The Gittins Policy in the M/G/1 Queue
(Extended and Revised Version)

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 “cµ rule”, which minimizes class-weighted mean response time in the multiclass M/M/I, is a special case of Gittins.

However, despite the extensive literature on Gittins on 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, 2]. 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 “cµ rule”, which minimizes weighted mean response time in the multiclass M/M/I 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 [3, 4], 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.

*Supported by NSF grant nos. CMMI-1938909 and CSR-1763701 and a Google Faculty Award.
†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.
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. These limitations are significant because they put some widely-believed results on uncertain theoretical foundations.

A final limitation that applies to all prior proofs is that although different jobs may have different holding costs, each job’s holding cost is constant. To the best of our knowledge, ours is the first presentation of Gittins and proof of its optimality that allows jobs’ holding costs to change during service.

We now present prior work on the Gittins policy in rough chronological order, giving each decade a theme. The decades should be understood loosely, as the themes do not fit perfectly into decades. Throughout, when we refer to “M/G/1 scheduling”, we mean the problem of minimizing mean holding cost in an $M/G/1$ queue or similar model.

A. 1960s: Known Service Times

The earliest results in M/G/1 scheduling all featured known service times. The most famous of these results is the proof that SRPT minimizes mean response time [12], but researchers also made progress in systems with variable holding costs and nonpreemptive service.

Prior Proof 1. Schrage [12].

Model: Preemptive single-server queue, known service times.

Holding costs: Same for all jobs.

Limitations: (ii-a).

Prior Proof 2. Fife [13, 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 [14, 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 [14, Conjecture 4-1] argues informally that an index policy should be optimal.

Prior Proof 4. Sevcik [14, Theorem 4-2].

Model: Preemptive M/G/1, unknown service times.

Holding costs: Based on class.

Limitations: (i-b), (i-c), and (iii). Sevcik [14, Conjecture 4-3] argues informally that an index policy should be optimal.

Prior Proof 5. Von Olivier [15].

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 [15] 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.

A partially preemptive M/G/1 problem was solved by Klimov [16], who 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. This model is partially preemptive in that a job returned to the system via feedback need not be served immediately. Another way of viewing systems with feedback is that each job is a discrete semi-Markov chain where each job class is a state. Klimov’s model is thus notable in that in addition to having unknown service times, jobs take stochastic paths through a state space.

Prior Proof 6. Klimov [16].

Model: Nonpreemptive M/G/1+fbk, unknown service times.

Holding costs: Based on class.

2The word “state” here differs from our job model’s terminology (Section III-A). In our terminology, each state in a semi-Markov chain corresponds to a connected set of states in a piecewise-deterministic Markov process.
which involves taking the vanishing-discount limit. Work by Klimov [16] on the nonpreemptive M/G/1+fbk was first developed. See Gittins [11] for a survey of early developments and Gittins et al. [4] for a modern overview.

At first glance, the MAB problem seems very different from M/G/1 scheduling.

- The MAB problem involves maximizing exponentially discounted reward, whereas M/G/1 scheduling typically involve minimizing long-run average costs.
- The MAB problem involves a fixed set of alternatives, whereas M/G/1 scheduling features a dynamically changing set of jobs.

Nevertheless, as early as 1973, Nash [17] showed that a version of the MAB problem becomes M/G/1 scheduling in the limit as the discount rate vanishes. In the 1980s, several researchers further pursued these ideas. Lai and Ying [18] reexamined work by Klimov [16] on the nonpreemptive M/G/1+fbk, connected it to the MAB problem, and extended it to the preemptive M/M/1+fbk. Gittins [3] extended work by Nash [17] to continuous time.

**Prior Proof 7.** Lai and Ying [18].
*Model:* Preemptive M/M/1+fbk, unknown service times.
*Holding costs:* Based on class.
*Limitations:* (i-c) and (ii-c).

**Prior Proof 8.** Gittins [3, 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 [3] is often cited in the literature as proving the Gittins policy’s optimality in the M/G/1 [5–10]. 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\(^3\) [3, 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. This side condition comes from translating a MAB result to the M/G/1, which involves taking the vanishing-discount limit.

The second step uses a limit argument to allow unrestricted preemption [3, 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. These limitations could be relaxed, but only by checking the first step’s side condition for every system considered in the limit argument.

\(^3\)In this setting, a job’s state is the pair of its class and attained service.

**D. 1990s: Achievable Region Approach**

The achievable region approach was a new way of thinking about a wide variety of stochastic control problems, including M/G/1 scheduling and the MAB problem. Bertsimas [19] and Dacre et al. [20] give surveys of the area. While the achievable region method introduced important new ideas, it did not extend the known scope of Gittins’s optimality in M/G/1 scheduling.

