The Gittins Index: A Design Principle for Decision-Making Under Uncertainty

Ziv Scully
School of Operations Research and Information Engineering, Cornell University, Ithaca, NY, USA
zivscully@cornell.edu

Alexander Terenin
Center for Data Science for Enterprise and Society, Cornell University, Ithaca, NY, USA
avt28@cornell.edu

Abstract The Gittins index is a tool that optimally solves a variety of decision-making problems involving uncertainty, including multi-armed bandit problems, minimizing mean latency in queues, and search problems like the Pandora's box model. However, despite the above examples and later extensions thereof, the space of problems that the Gittins index can solve perfectly optimally is limited, and its definition is rather subtle compared to those of other multi-armed bandit algorithms. As a result, the Gittins index is often regarded as being primarily a concept of theoretical importance, rather than a practical tool for solving decision-making problems.

The aim of this tutorial is to demonstrate that the Gittins index can be fruitfully applied to practical problems. We start by giving an example-driven introduction to the Gittins index, then walk through several examples of problems it solves—some optimally, some suboptimally but still with excellent performance. Two practical highlights in the latter category are applying the Gittins index to Bayesian optimization, and applying the Gittins index to minimizing tail latency in queues.

Keywords Gittins index; Pandora's box; multi-armed bandit; scheduling; M/G/1 queue; Bayesian optimization; tail latency

Contents

1 Introduction **2**

2 An illustrative example: Pandora's box **3**
2.1 Pandora's box as a Markov decision process 4
2.2 Why obvious greedy policies are suboptimal 5
2.2.1 Connecting expected improvement to one-step lookahead 6
2.3 Defining the Gittins index for Pandora's box 7
2.4 Extensions and limitations: what else can the Gittins index do? 8

3 General formulation of the Gittins index **9**
3.1 The Markov Chain Selection decision problem 9
3.1.1 A note on the name *Markov chain selection* 11
3.2 Defining the Gittins index via the local MDP 11
3.3 Optimality of the Gittins policy 13
3.4 Computation 14

4 Examples: optimal policies **14**
4.1 Two-stage Pandora's box . . . 15
4.2 Bayesian Bernoulli bandits with discounting . . . 16
4.3 Selecting multiple boxes, or finishing multiple Markov chains . . . 16
4.4 Scheduling in queues . . . 17
4.4.1 Representing batch scheduling as MCS . . . 18
4.4.2 From batch scheduling to queue scheduling . . . 19
4.5 Branching bandits . . . 21

5 Examples: beyond optimality **22**
5.1 Bayesian optimization . . . 22
5.1.1 Defining a Gittins index for Bayesian optimization . . . 23
5.1.2 Maximizing the Gittins index numerically . . . 24
5.1.3 Performance of the Gittins index in Bayesian optimization . . . 24
5.2 Pandora's box with optional inspection . . . 25
5.2.1 Why MDP selection is harder than Markov chain selection . . . 26
5.2.2 Approximate solutions to MDP selection using the Gittins index . . . 28
5.3 Minimizing tail latency in queues . . . 29

6 Conclusion **30**
6.1 Additional topics . . . 31
6.2 Open problems . . . 31

References **32**

A Optimality of the Gittins index policy **38**
A.1 Optimality via dynamic programming using surrogate values . . . 38
A.2 Generalization: finishing multiple Markov chains . . . 43
A.3 Comparison to other optimality proofs . . . 44

1. Introduction

Making effective decisions under uncertainty is a central theme in many areas of science, engineering, and technology. Algorithms for making decisions are therefore central to many fields—from operations research to economics and artificial intelligence, to name a few. In many—though certainly not all—such situations, it can be appropriate to model the decision-making problem stochastically, by formalizing it as a fully specified Markov decision process.

An important subclass of such Markov decision processes represents, loosely speaking, *choosing the best option among a set of alternatives* in the face of stochastic feedback about which option is best. This includes the class of Bayesian multi-armed bandits, but also a number of other decision problems which at first glance might appear to be of a rather different character, such as consumer choice problems arising in economics, or certain optimal scheduling problems arising in queueing theory.

An underappreciated fact about such problems is that *there is a general way to solve them exactly* using a class of techniques broadly known as *Gittins index theory*. These techniques have been rediscovered in various communities, often through the lens of a surprising solution to a specific decision problem. As an example, let us quote a widely known exchange between two colleagues as recalled by Peter Whittle [45] and abridged by Richard Weber [105]:

> A colleague of high repute asked an equally well-known colleague:
>
> — What would you say if you were told that the multi-armed bandit problem had been solved?
>
> — Sir, the multi-armed bandit problem is not of such a nature that it *can* be solved.

Reflecting this sentiment, our first goal in this tutorial will be to illustrate how such solutions work, focusing chiefly on the *key definitions*. In doing so, we show that Gittins index theory tells us, intuitively: to choose the best option among a set of alternatives under stochastic feedback, *compare each stochastic option with an equivalent deterministic option.*

A second, even more underappreciated fact about the decision problems we study is that *the definitions arising from exact solutions of simple problems can also yield strong solutions for more complex problems* where there is no hope of an exact solution. In his course on information-directed sampling, Tor Lattimore described the setup covered by Gittins index theory as the *"miraculous case"* [70]—to contrast it with more general situations where optimal policies are intractable. While this might tempt one to conclude that the Gittins index is not an appropriate technical tool for such settings, the basic idea of comparing stochastic options with equivalent deterministic options often continues to make intuitive sense, even though it is no longer optimal.

Against this background, our second goal is to illustrate how Gittins index theory can be used as a design principle for general decision problems. For this, we showcase problems where policies based on the Gittins index, while not optimal, are known to perform strongly—either in a theoretical or an empirical sense, covering examples from queueing theory and Bayesian optimization.

Our tutorial begins by covering what Gittins indices are and how they arise, intuitively. This is done by way of analyzing Pandora's box, arguably the simplest concrete example, in Section 2—and then placing it into a suitable abstract framework in Section 3. We then survey various extensions and advanced examples. Section 4 covers settings where optimality holds, while Section 5 covers settings where optimality does not hold, but the Gittins index still either (a) has excellent empirical performance, (b) satisfies an approximate or asymptotic optimality guarantee, or (c) both. Two particularly practically relevant examples are Bayesian optimization (Section 5.1) and scheduling to minimize tail latency (Section 5.3). We give a more detailed outline of the themes covered by the examples in Section 2.4. We conclude our tutorial in Section 6 with a discussion of open problems.

2. An illustrative example: Pandora's box

The easiest way to understand the Gittins index is by way of example: for this, we present the Pandora's box problem from economics, originally due to Weitzman [108]. In the Pandora's box problem, the decision-making agent is presented with a set of n *boxes*, labeled $i \in \{1, \dots, n\}$. For a set X, let $\mathcal{P}(X)$ be the space of probability measures over X. Each box is associated with two quantities:

1. A *cost to open* $c_i > 0$.
2. A *reward distribution* $p_i \in \mathcal{P}(\mathbb{R})$, assumed to have finite mean.

At the starting time, each box is assumed to be *closed*: we conceptually imagine it to contain a reward inside which is unknown to the agent and viewed as random. Starting from the state where all boxes are closed, at each time point, the agent is allowed to either:

1. *Open a closed box* i of their choosing: the agent pays a cost of c_i, and learns the precise value of the reward $v_i \sim p_i$ which is contained inside the box.
2. *Select an open box* i from the set of open boxes: the agent's decision-making process ends, and they receive the reward v_i which is inside the box.

Letting T be the time when the agent selects a box, and i_t be the box opened resp. selected at time t, the agent's aim is to maximize their expected total value, which is

$$\mathbb{E}\left[v_{i_T} - \sum_{t=1}^{T-1} c_{i_t}\right]. \tag{2.1}$$

A very important aspect of this formulation is that, even though the agent can open multiple boxes and must pay a cost *for each box*, they ultimately receive *only one reward*—namely, the value in the box they selected at the very end. This results in an explore-exploit tradeoff: should the agent pay to learn about more possible rewards they might eventually select, or are the rewards revealed already good enough?

2.1. Pandora's box as a Markov decision process

We can formulate the Pandora's box problem (denoted PB in subscripts) as a discrete-time Markov decision process (MDP) as follows. Define the state space to be the set of tuples

$$S_{\text{PB}} = \{(s_1, \dots, s_n) : s_i \in \{\boxtimes\} \cup \mathbb{R} \cup \{\checkmark\}\} \tag{2.2}$$

where $s_i = \boxtimes$ represents box i being closed, $s_i \in \mathbb{R}$ represents box i being open with reward $v_i = s_i$, and $s_i = \checkmark$ represents box i being selected. This is a finite-horizon undiscounted MDP, whose initial state is $(\boxtimes, \dots, \boxtimes)$, and terminal states are tuples $(s_1, \dots, s_n)$ with $s_i = \checkmark$ for some i. The action space is $A_{\text{PB}} = \{1, \dots, n\}$, which refers to indices of boxes. Each action $a \in A_{\text{PB}}$ corresponds to either opening the respective box, or selecting it: the transition kernel's action replaces the corresponding identifier in the tuple. For example, for a three-closed-box state and $a = 1$, this occurs by

$$(\boxtimes, \boxtimes, \boxtimes) \overset{a=1}{\mapsto} (v_1, \boxtimes, \boxtimes) \tag{2.3}$$

where $v_1 \sim p_1$ is the revealed reward. Throughout this section, we focus on the classical variant where all boxes' rewards are independent. The MDP's reward function, defined for non-terminal states and all actions, is

$$r_{\text{PB}}(s, a) = \begin{cases} -c_a & s_a = \boxtimes \\ s_a & s_a \in \mathbb{R} \end{cases} \tag{2.4}$$

which returns either the negated costs of a closed box, or the previously revealed reward of an open box—here and throughout, we work with MDPs that allow negative rewards. Our expected total value defined previously is therefore equal to the MDP's value function. Continuing our three-box example, choosing $a = 1$ a second time would transition

$$(v_1, \boxtimes, \boxtimes) \overset{a=1}{\mapsto} (\checkmark, \boxtimes, \boxtimes) \tag{2.5}$$

with a reward of v_1, for an overall value of $v_1 - c_1$.

By general MDP theory, there exists an optimal policy which maximizes the agent's expected value. At first glance, however, it is unclear how much more one can expect to say about this policy. On the one hand, the decision problem is, at the vaguest level, clean enough that one might hope for a surprisingly straightforward solution—as occurs, in, say, the secretary problem. On the other hand, for general Markov decision processes, there is little hope of saying much about the optimal policy. It will turn out that, using the Gittins index, one can obtain the optimal policy analytically—producing a solution that will turn out to be both straightforward and subtle at the same time.

Before proceeding, we conclude with a few comments. First, note that the agent's policy can be adaptive: they can—and should—use values from opened boxes to decide what action to take next. Second, note that decision problem is not an unknown-MDP statistical learning problem: all of MDP parameters, and in particular all reward distributions p_i defining the transition kernel, are known to the agent. In this problem, therefore, learning takes place in the sense of conditional probability: the agent learns the box's value by opening it.

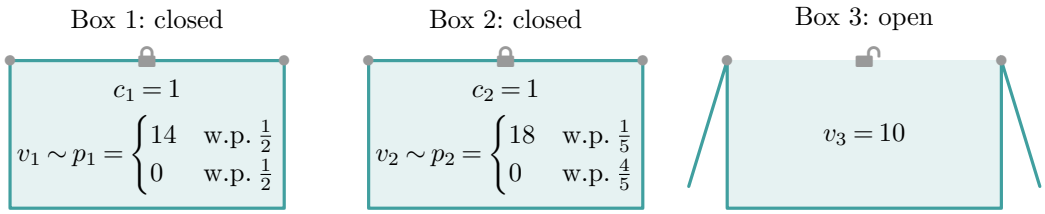


FIGURE 2.1. An instance of the Pandora's box problem with two closed boxes and one opened box. Here, we know the realized value of the opened box 3, but only the reward distributions and not the realized value of the closed boxes 1 and 2.

2.2. Why obvious greedy policies are suboptimal

Before explaining how to solve the Pandora's box problem, it is worth walking through an example that illustrates why the problem is non-trivial. In particular, we will see that the most obvious greedy policy, consisting of choosing boxes according to the expected difference between rewards and costs, is suboptimal. We will then see that the second-most obvious greedy policy, specifically one-step lookahead, is also suboptimal. This might make one think greedy policies are not the right approach: remarkably, however, the optimal policy *will* turn out to be a greedy policy—albeit with respect to a more subtle objective compared to the above two policies.

Consider the three-box scenario shown in Figure 2.1. In the situation represented by this state, we have already opened box 3, so we take its realized value $v_3 = 10$ to be fixed throughout this example, and ignore its cost because it has been paid already. To decide what to do next, we have to answer two questions:

(a) Is it worth opening another box, or should we stop now?

(b) If we open another box, should we open box 1 or box 2?

For (a), a naïve idea is to think about how valuable each closed box i would be if it were the only box we had, which is simply $\mathbb{E}[v_i] - c_i$. We compute

$$\mathbb{E}[v_1] - c_1 = \tfrac{1}{2} \cdot 14 - 1 = 6 \tag{2.6}$$
$$\mathbb{E}[v_2] - c_2 = \tfrac{1}{5} \cdot 18 - 1 = 2.6. \tag{2.7}$$

Both of these are much less than $v_3 = 10$, so it may seem like boxes 1 and 2 are less valuable than box 3, which might suggest we should stop now.

However, the above computation fails to account for a crucial fact: *box 3's reward remains available even if we open another box.* That is, if we were to open box i, and then stop, we can receive a reward of either v_i or v_3 when we stop, depending on our selection, and not simply v_i. We assume henceforth that we always select the best open box: this means our reward is $\max(v_i, v_3)$.[1] Computing the expected value of opening each box and then stopping, we find

$$\mathbb{E}[\max(v_1, v_3)] - c_1 = \tfrac{1}{2} \cdot 14 + \tfrac{1}{2} \cdot 10 - 1 = 11 \tag{2.8}$$
$$\mathbb{E}[\max(v_2, v_3)] - c_2 = \tfrac{1}{5} \cdot 18 + \tfrac{4}{5} \cdot 10 - 1 = 10.6. \tag{2.9}$$

Both of these are greater than 10, so we conclude that opening either closed box and then stopping is better than stopping now—thus, we should definitely open at least one more box.

[1] Recall again that we can ultimately only receive the reward from one box: this is why we receive the maximum of v_i and v_3, and not their sum.

Having decided to open a box, we move on to (b): which box should we open? As a first step towards answering this, it will help to review (2.8) and (2.9) from a different perspective. Let the *expected improvement* of box i over a baseline α be

$$\mathrm{EI}_i(\alpha) = \mathbb{E}[\max(v_i - \alpha, 0)] - c_i. \tag{2.10}$$

That is, if the current best value (among open boxes) is α, then $\mathrm{EI}_i(\alpha)$ is the expected amount we gain by opening box i instead of stopping now, accounting for both the cost of opening the box $-c_i$ and increasing the best value from α to $\max(v_i, \alpha)$. The expected value of opening box i and then stopping is thus $\alpha + \mathrm{EI}_i(\alpha)$. We can recognize (2.8) and (2.9) as $\alpha + \mathrm{EI}_i(\alpha)$ values with baseline $\alpha = v_3 = 10$:

$$\mathrm{EI}_1(v_3) = \tfrac{1}{2} \cdot (14 - 10) - 1 = 1 \tag{2.11}$$

$$\mathrm{EI}_2(v_3) = \tfrac{1}{5} \cdot (18 - 10) - 1 = 0.6. \tag{2.12}$$

So, box 1 has better expected improvement than box 2, which might suggest we should open box 1. In fact, if we were only allowed to open one more box, then this computation shows that opening box 1 would be best.

Of course, we are not limited to opening just one more box. In particular, if we open a box but find its realized value is low (say, 0), then $v_3 = 10$ would still be our best value. In this case, the remaining closed box would still have positive expected improvement, so it would make sense to open it. This reasoning suggests the following policy:

- We first open one of the closed boxes.
- If the realized value is high (say, 14 or 18), we stop.
- If the realized value is low (say, 0), we open the other closed box.

We can carry this out starting by opening either box 1, which we call *policy P1*, or box 2, which we call *policy P2*. Our expected value under P1 is

$$\mathbb{E}\big[\mathbb{1}(v_1 = 14) \cdot 14 + \mathbb{1}(v_1 = 0) \cdot (\max(v_2, v_3) - c_2)\big] - c_1 = 11.3 \tag{2.13}$$

which is better than opening box 1 then stopping. But our expected value under P2 is

$$\mathbb{E}\big[\mathbb{1}(v_2 = 18) \cdot 18 + \mathbb{1}(v_2 = 0) \cdot (\max(v_1, v_3) - c_1)\big] - c_2 = 11.4 \tag{2.14}$$

which is even better. Thus, even though box 1 has higher expected improvement than box 2, policy P2 outperforms P1. In fact, with a little bit more casework, one can show P2 is optimal!

2.2.1. Connecting expected improvement to one-step lookahead There is a connection between expected improvement and the one-step lookahead policy. Specifically, the *one-step lookahead* policy behaves as follows at every time step:

- Let $v_{\max}$ be the maximum realized value among opened boxes.
- If $\max_i \mathrm{EI}_i(v_{\max}) < 0$, namely if every box's expected improvement is negative, then stop.
- Otherwise, open $\arg\max_i \mathrm{EI}_i(v_{\max})$, namely the box of greatest expected improvement.

Policies like this are a standard approach used for constructing Bayesian optimization algorithms for black-box optimization, where they give rise to the *expected improvement acquisition function*: we will return to this in Section 5.

The example from Figure 2.1 we have just worked through demonstrates that *one-step lookahead is suboptimal*. In particular, one can check that P1 is the one-step lookahead policy, and its expected value is 0.1 less than that of P2. This might not seem so bad, but there are other Pandora's box instances where the performance gap between one-step lookahead and the optimal policy can be arbitrarily large—a performance counterexample is given by Singla [96, Appendix A.1]. With this background, we proceed to study the structure of P2.

2.3. Defining the Gittins index for Pandora's box

We have seen in Section 2.2 that one-step lookahead does not solve Pandora's box. To find the optimal policy, we essentially had to resort to brute force in (2.13) and (2.14). This will not work in larger instances with more boxes. Can we still solve Pandora's box in such cases?

Remarkably—as first shown by Weitzman [108] for Pandora's box, and Gittins [44] in a general abstract setting—the optimal policy, which is called the *Gittins index policy* or, more concisely, the *Gittins policy*, is nearly as simple as one-step lookahead. Specifically, both the Gittins policy and one-step lookahead are *index policies*: they work by computing a numeric rating for each box, called the box's *index*, then opening the box with the best index. Such policies are also known in the Bayesian optimization literature as *acquisition functions*: we return to this connection in Section 5. We consider the following index policies:

a. Under one-step lookahead, a box's index is its expected improvement over the current best value.

b. Under the Gittins policy, a box's index is a quantity called, appropriately, its *Gittins index*.

Below, we give a quick definition of the Gittins index of a (closed) box and discuss its relationship to expected improvement. Later on, we discuss in more depth where this definition comes from, and *why* it is natural.

We first briefly review one-step lookahead. Suppose box i is closed, and suppose the current best value is $v_{\max}$. One-step lookahead sets box i's index to $\mathrm{EI}_i(v_{\max})$, defined in (2.10). Roughly speaking, this index answers the question: *how valuable is it to open box i?*

The Gittins index comes from a related but different question. *Ignoring the actual value of $v_{\max}$*, we ask: *hypothetically, if we had $v_{\max} = \alpha$, how large would α need to be to rule out opening box i?* If box's i's expected improvement over α were negative, say if $\mathrm{EI}_i(\alpha) < 0$, we could rule out opening box i: simply stopping with reward α would be a better action. So, define the *Gittins index of box i*, denoted G_i, to be the solution to the root-finding problem

$$\mathrm{EI}_i(G_i) = 0 \tag{2.15}$$

or, equivalently, $\mathbb{E}[\max(v_i - G_i, 0)] = c_i$. To see that the Gittins index is well defined, meaning there is a unique solution G_i in (2.15), note that $\mathrm{EI}_i(\alpha)$ is convex (and hence, since its domain is the real line, continuous), decreasing as a function of α, and satisfies

$$\lim_{\alpha \to \infty} \mathrm{EI}_i(\alpha) = -c_i < 0 < \infty = \lim_{\alpha \to -\infty} \mathrm{EI}_i(\alpha). \tag{2.16}$$

Note also that a higher Gittins index corresponds to a more desirable box. For instance, increasing c_i decreases G_i.[2]

Having defined the Gittins index, we can define how the Gittins policy behaves:

- Let $v_{\max}$ be the maximum realized value among opened boxes.
- If $\max_i G_i < v_{\max}$, meaning if every box's Gittins index is worse (less) than $v_{\max}$, then stop and select the best open box.
- Otherwise, open $\arg\max_i G_i$, the box of best (greatest) Gittins index.

In fact, if we extend the Gittins index to also cover open boxes, by letting the Gittins index of an open box be its revealed reward value, then—up to a choice of how to resolve ties—the Gittins policy reduces to one rule:

($\star$) *Always take the action of greatest Gittins index.*

[2] Here and throughout, *increasing* and *decreasing* are meant in their non-strict forms by default.

