FDR Control with Adaptive Sequential Experimental Design

Kevin Jamieson[†,⋆]

Optimizely[†], and
Paul G. Allen School of Computer Science & Engineering[⋆]
University of Washington
Seattle, WA 98195

October 15, 2017

1 Problem Statement

Adopting the nomenclature of *multi-armed bandits* (c.f., [1]), an *arm* is a stochastic source that when *pulled* by an algorithm, emits a sample drawn i.i.d. from a stationary distribution. That is, if an arm is a Bernoulli source with probability μ, then an algorithm that pulls the arm m times observes m i.i.d. draws of a Bernoulli random variable with mean μ. This work considers a single control arm–a baseline distribution– and n different alternative arms. Using as few samples as possible (or as little time as possible), the objective is to identify which alternatives have distributions with means that differ from the control arm. This work can be viewed as an advanced version of traditional A/B testing.

Consider a control arm with index 0 and n additional arms indexed $i = 1, \dots, n$ where samples from the jth arm for $j \in 0, 1, \dots, n$ are sub-Gaussian[1] distributed with mean μ_i. The arm's distribution either follows the null distribution $\mathcal{H}_0$, or the alternative distribution $\mathcal{H}_1$:

$$\begin{aligned} \mu_i &= \mu_0 \quad \text{if } i \in \mathcal{H}_0 \\ \mu_i &\neq \mu_0 \quad \text{if } i \in \mathcal{H}_1. \end{aligned}$$

The value of the means and the assignments to $\mathcal{H}_0$ or $\mathcal{H}_1$ are unknown. At each time t, the algorithm chooses an index i in $0, 1, \dots, n$ and observes a stochastic realization $X_{i,t}$ where $\mathbb{E}[X_{i,t}] = \mu_i$. The algorithm uses these noisy observations to make estimates about the assignments of each $i \in [n]$ to either $\mathcal{H}_1$ or $\mathcal{H}_0$.

The algorithm can pull the arms in any order it desires, and in particular, can use all the observations up to time $t-1$ to decide which arm to pull at time t. Given $\delta \in (0, 1)$, at every time $t \in \mathbb{N}$ the algorithm is tasked with outputting a set $\mathcal{S}_t$ such that $\mathbb{E}[\frac{|\mathcal{H}_0 \cap \mathcal{S}_t|}{|\mathcal{S}_t|}] \leq \delta$. The objective is to get the ratio $\mathbb{E}[\frac{|\mathcal{S}_t \cap \mathcal{H}_1|}{|\mathcal{H}_1|}]$ as close to 1 as possible, as fast as possible (i.e. for small t, taking as few samples as possible), while satisfying the constraint $\mathbb{E}[\frac{|\mathcal{H}_0 \cap \mathcal{S}_t|}{|\mathcal{S}_t|}] \leq \delta$. The set $\mathcal{S}_t$ should be interpreted as *discoveries* or *rejections of the null-hypothesis* (i.e., a rejection is equivalent to declaring $i \notin \mathcal{H}_0$). The constraint that $\mathbb{E}[\frac{|\mathcal{H}_0 \cap \mathcal{S}|}{|\mathcal{S}|}] \leq \delta$ is known as controlling the *false discovery rate* at level δ.

1.1 When does adaptivity help?

The baseline approach to the above problem is uniform sampling: at each time pull an arm chosen uniformly at random – this is the traditional A/B/n testing approach. Alternatively, adaptive methods use the past observations to choose which arms to pull next so that traffic allocations to the arms change over time. Adaptivity can speed up the time to significance in a couple different ways.

To see the most clear way that adaptivity helps, consider just two coins, one with probability of heads equal to μ_0, and the other with probability $\mu_1 \neq \mu_0$. Suppose we wish to identify which one has the largest probability of heads by just flipping the coins (i.e., we have no knowledge of μ_0, μ_1). If we flip each coin

[1] A random variable X is said to be sub-Gaussian if $\mathbb{E}[\exp(\lambda(X - \mathbb{E}[X])] \leq \exp(\lambda^2/2)$. Note that if X is a Bernoulli random variable in $\{0, 1\}$, then $2X - 1$, and trivially X, are sub-Gaussian.

m times, take the difference of their *empirical* means, the standard deviation of this difference up to first approximation acts like $\sqrt{1/m}$. Thus, to determine the sign of $(\mu_1 - \mu_0)$, we need $\sqrt{1/m} \ll |\mu_1 - \mu_0|$, which means each coin must be flipped at least $m \geq (\mu_1 - \mu_0)^{-2}$ times. This is simple since each of the two coins is sampled the same number of times.

