Multi-armed Bandit Problems with Dependent Arms

Sandeep Pandey SPANDEY@YAHOO-INC.COM
Deepayan Chakrabarti DEEPAY@YAHOO-INC.COM
Deepak Agarwal DAGARWAL@YAHOO-INC.COM
Yahoo! Research, Sunnyvale, CA

Abstract

We provide a framework to exploit dependencies among arms in multi-armed bandit problems, when the dependencies are in the form of a generative model on clusters of arms. We find an optimal MDP-based policy for the discounted reward case, and also give an approximation of it with formal error guarantee. We discuss lower bounds on regret in the undiscounted reward scenario, and propose a general two-level bandit policy for it. We propose three different instantiations of our general policy and provide theoretical justifications of how the regret of the instantiated policies depend on the characteristics of the clusters. Finally, we empirically demonstrate the efficacy of our policies on large-scale real-world and synthetic data, and show that they significantly outperform classical policies designed for bandits with independent arms.

1. Introduction

Multi-armed bandit problems have been an active area of research since the 1950s. The problem can be stated as follows (J.C.Gittins, 1979): there are N arms, each having an unknown success probability of emitting a unit reward. The success probabilities of the arms are assumed to be independent of each other. The objective is to pull arms sequentially so as to maximize the total reward. Many policies have been proposed for this problem under the independent-arm assumption (Lai & Robbins, 1985; P.Auer et al., 2002). In this paper we drop this assumption and focus on the bandit problem where the arms are *dependent*. For example, consider a simple bandit instance which has 3 arms, with success probabilities θ_1, θ_2 and θ_3, where one also has a-priori knowledge that $|\theta_1 - \theta_2| < .001$. This constraint induces dependence between arms 1 and 2. Is it possible to construct policies that perform better than those for independent bandits by exploiting the similarity of the first two arms?

This question is not merely of theoretical interest. For instance, the lucrative Internet advertising business is based on selecting ads to display on webpages. This ad-selection problem can be cast as a bandit problem where each ad corresponds to an arm, displaying an ad corresponds to an arm pull, and user clicks are the reward. Ads with similar text, "bidding phrase," and advertiser information are likely to have similar click probabilities, and this creates dependencies between the arms of the bandit.

We formalize this problem in the paper. In particular, we propose a new variant of the multi-armed bandit problem where the arms have been grouped into *clusters*. For the toy example discussed previously, one can consider arms 1 and 2 together as a cluster, arm 3 as another cluster, and "reduce" the 3-arm problem to a 2-*cluster* problem. The latter may be more efficient to solve due to fewer number of clusters. We show that this intuition is indeed justified, and design policies that exploit such dependencies.

Our contributions: We formalize and study multi-armed bandits with dependent arms (henceforth, *dependent bandits*) for both the discounted and undiscounted reward scenarios. For the discounted reward objective, we find the optimal MDP-based solution for dependent bandits. At each timestep, this policy computes an (index, arm) pair for each cluster, then picks the cluster with the highest index and pulls the corresponding arm. However, as with independent bandits, computing the optimal is often infeasible and approximations are necessary. We provide error bounds on a simple approximation to the optimal policy.

For the undiscounted reward scenario, we first discuss an upper bound on the performance of any bandit policy. We then present a general and computationally

Appearing in *Proceedings of the 24th International Conference on Machine Learning*, Corvallis, OR, 2007.

feasible two-stage policy that pulls an arm by first choosing a cluster, and then choosing an arm within that selected cluster. We give three intuitive instantiations of this general policy. We explore their properties and the effects of cluster characteristics on their performance. We also empirically demonstrate, over simulated as well as real-world data, the significant performance boost they provide over traditional policies for independent bandits.

The rest of the paper is organized as follows. We discuss related work in Section 2, and formalize our problem in Section 3. We then discuss the discounted and undiscounted reward scenarios in Sections 4 and 5. Section 6 details empirical results on both real-world and simulated data. Finally, we conclude in Section 7.

2. Related Work

The multi-armed bandit problem has a rich literature. J.C.Gittins (1979) showed the optimal solution to the k-armed problem that maximizes the expected total discounted reward is obtained by decoupling and solving k independent one-armed problems, dramatically reducing the dimension of the state space (see also (Frostig & Weiss, 1999)). For the finite horizon undiscounted reward scenario, the asymptotic lower bound on regret has been shown to be $\Omega(\log T)$ while the average Bayes risk is bounded below by $\Omega((\log T)^2)$ for a large class of priors, where T is the total number of arm pulls (Lai & Robbins, 1985; Lai, 1987). Policies to achieve the lower bound have also been developed (Lai & Robbins, 1985; Agrawal, 1995; P.Auer et al., 2002; Kocsis & Szepesvári, 2006). In particular, the UCB1 scheme (P.Auer et al., 2002) achieves the $O(\log T)$ bound on regret uniformly instead of asymptotically.