Returning to the example boxes in Figure 2.1, their Gittins indices are

$$\mathrm{EI}_1(\alpha) = \tfrac{1}{2} \cdot \max(14 - \alpha, 0) - 1 \qquad \Rightarrow \qquad g_1 = 12 \tag{2.17}$$
$$\mathrm{EI}_2(\alpha) = \tfrac{1}{5} \cdot \max(18 - \alpha, 0) - 1 \qquad \Rightarrow \qquad g_2 = 13. \tag{2.18}$$

Box 2 thus has better Gittins index than either box 1 or the open box ($v_{\max} = V_3 = 10$), so the Gittins policy would open box 2—the optimal action we saw in Section 2.2!

2.4. Extensions and limitations: what else can the Gittins index do?

At a high level, the Pandora's box problem is a model of *search with costly information acquisition*. Our goal is to in find a good value, but without paying too much opening costs. In general, search problems like this can, at first, appear to be completely different than Pandora's box—indeed, the seminal work of Gittins and Jones [46] was motivated by discounted Bayesian multi-armed bandits (Section 4.2). Even direct generalizations can have many features that go beyond the classical Pandora's box:

1. *Multiple stages of inspection*. In Pandora's box, the reward is revealed immediately. In general, it might be revealed gradually over time (Section 4.1).
2. *Searching for multiple values*. In Pandora's box, our reward comes from just one box. In other settings, it may depend on the state of multiple boxes (Section 4.3).
3. *Dynamic sets of options*. In Pandora's box, the set of boxes is fixed. In general, new actions might become possible over time, such as when scheduling a set of randomly arriving jobs in a queueing system (Section 4.4), or more generally (Section 4.5).
4. *Correlations between values*. In Pandora's box, the rewards of different boxes are assumed independent. In general, they can be correlated. This is common in Bayesian optimization, where rewards are modeled using Gaussian processes (Section 5.1).
5. *Optional inspection*. In Pandora's box, a box must be opened before being selected. In general, it might be possible to select a box and receive its reward without first opening it (Section 5.2).
6. *Metrics beyond expected value*. In Pandora's box, the objective is to maximize expected net reward. In general, one might hope to optimize metrics beyond expected value. For instance, when scheduling in a queueing system, it is often more important to prevent very long delays than to reduce the average delay (Section 5.3).

In which of these situations can the Gittins index be defined? In what cases is the resulting Gittins policy still optimal? What if we have several of these aspects simultaneously, such as multiple stages of inspection, with parts that can be skipped?

A principal aim of the rest of this tutorial is to shed light on these questions. To do so, we first need to be able to precisely describe the potential features listed above. To that end, over the next few sections, we present a unifying decision-making framework which includes Pandora's box—as well as well as related problems from seemingly different settings such as optimal queueing—into a common language.

At this level of generality, the Gittins index can be defined, and the resulting Gittins policy is optimal. We then generalize our framework further to capture the more difficult of the advanced variants presented above: in most such situations, the Gittins index can still be defined, but can only be expected to yield a strong policy—in the spirit, of, say, upper confidence bounds or information-theoretic decision rules—rather than an outright optimal one. We conclude the tutorial by surveying applications where the Gittins policy, despite being suboptimal, has excellent theoretical or empirical performance.

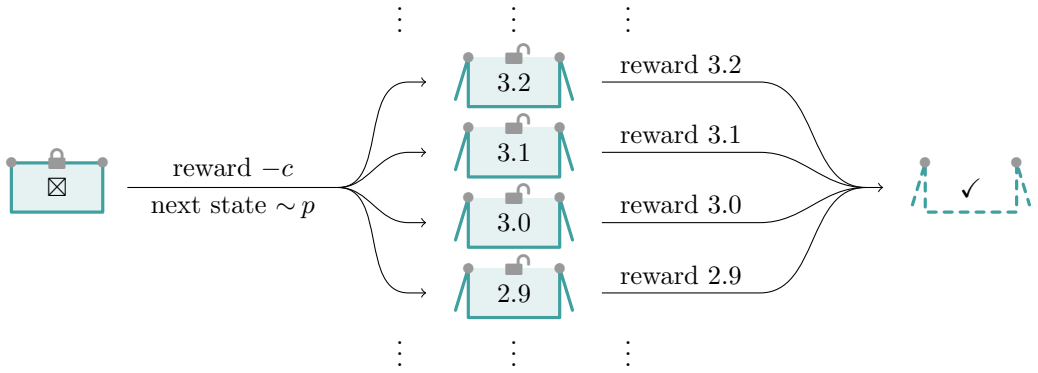


FIGURE 3.1. Illustration of the Markov chain for a box with opening cost c and reward distribution p. The states are *closed*, denoted ⊠; *opened with reward* $v \in \mathbb{R}$, denoted v; and *selected*, denoted ✓.

3. General formulation of the Gittins index

Having seen the Pandora's box problem, its formulation as a Markov decision process, and its solution, one can ask: *is there a general theory this solution is an example of?* We now present such a theory, focusing on a formulation that generalizes well beyond Pandora's box.

The key idea is to conceptualize each individual Pandora's box as a *transient Markov chain* with one absorbing state, with the collection of all boxes represented as tuples whose ith element is the Markov chain state corresponding to box i. Figure 3.1 provides an illustration. An action i then amounts to advancing the ith Markov chain forward by one step, and collecting whatever rewards arise as a result—where costs are represented as negative rewards. The agent must select which Markov chain to advance at each time point.

This formulation enables one to handle many situations which, at first, appear to have little to do with Pandora's box. This includes Bayesian variants of the multi-armed bandit problem [44, 46] and mean delay minimization in single-server queues [67, 68, 94, 95, 103], both of which predate the Pandora's box of Weitzman [108], and for which Gittins indices were discovered independently. We now study a framework that enables one to see how optimal policies arise in all of these setups.

3.1. The Markov Chain Selection decision problem

We begin by formulating our general decision problem, which we call *Markov chain selection (MCS)*, as a Markov decision process.

Definition 3.1 (Markov chain)**.** A *Markov chain*[3] *with rewards* is a tuple $(S, \partial S, p, r)$ consisting of:

1. The *state space* S.
2. A subset of *terminal states* $\partial S \subseteq S$, which may be empty.
3. A *transition kernel* $p : S \to \mathcal{P}(S)$.
4. A *reward function* $r : S \to \mathbb{R}$, which may take negative values.

We require that the terminal states be absorbing with zero reward, namely $p(s)$ deterministically maps $s \in \partial S$ to itself, and $r(s) = 0$ for all $s \in \partial S$.

[3] Note that our definition of a Markov chain (without rewards) is is equivalent to the usual random-variable-theoretic formulation from probability theory. Here, we opt to work with transition kernels, rather than collections of random variables, because this will make it notationally cleaner to track relationships between various different Markov decision processes that arise in our context.

To ease terminology, we often omit *with rewards*. Note that when specifying a Markov chain, it suffices to define the part of the transition kernel which describes transitions out of non-terminal states, and similarly it suffices to define rewards for non-terminal states.

Definition 3.2 (Markov chain selection)**.** Define the *Markov chain selection (MCS)* problem with Markov chains $(S_i, \partial S_i, p_i, r_i)$, for $i \in \{1, \dots, n\}$, to be the MDP given as follows:

1. State space: let the state space be
$$S_{\text{MCS}} = \{(s_1, \dots, s_n) : s_i \in S_i \text{ for all } i\} \tag{3.1}$$
together with an initial state whose components are all non-terminal.
2. Terminal states: let the terminal state set be
$$\partial S_{\text{MCS}} = \{(s_1, \dots, s_n) \in S_{\text{MCS}} : s_i \in \partial S_i \text{ for some } i\}. \tag{3.2}$$
3. Action space: let $A_{\text{MCS}} = \{1, \dots, n\}$.
4. Reward function: let
$$r_{\text{MCS}}(s, a) = r_a(s_a). \tag{3.3}$$
5. Transition kernel: given a state $(s_1, \dots, s_n)$ and action a, we transition the MDP into a new state by replacing s_a with $s'_a \sim p_a(s_a)$ according to that respective Markov chain's transition kernel, leaving other states unchanged.
6. Discount factor: let $\gamma \in (0, 1]$.

To understand this definition, note that Pandora's box is a special case where each Markov chain, illustrated Figure 3.1, is as follows. For box i, the state space is $S_i = \{\boxtimes\} \cup \mathbb{R} \cup \{\checkmark\}$ and transition kernel given by $p_i(\boxtimes) = p_i$ and $p_i(v_i) = p_i(\checkmark) = \delta_\checkmark$, where $\delta_\checkmark$ is the Dirac measure at the symbol $\checkmark$. In this Markov chain, $\checkmark$ is the unique absorbing state, which is also terminal, and all other states are transient. Here, we take $\gamma = 1$.

In general, choosing an action thus corresponds to choosing which Markov chain to transition—an abstract generalization of choosing which box to open. Our formulation works with time-homogeneous Markov chains: this is without loss of generality, as non-time-homogeneous chains can be handled by adjoining time to their state space, transforming them into time-homogeneous ones. To ensure well-definedness, we make the following assumption on each individual Markov chain.

Assumption 3.3. At least one of the following two conditions holds:

(a) From any initial state, the Markov chain with transition kernel p reaches a terminal state in finite time with probability one, and the sum of absolute values of rewards of all transitions until termination has finite expectation.

(b) We have $\gamma < 1$, and r is uniformly bounded in absolute value.

Our arguments will proceed by showing that (b) reduces to (a), then studying that case. It is also possible to work in slightly greater generality: we adopt the formulation here to minimize technicalities while keeping results sufficiently general.

Compared to the situation of Pandora's box, it is, at first, even less obvious whether anything can be said about the defined MDP's optimal policy—particularly given that the MDP is much less concrete than before, and its abstract nature could potentially give rise to rather different looking examples.

The insight of Gittins [44]—which is particularly remarkable given it was first discovered in essentially the same generality we consider here—is that this class of MDPs can be solved in much the same manner as we presented Pandora's box in Section 2. Namely, the idea will be to look for a way to compare Markov chains with each other, just as before we looked for a way to compare Pandora's boxes with each other. Specifically, we seek a way compare Markov chains, which are stochastic, with real numbers, which are not.

3.1.1. A note on the name *Markov chain selection* We conclude with a note on terminology. What we call MCS is, roughly speaking, usually called the *Markovian multi-armed bandit problem* in the Gittins index literature. We introduce the new name *Markov chain selection* for two reasons. First, the Markovian multi-armed bandit is typically defined slightly more restrictively, namely ruling out undiscounted settings and terminal states [45], and we want to emphasize that we do not impose these restrictions.

Second, we wish to slightly *de-emphasize* the link between Gittins indices and multi-armed bandits. The broader multi-armed bandit literature is vast [71, 97], and in the context of this vast literature, the Gittins index might only seem useful for solving one corner case, namely discounted Bayesian bandits (Section 4.2). On the other hand, in our opinion, *many applications of the Gittins index do not superficially resemble bandit problems*. The feature that unifies Gittins index applications—Pandora's box, bandits, queue scheduling, and more—is repeatedly choosing which of multiple independent Markov chains to advance. Our hope is that the *Markov chain selection* name directly evokes this unifying feature.

3.2. Defining the Gittins index via the local MDP

Mirroring the approach used in Pandora's box—namely, defining the Gittins index of a single closed box—we now define the Gittins index of a general Markov chain equipped with a general reward function. We do so using the following notion.

Definition 3.4 (Local MDP)**.** Let $(S, \partial S, p, r)$ be a Markov chain satisfying Assumption 3.3. For every *alternative option* $\alpha \in \mathbb{R}$ and *initial state* $s \in S$, define a Markov decision process, called the (s, α)-*local MDP*, as follows:

1. State space: let $S_{\text{loc}} = S \cup \{\checkmark\}$,[4] with initial state s.
2. Terminal state: $\partial S_{\text{loc}} = \{\checkmark\}$.
3. Action space: let $A_{\text{loc}} = \{\triangleright, \square\}$, called *go* and *stop*, respectively.
4. Reward function: for $s \in S$, let $r_{\text{loc}}(s, \triangleright) = r(s)$, $r_{\text{loc}}(s, \square) = \alpha$, and $r_{\text{loc}}(\checkmark, \triangleright) = r_{\text{loc}}(\checkmark, \square) = 0$.
5. Transition kernel: if $s \in S$ and $a = \triangleright$, then let $s' \sim p(s)$, otherwise if $s = \checkmark$ or $a = \square$ let $s' = \checkmark$.
6. Discount factor: let $\gamma \in (0, 1]$.

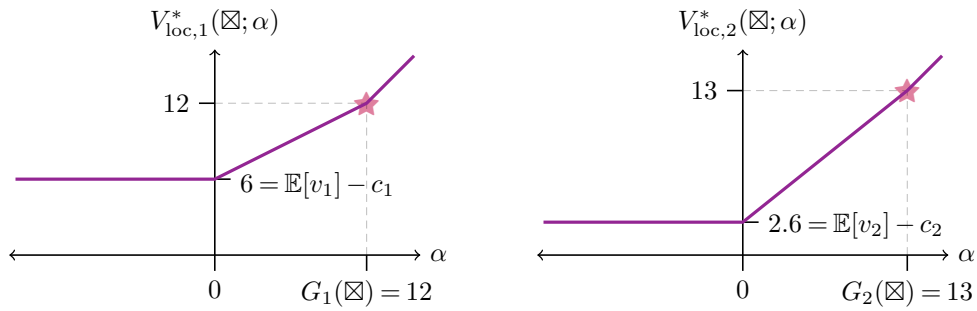
The intuition behind the local MDP is that it simplifies a *global* MCS instance down to the *local* perspective of one Markov chain. Specifically, it captures the tradeoff between advancing the Markov chain in state s vs. taking some other action. While MCS has are many other actions to choose from, the local MDP simplifies the tradeoff by providing just one other action $\square$ with a very clear value α. This generalizes the idea from Section 2.3 of comparing a closed Pandora's box with an open box—that is, comparing whether to transition the respective Markov chain one state forward, or simply take the value α from an alternative open box.

To make this precise, let $V^*_{\text{loc}}(s; \alpha)$ be the value function of the local MDP, which is well defined by Assumption 3.3. In the *undiscounted* setting, we can write

$$V^*_{\text{loc}}(s; \alpha) = \sup_{\pi : S_{\text{loc}} \to A_{\text{loc}}} (E_\pi(s) + \alpha P_\pi(s)) \tag{3.4}$$

where $E_\pi(s)$ is the expected total reward π receives while playing $\triangleright$ when starting from s, and $P_\pi(s)$ is the probability π eventually plays $\square$ when starting from s. In the *discounted* setting, we can still write (3.4), but $E_\pi(s)$ is discounted reward, and $P_\pi(s) = \mathbb{E}[\gamma^{t_\pi}]$, where t_π is either the time when π plays $\square$, or ∞ if $\square$ is never played. Figure 3.2 gives an illustration of the local MDP's value function for the closed boxes from Figure 2.1.

[4] Throughout this work, we adopt the convention that unions with symbols such as $\checkmark$ are disjoint unions. That is, it is understood that the union $S \cup \{\checkmark\}$ uses a symbol $\checkmark \notin S$ which is not part of the original states.



(a) Local MDP value for box 1 from Figure 2.1. (b) Local MDP value for box 2 from Figure 2.1.

FIGURE 3.2. Optimal value of the $(\boxtimes, \alpha)$-local MDP for the two closed boxes in Figure 2.1. We include the box identifier in subscripts throughout for disambiguation. For sufficiently small α, namely $\alpha \le 0$, the optimal policy opens the box and takes its reward, meaning it plays $\triangleright$ twice, yielding value $\mathbb{E}[v_i] - c_i$. For intermediate values of α, the optimal policy opens the box but might then take the alternative, meaning it plays $\triangleright$ at least once. The value's slope in this regime is the probability of eventually taking the alternative, namely (a) $\frac{1}{2}$ or (b) $\frac{4}{5}$. For sufficiently large α, the optimal policy takes the alternative without opening the box, meaning it plays $\square$. The value of α at the boundary between the latter two regimes—that is. the unique value at which both $\triangleright$ and $\square$ are optimal—is the Gittins index $G_i(\boxtimes)$.

The general definition of the Gittins index of a state s similarly generalizes the Pandora's box Gittins index definition from Section 2.3. The definition follows from two key observations about the local MDP with initial state s and varying alternative:

1. If playing $\square$ is optimal when the alternative is α, then $\square$ is still optimal for any *better* alternative $\alpha' > \alpha$.
2. If playing $\triangleright$ is optimal when the alternative is α, then $\triangleright$ is still optimal for any *worse* alternative $\alpha' < \alpha$.

One can prove this using the fact that $\square$ is optimal if and only if $V^*_{\mathrm{loc}}(s;\alpha) = \alpha$ and the following properties of V^*_{loc}.

Lemma 3.5. *For any state s of any Markov chain satisfying Assumption 3.3, the function $\alpha \mapsto V^*_{\mathrm{loc}}(s;\alpha)$ has the following properties:*

(a) It is convex and non-decreasing.

(b) It has left and right derivatives taking values in $[0,1]$.

(c) It is bounded below by $V^*_{\mathrm{loc}}(s;\alpha) \ge \alpha$.

Proof. We argue as follows:

(a) By (3.4), $\alpha \mapsto V^*_{\mathrm{loc}}(s;\alpha)$ is a supremum of convex (namely, affine) and non-decreasing functions, so it is also convex non-decreasing.

(b) This follows from (3.4), the convexity of $\alpha \mapsto V^*_{\mathrm{loc}}(s;\alpha)$, the fact that $P_\pi(s) \in [0,1]$, and a standard envelope theorem [78, Theorem 1]. See Xie et al. [114, Appendix B.6] for additional discussion about envelope theorems in this setting.

(c) Playing $\square$ immediately yields value α, and the optimal policy does at least as well. $\square$

Using these properties, one can show that there is a unique alternative α such that *both* $\square$ and $\triangleright$ are optimal, namely the minimum value of α such that $\square$ is optimal. We define the Gittins index $G(s)$ of s to be this transition point. Figure 3.2 illustrates an example.

Definition 3.6 (Gittins index)**.** Let $(S, \partial S, p, r)$ be a Markov chain satisfying Assumption 3.3. The *Gittins index function* for the Markov chain, denoted $G : S \to \mathbb{R} \cup \{\infty\}$, maps each state

$s \in S$ to either the unique number $g \in \mathbb{R}$ such that both $\triangleright$ and $\square$ are optimal actions for the (g, s)-local MDP at its initial state, or to ∞ if no such number exists.[5]

We call $G(s)$ the *Gittins index of state* s, and there are many equivalent ways to formulate it, for instance using stopping times. In particular, one can define the Gittins index to be the supremum of alternative values for which $\triangleright$ is strictly optimal, or the minimum value of g such that playing $\square$—which we note results in value g—is optimal:

$$G(s) = \sup\{g \in \mathbb{R} : V^*_{\mathrm{loc}}(s; g) > g\} = \inf\{g \in \mathbb{R} : V^*_{\mathrm{loc}}(s; g) = g\} \tag{3.5}$$

with the convention that infima and suprema of empty sets are ∞. One can similarly view $G(s)$ as the solution to a root finding problem, as we did for Pandora's box in (2.15): letting $V^{\triangleright}_{\mathrm{loc}}(s; g)$ be the optimal value achievable when playing $\triangleright$ at least once, we have

$$G(s) = V^{\triangleright}_{\mathrm{loc}}(s; G(s)). \tag{3.6}$$

There are other ways to express $G(s)$, particularly in terms of optimization problems over stopping policies, or equivalently, stopping sets. We refer the interested reader to prior expositions of the Gittins index for details [25, 45, 50, 117].

When needed for disambiguation between multiple Markov chains $(S_i, \partial S_i, p_i, r_i)$, we add a subscript i as needed, for instance G_i and $V^*_{\mathrm{loc},i}$, as was used in Figure 3.2.

3.3. Optimality of the Gittins policy

We now state the key optimality theorem for Definition 3.6, deferring its proof to Appendix A.

Theorem 3.7. *Consider an instance of MCS (Definition 3.2) with all Markov chains satisfying Assumption 3.3. A policy for MCS is optimal if and only if it always selects an action of maximal Gittins index—meaning, if when in state $(s_1, \dots, s_n)$, it selects*

$$a \in \operatorname*{arg\,max}_{i \in \{1,\dots,n\}} G_i(s_i). \tag{3.7}$$

In general, there may be several such policies, parameterized by an appropriately defined tie-breaking rule which chooses a maximizer in the event it is non-unique. We will implicitly assume that a tie-breaking rule has been chosen, and will refer to any policy satisfying the condition in Theorem 3.7 as *the Gittins policy*. We can summarize Theorem 3.7 as follows:

($\star$) *Always choose the Markov chain of greatest Gittins index.*

One can view this not just as a policy for MCS, but as a general design principle. If we can compare an action to a deterministic alternative in the style of the local MDP, then we can usually define the action's Gittins index just as in Definition 3.6. For example, in Section 5.1, we explain how this design principle applies to Bayesian optimization, a problem which may at first appear dissimilar to MCS.

