JMLR: Workshop and Conference Proceedings 24:25–42, 2012 10th European Workshop on Reinforcement Learning

Planning in Reward-Rich Domains via PAC Bandits

Sergiu Goschin SGOSCHIN@CS.RUTGERS.EDU
Ari Weinstein AWEINST@CS.RUTGERS.EDU
Michael L. Littman MLITTMAN@CS.RUTGERS.EDU
Erick Chastain ERICKC@CS.RUTGERS.EDU
Rutgers University, Piscataway, NJ 08854 USA

Editor: Marc Peter Deisenroth, Csaba Szepesvári, Jan Peters

Abstract

In some decision-making environments, successful solutions are common. If the evaluation of candidate solutions is noisy, however, the challenge is knowing when a "good enough" answer has been found. We formalize this problem as an infinite-armed bandit and provide upper and lower bounds on the number of evaluations or "pulls" needed to identify a solution whose evaluation exceeds a given threshold r_0. We present several algorithms and use them to identify reliable strategies for solving screens from the video games *Infinite Mario* and *Pitfall!* We show order of magnitude improvements in sample complexity over a natural approach that pulls each arm until a good estimate of its success probability is known.

Keywords: Multi-armed bandits, Planning under uncertainty, Stochastic optimization

1. Introduction

Consider the following trivial problem. A huge jar of marbles contains some fraction ρ of black (success) marbles and the rest white (failure) marbles. We want to find a black marble as quickly as possible. If the black marbles are sufficiently plentiful in the jar, the problem is simple: Repeatedly draw marbles from the jar until a black one is found. The expected sample complexity is $\Theta(1/\rho)$. This kind of generate-and-test approach is simple, but can be extremely effective when solutions are common—for example, finding an *un*satisfying assignment for a randomly generated CNF formula is well solved with this approach.

The corresponding noisy problem is distinctly more challenging. Imagine the marbles in our jar will be used to roll through some sort of obstacle course and (due to weight or balance or size) some marbles are more successful at completing the course than others. If we (quickly) want to find a marble that navigates the obstacle course successfully at least $r_0 = 25\%$ of the time, how do we best allocate our test runs on the course? When do we run another evaluation of an existing marble and when do we grab a new one out of the jar? How do we minimize the (expected) total number of runs while still assuring (with high probability) that we end up with a good enough marble?

We formalize this problem as an infinite-armed bandit and provide a lower bound on the number of arm pulls needed to find an arm with payoff above r_0. We describe and analyze several algorithms that are close to this lower bound. We include comparisons of these algorithms using data from two well known video games—*Infinite Mario* and *Pitfall!*

2. Infinite-Armed Bandit Problem

We define an arm as a probability distribution (D_a) over possible reward values within a bounded range $[r_{\min}, r_{\max}]$. When an arm is *pulled*, it returns a reward value (sampled from D_a). One arm a is preferred to another a' if it has a higher expected reward value, $E_{r_a \sim D_a}[r_a] > E_{r_{a'} \sim D_{a'}}[r_{a'}]$. Arms are sampled from an *arm space* S, possibly infinitely large. The distribution D over the arm space defines an infinite-armed bandit problem IB(D).

We seek algorithms that take a reward level r_0 as input and attempt to minimize the number of pulls needed to identify an arm with expected value of r_0 or more. This *sample complexity* has a dependence on D and r_0, as it may be likely or unlikely to encounter an arm with high enough reward. Specifically, define $\rho = P_{a \sim D}(E_{r_a \sim D_a}[r_a] \geq r_0)$ as the probability of sampling a "good enough" arm. We assume the domain is reward rich—specifically, that ρ is bounded away from zero. By allowing the agent to aspire to any reward level, this definition of the performance measure is akin to the earlier work of Wang et al. [2008], which is constrained to finding the optimal reward value.

Formally, we define an (ϵ, δ, r_0)-correct algorithm ALG for an IB(D) problem to be an algorithm that after a number of samples $T(\epsilon, \delta, r_0, D)$ (that is finite with probability 1) returns an arm a with expected value $E[r_a] \geq r_0 - \epsilon$ with probability at least $1 - \delta$. This formalism is a variation of the PAC-Bandit model [Even-Dar et al., 2002] to an infinite number of arms.

Regarding the motivation of our model, we target a class of optimization problems where standard local search approaches fail: in particular, relations between arms are either not predictive of relations between their associated reward values or such relations can be ignored without sacrificing much in terms of the number of evaluations needed to get an approximately optimal solution. We view planning as an optimization problem, with every possible plan being an 'arm' in the infinite bandit model described above.

Our claim is that some apparently hard planning problems can be solved via a sampling and testing approach that cannot be solved by algorithms that depend on the existence of local structure for search. We documented this phenomenon in the video games *Infinite Mario* and *Pitfall!* where policies can be very similar but have vastly different outcomes (one different action in a long sequence can lead the agent to failure as opposed to successfully completing a level).

In Section 3, we introduce the problem by considering the simple case in which arms are deterministic, $P(E[r_a] = r_a) = 1$. In Sections 4 and 5, we address the more general case of arms with stochastic rewards. We will mostly focus on arms with a Bernoulli distribution D_a over rewards, but our bounds are extendable to arbitrary distributions with bounded support.

2.1. Related Work

The framework used in our work is closely related to several models from the multi-armed bandit literature. While the case of a finite number of arms is well understood [Auer et al., 2002], in the past few years, papers discussing bandits with infinitely many arms have appeared [Kleinberg et al., 2008; Bubeck et al., 2008]. Our work extends the PAC-Bandit setting [Even-Dar et al., 2002] to an infinite number of arms [Wang et al., 2008], but we replace the assumption about the probability of drawing a near optimal arm with a given

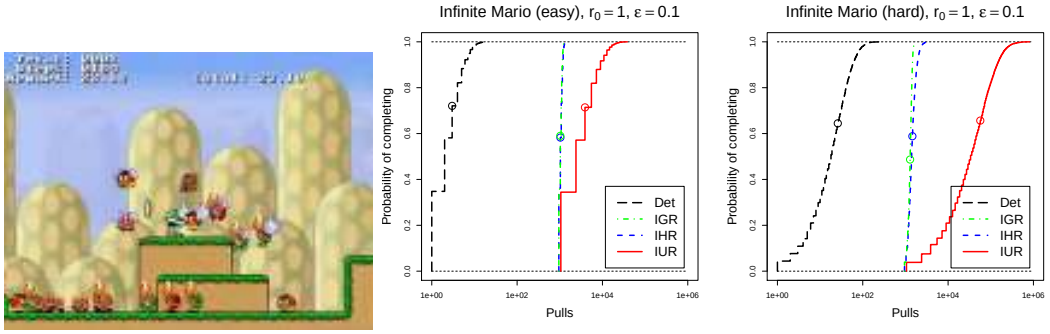

Figure 3.1: A screenshot of *Infinite Mario* and plots of the distribution of the sample complexity for an algorithm that pulls each arm once (x-axis log-scale). The distributions are plotted for 2 of the 50 *Infinite Mario* levels corresponding to the first (easy) and third (hard) quartiles. See the **Algorithms** section for more details.

target threshold. It is worth noting here that a typical assumption in the literature on continuous-armed bandits (one class of infinite-armed bandits) is that the structure in the arm space induces structure in the space of expected rewards of the arms. The mean-reward function is usually assumed to be Lipschitz and algorithms are created that take advantage of this smoothness assumption. In some cases, including our example problems, this assumption does not hold and algorithms that depend on it can fail.