The dependent bandit problem is also related to bandit problems with side observations (Wang et al., 2005). However, the latter assumes a separate process $\{X_t\}$ that provide additional information about the reward process at each time point; no such separate information about the reward process is present in the dependent bandit.

Another related area is active learning, where the goal is typically to build a classifier over the entire input space by sequentially choosing new examples to get labeled from portions of the space where the classifier confidence is low (MacKay, 1992; Schneider & Moore, 2002). The number of examples needed to achieve a given prediction accuracy has been studied thoroughly (Dasgupta, 2005). However, dependent arms have not been studied in this context.

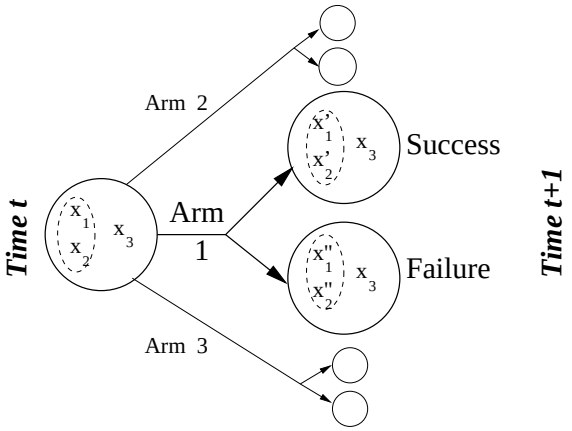


Figure 1. State evolution in the dependent bandit: Solid circles represent the states of all arms at any time, and dashed ellipses the clusters of arms. On pulling arm 1, state $x_1(x_2)$ changes to $x'_1(x'_2)$ or $x"_1(x"_2)$, but x_3 does not change.

3. Problem Formulation

Formally, the dependent bandit problem is defined as follows. There is a slot machine with N arms that are grouped into K known clusters. Each arm i has a fixed but unknown *success probability* θ_i. Let $[i]$ denote the cluster of arm i. Let $C_{[i]}$ be the set of all arms in cluster $[i]$ (including i itself), and let $C_{[i]}^{(-i)} = C_{[i]} \setminus \{i\}$.

In this paper, we assume that the dependencies among arms in a cluster can be described by a generative model. The form of the generative model is known but its parameters are unknown. In particular, let $s_i(t)$ be the number of times arm i generated a unit reward when pulled ("successes"), and $f_i(t)$ the number of "failures." Then, we assume that

$$s_i(t) \mid \theta_i \sim Bin(s_i(t) + f_i(t), \theta_i) \quad (1)$$

$$\theta_i \sim \eta(\pi_{[i]}) \quad (2)$$

where $\eta(.)$ is a probability distribution and $\pi_{[i]}$ is the parameter set for cluster $[i]$. Intuitively, π_C abstracts out the dependence of arms in cluster C on each other; given π_C, each arm is independent of all other arms.

In each timestep t, one arm i must be chosen ("pulled"), and it emits a reward $R(t)$ which is 1 with probability θ_i, and 0 otherwise. The objective is to pull arms so as to maximize the expected discounted reward

$$E[\text{Reward}_{disc}] = \sum_{t=0}^{\infty} \alpha^t E\left[R(t)\right] \quad (3)$$

where $0 < \alpha < 1$ is a discounting factor. Alternatively, we could maximize the expected undiscounted finite-time reward $E[\text{Reward}_{fin}(T)] = \sum_{t=0}^{T} E\left[R(t)\right]$ for a given time horizon T. Maximizing the objective function is equivalent to minimizing the expected regret

In each timestep t

- *Policy step:* A bandit policy is applied to choose the next arm to pull.
- *Update step:* The result of the arm pull (i.e., reward) is used to update the parameters of the policy.

Figure 2. Framework for bandit algorithms

$E[\text{Reg}(T)]$ until time T, where the regret of a policy measures the loss it incurs compared to a policy that always pulls the optimal arm, i.e., the arm with the highest θ_i.

Next, we give an equivalent formulation of our dependent bandit problem (Puterman, 2005), which is later used in deriving the optimal solution.

Equivalent State-space formulation: Associated with each arm i at time t is a state $x_i(t)$ containing sufficient statistics for the posterior distribution of θ_i given all observations until t: $x_i(t) = (s_i(t), f_i(t), \pi_{[i]}(t))$, where $\pi_{[i]}(t)$ is the maximum likelihood estimate of $\pi_{[i]}$ at time t.

If arm i is pulled at time t, it can transition to a "success" state with probability $p_i(x_i(t))$ and emit unit reward, or to a "failure" state and emit zero reward. Here, $p_i(x_i(t))$ is the MAP estimate of θ_i. This new observation (success or failure) changes $\pi_{[i]}(t)$, which simultaneously changes the states for each arm $j \in C_{[i]}$. For arms not in $C_{[i]}$, the state at $t+1$ is identical to that at t. For example, in Figure 1, pulling arm 1 changes both states x_1 and x_2 (due the the dependency between the two arms), while leaving x_3 intact.

