Methods and Applications of Statistics in Clinical Trials: Planning, Analysis, and Inferential Methods. Edited by N. Balakrishnan.
© 2014 John Wiley & Sons, Inc. Published 2014 by John Wiley & Sons, Inc.

24 Multi-Armed Bandits, Gittins Index, and Its Calculation

Jhelum Chakravorty and Aditya Mahajan

24.1 Introduction

Multi-armed bandit is a colorful term that refers to the dilemma faced by a gambler playing in a casino with multiple slot machines (which were colloquially called one-armed bandits). What strategy should a gambler use to pick the machine to play next? It is the one for which the posterior mean of winning is the highest and thereby maximizes current expected reward, or the one for which the posterior variance of winning is the highest, and has the potential to maximize the future expected reward. Similar *exploration vs exploitation trade-offs* arise in various application domains including clinical trials [5], stochastic scheduling [25], sensor management [33], and economics [4].

Clinical trials fit naturally into the framework of multi-armed bandits and have been a motivation for their study since the early work of Thompson [31]. Broadly speaking, there are two approaches to multi-armed bandits. The first, following Bellman [2], aims to maximize the expected total discounted reward over an infinite horizon. The second, following Robbins [27], aims to minimize the rate of regret for the expected total reward over a finite horizon. In some of the literature, the first setup is called geometric discounting while the second is called uniform discounting. For a large time, the multi-armed bandit problem, in both formulations, was considered unsolvable until the pioneering work of Gittins and Jones [13] for the discounted setup and that of Lai and Robbins [22] for the regret setup characterized the nature of the optimal strategy.

In this chapter, we restrict our attention to the discounted setup; we refer the reader to a recent survey by Bubeck and Cesa-Bianchi [9] and references therein for the regret setup and to Kuleshov and Precup [20], who discuss the regret setup in the context of clinical trials.

24.2 Mathematical Formulation of Multi-Armed Bandits

Multi-armed bandit (MAB) is a sequential decision problem in which, at each time,

a player must play one among n available bandit process. In the simplest setup, a bandit process is a controlled Markov process. The player has two controls: either play the process or not. If the player chooses to play the bandit process i, $i = 1,\ldots, n$, the state of the bandit process i evolves in a Markovian manner while the state of all other processes remains frozen (i.e., it does not change). Such a bandit process is called a *Markovian* bandit process. More sophisticated setups assume that when a bandit processes is played, its state evolves according to an arbitrary stochastic process. To focus on the key conceptual ideas, we restrict our attention to Markovian bandit processes. For more general bandit processes, the description of the solution concept and its computation are more involved.

Formally, let $\{X_t^i\}_{t=1}^{\infty}$ denote the bandit process. The state X_t^i of the bandit process i, $i = 1, \ldots, n$, takes values in an arbitrary space $\mathcal{X}^i$. For simplicity, we assume in most of the discussion in this chapter that $\mathcal{X}^i$ is finite or countable.

Let $\mathbf{u}_t = (u_t^1, \ldots, u_t^n)$ denote the decision made by the player at time t. The component u_t^i is binary valued and denotes whether the player chooses to play the bandit process i ($u_t^i = 1$) or not ($u_t^i = 0$). Since the player may only choose to play one bandit process at each time, $\mathbf{u}_t$ must have only one nonzero component, or equivalently

$$\sum_{i=1}^{n} u_t^i = 1, \quad \forall t.$$

Let $\mathcal{U} \in \{0,1\}^n$ denote all vectors with this property. The collection $\{\mathbf{X}_t = (X_t^1, \ldots, X_t^n)\}_{t=0}^{\infty}$ evolves as follows: $\forall i = 1, \ldots, n$

$$X_{t+1}^i = \begin{cases} f^i(X_t^i, W_t^i), & \text{if } u_t^i = 1, \\ X_t^i, & \text{if } u_t^i = 0, \end{cases}$$

where $\{W_t^i\}_{t=0}^{\infty}$, $i = 1,\ldots,n$, are mutually independent i.i.d. processes. Thus, when $u_t^i = 1$, X_t^i evolves in a Markovian manner; otherwise it remains frozen.

When bandit process i is played, it yields a reward $r^i(X_t^i)$ and all other processes yield no reward. The objective of a player is to choose a decision strategy $g = \{g_t\}_{t=0}^{\infty}$, where $g_t : \prod_{i=1}^{n} \mathcal{X}^i \to \mathcal{U}$, so as to maximize the expected total discounted reward

$$\mathbb{E}^g \Big[\sum_{t=0}^{\infty} \beta^t \sum_{i=1}^{n} r^i(X_t^i) u_t^i \,\Big|\, \mathbf{X}_0 = \mathbf{x}_0 \Big]$$

where $\mathbf{x}_0 = (x_0^1, \ldots, x_0^n)$ is the initial starting state of all bandit processes and $\beta \in (0,1)$ is the discount factor.

24.2.1 An Example

Consider an adaptive clinical trial with n treatment options. When the tth patient comes for treatment, she may be prescribed any one of the n available options based on the results of the previous trials. The sequence of success and failure of each treatment is an Bernoulli process with an unknown success probability s^i, $i = 1, \ldots, n$. The objective is to design an adaptive strategy that maximizes the expected total discounted successful treatments.

The above adaptive clinical trial may be viewed as a MAB problem as follows. Suppose the prior distribution of s^i has a beta distribution that is independent across treatments. Then the posterior distribution of s^i, which is calculated using Bayes' rule, is also a beta distribution, say Beta(p_t^i, q_t^i) at time t. Therefore, each treatment may be modeled as a Markovian bandit process where the state X_t^i is (p_t^i, q_t^i), the state update function f corresponds to Bayes update of the posterior distribution, and the reward function $r^i(X_t^i) = p_t^i/(p_t^i + q_t^i)$ captures the probability of success.

24.2.2 Solution Concept and the Gittins Index

One possible solution concept for the MAB problem is to set it up as a Markov decision process (MDP) and use Markov decision theory [26] to solve it. However, such an approach does not scale well with the number of bandit processes because of the *curse of dimensionality*. To see this, assume that the state space $\mathcal{X}^i$ of each bandit process is a finite set with m elements. Then the state space of the resultant MDP, which is the space of realizations of $\mathbf{X}_t$, has size m^n. The solution complexity of solving a MDP is proportional to the size of the state space. Hence the complexity of solving MAB using Markov decision theory increases exponentially with the number of bandit processes. A key breakthrough for the MAB problem was provided by Gittins and Jones [13] and Gittins [15], who showed that instead of solving the n-dimensional MDP with state-space $\prod_{i=1}^{n} \mathcal{X}^i$, an optimal solution is obtained by solving n 1-dimensional optimization problems: for each bandit i, $i = 1, \ldots, n$, and for each state $x^i \in \mathcal{X}^i$, compute

$$\nu^i(x^i) = \max_{\tau > 0} \frac{\mathbb{E}\left[\sum_{t=0}^{\tau} \beta^t r^i(X_t^i) \,\middle|\, X_0^i = x^i\right]}{\mathbb{E}\left[\sum_{t=0}^{\tau} \beta^t \,\middle|\, X_0^i = x^i\right]} \tag{1}$$

where τ is a $\{\sigma(X_1^i, \ldots, X_t^i)\}_{t=1}^{\infty}$ measurable stopping time. The function $\nu^i(x^i)$ is called *Gittins index* of bandit process i at state x^i. The optimal τ in (1), which is called the optimal stopping time at x^i, is given by the hitting time $T(S(x^i))$ of a stopping set $S(x^i) \subseteq \mathcal{X}^i$, that is, the first time the bandit process enters the set $S(x^i)$. Algorithms to compute the Gittins index of a bandit process are presented in Section 24.3.

Gittins and Jones [13] and Gittins [15] characterized the optimal strategy as follows:

- *At each time, play the arm with the highest Gittins index.*

Thus, to implement the optimal strategy, compute and store the Gittins index $\nu^i(x^i)$ of all states $x^i \in \mathcal{X}^i$ of all bandit processes i, $i = 1, \ldots, n$. Off-line algorithms that compute the Gittins index of all states of a bandit process are presented in Section 24.3.

An equivalent interpretation of the Gittins index strategy is the following:

- *Pick the arm with the highest Gittins index and play it until its optimal stopping time (or equivalently, until it enters the corresponding stopping set) and repeat.*

Thus, an alternative way to implement the optimal strategy is to compute the Gittins index $\nu^i(x_t^i)$ and the corresponding stopping time [or equivalently, the corresponding stopping set $S(x^i)$] for the current state x_t^i of all bandit processes i, $i = 1, \ldots, n$. On-line algorithms that compute the Gittins index and the corresponding stopping set for an arbitrary state of a bandit process are presented in Section 24.4.

Off-line implementation is simpler and more convenient for bandit processes with finite and moderately sized state space. On-line implementation becomes more convenient for bandit processes with large or infinite state space.

24.2.3 Salient Features of the Model

As explained by Gittins [15], MAB problems admit a simpler solution than general multistage decision problems because they can be optimally solved with forward induction. In general, forward induction is not optimal and one needs to resort to backward induction to find the optimal strategy. Forward induction is optimal for

MAB setup because it has the following features:

1. The player plays only one bandit process, and that process evolves in an uncontrolled manner.
2. The processes that are not played are frozen.
3. The current reward depends only on the current state of the process that is played and is not influenced by the state of the remaining processes and the history of previous plays.

Because the above features, decisions made at each stage are not irrevocable and hence forward induction is optimal. On the basis of the above insight, Gittins [15] proved the optimality of the index strategy, using an interchange argument. Since then, various other proofs of the Gittins index strategy have been proposed (see [14] for a detailed summary).

Several variations of MAB problems have been considered in the literature. These variations remove some of the above features, and, as such, index strategies are not necessarily optimal for these variations. We refer the reader to the survey by Mahajan and Teneketzis [23] and references therein for details on the variations of the MAB problem.

24.3 Off-Line Algorithms for Computing Gittins Index

Since the Gittins index of a bandit process depends just on that process, we drop the label i and denote the bandit process by $\{X_t\}_{t=0}^{\infty}$.