Now suppose I have one control coin with mean μ_0 and n alternative coins each with mean μ_i for $i = 1, \ldots, n$. If I wanted to determine all those means μ_i such that $\mu_i \neq \mu_0$ (i.e., those in $\mathcal{H}_1$ versus $\mathcal{H}_0$), an intuitive strategy would be to sample each coin in a uniform or round-robin strategy and stop sampling the ith coin when it can safely be determined that $\mu_i \neq \mu_0$ using the logic of the preceding paragraph. It is intuitive that the total number of flips one would need to make before discovering all those coins μ_i such that $\mu_i \neq \mu_0$ would scale roughly as the sum of all the individual tests, plus all the coins in $|\mathcal{H}_0|$ we haven't decided about yet (and shouldn't ever declare as different than μ_0). Thus,

$$\text{(imprecise) \# samples for a simple adaptive algorithm} : \sum_{i \in \mathcal{H}_1} (\mu_i - \mu_0)^{-2} + |\mathcal{H}_0| \max_{j \in \mathcal{H}_1} (\mu_j - \mu_0)^{-2} \tag{1}$$

By stopping the sampling of coins we've already determined as being larger or smaller than μ_0, we are employing an *adaptive* strategy.

For a *non-adaptive* strategy, such as sampling each coin the same number of times for all time in a uniform or round-robin order (i.e., traditional A/B/n testing), we must wait until the coin i with the mean closest to μ_0 has been sufficiently sampled before discovering all the coins with different means; this coin must be sampled $\max_{j \in \mathcal{H}_1} (\mu_j - \mu_0)^{-2}$ times. But because we're employing a uniform strategy, *all* the coins would have had to be flipped this many times. Thus,

$$\text{(imprecise) \# samples for a non-adaptive algorithm} : (|\mathcal{H}_1| + |\mathcal{H}_0|) \max_{j \in \mathcal{H}_1} (\mu_j - \mu_0)^{-2}. \tag{2}$$

The difference between Equations 1 and 2 is largest when there is a large amount of *diversity* in the means of $\mathcal{H}_1$: some μ_i are very different than μ_0, and some are very close to μ_0. However, when the means of the alternatives μ_i are all roughly equal, the claimed number of samples of the two equations above appear equal – this is only because we were using rough, imprecise approximations. When there is a lack of diversity between the means there are subtle statistical effects that must be taken into account, and this is the second way adaptivity can help.

The above argument was very high-level and glosses over many statistical details that come into play when considering multiple discoveries and many sources of independent randomness. To provide intuition, if I flip $n = 10$ fair coins 100 times, we expect the vast majority of these coins to have around 50 heads. But as n gets very large, such as $n = 1000$, we start to expect to observe some rare events, such as at least one coin having greater than 90 heads. The point is that when the number of coins is very small, we expect *all* the empirical means to concentrate tightly around their true means. But when the number of coins increases, we expect at least one or a small number of coins to have large deviations about their true mean. Adaptivity identifies these misbehaving coins and allocates more effort to them to control their means, whereas a non-adaptive method would make every coin be sampled as much as the rare coin that deviates the most.

The theoretical limits of adaptive methods relative to non-adaptive methods is an active area of research [2] and beyond the scope of this work. Here we focus on the advantages of adaptivity through empirical studies modeled on common workloads experience on the Optimizely platform, particularly in Section 3.

2 Proposed Bandit Algorithm

For any arm $i \in [n]$ we will assume the existence of an anytime confidence interval ϕ that satisfies $|(\widehat{\mu}_{i,T_i(t)} - \widehat{\mu}_{0,T_0(t)}) - (\mu_i - \mu_0)| \leq \phi(\widehat{\mu}_{i,T_i(t)}, T_i(t), \widehat{\mu}_{0,T_0(t)}, T_0(t), \delta)$ for all t simultaneously with probability at least $1 - \delta$, where $T_j(t)$ denotes the number of times the jth arm has been pulled up to time t and $\widehat{\mu}_{i,T_i(t)}$ is the empirical mean. For the purposes of our simulations, we employ the confidence interval from the *Optimizely Stats Engine* that in actuality takes more information about the samples to construct the confidence bound than just the empirical mean and number of samples.