Note the difference from the independent bandit problem: once an arm i is pulled, the state changes for not only i but also all arms in $C_{[i]}^{(-i)}$. Intuitively, the dependencies among arms in a cluster imply that the feedback $R(t)$ for one arm i also provides information about all arms in $C_{[i]}^{(-i)}$, thus changing their states.

Bandit algorithms: Typically, algorithms for bandit problems iterate over two steps, as shown in Figure 2. For the independent bandit, the update step needs to look only at the pulls and rewards of each arm in isolation. For the dependent bandit, the update step involves computing $\pi_{[i]}(t)$ given data on all prior arm pulls and corresponding rewards from each cluster; but this is typically a well understood statistical procedure. However, incorporating dependence information in the policy step is non-trivial and not well studied in the literature, and we discuss it in the following two sections.

4. Policy for Discounted Reward

We first discuss the optimal policy for dependent bandits with discounted reward (Equation (3)). Every timestep, it computes an (index, arm) pair for each cluster, and then picks the cluster with the highest index and pulls the corresponding arm. However, computing the index exactly is infeasible. We prove error bounds for a policy that approximates the optimal policy, and show that it gets arbitrarily close to the optimal policy with increasing computing power.

4.1. Optimal Policy

Consider the following MDP $\mathcal{M}$. Every state $\mathbf{i}$ is a vector of the number of successes and failures of *all* arms. When an arm is pulled, the corresponding state changes to one of two possible states depending on whether the reward was zero or one, as discussed in the equivalent state-space formulation of Section 3. Note that the prior $\pi_C(t)$ can be computed from the state vector itself, and the transition probabilities using $\pi_C(t)$. By the theory of dynamic programming, there is a value function $V(\mathbf{i})$ for every state $\mathbf{i}$:

$$V(\mathbf{i}) = \max_{1 \le a \le N} \left\{ \sum_{\mathbf{j} \in S(\mathbf{i},a)} p(\mathbf{i},\mathbf{j}) \cdot (R(\mathbf{i},\mathbf{j}) + \alpha V(\mathbf{j})) \right\}, \tag{4}$$

where α is the discounting factor, a represents any arm that can be pulled, $S(\mathbf{i}, a)$ the set of possible states this pull can lead to (i.e., the "success" and "failure" states), and $R(\mathbf{i}, \mathbf{j})$ is one when $\mathbf{j}$ is reached by a success from $\mathbf{i}$ and zero otherwise. The optimal policy for $\mathcal{M}$ picks the action (i.e., pulls the arm) that maximizes $V(\mathbf{i})$, and this is clearly the optimal policy for our bandit problem.

We show now that it is not necessary to solve the full MDP described above. Instead, we can solve slightly modified MDPs that are restricted to the individual clusters, and combine those results to achieve the same optimal policy. In particular, in the *restricted* MDP for cluster c, we allow each state to have a "retirement option," which is a transition to a final *rest* state with a one-time reward of M, as in (Whittle, 1980).

Define $V_c(\mathbf{i}_c, M)$ to be the value function for the restricted MDP for cluster c:

$$V_c(\mathbf{i}_c, M) = \max \left\{ M, \max_{a \in C_c} \sum_{\mathbf{j}_c \in S(\mathbf{i}_c,a)} p(\mathbf{i}_c,\mathbf{j}_c) \cdot (R(\mathbf{i}_c,\mathbf{j}_c) + \alpha V_c(\mathbf{j}_c, M)) \right\} \tag{5}$$

where $\mathbf{i}_c$ contains only the entries of $\mathbf{i}$ belonging to

cluster c. Let $a(\mathbf{i}_c, M)$ be the action (possibly retirement) that maximizes $V_c(\mathbf{i}_c, M)$, but with ties broken in favor of arm pulls. Define the *cluster index* γ_c as

$$\gamma_c = \inf\{M \mid V_c(\mathbf{i}_c, M) = M\} \tag{6}$$

Let the largest cluster index belong to cluster c^*.

Theorem 1. *The optimal policy at state* $\mathbf{i}$ *for the dependent bandit is to choose action* $a(\mathbf{i}_{c^*}, \gamma_{c^*})$.

The proof is similar to that of the optimality of Whittle's policy (Whittle, 1980) as shown by by Frostig and Weiss (1999).

Note that the optimal action $a(\mathbf{i}_{c^*}, \gamma_{c^*})$ cannot be the retirement option (which does not exist in the dependent bandit) otherwise we could reduce M further in Eq. (6), and γ_c would not be the infimum.

The key point is that the optimal policy can be computed by considering *each cluster in isolation*, instead of all N arms together. Thus, the size of the state space that we must work with is reduced from $\mathbb{R}^N$ to $\mathbb{R}^{N^*}$, where N^* is the size of the largest cluster. This can be a huge advantage when N is in the millions. Also note that this policy can be expressed in terms of an index γ_c on each cluster c, paralleling Gittins' Dynamic Allocation Indices for each arm of an independent bandit (J.C.Gittins, 1979).