Off-line algorithms use the following property of the Gittins index. The Gittins index $\nu : \mathcal{X} \to \mathbb{R}$ induces a total order $\succeq$ on $\mathcal{X}$ that is given by

$$\forall a, b \in \mathcal{X}, \quad a \succeq b \iff \nu(a) \geq \nu(b).$$

Using this total order, for any state $a \in \mathcal{X}$, the state space $\mathcal{X}$ may be split into two sets

$$C(a) = \{b \in \mathcal{X} : b \succ a\},$$
$$S(a) = \{b \in \mathcal{X} : a \succeq b\}.$$

These sets are, respectively, called the *continuation set* and the *stopping set*. The rationale behind the terminology is that if we start playing a bandit process in state a, then it is optimal to continue playing the bandit process in all states $C(a)$ because for any $b \in C(a)$, $\nu(b) > \nu(a)$. Thus, the stopping time that corresponds to starting the bandit process in state a is the hitting time $T(S(a))$ of set $S(a)$, that is, the first time the bandit process enters set $S(a)$. Using this characterization, the expression for Gittins index (1) simplifies to

$$\nu(a) = \max_{S(a) \subseteq \mathcal{X}} \frac{\mathbb{E}\left[\sum_{t=0}^{T(S(a))} \beta^t r(X_t) \,\middle|\, X_0 = a\right]}{\mathbb{E}\left[\sum_{t=0}^{T(S(a))} \beta^t \,\middle|\, X_0 = a\right]} \tag{2}$$

where $T(S(a)) = \inf\{t > 0 : X_t \in S(a)\}$ is the hitting time of set $S(a)$. The off-line algorithms use this simplified expression to compute the Gittins index.

For ease of exposition, assume that X_t takes values in a finite space $\mathcal{X} = \{1, \ldots, m\}$. Most of the algorithms extend to countable state spaces under appropriate conditions. Denote the $m \times m$ transition probability matrix corresponding to the Markovian update of the bandit process by $\mathbf{P} = [\mathbf{P}_{a,b}]$ and represent the reward function using a $m \times 1$ vector $\mathbf{r}$, i.e. $\mathbf{r}_a = r(a)$. Furthermore, let $\mathbf{1}$ be the $m \times 1$ vector of ones, $\mathbf{0}_{m\times 1}$ be the $m \times 1$ vector of zeros, $\mathbf{I}$ be the $m \times m$ identity matrix, and $\mathbf{0}_{m\times m}$ be the $m \times m$ matrix of zeros.

24.3.1 Largest-Remaining-Index Algorithm (Varaiya, Walrand, and Buyukkoc)

Varaiya, Walrand, and Buyukkoc [32] presented an algorithm to compute the Gittins index, which we refer to as the *largest-remaining-index algorithm*. The key idea behind this algorithm is to identify the states in $\mathcal{X}$ according to the decreasing order

$$\alpha_1 \succeq \alpha_2 \succeq \cdots \succeq \alpha_m$$

where $(\alpha_1, \ldots, \alpha_m)$ is a permutation of $(1, \ldots, m)$.

The algorithm proceeds as follows:

Initialization: The first step of the algorithm is to identify the state α_1 with the highest Gittins index. Since α_1 has the highest Gittins index, $S(\alpha_1) = \mathcal{X}$. Substituting this in (2), we get that $\nu(\alpha_1) = r(\alpha_1) = \mathbf{r}_{\alpha_1}$. Then, choose

$$\alpha_1 = \arg\max_{a \in \mathcal{X}} \mathbf{r}_a$$

where ties are broken arbitrarily. The corresponding Gittins index is

$$\nu(\alpha_1) = \mathbf{r}_{\alpha_1}.$$

Recursion step: After the states α_1, $\ldots$, α_{k-1} have been identified, the next step is to identify the state α_k with the kth largest Gittins index. Even though α_k is not known, we know that $C(\alpha_k) = \{\alpha_1, \ldots, \alpha_{k-1}\}$ and $S(\alpha_k) = \mathcal{X} \setminus \{\alpha_1, \ldots, \alpha_{k-1}\}$. Substituting this in (2), we can compute $\nu(\alpha_k)$ as follows. Define the $m \times m$ matrix by $\forall a, b \in \mathcal{X}$,

$$\mathbf{Q}_{a,b}^{(k)} = \begin{cases} \mathbf{P}_{a,b}, & \text{if } b \in C(\alpha_k), \\ 0, & \text{otherwise;} \end{cases} \tag{3}$$

and define the $m \times 1$ vectors:

$$\begin{aligned} \mathbf{d}^{(k)} &= [\mathbf{I} - \beta \mathbf{Q}^{(k)}]^{-1} \mathbf{r}, \\ \mathbf{b}^{(k)} &= [\mathbf{I} - \beta \mathbf{Q}^{(k)}]^{-1} \mathbf{1}. \end{aligned} \tag{4}$$

Then, choose

$$\alpha_k = \arg\max_{a \in C(\alpha_k)} \frac{\mathbf{d}_a^{(k)}}{\mathbf{b}_a^{(k)}}$$

where ties are broken arbitrarily. The corresponding Gittins index is

$$\nu(\alpha_k) = \frac{\mathbf{d}_{\alpha_k}^{(k)}}{\mathbf{b}_{\alpha_k}^{(k)}}.$$

Computational complexity: The performance bottleneck of this algorithm is the two systems of the linear equations (4) that need to be solved at each step. At step k, each system effectively has k variables, so solving each requires $(2/3)k^3 + O(k^2)$ arithmetic operations. Thus, overall this algorithm requires

$$2 \sum_{k=1}^{m} \left[\tfrac{2}{3}k^3 + O(k^2)\right] = \tfrac{1}{4}m^4 + O(m^3)$$

arithmetic operations. This algorithm has the worst computational complexity of all the algorithms presented.

Example 24.3.1 Consider a bandit process with state space $\mathcal{X} = \{1, 2, 3, 4\}$, reward vector $\mathbf{r} = [16 \quad 19 \quad 30 \quad 4]^\intercal$, and probability transition matrix

$$\mathbf{P} = \begin{bmatrix} 0.1 & 0 & 0.8 & 0.1 \\ 0.5 & 0 & 0.1 & 0.4 \\ 0.2 & 0.6 & 0 & 0.2 \\ 0 & 0.8 & 0 & 0.2 \end{bmatrix}.$$

Let the discount factor be $\beta = 0.75$.

Using the largest-remaining-index algorithm, the Gittins index for this bandit process is calculated as follows:

Initialization: Start by identifying the state with the highest Gittins index:

$$\begin{aligned} \alpha_1 &= \arg\max_{a \in \mathcal{X}} \mathbf{r}_a = 3; \\ \nu(\alpha_1) &= \mathbf{r}_3 = 30. \end{aligned}$$

Recursion step:

1. **Step $k = 2$:** Although α_2 is not known, we know that $C(\alpha_2) = \{3\}$ and $S(\alpha_2) = \{1, 2, 4\}$. Using (3) and (4) we get

$$\mathbf{Q}_{a,b}^{(2)} = \begin{bmatrix} 0 & 0 & 0.8 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

$$\mathbf{d}^{(2)} = \begin{bmatrix} 34 \\ 21.25 \\ 30 \\ 4 \end{bmatrix}, \quad \mathbf{b}^{(2)} = \begin{bmatrix} 1.6 \\ 1.075 \\ 1 \\ 1 \end{bmatrix}.$$

Hence,

$$\alpha_2 = \arg \max_{a \in \{1,2,4\}} \frac{\mathbf{d}_a^{(2)}}{\mathbf{b}_a^{(2)}} = 1;$$

$$\nu(\alpha_2) = \frac{\mathbf{d}_{\alpha_2}^{(2)}}{\mathbf{b}_{\alpha_2}^{(2)}} = 21.25.$$

2. **Step $k = 3$:** Although α_3 is not known, we know that $C(\alpha_3) = \{1, 3\}$ and $S(\alpha_3) = \{2, 4\}$. Using (3) and (4) we get

$$\mathbf{Q}_{a,b}^{(3)} = \begin{bmatrix} 0.1 & 0 & 0.8 & 0 \\ 0.5 & 0 & 0.1 & 0 \\ 0.2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

$$\mathbf{d}^{(3)} = \begin{bmatrix} 40.719 \\ 36.977 \\ 36.108 \\ 4 \end{bmatrix}, \quad \mathbf{b}^{(3)} = \begin{bmatrix} 1.916 \\ 1.815 \\ 1.288 \\ 1 \end{bmatrix}.$$

Hence,

$$\alpha_3 = \arg \max_{a \in \{2,4\}} \frac{\mathbf{d}_a^{(3)}}{\mathbf{b}_a^{(3)}} = 2;$$

$$\nu(\alpha_3) = \frac{\mathbf{d}_{\alpha_3}^{(3)}}{\mathbf{b}_{\alpha_3}^{(3)}} = 20.372.$$

3. **Step $k = 4$:** As before, although α_4 is not known, we know that $C(\alpha_4) = \{1, 2, 3\}$ and $S(\alpha_4) = \{4\}$. As in the previous steps, using (3) and (4) we get

$$\mathbf{Q}_{a,b}^{(4)} = \begin{bmatrix} 0.1 & 0 & 0.8 & 0 \\ 0.5 & 0 & 0.1 & 0 \\ 0.2 & 0.6 & 0 & 0 \\ 0 & 0.8 & 0 & 0 \end{bmatrix},$$

$$\mathbf{d}^{(4)} = \begin{bmatrix} 54.929 \\ 43.95 \\ 58.017 \\ 30.371 \end{bmatrix}, \quad \mathbf{b}^{(4)} = \begin{bmatrix} 2.613 \\ 2.157 \\ 2.363 \\ 2.295 \end{bmatrix}.$$

Since $S(\alpha_4) = \{4\}$ has only one element, we have

$$\alpha_4 = 4, \text{ and } \nu(\alpha_4) = \frac{\mathbf{d}_{\alpha_4}^{(4)}}{\mathbf{b}_{\alpha_4}^{(4)}} = 13.236.$$

Thus, the vector of the Gittins index of this bandit process is

$$\boldsymbol{\nu} = \begin{bmatrix} 21.25 \\ 20.372 \\ 30 \\ 13.236 \end{bmatrix}.$$

24.3.2 State-Elimination Algorithm (Sonin)

Sonin [30] presented an algorithm to compute the Gittins index, which we refer to as the *state-elimination algorithm*. As with the largest-remaining-index algorithm, the main idea behind the state-elimination algorithm is to iteratively solve (2) in the decreasing order of $\succeq$. The computations are based on a relation of (2) with optimal stopping problems [29].