In the situation formalized here—and a small set of generalizations, some of which we discuss in further detail in Section 4—this decision-making principle is outright optimal. However, the specific statement in Theorem 3.7 is rather fragile. It is not uncommon, in more general situations, for optimality to fail, and do so in a manner that suggests the proof technique breaks down completely.

In such situations, it be very tempting to conclude that this is because Gittins indices are not a good approach. However, there are cases where the definition continues to make intuitive sense, and one can either (a) show various notions of near-optimality, such as regret

[5] Under Assumption 3.3, it is always possible to pass to the setting where $G(s)$ is necessarily finite: one can replace the given MCS instance with a modified MCS instance over a slightly smaller state space in a manner that preserves value functions and policies. See Assumption A.2(b) and the following discussion for details.

[36, 69] or approximation ratio [27, 43, 87] bounds, or (b) show strong empirical performance [114]. In such cases, it can be more insightful to instead interpret the lack of an optimality theorem as a *statement about the problem's richness*, rather than of Gittins-index-style stochastic-to-deterministic comparisons being the wrong decision-making approach.

Understanding where the Gittins policy is strong, even if not outright optimal, remains an active research area. In Section 5, we cover some examples where the Gittins policy is suboptimal but nevertheless has strong theoretical or empirical performance.

3.4. Computation

We now discuss computational properties of the Gittins index. This problem is well studied for finite-state Markov chains: Chakravorty and Mahajan [25] give a survey of the topic. The current state-of-the-art algorithm is that of Gast et al. [41], which runs in sub-cubic time. We note also that even faster algorithms are possible under additional structure: see for instance Scully et al. [91, Section 4.2 and Appendix B].

For infinite-state Markov chains, computation remains a significant challenge [36, 45, 64, 65]. Continuous state spaces have received comparatively little attention beyond Pandora's box, and we view computing the Gittins index of continuous-state Markov chains to be a significant open problem: we return to this in Section 6.2. We now discuss the key challenges for doing so.

As expressed by (3.6), computing the Gittins index $G(s)$ of a given Markov chain state s requires one to solve a root-finding problem defined in terms of the local MDP's value function V^*_{loc}. As such, one can expect computation of $G(s)$ in a given concrete setting to potentially involve elements of dynamic programming, together with root-finding algorithms. This gives rise to two challenges:

1. For continuous-state Markov chains, one must usually perform dynamic programming approximately [10].
2. The local MDP must be solved for enough different values of the parameter α to determine the Gittins index.

In some classical problems—for instance, Gaussian Pandora's box—one can compute V^*_{loc} analytically. In such cases, owing to monotonicity of $\alpha \mapsto V^*_{\mathrm{loc}}(s;\alpha)$, the Gittins index can be computed efficiently using bisection search.

Even when V^*_{loc} cannot be computed analytically, we suspect one can do better than simply apply off-the-shelf approximate dynamic programming algorithms. This is because the local MDP possesses two properties which generic MDPs do not: its action space $\{\square, \triangleright\}$ is very small, and the value of $\square$ is always g. At present, methods that leverage these properties have largely yet to be developed. The key challenge is in effectively handling the state space S in situations where it is sufficiently high-dimensional to render discretization unviable.

Finally, there are extensions of MCS where there are infinitely many Markov chains, possibly uncountably many. This introduces another obstacle: finding the chain of maximum Gittins index. In general, this needs to be performed using gradient-based optimization. We discuss how to do so in the context of Bayesian optimization in Section 5.1.2.

4. Examples: optimal policies

We now work through a number of examples. We begin with a multi-stage Pandora's box, as a simple illustration of how the developed framework allows us to handle a slight generalization of our initial illustrative setup (Section 4.1). We then follow up with a discounted Bayesian bandit, illustrating how discounted chains without terminal states are handled (Section 4.2). Finally, we discuss three mild generalizations of the developed framework: (a) a variant which models Markovian search for multiple items, which therefore involves termination of more than one Markov chain (Section 4.3); (b) a queueing example with a seemingly different objective, but which is handled similarly to the aforementioned Markovian search

(Section 4.4); and (c) a branching bandit, where the action space at each time point varies stochastically in a manner that can depend on the chosen actions (Section 4.5). In all cases, an optimality result in the spirit of Theorem 3.7 continues to hold.

4.1. Two-stage Pandora's box

Consider the following variant of Pandora's box where there are two stages of inspection for a box.

1. The first stage reveals some partial information about the box's contents, which we call its *label*. This inspection incurs some cost.
2. The second stage opens the box and reveals its reward, just like in ordinary Pandora's box. This incurs additional cost.
3. Finally, we can select a box and receive its reward once it is fully open.

To model a two-stage box as a Markov chain, we use much the same approach as for ordinary Pandora's box in Figure 3.1, but with additional states representing the label. Specifically, letting L be the set of labels, assumed disjoint from the remaining states, define

$$S_{\text{2-PB}} = \{\boxtimes\} \cup L \cup \mathbb{R} \cup \{\checkmark\}. \tag{4.1}$$

The transition kernel $p_{\text{2-PB}} : S \to \mathcal{P}(S)$ described above then has the following form.

1. For the first stage, we always transition from $\boxtimes$ to some label $\ell \in L$. We thus write $p_{\text{2-PB}}(\boxtimes) \in \mathcal{P}(L)$ as the distribution over labels.
2. For the second stage, when in state $\ell \in L$, we transition to a state $v \in \mathbb{R}$ by sampling $v \sim p_{\text{2-PB}}(\ell)$. Here, $p_{\text{2-PB}}(\ell)$ is the reward distribution of a box given that its label is ℓ.
3. Finally, from any fully inspected state $v \in \mathbb{R}$, we always transition to the selected state $\checkmark$.

This encodes how information is revealed as part of the inspection process. Similarly, the reward function $r : S \to \mathbb{R}$ encodes the inspection costs.

1. If the cost of the first inspection stage is $c(\boxtimes)$, then $r(\boxtimes) = -c(\boxtimes)$.
2. If, given a label ℓ, the cost of the second stage is $c(\ell)$, then $r(\ell) = -c(\ell)$.
3. Finally, for any fully inspected state $v \in \mathbb{R}$, the reward is simply $r(v) = v$.

What does the Gittins index look like for this two-stage box? Following Theorem 3.7, to compute the Gittins index $G(s)$ of a state s, we need to understand how the local MDP's value function $V^*_{\text{loc}}(s;\alpha)$ depends on α. Specifically, the point where $\square$ and $\triangleright$ are co-optimal is by definition also the smallest value of α for which $V^*_{\text{loc}}(s;\alpha) = \alpha$.

To obtain the value function, we apply dynamic programming to the local MDP's, which is tractable because there are only two possible actions. Working backwards from the terminal state $\checkmark$, we find:

3. For fully inspected states $v \in \mathbb{R}$, we clearly have
$$V^*_{\text{loc}}(v;\alpha) = \max(v, \alpha). \tag{4.2}$$
So, as in the one-stage Pandora's box, co-optimality implies $G(v) = v$.
2. For partially inspected states $\ell \in L$, we essentially have the same situation as for ordinary Pandora's box:
$$V^*_{\text{loc}}(\ell;\alpha) = \max\big(\mathbb{E}_{v\sim p(\ell)}[V^*_{\text{loc}}(v;\alpha)] - c(\ell), \alpha\big) \tag{4.3}$$
$$= \max\big(\mathbb{E}_{v\sim p(\ell)}[\max(v,\alpha)] - c(\ell), \alpha\big). \tag{4.4}$$
So computing $G(\ell)$ amounts to the same root-finding problem as in (2.15), namely
$$\mathbb{E}_{v\sim p(\ell)}[\max(v - G(\ell), 0)] - c(\ell) = 0. \tag{4.5}$$

1. Finally, for the initial uninspected state $\boxtimes$, we have

$$V^*_{\text{loc}}(\boxtimes;\alpha) = \max\big(\mathbb{E}_{\ell\sim p(\boxtimes)}[V^*_{\text{loc}}(\ell;\alpha)] - c(\boxtimes), \alpha\big) \tag{4.6}$$

which yields a second kind of root-finding problem for $G(\boxtimes)$, namely

$$\mathbb{E}_{\ell\sim p(\boxtimes)}[V^*_{\text{loc}}(\ell;\alpha) - \alpha] - c(\boxtimes) = 0. \tag{4.7}$$

The Gittins index for this problem can therefore be expressed as a solution to a sequence of root-finding problems. One can generalize this to more than two stages while maintaining a similar structure. In general, following Section 3.4, one can expect that computing G will require numerical methods—albeit ones which involve each local MDP individually, rather than the full Markov chain selection MDP.

4.2. Bayesian Bernoulli bandits with discounting

In the *Bernoulli multi-armed bandit problem*, an agent is presented with n coins, where each coin i has unknown heads probability x_i. Our goal, roughly, is to repeatedly flip coins in a way that maximizes the expected (discounted) number of heads. In the *Bayesian* version of the problem, the agent has a prior distribution on x_i. This problem fits into the framework of Definition 3.2 with discount parameter $\gamma < 1$: each Markov chain corresponds to one coin, advancing the Markov chain corresponds to flipping the coin: the Markov chain's state represents the current posterior distribution the agent has on x_i.

We focus on the traditional beta-distributed prior $x_i \sim \text{Beta}(a_i, b_i)$ for some $a_i, b_i > 0$. In this case, by standard properties of the beta distribution, we can express each coin i as a Markov chain as follows:

1. The state space is $S = (0,\infty)^2$, with an initial state is (a_i, b_i).
2. The transition kernel is

$$p(a,b) = \begin{cases} (a+1,b) & \text{w.p. } \frac{a}{a+b} \\ (a,b+1) & \text{w.p. } \frac{b}{a+b}. \end{cases} \tag{4.8}$$

3. The reward function is $r(a,b) = \frac{a}{a+b}$.

Definition 3.2, with Markov chains of the above, is therefore our Bayesian Bernoulli bandit of interest. To compute the Gittins index, we need to solve the respective local MDP of Definition 3.4 for the above Markov chain. However, because the Markov chain has a countably infinite state space and no terminal states, computing the Gittins index is nontrivial, though several practical approaches for approximating or bounding it are known: see Farias and Gutin [36], Gittins et al. [45], Kelly [64], Kim and Lim [65] for examples.

4.3. Selecting multiple boxes, or finishing multiple Markov chains

One can view Pandora's box, and more generally MCS with terminating Markov chains, as a model of searching for a single item. What if one is instead searching for multiple items? It turns out that in the undiscounted setting—that is, with $\gamma = 1$—the Gittins policy optimally solves this version of the problem, too. Specifically, for any k, the Gittins policy maximizes the total expected reward received up until the point the first k Markov chains reach terminal states. We now state this more formally.

Definition 4.1. The *k-finish Markov chain selection (MCS-k)* problem for $k \le n$ Markov chains is an MDP defined in the same way as Definition 3.2, except the process does not end until k Markov chains reach terminal states, meaning the terminal states are

$$\partial S_{\text{MCS-}k} = \{(s_1,\dots,s_n) \in S_{\text{MCS}} : s_i \in \partial S_i \text{ for } k \text{ distinct } i\}. \tag{4.9}$$

Additionally, action i is disallowed if Markov chain i is in a terminal state, meaning this MDP has a state-dependent action space, which is defined to be

$$A_{\text{MCS-}k}(s_1, \dots, s_n) = \{i : s_i \notin \partial S_i\}. \tag{4.10}$$

The case $k = 1$ reduces to Definition 3.2. The following result shows Theorem 3.7 generalizes to $k \geq 2$. Moreover, the proof is very similar: see Appendix A.2.

Theorem 4.2. *The Gittins policy is optimal for undiscounted MCS-k.*

In fact, one can use the Gittins index to solve an even more general problem than MCS-k involving *combinatorial constraints*. For example, consider the following problem introduced by Singla [96], which we call *spanning-tree Pandora's box*. We are given a graph, where each edge of the graph is associated with a box with some opening cost and reward distribution. The problem proceeds much like the k-finish variant of Pandora's box, except the set of boxes we take must contain no cycles. One can view this as a max-weight spanning tree variant where we replace deterministic edge weights with Pandora's boxes.

The classical max-weight spanning tree problem is famously optimally solvable by greedy algorithms. The simplest of these, arguably, is Kruskal's algorithm, which accumulates a spanning tree by repeatedly adding the edge of greatest weight that would not form a cycle. Singla [96] shows that essentially the same algorithm solves spanning-tree Pandora's box if one uses *Gittins indices in place of deterministic edge weights*. That is, at every time step, we take the action corresponding to the box of greatest Gittins index, provided that box's edge would not form a cycle with selected edges.

Moreover, there is nothing special about Pandora's box here, and indeed, Gupta et al. [55] show that essentially the same algorithm solves the version of the problem where each edge has an arbitrary (terminating) Markov chain. For example, one could use the two-stage Pandora's box variant of Section 4.1 to model max-weight spanning tree problems where each edge requires two rounds of inspection to reveal its value—perhaps an inexpensively acquired initial estimate followed by an expensively acquired precise valuation. The resulting strategy of taking a classic combinatorial algorithm and plugging in Gittins indices in place of deterministic weights is potentially very general.

In what situations does this strategy work? The answer, roughly speaking, is for *greedy algorithms* of a certain form identified by Singla [96]. For instance, one can optimally solve what we might call *matroid-finish MCS*, because matroid packing problems—of which max-weight spanning tree is a special case—are solved by greedy algorithms. Moreover, for situations where greedy algorithms only give approximately optimal solutions, Singla [96] (for Pandora's box) and Gupta et al. [55] (for general Markov chains) show that the Gittins index version of the greedy algorithm achieves the same approximation ratio as the algorithm would achieve in the classical deterministic-weight setting. Thus, one can view Gittins indices as providing a good definition for what *greedy* should mean in stochastic contexts.

4.4. Scheduling in queues

Another famous application of the Gittins policy is scheduling in single-server queues, particularly the M/G/1 queue [1, 3, 11, 13, 67, 68, 90, 94, 95, 103, 112]. Here the problem is to schedule jobs that arrive over time on a single server in order to minimize their *mean latency*—also called their response time or sojourn time—which is the mean amount of time between a job arrives and when it completes.

In this section, we give a brief overview of how to formulate job scheduling in terms of MCS. We first explain the arrival-free *batch setting*, then explain how the theory extends to handle (time-homogeneous) *Poisson arrivals*. For simplicity, we focus throughout on the unweighted case. See Scully and Harchol-Balter [90] for an account of most of the features the Gittins policy is known to handle in the M/G/1, including unknown and state-varying job weights, and Glazebrook [49] for an additional discussion of priority constraints.

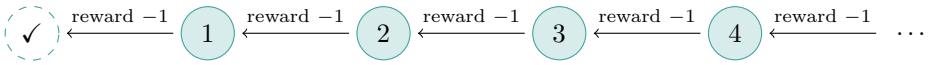


FIGURE 4.1. Markov chain of a job with *known* service time. The job's state is its *remaining service*, namely how many more time units of service until it completes—meaning, transitions to the finished state ✓. Every transition yields reward −1, representing one unit of time passing. The transitions are all deterministic, and one can confirm that ✓ is reached.

4.4.1. Representing batch scheduling as MCS Broadly, (single-server, discrete-time) *batch scheduling* is a family of problems where an agent has n *jobs* to complete using a single server. Every time step, the agent chooses one job to receive one time unit of service. Each job requires a (strictly positive) number of time units at the server to complete, which we call the job's *service time*. As we discuss below, these service times might be known, unknown, or partially known to the agent. The agent's goal is to optimize some metric related to the jobs' *completion times*, where a job's completion time is the time step when it completes.[6] We focus on minimizing the expected values of the following three metrics:

1. The *earliest completion time* is the minimum of all jobs' completion times.
2. The *kth completion time* is the kth least among all jobs' completion times. First completion time is then the special case where $k = 1$.
3. The *total completion time* is the sum of all jobs' completion times. An equivalent metric is *mean completion time*, which is total completion time times $1/n$. These are the metrics most closely related to minimizing mean latency in queues with arrivals.

In the above, we stated that service times might be known, unknown, or partially known. Specifically, we assume that *each job's service can be modeled as a Markov chain*, with different jobs' Markov chains evolving independently. A job's service time is then the number of transitions it takes to reach a *completed* state, which we denote by ✓. The appropriate Markov chain for each job depends on what the agent knows about that job's service time. For example:

a. For jobs with known service time, we use the Markov chain in Figure 4.1. A job's state here is its *remaining service time*. With each unit of service, the remaining service time decrements by 1, with completed state ✓ taking the place of 0 remaining service time. In this case, the Gittins index of a job is simply its (negative) remaining service time

$$G(s) = -s \tag{4.11}$$

and the Gittins policy reduces to (discretized) shortest remaining processing time [85].

b. For jobs with unknown service time sampled from some known distribution m, we use the Markov chain in Figure 4.2. A job's state here is its *attained service*. With each unit of service, the attained service typically increments by 1, but there is a chance to transition to the completed state ✓. In this case, the Gittins index can be written [1, 2]

$$G(s) = -\inf_{s'>s} \frac{\mathbb{E}_{t\sim m}[\min(t, s') - s \mid v > s]}{\mathbb{P}_{t\sim m}[t \le s' \mid t > s]} \tag{4.12}$$

where $t \sim m$ is service time of the job, namely the random number of steps it takes to reach ✓ starting from 0. The intuition is that s' represents a deadline by which we might hope the job will finish, the fraction is the *mean time per completion* ratio for deadline s', and $G(s)$ is the (negated) best—namely, least—such ratio achievable.

[6] To be precise: letting the first time step have index 1, a job's completion time is the index of the time step during which it receives its last unit of service. For example, if a job with service time t is served starting at time 1 until it completes, then it receives service at times $1, \dots, t$ and has completion time t.

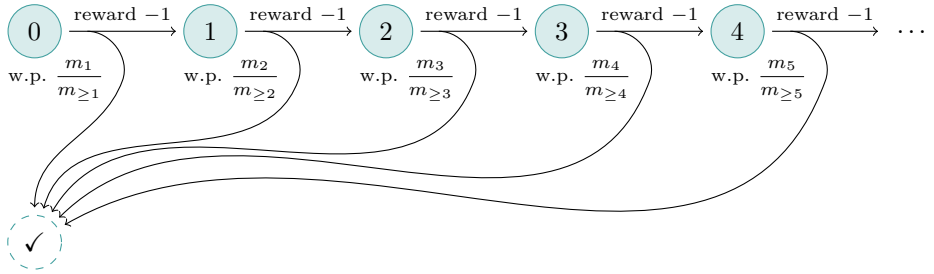


FIGURE 4.2. Markov chain of a job with *unknown* service time sampled from distribution m—that is, the job's service time is t with probability m_t, and it is at least t with probability $m_{\geq t} = \sum_{u=t}^{\infty} m_u$. The job's state is its *attained service*, namely how many time units of service it has already received. Every transition yields reward -1, representing one unit of time passing. The transition probabilities come from the fact that if the job is in state s, then because the job has not yet completed, its service time must be at least $s+1$. Given this, a job in state s completes within its next unit of service with probability $m_{s+1}/m_{\geq s+1}$.

c. An intermediate case where the agent receives partial information about a job's service time is illustrated in Figure 4.3. Here a job's service has two stages, and while the agent does not know a priori how long either stage will take, the agent is notified when the job advances from the first stage to the second.

Throughout, we give all states reward -1 to represent the fact that one unit of time passes per transition—though, as we discuss below, this is not essential. The three examples presented here generalize easily to a large number of possible Markov chain variants.

Having specified the job model we are working with, we can observe the following about the different metrics we hope to optimize. The main takeaway is that *all the problems are variants of undiscounted MCS*—and, thus, they can be solved with the Gittins policy.

i. Minimizing expected earliest completion time is MCS (Definition 3.2), because we receive reward -1 per time step until a job completes. Therefore, by Theorem 3.7, the Gittins policy minimizes expected earliest completion time.

ii. Minimizing expected kth completion time is MCS-k (Definition 4.1), because we receive reward -1 per time step until k jobs complete. Therefore, by Theorem 4.2, the Gittins policy minimizes expected kth completion time.

iii. Minimizing expected total completion time does not directly fit into MCS or MCS-k. However, we can express total completion time C as the sum of kth completion times $C_{(k)}$, namely

$$C = \sum_{k=1}^{n} C_{(k)}. \tag{4.13}$$

Because the Gittins policy minimizes $\mathbb{E}[C_{(k)}]$ for all k, it also minimizes $\mathbb{E}[C]$.

Before discussing adding arrivals, let us briefly revisit one of our assumptions, namely that every transition incurs cost 1 (and thus yields reward -1). The MCS framework allows different costs in different states, so the entire discussion above generalizes to jobs where different transitions incur different costs. One can interpret this as scheduling jobs where *different transitions take different amounts of time*. In particular, transitions that take a long time can be viewed as *uninterruptible segments*. That is, one can represent an interruptible part of a job using many low-cost transitions, and one can represent a uninterruptible part using a single high-cost transition.