If $V_c(\mathbf{i}_c, M)$ could be computed exactly, a binary search on M would give the value of the index γ_c. However, the unbounded size of the state space renders exact computation infeasible. In the following paragraphs, we look at an approximation to the optimal policy.

4.2. MDP-based Approximation Policy

We first show that an approximate $\hat{V}_c(\mathbf{i}_c, M)$ still gives bounds on the index. Let $\hat{\gamma}_c$ denotes the approximate value of index γ_c computed using $\hat{V}_c(\mathbf{i}_c, M)$, *i.e.*, $\hat{\gamma}_c = \inf\left\{M \mid \hat{V}_c(\mathbf{i}_c, M) = M\right\}$.

Lemma 1. *If* $V_c(\mathbf{i}_c, M) \in [\hat{V}_c(\mathbf{i}_c, M), \hat{V}_c(\mathbf{i}_c, M) + \delta]$, *then* $\gamma_c \in [\hat{\gamma}_c, \hat{\gamma}_c + \delta]$.

Proof.

$$\begin{aligned}
\gamma_c &= \inf_M\{V_c(\mathbf{i}_c, M) = M\} \\
&= \inf_M\{\hat{V}_c(\mathbf{i}_c, M) + \phi = M\} \quad \{\text{for some } 0 \le \phi \le \delta\} \\
&\le \inf_M\{\hat{V}_c(\mathbf{i}_c, M - \phi) = M - \phi\} \\
&\qquad \{\text{since } \hat{V}_c(\mathbf{i}_c, M - \phi) \le \hat{V}_c(\mathbf{i}_c, M), \forall \phi \ge 0\} \\
&= \hat{\gamma}_c + \phi \\
&\le \hat{\gamma}_c + \delta
\end{aligned}$$

Also, since $\hat{V}_c(i, \gamma_c) \le V_c(i, \gamma_c) = \gamma_c$, we know that $\hat{\gamma}_c \le \gamma$. Hence, $\gamma_c \in [\hat{\gamma}_c, \hat{\gamma}_c + \delta]$. □

A common method to approximate policies for large MDPs is to estimate the value function $V_c(\mathbf{i}_c, M)$ by a k-step lookahead: given the current state $\mathbf{i}_c$, it expands the MDP out to a depth of k, assigns to each state $\mathbf{j}_c$ on the frontier any value $\hat{V}_c(\mathbf{j}_c, M)$ between M and $\max\{M, 1/(1-\alpha)\}$, and then computes $\hat{V}_c(\mathbf{i}_c, M)$ exactly for this finite MDP. The maximum possible reward from any state onwards, without taking the retirement option, is $\sum_{k=0}^{\infty} 1 \cdot \alpha^k = 1/(1-\alpha)$, so $V_c(\mathbf{j}_c, M) \le \max\{M, 1/(1-\alpha)\}$. Also, $V_c(\mathbf{j}_c, M) \ge M$ since the retirement option immediately gives that reward. Thus, $|\hat{V}_c(\mathbf{j}_c, M) - V_c(\mathbf{j}_c, M)| \le \max\{M, 1/(1-\alpha)\} - M$, which translates to a maximum error of $\delta = \alpha^k \cdot (\max\{M, 1/(1-\alpha)\} - M)$ in $\hat{V}_c(\mathbf{i}_c, M)$. Note that even though errors may be made on an exponential number of states, their effect on δ is not cumulative; this is because only *one* best action is chosen for each state (due to the "max" in Eqs. 4,5) instead of, say, a weighted sum of these actions. From Lemma 1, the value of δ also bounds the error of the computed index $\hat{\gamma}_c$ from the optimal.

While Lemma 1 does bound the error, this bound may not be tight enough in practice. For example, an application that chooses ads to display on webpages from a database of $N \sim 10^6$ ads may be expected to converge to the best ad in (say) 10^7 displays. Equating this with the "effective time horizon" $1/(1-\alpha)$ yields a discount factor of $\alpha = 0.9999999$, for which the bounds on δ for reasonable values of the lookahead k are useless. Such problems are not just specific to our MDP-based policy; even the best known approximations for Gittins' index policy under the independence assumption break down when observations are few and $\alpha > 0.95$ (Chang & Lai, 1987). Such long time horizons are better handled under the *undiscounted* reward scenario; indeed, several policies for undiscounted reward actually approximate the Gittins' index for discounted reward, in the limit of $\alpha \to 1$ (Chang & Lai, 1987). Hence, we next discuss the undiscounted reward case, and propose our Two-Level Policy for it.

5. Policies for Undiscounted Reward

We first discuss lower bounds on the expected regret for the undiscounted reward objective, the conditions under which such results hold, and the constraints they place on the generative model of the dependent bandit. Then, we describe our proposed Two-Level Policy, and investigate how its performance depends on the characteristics of the clusters.