**Prior Proof 9.** Achievable region approaches [19, 20].
*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).

**E. 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 [5, 6] and analyzed its performance [7–10, 21]. A performance analysis by Whittle [21] based on dynamic programming also resulted in an optimality proof, but it did not expand the known scope of Gittins’s optimality.

**Prior Proof 10.** Whittle [21].
*Model:* Preemptive M/M/1+fbk, unknown service times.
*Holding costs:* Based on class.
*Limitations:* (i-c) and (ii-b).

**F. 2020: Modeling Jobs as General Markov Processes**

In 2020, Scully et al. [22] studied minimizing mean response time in the preemptive M/G/k, showing that Gittins is near-optimal in a certain sense. As a byproduct of their analysis of Gittins in the M/G/k, they gave a new proof of Gittins’s optimality in the M/G/1. Their technique overcomes many limitations of prior proofs, particularly limitation (i), but it applies only to the metric of mean response time.

**Prior Proof 11.** Scully et al. [22, Theorem 7.3].
*Model:* Preemptive M/G\(_M\)/1, i.e. the preemptive M\(_B\)/G\(_M\)/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. [22] do not include are various holding cost metrics, nonpreemptible or partially preemptible jobs, and batch arrivals.

**III. SYSTEM MODEL: THE M\(_B\)/G\(_M\)/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 \(\lambda\), the service time distribution is \(S\), and the load is \(\rho = \lambda E[S]\). We assume \(\rho < 1\) for stability.

We call our model the \(M_B/G_M/1\) queue. The “\(M_B\)” indicates that jobs arrive in batches with Poisson arrival times. The “\(G_M\)” 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 \( X \) 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 \( X \), 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 \( E[S] < \infty \), every job eventually reaches \( x_{done} \) with probability 1. For ease of notation, we follow the convention that \( x_{done} \notin X \).

**Example III.1.** To model known service times, let a job’s state be its remaining service time. The state space is \( X = (0, \infty) \), the initial state distribution \( X_{new} \) is the service time distribution \( S \), and the final state is \( x_{done} = 0 \). 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 \( X = [0, \infty) \), all jobs start in initial state \( X_{new} = 0 \), 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

\[
P[S \leq y \mid S > x].
\]

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 \( X_P \) for the set of preemptible states and \( X_{NP} = X \setminus X_P \) 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. \( X_{new} \subseteq X_P \) 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 \( i \)th state is the initial state of the \( i \)th job in the batch. The batch vector has distribution \( X_{batch} = (X_{batch,1}, \ldots, X_{batch,B}) \), where \( B \) is the distribution of the number of jobs per batch. The batch arrival times are a Poisson process of rate \( \lambda/\bar{E}[B] \), with each batch drawn independently from \( X_{batch} \). The initial state distribution \( X_{new} \) is an aggregate distribution determined by picking a random element from a length-biased sample of \( X_{batch} \).

We allow \( X_{batch} \) 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 \( M^B/G/1 \), in an important way. In the \( M^B/G/1 \), 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 \( x_i \in X \) is the state of the \( i \)th job. We denote the equilibrium distribution of the system state as \((X_{1,\ldots,N})\), 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 \( \pi \), as in \( N^\pi \), to refer to the distribution under scheduling policy \( \pi \).

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 \( x \in X \) by \( \text{hold}(x) \). Holding costs have dimension \text{COST/TIME}. We assume that holding costs are deterministic,
We can lower a job’s holding cost by completing it, in which it until it reaches a state with lower holding cost. Optimal stopping of general Markov processes F. What Does the Scheduler Know?

We assume the scheduler knows a description of the job model: the state space \( X \), the subset of preemptible states \( X_p \subseteq X \), 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 \( \text{hold}(x) \) of each state \( x \in X \). 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.

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 \( X \), one should make some topological assumptions on \( X \) and \( X_p \), as well as some continuity assumptions on holding costs. As another example, when discussing subsets \( \mathcal{Y} \subseteq X_p \) (Definitions VI.1 and IV.2), one should restrict attention to measurable subsets. See Scully et al. [22, 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 [23]. 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 \( M/G/G \) (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 \( \text{hold}(x_{\text{done}}) = 0 \), 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 \( \text{hold}(x) - \text{hold}(y) \), which means

\[
\text{(holding cost decrease rate from } x \text{ to } y) = \frac{\text{hold}(x) - \text{hold}(y)}{x - y}.
\]

To find the fastest possible decrease, we optimize over \( y \):

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

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 dynamics can be stochastic, we need to consider serving it until it enters a set of states \( \mathcal{Y} \). Secondly, because we cannot stop serving a job while it is nonpreemptible, we require \( \mathcal{Y} \subseteq X_p \).