The algorithm is based on the popular upper confidence bound heuristic that has been successfully applied for balancing exploration-versus-exploitation [1], estimation [3], and most relevant to our situation, best-arm

Algorithm 1: FDR Control for MAB with control arm

Input: Confidence δ, confidence interval $\phi(\cdot,\cdot)$, n arms and control arm

Initialize: Pull each arm $a \in \{0\} \cup [n]$ once and let $T_a(t)$ denote the number of times arm a has been pulled up to time t.

Set for all time t and $a \in [n]$

$$\mathrm{UCB}_a(t,\delta) = \widehat{\mu}_{a,T_a(t)} - \widehat{\mu}_{0,T_0(t)} + \phi_+(\widehat{\mu}_{a,T_a(t)}, T_a(t), \widehat{\mu}_{0,T_0(t)}, T_0(t), \delta)$$
$$\mathrm{LCB}_a(t,\delta) = \widehat{\mu}_{a,T_a(t)} - \widehat{\mu}_{0,T_0(t)} - \phi_-(\widehat{\mu}_{a,T_a(t)}, T_a(t), \widehat{\mu}_{0,T_0(t)}, T_0(t), \delta)$$

For rounds $t = n+2, n+3, \ldots$

  For all $a \in [n]$, **if** $\widehat{\mu}_{a,T_a(t)} - \widehat{\mu}_{0,T_0(t)} > 0$

$$p_{a,T_a(t)} = \sup\left\{\rho \in (0,1) : \widehat{\mu}_{a,T_a(t)} - \widehat{\mu}_{0,T_0(t)} \le \phi_+(\widehat{\mu}_{a,T_a(t)}, T_a(t), \widehat{\mu}_{0,T_0(t)}, T_0(t), \rho)\right\}$$

  else

$$p_{a,T_a(t)} = \sup\left\{\rho \in (0,1) : -\widehat{\mu}_{a,T_a(t)} + \widehat{\mu}_{0,T_0(t)} \le \phi_-(\widehat{\mu}_{a,T_a(t)}, T_a(t), \widehat{\mu}_{0,T_0(t)}, T_0(t), \rho)\right\}$$

  Set $\mathcal{S}$ as the output of Benjamini-Hochberg acting on $\{p_{a,T_a(t)}\}_{a=1}^n$

  Pull arm $\widehat{a}$ **where**

$$\widehat{a} = \arg\max_{a \in [n]\setminus\mathcal{S}} \max\{\mathrm{UCB}_a(t,\delta), -\mathrm{LCB}_a(t,\delta)\}$$

  If $T_0(t) < \max_{a\in[n]\setminus\mathcal{S}} T_a(t)$, **pull arm** 0

or best-subset identification [4, 2]. Algorithm 1 combines the best-arm identification strategy of [4] with the celebrated method of Benjamini-Hochberg [5, 6] for controlling false discovery rate. The algorithm has known theoretical correctness and sample complexity guarantees [7] that we omit for now.

2.1 Comparing different algorithms

To benchmark the different adaptive allocation methods, we consider a control arm and n alternative arms. Each alternative arm $i \in [n]$ obeys $i \in \mathcal{H}_0 \iff \mu_i = \mu_0$ or $i \in \mathcal{H}_1 \iff \mu_i = \mu_0 + \Delta$. We consider different values of $n, |\mathcal{H}_1|, \mu_0, \Delta$, informed by the workloads used by Optimizely customers.

At each time t each algorithm outputs a set $\mathcal{S}$. There are two metrics of interest: (1) the false-discovery-rate $\frac{|\mathcal{S}\cap\mathcal{H}_0|}{|\mathcal{S}|}$, and (2) the proportion of true discoveries $\frac{|\mathcal{S}\cap\mathcal{H}_1|}{|\mathcal{H}_1|}$. The first measures the rate at which what is declared as significant, is actually not. And the second measures the proportion of true non-nulls are declared as significant. Both are in $[0,1]$ and we wish the first to be bounded by δ, and the second to be as close to 1 as fast as possible.