A simplified version of the state-elimination algorithm is presented below. See [30] for a more efficient version that also works when the discount factor β depends on the state.

Initialization: The initialization step is identical to that of the largest-remaining-index algorithm: Identify the state α_1 with highest Gittins index, which is given by

$$\alpha_1 = \arg \max_{a \in \mathcal{X}} \mathbf{r}_a$$

where ties are broken arbitrarily. The corresponding Gittins index is

$$\nu(\alpha_1) = \mathbf{r}_{\alpha_1}.$$

Step k of the recursion uses a $\tilde{m} \times \tilde{m}$ matrix $\mathbf{Q}^{(k)}$ and $\tilde{m} \times 1$ vectors $\mathbf{d}^{(k)}$ and $\mathbf{b}^{(k)}$, where $\tilde{m} = |S(\alpha_k)| = m - k + 1$. These are initialized as $\mathbf{Q}^{(1)} = \mathbf{P}$, $\mathbf{d}^{(1)} = \mathbf{r}$, and $\mathbf{b}^{(1)} = (1 - \beta)\mathbf{1}$.

Recursion step: After the states α_1, $\ldots$, α_{k-1} have been identified, the next step is to identify the state α_k with the kth largest Gittins index. Even though α_k is not known, we know that $C(\alpha_k) = \{\alpha_1, \ldots, \alpha_{k-1}\}$ and $S(\alpha_k) = \mathcal{X} \setminus \{\alpha_1, \ldots, \alpha_{k-1}\}$. Let the model in the step $k-1$ be $\mathbf{Q}^{(k-1)}$, $\mathbf{d}^{(k-1)}$ and $\mathbf{b}^{(k-1)}$. Define

$$\lambda_{k-1} = \frac{\beta}{1 - \beta \mathbf{Q}^{(k-1)}_{\alpha_{k-1}, \alpha_{k-1}}}.$$

Let $\tilde{m} = |S(\alpha_k)|$. Define $\tilde{m} \times \tilde{m}$ matrix by $\forall a, b \in S(\alpha_k)$:

$$\mathbf{Q}^{(k)}_{a,b} = \mathbf{Q}^{(k-1)}_{a,b} + \lambda_{k-1} \mathbf{Q}^{(k-1)}_{a,\alpha_{k-1}} \mathbf{Q}^{(k-1)}_{\alpha_{k-1},b}. \tag{5}$$

and define $\tilde{m} \times 1$ vectors by $\forall a \in S(\alpha_k)$

$$\begin{aligned} \mathbf{d}^{(k)}_a &= \mathbf{d}^{(k-1)}_a + \lambda_{k-1} \mathbf{Q}^{(k-1)}_{a,\alpha_{k-1}} \mathbf{d}^{(k-1)}_{\alpha_{k-1}}, \\ \mathbf{b}^{(k)}_a &= \mathbf{b}^{(k-1)}_a + \lambda_{k-1} \mathbf{Q}^{(k-1)}_{a,\alpha_{k-1}} \mathbf{b}^{(k-1)}_{\alpha_{k-1}}. \end{aligned} \tag{6}$$

Note that the entries of $\mathbf{Q}^{(k)}$, $\mathbf{d}^{(k)}$ and $\mathbf{b}^{(k)}$ are labeled according to the set $S(\alpha_k)$. For example, if $\mathcal{X} = \{1, 2, 3\}$, $\alpha_1 = 2$ then $S(\alpha_1) = \{1, 3\}$ and

$$\mathbf{Q}^{(2)} = \begin{bmatrix} \mathbf{Q}^{(2)}_{1,1} & \mathbf{Q}^{(2)}_{1,3} \\ \mathbf{Q}^{(2)}_{3,1} & \mathbf{Q}^{(2)}_{3,3} \end{bmatrix},$$

$$\mathbf{d}^{(2)} = \begin{bmatrix} \mathbf{d}^{(2)}_1 \\ \mathbf{d}^{(2)}_3 \end{bmatrix} \quad \text{and} \quad \mathbf{b}^{(2)} = \begin{bmatrix} \mathbf{b}^{(2)}_1 \\ \mathbf{b}^{(2)}_3 \end{bmatrix}.$$

After calculating $\mathbf{Q}^{(k)}$, $\mathbf{d}^{(k)}$ and $\mathbf{b}^{(k)}$, choose

$$\alpha_k = \arg \max_{a \in S(\alpha_k)} \frac{\mathbf{d}^{(k)}_a}{\mathbf{b}^{(k)}_a}$$

where ties are broken arbitrarily. The corresponding Gittins index is

$$\nu(\alpha_k) = (1 - \beta) \frac{\mathbf{d}^{(k)}_{\alpha_k}}{\mathbf{b}^{(k)}_{\alpha_k}}.$$

Computational complexity: The performance bottleneck of this algorithm is the update of matrix $\mathbf{Q}^{(k)}$ at each step. In step k, the matrix $\mathbf{Q}^{(k)}$ has $(m-k+1)^2$ elements and update of each element requires 2 arithmetic operations [since we can precompute $\lambda_{k-1} \mathbf{Q}^{(k-1)}_{a,\alpha_{k-1}}$, for all $a \in S(\alpha_k)$ before updating $\mathbf{Q}^{(k)}$]. Thus, overall this algorithm requires

$$\sum_{k=1}^{m} [2(m-k+1)^2 + O(k)] = \tfrac{2}{3} m^3 + O(m^2)$$

arithmetic operations. Our calculations differ from the $m^3 + O(m^2)$ calculations reported in [24] because [24] assumes that the update of each element of $\mathbf{Q}^{(k)}$ takes 3 arithmetic operations, but, as we argued above, this update can be done in 2 operations if we pre-multiply row α_{k-1} of $\mathbf{Q}^{(k-1)}$ with λ_{k-1}. Furthermore, in the implementation presented in [24], $\mathbf{b}^{(k)}$ is computed from $\mathbf{Q}^{(k)}$, requiring additional $O(m^2)$ arithmetic operations. The above implementation avoids those by keeping track of $\mathbf{b}^{(k)}$.

Example 24.3.2 Reconsider the bandit process of Example 24.3.1. Using the state-elimination algorithm, the Gittins index for this bandit process is calculated as follows.

Initialization: Start by identifying the state with the highest Gittins index:

$$\alpha_1 = \arg \max_{a \in \mathcal{X}} \mathbf{r}_a = 3;$$

$$\nu(\alpha_1) = \mathbf{r}_3 = 30.$$

Initialize:

$$\mathbf{Q}^{(1)} = \begin{bmatrix} 0.1 & 0 & 0.8 & 0.1 \\ 0.5 & 0 & 0.1 & 0.4 \\ 0.2 & 0.6 & 0 & 0.2 \\ 0 & 0.8 & 0 & 0.2 \end{bmatrix},$$

$$\mathbf{d}^{(1)} = \begin{bmatrix} 16 \\ 19 \\ 30 \\ 4 \end{bmatrix}, \quad \mathbf{b}^{(1)} = \begin{bmatrix} 0.25 \\ 0.25 \\ 0.25 \\ 0.25 \end{bmatrix}.$$

Recursion step:

1. **Step $k = 2$:** Although α_2 is not known, we know that $C(\alpha_2)$ is $\{3\}$, $S(\alpha_2) = \{1, 2, 4\}$, and $\lambda_1 = \beta/(1 -$

$\beta \mathbf{Q}^{(1)}_{3,3}) = 0.75$. Using (5) and (6) we get

$$\mathbf{Q}^{(2)} = \begin{array}{c} \\ 1 \\ 2 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 4 \end{array} \\ \begin{bmatrix} 0.22 & 0.36 & 0.22 \\ 0.515 & 0.045 & 0.415 \\ 0 & 0.8 & 0.2 \end{bmatrix} \end{array},$$

$$\mathbf{d}^{(2)} = \begin{array}{c} \\ 1 \\ 2 \\ 4 \end{array} \begin{bmatrix} 34 \\ 21.25 \\ 4 \end{bmatrix}, \quad \mathbf{b}^{(2)} = \begin{array}{c} \\ 1 \\ 2 \\ 4 \end{array} \begin{bmatrix} 0.4 \\ 0.269 \\ 0.25 \end{bmatrix}.$$

Hence,

$$\alpha_2 = \arg \max_{a \in S(\alpha_2)} \frac{\mathbf{d}^{(2)}_a}{\mathbf{b}^{(2)}_a} = 1;$$

$$\nu(\alpha_2) = (1-\beta)\frac{\mathbf{d}^{(2)}_{\alpha_2}}{\mathbf{b}^{(2)}_{\alpha_2}} = 21.25.$$

2. **Step $k = 3$:** Although α_3 is not known, we know that $C(\alpha_3) = \{1, 3\}$, $S(\alpha_3) = \{2, 4\}$, and $\lambda_2 = \beta/(1 - \beta \mathbf{Q}^{(2)}_{1,1}) = 0.898$. Using (5) and (6) we get

$$\mathbf{Q}^{(3)} = \begin{array}{c} \\ 2 \\ 4 \end{array} \begin{array}{c} \begin{array}{cc} 2 & 4 \end{array} \\ \begin{bmatrix} 0.21 & 0.52 \\ 0.8 & 0.2 \end{bmatrix} \end{array},$$

$$\mathbf{d}^{(3)} = \begin{array}{c} \\ 2 \\ 4 \end{array} \begin{bmatrix} 36.98 \\ 4 \end{bmatrix}, \quad \mathbf{b}^{(3)} = \begin{array}{c} \\ 2 \\ 4 \end{array} \begin{bmatrix} 0.45 \\ 0.25 \end{bmatrix}.$$