Regarding our chosen performance measure, we depart from the often used cumulative regret setting and choose a setting that only requires an agent to have a good answer after some finite experimentation. This setting is related to some recent work [Bubeck et al., 2009; Audibert et al., 2010], but we chose the PAC-Bandit performance measure instead of the simple regret setting defined in the aforementioned work. Besides existing PAC-Bandit algorithms, we also draw algorithmic inspiration from the Hoeffding/Bernstein races framework [Maron and Moore, 1997; Heidrich-Meisner and Igel, 2009; Mnih et al., 2008], which we extend to our infinite-armed bandit setting, and from the empirical success of the Biased Robin algorithm from the Budgeted Learning framework [Madani et al., 2003], about which our analysis may offer some insight. Applicationwise, we target policy search and planning under uncertainty.

3. Experiment: *Infinite Mario*

Our first experiment used a version of *Infinite Mario* (a clone of the *Super Mario* video game, see the left panel of Figure 3.1) that was modified for the Reinforcement Learning Competition [Whiteson et al., 2010]. It was also used for other competitions [Togelius et al., 2010] and it is considered to be an interesting benchmark for planning and learning. The game is deterministic and gives us an opportunity to present a natural problem that illustrates the "reward richness" phenomenon motivating our work.

We treated starting screens in *Infinite Mario* as bandits, where each arm encodes an action sequence 50-steps long. In the experiments, the agent's goal was to reach a threshold

on the right side of the initial screen. The action set of the agent was restricted by removing the backward action (unnecessary for solving any level), resulting in 8 total actions and an arm space of size 8^{50}. Action sequences were tested in the actual game, assigning rewards of -1 if the agent was destroyed, 0 if it did not reach the goal in 50 steps, and a value of $100-t$, otherwise (where t was the number of steps taken before reaching the goal). Since the domain is deterministic, the agent simply sampled uniformly at random new arms until one was found with reward greater than 0. Sampling uniformly from the space of arms induces an unknown distribution D over the space of possible rewards, which is the distribution over arms described in section 2.

The average number of pulls needed to find a strategy for completing the first screen over a set of 50 levels ranged from 1 to 1000, with a median of 7.7 pulls and a mean of 55.7 (due to a few very difficult levels). Thus, testing just a handful of randomly generated action sequences was sufficient to find a successful trajectory in this game. The performance of the method is conveyed by the black (leftmost) lines in the plots in Figure 3.1 (the algorithms corresponding to the other curves are described in the next sections). The results show that nearly all screens were solved in well under 100 samples.

As an extension to this experiment, we 'chained' trajectories together to completely solve each of the 50 levels (as opposed to just treating the starting screens). Using a cap of 3000 pulls for each screen in a level, this simple method was able to complete 40 out of the 50 levels. (See the auxiliary material for links to videos of the discovered solutions.)

4. Sample Complexity Lower Bound

When the sampled arms are not deterministic, the problem of allocating pulls is more complex. The agent is faced with a choice between getting better accuracy estimates of previously sampled arms versus sampling new arms to find one with higher value. In the following, we state and prove a lower bound on the expected sample complexity of a correct algorithm for the case of Bernoulli arms.

Theorem 1 *Any (ϵ, δ, r_0)-correct algorithm for an $IB(D)$ problem has an expected sample complexity of at least $T(\epsilon, \delta, r_0, D) = \Omega(\frac{1}{\epsilon^2}(\frac{1}{\rho} + \log \frac{1}{\delta}))$.*

Due to space constraints, we leave the proof of the theorem for the auxiliary material (Appendix C). The key element of the proof is the use of Theorem 13 of Mannor et al. [2004], which states a lower bound on the expected sample complexity of any correct algorithm in the PAC-Bandit setting when the expected rewards of the arms are known up to a permutation.

5. Algorithms

When ρ is known, the infinite-armed bandit problem can be reduced to the classic PAC-Bandit setting and algorithms for that setting can be applied [Even-Dar et al., 2002; Mannor et al., 2004]. Specifically, if one samples a number of arms such that, with high probability, at least one has high expected reward (at least r_0) and then uses a version of Median Elimination [Mannor et al., 2004] to solve the resulting finite PAC-Bandit problem, the sample complexity bound will be $O(\frac{1}{\rho\epsilon^2} \log \frac{1}{\delta})$, roughly a factor of $\log \frac{1}{\delta}$ away from the lower bound. (Details available in Appendix F).

For the more general case when ρ is not known, this section introduces three new algorithms: one that is an incremental version of a naïve strategy [Even-Dar et al., 2002], one inspired by the Hoeffding Races framework [Maron and Moore, 1997; Heidrich-Meisner and Igel, 2009], and another that uses ideas from ballot-style theorems for random walks [Addario-Berry and Reed, 2008] to quickly reject unpromising arms.

All the algorithms we introduce have the structure of the **Generic Algorithm** (Algorithm 1): They sample an arm, make a bounded number of pulls for the arm, check if the arm should be accepted (and in this case, stop and return the arm) or rejected (sample a new arm from D and repeat). The decision rule for acceptance / rejection and when it can be applied is what differentiates the algorithms.

Algorithm 1:
GenericAlgorithm $(\epsilon, \delta, r_0, \mathit{RejectionFunction})$

$i = 1$, $\mathit{found} =$ **false**
for $i = 1, 2, ...$ **do**
  Sample a new arm $a_i \sim D$
  $\mathit{decision} = \mathit{RejectionFunction}(a_i, i, \epsilon, \delta, r_0)$
  if $\mathit{decision} = \mathit{ACCEPT}$ **then**
    return a_i
  end if
end for

5.1. Iterative Uniform Rejection (IUR)

Iterative Uniform Rejection (Algorithm 2) is an incremental version of the naïve strategy by Even-Dar et al. [2002]. The algorithm pulls an arm a fixed number of times to decide with high confidence if the arm has an expected reward less than or greater than $r_0 - \epsilon$. It samples arms in this manner until one with an estimated mean reward of at least $r_0 - \epsilon$ is found.

Algorithm 2: IterativeUniformRejection (ϵ, δ, r_0)

return $\mathit{GenericAlgorithm}(\epsilon, \delta, r_0, \mathit{UniformRejection})$

Function $\mathit{UniformRejection}(a, i, \epsilon, \delta, r_0)$

$n_0 = \frac{4}{\epsilon^2} \ln \frac{2i^2}{\delta}$
for $k = 1, 2, ..., n_0$ **do**
  Pull the arm to get reward $r_k \sim a$
end for
$\hat{r}_a = \frac{1}{n_0} \sum_{k=1}^{n_0} r_k$
if $\hat{r}_a < r_0 - \frac{\epsilon}{2}$ **then**
  return REJECT
end if
return ACCEPT

Theorem 2 *Algorithm* **Iterative Uniform Rejection** *is an* (ϵ, δ, r_0)*-correct algorithm for any IB*(D) *problem and its expected sample complexity is upper bounded by* $O(\frac{1}{\rho\epsilon^2} \log \frac{1}{\rho\delta})$.

The **IUR** algorithm is simple, correct, and achieves a bound close to the lower bound for the problem. We leave the proof of the theorem for the auxiliary material (Appendix A).

5.2. Iterative Hoeffding Rejection (IHR)

One problem with **IUR** is that it is very conservative in the sense of taking a large number of samples for each arm (with the dominant term being $\frac{1}{\epsilon^2}$). The algorithm does not take advantage of the fact that it may be possible to tell that an arm is highly unlikely to be better than $r_0 - \epsilon$ long before all n_0 pulls are performed. As a result, the algorithm wastes pulls deciding precisely *how* good or bad the arm is, when it just needs to know *whether* it is good or bad. **Iterative Hoeffding Rejection** (Algorithm 3) exploits the situation in which Δ_i, the difference between the expected reward of arm a_i and r_0, might be larger than ϵ, so an unpromising arm could be rejected before reach the decision threshold from **IUR**—an insight from the Hoeffding Races framework [Maron and Moore, 1997]. The main idea of the **IHR** algorithm is to maintain confidence intervals built using the Hoeffding bound around the empirical average for the sampled arm and to reject the arm as soon as the upper bound of the confidence interval drops below a certain threshold. If this threshold is not reached after a particular number of pulls, the arm is accepted.