Because of the way we have designed these algorithms with anytime confidence bounds, they are guaranteed to control FDR at level δ [7]. Ideally, an algorithm would actually have an FDR rate of exactly δ (and not much smaller) indicating that the algorithm is being as aggressive as possible while staying inbounds of the FDR constraint. So what we really want to see is the second metric, $\frac{|\mathcal{S}\cap\mathcal{H}_1|}{|\mathcal{H}_1|}$, increasing as fast as possible.

We consider three algorithms:

- Uniform: At each time, the arm in $\{0\} \cup [n]$ that has been pulled the fewest number of times is pulled.
- Successive Elimination: At each time, the arm in $\{0\} \cup ([n] - \mathcal{S})$ that has been pulled the fewest number of times is pulled. That is, if an arm is declared as a discovery on the previous round, it is not a candidate for being pulled.

- Bandit: The procedure of Algorithm 1.

For various values of n and $\mathcal{H}_1$ the Figures 1, 2, 3 consider the performance of the algorithms for different values of μ_0 and Δ.

3 From one to B samples look-ahead

For practical reasons, one may not always immediately observe the stochastic reward $X_{i,t}$ after one requests to sample arm i at time t. Instead, at some wall-clock time τ a serving platform sets a static sampling allocation p_τ that prescribes with what probability each arm $0, 1, \ldots, n$ should be sampled from. Until this sampling allocation vector is replaced, all requests for an index to pull will be sampled independently from p_τ and the stochastic rewards are returned asynchronously at different times after time τ. Given a limit on how often p_τ can be replaced, the task is to choose p_τ. Due to the importance of this problem in web settings, this problem has been studied previously [8, 9]

3.1 Batching algorithm

In what follows, we make a number of simplifying assumptions. We work in discrete time such that at each time $t \in \mathbb{N}$ the algorithm prescribes an allocation p_t, exactly B samples are drawn from p_t and their stochastic rewards are immediately observed. Choosing p_t with the caveat that B samples will be sampled from it before the first stochastic reward is observed is a surrogate for the limitations on how often the allocation can be updated and the delays in receiving the stochastic rewards of prescribed indices. The algorithm will proceed exactly as in Algorithm 1 except Line 7 of Algorithm 1 will be replaced with sampling B indices from p_t, pulling the arms associated with those indices, and then updating the statistics. Our strategy is inspired by the so-called "hallucination" method of [9]. Recall the notation of Algorithm 1.

To ease notation, set

$$
\begin{aligned}
&\text{If } \mathrm{UCB}_a(t,\delta) \geq -\mathrm{LCB}_a(t,\delta):\\
&\quad \widehat{\Delta}_a = \widehat{\mu}_{a,T_a(t)} - \widehat{\mu}_{0,T_0(t)}\\
&\quad \phi_a = \phi_+(\widehat{\mu}_{a,T_a(t)}, T_a(t), \widehat{\mu}_{0,T_0(t)}, T_0(t), \delta)\\
&\text{Else:}\\
&\quad \widehat{\Delta}_a = -\widehat{\mu}_{a,T_a(t)} + \widehat{\mu}_{0,T_0(t)}\\
&\quad \phi_a = \phi_-(\widehat{\mu}_{a,T_a(t)}, T_a(t), \widehat{\mu}_{0,T_0(t)}, T_0(t), \delta)
\end{aligned}
$$

and let $T_a = T_a(t)$, and $\widehat{i} = \arg\max_{a=1,\ldots,n} \widehat{\Delta}_a + \phi_a$. Fix some p_0 that we will determine later. Now let $q \in \mathbb{R}_+^n : \sum_{i=1}^n q_i = 1$ be the minimizer of

$$
\max_{a=1,\ldots,n} \widehat{\Delta}_a + \sqrt{\frac{\frac{1}{T_a + Bq_a(1-p_0)} + \frac{1}{T_0 + Bp_0}}{\frac{1}{T_a + Bq_{\widehat{i}}(1-p_0)} + \frac{1}{T_0 + Bp_0}}}\phi_a
$$