Definition IV.1. For all \( x \in X \) and \( \mathcal{Y} \subseteq X_p \), let

\[
\text{Serve}(x, \mathcal{Y}) = \begin{cases} 
\text{service needed for a job starting in state } x \\
\text{to first enter } \mathcal{Y} \cup \{x_{\text{done}}\}
\end{cases},
\]

\[
\text{serve}(x, \mathcal{Y}) = \mathbb{E}[\text{Serve}(x, \mathcal{Y})],
\]

\[
\text{Hold}(x, \mathcal{Y}) = \begin{cases} 
\text{holding cost of a job starting in state } x \\
\text{when it first enters } \mathcal{Y} \cup \{x_{\text{done}}\}
\end{cases},
\]

\[
\text{hold}(x, \mathcal{Y}) = \mathbb{E}[\text{Hold}(x, \mathcal{Y})].
\]

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

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

Definition IV.2. The (Gittins) index of state \( x \in X \) is

\[
\text{index}(x) = \sup_{\mathcal{Y} \subseteq X_p} \frac{\text{hold}(x) - \text{hold}(x, \mathcal{Y})}{\text{serve}(x, \mathcal{Y})}.
\]

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.

**Example IV.3** (Gittins for mean response time). 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 response time, we give all jobs holding cost 1. The Gittins index (Definition IV.2) is then given by a formula well-known in the literature [5, 6, 8]:

\[
\text{index}(x) = \sup_{y > x} \frac{\mathbb{P}[S \leq y \mid S > x]}{E[\min\{S, y\} - x \mid S > x]}.
\]

**B. Gittins Rank**

Some work on the Gittins policy refers to the (Gittins) rank of a state [8, 14, 22, 24], which is the reciprocal of its index:

\[
\text{rank}(x) = \frac{1}{\text{index}(x)}.
\]

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, a job’s rank is its remaining service time, and thus Gittins reduces to SRPT.

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 \( M_0/G_{\text{MP}}/1 \) 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 \( M_0/G_{\text{MP}}/1 \). That is, for all nonclairvoyant policies \( \pi \),

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

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 \( x \in X \), let

\[
\text{hold}(x) = E[\text{unknown holding cost of a job in state } x \mid \text{job reached state } x],
\]

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

\[
\text{hold}(x) = E[S^{-1} \mid S > x],
\]

and the Gittins index (Definition IV.2) becomes

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

**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 \( x \in X \) and a penalty parameter \( r \geq 0 \), which has dimension \( \text{TIME}^2/\text{COST} \).

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

- We begin by serving the job. The job’s state evolves as usual during service (Section III-A). If the job completes, namely by reaching state \( x_{\text{done}} \), 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 \( r \text{hold}(y) \), where \( y \in X_0 \) 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 \( Y \subseteq X_0 \), give up when the job’s state first enters \( Y \). The strong Markov property implies that this set \( Y \) 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 $x \in X$ and penalty parameter $r$, and the control is a give-up set $Y \subseteq X_P$. The cost of give-up set $Y$ is

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

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

$$
\text{game}(x, r) = \inf_{Y \subseteq X_P} \text{game}(x, r, Y). \tag{VI.1}
$$

**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 $x \in X$ and $r \geq 0$, the cost-to-go function $\text{game}(x, r)$ is

(i) nondecreasing in $r$,

(ii) concave in $r$,

(iii) bounded by $\text{game}(x, r) \leq \text{serve}(x, X_P) + r \text{hold}(x, X_P)$,

(iv) bounded by $\text{game}(x, r) \leq \text{serve}(x, \emptyset)$.

When $x \in X_P$, property (iii) becomes $\text{game}(x, r) \leq r \text{hold}(x)$.

**Proof.** Properties (i) and (ii) follow from (VI.1), which expresses $\text{game}(x, r)$ as an infimum of nondecreasing concave functions of $r$. Properties (iii) and (iv) follow from the fact that two possible give-up sets are $X_P$, meaning giving up as soon as possible, and $\emptyset$, meaning never giving up. The simplification when $x \in X_P$ 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

$$
\text{Optimal give-up set}(r) = \{ x \in X_P \mid \text{game}(x, r) = r \text{hold}(x) \}.
$$

Note that $\text{Optimal give-up set}(0) = X_P$. We also let $\text{Optimal give-up set}(\infty) = \emptyset$. For simplicity of language, we call $\text{Optimal give-up set}(r)$ “the” optimal give-up set, even though there may be other optimal give-up sets.