Policy step for timestep t

1. For each cluster i calculate a reward estimate ($\hat{r}_i(t)$) and variance estimate ($\hat{\sigma}_i(t)$).
2. Call $POL(\hat{r}_1(t), \hat{\sigma}_1(t), \ldots, \hat{r}_K(t), \hat{\sigma}_K(t))$ to select a cluster, say $c(t)$.
3. Call $POL(E[\theta_1], Var(\theta_1), \ldots)$ on the arms in cluster $c(t)$.

Figure 3. Two-level Policy (TLP): This is used as the *policy step* in the bandit algorithm framework of Figure 2.

5.1. Performance bounds

A lower bound on the expected regret $E[\text{Reg}(T)]$ after T pulls is known for independent bandits.

Theorem 2 ((Lai & Robbins, 1985))**.** *Under mild conditions on the success probabilities of the arms, for any policy with sub-polynomial regret (i.e., $o(T^a)$ for any $a > 0$), each suboptimal arm is pulled at least $O(\log T)$ times asymptotically and hence the expected regret $E[Reg(T)]$ is $O(\log T)$ as $T \to \infty$.*

Under mild conditions on the generative model (identifiability and the prior putting all mass on an open set), the theorem applies to dependent bandits as well, showing that asymptotically, the $O(\log T)$ growth rate for regret cannot be improved even if we exploit the additional information available from the clustering. However, it is worth noting that the constants involved in $O(\log T)$ can significantly affect performance both asymptotically and for a finite time horizon.

5.2. Two-level Policy

The generative model for dependence (Eqs. 1-2) draws the success probabilities θ_i of all arms in a cluster from the same distribution $\eta(.)$, and if this distribution is tightly centered around its mean, the θ_i values will all be similar. Thus, we could combine the observations from all arms of a cluster as if they had come from one hypothetical arm representing the entire cluster. This is the intuition behind the two-level policy (TLP): it uses as a subroutine any policy for independent bandits (say, POL), first running POL over the hypothetical "cluster-arms" to pick a cluster, and then inside that cluster to pick an arm. Figure 3 gives the details.

At the beginning of every timestep, the algorithm computes two numbers for every cluster: (1) its reward estimate $\hat{r}_i(t)$, corresponding to the success probability of the hypothetical cluster-arm, and (2) the estimated variance $\hat{\sigma}_i(t)$ of the reward estimate. We describe how these cluster reward and variance estimates are computed in more detail later. Based on these estimates, TLP chooses a cluster $c(t)$ by running POL on the cluster-arms. Then, it chooses an arm from within cluster $c(t)$ using POL again, using the mean and variance of the success probability θ_i of each arm i as its reward and variance estimate.

TLP incorporates intra-cluster dependence in two ways. First, by operating on the cluster-arms, it implicitly clubs all arms of a cluster together. Second, the estimates $\hat{r}_i(t)$ and $\hat{\sigma}_i(t)$ are computed based on the observed data *and* the generative model $\eta(.)$, if available. Note, however, that even if the form of $\eta(.)$ is unknown, TLP still uses the fact that the arms are partitioned into clusters, and performs well as a result.

We note that one specific instance of TLP was proposed in (Kocsis & Szepesvári, 2006; Pandey et al., 2007) but our work is significantly different: (1) our TLP formulation is more general, (2) we propose other instantiations of the general TLP, and (3) we investigate the performance of this model in terms of the clustering quality.

In this paper, we set the policy POL to be UCT (Kocsis & Szepesvári, 2006), an extension of UCB1 (P.Auer et al., 2002) that has $O(\log T)$ regret. At each timestep, UCT assigns to each arm i a priority $pr(i) = s_i/(s_i + f_i) + C_p \cdot \sqrt{(\log T)/T_i}$, where C_p is a constant, T_i is the number of arm pulls for i, and $T = \sum_i T_i$. The arm with the highest priority is pulled at each timestep. UCT reduces to UCB1 when $C_p = \sqrt{2}$.

TLP allows for several possible forms of $\hat{r}_i$ and $\hat{\sigma}_i$. We discuss three intuitive versions, which cover the spectrum from simple to complex. Then, we discuss how the various cluster characteristics affect performance.

Cluster Reward and Variance Estimates: Intuitively, to minimize regret, we must quickly find the best arm, and hence the cluster containing that arm. The cluster reward estimate $\hat{r}_i$ should tell us the expected *maximum* success probability of all arms in the cluster, so that the best cluster is chosen in step 2 of TLP as often as possible. A good reward estimate must be accurate and converge quickly (i.e., $\hat{\sigma}_i \to 0$ quickly). We propose three such strategies below.