4.4.2. From batch scheduling to queue scheduling Broadly, a (single-server, discrete-time) *queue scheduling* problem is a batch scheduling problem with the following changes:

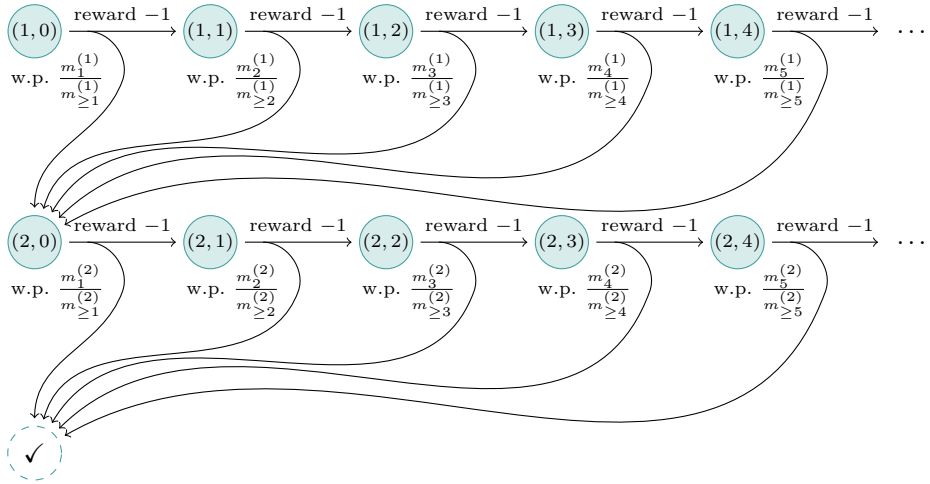


FIGURE 4.3. Markov chain of a job with *two stages* of service. Each stage i requires an unknown amount of service sampled from distribution $m^{(i)}$, so the service time is a priori unknown, but the agent learns when the transition from the first to the second stage occurs. A job's state is thus a pair consisting of the current stage and the attained service of the current stage. The result is, roughly speaking, two stacked instances of the Markov chain from Figure 4.2.

1. Instead of n jobs being present at time 0, jobs arrive over time. We usually consider an infinite sequence jobs arriving according to some stochastic process.
2. Instead of optimizing metrics related to completion time, we optimize metrics related to *latency* (also called response time or sojourn time), where a job's latency is its completion time minus its arrival time. We usually consider metrics that are (limiting) averages over the sequence of arriving jobs, such as mean latency.

If new Markov chains can appear over time in an arbitrary fashion, then the Gittins policy is no longer optimal for MCS, let alone MCS-k. However, there is a more general case where the Gittins policy remains optimal: when arrivals are generated by a *memoryless and time-homogeneous* stochastic process [110]. This means *Poisson arrival times*, where each arriving job's initial state is drawn i.i.d. from some initial state distribution (or a similar arrival process—see Remark 4.5). In the language of queueing theory, this is the *M/G/1* queueing model, which allows Poisson arrival times and general service time distributions. We now illustrate one type of result that can be shown, though we emphasize that it is not the most general result possible [49, 90].

Definition 4.3. A *Markov chain M/G/1* model consists of:

1. A *job Markov chain* $(S, \partial S, p, r)$ such that $r(s) < 0$ for all non-terminal states $s \in S \setminus \partial S$.
2. An *initial state distribution* $p_0 \in \mathcal{P}(S \setminus \partial S)$.
3. An *arrival rate* $\lambda > 0$.

Here, jobs arrive at the increments of a Poisson process of rate λ, and each job's initial state is sampled i.i.d. from p_0. Each job's Markov chain advances while it is in service. It takes $-r(s)$ time for a job to transition from state s to its next state (possibly with some randomness—see Remark 4.5), and a job cannot be interrupted during a state transition. A *scheduling policy* for the Markov chain M/G/1 repeatedly chooses which job to serve next, at which point the job remains in service until it has completed a state transition. We also allow the scheduling policy to leave the server idle for a time.

We say the M/G/1 is *stable* if the expected interarrival time—that is, $1/\lambda$—is greater than the expected service time—that is, the amount of time it takes a job to transition from an

initial state drawn from p_0 to a terminal state in ∂S. This ensures ergodicity of the system under any (non-idling) scheduling policy, in which case the following optimality statement holds for the Gittins policy.

Theorem 4.4. *In any stable Markov chain M/G/1, among all scheduling policies, the Gittins policy minimizes mean latency.*

Unfortunately, to the best of our knowledge, there are no proofs of Theorem 4.4 that build directly on the techniques used to prove Theorems 3.7 and 4.2. Proofs using a *vanishing discount* approach—see Gittins et al. [45, Chapter 3]—come the closest, but they require restrictive assumptions, such as for instance the job Markov chain having a finite state space. Thus far, the approaches that yield the most general results use *work conservation laws* or similar M/G/1-specific reasoning [13, 49, 90].

Remark 4.5. The optimality of the Gittins policy continues to hold under the following two types of extensions:

(a) One can generalize slightly beyond Poisson arrivals to other time-homogeneous arrival processes. The simplest case of this is *batch Poisson* arrivals, where at each arrival time, multiple jobs can arrive at once [90]. In this batch setting, the lists of initial states of jobs in a batch must be i.i.d. across batches. Another time-homogeneous case occurs if all jobs are time-slotted—meaning, if all jobs' transitions take an integer number of time unit—then one can work with arrival processes that are similarly time-slotted, even if they are not Poisson.

(b) One can generalize from deterministic to randomized transition times in job Markov chains, in which case $-r(s)$ should be the *expected* amount of time it takes to transition from s to the next state. In this more general model, instead of specifying a job via p and r, one specifies a kernel $q : S \to \mathcal{P}(S \times [0, \infty))$, where $q(s)$ is the joint distribution over (next state, transition time) pairs. Even though one (p, r) pair can arise from many possible kernels q, the Gittins index depends only on p and r, though the mean latency achieved is affected by q. This generalization is possible because it embeds into the continuous-time framework of [90].

Of course, the M/G/1 is a rather specialized queueing model, so it is natural to ask whether the Gittins policy performs strongly in more general settings—say, with more than one server or with non-Poisson arrivals. Thus far, it appears the answer is often *yes*: there are now several results showing the Gittins policy is in some sense approximately optimal, including the multiserver M/G/k [49, 51, 52, 86, 88] and, most recently, the non-Poisson G/G/1 and G/G/k [58]. Even under adversarial arrivals—in the sense of adversarially chosen arrival times and initial states, but where the job still evolves stochastically according to a known transition kernel during service—it is known that a variation of the the Gittins policy is a 2-approximation for mean *completion time* [76], meaning each job's clock starts at time 0. It is an open question whether a similar guarantee holds for mean *latency*, where each job's clock starts when it arrives.

4.5. Branching bandits

To arrive at a general formulation of the Gittins index, following our exposition of Pandora's box, we asked: *is there a general theory this solution is an example of?* In the same spirit, let us note that the Gittins index for the M/G/1 queue does not directly arise as an instance of MCS, and again ask the same question.

Compared to MCS, what is new about the M/G/1 setting is that every time an action is chosen, a new set of Markov chains arrives, according to the Poisson process and initial state distribution, and becomes part of the total action space. More precisely, when a Markov

chain in state s transitions to a new state $s' \sim p(s)$, it also gives rise to a random set of new Markov chains in some initial states drawn from a known distribution.

The appropriate abstract generalization of the M/G/1 setting is the so-called *branching bandit* setting of Weiss [107]. The branching bandit problem is like MCS, but it replaces each Markov chain with a particular kind of *branching process* (namely, a multi-type Galton–Watson process), defined in the following sense. When advancing a branching process, instead of its state s transitioning to exactly one next state, it transitions by replacing the current state with multiple new states according to a suitable probability kernel, which describes the joint distribution over the number of new states and their values. This means that the overall problem's state space is now described by tuples $(s_1, .., s_n)$ of variable length, where n varies according to the actions selected by the policy and the random transition outcomes.

Variants of the branching bandit problem have been studied in the discounted setting [45, 77, 107], as well as the undiscounted Pandora's box setting [19]: in these cases, one can define a Gittins index in an appropriate sense, and prove that the resulting Gittins policy is optimal. However, to the best of our knowledge, there is not yet a proof that holds at the level of generality of Theorem 3.7. For instance, Weiss [107] requires each Markov chain to have finitely many states and satisfy Assumption 3.3(b). In principle, a more general dynamic programming proof similar to the one of Appendix A should be feasible, but we are not aware of one.[7]

5. Examples: beyond optimality

We now discuss three examples where Gittins indices can be defined and applied, but do not result in an optimality proof. These are (a) certain forms of Bayesian optimization, where optimality fails due to the presence of correlations between different Markov chains (Section 5.1); (b) Pandora's box with optional inspection, where additional decisions that can be made render the problem more complicated (Section 5.2); and (c) minimizing tail latency in queues, where one seeks to perform well in terms of objectives beyond average rewards (Section 5.3).

5.1. Bayesian optimization

Bayesian optimization [37, 40] is a broad class of algorithms for global optimization of unknown functions which are expensive to evaluate. In most instances, such algorithms require only *black-box access* to the unknown function, meaning the only way to learn about the function is by evaluating it. Bayesian optimization is a workhorse tool in areas like machine learning hyperparameter tuning [98], where it is deployed in production at most major technology companies, and is available as standard functionality in popular artificial intelligence operations platforms.

Let $f : X \to \mathbb{R}$ be the unknown function, where X is allowed to be a general set, with $X = [0, 1]^d$ a typical choice. Bayesian optimization works by building a *probabilistic model* for the unknown function f, typically in a Bayesian manner by placing a Gaussian process prior over it. Function evaluations are incorporated into the probabilistic model using Bayes' Rule, by conditioning the prior on the location and value of previous function evaluations. With this setup, one aims to design a policy that adaptively chooses inputs $x_1, \dots, x_T$ in order to find the global optimum in as few function evaluations as possible.

There are multiple distinct criteria one use to can study performance of Bayesian optimization algorithms. The simplest, arguably, is expected *simple regret* with respect to the prior, namely

$$\mathbb{E}\left[\sup_{x \in X} f(x) - f_T^*\right] \tag{5.1}$$

[7] In the language of Section 5.2.2, we believe that one should be able to show that all branching processes satisfy the Whittle condition in an appropriate sense, and use this to construct an optimality argument.

where $f_t^* = \max_{u \in \{1,\dots,t\}} f(x_u)$ and the unknown function f is sampled from the prior. In this setting, the question of how to adaptively choose the next data point x_{t+1}, given a set of previous function evaluations $(x_1, f(x_1)), \dots, (x_t, f(x_t))$ can be formalized to define an MDP. Solving this MDP—and, indeed, almost all other MDPs which occur in Bayesian optimization contexts—is known to be intractable. However, by applying a one-step greedy approximation to this MDP's dynamic programming equations, one arrives at the *expected improvement acquisition function*

$$\mathrm{EI}_t(x) = \mathbb{E}[\max(f(x) - f_t^*, 0)]. \tag{5.2}$$

A very similar expected improvement formula above previously came up in Section 2. This is not a coincidence: consider a mild generalization known as *cost-aware Bayesian optimization*, specifically the *cost-per-sample* formulation, where one adds a sequence of costs $c(x_t) > 0$ to the above objective, and allows the algorithm to decide when to stop, as opposed to having T be a fixed hyperparameter. Note that that $\mathbb{E}[\sup_{x \in X} f(x)]$ is constant with respect to the policy. Using this, following Xie et al. [114], if we switch from minimizing regret to maximizing the negation of all terms, and drop the aforementioned constant, we obtain

$$\mathbb{E}\left[f_T^* - \sum_{t=1}^{T} c(x_t)\right] \tag{5.3}$$

which is the same as the objective of Pandora's box in (2.1). We therefore conclude:

($\star$) *Cost-per-sample Bayesian optimization, under the expected simple regret performance criterion, is a Pandora's box problem with correlations between boxes.*

Compared to the Pandora's box of Section 2, the crucial difference here is that there is now a potentially uncountable number of boxes, indexed by X, and rewards in different boxes are correlated. These correlations completely break the argument of Theorem 3.7, which no longer applies. On the other hand, the same correlations also make the optimal policy intractable: one can therefore ask whether the Gittins policy at least makes sense as a candidate to consider implementing in practice.

5.1.1. Defining a Gittins index for Bayesian optimization To proceed, we view cost-per-sample Bayesian optimization as a variant of MCS where whenever one Markov chain advances, *the transition kernels of all other Markov chains also update.* In Bayesian optimization, this transition kernel update occurs due to updating the posterior distribution of the unknown function f. The appropriate definition of a Gittins policy for this variant of MCS—and thereby for Bayesian optimization—is just like the Gittins policy for ordinary MCS, but where we make sure to *always use the updated transition kernel when defining Gittins indices.* We now make this precise.

Suppose we have already observed the values $f(x_1), \dots, f(x_t)$, resulting in a posterior distribution p_t for $f \mid f(x_1), \dots, f(x_t)$. Based on this posterior distribution, we define the *Gittins index of an input point* x in much the same way as the Gittins index of a Pandora's box in Section 2.3. Specifically, we think of input point x as corresponding to a box whose opening cost is $c(x)$ and whose value distribution is that of $f(x)$ for $f \sim p_t$: That is, we define the expected improvement function at time t to be

$$\mathrm{EI}_t(x, \alpha) = \mathbb{E}_{f \sim p_t}[\max(f(x) - \alpha, 0)] - c(x) \tag{5.4}$$

$$= (\mu_t(x) - \alpha)\Phi\left(\frac{\mu_t(x) - \alpha}{\sigma_t(x)}\right) + \sigma_t(x)\varphi\left(\frac{\mu_t(x) - \alpha}{\sigma_t(x)}\right) - c(x) \tag{5.5}$$

where μ_t and σ_t are the mean and standard deviation of the posterior Gaussian process, and φ and Φ are the standard Gaussian PDF and CDF, respectively. From this, we define the Gittins index function at time t to be G_t, where $G_t(x)$ solves the root-finding problem

$$\mathrm{EI}_t(x, G_t(x)) = 0. \tag{5.6}$$

Here, the expected improvement function in (5.2) is the special case of (5.4) where we always plug in $\alpha = f_t^*$. At time t, meaning after evaluating $f(x_1), \dots, f(x_t)$, we need to choose between one of the following actions.

a. Choose a next point x_{t+1} to evaluate. As described above, choosing to evaluate $x_{t+1} = x$ has Gittins index $G_t(x)$.

b. Choose to stop by setting $T = t$ and taking an observed value, namely $f_T^* = \max(f(x_1), \dots, f(x_T))$. This action yields reward f_T^* and ends the process—so, just as in ordinary Pandora's box, it has Gittins index f_T^*.

The *Gittins policy for Bayesian optimization*, also called the *Pandora's box Gittins index (PBGI)* by Xie et al. [114] to emphasize its connection with Pandora's box, is then the policy that always takes the action of maximum Gittins index.

One point of subtlety is that, even though (5.4) and (5.6) are essentially the same as their Pandora's box counterparts in (2.10) and (2.15), which involve no correlations, it is *not correct* to say that (5.4) and (5.6) ignore correlations. This is because they use the posterior distribution p_t, which accounts for correlations in its definition. Thus, a more accurate description would be to say that (5.4) and (5.6), in some sense, account for *correlations from the past*, but disregard *correlations in the future*.

5.1.2. Maximizing the Gittins index numerically Since the state space is infinite, maximizing $G_t(x)$ in order to select the next data point requires gradient-based optimization. In Bayesian optimization, this step is called *acquisition function optimization*, and is generally performed numerically using multi-start variants of either LBFGS or ADAM. The challenge now is that one needs to compute the gradient $\nabla G_t(x)$, which involves automatically differentiating through the root-finding problem. To avoid differentiating through the individual steps of bisection search or other root-finding algorithm, Xie et al. [114] show that $\nabla G_t(x)$ admits a particularly simple form, namely

$$\nabla G_t(x) = \nabla \mu(x) + \frac{\varphi\left(\frac{\mu_t(x) - G_t(x)}{\sigma_t(x)}\right) \nabla \sigma_t(x) - \nabla c(x)}{\Phi\left(\frac{\mu_t(x) - G_t(x)}{\sigma_t(x)}\right)}. \tag{5.7}$$

Using this, one can compute $G_t(x)$ numerically using bisection search, then plug the result in to (5.7) to obtain the gradient. This approach is an instance of a general principle used throughout the automatic differentiation literature [4, 18]: one can differentiate through the solution of a root-finding problem numerically by expressing the respective derivative in terms of the function defining the root-finding problem, together with the root.

5.1.3. Performance of the Gittins index in Bayesian optimization With an appropriate Gittins policy—which is not optimal—defined, the question becomes: is it strong? Empirically, at least for the kind of Gaussian processes which are used in Bayesian optimization benchmarking, the answer appears to be *yes*: Xie et al. [114] show that the Gittins policy either matches or outperforms most baselines—this is shown in Figure 5.1.

This connection appears to be new: to the best of our knowledge, it remained unnoticed until the recent work of Xie et al. [114], and prior to that, only Persky [84] had approached Bayesian optimization using a discounted-bandit variant of Gittins indices. One benefit the Gittins index perspective brings to Bayesian optimization is that Gittins indices naturally handle different input points having different function evaluation costs—an ongoing challenge in Bayesian optimization [72, 73, 114]—because the Pandora's box problem naturally allows different boxes to have different opening costs.

Developing a theory that characterizes the strengths and limitations of the Gittins policy's performance in Bayesian optimization—for instance, in the language of regret or approximation ratio bounds—is the subject of ongoing research. Recent results on Pandora's box

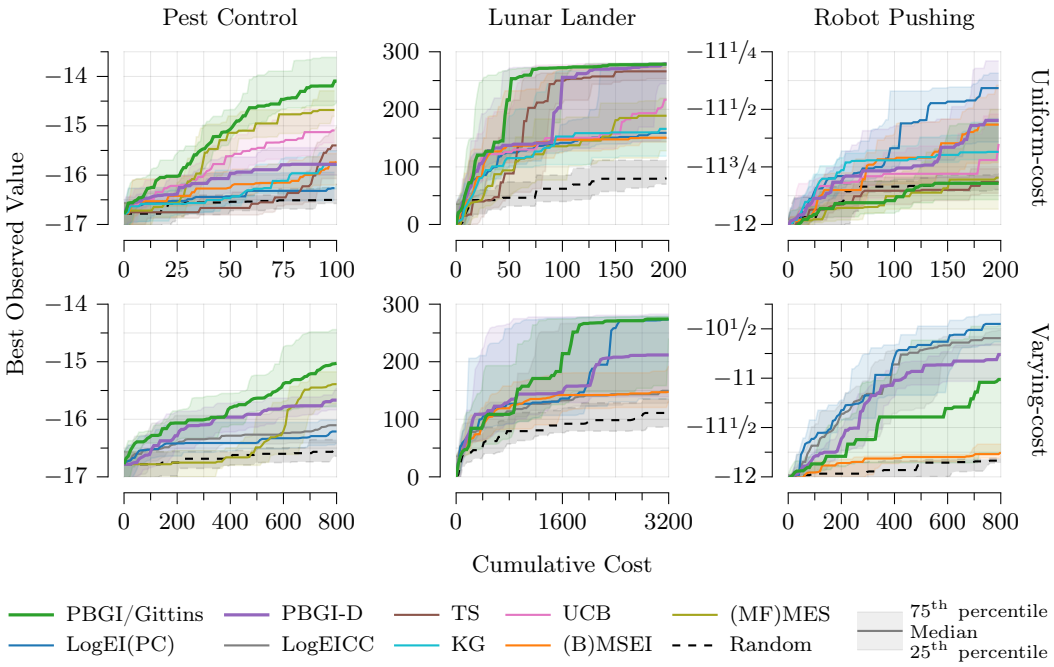


FIGURE 5.1. Results reproduced from Xie et al. [114]: empirical performance (higher is better) of the Gittins policy for Bayesian optimization, also called *PBGI* (green) in the legend, against other baseline policies, shown in terms of medians and quartiles over 16 seeds. The task here is to optimize benchmark objective functions, constructed to resemble real-world black-box optimization settings. We plot the best observed function value, in terms of medians and quartiles from 16 trials with different random initializations. We see that PBGI, along with a minor variant called PBGI-D (purple), claim the top-performing spot in the first two problems, and are reasonably competitive in the third. This holds for both $c(x)$ chosen to be a constant function, termed the *uniform-cost* setting, and $c(x)$ non-constant, termed the *varying-cost* setting.

with general joint value distributions [28, 43] may provide a good starting point, though it is likely that stronger guarantees might be possible when focusing on the multivariate Gaussian distributions that arise in typical Bayesian optimization priors.

Another rich future direction is developing versions of the Gittins policy for more advanced Bayesian optimization settings, such as *multi-fidelity optimization* for applications like hyperparameter tuning [35, 118]. These are problems where there are multiple actions one can take at any given input point—for instance, one can either fully evaluate the function, or obtain a cheap but noisy value estimate. Some such problems may share features of our next example: a Pandora's box variant with two actions available for closed boxes.

5.2. Pandora's box with optional inspection

A natural question about the Pandora's box problem is: what changes if one is allowed to select a closed box—without opening it first? This variant of the problem is called Pandora's box with *optional inspection* (also known as nonobligatory inspection) [33], in contrast with the original problem's *required inspection* (also known as obligatory inspection). Optional inspection results in a much harder problem than required inspection, with Fu et al. [39] showing it is NP-hard in an appropriate computational sense. In particular, while the Gittins index can still be defined, the Gittins policy is no longer optimal for reasons we explain below. With this said, the Gittins index is still a critical tool for the optional inspection setting: a number of approximation algorithms for Pandora's box with optional inspection are known [15, 17, 27, 39, 54, 87], and all of them use the Gittins policy as a subroutine.

