Robbins (1985) shows that, for the undiscounted case, no possible algorithm can have regret that grows more slowly than $O(\log N)$. Several different choices of g lead to a UCB policy that matches this growth; for example, we can use $g(N) = (2\log(1 + N\log^2 N))^{1/2}$.

Thompson sampling

A second method, **Thompson sampling** (Thompson, 1933), chooses an arm randomly according to the probability that the arm is in fact optimal, given the samples so far. Suppose that $P_i(\mu_i)$ is the current probability distribution for the true value of arm M_i. Then a simple way to implement Thompson sampling is to generate one sample from each P_i and then pick the best sample. This algorithm also has a regret that grows as $O(\log N)$.

17.3.4 Non-indexable variants

Bandit problems were motivated in part by the task of testing new medical treatments on seriously ill patients. For this task, the goal of maximizing the total number of successes over time clearly makes sense: each successful test means a life saved, each failure a life lost.

If we change the assumptions slightly, however, a different problem emerges. Suppose that, instead of determining the best medical treatment for each new human patient, we are instead testing different drugs on samples of bacteria with the goal of deciding which drug is best. We will then put that drug into production and forgo the others. In this scenario there is no additional cost if the bacteria dies—there is a fixed cost for each test, but we don't have to minimize test failures; rather we are just trying to make a good decision as fast as possible.

Selection problem

The task of choosing the best option under these conditions is called a **selection problem**. Selection problems are ubiquitous in industrial and personnel contexts. One often must decide which supplier to use for a process; or which candidate employees to hire. Selection problems are superficially similar to the bandit problem but have different mathematical properties. In particular, *no index function exists for selection problems.* The proof of this fact requires showing any scenario where the optimal policy switches its preferences for two arms M_1 and M_2 when a third arm M_3 is added (see Exercise 17.SELC).

Chapter 5 introduced the concept of **metalevel** decision problems such as deciding what computations to make during a game-tree search prior to making a move. A metalevel decision of this kind is also a selection problem rather than a bandit problem. Clearly, a node expansion or evaluation *costs* the same amount of time whether it produces a high or a low output value. It is perhaps surprising, then, that the Monte Carlo tree search algorithm (see page 163) has been so successful, given that it tries to solve selection problems with the UCB heuristic, which was designed for bandit problems. Generally speaking, one expects optimal bandit algorithms to explore much less than optimal selection algorithms, because the bandit algorithm assumes that a failed trial costs real money.

Bandit superprocess
BSP

An important generalization of the bandit process is the **bandit superprocess** or **BSP**, in which each arm is a full Markov decision process in its own right, rather than being a Markov reward process with only one possible action. All other properties remain the same: the arms are independent, only one (or a bounded number) can be worked on at a time, and there is a single discount factor.

Examples of BSPs include daily life, where one can attend to one task at a time, even though several tasks may need attention; project management with multiple projects; teaching with multiple pupils needing individual guidance; and so on. The ordinary term for this is

Multitasking

multitasking. It is so ubiquitous as to be barely noticeable: when formulating a real-world decision problem, decision analysts rarely ask if their client has other, unrelated problems.