Algorithm 3: IterativeHoeffdingRejection (ϵ, δ, r_0)

return *GenericAlgorithm*$(\epsilon, \delta, r_0,$ *HoeffdingRejection*$)$

Function HoeffdingRejection $(a, i, \epsilon, \delta, r_0)$

Let $\delta_0 = \frac{\delta}{2i^2}, j = 1$
$n_0 = \frac{4}{\epsilon^2} \ln \frac{1}{\epsilon\delta_0}$
for $j = 1, 2, ..., n_0$ **do**
  $r_j \sim a$; $\hat{r}_j^a = \frac{1}{j} \sum_{k=1}^{j} r_k$
  if $\hat{r}_j^a < r_0 - \sqrt{\frac{2\log(2j^2/\delta_0)}{j}}$ **then**
    return *REJECT*
  end if
  $j = j + 1$
end for
return *ACCEPT*

Theorem 3 *Algorithm* **Iterative Hoeffding Rejection** *is an* (ϵ, δ, r_0)*-correct algorithm for any IB*(D) *problem and its expected sample complexity is upper bounded by* $O(\frac{1}{\rho\epsilon^2} \log \frac{1}{\epsilon\rho\delta})$.

We leave the proof of the theorem for the auxiliary material (Appendix D). The analysis of the algorithm is tight in the worst case (consider the domain used to prove the lower bound in Theorem 1). Nevertheless, as we will show in the experiments section, the algorithm has a much better practical behavior than **IUR**. The reason can be understood by emphasizing

the differences between the arms in the upper bound for **IHR**. Define $\Delta_a = E[r_a] - r_0$ to be a random variable that encodes the difference between r_0 and the expectation of an arm sampled from D, and define Δ_- such that $\frac{1}{\Delta_-^2} = E[\frac{1}{\max(\epsilon^2, \Delta_a^2)} | \Delta_a < 0]$. ($\Delta_-$ is lower bounded by ϵ and it encodes the relevant difference for rejecting an arm if the arm has an expectated value smaller than r_0.) It can then be shown (see the proof in Appendix E) that:

Theorem 4 *The expected sample complexity of* **Iterative Hoeffding Rejection** *is upper bounded by* $O((\frac{1}{\epsilon^2} + \frac{1}{\rho\Delta_-^2}) \log \frac{1}{\epsilon\rho\delta})$ *with probability at least* $1 - \delta$.

For situations where Δ_- is larger than ϵ, the number of pulls needed to classify a 'bad' arm is actually much smaller (ignoring log factors, the difference is between $O(\frac{1}{\Delta_-^2})$ and $O(\frac{1}{\epsilon^2})$ pulls per arm). This difference is the reason why the algorithm has the potential to be much more useful then **IUR** in practice.

The algorithm can be improved by accepting an arm faster if the lower bound of the confidence interval for the empirical average of a particular arm becomes larger than r_0. Another immediate extension is to use Bernstein bounds [Mnih et al., 2008] instead of Hoeffding bounds to take advantage of the case where the distributions associated with each arm have low variance. We leave these straightforward improvements for an extended version of the paper.

5.3. Iterative Greedy Rejection (IGR)

Both **IUR** and **IHR** carefully decide whether an arm is good or bad before deciding to reject. As a consequence, with high probability, the first time they encounter a 'good' arm, they accept it. This strategy is reasonable in general, but it has one disadvantage: when the proportion of good arms is relatively low, these algorithms will spend a long time sampling and discarding bad arms. In some cases, a better strategy could be to reject faster—without being sure with high probability whether an arm is good or bad. This approach is permissible in our framework since failure to accept a good arm is only penalized in terms of sample complexity and does not compromise correctness. Related examples of empirically successful algorithms that quickly reject are Biased Robin [Madani et al., 2003] in the Budgeted Bandit setting and evolutionary algorithms in noisy settings [Fitzpatrick and Grefenstette, 1988].

We next describe an algorithm that implements such a strategy by constraining the empirical average of each sampled arm to stay above a certain threshold for it not to be rejected. While the worst-case bound we prove is a factor of $O(\frac{1}{\epsilon})$ worse than that of the other algorithms, the algorithm is strong experimentally for real data sets, and is still polynomial in the worst case.

Synthetic experiments that investigate various settings of the parameters (see Appendix G) indicate the upper bound is loose and point to a tighter upper bound that is similar to the one from Theorem 4 (with an advantage in some practical situations). We were unable to prove this bound though, and thus leave the improvement of the bound for **IGR** as an open problem.

Theorem 5 *Algorithm* **Iterative Greedy Rejection** *is an* (ϵ, δ, r_0)*-correct algorithm for any IB*(D) *problem and its expected sample complexity is upper bounded by* $O(\frac{1}{\rho\epsilon^3} \log \frac{1}{\epsilon\rho\delta})$*, if*

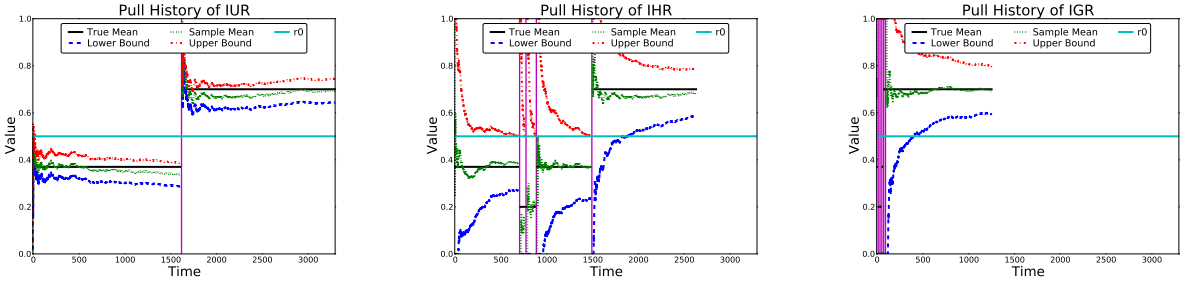

Figure 5.1: An illustration of the behavior of three algorithms in a simple domain.

Algorithm 4: IterativeGreedyRejection (ϵ, δ, r_0)

1: **return** *GenericAlgorithm*$(\epsilon, \delta, r_0,$ *GreedyRejection*$)$

Function GreedyRejection $(a, i, \epsilon, \delta, r_0)$

The function is identical to HoeffdingRejection with the exception of Line 5:
5′: **if** $\hat{r}_j^a < r_0 - \frac{\epsilon}{2}$ **then**

$r_0 > \epsilon$ (if $r_0 \leq \epsilon$, an algorithm can simply return the very first arm, which is guaranteed to be 'good' because its reward r satisfies $r \geq 0 \geq r_0 - \epsilon$.)

The proof can be found in Appendix B and it is one of the core contributions of this paper as it uses a new proving technique to upper bound the sample complexity.

The key idea of the proof is to interpret the evolution of the empirical average for a good arm as a random walk and then apply a ballot-style theorem [Addario-Berry and Reed, 2008] to bound the probability that the average will always be higher than a fixed threshold. Doing so allows us to lower bound the probability of accepting a good arm, which is the key to upper bounding the expected sample complexity.