For any value of p_0, we note that $\widehat{i} = \arg\max_{a=1,\ldots,n} q_a$ since the square-root term is at least 1, and this value is achieved with $q_{\widehat{i}} = \max_{i=1}^n q_i$. We conclude that the value of the optimal solution is equal to $c = \widehat{\Delta}_{\widehat{i}} + \phi_{\widehat{i}}$, so rearranging we find

$$
\frac{1}{T_a/B + q_a(1-p_0)} = \left(\left(\frac{c - \widehat{\Delta}_a}{\phi_a}\right)^2 - 1\right)\frac{1}{T_0/B + p_0} + \left(\frac{c - \widehat{\Delta}_a}{\phi_a}\right)^2 \frac{1}{T_a/B + q_{\widehat{i}}(1-p_0)}
$$

and

$$
q_a(1-p_0) = \left[\frac{1}{\left(\left(\frac{c-\widehat{\Delta}_a}{\phi_a}\right)^2 - 1\right)\frac{1}{T_0/B+p_0} + \left(\frac{c-\widehat{\Delta}_a}{\phi_a}\right)^2 \frac{1}{T_a/B + q_{\widehat{i}}(1-p_0)}} - T_a/B\right]_+ \quad \text{for } p_0 \text{ given} \tag{3}
$$

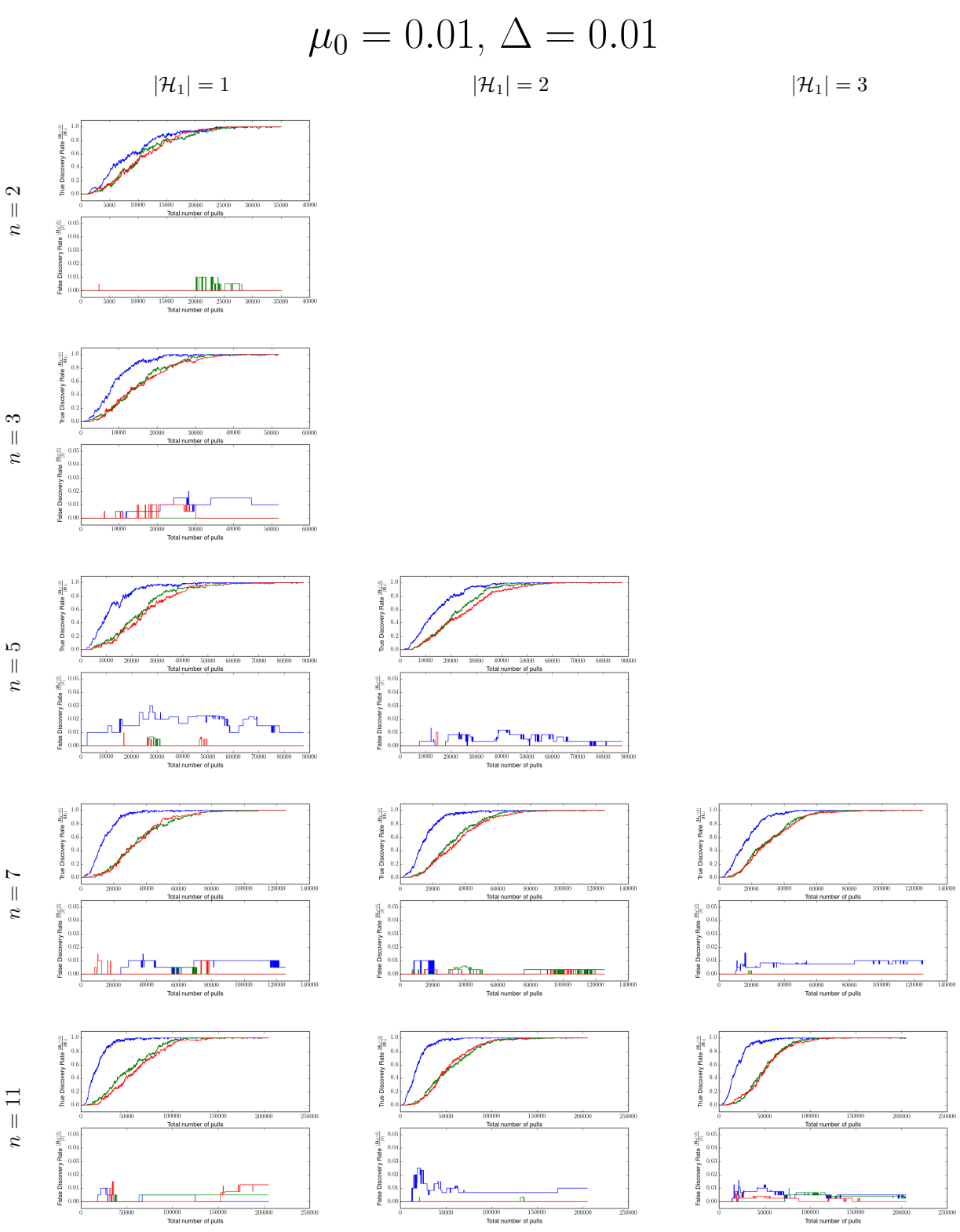

Figure 1: Each plot represents 100 trials of the different allocation methods run with confidence level 0.05. Within each setting of $(n, |\mathcal{H}_1|)$