The *MEAN* strategy is the simplest: it sets $\hat{r}_i$ to the average success rate of arms in the cluster, *i.e.*, $\hat{r}_i = \sum_j s_j/(\sum_j s_j + f_j)$ for all arms $j \in C_i$, and $\hat{\sigma}_i = (\sum_j s_j + f_j) \cdot \hat{r}_i \cdot (1 - \hat{r}_i)$ is the corresponding Binomial variance. The values of successes s_j and failures f_j in the above formulas are the *posterior* values obtained after taking the previous observations into account. For example, if the generative model η is estimated to

be $Beta(a, b)$, then the above formulas use $s'_j = s_j + a$ and $f'_j = f_j + b$. Note that in the *MEAN* strategy, the $\hat{r}_i$ of the cluster with the best arm is dragged down by its suboptimal siblings; the more arms in the cluster, the slower the convergence.

The *MAX* strategy picks from cluster i the arm $j \in C_i$ with the highest expected success probability $E[\theta_j]$, and sets $\hat{r}_i$ and $\hat{\sigma}_i$ to $E[\theta_j]$ and $Var(\theta_j)$ respectively. Thus, each cluster is represented by the arm that is currently the best in it. Intuitively, this value should be closer, as compared to *MEAN*, to the maximum success probability of cluster i. Also, $\hat{r}_i$ is not dragged down by the suboptimal arms of cluster i, reducing the adverse effects of large cluster sizes. However, it neglects all observations from the other arms in the cluster.

Our final strategy, called *PMAX*, achieves this by computing the posterior distribution of the maximum success probability among all the arms in C_i, given all observations from the cluster. Where analytic formulas for the posterior are not available, Monte Carlo sampling is used. These three strategies cover the spectrum of possibilities, from a simple but biased *MEAN*, to the computationally slow *PMAX* that gives the most unbiased estimate of the maximum success probability in the cluster.

Next we discuss how does the performance depend on the quality of the clustering, such as the "cohesiveness" of the clusters, the separation between clusters, and the sizes of the clusters.

5.3. Effects of cluster characteristics

Let the best arm be i^* from cluster *opt*. Intuitively, for TLP to find the best arm, two things must happen: cluster *opt* must become the top ranked cluster among all clusters, and arm i^* must be differentiated from its siblings in *opt*. Until the first is accomplished, cluster *opt* will receive only $O(\log T)$ pulls in step 2 of TLP and little progress can be made on the second. Thus, the effectiveness of TLP depends critically on the "crossover time" T_c for *opt* to finally achieve the highest reward estimate $\hat{r}_{opt}(T_c)$ among all clusters, and become the top ranked cluster. Below, we analyze T_c for *MEAN*. Doing this analysis for *MAX* and *PMAX* is more difficult and we leave it as future work.

Under the *MEAN* strategy, the expected reward estimate for each cluster i is given by $E[\hat{r}_i(T_i)] = \mu_i - \text{Reg}(T_i)/T_i$, where μ_i is the success probability of the best arm in cluster i, and $\text{Reg}(T_i)$ is the expected regret of *POL* in T_i pulls. For $POL = UCT$, $\text{Reg}(T_i)$ is $O(\log T_i)$. Thus, $E[\hat{r}_i(T_i)]$ grows monotonically towards μ_i for increasing T_i. Let the best arm i^* have success probability μ_{opt}. Among all arms not in *opt* cluster, let the best arm be in cluster s, with success probability μ_s. Define $\Delta = \mu_{opt} - \mu_s$ to be the separation between these top two clusters. Now, T_c is no more than the total pulls it takes to ensure that $\hat{r}_{opt}(T_{opt}) \geq \mu_s$, since that is the highest possible expected reward estimate for any other cluster.

$$\begin{aligned} T_c &\leq \min_T \{\hat{r}_{opt}(T_{opt}(T)) \geq \mu_s\} \\ &\leq \min_T \left\{ \left(\mu_{opt} - \frac{\text{Reg}(T_{opt}(T))}{T_{opt}(T)} \right) \geq \mu_s \right\} \\ &\leq \min_T \left\{ \Delta \geq \frac{\text{Reg}(T_{opt}(T))}{T_{opt}(T)} \right\} \end{aligned} \quad (7)$$

where $T_{opt}(T)$ is used to remind the reader that T_{opt} is a function of the total number of pulls T.

For $POL = UCT$, assuming that the known lower bounds on its regret (Kocsis & Szepesvári, 2006; P.Auer et al., 2002) are relatively tight,

$$\text{Reg}(T_i) \sim \min\{T_i \cdot \delta_i^{avg}, c \cdot A_i \cdot (\log T_i)/\delta_i^{min}\} \quad (8)$$

where δ_i^{avg} and δ_i^{min} are the average and minimum differences in success probability between the best arm in cluster i and its sibling arms, A_i the number of arms in cluster i (its size), and c is a constant. Thus, $\text{Reg}(T_i)/T_i \sim \delta_i^{avg}$ for small T_i and then it decays slowly until the functional form becomes $\text{Reg}(T_i)/T_i \sim cA_i/\delta_i^{min} \cdot (\log T_i)/T_i$ for large T_i. Using this in Equation (7), we can see the effect of the cluster properties on T_c.

(a) *Cluster separation* (Δ): As the best cluster becomes more separated from the rest, Δ increases and hence T_c decreases.