One possible looseness in the analysis comes from ignoring the fact that in non-worst case scenarios, bad arms will be rejected before $n_{\max}(i)$ samples (which is why this strategy is successful in practice).

6. Illustrating the Algorithms

Figure 5.1 illustrates the behavior of all three algorithms in a simple setting where arms can take on three different values (0.20, 0.37, and 0.70) with equal probability ($r_0 = 0.5$, $\epsilon = 0.1$, $\delta = 0.01$). Thus, the goal is to recognize when a 0.70 arm is drawn. We note that this experiment is not meant to compare the algorithms empirically, but only to give insights about their different strategies for a simple example.

The first plot shows the behavior of IUR. Every time it draws an arm, it uses 1600 pulls to accurately estimate its payoff before deciding whether to accept or reject. In the plot, we show the algorithm's estimate of the mean of each arm it draws as it accumulates more data. We also plot a fixed interval of $\epsilon/2$ around this mean, which is the confidence interval of the estimate after 1600 pulls.

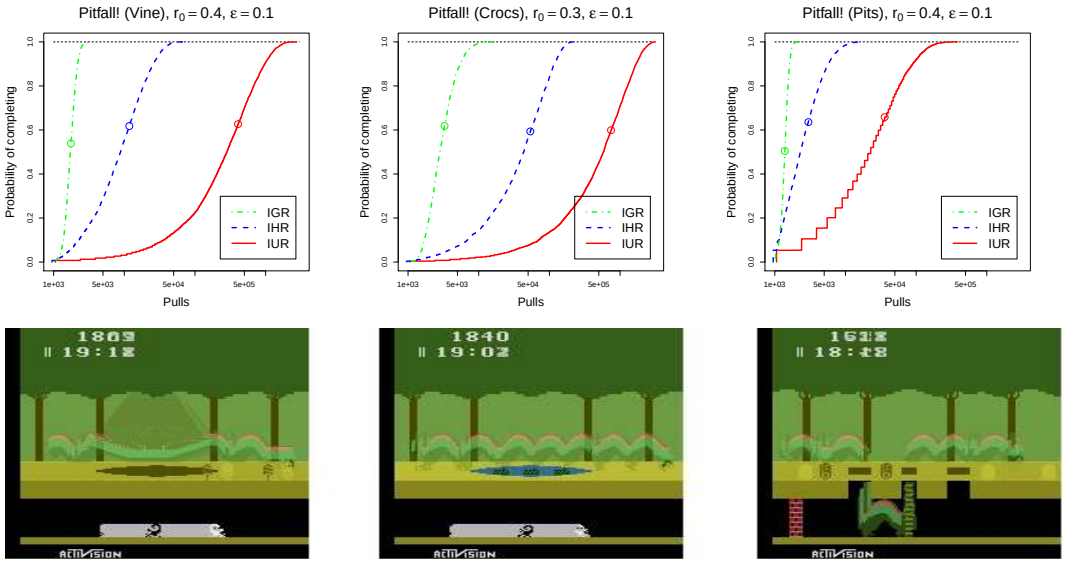

Figure 6.1: Plot of distribution of the sample complexity (pulls needed) for **IGR** (Iterative Greedy Rejection), **IHR** (Iterative Hoeffding Rejection) and **IUR** (Iterative Uniform Rejection) over a set of 5000 repetitions. The distributions are plotted for 3 different *Pitfall!* levels (shown along with a representation of a successful policy in the lower half of the figure). Average sample complexity for each algorithm is marked with a circle. All experiments used $\delta = 0.01$.

The second plot depicts the behavior of IHR, which maintains confidence intervals on the sample mean throughout. If the upper confidence interval drops below r_0, the arm is rejected. The result in the experiment is that **IHR** is able to reject 4 arms in the time it takes **IUR** to reject 1.

The third plot provides the analogous illustration of IGR. Note that this algorithm rejects an arm immediately if it fails on the first pull, and the sample mean must remain above r_0 long enough to verify the arm's payoff. The plot shows that the algorithm is able to test many different arms very quickly, occasionally discarding good arms. Ultimately, it settles in on a high scoring arm and evaluates it exhaustively.

7. Experiment: *Pitfall!*

Originally developed for the Atari 2600, *Pitfall!* is a game where the objective is to guide the protagonist through the jungle collecting treasure while avoiding items that harm him. In our experiments, interaction with the game was done via an emulator [JSt, 2008], which was modified to allow for software control. In the experiment, the agent's goal was defined simply as arriving at the right side of the screen on the top tier (some levels can be finished on the lower tier). The evaluation used the same 8 actionsfrom the *Infinite Mario* experiment. Stochasticity was added by randomly changing the joystick input to a centered joystick with no button press 5% of the time.

In this game, constructing a policy as a mapping from states to actions is difficult because it is unclear exactly what state representation to use. The Atari 2600 has 128 bytes of RAM, which means the actual size of the state space can be 8^{128}, much too large to effectively plan in directly. Treating the game as a collection of objects greatly simplifies the problem and has been used in *Pitfall!* for learning the dynamics of the first screen and navigating it successfully [Diuk et al., 2008], but requires prior domain knowledge to define what kind of interactions can occur between objects.

Because the issue of state in *Pitfall!* is problematic, one approach to planning in this domain is to not factor in state at all but to execute action sequences (which are policies) blindly (conditioned only on time, as opposed to state). The search space for sequences of 500 actions is 8^{500} possible plans. However, on average far fewer than 8^4 of the possible sequences actually need to be sampled uniformly at random before a successful one is found. This result is surprising, as the more difficult screens do not tolerate errors of more than a couple of pixels of placement. The success indicates that, like *Infinite Mario*, *Pitfall!* is reward rich. Figure 6.1 illustrates the results of running the three algorithms presented in this paper on the *Pitfall!* levels with stochasticity introduced. In all three cases, the random-walk-based **IGR** outperformed the races-based **IHR**, which outperformed the highly conservative **IUR** by a very large margin. The percentage improvement was greatest for the most challenging levels (Vine and Crocs). The same pattern can also be seen in Figure 3.1, where the algorithms were used on a deterministic domain without modification. Note that the deterministic strategy used for *Infinite Mario* is not successful in *Pitfall!* because of the noise we introduced. The deterministic algorithm assumes an arm is good if it is successful on the first pull, which can be very misleading. In Crocs, for example, the deterministic strategy results in over 90% of the runs erroneously returning a bad arm.

Of all the advantages of the infinite-armed bandit algorithms discussed here, the most significant may be the weak assumptions that are made: the only requirement is the probability of sampling a good enough arm be nonzero. It is thus an important topic of future research to compare the strategies from this paper with local search strategies and planning algorithms that are designed to take advantage of relations between arms or policies [Kleinberg et al., 2008; Bubeck et al., 2008; Bubeck and Munos, 2010].

8. Conclusion

We introduced an infinite-armed bandit framework that is tailored to optimization problems to which local search cannot be applied. It is closely related to the finite PAC multi-armed bandit model. We presented an almost-tight lower bound and three algorithms that solve the problem and provided analyses (including a novel random-walk-based method) proving that these algorithms achieve polynomial sample complexity bounds. We showed how a decision maker can balance between allocating pulls to get high-accuracy estimates and sampling new arms to find ones with higher expected rewards.

The framework models applications where good solutions are plentiful—where a good arm can be found by random sampling. It was shown that some non-trivial planning problems (such as two encountered in established video games) can be solved handily by exploiting this insight, even in the face of stochastic outcomes.

References

Jstella project, atari 2600, 2008. URL http://jstella.sourceforge.net/.

L. Addario-Berry and B. A. Reed. Ballot theorems, old and new. In *Horizons of Combinatorics*. Springer Berlin Heidelberg, 2008.