Hence,

$$\alpha_3 = \arg \max_{a \in S(\alpha_3)} \frac{\mathbf{d}^{(3)}_a}{\mathbf{b}^{(3)}_a} = 2;$$

$$\nu(\alpha_3) = (1-\beta)\frac{\mathbf{d}^{(3)}_{\alpha_3}}{\mathbf{b}^{(3)}_{\alpha_3}} = 20.372.$$

3. **Step $k = 4$:** Although α_4 is not known, we know that $C(\alpha_4) = \{1, 2, 3\}$, $S(\alpha_4) = \{4\}$, and $\lambda_3 = \beta/(1 - \beta \mathbf{Q}^{(3)}_{2,2}) = 0.882$. As in the previous steps, using (5) and (6) we get

$$\mathbf{Q}^{(4)} = \begin{array}{c} \\ 4 \end{array} \begin{array}{c} \begin{array}{c} 4 \end{array} \\ \begin{bmatrix} 0.569 \end{bmatrix} \end{array},$$

$$\mathbf{d}^{(4)} = \begin{array}{c} \\ 4 \end{array} \begin{bmatrix} 30.37 \end{bmatrix}, \quad \mathbf{b}^{(4)} = \begin{array}{c} \\ 4 \end{array} \begin{bmatrix} 0.574 \end{bmatrix}.$$

Since $S(\alpha_4) = \{4\}$ has only one element, we have

$$\alpha_4 = 4, \text{ and}$$

$$\nu(\alpha_4) = (1-\beta)\frac{\mathbf{d}^{(4)}_{\alpha_4}}{\mathbf{b}^{(4)}_{\alpha_4}} = 13.236.$$

24.3.3 Triangularization Algorithm (Denardo, Park, and Rothblum)

Denardo, Park, and Rothblum [12], independently of Sonin, presented an algorithm that is similar to the state-elimination algorithm. We refer to this algorithm as the *triangularization algorithm*. Although the main motivation for this algorithm was to compute the performance of any index strategy, it can be used to compute the Gittins index as well.

A slightly modified version of the triangularization algorithm is presented below. See [12] for the original version.

Initialization: The initialization step is identical to the previous two algorithms: Identify the state α_1 with the highest Gittins index, which is given by

$$\alpha_1 = \arg \max_{a \in \mathcal{X}} \mathbf{r}_a$$

where ties are broken arbitrarily. The corresponding Gittins index is

$$\nu(\alpha_1) = \mathbf{r}_{\alpha_1}.$$

The recursion step uses a $m \times (m+2)$ tableau

$$\mathbf{M}^{(k)} = [\, \mathbf{Q}^{(k)} \,|\, \mathbf{d}^{(k)} \,|\, \mathbf{b}^{(k)} \,]$$

where $\mathbf{Q}^{(k)}$ is a $m \times m$ square matrix and $\mathbf{d}^{(k)}$ and $\mathbf{b}^{(k)}$ are $m \times 1$ vectors. Initialize these as $\mathbf{Q}^{(1)} = \mathbf{I} - \beta \mathbf{P}$, $\mathbf{d}^{(1)} = \mathbf{r}$, and $\mathbf{b}^{(1)} = (1-\beta)\mathbf{1}$.

Recursion step: After the states α_1, $\ldots$, α_{k-1} have been identified, the next step is to identify the state α_k with the kth largest Gittins index. As before, even though α_k is not known, we know that $C(\alpha_k) = \{\alpha_1, \ldots, \alpha_{k-1}\}$ and $S(\alpha_k) = \mathcal{X} \setminus \{\alpha_1, \ldots, \alpha_{k-1}\}$. Suppose the tableau in step $k-1$ is

$$\mathbf{M}^{(k-1)} = [\,\mathbf{Q}^{(k-1)} \,|\, \mathbf{d}^{(k-1)} \,|\, \mathbf{b}^{(k-1)}\,].$$

Update this tableau using the following elementary row operations

1. Let $\lambda_{k-1} = 1/\mathbf{Q}^{(k-1)}_{\alpha_{k-1},\alpha_{k-1}}$. Scale row α_{k-1} of tableau $\mathbf{M}^{(k-1)}$ by λ_{k-1} [i.e., rescale row α_{k-1} such that the $(\alpha_{k-1}, \alpha_{k-1})$ entry of $\mathbf{M}^{(k-1)}$ is 1].

2. For each state $a \in S(\alpha_k)$, subtract row α_{k-1} times the constant $\mathbf{Q}^{(k-1)}_{a,\alpha_{k-1}}$ from row a [these operations set $\mathbf{Q}^{(k)}_{a,b}$ to 0 for $b \in C(\alpha_k)$]. The updated tableau is $\mathbf{M}^{(k)}$.

After updating the tableau, choose

$$\alpha_k = \arg \max_{a \in S(\alpha_k)} \frac{\mathbf{d}^{(k)}_a}{\mathbf{b}^{(k)}_a}$$

where ties are broken arbitrarily. The corresponding Gittins index is

$$\nu(\alpha_k) = (1-\beta)\frac{\mathbf{d}^{(k)}_{\alpha_k}}{\mathbf{b}^{(k)}_{\alpha_k}}.$$

Computational complexity: The performance bottleneck of this algorithm is the elementary row operations performed at each step to update the tableau. In step k, this algorithm performs $(m-k+2)$ row operations and each row operation requires $2(m-k+1)$ arithmetic operations [This is because columns corresponding to $C(\alpha_{k-1})$ need not be updated because $Q^{(k-1)}_{a,\alpha_{k-1}}$ is 0]. Thus, overall the algorithm requires

$$\sum_{k=1}^{m} [2(m-k+1)(m-k+2) + O(k^2)] = \tfrac{2}{3}m^3 + O(m^2)$$

arithmetic operations.

Example 24.3.3 Reconsider the bandit process of Example 24.3.1. Using the triangularization algorithm, the Gittins index for this bandit process is calculated as follows:

Initialization: As in the other two algorithms, the state with the highest Gittins index is

$$\alpha_1 = \arg \max_{a \in \mathcal{X}} \mathbf{r}_a = 3;$$
$$\nu(\alpha_1) = 30.$$

Initialize the tableau $M^{(1)}$ as

$$\mathbf{M}^{(1)} = \left[\begin{array}{cccc|c|c} 0.925 & 0 & -0.6 & -0.075 & 16 & 0.25 \\ -0.375 & 1 & -0.075 & -0.3 & 19 & 0.25 \\ -0.15 & -0.45 & 1 & -0.15 & 30 & 0.25 \\ 0 & -0.6 & 0 & 0.85 & 4 & 0.25 \end{array}\right].$$

Recursion step:

1. **Step $k = 2$:** Since $\alpha_1 = 3$, set $\lambda_1 = 1/\mathbf{Q}^{(1)}_{3,3} = 1$. Using the elementary row operations, the tableau $\mathbf{M}^{(2)}$ is updated to

$$\left[\begin{array}{cccc|c|c} 0.835 & -0.27 & 0 & -0.165 & 34 & 0.4 \\ -0.386 & 0.966 & 0 & -0.311 & 21.25 & 0.269 \\ -0.15 & -0.45 & 1 & -0.15 & 30 & 0.25 \\ 0 & -0.6 & 0 & 0.85 & 4 & 0.25 \end{array}\right].$$

   Hence,

$$\alpha_2 = \arg \max_{a \in \{1,2,4\}} \frac{\mathbf{d}^{(2)}_a}{\mathbf{b}^{(2)}_a} = 1;$$
$$\nu(\alpha_2) = (1-\beta)\frac{\mathbf{d}^{(2)}_{\alpha_2}}{\mathbf{b}^{(2)}_{\alpha_2}} = 21.25.$$

2. **Step $k = 3$:** Since $\alpha_2 = 1$, set $\lambda_2 = 1/\mathbf{Q}^{(2)}_{1,1} = 1.198$. Using the elementary row operations, the tableau $\mathbf{M}^{(3)}$ is updated to

$$\left[\begin{array}{cccc|c|c} 1 & -0.323 & 0 & -0.198 & 40.719 & 0.479 \\ 0 & 0.841 & 0 & -0.388 & 36.978 & 0.454 \\ -0.15 & -0.45 & 1 & -0.15 & 30 & 0.25 \\ 0 & -0.6 & 0 & 0.85 & 4 & 0.25 \end{array}\right].$$

Hence,

$$\alpha_2 = \arg\max_{a\in\{2,4\}} \frac{\mathbf{d}_a^{(2)}}{\mathbf{b}_a^{(2)}} = 2;$$

$$\nu(\alpha_3) = (1-\beta)\frac{\mathbf{d}_{\alpha_3}^{(3)}}{\mathbf{b}_{\alpha_3}^{(3)}} = 20.362.$$

3. **Step $k = 4$:** Since $\alpha_3 = 2$, set $\lambda_3 = 1/\mathbf{Q}_{2,2}^{(3)} = 1.189$. Using the elementary row operations, the tableau $\mathbf{M}^{(4)}$ is updated to

$$\left[\begin{array}{cccc|c|c} 1 & -0.323 & 0 & -0.198 & 40.72 & 0.479 \\ 0 & 1 & 0 & -0.461 & 43.95 & 0.539 \\ -0.15 & -0.45 & 1 & -0.15 & 30 & 0.25 \\ 0 & 0 & 0 & 0.574 & 30.37 & 0.574 \end{array}\right].$$

Since $S(\alpha_4) = \{4\}$ has only one element, we have

$$\alpha_4 = 4 \text{ and}$$

$$\nu(\alpha_4) = (1-\beta)\frac{\mathbf{d}_{\alpha_4}^{(4)}}{\mathbf{b}_{\alpha_4}^{(4)}} = 13.236.$$

24.3.4 Fast-Pivoting Algorithm (Niño-Mora)

Niño-Mora [24] presented an algorithm to compute the Gittins index that refines a previous algorithm proposed by Kallenberg [16]. We refer to this algorithm as the *fast-pivoting algorithm.* As with the other algorithms, the main idea behind the fast-pivoting algorithm is to iteratively solve (2) in the decreasing order of $\succeq$. The computations are based on a relation of (3) with *marginal productivity rate* [24].

A modified version of the algorithm is presented below. See [24] for the original version.