(b) *Cluster size* (A_{opt}): As the cluster size A_{opt} increases, T_c increases.

(c) *Cohesiveness* ($1 - \delta_{opt}^{avg}$): Equation 8 shows that high cohesiveness (i.e., small δ_{opt}^{avg}) leads to smaller T_c. In fact, when $(1 - 1/A_{opt}) \cdot \delta_{opt}^{avg} < \Delta$, cluster *opt* has the highest reward estimate from the start and $T_c = 0$, which is the best case for *MEAN*.

The worst case occurs when the clustering is not good: Δ is very small and δ_{opt}^{avg} is large, implying a large T_c. Since $\text{Reg}(T_{opt})/T_{opt}$ decays as $(\log T_{opt})/(T_{opt}\delta_{opt}^{min})$ for large T_c, having a larger δ_{opt}^{min} could reduce T_c marginally in this setting.

6. Experiments

We first report performance results on a large real-world dataset, showing more than threefold improve-

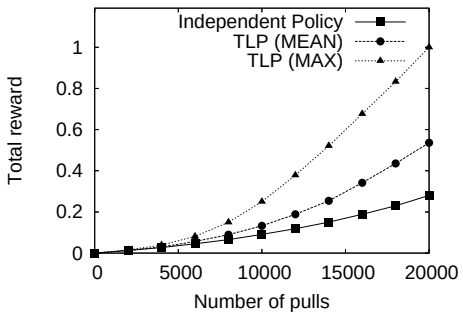


Figure 4. Performance over real-world data

ment in revenue over an independent bandit policy for the same number of arm pulls. Then, we empirically confirm our theoretical analysis of TLP and its versions, including its dependence on cluster characteristics; these are performed on synthetic datasets designed to demonstrate the desired effects. The time horizon ($\sim 10^4$) is too large for the discounted reward scenario to be applicable, as described in Section 4. All experiments were repeated 200 times, and the average performance is reported in all the results.

6.1. Performance improvement using TLP

The real-world dataset was collected from a week's worth of data in a large-scale ad-matching application. It has 10 clusters with 11.3 arms/cluster on average. The optimal cluster (i.e., cluster with the optimal arm) has 31 arms, a cohesiveness of $1 - \delta_{opt}^{avg} = 0.75$, and is separated from the next best cluster by $\Delta = 0.08$.

Figure 4 compares the reward of the *MEAN* and *MAX* versions of TLP against the UCB1 policy for independent bandits (P.Auer et al., 2002). The rewards are scaled from 0 to 1 for reasons of confidentiality.

We observe that both the *MEAN* and *MAX* versions of TLP perform much better than UCB1. This shows the importance of taking dependencies into account in the bandit policy. Also, *MAX* is better than *MEAN*; as shown in the following section, this effect is because the relatively large size of the optimal cluster compared to the others.

6.2. Empirical analysis of TLP

We start with a base dataset that has 10 clusters of 10 arms each, with a Beta distribution modeling the intra-cluster dependence. The optimal cluster has a cohesiveness of $\delta_{opt}^{avg} = 0.30$ and with a maximum success probability of $\mu_{opt} = 0.63$. All suboptimal clusters are identical, with a cohesiveness of $\delta_i^{avg} = 0.10$ and maximum success probability of $\mu_s = 0.50$. Hence, $\Delta = \mu_{opt} - \mu_s = 0.13$. Next, we study the effects of cluster characteristics by varying each parameter (i.e., Δ, δ_{opt}^{avg}, and number of arms) one at a time. While performance depends on the cluster characteristics of all clusters, we vary only the optimal cluster for simplicity. This is enough to show all the desired effects.

Figure 5(a) shows the effect of the cluster separation Δ on the total reward after $12,000$ pulls. All arms of the optimal cluster are translated by the same value to vary Δ. Both *MAX* and *MEAN* benefit with increasing Δ, as expected.

Figure 5(b) shows the effect of the number of arms A_{opt} in the optimal cluster. The performance of *MEAN* decreases significantly with increasing A_{opt}, since T_c increases linearly with increasing regret (Equation (7)), and the regret itself grows linearly with A_{opt} (Equations (8)). *MAX* is more robust to changes in A_{opt} since its reward estimate (and hence its T_c) depends only on the estimated success probability of *one* arm, unlike *MEAN* which depends on all.

Figure 5(c) shows the effect of cohesiveness of the optimal cluster. As expected, higher cohesiveness (i.e., smaller δ_{opt}^{avg} leads to better performance for *MEAN*, while *MAX* is not significantly affected.