Jean-Yves Audibert, Sébastien Bubeck, and Rémi Munos. Best arm identification in multi-armed bandits. In *COLT*, 2010.

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *ML*, 2002.

Sébastien Bubeck and Rémi Munos. Open loop optimistic planning. In *Proceedings of the 23rd Annual Conference on Learning Theory (COLT)*, 2010.

Sébastien Bubeck, Rémi Munos, Gilles Stoltz, and Csaba Szepesvári. Online optimization in x-armed bandits. In *NIPS*, 2008.

Sébastien Bubeck, Rémi Munos, and Gilles Stoltz. Pure exploration in multi-armed bandits problems. In *ALT*, 2009.

Carlos Diuk, Andre Cohen, and Michael L. Littman. An object-oriented representation for efficient reinforcement learning. In *ICML*, 2008.

Eyal Even-Dar, Shie Mannor, and Yishay Mansour. Pac bounds for multi-armed bandit and markov decision processes. In *COLT*, 2002.

J. Michael Fitzpatrick and John J. Grefenstette. Genetic algorithms in noisy environments. *Machine Learning*, 1988.

Verena Heidrich-Meisner and Christian Igel. Hoeffding and bernstein races for selecting policies in evolutionary direct policy search. In *ICML*, 2009.

Olav Kallenberg. Ballot theorems and sojourn laws for stationary processes. *The Annals of Probability*, 1999.

Robert Kleinberg, Aleksandrs Slivkins, and Eli Upfal. Multi-armed bandits in metric spaces. In *ACM Aymposium on Theory of Computing*, 2008.

Omid Madani, Daniel J. Lizotte, and Russell Greiner. Budgeted learning, part 1: The multi-armed bandit case. Technical report, University of Alberta, 2003.

Shie Mannor, John N. Tsitsiklis, Kristin Bennett, and Nicolò Cesa-Bianchi. The sample complexity of exploration in the multi-armed bandit problem. *JMLR*, 2004.

Oded Maron and Andrew Moore. The racing algorithm: Model selection for lazy learners. In *Artificial Intelligence Review*, 1997.

Volodymyr Mnih, Csaba Szepesvári, and Jean-Yves Audibert. Empirical bernstein stopping. In *ICML*, 2008.

Lajos Takacs. *Combinatorial methods in the theory of stochastic processes*. Wiley New York, 1967.

Julian Togelius, Sergey Karakovskiy, and Robin Baumgarten. The 2009 mario AI competition. In *IEEE CEC 2010*, 2010.

Yizao Wang, Jean-Yves Audibert, and Rémi Munos. Algorithms for infinitely many-armed bandits. In *NIPS*, 2008.

Shimon Whiteson, Brian Tanner, and Adam White. The reinforcement learning competitions. *AI Magazine*, 2010.

Appendix A. Proof of Theorem 2

Proof

Sample Complexity. We will first show that there is a constant lower bound on the probability of the algorithm stopping, which will in turn help us show the expected sample complexity is finite. Let A_i be the event of accepting the i'th sampled arm ($i \in 1, 2, ...$), conditioned on rejecting the first $i-1$ arms. Let N be a random variable that stands for the number of arms sampled until the algorithm returns an arm, and SC be a random variable that stands for the sample complexity. Note that $P(A_i) = P(\text{accept arm } a_i | a_i \text{ is 'good'})P(a_i \text{ is 'good'}) + P(\text{accept arm } a_i | a_i \text{ is 'bad'})P(a_i \text{ is 'bad'}) \geq P(\text{accept arm } a_i | a_i \text{ is 'good'})P(a_i \text{ is 'good'}) \geq (1 - \frac{\delta}{2i^2})\rho \geq \frac{\rho}{2}, \forall i \in 1, 2, \ldots$ (where the third inequality holds due to an application of the Hoeffding bound). Thus $E[N] \leq \frac{2}{\rho}$ (by the properties of the geometric distribution, where A_i stands for "success").

The expected sample complexity is $E[SC] = E[\sum_{i=1}^{N} \frac{4}{\epsilon^2} \log \frac{2i^2}{\delta}] \leq \frac{4}{\epsilon^2} E[N \log \frac{2N^2}{\delta}] \leq \frac{8}{\epsilon^2}(E[N \log N] + \frac{1}{\rho} \log \frac{2}{\delta})$ (with the right hand side of the first equality determined by the Hoeffding bound). The expectation for sample complexity is taken with respect to both sampling the arms from D and noise in the pulls themselves. Now, $E[N \log N] \leq \sqrt{E[N^2]E[\log^2 N]} \leq \sqrt{\frac{4}{\rho^2} E[\log^2 N]}$ (where the first inequality is Cauchy-Schwarz and the second is due to the properties of the geometric distribution). To bound $E[\log^2 N]$ we use the fact that the function $\log^2(x)$ is concave for $x \geq 3$ (assuming a natural logarithm) and we apply Jensen's inequality to get $E[\log^2 N] \leq \log^2 E[N] \leq \log^2 \frac{2}{\rho}$.

So, for $E[N] \geq 3$, $E[N \log N] \leq \frac{2}{\rho} \log \frac{2}{\rho}$. For $E[N] < 3$, $E[N \log N] \leq E[N^2] < 9$. As a consequence, $E[N \log N] \leq \max(9, \frac{2}{\rho} \log \frac{2}{\rho})$ and the bound follows.

Correctness. The algorithm will stop (with probability 1, since its expected sample complexity is finite) and recommend either a 'good' arm (with reward $r \geq r_0 - \epsilon$) or a 'bad' one (reward $r < r_0 - \epsilon$). The failure probability is $P(\text{failure}) \leq P(\bigcup_{i \geq 1} \{\text{incorrect recommendation at step } i\}) \leq \sum_{i \geq 1} \frac{\delta}{2i^2} \leq \delta$ (the second inequality follows via the Hoeffding bound given the number of samples $n_0(i)$ for each arm). ■

Appendix B. Proof of Theorem 5

Before we give the actual proof we will restate for completeness (and to unify notation) Corollary 2.3 from Kallenberg [1999].

Theorem A 1 *[Kallenberg [1999]] Let $(Z_1, Z_2, ...)$ a finite or infinite, stationary sequence of random variables with values in $\mathbb{R}_+ = [0, \infty]$ and let $T_j = \sum_{i \leq j} Z_i$ and $\beta = E[Z_1]$. Then there exists a random variable σ, uniform over $(0, 1)$ (and independent of Z_i) such that:*

$$P_{\sigma, Z_i, i \geq 1}[\sup_{j>0} \frac{T_j}{j} \leq \frac{\beta}{\sigma}] = 1.$$

Now we can prove theorem 5.

Proof We note that while the proof is done for Bernoulli arms, it is extendable to arbitrary distributions with bounded support. We use the notation from Theorem 2 and we will only discuss the sample complexity (the correctness follows similarly to the other algorithms). As mentioned, the main challenge, given the aggressiveness of the rejection procedure, is to get a positive lower bound on the probability of accepting a good arm. Let B be the event of accepting an arm if the expected reward associated with that arm is r_0 (the bound follows immediately for all arms with $r \geq r_0$, since the probability of acceptance will be at least as large as for r_0). Define $X = \text{Bernoulli}(r_0)$ as a Bernoulli distributed random variable.

Define $Y = \frac{X - r_0 + \epsilon/2}{1 - r_0 + \epsilon/2}$ as an affine transformation of X. Then, let $\alpha = E[Y] = \frac{\epsilon/2}{1 - r_0 + \epsilon/2} \geq \frac{\epsilon}{2}$ (since $r_0 > \epsilon$). Since $Y = 1$ with probability r_0 and $Y = \frac{-r_0 + \epsilon/2}{1 - r_0 + \epsilon/2} < 0$ with probability $1 - r_0$, we can interpret the series $\{S_j\}$ (with $S_j = \sum_{i=1}^{j} Y_i$, with Y_i being i.i.d. samples of Y, and implicitly X_i being i.i.d. samples of X) as a random walk.