Initialization: The initialization step is identical to the previous three algorithms: identify the state α_1 with the highest Gittins index, which is given by

$$\alpha_1 = \arg\max_{a\in\mathcal{X}} \mathbf{r}_a$$

where ties are broken arbitrarily. The corresponding Gittins index is

$$\nu(\alpha_1) = \mathbf{r}_{\alpha_1}.$$

The recursion step uses a $\tilde{m}\times m$ matrix $\mathbf{Q}^{(k)}$ and $\tilde{m}\times 1$ vectors $\mathbf{b}^{(k)}$ and $\mathbf{d}^{(k)}$, where $\tilde{m} = |S(\alpha_k)| = m - k + 1$. Initialize these as $\mathbf{Q}^{(1)} = \mathbf{0}_{m\times m}$, $\mathbf{b}^{(1)} = \mathbf{1}$, $\mathbf{d}^{(1)} = \mathbf{r}$.

Recursion step: After the states $\alpha_1,\ldots,\alpha_{k-1}$ have been identified, the next step is to identify the state α_k with the kth largest Gittins index. Even though α_k is not known, we know that $C(\alpha_k) = \{\alpha_1,\ldots,\alpha_{k-1}\}$ and $S(\alpha_k) = \mathcal{X}\setminus\{\alpha_1,\ldots,\alpha_{k-1}\}$. Let the model in the step $k-1$ be $\mathbf{Q}^{(k-1)}$, $\mathbf{b}^{(k-1)}$ and $\mathbf{d}^{(k-1)}$ and let $\tilde{m} = |S(\alpha_k)|$.

Define the $(\tilde{m}+1)\times 1$ vector by $\forall a \in S(\alpha_{k-1})$,

$$\mathbf{h}_a^{(k-1)} = \mathbf{P}_{a,\alpha_{k-1}} - \sum_{b\in C(\alpha_k)} \mathbf{Q}_{a,b}^{(k-1)}\mathbf{P}_{b,\alpha_{k-1}}. \tag{7}$$

Let

$$\lambda_{k-1} = \frac{\beta}{1-\beta\mathbf{h}_{\alpha_{k-1}}^{(k-1)}}.$$

Define the $\tilde{m}\times m$ matrix by $\forall a \in S(\alpha_k)$ and $b \in \mathcal{X}\setminus\{\alpha_{k-1}\}$,

$$\mathbf{Q}_{a,b}^{(k)} = \mathbf{Q}_{a,b}^{(k-1)} + \lambda_{k-1}\mathbf{h}_a^{(k-1)}\mathbf{Q}_{\alpha_{k-1},b}^{(k-1)},$$
$$\mathbf{Q}_{a,\alpha_{k-1}}^{(k)} = -\lambda_{k-1}\mathbf{h}_a^{(k-1)}. \tag{8}$$

and the $\tilde{m}\times 1$ vectors by $\forall a \in S(\alpha_k)$

$$\mathbf{b}_a^{(k)} = \mathbf{b}_a^{(k-1)} + \lambda_{k-1}\mathbf{h}_a^{(k-1)}\mathbf{b}_{\alpha_{k-1}}^{(k-1)}, \tag{9}$$

$$\mathbf{d}_a^{(k)} = \mathbf{d}_{\alpha_{k-1}}^{(k-1)} - \frac{\mathbf{b}_a^{(k-1)}}{\mathbf{b}_a^{(k)}}\left[\mathbf{d}_{\alpha_{k-1}}^{(k-1)} - \mathbf{d}_a^{(k-1)}\right]. \tag{10}$$

After the model has been updated, choose

$$\alpha_k = \arg\max_{a\in S(\alpha_k)} \mathbf{d}_a^{(k)}$$

and the corresponding Gittins index is given by

$$\nu(\alpha_k) = \mathbf{d}_{\alpha_k}^{(k)}.$$

Computational complexity: The performance bottleneck of this algorithm is the calculation of $\mathbf{h}^{(k-1)}$ and the update of $\mathbf{Q}^{(k)}$, each of which requires $2(m-k+2)k$ additions and multiplications [for the update of $\mathbf{Q}^{(k)}$, we know that for $b \in S(\alpha_k)$, $\mathbf{Q}^{(k)}$ is 0, hence we do not need to calculate those using (8)]. Thus, overall the algorithm requires

$$\sum_{k=1}^{m} \Big[k\big(4(m-k)+1\big)+O(k)\Big] = \tfrac{2}{3}m^3+O(m^2)$$

arithmetic operations.

Example 24.3.4 Reconsider the bandit process of Example 24.3.1. Using the fast-pivoting algorithm, the Gittins index of this bandit process is computed as follows:

Initialization: As in the other two algorithms, the state with the highest Gittins index is

$$\alpha_1 = \arg\max_{a\in\mathcal{X}} \mathbf{r}_a = 3;$$
$$\nu(\alpha_1) = 30.$$

Initialize:

$$\mathbf{Q}^{(1)} = \begin{bmatrix} 0&0&0&0\\ 0&0&0&0\\ 0&0&0&0\\ 0&0&0&0 \end{bmatrix},$$

$$\mathbf{b}^{(1)} = \begin{bmatrix} 1\\1\\1\\1 \end{bmatrix}, \quad \mathbf{d}^{(1)} = \begin{bmatrix} 16\\19\\30\\4 \end{bmatrix}.$$

Recursion step:

1. **Step $k = 2$:** Since $\alpha_1 = 3$, using (7) we get

$$\mathbf{h}^{(1)} = \begin{bmatrix} 0.8\\0.1\\0\\0 \end{bmatrix}.$$

and $\lambda_1 = \beta/(1-\beta\mathbf{h}_3^{(1)}) = 0.75$. Using (8)–(10) we get

$$\mathbf{Q}^{(2)} = \begin{matrix}1\\2\\4\end{matrix}\begin{bmatrix} 0&0&-0.6&0\\ 0&0&-0.075&0\\ 0&0&0&0 \end{bmatrix},$$

$$\mathbf{b}^{(2)} = \begin{matrix}1\\2\\4\end{matrix}\begin{bmatrix} 1.6\\1.075\\1 \end{bmatrix}, \quad \mathbf{d}^{(2)} = \begin{matrix}1\\2\\4\end{matrix}\begin{bmatrix} 21.25\\19.767\\4 \end{bmatrix}.$$

Hence,

$$\alpha_2 = \arg\max_{a\in\{1,2,4\}} \mathbf{d}_a^{(2)} = 1;$$
$$\nu(\alpha_2) = \mathbf{d}_{\alpha_2}^{(2)} = 21.25.$$

2. **Step $k = 3$:** Since $\alpha_2 = 1$, using (7) we get

$$\mathbf{h}^{(2)} = \begin{matrix}1\\2\\4\end{matrix}\begin{bmatrix} 0.22\\0.515\\0 \end{bmatrix}$$

and $\lambda_2 = \beta/(1-\beta\mathbf{h}_1^{(2)}) = 0.898$. Using (8)–(10) we get

$$\mathbf{Q}^{(3)} = \begin{matrix}2\\4\end{matrix}\begin{bmatrix} -0.463&0&-0.353&0\\ 0&0&0&0 \end{bmatrix},$$

$$\mathbf{b}^{(3)} = \begin{matrix}2\\4\end{matrix}\begin{bmatrix} 1.815\\1 \end{bmatrix}, \quad \mathbf{d}^{(3)} = \begin{matrix}2\\4\end{matrix}\begin{bmatrix} 20.372\\4 \end{bmatrix}.$$

Hence,

$$\alpha_3 = \arg\max_{a\in S(\alpha_3)} \mathbf{d}_a^{(3)} = 2;$$
$$\nu(\alpha_3) = \mathbf{d}_{\alpha_3}^{(3)} = 20.372.$$

3. **Step $k = 4$:** Since $\alpha_3 = 2$, using (7) we get

$$\mathbf{h}^{(3)} = \begin{matrix}2\\4\end{matrix}\begin{bmatrix} 0.212\\0.8 \end{bmatrix}$$

and $\lambda_3 = \beta/(1-\beta\mathbf{h}_2^{(3)}) = 0.891$. Using (8)–(10), we get

$$\mathbf{Q}^{(4)} = \begin{matrix}4\end{matrix}\begin{bmatrix} 0&-0.713&0&0 \end{bmatrix},$$

$$\mathbf{b}^{(4)} = \begin{matrix}4\end{matrix}\begin{bmatrix} 2.294 \end{bmatrix}, \quad \mathbf{d}^{(4)} = \begin{matrix}4\end{matrix}\begin{bmatrix} 13.236 \end{bmatrix}.$$

Since $S(\alpha_4) = \{4\}$ has only one element, we have

$$\alpha_4 = 4, \text{ and } \nu(\alpha_4) = \mathbf{d}_{\alpha_4}^{(4)} = 13.236.$$

24.3.5 An Efficient Method to Compute Gittins Index of Multiple Processes

Consider a MAB problem in which all processes have a finite state space $\mathcal{X}^i = \{1, \ldots, m^i\}$. Let $\alpha^i_{m^i}$ be the state with the smallest Gittins index in process i, $i = 1, \ldots, n$. Let

$$i^* = \arg\max_i \nu^i(\alpha^i_{m^i})$$

be the process whose smallest Gittins index is the largest among all alternatives. Then, a Gittins index strategy will eventually settle on process i^* and play it forever.

Based on the above insight, Denardo, Feinberg, and Rothblum [11] proposed the following method to efficiently compute the Gittins index of multiple processes in parallel.

Initialization: Identify the state α^i_1 with the highest Gittins index for all processes and calculate its Gittins index $\nu^i(\alpha^i_1)$ (using any of the methods described above).

Recursion step: Suppose we have computed the Gittins index of the k^ith highest state of process i, $i = 1, \ldots, n$. Let

$$j = \arg\max_i \nu^i(\alpha^i_{k^i}). \tag{11}$$

If $k^j < m^j$, then identify the state with the next highest Gittins index in process j (using any one of the methods described above), compute the Gittins index of that state, set $k^j = k^j + 1$, and repeat. Otherwise, if $k^j = m^j$, then $j = i^*$ and we don't need to compute the Gittins index of the remaining states as the Gittins index strategy will never play a process in those states (and instead prefers to play process i^*).