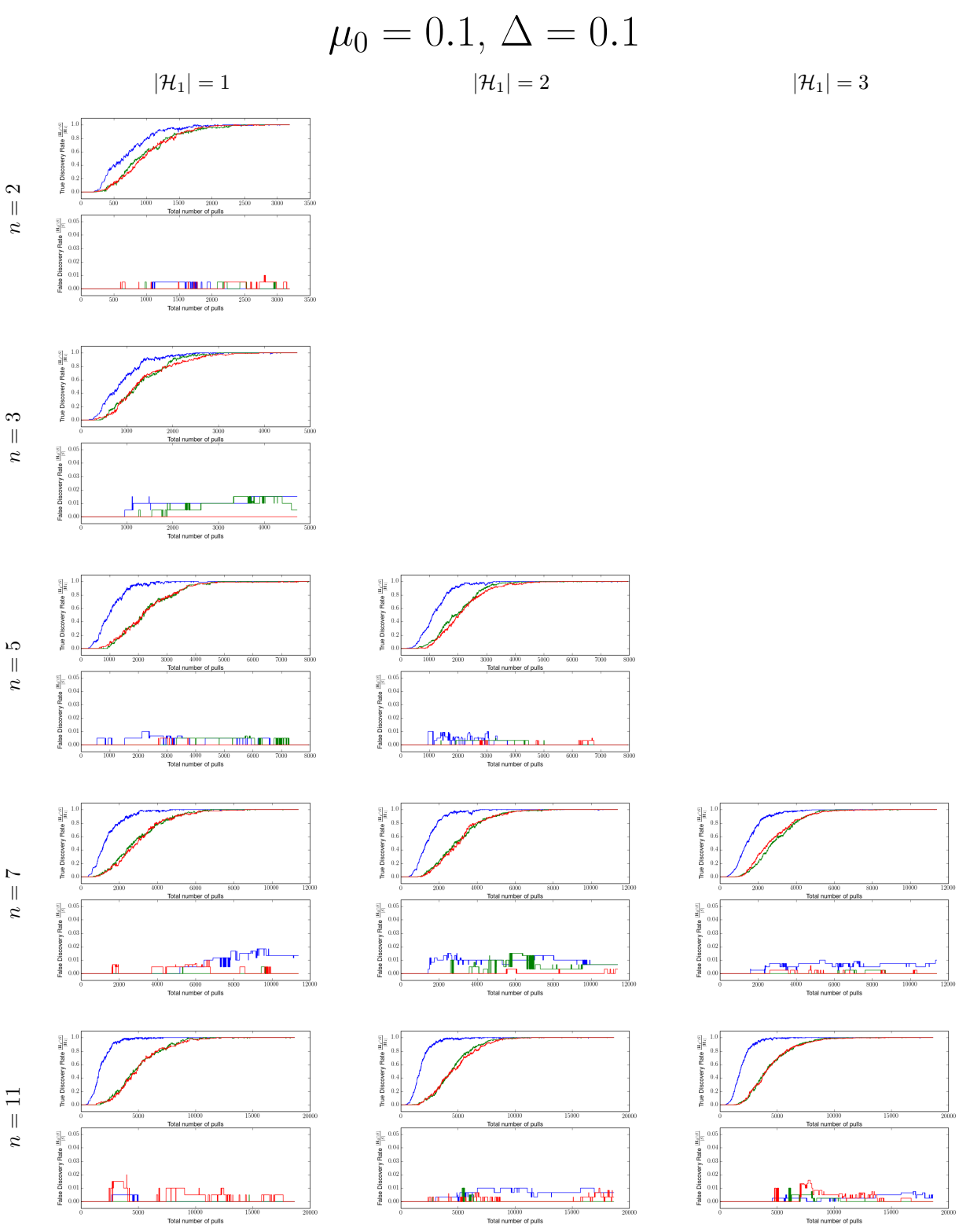

Figure 2: Each plot represents 100 trials of the different allocation methods run with confidence level 0.05. Within each setting of $(n, |\mathcal{H}_1|)$