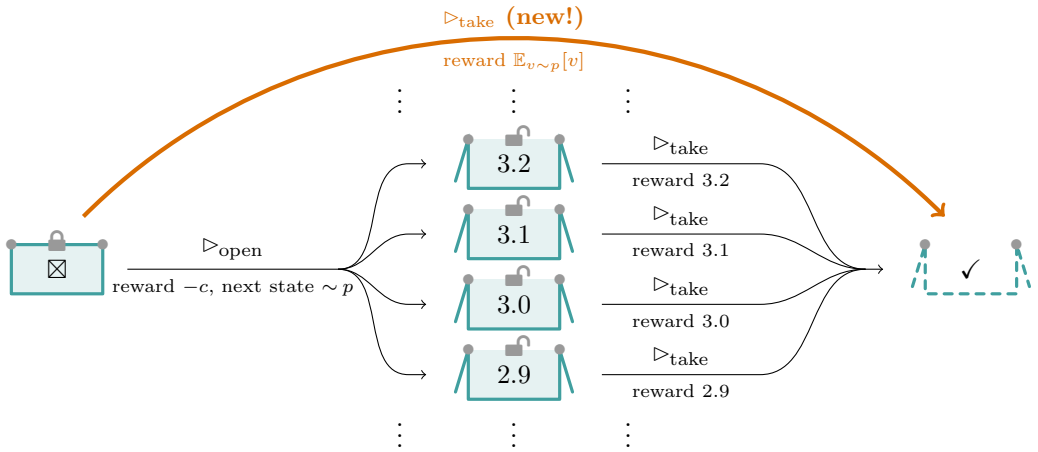


FIGURE 5.2. A single Pandora's box with optional inspection is not simply a Markov chain, but an MDP: specifically, from the closed state ☒, one can take either the $\triangleright_{\text{open}}$ action, which incurs cost but reveals the box's reward, or the $\triangleright_{\text{take}}$ action, which takes the box without opening it first. Once a box is opened, the only available action is $\triangleright_{\text{take}}$.

The core reason why the Pandora's box problem with optional inspection is difficult is that it is *not* an instance of Markov chain selection (MCS; see Definition 3.2). Instead, it is an instance of what we call *MDP selection*, where instead of choosing which one of multiple Markov chains to advance at each time step, one chooses which one of multiple MDPs to advance, *along with which action to take in the chosen MDP*. MDP selection is usually referred to in the Gittins index literature as a the Markovian multi-armed bandit with *bandit superprocesses* [23, 56, 80, 109], but we introduce the MDP selection name for consistency with MCS. (See also the discussion in Section 3.1.1.)

The reason Pandora's box with optional inspection is an instance of MDP selection, as opposed to the simpler MCS, is that each box admits two possible actions when it is closed:

a. *Open*, denoted $\triangleright_{\text{open}}$ behaves like the Markov chain for required inspection, yielding reward $-c$ and advances the box to an open state $v \sim p$.

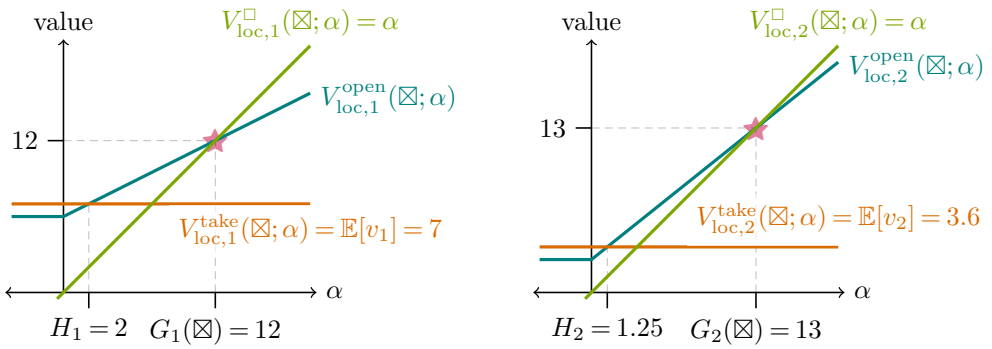
b. *Take*, denoted $\triangleright_{\text{take}}$, takes the box without opening it, yielding (expected) reward $\mathbb{E}_{v\sim p}[v]$.

Figure 5.2 gives an illustration.

5.2.1. Why MDP selection is harder than Markov chain selection One might hope that the Gittins index approach might extend from MCS to MDP selection. Indeed, it turns out that one can still define the Gittins index of an MDP in essentially the same way as for a Markov chain, namely using a local MDP (Definition 3.4). Unfortunately, the resulting Gittins policy for MDP selection is generally not optimal. This is not a surprise: MDP selection is NP-hard, thanks to the aforementioned NP-hardness of Pandora's box with optional inspection [39], so we should not expect the Gittins policy—which can be computed in polynomial time in the finite-state case [25, 41, 63]—to solve it. But what specifically prevents the Gittins policy from being optimal?

Let us first clarify what the local MDP and Gittins index look like for an MDP with action space A instead of a Markov chain. Just like in Definition 3.4, the local MDP is essentially the original MDP with an extra action $\square$ that terminates the process and yields reward α, meaning that we have

$$A_{\text{loc}} = A \cup \{\square\}. \tag{5.8}$$



(a) Local MDP value for box 1 from Figure 2.1. (b) Local MDP value for box 2 from Figure 2.1.

FIGURE 5.3. Analogue of Figure 3.2 for Pandora's box with optional inspection. Value functions of the $(\boxtimes,\alpha)$-local MDP for the two closed boxes in Figure 2.1 with three different initial actions: $\triangleright_{\mathrm{open}}$ (teal), $\triangleright_{\mathrm{take}}$ (orange), and $\square$ (violet). As in Figure 3.2, the optimal action is $\square$ for all values of α above a threshold, and we define the Gittins index $G_i(\boxtimes)$ to be that threshold. In these cases, the action that is co-optimal with $\square$ when $\alpha = G_i(\boxtimes)$ is $\triangleright_{\mathrm{open}}$. But when α is lower than another threshold H_i, the optimal first action is $\triangleright_{\mathrm{take}}$. This means that in the context of a larger Pandora's box with optional inspection problem, or more generally MDP selection, even if we are confident about wanting to play an action on box i, whether we prefer $\triangleright_{\mathrm{open}}$ or $\triangleright_{\mathrm{take}}$ may depend on the states of the other MDPs.

But this action space is no longer a two-element set: instead of a single action $\triangleright$ that advances the Markov chain, now each of the MDP's actions $a \in A$ advances it *using action* a, which must be specified. In spite of this, one can show that Definition 3.6 continues to make sense, yielding a well-defined Gittins index $G(s)$ of each state s among MDPs. We can then define the Gittins policy as the policy that always plays an action from the MDP of greatest Gittins index $G(s)$, choosing an action, other than $\square$, that is optimal for the $(G(s), s)$-local MDP.

The core issue is that optimality of the Gittins policy relies on the following fact about the local MDP with a Markov chain and any fixed starting state s:

($\star$) *If the $\triangleright$ action is optimal under some alternative α, then the same $\triangleright$ action is also optimal with any worse alternative $\alpha' < \alpha$.*

This property fails in general when using an MDP instead of a Markov chain, because the single action $\triangleright$ is replaced by the MDP's action space A. In particular, the optimal action for the local MDP with alternative $G(s)$ might be different than the optimal action with lower alternative option. Intuitively, this is a problem because it means that in full MDP selection, the optimal action to take within one MDP might depend on the states of the other MDPs. See Remark A.8 for details on exactly where the proof of Theorem 3.7 breaks down when generalizing from MCS to MDP selection.

For example, consider the MDP of a box in Pandora's box with optional inspection (Figure 5.2) in the closed state $\boxtimes$. We show the values in the local MDP for three different initial actions in Figure 5.3.

(a) For sufficiently large values of α, as usual, the optimal action is $\square$.

(b) For intermediate values of α, the optimal action is $\triangleright_{\mathrm{open}}$. The intuition is that the box has a good chance of being either significantly greater or significantly less than α, so it is worth paying the opening cost to learn the box's value.

(c) For sufficiently small values of α, the optimal action is $\triangleright_{\mathrm{take}}$. The intuition is that the alternative α is so low that we are very unlikely to prefer it to the box's value, so we are happy taking the box without paying the cost to open it.

This means that in the context of a broader MDP selection instance with multiple boxes, the optimal $\triangleright_{\text{open}}$ vs. $\triangleright_{\text{take}}$ choice within one box's MDP might depend on the states of the other MDPs. This can cause prioritizing by Gittins index to be suboptimal: Doval [33] gives a concrete example.

5.2.2. Approximate solutions to MDP selection using the Gittins index Despite the above challenges, many approximation algorithms have been proposed for Pandora's box with optional inspection [15, 17, 27, 39, 54, 87], and similarly for other Pandora's box variants and applications [7, 16, 21, 27, 66]. The Gittins index plays a critical role in most of these algorithms. For example, Fu et al. [39] and Beyhaghi and Cai [15] show that the optimal policy for the optional inspection setting is a two-phase policy, the second phase of which is to use the Gittins policy; and, while optimally choosing the phase boundary is intractable, they use this insight to construct a polynomial-time approximation scheme for the problem.

There are a few sufficient conditions under which the Gittins policy is known to be *optimal* for MDP selection. Doval [33] identifies one such condition for Pandora's box with optional inspection. In the general MDP selection setting, Whittle [109] identifies a condition, now called the *Whittle condition* [23, 48, 56] that can be checked separately for each local MDP, with the Gittins policy being optimal if all local MDPs satisfy it.

An MDP satisfies the Whittle condition if, roughly, it can be reduced to a Markov chain with no loss of value in the local MDP. Specifically, it requires that in every state s of the MDP, there is a single action a such that for all alternative values α, either $\square$ or a is optimal in the (s, α)-local MDP. This precludes the cases shown in Figure 5.3, where either $\triangleright_{\text{open}}$ or $\triangleright_{\text{take}}$ can be optimal. One can show, in Pandora's box with optional inspection, that a box MDP (Figure 5.2) satisfies the Whittle condition only in the trivial case where its opening cost is so large that $\triangleright_{\text{open}}$ is never optimal in the local MDP [33]. The Whittle condition is thus relatively limited in scope, though there are some notable classes of MDPs that satisfy it [47, 48, 107].

However, recent work has revealed fresh promise for the idea of reducing MDPs to Markov chains as a general approach for solving MDP selection: Scully and Doval [87] and Chawla et al. [27] introduce a relaxation of the Whittle condition called *local β-approximation* and show that many MDPs that fail the Whittle condition admit local approximations. Roughly speaking, an MDP admits a local β-approximation if it can be reduced to a Markov chain such that *if the rewards are then scaled by β*, there is no loss of value in the local MDP *with any alternative α*, as compared to the original local MDP with the same alternative α (and without any scaling). This slightly unusual approximation requirement—which is *not* equivalent to simply achieving a β-approximation in the local MDP—ensures the following guarantee: in an MDP selection instance where all the MDPs admit (possibly randomized) local β-approximations, the Gittins policy is a β-approximation of the optimal policy [27, 87]. Moreover, this guarantee also holds in the k-finish and combinatorial settings described in Section 4.3 [27, 87]. We suspect that local approximation is related to the results of Clarkson et al. [29], who prove an approximation guarantee for a special case of MDP selection without explicitly reasoning using the local MDP.

The local approximation approach described above reduces MDPs to Markov chains by attempting to solve the local MDP in a way that is in some sense good for any alternative value α. A complementary approach to could be to figure out, based on the specific MDP selection instance, what alternative value α_i is, in some sense, most relevant for each individual MDP i, then take actions within MDP i that would solve its local MDP with alternative α_i. Bowers et al. [20] prove a 1/2-approximation result that is a first step in this direction. In fact, they obtain their result even under an additional *take it or leave it* constraint, so it is possible that there are even stronger guarantees waiting to be shown with this approach.

5.3. Minimizing tail latency in queues

As our last example, we revisit the queue scheduling setting from Section 4.4, where we saw in Definition 4.3 and Theorem 4.4 that the Gittins policy minimizes *mean latency* in the M/G/1 queue. However, in many settings, a more relevant objective than minimizing mean latency is minimizing *tail latency*. Tail latency is a broad term that refers to one of a number of related metrics that capture how likely jobs are to have especially large latency. Optimizing tail latency is of direct importance to efficiently meeting *service level objectives* in a wide variety of queueing systems in service, computing, healthcare, and other domains.

The specific metric we focus on minimizing *tail probabilities*, namely the probabilities a job has latency greater than large thresholds t. That is, if an M/G/1 scheduling policy induces latency distribution L, then the tail probabilities are $\mathbb{P}[L > t]$. We could equivalently work with *tail quantiles*, namely the $(1-\varepsilon)$th quantiles of L for small values of ε.

Given a fixed threshold t, one might hope to use the Gittins index to minimize $\mathbb{P}[L > t]$ by having a job's Markov chain incur cost 1 (meaning, yield a reward of -1) once it has been in the system for time t. Unfortunately, this type of cost structure cannot be encoded as part of standard MCS (Definition 3.2) or its M/G/1 variant. The issue is that for the job to incur cost 1 after spending time t in the system, one would need to keep track of the job's time in the system so far as part of its state. But this quantity changes *even when the job is not in service*, whereas in MCS, a Markov chain only advances when its action is played—meaning, here, that a job changes state only when in service.

There is an extension of MCS, called the *restless bandit* problem [111], in which all Markov chains advance each step, with the selected Markov chain advancing according to a different (typically thought of as *better*) transition kernel and reward function than non-selected Markov chains. In some cases, the Gittins index can be generalized to the restless bandit setting—in which it is called the *Whittle index*. This approach has been used to for problems similar to minimizing tail probabilities [6, 116], but tends to obtain theoretical guarantees that are much weaker than optimality [106].

Nevertheless, there is a limited way in which ordinary MCS and the Gittins index can handle jobs undergoing some sort of change even when not in service: *discounting*. Suppose, for instance, that we consider job Markov chains with similar transitions to those in Figures 4.1–4.3, but instead of incurring cost 1 with all transitions, most transitions incur cost 0, with only transitions to the terminal state ✓ yielding *reward* 1. Then a job completed at time t would yield reward γ^t, where $\gamma < 1$ is the discount factor. Notably, this reward is affected by the *global* time t that advances every time step—no matter which job is served—which is exactly the type of phenomenon that one typically needs restless bandits to capture.

Translating the above discussion to the M/G/1 setting, it suggests that one might be able to use the Gittins index to maximize a metric like $\mathbb{E}[\gamma^L]$ for $\gamma < 1$.[8] Unfortunately, while this metric does incentivize completing jobs sooner rather than later, it does not capture tail scheduling well: once a job has accrued large latency, it becomes less and less urgent, because one is already guaranteed to receive a very small reward from it.

However, recent work by Yu and Scully [115] and Harlev et al. [57] shows that with a small tweak, the metric $\mathbb{E}[\gamma^L]$ becomes a good proxy for tail probabilities:

($\star$) *Instead of aiming to* MAXIMIZE $\mathbb{E}[\gamma^L]$ *with* DISCOUNT *factor* $\gamma < 1$*, one should aim to* MINIMIZE $\mathbb{E}[\gamma^L]$ *with* INFLATION *factor* $\gamma > 1$.

Indeed, when $\gamma > 1$, the goal of minimizing $\mathbb{E}[\gamma^L]$ not only incentivizes completing jobs earlier, but also causes jobs to become more urgent the longer they have waited: the cost to be eventually paid upon completion increases exponentially over time! Specifically, it is known

[8] Specifically, there are issues to do with arrivals, because one cannot capture the $\mathbb{E}[\gamma^L]$ metric using time-homogeneous arrivals [57, Section 3.3]. The work of Yu and Scully [115] and Harlev et al. [57], which we soon discuss, does resolve these issues, but doing so is among their main technical contributions.

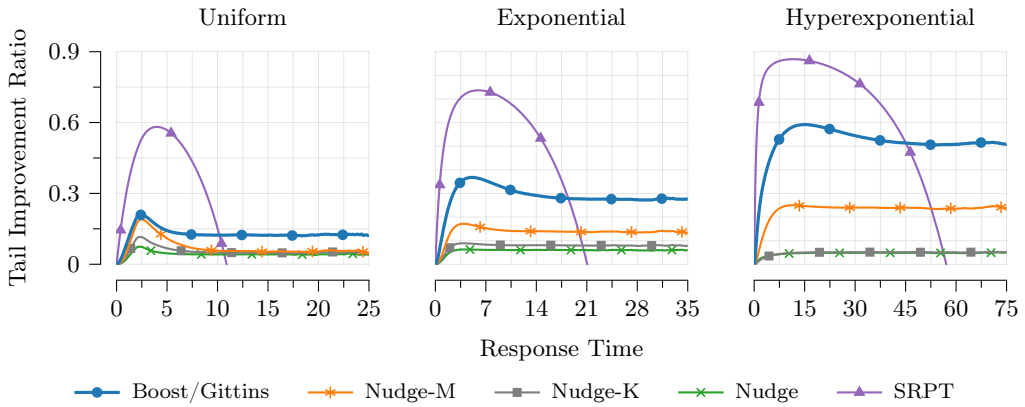


FIGURE 5.4. Results reproduced from Yu and Scully [115]: empirical performance (higher is better) of the Gittins policy for minimizing tail probabilities, also called *Boost* (blue) in the legend, against other baseline policies, simulated in three different M/G/1 models with different service time distributions. The job model is a continuous-time analogue of the known-size Markov chain from Figure 4.1. The metric shown is *tail improvement ratio* relative to *first-come first-served (FCFS)*, which for policy π and response time threshold t is $1 - \mathbb{P}[L_\pi > t]/\mathbb{P}[L_{\text{FCFS}} > t]$. The probabilities $\mathbb{P}[L_\pi > t]$ are approximated by simulating the policies on a trace of 50 million randomly generated arrivals. The Gittins policy is the clear winner over the Nudge family of baselines [26, 53, 101], with larger improvement when the service time distribution's coefficient of variation is larger. *Shortest remaining processing time (SRPT)* (purple) performs better than the Gittins policy for small thresholds t, but SRPT's performance suddenly collapses as t increases. This is because although SRPT minimizes mean latency [85], under light-tailed service times, it has the worst possible asymptotic tail probabilities as $t \to \infty$ [82, 83].

that under certain light-tail assumptions on the service time distribution, minimizing $\mathbb{E}[\gamma^L]$ for a carefully chosen value of $\gamma > 1$ results in *asymptotically minimal tail probabilities*—that is, minimizing the asymptotics of $\mathbb{P}[L > t]$ in the $t \to \infty$ limit in a certain precise sense [22, 113]. This idea led to the first policies that achieve better tail probabilities than simple first-come first-served (FCFS) policies for light-tailed service times [26, 53, 101]. However, it was viewing the problem as MCS with inflation that led to the discovery of the asymptotically optimal policies—which are instances of the Gittins policy—first for known service times [115], then for general job Markov chains [57]. See Harlev et al. [57, Appendix D] for a general account of MCS with inflation.

We conclude that, by using inflation instead of discounting, one can use the Gittins policy to asymptotically minimize tail probabilities $\mathbb{P}[L > t]$ as $t \to \infty$. Moreover, this translates into state-of-the-art empirical performance for practical values of t:

a. For known service times, Figure 5.4 shows that the respective Gittins policy makes a substantial improvement over other baselines.

b. For unknown service times, the Gittins policy is the first policy known to improve upon FCFS's tail asymptotics, so there are no other baselines to compare against.

We refer the interested reader to Yu and Scully [115] and Harlev et al. [57] for further details.

6. Conclusion

We have presented the *Gittins index* (Definition 3.6), a tool for solving decision-making problems under uncertainty that require choosing among multiple processes to advance. The Gittins index yields an optimal policy when these processes are independent—specifically, it solves MDPs that can be expressed as an instance of *Markov chain selection* (MCS, Definition 3.2)—and many problems fit directly into this framework (Section 4). The key

idea behind the Gittins index definition is to compare a stochastic action to a deterministic alternative in the *local MDP* (Definition 3.4), which continues to make sense in problems beyond MCS. In various cases, the Gittins index continues to yield strong policies in these more difficult problems (Section 5). In particular, we highlighted two practical applications where the Gittins index shows particular promise: Bayesian optimization (Section 5.1) and scheduling to minimize tail latency (Section 5.3).