Example 24.3.5 To illustrate the above method, consider a MAB problem with three bandit processes, each with state space $\mathcal{X} = \{1, 2, 3, 4\}$, transition matrix

$$\mathbf{P} = \begin{bmatrix} 0.1 & 0 & 0.8 & 0.1 \\ 0.5 & 0 & 0.1 & 0.4 \\ 0.2 & 0.6 & 0 & 0.2 \\ 0 & 0.8 & 0 & 0.2 \end{bmatrix}$$

and reward vectors

$$\mathbf{r}^1 = \begin{bmatrix} 16 \\ 19 \\ 30 \\ 4 \end{bmatrix}, \quad \mathbf{r}^2 = \begin{bmatrix} 40 \\ 10 \\ 8 \\ 15 \end{bmatrix}, \quad \mathbf{r}^3 = \begin{bmatrix} 20 \\ 15 \\ 22 \\ 25 \end{bmatrix}.$$

Let the discount factor be $\beta = 0.75$. Note that the first bandit process is identical to that which we have considered in the previous examples. Using the above method, we proceed as follows:

Initialization: Compute the state with the highest Gittins index for each of the processes to get:

$$\begin{aligned} \alpha^1_1 &= 3, & \nu^1(\alpha^1_1) &= 30; \\ \alpha^2_1 &= 1, & \nu^2(\alpha^2_1) &= 40; \\ \alpha^3_1 &= 4, & \nu^3(\alpha^3_1) &= 25. \end{aligned}$$

Recursion step:

1. **Step $(k^1, k^2, k^3) = (1, 1, 1)$.** Using (11) gives that $j = 2$. So, find the next highest Gittins index of bandit process 2, which gives

$$\alpha^2_2 = 2, \quad \nu^2(\alpha^2_2) = 18.654.$$

2. **Step $(k^1, k^2, k^3) = (1, 2, 1)$.** Using (11) gives that $j = 1$. So, find the next highest Gittins index of bandit process 1, which gives

$$\alpha^1_2 = 1, \quad \nu^1(\alpha^1_2) = 21.25.$$

3. **Step $(k^1, k^2, k^3) = (2, 2, 1)$.** Using (11) gives that $j = 3$. So, find the next highest Gittins index of bandit process 3, which gives

$$\alpha^3_2 = 3, \quad \nu^3(\alpha^3_2) = 22.45.$$

4. **Step $(k^1, k^2, k^3) = (2, 2, 2)$.** Using (11) gives that $j = 3$. So, find the next highest Gittins index of bandit process 3, which gives

$$\alpha_3^3 = 1, \quad \nu^3(\alpha_3^3) = 21.209.$$

5. **Step $(k^1, k^2, k^3) = (2, 2, 3)$.** Using (11) gives that $j = 1$. So, find the next highest Gittins index of bandit process 1, which gives

$$\alpha_3^1 = 2, \quad \nu^1(\alpha_3^1) = 20.372.$$

6. **Step $(k^1, k^2, k^3) = (3, 2, 3)$.** Using (11) gives that $j = 3$. So, find the next highest Gittins index of bandit process 3, which gives

$$\alpha_4^3 = 2, \quad \nu^3(\alpha_4^3) = 19.113.$$

7. **Step $(k^1, k^2, k^3) = (3, 2, 4)$.** Since $k^3 = 4$, stop the algorithm. Note that, although we have not calculated the Gittins index of state 4 of process 1 and of states $\{3, 4\}$ of process 2, we know that Gittins index of these states is less than 19.113, the smallest Gittins index of process 3. So, the Gittins index strategy will never play process 1 in state 4 or process 2 in states $\{3, 4\}$.

24.4 On-Line Algorithms for Computing Gittins Index

As in the case of off-line algorithms, we consider a single bandit process and drop the label i of the bandit processes. For ease of exposition, assume that X_t takes value in a finite space $\mathcal{X} = \{1, \ldots, m\}$. These algorithms are based on dynamic programming and linear programming and extend to countable state spaces based on standard extensions of dynamic and linear programming to countable state spaces. As before, denote the $m \times m$ transition probability matrix by $\mathbf{P} = [\mathbf{P}_{a,b}]$ and represent the reward function using a $m \times 1$ vector $\mathbf{r}$. Furthermore, let $\mathbf{1}$ denote the $m \times 1$ vector of ones and $\mathbf{0}_{m\times 1}$ denote the $m \times 1$ vector of zeros.

On-line algorithms for Gittins index are based on bandit processes with the *retirement option* introduced by Whittle [34]. In such a bandit process, a player has the option to either play a regular bandit process and receive a reward according to the reward function r or choose to retire and receive a one-time retirement reward of M. Thus, the player is faced with an optimal stopping problem of maximizing

$$\mathbb{E}\left[\sum_{t=0}^{\tau-1} \beta^t r(X_t) + \beta^\tau M \,\middle|\, X_0 = x_0\right]$$

over all $\{\sigma(X_1, \ldots, X_t)\}_{t=0}^{\infty}$ measurable stopping times τ.

For a fixed value of retirement reward, an optimal solution is given by the solution of the following dynamic program. Let the $m \times 1$ vector $\mathbf{v}^{(M)}$ be the unique fixed point of

$$\mathbf{v}^{(M)} = \max\left\{\mathbf{r} + \beta \mathbf{P}\mathbf{v}^{(M)}, M\mathbf{1}\right\} \quad (12)$$

where the max is element-wise maximum of two vectors. Standard results in Markov decision theory [26] show that the above equation has a unique point.

Fix a state $\alpha \in \mathcal{X}$ and let M_α denote the smallest retirement reward such that a player is indifferent between playing the bandit process starting at α or retiring, that is,

$$M_\alpha = \min\left\{M \,\middle|\, \mathbf{v}_\alpha^{(M)} = M = \mathbf{r}_\alpha + \beta \sum_x \mathbf{P}_{\alpha x} \mathbf{v}_x^{(M)}\right\}. \quad (13)$$

Whittle [34] showed that

$$\nu(\alpha) = (1 - \beta) M_\alpha. \quad (14)$$

The on-line algorithms use this interpretation to compute the Gittins index of a particular state α.

24.4.1 Restart Formulation (Katehakis and Veinott)

Katehakis and Veinott [18] related Whittle's retirement option to a restart problem and then solved it using dynamic programming. The intuition behind their approach is the following. Fix a state $\alpha \in \mathcal{X}$. When the retirement reward is M_α, the player is indifferent between playing the bandit process starting in state α or retiring. Therefore the option of retirement reward of M_α is equivalent to the option of restarting in state α.

On the basis of the above equivalence, consider the following restart problem. At each time, the player has an option to play the bandit process or instantaneously restart in state α. To describe a solution of this restart problem, define $\mathbf{r}^{(0)} = \mathbf{r}_\alpha \mathbf{1}$ and $\mathbf{Q}^{(0)}$ as

$$\mathbf{Q}^{(0)}_{x,y} = \mathbf{P}_{\alpha,y}, \quad \forall x, y \in \mathcal{X}$$

to be the instantaneous reward and transition probability matrix for the restart option and $\mathbf{r}^{(1)} = \mathbf{r}$ and $\mathbf{Q}^{(1)} = \mathbf{P}$ to be the instantaneous reward and transition matrix for the continuation option. Then a solution to the restart problem is given by the following dynamic program. Let the $m \times 1$ vector $\mathbf{v}$ be the unique fixed point of

$$\mathbf{v} = \max\left\{\mathbf{r}^{(0)} + \beta\mathbf{Q}^{(0)}\mathbf{v}, \mathbf{r}^{(1)} + \beta\mathbf{Q}^{(1)}\mathbf{v}\right\} \tag{15}$$

where max is the element-wise maximum of two vectors. Then the Gittins index of state α is

$$\nu(\alpha) = (1-\beta)\mathbf{v}_\alpha$$

and the corresponding stopping set is

$$S(\alpha) = \{x \in \mathcal{X} \mid \mathbf{r}^{(0)} + \beta\mathbf{Q}^{(0)}\mathbf{v} \geq \mathbf{r}^{(1)} + \beta\mathbf{Q}^{(1)}\mathbf{v}\}.$$

The fixed point equation (15) depends on the state α [because $\mathbf{r}^{(0)}$ and $\mathbf{Q}^{(0)}$ depend on α]. This equation may be solved using standard tools for solving dynamic programs such as value iteration, policy iteration, or dynamic programming. Beale (in the discussion of [15]) proposes an online algorithm that can be viewed as a special form of policy iteration to solve (15). See [26] for details on numerically solving fixed point equations for the form (15).

Example 24.4.1 Reconsider the bandit process of Example 24.3.1. To compute the Gittins index of state 2 (which is the state with the third largest index), proceed as follows:

Define

$$\mathbf{r}^{(0)} = \begin{bmatrix} 19 \\ 19 \\ 19 \\ 19 \end{bmatrix}, \quad \mathbf{Q}^{(0)} = \begin{bmatrix} 0.5 & 0 & 0.1 & 0.4 \\ 0.5 & 0 & 0.1 & 0.4 \\ 0.5 & 0 & 0.1 & 0.4 \\ 0.5 & 0 & 0.1 & 0.4 \end{bmatrix}$$

and

$$\mathbf{r}^{(1)} = \begin{bmatrix} 16 \\ 19 \\ 30 \\ 4 \end{bmatrix}, \quad \mathbf{Q}^{(1)} = \begin{bmatrix} 0.1 & 0 & 0.8 & 0.1 \\ 0.5 & 0 & 0.1 & 0.4 \\ 0.2 & 0.6 & 0 & 0.2 \\ 0 & 0.8 & 0 & 0.2 \end{bmatrix}.$$

The dynamic program for the restart formulation is given by (15).

As mentioned above, there are various algorithms such as value iteration, policy iteration, linear programming, etc. to solve the above dynamic program. Using any one of these algorithms gives that

$$\mathbf{v} = \begin{bmatrix} 83.170 \\ 81.488 \\ 91.368 \\ 81.488 \end{bmatrix}$$

is the unique fixed point of (15). Hence

$$\nu(2) = (1-\beta)\mathbf{v}_2 = 20.372$$

and

$$S(2) = \{2, 4\}.$$

24.4.2 Linear Programming Formulation (Chen and Katehakis)

Chen and Katehakis [10] showed that M_α given by (13) may be computed using a

linear program. A modified version of this linear program is presented below.

