_{r\to\infty}\frac{\mathsf{game}(x, r)}{r}\\ &= \mathsf{hold}(x, 0) - 0 = \mathsf{hold}(x).\end{aligned}$$ □

Theorem VII.5 implies that to minimize $\mathbf{E}[H_{\mathsf{P}}]$, it suffices to minimize $\mathbf{E}[W(r)] - \mathbf{E}[W(0)]$ for all $r \geq 0$. It turns out that $\mathbf{E}[W(0)]$, much like $\mathbf{E}[H_{\mathsf{NP}}]$, is unaffected by the scheduling policy, so it suffices to minimize mean r-work $\mathbf{E}[W(r)]$. We omit the proof, as it is very similar to that of Lemma VII.2.

Lemma VII.6. *In the $M_B/G_{MP}/1$, the mean 0-work $\mathbf{E}[W(0)]$ has the same value under all scheduling policies.*

D. Gittins Minimizes Mean r-Work

Lemmas VII.2 and VII.6 and Theorem VII.5, together imply that if a scheduling policy minimizes mean r-work $\mathbf{E}[W(r)]$ for all $r \geq 0$, then it minimizes mean holding cost $\mathbf{E}[H]$. We show that Gittins does exactly this, implying Gittins's optimality.

Theorem VII.7. *The Gittins policy minimizes mean r-work in the $M_B/G_{MP}/1$. That is, for all scheduling policies π and $r \geq 0$,*

$$\mathbf{E}[W^{\mathrm{Gittins}}(r)] \leq \mathbf{E}[W^{\pi}(r)].$$

Before proving Theorem VII.7, we introduce the main ideas behind the proof. For the rest of this section, fix arbitrary $r \geq 0$. We classify jobs in the system into two types.

- A job is *r-good* if it is nonpreemptible or has Gittins rank less than r, i.e. its state is in $\mathbb{X} \setminus \mathbb{Y}^*(r)$.
- A job is *r-bad* jobs if it has Gittins rank at least r, i.e. its state is in $\mathbb{Y}^*(r)$.

During service, a job may alternate between being r-good and r-bad. Gittins minimizes r-work because the jobs that contribute to r-work are exactly the r-good jobs, and Gittins always prioritizes r-good jobs over r-bad jobs. This means that whenever the amount of r-work in the system is positive, Gittins decreases it at rate 1, which is as quickly as possible.

ISBN 978-3-903176-37-0 © 2021 IFIP

Given that Gittins decreases r-work as quickly as possible, does Theorem VII.7 immediately follow? The answer is no: we need to look not just at how r-work decreases but also at how it increases. Two types of events increase r-work.

- Arrivals can add r-work to the system.
- During service, a job can transition from being r-bad to being r-good as its state evolves. Using the terminology of Scully et al. [6, 18], we say call this *r-recycling* the job. Every r-recycling adds r-work to the system.

Arrivals are outside of the scheduling policy's control, but r-recyclings occur at different times under different scheduling policies. Because Gittins prioritizes r-good jobs over r-bad jobs, all r-recyclings occur when there is zero r-work. It turns out that because the batch arrival process is Poisson, this r-recycling timing minimizes mean r-work.

Proof of Theorem VII.7. We are comparing Gittins to an arbitrary scheduling policy π. It is convenient to allow π to be more powerful than an ordinary policy: we allow π to devote infinite processing power to r-bad jobs. This has two implications:

- Whenever there is r-work in the system, π controls at what rate it decreases, where 1 is the maximum rate.
- Regardless of the rate at which r-work is decreasing, whenever there is an r-bad job in the system, π controls at what moment in time it either completes or is r-recycled.

A straightforward interchange argument shows that it suffices to only compare against policies π which are "r-work-conserving", meaning they decrease r-work at rate 1 whenever r-work is nonzero. Gittins is also r-work-conserving.

It remains only to show that among r-work-conserving policies, mean r-work is minimized by only r-recycling jobs when r-work is zero. This follows from classic decomposition results for the M/G/1 with generalized vacations [22]. We first explain how to view the r-work in the $M_B/G_{MP}/1$ as the virtual work in a vacation system.[5]

- Interpret a batch adding s r-work to the $M_B/G_{MP}/1$ as an arrival of service time s in the vacation system.
- Interpret an r-recycling adding v r-work to the $M_B/G_{MP}/1$ as a vacation of length v in the vacation system.

Using the above interpretation, a vacation system result of Miyazawa [22, Theorem 3.3] implies

$$\mathbf{E}[W^\pi(r)] = c_1 + c_2 \mathbf{E}\begin{bmatrix} r\text{-work sampled immediately} \\ \text{before } \pi \text{ } r\text{-recycles a job} \end{bmatrix},$$

where c_1 and c_2 are constants that depend on the system parameters but not on the scheduling policy π. Because Gittins prioritizes r-good jobs over r-bad jobs, Gittins only r-recycles when r-work is zero. This means the expectation on the right-hand side is zero under Gittins. But the expectation is nonnegative in general, so Gittins minimizes mean r-work. □

VIII. Conclusion

We have given the first fully general statement (Theorem V.1) and proof of Gittins's optimality in the M/G/1. This simultaneously improves upon, unifies, and generalizes prior proofs, all which either apply only in special cases or require limiting technical assumptions (Section II).