6.1. Additional topics

There are many Gittins index topics that we did not cover. For example, we focused on one particular way of defining the Gittins index, but there are actually many equivalent definitions, each of which gives different intuition or insight [45, 63, 109]. We also only briefly touched on proofs, limiting ourselves to a dynamic programming argument in Appendix A: however, just as the Gittins index enjoys many definitions, it also enjoys many optimality proofs [12, 32, 34, 99, 100, 104, 107, 109]. See Frostig and Weiss [38] for an overview of four of the main optimality proof approaches and Appendix A.3 for additional discussion. Other important topics that we omitted or mentioned only briefly include:

1. *The history of the Gittins index*, for which we refer the reader to Gittins et al. [45] and Glazebrook et al. [50]. We also highlight the latter half of Gittins [44], which records discussion about the Gittins index shortly after its discovery.
2. *Efficiently computing the Gittins index*, which is well understood for finite-state Markov chains [25, 41, 63], but, as discussed in Section 3.4, remains a challenge for general infinite-state Markov chains [36, 45, 64, 65].
3. *Formulating the Gittins index in continuous time,* which is conceptually similar, but technically more difficult, than the discrete-time setting we focus on here [9, 61, 62, 75].
4. *Approximate optimality results* in two settings beyond standard MCS: when one has only approximately computed the Gittins index, and when multiple Markov chains must be played in parallel. See Gittins et al. [45, Sections 4.10 and 5.7] for a treatment of the discounted setting and Scully [86, Chapters 16 and 17] for a treatment of the queueing setting.
5. *Robust variants of the Gittins index* for settings with misspecified transition kernels [24, 31, 55, 65, 79, 89].
6. *Other modern work on the Gittins index*, including applying it to auction design [66], adapting it to fairness constraints [5], applying it to analyze games [30], better understanding its behavior in queues [1–3, 92, 93], and proving regret bounds for non-Bayesian bandit settings [36, 69].
7. *Restless bandit problems* [111], in which Markov chains can transition even on time steps when they are not played. In this setting, a generalization of the Gittins index, called the *Whittle index*, yields a good policy under certain conditions [106], and similar ideas have recently led to policies that achieve even better performance under more general conditions [8, 42, 59, 60, 102]. See Niño-Mora [81] for a recent survey.

6.2. Open problems

We conclude by listing several classes of open problems in Gittins indices, some of which have been mentioned throughout our exposition. The first a comprehensive understanding of *numerical computation* beyond finite-state settings, as discussed in Section 3.4. Are there general classes of infinite-state Markov chains for which the Gittins index can be efficiently computed, especially if the state space is high-dimensional? Can one utilize the very small and structured nature of the local MDP's action space to solve the corresponding dynamic program more efficiently than off-the-shelf approximate dynamic programming methods

would allow? An understanding of these questions would allow Gittins indices to be applied in substantially more complicated settings compared to those which are well understood today.

The second class of open problems is the *analysis beyond optimality* of the Gittins policy. We mention several results of this type in Section 6.1, but many open problems remain. One such problem, which is particularly important for Bayesian optimization (Section 5.1), is proving *regret bounds* on the Gittins policy. For finite-horizon bandits, an important initial step in this direction has been taken by Lattimore [69] and Farias and Gutin [36]. However, at present, even for simple regret in Pandora's box, a corresponding analysis has yet to be developed. An improved understanding of the Gittins policy's regret, and related quantities appropriate for other setups, could help understand in which situations the key definition is the right approach.

In this context, it is worth noting that compared to approximate optimality, exact optimality is a rigid notion—which, necessarily, captures all phenomena occurring in the problem, including those reflected in constant factors rather than rates. In contrast, approximate optimality arguments tend to work in greater generality, and can therefore provide a complementary understanding by clarifying which phenomena are specific and which are universal. Such analyses might therefore reveal properties of the Gittins index that complement what is known from its optimality theory.

A third class of open problems involves understanding *metrics beyond mean performance*. Here, we have illustrated a Gittins index variant that can be used to minimize tail latency in queueing. More broadly, many decision-making algorithms admit analogues that seek to perform well in terms of quantile regret, or in terms of high-probability bounds. We therefore expect there to be decision problems which may appear rather different from the classical Gittins index examples, but which nonetheless can be approached fruitfully using the presented toolkit.

Finally, we believe that, given the scope of generality presented in this tutorial—where Definition 3.2 allows for *arbitrary* Markov chains—that Gittins-index-based decision-making should be helpful for a broader set of domains that may otherwise appear to have little to do with queueing and economics, where Gittins indices have traditionally been applied. Domains where Bayesian optimization is popular, such as chemistry and material design, seem particularly promising: here, Gittins-index-based machinery might allow one to work with more-complex experimental pipelines—or with more-flexible probabilistic models defined by, for instance, diffusion models, rather than traditional Gaussian processes. To achieve this, developing the aforementioned understanding of numerical methods is a key initial step.

Acknowledgments

We thank the anonymous referees for their many helpful comments. Ziv Scully was supported by the NSF under grant no. CMMI-2307008. Alexander Terenin was supported by Cornell University, jointly via the Center for Data Science for Enterprise and Society, the College of Engineering, and the Ann S. Bowers College of Computing and Information Science.

References

[1] Aalto S, Ayesta U, Righter R (2009) On the Gittins index in the $M/G/1$ queue. *Queueing Systems* 63(1-4):437–458, `https://dx.doi.org/10.1007/s11134-009-9141-x`. Cited on pages 17, 18, and 31.

[2] Aalto S, Ayesta U, Righter R (2011) Properties of the Gittins index with application to optimal scheduling. *Probability in the Engineering and Informational Sciences* 25(3):269–288, `https://dx.doi.org/10.1017/S0269964811000015`. Cited on page 18.

[3] Aalto S, Scully Z (2023) Minimizing the mean slowdown in the M/G/1 queue. *Queueing Systems* 104(3-4):187–210, `https://dx.doi.org/10.1007/s11134-023-09888-6`. Cited on pages 17 and 31.

[4] Agrawal A, Amos B, Barratt S, Boyd S, Diamond S, Kolter JZ (2019) Differentiable convex optimization layers. *Advances in Neural Information Processing Systems (NeurIPS 2019)*, volume 32, 9562–9574 (Vancouver, BC: Curran Associates, Inc.), `https://dx.doi.org/10.48550/arXiv.1910.12430`. Cited on page 24.

[5] Aminian MR, Manshadi V, Niazadeh R (2023) Markovian search with socially aware constraints. https://dx.doi.org/10.2139/ssrn.4347447. Cited on page 31.

[6] Anand A, de Veciana G (2018) A Whittle's index based approach for QoE optimization in wireless networks. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 2(1):1–39, https://dx.doi.org/10.1145/3179418. Cited on page 29.

[7] Aouad A, Ji J, Shaposhnik Y (2020) Pandora's box problem with sequential inspections. https://dx.doi.org/10.2139/ssrn.3726167. Cited on pages 28, 43, and 44.

[8] Avrachenkov K, Borkar VS, Shah P (2024) Lagrangian index policy for restless bandits with average reward. https://dx.doi.org/10.48550/arXiv.2412.12641. Cited on page 31.

[9] Bank P, Küchler C (2007) On Gittins' index theorem in continuous time. *Stochastic Processes and their Applications* 117(9):1357–1371, https://dx.doi.org/10.1016/j.spa.2007.01.006. Cited on pages 31 and 44.

[10] Bertsekas DP (2012) *Dynamic Programming and Optimal Control, Volume 2: Approximate Dynamic Programming* (Belmont, MA: Athena Scientific), 4 edition, https://www.mit.edu/~dimitrib/dpbook.html. Cited on page 14.

[11] Bertsimas D (1995) The achievable region method in the optimal control of queueing systems; formulations, bounds and policies. *Queueing Systems* 21(3):337–389, https://dx.doi.org/10.1007/BF01149167. Cited on pages 17 and 44.

[12] Bertsimas D, Niño-Mora J (1996) Conservation laws, extended polymatroids and multiarmed bandit problems; a polyhedral approach to indexable systems. *Mathematics of Operations Research* 21(2):257–306, https://dx.doi.org/10.1287/moor.21.2.257. Cited on page 31.

[13] Bertsimas D, Niño-Mora J (1999) Optimization of multiclass queueing networks with changeover times via the achievable region approach: Part I, the single-station case. *Mathematics of Operations Research* 24(2):306–330, https://dx.doi.org/10.1287/moor.24.2.306. Cited on pages 17 and 21.

[14] Bertsimas D, Niño-Mora J (1999) Optimization of multiclass queueing networks with changeover times via the achievable region approach: Part II, the multi-station case. *Mathematics of Operations Research* 24(2):331–361, https://dx.doi.org/10.1287/moor.24.2.331. Cited on page 44.

[15] Beyhaghi H, Cai L (2023) Pandora's problem with nonobligatory inspection: Optimal structure and a PTAS. *Proceedings of the 55th Annual ACM Symposium on Theory of Computing (STOC 2023)*, 803–816 (Orlando, FL: ACM), https://dx.doi.org/10.1145/3564246.3585217. Cited on pages 25 and 28.

[16] Beyhaghi H, Cai L (2023) Recent developments in Pandora's box problem: Variants and applications. *ACM SIGecom Exchanges* 21(1):20–34, https://dx.doi.org/10.1145/3699814.3699817. Cited on pages 28 and 43.

[17] Beyhaghi H, Kleinberg R (2019) Pandora's problem with nonobligatory inspection. *Proceedings of the 2019 ACM Conference on Economics and Computation (EC 2019)*, 131–132 (Phoenix, AZ: ACM), https://dx.doi.org/10.1145/3328526.3329626. Cited on pages 25 and 28.

[18] Blondel M, Berthet Q, Cuturi M, Frostig R, Hoyer S, Llinares-Lopez F, Pedregosa F, Vert JP (2022) Efficient and modular implicit differentiation. *Advances in Neural Information Processing Systems (NeurIPS 2022)*, volume 35, 5230–5242 (New Orleans, LA: Curran Associates, Inc.), https://dx.doi.org/10.48550/arXiv.2105.15183. Cited on page 24.

[19] Boodaghians S, Fusco F, Lazos P, Leonardi S (2020) Pandora's box problem with order constraints. *Proceedings of the 21st ACM Conference on Economics and Computation (EC 2020)*, 439–458 (Budapest, Hungary: ACM), https://dx.doi.org/10.1145/3391403.3399501. Cited on page 22.

[20] Bowers R, Lindgren E, Waggoner B (2025) Prophet inequalities for bandits, cabinets, and DAGs. https://dx.doi.org/10.48550/arXiv.2502.08976. Cited on pages 28 and 43.

[21] Bowers R, Waggoner B (2024) Matching with nested and bundled Pandora boxes. http://arxiv.org/abs/2406.08711. Cited on pages 28 and 44.

[22] Boxma OJ, Zwart B (2007) Tails in scheduling. *ACM SIGMETRICS Performance Evaluation Review* 34(4):13–20, https://dx.doi.org/10.1145/1243401.1243406. Cited on page 30.

[23] Brown DB, Smith JE (2013) Optimal sequential exploration: Bandits, clairvoyants, and wildcats. *Operations Research* 61(3):644–665, https://dx.doi.org/10.1287/opre.2013.1164. Cited on pages 26, 28, 43, and 44.

[24] Caro F, Das Gupta A (2022) Robust control of the multi-armed bandit problem. *Annals of Operations Research* 317(2):461–480, https://dx.doi.org/10.1007/s10479-015-1965-7. Cited on page 31.

[25] Chakravorty J, Mahajan A (2014) Multi-armed bandits, Gittins index, and its calculation. Balakrishnan N, ed., *Methods and Applications of Statistics in Clinical Trials*, 416–435 (Hoboken, NJ: Wiley), https://dx.doi.org/10.1002/9781118596333.ch24. Cited on pages 13, 14, 26, and 31.

[26] Charlet N, Van Houdt B (2024) Tail optimality and performance analysis of the Nudge-M scheduling algorithm. http://arxiv.org/abs/2403.06588. Cited on page 30.

[27] Chawla S, Christou D, Harlev A, Scully Z (2024) Combinatorial selection with costly information. https://dx.doi.org/10.48550/arXiv.2412.03860. Cited on pages 14, 25, 28, 43, and 44.

[28] Chawla S, Gergatsouli E, McMahan J, Tzamos C (2022) Approximating Pandora's box with correlations. http://arxiv.org/abs/2108.12976. Cited on page 25.

[29] Clarkson J, Glazebrook KD, Lin KY (2020) Fast or slow: Search in discrete locations with two search modes. *Operations Research* 68(2):552–571, https://dx.doi.org/10.1287/opre.2019.1870. Cited on page 28.

[30] Clarkson J, Lin KY, Glazebrook KD (2023) A classical search game in discrete locations. *Mathematics of Operations Research* 48(2):687–707, https://dx.doi.org/10.1287/moor.2022.1279. Cited on page 31.

[31] Cohen SN, Treetanthiploet T (2022) Gittins' theorem under uncertainty. *Electronic Journal of Probability* 27(none), https://dx.doi.org/10.1214/22-EJP742. Cited on page 31.

[32] Dacre M, Glazebrook KD, Niño-Mora J (1999) The achievable region approach to the optimal control of stochastic systems. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 61(4):747–791, https://dx.doi.org/10.1111/1467-9868.00202. Cited on pages 31 and 44.

[33] Doval L (2018) Whether or not to open Pandora's box. *Journal of Economic Theory* 175:127–158, https://dx.doi.org/10.1016/j.jet.2018.01.005. Cited on pages 25 and 28.

[34] Dumitriu I, Tetali P, Winkler P (2003) On playing golf with two balls. *SIAM Journal on Discrete Mathematics* 16(4):604–615, https://dx.doi.org/10.1137/S0895480102408341. Cited on pages 31 and 44.

[35] Eggensperger K, Müller P, Mallik N, Feurer M, Sass R, Klein A, Awad N, Lindauer M, Hutter F (2021) HPOBench: A collection of reproducible multi-fidelity benchmark problems for HPO. *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS Datasets and Benchmarks 2021)* (Curran Associates, Inc.), https://dx.doi.org/10.48550/arXiv.2109.06716. Cited on page 25.

[36] Farias VF, Gutin E (2022) Optimistic Gittins indices. *Operations Research* 70(6):3432–3456, https://dx.doi.org/10.1287/opre.2021.2207. Cited on pages 14, 16, 31, and 32.

[37] Frazier PI (2018) Bayesian optimization. Gel E, Ntaimo L, Shier D, Greenberg HJ, eds., *Recent Advances in Optimization and Modeling of Contemporary Problems*, 255–278 (INFORMS), https://dx.doi.org/10.1287/educ.2018.0188. Cited on page 22.

[38] Frostig E, Weiss G (2016) Four proofs of Gittins' multiarmed bandit theorem. *Annals of Operations Research* 241(1-2):127–165, https://dx.doi.org/10.1007/s10479-013-1523-0. Cited on page 31.

[39] Fu H, Li J, Liu D (2023) Pandora box problem with nonobligatory inspection: Hardness and approximation scheme. *Proceedings of the 55th Annual ACM Symposium on Theory of Computing (STOC 2023)*, 789–802 (Orlando, FL: ACM), https://dx.doi.org/10.1145/3564246.3585229. Cited on pages 25, 26, and 28.

[40] Garnett R (2023) *Bayesian Optimization* (Cambridge, UK: Cambridge University Press), https://bayesoptbook.com. Cited on page 22.

[41] Gast N, Gaujal B, Khun K (2023) Testing indexability and computing Whittle and Gittins index in subcubic time. *Mathematical Methods of Operations Research* 97(3):391–436, https://dx.doi.org/10.1007/s00186-023-00821-4. Cited on pages 14, 26, and 31.

[42] Gast N, Narasimha D (2025) Model predictive control is almost optimal for restless bandit. https://dx.doi.org/10.48550/arXiv.2410.06307. Cited on page 31.

[43] Gergatsouli E, Tzamos C (2023) Weitzman's rule for Pandora's box with correlations. *Advances in Neural Information Processing Systems (NeurIPS 2023)*, volume 36, 12644–12664 (New Orleans, LA: Curran Associates, Inc.), https://dx.doi.org/10.48550/arXiv.2301.13534. Cited on pages 14 and 25.

[44] Gittins JC (1979) Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society: Series B (Methodological)* 41(2):148–164, https://dx.doi.org/10.1111/j.2517-6161.1979.tb01068.x. Cited on pages 7, 9, 10, and 31.

[45] Gittins JC, Glazebrook KD, Weber RR (2011) *Multi-Armed Bandit Allocation Indices* (Chichester, UK: Wiley), 2 edition, https://dx.doi.org/10.1002/9780470980033. Cited on pages 2, 11, 13, 14, 16, 21, 22, 31, and 44.

[46] Gittins JC, Jones DM (1974) A dynamic allocation index for the sequential design of experiments. Gani JM, Sarkadi K, Vincze I, eds., *Progress in Statistics*, 241–266, number 9 in Colloquia Mathematica Societatis János Bolyai (Amsterdam, The Netherlands: North-Holland), https://ci.nii.ac.jp/ncid/BA03522866. Cited on pages 8 and 9.

[47] Glazebrook KD (1979) Stoppable families of alternative bandit processes. *Journal of Applied Probability* 16(4):843–854, https://dx.doi.org/10.2307/3213150. Cited on page 28.

[48] Glazebrook KD (1982) On a sufficient condition for superprocesses due to Whittle. *Journal of Applied Probability* 19(1):99–110, https://dx.doi.org/10.2307/3213920. Cited on pages 28 and 44.

[49] Glazebrook KD (2003) An analysis of Klimov's problem with parallel servers. *Mathematical Methods of Operations Research* 58(1):1–28, https://dx.doi.org/10.1007/s001860300278. Cited on pages 17, 20, 21, and 44.

[50] Glazebrook KD, Hodge DJ, Kirkbride C, Minty RJ (2014) Stochastic scheduling: A short history of index policies and new approaches to index generation for dynamic resource allocation. *Journal of Scheduling* 17(5):407–425, https://dx.doi.org/10.1007/s10951-013-0325-1. Cited on pages 13 and 31.

[51] Glazebrook KD, Niño-Mora J (2001) Parallel scheduling of multiclass $M/M/m$ queues: Approximate and heavy-traffic optimization of achievable performance. *Operations Research* 49(4):609–623, https://dx.doi.org/10.1287/opre.49.4.609.11225. Cited on pages 21 and 44.

[52] Grosof I, Scully Z, Harchol-Balter M, Scheller-Wolf A (2022) Optimal scheduling in the multiserver-job model under heavy traffic. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 6(3), https://dx.doi.org/10.1145/3570612. Cited on page 21.

[53] Grosof I, Yang K, Scully Z, Harchol-Balter M (2021) Nudge: Stochastically improving upon FCFS. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 5(2), https://dx.doi.org/10.1145/3460088. Cited on page 30.

[54] Guha S, Munagala K, Sarkar S (2008) Information acquisition and exploitation in multichannel wireless networks. http://arxiv.org/abs/0804.1724. Cited on pages 25 and 28.

[55] Gupta A, Jiang H, Scully Z, Singla S (2019) The Markovian price of information. Lodi A, Nagarajan V, eds., *Integer Programming and Combinatorial Optimization, 20th International Conference (IPCO 2019)*, volume 11480 of *Lecture Notes in Computer Science*, 233–246 (Cham, Switzerland: Springer), https://dx.doi.org/10.1007/978-3-030-17953-3_18. Cited on pages 17, 31, 43, and 44.

[56] Hadfield-Menell D, Russel S (2015) Multitasking: Efficient optimal planning for bandit superprocesses. *31st Conference on Uncertainty in Artificial Intelligence (UAI 2015)*, 345–354 (Amsterdam, The Netherlands: AUAI Press), https://auai.org/uai2015/proceedings.shtml. Cited on pages 26 and 28.

[57] Harlev A, Yu G, Scully Z (2025) A Gittins policy for optimizing tail latency. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 9(2), https://dx.doi.org/10.1145/3727109. Cited on pages 29 and 30.

[58] Hong Y, Scully Z (2024) Performance of the Gittins policy in the G/G/1 and G/G/k, with and without setup times. *Performance Evaluation* 163, https://dx.doi.org/10.1016/j.peva.2023.102377. Cited on page 21.

[59] Hong Y, Xie Q, Chen Y, Wang W (2023) Restless bandits with average reward: Breaking the uniform global attractor assumption. *Advances in Neural Information Processing Systems (NeurIPS 2023)*, 12810–12844 (New Orleans, LA: Curran Associates, Inc.), https://dx.doi.org/10.48550/arXiv.2306.00196. Cited on page 31.

[60] Hong Y, Xie Q, Chen Y, Wang W (2024) Achieving exponential asymptotic optimality in average-reward restless bandits without global attractor assumption. https://dx.doi.org/10.48550/arXiv.2405.17882. Cited on page 31.

[61] Karoui NE, Karatzas I (1994) Dynamic allocation problems in continuous time. *The Annals of Applied Probability* 4(2):255–286, https://dx.doi.org/10.1214/aoap/1177005062. Cited on pages 31 and 44.

[62] Kaspi H, Mandelbaum A (1998) Multi-armed bandits in discrete and continuous time. *The Annals of Applied Probability* 8(4):1270–1290, https://dx.doi.org/10.1214/aoap/1028903380. Cited on pages 31 and 44.

[63] Katehakis MN, Veinott AF (1987) The multi-armed bandit problem: Decomposition and computation. *Mathematics of Operations Research* 12(2):262–268, https://dx.doi.org/10.1287/moor.12.2.262. Cited on pages 26 and 31.

[64] Kelly FP (1981) Multi-armed bandits with discount factor near one: The Bernoulli case. *The Annals of Statistics* 9(5), https://dx.doi.org/10.1214/aos/1176345578. Cited on pages 14, 16, and 31.