We will now make two simplifying assumptions (and then describe at the end of the proof how to remove them). We assume: (1) $r_0 - \frac{\epsilon}{2} \geq \frac{1}{2}$ and (2) $\frac{-r_0 + \epsilon/2}{1 - r_0 + \epsilon/2} \in \mathbb{Z}^-$ (the set of negative integers). Then, $\{S_j\}$ is a positively biased random walk on the integers with maximum step value 1. In this case, we can apply a classic ballot-style result that says that $P(S_j > 0, \forall j = 1, 2, ...) = \max(E[Y], 0) = \alpha$, for example, Theorem 3 from Addario-Berry and Reed [2008], which is based on a result by Takacs [1967].

But, $\alpha = P(S_j > 0, \forall j = 1, 2, ...) \leq P(S_j \geq 0, \forall j = 1, 2, ...) = P(\frac{\sum_{i=1}^{j} X_i}{j} \geq r_0 - \frac{\epsilon}{2}, \forall j = 1, 2, ...) = P(\hat{X}_j \geq r_0 - \frac{\epsilon}{2}, \forall j = 1, 2, ...) \leq P(\hat{X}_j \geq r_0 - \frac{\epsilon}{2}, \forall j = 1, 2, ..., n_{\max}(i))$ (where $\hat{X}_j$ is the empirical average after j samples and corresponds to $\hat{r}_j^a$ from Line 5′ of the algorithm). Thus, $P(B) \geq \alpha \geq \frac{\epsilon}{2}$.

So, as in Theorem 2, $P(A_i) \geq P(B)\rho \geq \frac{\epsilon\rho}{2}$. This fact implies $E[N] \leq \frac{2}{\epsilon\rho}$ and the proof for the expected sample complexity bound follows similarly to that of Theorem 3 with an extra $\frac{2}{\epsilon}$ factor in the bound that comes via the upper bound on $E[N]$.

To complete the proof, one needs to show that a 'bad' arm (with expected value smaller than $r_0 - \epsilon$) will be rejected after at most $n_{\max}(i)$ samples. This claim follows via the same application of the Hoeffding bound as for the previous algorithms (the probability that a 'bad' arm is accepted after $n_{\max}(i)$ samples is smaller than δ_0, which is enough to bound the probability of error for the entire execution of the algorithm).

To remove Assumptions 1 and 2, we will apply ballot-style theorems for random variables with real values. The result will then follow by applying Corollary 2.3 from Kallenberg [1999].

The goal of the last part of the proof is to complete the proof for IGR for the general case of random walks on the real numbers.

Let $Z = 1 - Y$ (where Y was defined in the main text as a transformation of the Bernoulli random variable X). A set of i.i.d. samples of X induces a set of i.i.d. samples of Y and implicitely of Z $(Z_1, Z_2, ...)$. But $(Z_1, Z_2, ...)$ is a stationary sequence of variables with $E[Z_1] = E[Z] = 1 - E[Y] = 1 - \alpha$. The support of Z is $\{0, \frac{1}{1-r_0+\frac{\epsilon}{2}}\} \subset \mathbb{R}_+$.

Let $T_j = \sum_{i \leq j} Z_i$. Then, the conditions for theorem 1 hold and so there exists a Uniform$(0, 1)$ random variable σ such that $P[\sup_{j>0} \frac{T_j}{j} \leq \frac{1-\alpha}{\sigma}] = 1$.

Let's note $V = \sup_{j>0} \frac{T_j}{j}$ and $W = \frac{1-\alpha}{\sigma}$ two transformed random variables of Z_i and σ respectively. We know that $P[V \leq W] = 1$ (a relation known under the name of absolute stochastic dominance or statewise stochastic dominance). It is known that this relation implies the usual notion of (first order) stochastic dominance (which states that a random variable Y stochastically dominates a random variable X if for all elements x in the support of X and Y, $P(Y > x) \geq P(X > x)$ or equivalently $P(Y \leq x) \leq P(X \leq x)$).

So since W stochastically dominates V in an absolute sense, it also dominates it in a first-order sense. We will pick $1 \in \mathbb{R}_+$ and it follows that $P(W \leq 1) \leq P(V \leq 1)$.

But $P(W \leq 1) = P_\sigma(\frac{1-\alpha}{\sigma} \leq 1) = P_\sigma(\sigma \geq 1 - \alpha) = 1 - (1 - \alpha) = \alpha$ (where the third equality follows immediately from the properties of the uniform distribution).

We've thus shown that $\alpha \leq P(V \leq 1)$. Then $\alpha \leq P(\sup_{j>0} \frac{T_j}{j} \leq 1) = P(\frac{T_j}{j} \leq 1, \forall j > 0) = P(\sum_{i \leq j}(1 - Y_i) \leq j, \forall j > 0) = P(S_j \geq 0, \forall j > 0)$. We know from the proof of theorem 5 in the main text that $P(S_j \geq 0, \forall j > 0) \leq P(\hat{X}_j \geq r_0 - \frac{\epsilon}{2}, \forall j = 1, 2, ..., n_{\max}(i))$. Thus the desired relation $(P(B) \geq \alpha)$ holds even when Assumptions 1 and 2 are removed, and we consider random walks with steps taking real values. The rest of the proof remains unchanged. ■

Appendix C. Proof of Theorem 1

We will shortly introduce the PAC-Bandit setting as its definition is needed for the proof. For a thorough introduction we refer the reader to Even-Dar et al. [2002].

The PAC-Bandit problem is defined as follows: Given n arms and two parameters ϵ and δ, stop in finite time with probability 1 and return an arm at most ϵ away from the arm with the highest expected reward among the n with probability at least $1 - \delta$. The lower bound we will use from Mannor et al. [2004] has the form $\Omega(\frac{1}{\epsilon^2}(n + \log \frac{1}{\delta}))$.

Proof We will use contradiction and assume there exists an (ϵ, δ, r_0)-correct algorithm ALG that solves any IB(D) problem with expected sample complexity $o(\frac{1}{\epsilon^2}(\frac{1}{\rho} + \log \frac{1}{\delta}))$. The goal is to show that ALG would imply a correct algorithm for the PAC-Bandit problem with expected sample complexity $o(\frac{1}{\epsilon^2}(n + \log \frac{1}{\delta}))$, which would contradict the known lower bound in the PAC-Bandit setting.

Let D be a categorical probability distribution with 2 values in its support: $0.5 - \epsilon$ (a bad arm) and $0.5 + \epsilon$ (a good arm) with probability mass $1 - x$ on the first value and x on the second. Let's choose an arbitrary $r_0 \in (0.5, 0.5 + \epsilon]$ (so that we follow the constraint that ρ is bounded away from zero). Then $\rho = x$. Now, define a PAC-Bandit problem as follows: assume we are given n arms, $n - 1$ of which have expected reward of $0.5 - \epsilon$ and one of which has expected reward of $0.5 + \epsilon$. To be precise, it is worth mentioning that we allow

the algorithms in the PAC-Bandit setting to resample arms and ignore any previous pulls taken for those arms (this actually makes the PAC-Bandit problem harder, so the lower bound still has to hold).