Basic results in optimal stopping theory [23] imply that

$$
\text{game}(x, r) = \text{game}(x, r, \text{Optimal give-up set}(r)),
$$

so the infimum in (VI.1) is always attained, namely by $\text{Optimal give-up set}(r)$.

The sets $\text{Optimal give-up set}(r)$ are monotonic in $r$, i.e. $\text{Optimal give-up set}(r) \supseteq \text{Optimal give-up set}(r')$ 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 $x \in X$ and $r \geq 0$, let

$$
\text{Serve}(x, r) = \text{Serve}(x, \text{Optimal give-up set}(r))
$$

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

**D. Derivative of the Cost-To-Go Function**

Suppose we solve the Gittins game for penalty parameter $r$, then change the penalty parameter to $r + \varepsilon$ for some small $\varepsilon > 0$. One would expect that the give-up set $\text{Optimal give-up set}(r)$ is nearly optimal for the new penalty parameter $r + \varepsilon$, which would imply $\text{game}(x, r + \varepsilon) \approx \text{serve}(x, r) + (r + \varepsilon) \text{hold}(x, r)$. One can use Lemma VI.2 and a classic envelope theorem [25, Theorem 1] to formalize this argument.

**Lemma VI.5.** For all $x \in X_P$, the function $r \mapsto \text{game}(x, r)$ is differentiable almost everywhere with derivative

$$
\frac{d}{dr} \text{game}(x, r) = \text{hold}(x, r).
$$

For brevity, we omit the proof of Lemma VI.5. See Scully et al. [22, Lemma 5.3] for a similar proof.

**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.

Section VI-A describes the goal of the Gittins game as being to end the game as quickly as possible. An alternative intuition is that the goal is to reduce the job’s holding cost to zero as quickly as possible. Under this intuition, we think of giving up as starting a process that decreases the job’s holding cost at constant rate $1/r$, where $r$ is the penalty parameter. Giving up in preemptible state $x$ thus takes $r \text{hold}(x)$ time.

Consider playing the Gittins game with starting state $x \in X_P$ and penalty parameter $r \geq 0$. How do we decide whether or not to give up? That is, how do we determine whether $x \in \text{Optimal give-up set}(r)$? On one hand, Definition IV.2 tells us that by serving the job, we can decrease its holding cost at expected rate $1/\text{rank}(x)$. On the other hand, giving up decreases the holding cost at rate $1/r$. The natural conclusion is that giving up is optimal, i.e. $x \in \text{Optimal give-up set}(r)$, if and only if $\text{rank}(x) \geq r$.

The intuition above turns out to be exactly right: $\text{rank}(x)$ is the maximum penalty parameter $r$ such that giving up is still optimal when in state $x$.

**Lemma VI.6.**

(i) For all $r \geq 0$, we can write the optimal give-up set as

$$
\text{Optimal give-up set}(r) = \{ x \in X_P \mid \text{rank}(x) \geq r \}.
$$

(ii) For all $x \in X_P$, we can write the Gittins rank of $x$ as

$$
\text{rank}(x) = \max \{ r \geq 0 \mid x \in \text{Optimal give-up set}(r) \}
$$

$$
= \max \{ r \geq 0 \mid \text{game}(x, r) = r \text{hold}(x) \}
$$

$$
= \inf \{ r \geq 0 \mid x \notin \text{Optimal give-up set}(r) \}
$$

$$
= \inf \{ r \geq 0 \mid \text{game}(x, r) < r \text{hold}(x) \}.
$$

7
For brevity, we omit the proof of Lemma VI.6. See Scully et al. [22, Lemma 5.4] for a similar proof.

VII. PROVING GITTINS’S OPTIMALITY

We now prove Theorem V.1, namely that Gittins minimizes mean holding cost in the $M_B/G_MP/1$. Our proof has four steps.

- We begin by showing that minimizing mean holding cost $E[H]$ is equivalent to minimizing the mean preemptible holding cost $E[H_P]$, 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 $r$” (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 $E[H]$ (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

$$H_P = \sum_{i=1}^{N} \mathbb{1}(X_i \in X_P) \text{hold}(X_i),$$

where $\mathbb{1}$ is the indicator function. The nonpreemptible holding cost is defined analogously and has equilibrium distribution

$$H_{NP} = \sum_{i=1}^{N} \mathbb{1}(X_i \in X_{NP}) \text{hold}(X_i).$$

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

Lemma VII.2. In the $M_B/G_MP/1$, the mean preemptible holding cost is the same under all scheduling policies:

$$E[H_{NP}] = \lambda E \left[ \text{total cost a job accrues while in a nonpreemptible state during service} \right].$$