[65] Kim MJ, Lim AE (2016) Robust multiarmed bandit problems. *Management Science* 62(1):264–285, https://dx.doi.org/10.1287/mnsc.2015.2153. Cited on pages 14, 16, and 31.

[66] Kleinberg R, Waggoner B, Weyl EG (2016) Descending price optimally coordinates search. *Proceedings of the 2016 ACM Conference on Economics and Computation (EC 2016)*, 23–24 (Maastricht, The Netherlands: ACM), https://dx.doi.org/10.1145/2940716.2940760. Cited on pages 28, 31, and 44.

[67] Klimov GP (1974) Time-sharing service systems. I. *Theory of Probability & Its Applications* 19(3):532–551, https://dx.doi.org/10.1137/1119060. Cited on pages 9 and 17.

[68] Klimov GP (1978) Time-sharing service systems. II. *Theory of Probability & Its Applications* 23(2):314–321, https://dx.doi.org/10.1137/1123034. Cited on pages 9 and 17.

[69] Lattimore T (2016) Regret analysis of the finite-horizon Gittins index strategy for multi-armed bandits. Feldman V, Rakhlin A, Shamir O, eds., *29th Annual Conference on Learning Theory (COLT 2016)*, volume 49 of *Proceedings of Machine Learning Research*, 1214–1245 (New York, NY: PMLR), https://proceedings.mlr.press/v49/lattimore16.html. Cited on pages 14, 31, and 32.

[70] Lattimore T (2021) Lectures on information directed sampling. https://rlforum.stanford.edu/p/lec-ids/, Lecture 1 Recording, timestamp 31:10. Cited on page 3.

[71] Lattimore T, Szepesvári C (2020) *Bandit Algorithms* (Cambridge University Press), 1 edition, https://dx.doi.org/10.1017/9781108571401. Cited on page 11.

[72] Lee E, Eriksson D, Bindel D, Cheng B, Mccourt M (2020) Efficient rollout strategies for bayesian optimization. Peters J, Sontag D, eds., *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 124 of *Proceedings of Machine Learning Research*, 260–269 (PMLR), https://proceedings.mlr.press/v124/lee20a.html. Cited on page 24.

[73] Lee EH, Eriksson D, Perrone V, Seeger M (2021) A nonmyopic approach to cost-constrained Bayesian optimization. de Campos C, Maathuis MH, eds., *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence (UAI 2021)*, volume 161 of *Proceedings of Machine Learning Research*, 568–577 (PMLR), https://proceedings.mlr.press/v161/lee21a.html. Cited on page 24.

[74] Mandelbaum A (1986) Discrete multi-armed bandits and multi-parameter processes. *Probability Theory and Related Fields* 71(1):129–147, https://dx.doi.org/10.1007/BF00366276. Cited on page 44.

[75] Mandelbaum A (1987) Continuous multi-armed bandits and multiparameter processes. *The Annals of Probability* 15(4):1527–1556, https://dx.doi.org/10.1214/aop/1176991992. Cited on pages 31 and 44.

[76] Megow N, Vredeveld T (2014) A tight 2-approximation for preemptive stochastic scheduling. *Mathematics of Operations Research* 39(4):1297–1310, https://dx.doi.org/10.1287/moor.2014.0653. Cited on page 21.

[77] Meister M, Kleinberg J (2021) Optimizing the order of actions in contact tracing. http://arxiv.org/abs/2107.09803. Cited on page 22.

[78] Milgrom P, Segal I (2002) Envelope theorems for arbitrary choice sets. *Econometrica* 70(2):583–601, https://dx.doi.org/10.1111/1468-0262.00296. Cited on page 12.

[79] Moseley B, Newman H, Pruhs K, Zhou R (2025) Robust Gittins for stochastic scheduling. https://dx.doi.org/10.48550/arXiv.2504.10743. Cited on page 31.

[80] Nash P (1973) *Optimal Allocation of Resources between Research Projects*. Ph.D. thesis, University of Cambridge, Cambridge, UK, https://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.466839. Cited on page 26.

[81] Niño-Mora J (2023) Markovian restless bandits and index policies: A review. *Mathematics* 11(7):1639, https://dx.doi.org/10.3390/math11071639. Cited on page 31.

[82] Nuyens M, Wierman A, Zwart B (2008) Preventing large sojourn times using SMART scheduling. *Operations Research* 56(1):88–101, https://dx.doi.org/10.1287/opre.1070.0504. Cited on page 30.

[83] Nuyens M, Zwart B (2006) A large-deviations analysis of the $GI/GI/1$ SRPT queue. *Queueing Systems* 54(2):85–97, https://dx.doi.org/10.1007/s11134-006-8767-1. Cited on page 30.

[84] Persky E (2021) *Exploration and Exploitation: From Bandits to Bayesian Optimisation*. Master's thesis, University of Cambridge, Cambridge, UK, https://www.mlmi.eng.cam.ac.uk/files/2020-2021_dissertations/exploration_and_exploitation.pdf. Cited on page 24.

[85] Schrage LE (1968) A proof of the optimality of the shortest remaining processing time discipline. *Operations Research* 16(3):687–690, https://dx.doi.org/10.1287/opre.16.3.687. Cited on pages 18 and 30.

[86] Scully Z (2022) *A New Toolbox for Scheduling Theory*. Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA, https://ziv.codes/pdf/scully-thesis.pdf. Cited on pages 21, 31, and 44.

[87] Scully Z, Doval L (2024) Local hedging approximately solves Pandora's box problems with nonobligatory inspection. https://dx.doi.org/10.48550/arXiv.2410.19011. Cited on pages 14, 25, 28, 43, and 44.

[88] Scully Z, Grosof I, Harchol-Balter M (2020) The Gittins policy is nearly optimal in the M/G/k under extremely general conditions. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 4(3), https://dx.doi.org/10.1145/3428328. Cited on pages 21 and 44.

[89] Scully Z, Grosof I, Mitzenmacher M (2022) Uniform bounds for scheduling with job size estimates. *13th Innovations in Theoretical Computer Science Conference (ITCS 2022)*, Leibniz International Proceedings in Informatics (LIPIcs) (Berkeley, CA: Schloss Dagstuhl – Leibniz-Zentrum für Informatik), https://dx.doi.org/10.4230/LIPIcs.ITCS.2022.114. Cited on page 31.

[90] Scully Z, Harchol-Balter M (2021) The Gittins policy in the M/G/1 queue. *19th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt 2021)*, 248–255 (Philadelphia, PA: IFIP), https://dx.doi.org/10.23919/WiOpt52861.2021.9589051. Cited on pages 17, 20, 21, and 44.

[91] Scully Z, Harchol-Balter M, Scheller-Wolf A (2018) Optimal scheduling and exact response time analysis for multistage jobs. https://dx.doi.org/10.48550/arXiv.1805.06865. Cited on pages 14 and 38.

[92] Scully Z, van Kreveld L (2024) When does the Gittins policy have asymptotically optimal response time tail in the M/G/1? *Operations Research* 72(2), https://dx.doi.org/10.1287/opre.2022.0038. Cited on page 31.

[93] Scully Z, van Kreveld L, Boxma OJ, Dorsman JP, Wierman A (2020) Characterizing policies with optimal response time tails under heavy-tailed job sizes. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 4(2), https://dx.doi.org/10.1145/3392148. Cited on page 31.

[94] Sevcik KC (1971) *The Use of Service Time Distributions in Scheduling*. Ph.D. thesis, University of Chicago, Chicago, IL, https://dx.doi.org/10.2172/4710384. Cited on pages 9 and 17.

[95] Sevcik KC (1974) Scheduling for minimum total loss using service time distributions. *Journal of the ACM* 21(1):66–75, https://dx.doi.org/10.1145/321796.321803. Cited on pages 9 and 17.

[96] Singla S (2018) The price of information in combinatorial optimization. *Proceedings of the 2018 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2018)*, 2523–2532 (New Orleans, LA: SIAM), https://dx.doi.org/10.1137/1.9781611975031.161. Cited on pages 6, 17, and 43.

[97] Slivkins A (2019) Introduction to multi-armed bandits. *Foundations and Trends in Machine Learning* 12(1-2):1–286, https://dx.doi.org/10.1561/2200000068. Cited on page 11.

[98] Snoek J, Larochelle H, Adams RP (2012) Practical bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems (NIPS 2012)*, volume 25, 2951–2959

(Lake Tahoe, NV: Curran Associates, Inc.), https://dx.doi.org/10.48550/arXiv.1206.2944. Cited on page 22.

[99] Tsitsiklis J (1986) A lemma on the multiarmed bandit problem. *IEEE Transactions on Automatic Control* 31(6):576–577, https://dx.doi.org/10.1109/TAC.1986.1104332. Cited on page 31.

[100] Tsitsiklis JN (1994) A short proof of the Gittins index theorem. *The Annals of Applied Probability* 4(1):194–199, https://dx.doi.org/10.1214/aoap/1177005207. Cited on pages 31 and 44.

[101] Van Houdt B (2022) On the stochastic and asymptotic improvement of first-come first-served and nudge scheduling. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 6(3):1–22, https://dx.doi.org/10.1145/3570610. Cited on page 30.

[102] Verloop IM (2016) Asymptotically optimal priority policies for indexable and nonindexable restless bandits. *The Annals of Applied Probability* 26(4), https://dx.doi.org/10.1214/15-AAP1137. Cited on page 31.

[103] von Olivier G (1972) Kostenminimale Prioritäten in Wartesystemen vom Typ M/G/1 [Cost-minimum priorities in queueing systems of type M/G/1]. *Elektronische Rechenanlagen* 14(6):262–271, https://dx.doi.org/10.1524/itit.1972.14.16.262. Cited on pages 9 and 17.

[104] Weber RR (1992) On the Gittins index for multiarmed bandits. *The Annals of Applied Probability* 2(4):1024–1033, https://dx.doi.org/10.1214/aoap/1177005588. Cited on pages 31 and 44.

[105] Weber RR (2014) Multi-armed bandits and the Gittins index theorem. https://www.statslab.cam.ac.uk/~rrw1/oc/ocgittins.pdf. Cited on page 2.

[106] Weber RR, Weiss G (1990) On an index policy for restless bandits. *Journal of Applied Probability* 27(3):637–648, https://dx.doi.org/10.2307/3214547. Cited on pages 29 and 31.

[107] Weiss G (1988) Branching bandit processes. *Probability in the Engineering and Informational Sciences* 2(3):269–278, https://dx.doi.org/10.1017/S0269964800000826. Cited on pages 22, 28, 31, and 44.

[108] Weitzman ML (1979) Optimal search for the best alternative. *Econometrica* 47(3):641, https://dx.doi.org/10.2307/1910412. Cited on pages 3, 7, and 9.

[109] Whittle P (1980) Multi-armed bandits and the Gittins index. *Journal of the Royal Statistical Society: Series B (Methodological)* 42(2):143–149, https://dx.doi.org/10.1111/j.2517-6161.1980.tb01111.x. Cited on pages 26, 28, 31, 38, and 44.

[110] Whittle P (1981) Arm-acquiring bandits. *The Annals of Probability* 9(2), https://dx.doi.org/10.1214/aop/1176994469. Cited on page 20.

[111] Whittle P (1988) Restless bandits: Activity allocation in a changing world. *Journal of Applied Probability* 25(A):287–298, https://dx.doi.org/10.2307/3214163. Cited on pages 29 and 31.

[112] Whittle P (2005) Tax problems in the undiscounted case. *Journal of Applied Probability* 42(3):754–765, https://dx.doi.org/10.1239/jap/1127322025. Cited on page 17.

[113] Wierman A, Zwart B (2012) Is tail-optimal scheduling possible? *Operations Research* 60(5):1249–1257, https://dx.doi.org/10.1287/opre.1120.1086. Cited on page 30.

[114] Xie Q, Astudillo R, Frazier P, Scully Z, Terenin A (2024) Cost-aware Bayesian optimization via the Pandora's box Gittins index. *Advances in Neural Information Processing Systems (NeurIPS 2024)*, volume 37 (Vancouver, BC: Curran Associates, Inc.), https://dx.doi.org/10.48550/arXiv.2406.20062. Cited on pages 12, 14, 23, 24, 25, and 40.

[115] Yu G, Scully Z (2024) Strongly tail-optimal scheduling in the light-tailed M/G/1. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 8(2), https://dx.doi.org/10.1145/3656011. Cited on pages 29 and 30.

[116] Yu Z, Xu Y, Tong L (2018) Deadline scheduling as restless bandits. *IEEE Transactions on Automatic Control* 63(8):2343–2358, https://dx.doi.org/10.1109/TAC.2018.2807924. Cited on page 29.

[117] Zhao Q (2020) *Multi-Armed Bandits: Theory and Applications to Online Learning in Networks*. Synthesis Lectures on Learning, Networks, and Algorithms (Cham, Switzerland: Springer), https://dx.doi.org/10.1007/978-3-031-79289-2. Cited on page 13.

[118] Zimmer L, Lindauer M, Hutter F (2021) Auto-PyTorch: Multi-fidelity MetaLearning for efficient and robust AutoDL. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43(9):3079–3090, https://dx.doi.org/10.1109/TPAMI.2021.3067763. Cited on page 25.

A. Optimality of the Gittins index policy

In this appendix, we prove Theorem 3.7, namely optimality of the Gittins policy for MCS (Definition 3.2). The result is a direct corollary of Theorem A.7, the main result of this appendix, which gives an explicit formula for the value function of MCS. Our proof, given in Appendix A.1, consists of the following steps:

1. We state the specific assumptions needed for our proof. Some involve dynamic programming (Assumption A.1), while others are taken to ease presentation (Assumption A.2).
2. We define the *surrogate value* of a Markov chain (Definition A.4), a random variable that gives a probabilistic interpretation of the local MDP value function (Lemma A.5).
3. We define a guess for the MCS value function by appropriately combining the Markov chains' surrogate prices, then show that it solves the MCS Bellman equation (Theorem A.7). The rough idea is that our MCS value function guess inherits the respective Bellman inequalities of the Markov chains' local MDPs.

After the proof, we explain how to extend it from MCS to MCS-k (Appendix A.2), then discuss other proofs of the Gittins policy's optimality from the literature (Appendix A.3). Our proof is primarily based on that of Whittle [109] but uses ideas from other proofs, too.

A.1. Optimality via dynamic programming using surrogate values

To begin, define the *Bellman operator of action i*, denoted $\mathcal{B}_i$, to be an affine operator that acts on all $f : S_i \to \mathbb{R}$ for which $\mathbb{E}_{s_i' \sim p_i(s_i)}[f(s_i')]$ is finite for all s_i by

$$(\mathcal{B}_i f)(s_i) = r_i(s_i) + \mathbb{E}_{s_i' \sim p_i(s_i)}[f(s_i')]. \tag{A.1}$$

Using Assumption 3.3, by standard theory, the local MDP's optimal value function $V^*_{\mathrm{loc},i} : S_i \times \mathbb{R} \to \mathbb{R}$ is well defined and solves the Bellman optimality equation

$$V^*_{\mathrm{loc},i}(s_i; \alpha) = \max\bigl(\alpha, \mathcal{B}_i V^*_{\mathrm{loc},i}(s_i; \alpha)\bigr). \tag{A.2}$$

The corresponding Bellman optimality equation for MCS is

$$V^*_{\mathrm{MCS}}(s_1, \dots, s_n) = \max_{i \in \{1,\dots,n\}} \mathcal{B}_i V^*_{\mathrm{MCS}}(s_1, \dots, s_n). \tag{A.3}$$

where the Bellman operator $\mathcal{B}_i$ is understood as acting on the function $s_i \mapsto V^*_{\mathrm{MCS}}(s_1, \dots, s_n)$, and similarly throughout for other expressions using the variables $s_1, \dots, s_n$. We first make the following assumption, to ensure dynamic programming works the way one expects it to.

Assumption A.1. The optimal value function $V^*_{\mathrm{MCS}} : S_1 \times \cdots \times S_n \to \mathbb{R}$ is well defined, and is achieved by at least one policy $\pi^* : S_1 \times \cdots \times S_n \to \{1, \dots, n\}$. Moreover, a given policy is optimal if and only if its value function satisfies Bellman's optimality equation (A.3).

We expect Assumption A.1 to follow from Assumption 3.3, but do not rigorously check this in order to focus our presentation on Gittins-index-theoretic aspects. Note, however, that Assumption 3.3 guarantees that $V^*_{\mathrm{loc},i}(s_i; \alpha)$ is bounded by the sum of α plus the maximum absolute sum of the rewards, both of which are finite: this implies $V^*_{\mathrm{loc},i}$ is never infinite and hence well defined. Using this and finiteness of MCS' action space, one can show that the maximum in (A.3) is achieved, and therefore defines a policy. By unrolling this policy's respective Bellman equation and applying the fact that Assumption 3.3 guarantees either discounting or termination in finite time, one can check that the resulting policy achieves a value of V^*_{MCS}, which is always finite. In the other direction, any policy which does not satisfy Bellman's optimality equation is suboptimal, as it can be improved by replacing its action in some state with one that maximizes (A.3). A final subtlety worth noting is that, in our setting, (A.3) *need not* admit a unique solution V^*_{MCS}, and spurious solutions can occur in practical settings—see Scully et al. [91, Appendix D] for an example.

For ease of exposition, we make two additional assumptions without loss of generality.

Assumption A.2. Both of the following hold:

(a) There is no discounting: $\gamma = 1$.

(b) All Markov chains have no free states, where a Markov chain state is called *free* if it has non-negative reward and zero probability of transitioning directly into a terminal state.

The intuition behind Assumption A.2(b) is in the name: *free, as in beer*. Free states always give a reward, and never cause the decision problem to end, resulting in only gain with no downside. So, any reasonable algorithm should always play them whenever they occur. From a Gittins index viewpoint, this manifests in the form of having to deal with extended-valued functions: this is a technical nuisance, so we will assume without loss of generality that none occur. To avoid casework, we work in the undiscounted setting, also without loss of generality.

We now briefly argue why Assumption A.2 can be made without loss of generality. For (a), one can use the standard trick of replacing discount factor γ by probability-$(1-\gamma)$ transitions from all states to a terminal state. For (b), in an MCS instance with free states, one can show that an optimal agent always prioritizes playing Markov chains in free states before those in non-free states. We can thus eliminate free states from each Markov chain, altering the transition kernels and reward functions in light of the fact that the Markov chain will be advanced until it reaches a non-free state.

Provided there are no free states, the Gittins index is always finite, as shown below. In fact, it is this property that causes the MCS value function to admit a simple form.

Lemma A.3. *For any state non-terminal s of any Markov chain satisfying Assumption 3.3, if s is not free, then $G(s) < \infty$.*

Proof. Consider a Markov chain $(S, \partial S, p, r)$, and define the following quantities for all non-terminal states s:

- Let $q(s) = \mathbb{P}_{s' \sim p(s)}[s' \in \partial S]$ be the probability of transitioning directly from s to a terminal state.
- Recall that $r(s)$ is the immediate reward of s.
- Let random variable $R(s)$ be the total reward received on a random trajectory from s to a terminal state. Assumption 3.3 tells us $\mathbb{E}[R(s)] < \infty$.

Suppose s is not free, meaning either $q(s) > 0$ or $r(s) < 0$, and consider the (s, α)-local MDP It suffices to show there exists $\alpha \in \mathbb{R}$ such that playing $\square$ is strictly better than playing $\triangleright$.

There are two cases to consider, depending on the reason s is not free. In both cases, we bound $V^{\triangleright}_{\mathrm{loc}}(s; \alpha)$, the optimal value achievable in the local MDP when playing $\triangleright$ at least once. Specifically, we show $V^{\triangleright}_{\mathrm{loc}}(s; \alpha) < \alpha$ for sufficiently large $\alpha \in \mathbb{R}$ in both cases, which implies the result.

When $q(s) > 0$, we apply the bound

$$V^{\triangleright}_{\mathrm{loc}}(s; \alpha) \le \mathbb{E}[R(s)] + (1 - q(s))\alpha. \tag{A.4}$$

This holds because the maximum expected reward obtainable from the Markov chain is $\mathbb{E}[R(s)]$, and when playing $\triangleright$ first, the probability of ever playing $\square$ is at most $1 - q(s)$. Because $\mathbb{E}[R(s)] < \infty$ and $1 - q(s) < 1$, we have $V^{\triangleright}_{\mathrm{loc}}(s; \alpha) < \alpha$ for large enough α.

When $r(s) < 0$, we apply the bound

$$V^{\triangleright}_{\mathrm{loc}}(s; \alpha) \le r(s) + \mathbb{E}[\max(R(s) - r(s), \alpha)]. \tag{A.5}$$

This holds because the right-hand side is the value that would be achievable if after playing $\triangleright$ once, the agent learned the Markov chain's full trajectory and clairvoyantly chose between the alternative α and the remaining trajectory reward $R(s) - r(s)$. Because $\mathbb{E}[R(s)] < \infty$, we have

$$\lim_{\alpha \to \infty} \left(V^{\triangleright}_{\mathrm{loc}}(s; \alpha) - \alpha \right) \le r(s) + \lim_{\alpha \to \infty} \mathbb{E}[\max(R(s) - \alpha, 0)] = r(s). \tag{A.6}$$

Because $r(s) < 0$, this means $V^{\triangleright}_{\mathrm{loc}}(s; \alpha) < \alpha$ for large enough α. $\square$

We now introduce the machinery needed to state the MCS value function. At a high level, this value function is a certain combination of the value functions of the Markov chains' local MDPs. The specific combination is most easily written and understood using a probabilistic interpretation. Our next step is therefore to express a Markov chain's local MDP value function probabilistically. Throughout, let $(S, \partial S, p, r)$ be a Markov chain satisfying Assumptions A.2 and 3.3, and note again by Lemma A.3 that all Gittins indices in question are finite.