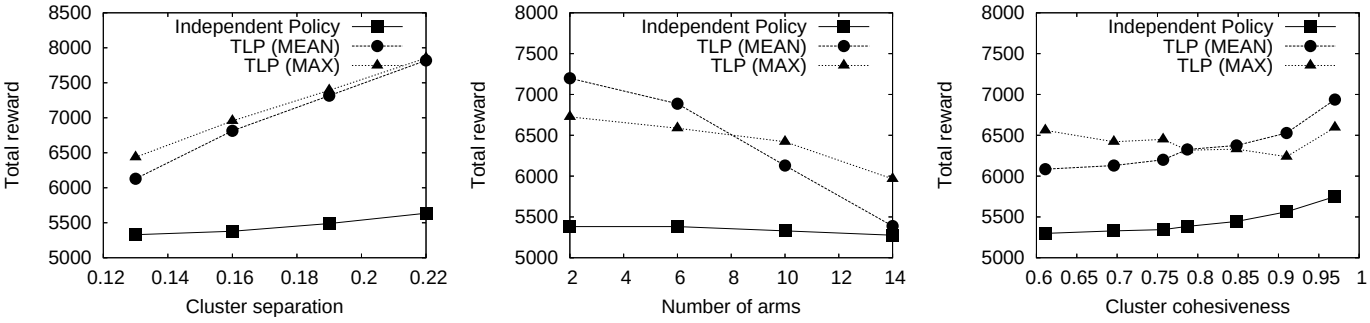
UCB1 is mostly unaffected by all these variations since the changes occur only for the 10 arms of the optimal cluster, and these changes are diluted when all 100 arms are considered together, as UCB1 does.

All of these results can be understood in terms of the bias-variance tradeoff. *MEAN* has low variance, since all observations T_i from a cluster i count towards the mean, but has high bias from the maximum success probability in the cluster. *MAX* has slightly higher variance, since only $T_i - O(\log T_i)$ observations count, but the bias is much lower than *MEAN*. In particular, the reward estimate under *MEAN* for the *opt* cluster is dragged down by all the suboptimal arms in *opt*, and as the number of arms increases, the bias of *MEAN* decreases more slowly and it performs worse. The *PMAX* version has the least bias, but its variance is too large for it to be effective; its performance was always dominated by *MEAN* or *MAX*, so we do not report it here.

7. Conclusions

We provide a framework to exploit dependencies among arms in high dimensional multi-armed bandit problems, when the dependencies are modeled using a generative model on the success probabilities.

We propose allocation rules that are both theoretically sound and computationally feasible for the discounted



(a) Revenue vs. Δ (b) Revenue vs. A_{opt} (c) Revenue vs. $(1 - \delta_{opt}^{avg})$

Figure 5. Effects of cluster characteristics. Only the optimal cluster is varied. Both *MEAN* and *MAX* are helped by increasing cluster separation, but *MEAN* is hurt far more than *MAX* with increasing number of arms.

and finite horizon reward scenarios. For discounted reward, we generalize the well known Gittins theorem and show that optimal allocation rules are obtained by decoupling the problem in terms of clusters instead of individual arms. For finite horizon reward, we provide a general two-stage allocation policy that selects a cluster followed by an arm in the selected cluster. We provide several instances of our general policy and also provide theoretical justifications on how the regret depends on the characteristics of the clusters. Empirically, we demonstrate the efficacy of our methods and show that they significantly outperform classical bandit solutions using real-world and synthetic data.

References

Agrawal, R. (1995). Sample mean based index policies with $O(\log n)$ regret for the multi-armed bandit problem. *Advances in Applied Probability*, *27*, 1054–1078.

Chang, F., & Lai, T. L. (1987). Optimal stopping and dynamic allocation. *Advances in Applied Probability*, *19*, 829–853.

Dasgupta, S. (2005). Coarse sample complexity bounds for active learning. *NIPS*.

Frostig, E., & Weiss, G. (1999). Four proofs of Gittins' multiarmed bandit theorem. *Applied Probability Trust*.

J.C.Gittins (1979). Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society, Series B*, *41*, 148–177.

Kocsis, L., & Szepesvári, C. (2006). Bandit based monte-carlo planning. *ECML*.

Lai, T. L. (1987). Adaptive treatment allocation and multi-armed bandit problem. *Annals of Statistics*, *15(3)*, 1091–1114.

Lai, T. L., & Robbins, H. (1985). Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, *6*, 4–22.

MacKay, D. (1992). Information-based objective functions for active data selection. *Neural Computation*, *4*, 590–604.

Pandey, S., Agarwal, D., Chakrabarti, D., & Josifovski, V. (2007). Bandits for taxonomies: A model-based approach. *SDM*.

P.Auer, N.Cesa-Bianchi, & P.Fischer (2002). Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, *47*, 235–256.

Puterman, M. L. (2005). *Markov decision processes: Discrete stochastic dynamic programming*. Wiley-Interscience. 2 edition.

Schneider, J., & Moore, A. (2002). Active learning in discrete input spaces. *The 34th Interface Symposium*.

Wang, C.-C., Kulkarni, S. R., & Poor, H. (2005). Bandit problems with side observations. *IEEE Transactions on Automatic Control*, *50(3)*, 338–355.

Whittle, P. (1980). Multi-armed bandits and the Gittins index. *Journal of the Royal Statistcal Society B*, *42*, 143–149.