We believe Gittins's optimality holds even more generally than we have shown. For example, our proof likely generalizes to settings with "branching" jobs or additional priority constraints on the scheduler [23, Section 4.7]. It is also sometimes possible to strengthen the sense in which Gittins is optimal. For example, SRPT is optimal for non-Poisson arrival times, and Gittins sometimes stochastically minimizes holding cost in addition to minimizing the mean.

[5] *Virtual work* in a vacation system is total remaining service time of all jobs in the system plus, if a vacation is in progress, remaining vacation time.

References

[1] S. L. Brumelle, "On the relation between customer and time averages in queues," *J. Appl. Probab.*, vol. 8, no. 3, pp. 508–520, 1971.

[2] J. C. Gittins, *Multi-Armed Bandit Allocation Indices*, 1st ed., ser. Wiley-Interscience Series in Systems and Optimization. Chichester, UK: Wiley, 1989.

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

[4] ——, "Properties of the Gittins index with application to optimal scheduling," *Prob. Eng. Inf. Sci.*, vol. 25, no. 3, pp. 269–288, Jul. 2011.

[5] E. Hyytiä, S. Aalto, and A. Penttinen, "Minimizing slowdown in heterogeneous size-aware dispatching systems," *SIGMETRICS Perform. Eval. Rev.*, vol. 40, no. 1, pp. 29–40, Jun. 2012.

[6] 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, Apr. 2018.

[7] Z. Scully, L. van Kreveld, O. J. Boxma, J.-P. Dorsman, and A. Wierman, "Characterizing policies with optimal response time tails under heavy-tailed job sizes," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 4, no. 2, Jun. 2020.

[8] J. C. Gittins, "Bandit processes and dynamic allocation indices," *J. R. Statist. Soc. B*, vol. 41, no. 2, pp. 148–164, Jan. 1979.

[9] L. E. Schrage, "A proof of the optimality of the shortest remaining processing time discipline," *Oper. Res.*, vol. 16, no. 3, pp. 687–690, Jun. 1968.

[10] D. W. Fife, "Scheduling with random arrivals and linear loss functions," *Manag. Sci.*, vol. 11, no. 3, pp. 429–437, Jan. 1965.

[11] K. C. Sevcik, "The use of service time distributions in scheduling," Ph.D. dissertation, University of Chicago, Chicago, IL, Aug. 1971.

[12] G. von Olivier, "Kostenminimale prioritäten in wartesystemen vom typ M/G/1 [Cost-minimum priorities in queueing systems of type M/G/1]," *Elektron. Rechenanl.*, vol. 14, no. 6, pp. 262–271, Dec. 1972.

[13] G. P. Klimov, "Time-sharing service systems. I," *Theory Probab. Appl.*, vol. 19, no. 3, pp. 532–551, 1974.

[14] T. L. Lai and Z. Ying, "Open bandit processes and optimal scheduling of queueing networks," *Adv. Appl. Probab.*, vol. 20, no. 2, pp. 447–472, 1988.

[15] D. Bertsimas, "The achievable region method in the optimal control of queueing systems; formulations, bounds and policies," *Queueing Syst.*, vol. 21, no. 3, pp. 337–389, Sep. 1995.

[16] M. Dacre, K. D. Glazebrook, and J. Niño-Mora, "The achievable region approach to the optimal control of stochastic systems," *J. R. Statist. Soc. B*, vol. 61, no. 4, pp. 747–791, 1999.

[17] P. Whittle, "Tax problems in the undiscounted case," *J. Appl. Probab.*, vol. 42, no. 3, pp. 754–765, Sep. 2005.

[18] Z. Scully, I. Grosof, and M. Harchol-Balter, "The Gittins policy is nearly optimal in the M/G/k under extremely general conditions," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 4, no. 3, Nov. 2020.

[19] G. Peskir and A. N. Shiryaev, *Optimal Stopping and Free-Boundary Problems*, ser. Lectures in Mathematics. ETH Zürich. Basel: Birkhäuser Verlag, 2006.

[20] K. C. Sevcik, "Scheduling for minimum total loss using service time distributions," *J. ACM*, vol. 21, no. 1, pp. 66–75, Jan. 1974.

[21] P. Milgrom and I. Segal, "Envelope theorems for arbitrary choice sets," *Econometrica*, vol. 70, no. 2, pp. 583–601, Mar. 2002.

[22] M. Miyazawa, "Decomposition formulas for single server queues with vacations : A unified approach by the rate conservation law," *Commun. Statist.—Stochastic Models*, vol. 10, no. 2, pp. 389–413, Jan. 1994.

[23] J. C. Gittins, K. D. Glazebrook, and R. Weber, *Multi-Armed Bandit Allocation Indices*, 2nd ed. Chichester, UK: Wiley, 2011.

ISBN 978-3-903176-37-9 © 2021 IFIP