Definition A.4. The *surrogate value* of non-terminal state $s \in S \setminus \partial S$, denoted $\Gamma(s)$, is the random variable equal to the minimum Gittins index on the trajectory from s (inclusive) to a terminal state in ∂S (exclusive). More formally, letting $s(0), s(1), \dots$ be a trajectory of the Markov chain with $s(0) = s$, and letting τ be the hitting time of ∂S, we have

$$\Gamma(s) = \min_{0 \le u < \tau} G(s(u)). \tag{A.7}$$

From (A.7), it is immediate that a state's surrogate value is at most its Gittins index:

$$\mathbb{P}[\Gamma(s) \le G(s)] = 1. \tag{A.8}$$

The following lemma strengthens this by relating the full distribution function of $\Gamma(s)$ to the local MDP. In what follows, we use the term *almost all* in the Lebesgue sense.

Lemma A.5. *The surrogate value and local MDP's value function satisfy*

$$\mathbb{P}[\Gamma(s) \le \alpha] = \frac{\mathrm{d}}{\mathrm{d}\alpha} V^*_{\mathrm{loc}}(s; \alpha) \qquad \text{for almost all } \alpha \in \mathbb{R} \tag{A.9}$$

$$\mathbb{E}[\max(\Gamma(s), \alpha)] = V^*_{\mathrm{loc}}(s; \alpha) \qquad \text{for all } \alpha \in \mathbb{R}. \tag{A.10}$$

Proof. We first show (A.9). By the reasoning in the proof of Lemma 3.5, $\frac{\mathrm{d}}{\mathrm{d}\alpha} V^*_{\mathrm{loc}}(s; \alpha)$, which exists for almost all α by convexity, is the probability that an optimal policy for the local MDP ever plays $\square$.[9] But we know that an optimal policy for the local MDP is to play $\square$ if and only if the current state s' has $G(s') \le \alpha$. The probability this policy ever plays $\square$ is the probability that some state s' on the trajectory from s to a terminal state has $G(s') \le \alpha$. By (A.7), this happens if and only if $\Gamma(s) \le \alpha$, so the probability is $\mathbb{P}[\Gamma(s) \le \alpha]$, as desired.

Having shown (A.9), we move on to showing (A.10). Recall the definition of $G(s)$ in (3.5):

$$G(s) = \sup\{g \in \mathbb{R} : V^*_{\mathrm{loc}}(s; g) > g\} = \inf\{g \in \mathbb{R} : V^*_{\mathrm{loc}}(s; g) = g\}. \tag{A.11}$$

and recall also from Lemma A.3 that $G(s) < \infty$. Combining (A.9) and (A.11) shows that for all $\alpha \ge G(s)$, we have

$$\mathbb{E}[\max(\Gamma(s), \alpha)] = \alpha = V^*_{\mathrm{loc}}(s; \alpha) \tag{A.12}$$

so (A.10) holds for $\alpha \ge G(s)$. By the tail integral formula, for almost all α, we get

$$\frac{\mathrm{d}}{\mathrm{d}\alpha} \mathbb{E}[\max(\Gamma(s), \alpha)] = \mathbb{P}[\Gamma(s) \le \alpha]. \tag{A.13}$$

Combining this with (A.9) implies (A.10) holds for $\alpha < G(s)$, too. $\square$

[9] One can say more: if the derivative fails to exist, it is because there are multiple optimal policies with different probabilities of playing $\square$. But left and right derivatives still exist in this case, and Xie et al. [114, Appendix B.6] show that they are the minimum and maximum probabilities of playing $\square$ among optimal policies.

We are now ready to prove Theorem 3.7. We consider an undiscounted MCS instance with n Markov chains $(S_i, \partial S_i, p_i, r_i)$, all satisfying Assumptions A.1, A.2, and 3.3. As previously discussed, our approach will be to guess the form of the MCS value function, then show that it satisfies the Bellman equation.

Let $V_{\text{MCS}} : S_1 \times \cdots \times S_n \to \mathbb{R}$ be the value function of MCS of this MCS instance. Because MCS terminates once any of its constituent Markov chains terminates, V^*_{MCS} satisfies the terminal condition

$$V^*_{\text{MCS}}(s_1, \dots, s_n) = 0 \qquad \text{if } (s_1, \dots, s_n) \in \partial S_{\text{MCS}} \tag{A.14}$$

where the latter is equivalent to $s_i \in \partial S_i$ for some i.

To solve the MCS Bellman equation (A.3), we need to consider what we know about the operators $\mathcal{B}_i$. The key piece of information is that $\mathcal{B}_i$ features in the Bellman equation for the local MDP of Markov chain i, as spelled out in Lemma A.6 below. This suggests that if we define V^*_{MCS} by combining the local MDP value functions $V^*_{\text{loc},i}$ in a suitable manner, we can control how V^*_{MCS} interacts with the Bellman operators $\mathcal{B}_i$. We do this in Theorem A.7 below, which relies crucially on the surrogate value interpretation of $V^*_{\text{loc},i}$ (Lemma A.5).

Lemma A.6. *For all $s_i \in S_i \setminus \partial S_i$ and all $\alpha \in \mathbb{R}$, we have*

$$V^*_{\text{loc},i}(s_i; \alpha) = \max\big(\alpha, \mathcal{B}_i V^*_{\text{loc},i}(s_i; \alpha)\big) = \begin{cases} \alpha & \text{if } \alpha \geq G_i(s_i) \\ \mathcal{B}_i V^*_{\text{loc},i}(s_i; \alpha) & \text{if } \alpha \leq G_i(s_i) \end{cases} \tag{A.15}$$

with equality between the two branches, namely $\alpha = \mathcal{B}_i V^*_{\text{loc},i}(s_i; \alpha)$ if and only if $\alpha = G_i(s_i)$.

Proof. The first equality is the Bellman equation of the local MDP for Markov chain i with alternative α, and the rest follows from Definition 3.6. Specifically, playing $\square$, which yields value α, is optimal if and only if $\alpha \geq G_i(s_i)$; and playing $\triangleright$, which yields expected value $\mathcal{B}_i V^*_{\text{loc},i}(s_i; \alpha)$, is optimal if and only if $\alpha \leq G_i(s_i)$. $\square$

Theorem A.7. *The MCS optimal value function is*

$$V^*_{\text{MCS}}(s_1, \dots, s_n) = \begin{cases} \mathbb{E}\left[\max\limits_{i \in \{1,\dots,n\}} \Gamma_i(s_i)\right] & \text{if } s_i \in S_i \setminus \partial S_i \text{ for all } i \\ 0 & \text{otherwise.} \end{cases} \tag{A.16}$$

Moreover, the maximizing actions in the Bellman equation (A.3) *are those with maximal Gittins index:*

$$\operatorname*{arg\,max}_{i \in \{1,\dots,n\}} \mathcal{B}_i V^*_{\text{MCS}}(s_1, \dots, s_n) = \operatorname*{arg\,max}_{i \in \{1,\dots,n\}} G_i(s_i) \tag{A.17}$$

and the value V^*_{MCS} is achieved by the Gittins policy.

Proof. For the purposes of this proof, let us take V^*_{MCS} to be the function *defined* by the expression (A.16), which we emphasize is not assumed to be the optimal value. By Assumption A.1, if we can show that this ansatz satisfies the Bellman equation (A.3) and is the value of some policy π^*, then V^*_{MCS} is indeed the true optimal value function.

Suppose $s_i \in S_i \setminus \partial S_i$ for all i. It suffices to show that for all i,

$$\mathcal{B}_i V^*_{\text{MCS}}(s_1, \dots, s_n) \leq V^*_{\text{MCS}}(s_1, \dots, s_n) \tag{A.18}$$

with equality if and only if Markov chain i has maximal Gittins index, meaning $G_i(s_i) \geq G_j(s_j)$ for all $j \neq i$.

Below, to reduce clutter, we shorten $G_i(s_i)$ to G_i and shorten $\Gamma_i(s_i)$ to Γ_i. Recall throughout that surrogate values of different Markov chains are mutually independent. Let

$$\Gamma_{\neq i} = \max_{j \neq i} \Gamma_j. \tag{A.19}$$

We first check that $\mathcal{B}_i V^*_{\mathrm{MCS}}$ is well defined: this follows by the fact that rewards are bounded in absolute value (Assumption 3.3). Next, using Lemma A.5, we can write the proposed MCS value function in terms of $\Gamma_{\neq i}$:

$$V^*_{\mathrm{MCS}}(s_1, \dots, s_n) = \mathbb{E}[\max(\Gamma_i, \Gamma_{\neq i})] = \mathbb{E}[V^*_{\mathrm{loc},i}(s_i, \Gamma_{\neq i})]. \tag{A.20}$$

One intuition is that (A.20) shows the perspective of i on the MCS instance, where the value of playing any action other than i has been summarized by the random variable $\Gamma_{\neq i}$. Applying the (affine) Bellman operator $\mathcal{B}_i$ and using Lemma A.6 yields

$$\begin{aligned}
\mathcal{B}_i V^*_{\mathrm{MCS}}(s_1, \dots, s_n) &= \mathcal{B}_i \mathbb{E}[V^*_{\mathrm{loc},i}(s_i; \Gamma_{\neq i})] && \text{(A.21)} \\
&= \mathbb{E}[\mathcal{B}_i V^*_{\mathrm{loc},i}(s_i; \Gamma_{\neq i})] && \text{(A.22)} \\
&\le \mathbb{E}[V^*_{\mathrm{loc},i}(s_i; \Gamma_{\neq i})] && \text{(A.23)} \\
&= V^*_{\mathrm{MCS}}(s_1, \dots, s_n) && \text{(A.24)}
\end{aligned}$$

with equality if and only if $\mathbb{P}[\Gamma_{\neq i} \le G_i] = 1$. Here, changing the order of expectations when going from (A.21) to (A.22) follows by Fubini's Theorem, using absolute integrability of the reward sum (Assumption 3.3). One subtlety is that while we have assumed s_i is non-terminal, it might be that the next state $s_i' \sim p_i(s_i)$, which is used implicitly when we apply the Bellman operator, is terminal. To handle this, one can check that (A.20) holds even when s_i is terminal, in which case it becomes $0 = 0 = 0$.

It remains only to show that $\mathbb{P}[\Gamma_{\neq i} \le G_i] = 1$ if and only if $G_i \ge G_j$ for all $j \ne i$. The *if* direction follows from (A.8) and (A.19), which together imply

$$\Gamma_{\neq i} = \max_{j \ne i} \Gamma_j \le \max_{j \ne i} G_j \le G_i \tag{A.25}$$

with probability 1. For the *only if* direction, because $\Gamma_{\neq i} \ge \Gamma_j$ for all j, it suffices to show that if $G_i < G_j$ for some j, then $\mathbb{P}[\Gamma_j \le G_i] < 1$. Lemmas A.5 and 3.5 together imply that for all $\alpha \in \mathbb{R}$, the following three expressions are equivalent:

$$\mathbb{P}[\Gamma_j \le \alpha] < 1 \qquad \frac{\mathrm{d}}{\mathrm{d}\alpha} V^*_{\mathrm{loc}}(s_j; \alpha) < 1 \qquad \alpha < G_j. \tag{A.26}$$

The desired statement follows by plugging in $\alpha = G_i$.

Finally, we argue that the value achieved by the Gittins policy is indeed given by V^*_{MCS}: in some sense, this ensures that V^*_{MCS} is not a spurious solution to the Bellman equation. Denote the Gittins policy, under an arbitrary tie-breaking rule, by $\pi^* : S_1 \times \cdots \times S_n \to \{1, \dots, n\}$, where, as with the notation used before, we do not assume optimality. Let $s(0), \dots, s(\tau)$, without subscripts, be the trajectory of the MCS state vector under π, where $s(0) = s = (s_1, \dots, s_n)$ is the initial state, and τ is the time when MCS terminates, namely the hitting time of $\partial S_{\mathrm{MCS}}$. By the preceding argument, we know that

$$V^*_{\mathrm{MCS}}(s) = \max_{i \in \{1, \dots, n\}} \mathcal{B}_i V^*_{\mathrm{MCS}}(s) = \mathcal{B}_{\pi^*(s)} V^*_{\mathrm{MCS}}(s). \tag{A.27}$$

For every finite $T \ge 1$, iterating the above expression and using the fact that $V^*_{\mathrm{MCS}}(s(\tau)) = 0$ by (A.14) gives

$$V^*_{\mathrm{MCS}}(s) = \mathbb{E}\left[\sum_{t=0}^{\min(T,\tau)-1} r_{\pi^*(s(t))}(s_{\pi^*(s(t))}) + V^*_{\mathrm{MCS}}(s(\min(T, \tau)))\right]. \tag{A.28}$$

Taking the $T \to \infty$ limit and applying dominated convergence via Assumption 3.3 yields

$$V^*_{\mathrm{MCS}}(s) = \mathbb{E}\left[\sum_{t=0}^{\tau-1} r_{\pi^*(s(t))}(s_{\pi^*(s(t))})\right]. \tag{A.29}$$

The right-hand side is exactly the value achieved by the Gittins policy, as desired. □

Remark A.8. As discussed in Section 5.2, one cannot in general extend Theorems A.7 and 3.7 from MCS to MDP selection. However, part of the argument still goes through:

(a) On one hand, the *Bellman inequality*, namely (A.21)–(A.24), continues to hold. This implies the value function guess in (A.16) is an upper bound for the MDP selection value function, a fact which plays a crucial role in many approximation results for MDP selection [7, 16, 20, 23, 27, 87].

(b) On the other hand, the *Bellman equation* generally fails: there need not exist an action that makes the Bellman inequality tight. The issue is that in (A.22), the random realization of $\Gamma_{\neq i}$ might influence which action is optimal, but we have to choose an action without knowledge of $\Gamma_{\neq i}$—recall that, here, Bellman operators are now parameterized by a pair (i, a), where a is an action in MDP i. One of the few cases when the Bellman equation holds despite this obstacle is when all the MDPs satisfy the Whittle condition (see Section 5.2.2).

A.2. Generalization: finishing multiple Markov chains

We now sketch how the statement and proof of Theorem A.7 can be generalized from MCS to MCS-k. The overall approach we take is the similar to that of Scully and Doval [87], although that work considers just the Pandora's box setting and its optional inspection variant. See Singla [96] and Gupta et al. [55] for alternative proofs that also yield results for variants of MCS-k involving combinatorial constraints.

The main difference here is that instead of the MCS value function involving the maximum single surrogate value, the argument involves the *sum of the k greatest surrogate values*. Specifically, when there are k items still to be selected, we have

$$V^*_{\text{MCS-}k}(s_1, \dots, s_n) = \mathbb{E}\left[\max_{\substack{I \subseteq \{1,\dots,n\} \\ |I|=k}} \sum_{i \in I} \Gamma_i(s_i)\right]. \tag{A.30}$$

More generally, when exactly $\ell \le k$ of the Markov chains are in terminal states, the same formula holds, except we use only the $k - \ell$ greatest surrogate values, meaning we replace $|I| = k$ by $|I| = k - \ell$. In particular, because the empty sum is 0, this gives a boundary condition $V_{\text{MCS-}k}(s_1, \dots, s_n) = 0$ when there are k Markov chains in terminal states.

To show that (A.30) gives a solution to the MCS-k Bellman equation, we use an analogue of (A.20), expressing the MCS-k value function in terms of the local MDP's value function. To reduce clutter, we again shorten $G_i(s_i)$ to G_i and $\Gamma_i(s_i)$ to Γ_i, and we assume without loss of generality that none of the Markov chains are in terminal states. Letting

$$\Gamma_{\text{with } i} = \max_{\substack{I \subseteq \{1,\dots,n\} \setminus \{i\} \\ |J|=k-1}} \sum_{j \in J} \Gamma_j \qquad \Gamma_{\text{without } i} = \max_{\substack{I \subseteq \{1,\dots,n\} \setminus \{i\} \\ |J|=k}} \sum_{j \in J} \Gamma_j \tag{A.31}$$

we can rewrite (A.30) as

$$\begin{aligned} V^*_{\text{MCS-}k}(s_1, \dots, s_n) &= \mathbb{E}[\max(\Gamma_i + \Gamma_{\text{with } i}, \Gamma_{\text{without } i})] && \text{(A.32)} \\ &= \mathbb{E}[\max(\Gamma_i, \Gamma_{\text{without } i} - \Gamma_{\text{with } i})] + \mathbb{E}[\Gamma_{\text{with } i}] && \text{(A.33)} \\ &= \mathbb{E}[V^*_{\text{loc},i}(s_i; \Gamma_{\text{without } i} - \Gamma_{\text{with } i})] + \mathbb{E}[\Gamma_{\text{with } i}]. && \text{(A.34)} \end{aligned}$$

Just as (A.20) can be thought of as i's perspective on MCS, (A.34) can be thought of as i's perspective on MCS-k. Because neither $\Gamma_{\text{with } i}$ nor $\Gamma_{\text{without } i}$ depend on s_i, applying the Bellman operator and then reasoning similarly to the end of proof of Theorem A.7 shows the that Bellman equation holds, with playing i being optimal if and only if G_i is among the k greatest Gittins indices.

A.3. Comparison to other optimality proofs

There are many proofs of the optimality of the Gittins policy in the literature, taking a variety of approaches and covering a variety of settings. We discuss just a few approaches here, referring the reader to Gittins et al. [45, Section 2.12] for a more comprehensive history.

To the best of our knowledge, there is no single theorem that unifies all of the known optimality results, particularly when infinite spaces, continuous time, or inflation (as in Section 5.3) are involved. We consider providing such a unifying account to be an open problem.

For settings where Markov chains have finitely many states, a common approach is to use *induction on the number of states*. See Tsitsiklis [100] for a particularly accessible proof of this form. Roughly speaking, these proofs show that the state of maximal index should be prioritized over all others, then reduce the Markov chain involved by removing that state. One advantage of this inductive approach is that it easily generalizes to branching bandits [107]. Another advantage is that the proofs are elegant and elementary. On the other hand, a key difficulty is that they are hard to generalize to infinite state spaces.

The proof we give above is based primarily on that of Whittle [109], which is the first *dynamic programming* proof of the Gittins policy's optimality. In particular, Whittle [109] discovered the form of the MCS value function as a combination of all the Markov chains' local MDP value functions, though without the probabilistic interpretation of Theorem A.7. Unlike the inductive approach, the dynamic programming approach works essentially whenever dynamic programming itself works—which our argument took as assumption.

An advantage of the dynamic programming approach is that it can be easily extended to yield results about MDP selection (Section 5.2.2), including optimality when the MDPs satisfy the Whittle condition [48, 109] and approximation results when they do not [7, 23, 27, 87]. Even more generally, in the setting where independent Markov chains are replaced by a generalization called *interleaved filtrations*, essentially the same construction still works [74], and extends to continuous time [9, 61, 62, 75].

Our proof is also influenced by proofs based on an *economic argument*, as pioneered by Weber [104] for the discounted setting and later replicated in undiscounted settings [21, 27, 34, 55, 66]. In the undiscounted setting, the argument consists of three main steps:

1. For each Markov chain i, define a random variable called its *surrogate value*, denoted Γ_i. This is exactly our Definition A.4.
2. Show that, in MCS, the expected value achieved by any policy π is at most $\mathbb{E}[\Gamma_{i_\pi}]$, where i_π is the identifier of the Markov chain that π finishes, noting that there is always exactly one such Markov chain under Assumption 3.3(a).
3. Show that the above inequality is in fact an equality when π is the Gittins policy.
4. Finally, observe that the Gittins policy *always* finishes the Markov chain of maximal surrogate value, and thus always obtains surrogate value $\max_{i\in\{1,\dots,n\}} \Gamma_i$.

The economic argument thus gives another interpretation of the value function we derive in Theorem A.7: the expected value achieved by Gittins is $\mathbb{E}[\max_{i\in\{1,\dots,n\}} \Gamma_i]$, and every other policy π achieves at most $\mathbb{E}[\Gamma_{i_\pi}]$. One can therefore loosely think of Γ_i as a kind of amortized value for each chain, hence the name *surrogate value* for it.

Finally, two more proof techniques, which are related to each other, are those based on the *achievable region* approach [11–14, 32] and the *WINE (work integral number equality)* queueing identity [86, 88, 90]. The main appeal of these approaches is that they can also be used to prove guarantees on the approximate optimality of approximate index policies, and to prove performance bounds for the multiserver case where multiple Markov chains are advanced at every time step [14, 49, 51]. We refer the reader to Dacre et al. [32] for a primer on the achievable region approach and to Scully [86] for a primer on WINE. There is not yet a full account of the relationship between these approaches, but see Scully [86, Section 2.2.3] for some initial discussion. A full unification would likely require a version of the achievable region method that works with measure-valued linear programs.