Let $x = \rho = \frac{1}{n}$. When we use ALG for the PAC-Bandit problem, each time the algorithm samples a new arm from the environment, it selects an arm uniformly at random, with replacement, from the n arms. Applying ALG, it will get the good arm with probability at least $1-\delta$ with an expected number of samples $o(\frac{1}{\epsilon^2}(\frac{1}{\rho} + \log\frac{1}{\delta})) = o(\frac{1}{\epsilon^2}(n + \log\frac{1}{\delta}))$, which contradicts the lower bound mentioned above (Theorem 13 Mannor et al. [2004]). ■

Appendix D. Proof of theorem 3

Proof Sample Complexity. We keep the same notation as in the proof of Theorem 2. We use $E[r_{a_i}]$ to represent the expected value associated with arm a_i, $\hat{r}_j^{a_i}$ the empirical average of a_i's rewards after its jth pull, and $CI(j) = \sqrt{\frac{2\log(2j^2/\delta_0)}{j}}$ the confidence interval for the empirical average at step j. Let $n_{\max}(i) = \frac{4}{\epsilon^2}\log\frac{1}{\epsilon\delta_0}$ be the maximum number of pulls for arm a_i.

Now, $P(A_i) = \rho(1 - P(\text{reject arm } a_i | \text{ arm } a_i \text{ is good})) = \rho(1 - P(\cup_{j=1}^{n_{\max}(i)}\{\hat{r}_j^{a_i} \notin [E[r_{a_i}] - CI(j), E[r_{a_i}] + CI(j)]\})) \geq \rho(1 - \sum_{j=1}^{n_{\max}(i)}\frac{\delta_0}{2j^2}) \geq \rho(1-\delta_0) \geq \frac{\rho}{2}$. So, as in Theorem 2, $E[N] \leq \frac{2}{\rho}$.

Since the sampling of each arm stops after at most $n_{max}(i)$ steps, $E[SC] \leq E[\sum_{i=1}^{N}\frac{4}{\epsilon^2}\log\frac{2i^2}{\epsilon\delta}]$ and then the sample complexity bound follows similarly to the proof of Theorem 2.

Correctness. The algorithm stops with probability 1 in finite time, and $P(\text{failure}) \leq \sum_{i\geq 1} P(\{\text{ incorrect recommendation at step } i\}) = \sum_{i\geq 1} P(\bigcup_{j=1}^{n_{\max}(i)}\{\hat{r}_j^{a_i} \notin [E[r_{a_i}] - CI(j), E[r_{a_i}] + CI(j)]\}) \leq \sum_{i\geq 1}\sum_{j\geq 1}\frac{\delta}{4i^2j^2} \leq \delta$. ■

Appendix E. Proof of Theorem 4

Proof Since this is a high probability statement, we can assume for the rest of the proof that we are in a situation where the algorithm commits no errors (which happens w.p. at least $1-\delta$ as it can be shown that, for the entire experiment, the algorithm fails w.p. at most δ). Let's define $SC(a)$ to be the same complexity of accepting or rejecting an arm a. Let's fix (for now) the total number of sampled arms to $N = n$, and fix an arm a, with $\Delta_a < 0$ (which we label as a 'bad' arm).

Then, using the Hoeffding bound, $SC(a) = \frac{4}{\max(\epsilon^2, \Delta_a^2)}\log\frac{2i^2}{\max(\epsilon, \Delta_a)\delta}$ (where i is the index of the arm among all n arms). Let's assume D is continuous (the discrete case is similar) and let's define $f(\Delta_a)$ to be the pdf of Δ_a. Then, since $a \sim D$, $E[SC(a)|a \text{ 'bad'}] = \int_{\Delta_a<0}\frac{4}{\max(\epsilon^2,\Delta_a^2)}\log\frac{2i^2}{\max(\epsilon,\Delta_a)\delta}f(\Delta_a)d\Delta_a \leq \log\frac{2n^2}{\epsilon\delta}\int_{\Delta_a<0}\frac{4}{\max(\epsilon^2,\Delta_a^2)}f(\Delta_a)d\Delta_a \leq \frac{4}{\Delta_-^2}\log\frac{2n^2}{\epsilon\delta}$.

So, $E[\text{samples from all 'bad' arms}|N=n] = \sum_{k=1}^{n}E[\text{samples from } k \text{ 'bad' arms}|N=n]P(k \text{ arms are bad}) \leq \frac{4}{\Delta_-^2}\log\frac{2n^2}{\epsilon\delta}\sum_{k=1}^{n}kP(k \text{ arms are bad}) = \frac{4(1-\rho)n}{\Delta_-^2}\log\frac{2n^2}{\epsilon\delta}$. Similarly, it can be shown that $E[\text{samples from all 'good' arms}|N=n] \leq \frac{4\rho n}{\epsilon^2}\log\frac{2n^2}{\epsilon\delta}$.

Then, for any N, $EB = E[\text{samples from all 'bad' arms}] \leq \sum_{n=1}^{\infty} \frac{4(1-\rho)n}{\Delta_-^2} \log \frac{2n^2}{\epsilon\delta} P(N = n) \leq \frac{4(1-\rho)}{\Delta_-^2} \log \frac{2}{\epsilon\delta} E[N] + \frac{8(1-\rho)}{\Delta_-^2} E[N \log(N)] \leq \frac{16(1-\rho)}{\rho\Delta_-^2} \log \frac{4}{\epsilon\rho\delta}$ (where the last inequality follows from the bounds for $E[N]$ and $E[N \log(N)]$ from Theorem 2). So, $EB = O(\frac{1}{\rho\Delta_-^2} \log(\frac{1}{\epsilon\rho\delta}))$.

Using a similar argument, one can show that $EG = E[\text{samples from all 'good' arms}] = O(\frac{1}{\epsilon^2} \log(\frac{1}{\epsilon\rho\delta}))$. Thus, $E[SC] = EB + EG = O((\frac{1}{\epsilon^2} + \frac{1}{\rho\Delta_-^2}) \log \frac{1}{\epsilon\rho\delta})$. ■

Appendix F. Known Probability to Get a "Good" Arm—Algorithms

The algorithms we introduced so far treat the scenario for which the concentration of rewards (ρ) is unknown. In this section, we will solve the problem for the case of known ρ by reducing it to the PAC-Bandit setting (that we introduced formally in Appendix C).

The high level strategy is to compute how many arms one needs to sample to get a "good" arm with high probability and then apply a PAC-Bandit algorithm to select the "good arm" from the sampled ones. Algorithm 5 implements this strategy. It is worth noting here that besides the standard algorithms (**Uniform Planning** (UP), **Sequential Elimination** (SE) or **Median Elimination** (ME) Even-Dar et al. [2002]), we can also apply an algorithm with a better upper bound due to our assumption that the domain is reward rich ($\rho > 0$)—the version of Median Elimination from Section 7.1 of Mannor et al. [2004] (which we label as **Median Elimination with Known Bias** (MEKB) from this point on). The last algorithm is built to take advantage of knowledge of the bias of the best arm.

Algorithm 5: PAC-Bandit Reduction $(\epsilon, \delta, r_0, \rho)$

1: Sample $n = \frac{1}{\rho} \log(\frac{2}{\delta})$ arms.
2: Execute a correct **PAC-Bandit Algorithm** with input $(\epsilon, \frac{\delta}{2}, n)$ on the n multi-armed bandit problem.
3: Return the output of the **PAC-Bandit Algorithm**

We denote as SC_{ALG} the expected sample complexity of an IB(D) algorithm that uses a PAC-Bandit algorithm ALG as a subroutine in Step 2.

Proof [Proof] **Correctness**. Let $a_1, a_2, ..., a_n \sim D$ i.i.d. Define event $A = \{\forall i \in \{1...n\}, E[a_i] < r_0\}$ (in words, A stands for the event that the expected value of all the sampled arms is smaller than r_0). Then, $P_{a_i \sim D, i \in \{1...n\}}(A) = (1-\rho)^n$. If we choose $n = \frac{1}{\rho} \log(\frac{2}{\delta})$, then $P(A) \leq \frac{\delta}{2}$.