Define $\mathbf{h} = \mathbf{1} - \mathbf{e}_\alpha$, where $\mathbf{e}_\alpha$ is the $m \times 1$ unit vector with 1 at the αth position. Let $\text{diag}(\mathbf{h})$ denote the $m \times m$ diagonal matrix with $\mathbf{h}$ as its diagonal. Let the $(m+1) \times 1$ vector $\mathbf{z} = \left[\mathbf{y}^\intercal \mid z\right]^\intercal$, where $\mathbf{y}$ is a $m \times 1$ vector and z is a scalar, be the solutions of the following linear program.

$$\text{minimize } \left[(1-\beta)(\mathbf{1})^\intercal \mid m\right]\mathbf{z}$$

subject to

$$\begin{gathered}\left[\text{diag}(\mathbf{h}) - \beta\mathbf{P} \mid \mathbf{1}\right]\mathbf{z} \geq \mathbf{r}, \\ \mathbf{y} \geq \mathbf{0}_{m\times 1}, \\ z \text{ unrestricted.}\end{gathered} \tag{16}$$

Then the Gittins index of state α is

$$\nu(\alpha) = z$$

and the corresponding stopping set is

$$S(\alpha) = \left\{x \in \mathcal{X} \,\middle|\, \mathbf{y}_x = 0\right\}.$$

The linear program (16) depends on the state α (because $\mathbf{h}$ depends on α). This linear program may be solved using standard tools. See [6] for details.

Example 24.4.2 Reconsider the bandit process of Example 24.3.1. To compute the Gittins index of state 2 (which is the state with the third largest index), proceed as follows:

Define $\mathbf{h} = \begin{bmatrix}1 & 1 & 0 & 1\end{bmatrix}^\intercal$. Let $\mathbf{z} = \left[\mathbf{y}^\intercal \mid z\right]^\intercal$ be the solution to the following linear program:

$$\text{minimize } \begin{bmatrix}0.25 & 0.25 & 0.25 & 0.25 & 4\end{bmatrix}\mathbf{z} \tag{17}$$

subject to

$$\begin{gathered}\begin{bmatrix} 0.925 & 0 & -0.6 & -0.075 & 1 \\ -0.375 & 1 & -0.075 & -0.3 & 1 \\ -0.15 & -0.45 & 0 & -0.15 & 1 \\ 0 & -0.6 & 0 & 0.85 & 1 \end{bmatrix}\mathbf{z} \geq \begin{bmatrix}16 \\ 19 \\ 30 \\ 4\end{bmatrix}, \\ \mathbf{y} \geq \mathbf{0}_{4\times 1}, \\ z \text{ unrestricted.}\end{gathered}$$

As mentioned above, there are various algorithms to solve linear programs. Using any of these algorithms gives that

$$\mathbf{z} = \left[\frac{\mathbf{y}}{z}\right] = \begin{bmatrix}1.68 \\ 0 \\ 9.88 \\ 0 \\ \hline 20.372\end{bmatrix}$$

is the unique optimal solution of (17). Hence,

$$\nu(2) = z = 20.372$$

and

$$S(2) = \left\{x \in \mathcal{X} \,\middle|\, \mathbf{y}_x = 0\right\} = \{2, 4\}.$$

24.5 Computing Gittins Index for the Bernoulli Sampling Process

In clinical trials, it is common for treatments to be modeled as a Bernoulli process $\{Y_t^i\}_{t=0}^\infty$ with unknown success probability s^i, $i = 1, \ldots, n$. Such a MAB setup is called a Bernoulli *sampling* process because the player must sample from one of the n available Bernoulli processes at each time. Assume that s^i has a $\text{Beta}(p_0^i, q_0^i)$ distribution[1] where p_0^i and q_0^i are nonnegative integers. Suppose at time t, sampling process i has resulted in k successes and ℓ failures. Then, the posterior distribution of s^i is $\text{Beta}(p_t^i, q_t^i)$ where

$$p_t^i = p_0^i + k, \quad q_t^i = q_0^i + \ell.$$

Therefore, $X_t^i \equiv (p_t^i, q_t^i)$ may be used as an *information state* (or a *sufficient statistic*) for the bandit process i. This state evolves in a Markovian manner. In particular if $X_t^i = (p_t^i, q_t^i)$ then

$$X_{t+1}^i \equiv (p_{t+1}^i, q_{t+1}^i)$$

[1] If a random variable s has $\text{Beta}(p, q)$ distribution, then its PDF (probability density function) $f(s)$ is proportional to $s^p(1-s)^q$. See [36] for methods to elicit Beta priors based on expert opinions in clinical trials.

$$= \begin{cases} (p_t^i + 1, q_t^i), & \text{w.p. } p_t^i/(p_t^i + q_t^i); \\ (p_t^i, q_t^i + 1), & \text{w.p. } q_t^i/(p_t^i + q_t^i) \end{cases}$$

where w.p. is an abbreviation for "with probability." The average reward on playing process i is the mean value of Beta(p_t^i, q_t^i), that is,

$$r^i(p_t^i + q_t^i) = \frac{p_t^i}{p_t^i + q_t^i}.$$

In this section, we describe the various algorithms to compute the Gittins index of such a Bernoulli sampling process. As before, since the computation of the index depends only on that process, we drop the label i and denote the sampling process by $\{(p_t, q_t)\}_{t=0}^{\infty}$ and its Gittins index by $\nu(p, q)$.

The main difficulty in computing the Gittins index $\nu(p, q)$ is that the state space $\mathcal{X} = \mathbb{Z}_+^2$ is countably infinite. Hence, an exact computation of the index is not possible. In this section we present algorithms that compute the Gittins index with arbitrary precision by restricting to a truncated state space $\mathcal{X}_L = \{(p, q) \mid p + q \leq L\}$ for sufficiently large value of L. The results based on these calculations are tabulated in [14]. Different approximations to $\nu(p, q)$ have also been proposed in the literature, and we describe a few of these in this section as well.

24.5.1 Dynamic Programming Formulation (Gittins)

Gittins [14] used the dynamic programming formulation of Whittle's retirement option [34], which was presented in Section 24.4, to compute the Gittins index $\nu(p, q)$. In the case of a Bernoulli sampling process, the dynamic program (12) simplifies to

$$\mathbf{v}_{p,q}^{(M)} = \max\{\mathbf{w}_{p,q}^{(M)}, M\} \qquad (18)$$

where

$$\mathbf{w}_{p,q}^{(M)} = \tfrac{p}{p+q}\left[1 + \beta \mathbf{v}^{(M)}(p+1, q)\right] + \tfrac{q}{p+1}\beta \mathbf{v}^{(M)}(p, q+1).$$

Let $M_{p,q}$ be defined as in (13). Then, the Gittins index is given by

$$\nu(p, q) = (1 - \beta) M_{p,q}.$$

Gittins [14] presents an approximate solution of (18) by restricting to $\mathcal{X}_L = \{(p, q) \mid p + q \leq L\}$, discretizing M, and using value-iteration. On the basis of these calculations, the values of $\nu(p, q)$ accurate up to four decimal places are tabulated in [14, Tables 8.4–8.8] for different values of $\beta \in [0.5, 0.99]$. MATLAB code for the above calculations is also available in [14].

Gittins [14] also observed that for large $p + q$ the Gittins index $\nu(p, q)$ is well approximated as follows: Let $n = p + q$ and $\mu = p/n$, then

$$\frac{1}{\nu(p, q) - \mu} \approx \sqrt{\frac{1 - \beta}{\mu(1 - \mu)}} \left[A + Bn + \frac{C}{n}\right]$$

where A, B, and C depend on β and μ. The fitted values of A, B, and C as a function of β and μ are tabulated in [14, Tables 8.9–8.11] for $\mu \in [0.025, 0.975]$ and $\beta \in [0.5, 0.99]$.

24.5.2 Restart Formulation (Katehakis and Derman)

Katehakis and Derman [17] used the restart formulation of Katehakis and Veinott [18], which was presented in Section 24.4.1, to compute the Gittins index $\nu(p, q)$. In the case of the Bernoulli sampling process, the dynamic program (15) for computing the Gittins index of a state (p_0, q_0) simplifies to

$$\mathbf{v}_{p,q} = \max\{\mathbf{w}_{p_0,q_0}, \mathbf{w}_{p,q}\} \qquad (19)$$

where

$$\mathbf{w}_{p,q} = \tfrac{p}{p+q}\left[1 + \beta \mathbf{v}(p+1,q)\right] + \tfrac{q}{p+1}\beta \mathbf{v}(p,q+1)$$

and $\mathbf{w}_{p_0,q_0}$ is defined similarly. Katehakis and Derman presented an approximate solution of (19) by restricting to $\mathcal{X}_L = \{(p,q) \mid p+q \leq L\}$. They also showed that for any $\varepsilon > 0$, there exists a sufficiently large $L = L(\varepsilon)$ such that L iterations of value-iteration on the truncated state space $\mathcal{X}_L$ gives a value function that is within ε of the true value function (Ben-Israel and Flåm [3] had derived similar similar bounds on value iteration for computing the Gittins index to a specific precision). Thus, this method may be used to calculate the Gittins index to an arbitrary precision.

24.5.3 Closed-Form Approximations

For large values of β, $\nu(p,q)$ of a Bernoulli process may be approximated using diffusion approximation and using the Gittins index for a Weiner process. Two such closed-form approximations are presented below.

An important result in the context of these approximation is by Katehakis and Veinott [18], who showed that if we follow an index strategy where the index is within ε of the Gittins index, then the performance of the index strategy is within ε of the optimal performance.

Whittle's approximation: Whittle [35] showed that for large $p+q$ and β, the Gittins index of a Bernoulli sampling process may be approximated as

$$\nu(p,q) \approx \mu + \frac{\mu(1-\mu)}{n\sqrt{(2c+n^{-1})\mu(1-\mu)} + \mu - \frac{1}{2}}$$

where $n = p+q$, $\mu = p/n$, and $c = \ln \beta^{-1}$.