Proof. By a generalization of Little’s law [1, 2],

$$E[H_{NP}] = \lambda E \left[ \text{total cost a job accrues while in a nonpreemptible state} \right].$$

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

B. Defining $r$-Work

Definition VII.3. The (job) $r$-work of state $x$ is $\text{Serve}(x,r)$, namely the amount of service it requires to either complete or enter a preemptible state of rank at least $r$. The (system) $r$-work

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

is the total $r$-work of all jobs in the system. Its equilibrium distribution, denoted $W(r)$, is

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

where $(X_1, \ldots, X_N)$ is the equilibrium system state (Section III-D). We define the (system) preemptible $r$-work $W_P(r)$ and (system) nonpreemptible $r$-work $W_{NP}(r)$ similarly, except we only count $r$-work from jobs in preemptible or nonpreemptible states:

$$W_P(r) = \sum_{i=1}^{N} \mathbb{1}(X_i \in X_P) \text{Serve}(X_i,r),$$

$$W_{NP}(r) = \sum_{i=1}^{N} \mathbb{1}(X_i \in X_{NP}) \text{Serve}(X_i,r).$$

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

$$E[W_P(r)] = E \left[ \sum_{i=1}^{N} \mathbb{1}(X_i \in X_P) \text{serve}(X_i,r) \right].$$

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

C. Relating $r$-Work to Holding Cost

Theorem VII.5. In the $M_B/G_MP/1$, under all nonclairvoyant policies,

$$E[H_P] = \int_{0}^{\infty} \frac{E[W_P(r)]}{r^2} \, dr.$$

Proof. By Lemma VII.4 and Definition VII.1, it suffices to show that for all $x \in X_P$,

$$\text{hold}(x) = \int_{0}^{\infty} \frac{\text{serve}(x,r)}{r^2} \, dr. \quad \text{(VII.1)}$$

Using Lemma VI.5, we compute

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

This means the integral in (VII.1) becomes a difference between two limits. Using Lemma VI.5 for the $r \to 0$ limit and Lemma VI.2(iv) for the $r \to \infty$ limit, we obtain

$$\int_{0}^{\infty} \frac{\text{serve}(x,r)}{r^2} \, dr = \lim_{r \to 0} \frac{\text{game}(x,r)}{r} - \lim_{r \to \infty} \frac{\text{game}(x,r)}{r} = \text{hold}(x,0) - 0 = \text{hold}(x). \quad \text{\qed}$$

Theorem VII.5 implies that to minimize $E[H_P]$, it suffices to minimize $E[W_P(r)] = E[W(r) - E[W_{NP}(r)]$ for all $r \geq 0$. It turns out that $E[W_{NP}(r)]$, much like $E[H_{NP}]$, is unaffected by the scheduling policy, so it suffices to minimize $E[W(r)]$.

Lemma VII.6. In the $M_B/G_MP/1$, the mean nonpreemptible $r$-work $E[W_{NP}(r)]$ is the same under all scheduling policies.

We omit the proof of Lemma VII.6, as it very similar to that of Lemma VII.2.
We classify jobs in the system into two types. A straightforward interchange argument shows that it suffices to explain how to view the $r$-work in the $M_B/G_{MP}/1$ as the virtual work in a vacation system.\footnote{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.}

- 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 [26, Theorem 3.3] implies

$$E[W^r(r)] = c_1 + c_2E\left[r\text{-work sampled immediately}\right] - E\left[r\text{-work} \right] + E\left[r\text{-work conserved}\right].$$

where $c_1$ and $c_2$ are constants that depend on the system parameters but not on the scheduling policy $\pi$. 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. $\square$


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 $E[W(r)]$ for all $r \geq 0$, then it minimizes mean holding cost $E[H]$. 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 $\pi$ and $r \geq 0$,

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

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 $X \setminus \mathcal{Y}^r(r)$.
- A job is $r$-bad if it has Gittins rank at least $r$, i.e. its state is in $\mathcal{Y}^r(r)$.

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.

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. [8, 22], 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 $\pi$. It is convenient to allow $\pi$ to be more powerful than an ordinary policy: we allow $\pi$ to devote infinite processing power to $r$-bad jobs. This has two implications:

- Whenever there is $r$-work in the system, $\pi$ 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, $\pi$ 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 $\pi$ 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 [26].
[12] 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.

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

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

[15] 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.

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

[17] P. Nash, “Optimal allocation of resources between research projects,” Ph.D. dissertation, University of Cambridge, Cambridge, UK, 1973.

[18] 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.

[19] 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.

[20] 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.

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

[22] 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.

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

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

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

[26] 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.