When ALG is executed with parameters from Step 2 in Algorithm 5, it will fail with probability at most $\frac{\delta}{2}$. Thus, by the union bound on $P(A)$ and the failure probability of ALG, the algorithm will fail with probability at most δ and will otherwise return an arm with the desired properties. ■

The sample complexity obviously depends on the algorithm chosen. It is easy to show that, in the worst case, $SC_{UP} = O(\frac{1}{\rho\epsilon^2} \log \frac{1}{\delta} \log \frac{1}{\rho\delta})$, $SC_{ME} = O(\frac{1}{\rho\epsilon^2} \log^2 \frac{1}{\delta})$ and $SC_{MEKB} =$

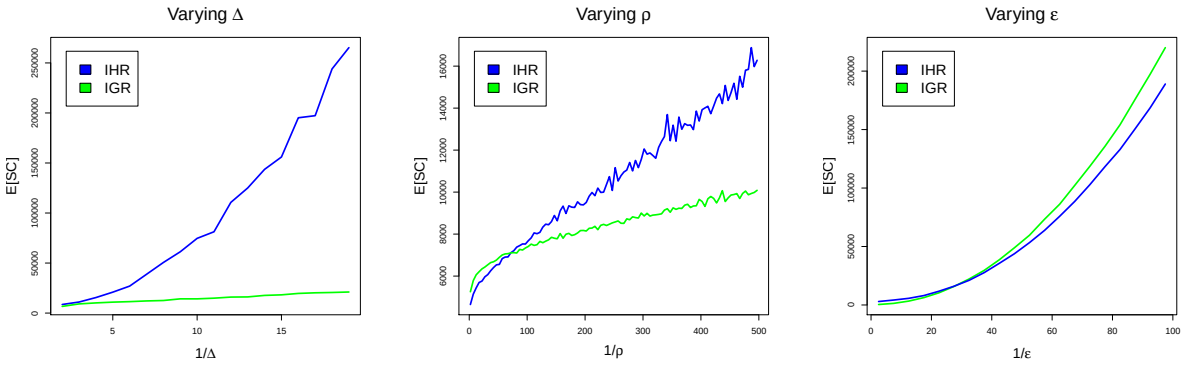

Figure G.1: Each data point in each graph is the average of 500 repetitions of the experiment. (a) In the left figure, we plot the dependence of the empirical average sample complexity (E[SC]) on $1/\Delta$. ϵ and ρ are fixed ($\epsilon = 0.05, \rho = 0.005$). (b) In the center figure, we plot the dependence of E[SC] on $1/\rho$. ϵ and Δ are fixed ($\epsilon = 0.05, \Delta = 0.4$). (c) In the right figure, we plot the dependence of E[SC] on $1/\epsilon$. ρ and Δ are fixed ($\rho = 0.005, \Delta = 0.4$). For all experiments, $\delta = 0.01$.

$O(\frac{1}{\rho\epsilon^2}\log\frac{1}{\delta})$. It can be observed that there is a gap of $O(\log\frac{1}{\delta})$ between the upper bound of MEKB and the lower bound from Theorem 1 (we leave the closing of this gap as an open problem)[1].

Appendix G. Experiments

G.1. *Pitfall!* and *Infinite Mario*

This section provides additional details about the experiments we ran for *Pitfall!* and *Infinite Mario*.

Four videos are included for *Pitfall!*. The experimental setting is described in the paper — it requires Harry to navigate from the left to the right side of the screen on the top tier without being killed or running out of time. Each video demonstrates one discovered trajectory, chosen as a representative of the others found. Each trajectory is run 20 times in each video. The locations of the videos are:

- Screen 3, Best Observed Policy, 3:11: http://www.youtube.com/watch?v=Jw-t7Ihe4Uc
- Screen 3, Worst Observed Policy, 3:36: http://www.youtube.com/watch?v=uAdXraDpUs0
- Screen 4, Best Observed Policy, 8:05: http://www.youtube.com/watch?v=r8Hr2Dc_NN0
- Screen 4, Worst Observed Policy, 7:06: http://www.youtube.com/watch?v=fOmAyGuXdvI

For *Infinite Mario*, we recorded videos of the performance of the simple strategy defined in the paper:

1. We remark here that while Median Elimination (and its variant MEKB) is the PAC-Bandit algorithm that offers the best sample complexity upper bound, it is not a practical algorithm.

- Success video (resulting from stitching together solutions to consecutive screens), 0:53: http://www.youtube.com/watch?v=tH5DRNrRS8I
- Partial success video, 0:49: http://www.youtube.com/watch?v=Wh8HGKIp7PQ
- First screens for 50 levels, 9:51: http://www.youtube.com/watch?v=tcJSQcVzRkc

Regarding the implementation, we took the *Infinite Mario* code from RLCompetition source code: http://code.google.com/p/rl-competition/ and RL-Glue code from RL-Glue source: http://glue.rl-community.org/wiki/Main_Page.

G.2. Synthetic Experiments

The role of this section is to further investigate the empirical sample complexity of IGR so as to give indications as to how the upper bound of IGR can be tightened and how does IGR compare with IHR in various scenarios.

We will use a simple but illustrative infinite bandit problem inspired by the sample complexity lower bound example. Let D (the probability distribution over the space of arms) be a categorical distribution with parameter Δ such that with probability ρ an arm is Bernoulli($\frac{1}{2}+\Delta$) (a 'good' arm) and with probability $1-\rho$ is is Bernoulli($\frac{1}{2}-\Delta$) (a 'bad' arm). The ρ parameter encodes (as usual) the reward richness of the domain while Δ encodes how large is the difference between the good and the bad arms (and plays the role of Δ_{-} from section 5.2).

The three parameters of interest are obviously ϵ, Δ and ρ. In figure G.1, in each graph, two parameters have fixed values and the third is varied. We plot the dependence of the empirical average sample complexity on the inverse of each parameter (since the sample complexity bounds depend on $1/\epsilon$, $1/\Delta_{-}$, and $1/\rho$). We only compare IGR with IHR, since the empirical sample complexity of IUR is always significantly larger than the other two algorithms and thus does not offer any interesting insight.

In the left graph from Figure G.1 it is visible that **IGR** increasingly dominates **IHR** as the difference between the good and the bad arms increases. We think this is one of the two factors that determine the success of **IGR** in practice, as it provides fast rejection of bad arms for a variety of types of bad arms (for a fixed accuracy ϵ and concentration of good arms ρ). The behavior is consistent for other fixed values of ϵ and ρ.

In the center graph from Figure G.1 the dependency of **IGR** and **IHR** on ρ is interesting. For values of $\rho > 0.01$, **IHR** dominates, while when the concentration of good arms decreases, **IGR** starts to have a better sample complexity. The better dependency on ρ as it decreases is the second factor that makes **IGR** strong empirically.

In the graph on the right side of Figure G.1 the dependency on ϵ is plotted. The performance comparison between **IGR** and **IHR** is reversed as compared to the experiment where ρ was varied. **IGR** dominates for values of $\epsilon > 0.04$ while **IHR** starts dominating when ϵ decreases further.

While the experiments (b) and (c) in Figure G.1 show that **IGR** and **IHR** are not totally ordered in terms of sample complexity performance, it is intuitive that for practical applications (where ϵ is usually fixed) **IGR** tends to behave better. All three synthetic experiments indicate that the bound from Theorem 5 is loose and the true bound of **IGR** is actually closer to Theorem 4.