Brezzi and Lai's approximation: Brezzi and Lai [8] showed that the Gittins index of a Bernoulli sampling process may be approximated as

$$\nu(p,q) \approx \mu + \sigma \psi\left(\frac{\sigma^2}{\rho^2 \ln \beta^{-1}}\right)$$

where μ and σ^2 are the mean and variance of Beta(p,q) distribution, that is,

$$\mu = \frac{p}{(p+q)}, \quad \sigma^2 = \frac{pq}{(p+q)^2(p+q+1)}, \tag{20}$$

ρ^2 is the variance of Bernoulli(μ) distribution, that is,

$$\rho^2 = \mu(1-\mu) = \frac{pq}{(p+q)^2},$$

and $\psi(s)$ is a nondecreasing, piecewise nonlinear function given by

$$\begin{cases} \sqrt{s/2}, & \text{if } s \leq 0.2; \\ 0.49 - 0.11 s^{-1/2}, & \text{if } 0.2 < s \leq 1; \\ 0.63 - 0.26 s^{-1/2}, & \text{if } 1 < s \leq 5; \\ 0.77 - 0.58 s^{-1/2}, & \text{if } 5 < s \leq 15; \\ \{2\ln s - \ln\ln s - \ln 16\pi\}^{1/2}, & \text{if } s > 15. \end{cases}$$

Using the notation $n = p+q$, $\mu = p/n$, and $c = \ln \beta^{-1}$ the above expression simplifies to

$$\nu(p,q) \approx \mu + \sqrt{\frac{\mu(1-\mu)}{n+1}}\psi\left(\frac{1}{(n+1)c}\right).$$

This closed-form expression provides a good approximation for $\beta \geq 0.8$ and $\min(p,q) \geq 4$.

24.5.4 Qualitative Properties of Gittins Index

Kelly [19] showed that the Gittins index is nondecreasing with the discount factor β. Brezzi and Lai [7] showed that

$$\mu \leq \nu(p,q) \leq \mu + \sigma\sqrt{\frac{\beta}{1-\beta}}$$

where μ and σ^2 are the mean and variance of Beta(p, q) distribution as given by (20). Bellman [2] showed that

$$\nu(p, q+1) < \nu(p, q) < \nu(p+1, q).$$

Thus, the optimal strategy has the *stay-on-the-winner* property defined by Robbins [27].

As $p+q \to \infty$, Kelly [19] showed that

$$\nu(p, q) \to \mu = \frac{p}{p+q} \to s$$

where s is the true success probability of the Bernoulli process. The rate of convergence slows down as $\beta \to 1$.

Kelly [19] showed that for any $k > 0$, there exists a sufficiently large β^* such that for $\beta > \beta^*$

$$\nu(p+k, q+1) < \nu(p, q).$$

Therefore, as $\beta \to 1$, the optimal strategy tends to *the least failure strategy*: Sample the process with the least number of failures and in case of ties select the process with the largest number of successes.

Brezzi and Lai [7] (and also Rothschild [28], Kumar and Varaiya [21], and Banks and Sundaram [1] in slightly different setups) have shown that the Gittins index strategy eventually chooses one process that it samples for ever, and there is a positive probability that the chosen process does not have the maximum s^i. Thus, there is *incomplete learning* when following the Gittins index strategy.

24.6 Conclusion

In this chapter, we have summarized various algorithms to compute the Gittins index. We began by considering general Markovian bandit processes and described off-line and on-line algorithms to compute the Gittins index. We then considered the Bernoulli sampling process, described algorithms that approximately compute the Gittins index, and presented closed-form approximations and qualitative properties of the Gittins index.

References

[1] J. S. Banks and R. K. Sundaram, Denumerable-armed bandits, *Econometrica: Journal of the Econometric Society*, vol. 60, no. 5, pp. 1071–1096, 1992.

[2] R. Bellman, A problem in the sequential design of experiments, *Sankhyā: The Indian Journal of Statistics*, vol. 16, no. 3/4, pp. 221–229, 1956.

[3] A. Ben-Israel and S. Flåm, A bisection/successive approximation method for computing Gittins indices, *Mathematical Methods for Operations Research*, vol. 34, no. 6, pp. 411–422, 1990.

[4] D. Bergemann and J. Valimaki, Bandit problems, in *The New Palgrave Dictionary of Economics*, S. N. Durlauf and L. E. Blume, Eds. Macmillan Press, pp. 336–340, 2008.

[5] D. A. Berry, Adaptive clinical trials: the promise and the caution, *Journal of Clinical Oncology*, vol. 29, no. 6, pp. 606–609, 2011.

[6] D. Bertsekas and J. N. Tsitsiklis, *Introduction to Linear Optimization*. Athena Scientific, 1997.

[7] M. Brezzi and T. L. Lai, Incomplete learning from endogenous data in dynamic allocation, *Econometrica*, vol. 68, no. 6, pp. 1511–1516, 2000.

[8] M. Brezzi and T. L. Lai, Optimal learning and experimentation in bandit problems, *Journal of Economic Dynamics and Control*, vol. 27, no. 1, pp. 87–108, 2002.

[9] S. Bubeck and N. Cesa-Bianchi, Regret analysis of stochastic and nonstochastic multi-armed bandit problems, *Foundations and Trends in Machine Learning*, vol. 5, no. 1, pp. 1–122, 2012.

[10] Y. R. Chen and M. N. Katehakis, Linear programming for finite state multi-armed bandit problems, *Mathematics of Operations Research*, vol. 11, no. 1, pp. 180–183, 1986.

[11] E. V. Denardo, E. A. Feinberg, and U. G. Rothblum, The multi-armed bandit, with constraints, *Annals of Operations Research*, vol. 208, pp. 37–62, 2013.

[12] E. V. Denardo, H. Park, and U. G. Rothblum, Risk-sensitive and risk-neutral multiarmed bandits, *Mathematics of Operations Research*, pp. 374–396, 2007.

[13] J. C. Gittins and D. M. Jones, A dynamic allocation index for the discounted multiarmed bandit problem, in *Progress in Statistics*, Amsterdam, Netherlands: North-Holland, vol. 9, pp. 241–266, 1974.

[14] J. Gittins, K. Glazebrook, and R. Weber, *Multi-Armed Bandit Allocation Indices*, Wiley, 2011.

[15] J. C. Gittins, Bandit processes and dynamic allocation indices, *Journal of the Royal Statistical Society, Series B (Methodological)*, vol. 41, no. 2, pp. 148–177, 1979.

[16] L. C. Kallenberg, A note on MN Katehakis' and Y.-R. Chen's computation of the Gittins index, *Mathematics of operations research*, vol. 11, no. 1, pp. 184–186, 1986.

[17] M. N. Katehakis and C. Derman, Computing optimal sequential allocation rules in clinical trials, *Lecture Notes-Monograph Series*, vol. 8, pp. 29–39, 1986.

[18] M. N. Katehakis and A. F. Veinott, The multi-armed bandit problem: decomposition and computation, *Mathematics of Operations Research*, vol. 12, no. 2, pp. 262–268, 1987.

[19] F. Kelly, Multi-armed bandits with discount factor near one: The Bernoulli case, *The Annals of Statistics*, vol. 9, no. 5, pp. 987–1001, 1981.

[20] V. Kuleshov and D. Precup, Algorithms for the multi-armed bandit problem, *Journal of Machine Learning Research*, vol. 1, pp. 1–48, 2000.

[21] P. R. Kumar and P. Varaiya, *Stochastic Systems: Estimation Identification and Adaptive Control*, Prentice Hall, 1986.

[22] T. L. Lai and H. Robbins, Asymptotically efficient adaptive allocation rules, *Advances in applied mathematics*, vol. 6, no. 1, pp. 4–22, 1985.

[23] A. Mahajan and D. Teneketzis, *Foundations and Applications of Sensor Management*, ch. Multi-armed bandits, Springer-Verlag, pp. 121–151, 2008.

[24] J. Niño-Mora, A $(2/3)n^3$ fast-pivoting algorithm for the Gittins index and optimal stopping of a Markov chain, *INFORMS Journal on Computing*, vol. 19, no. 4, pp. 596–606, 2007.

[25] J. Nino-Mora, Stochastic scheduling. *Encyclopedia of Optimization*, vol. 5, pp. 367–372, 2009.

[26] M. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, John Wiley and Sons, 1994.

[27] H. Robbins, Some aspects of the sequential design of experiments, *Bulletin of American Mathematics Society*, vol. 58, pp. 527–536, 1952.

[28] M. Rothschild, A two-armed bandit theory of market pricing, *Journal of Economic Theory*, vol. 9, no. 2, pp. 185–202, 1974.

[29] I. Sonin, The elimination algorithm for the problem of optimal stopping, *Mathematical methods of operations research*, vol. 49, no. 1, pp. 111–123, 1999.

[30] I. M. Sonin, A generalized Gittins index for a Markov chain and its recursive calculation, *Statistics & Probability Letters*, vol. 78, no. 12, pp. 1526–1533, 2008.

[31] W. R. Thompson, On the likelihood that one unknown probability exceeds another in view of the evidence of two samples, *Biometrika*, vol. 25, no. 3/4, pp. 285–294, 1933.

[32] P. Varaiya, J. Walrand, and C. Buyukkoc, Extensions of the multiarmed bandit problem: the discounted case, *IEEE Trans. Autom. Control*, vol. 30, no. 5, pp. 426–439, 1985.

[33] R. B. Washburn, Application of multi-armed bandits to sensor management, in *Foundations and Applications of Sensor Management*. Springer, pp. 153–175, 2008.

[34] P. Whittle, Multi-armed bandits and the Gittins index, *Journal of the Royal Statistical Society, Series B (Methodological)*, vol. 42, no. 2, pp. 143–149, 1980.

[35] P. Whittle, *Optimization Over Time*, ser. Wiley Series in Probability and Mathematical Statistics, vol. 2, John Wiley and Sons, 1983,.

[36] Y. Wu, W. J. Shih, and D. F. Moore, Elicitation of a beta prior for bayesian inference in clinical trials, *Biometrical Journal*, vol. 50, no. 2, pp. 212–223, 2008.