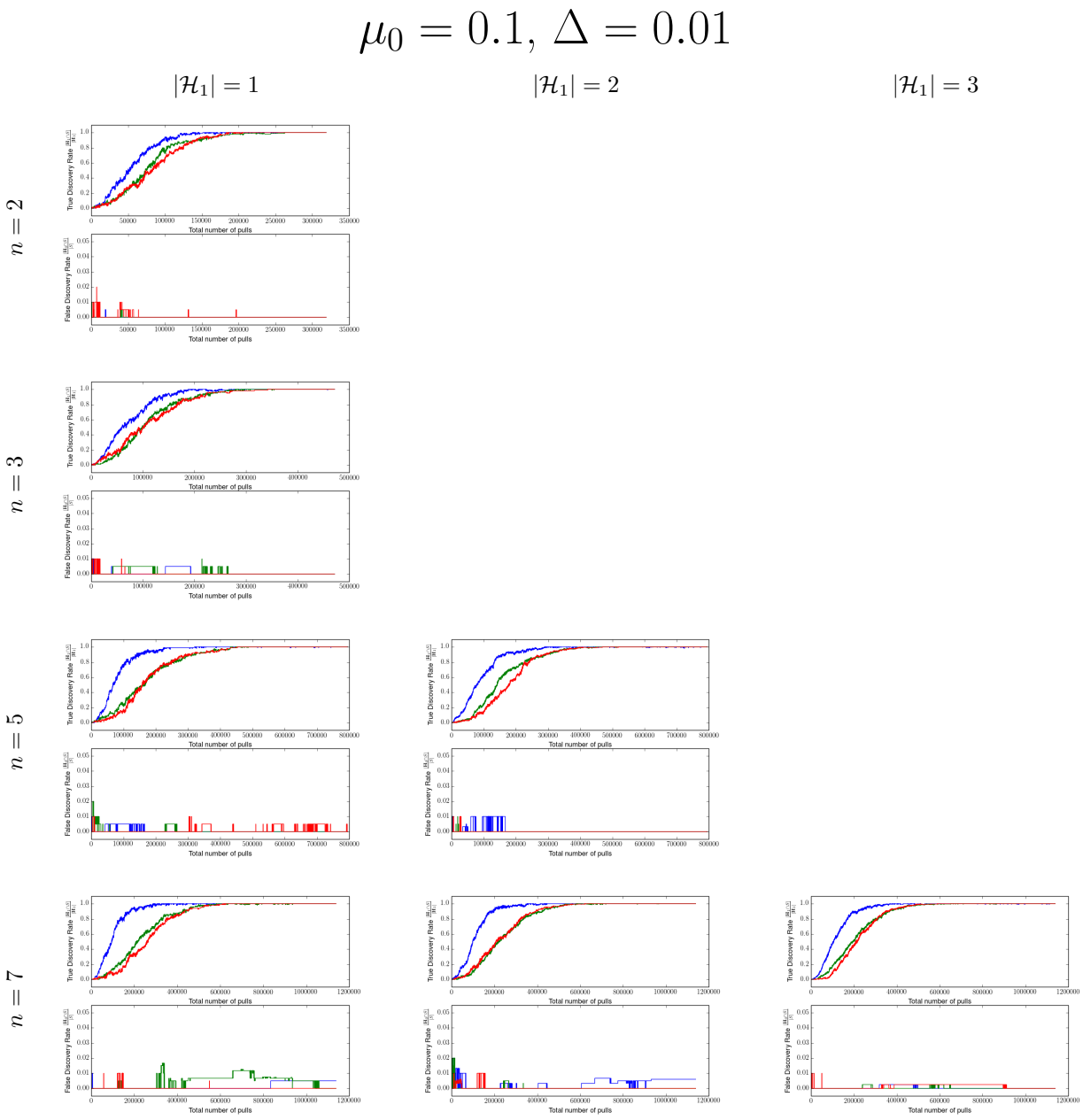

Figure 3: Each plot represents 100 trials of the different allocation methods run with confidence level 0.05. Within each setting of $(n, |\mathcal{H}_1|)$

where $[x]_+ := \max\{0, x\}$. To solve for the optimal q_a, one notes that q_a is non-decreasing in $q_{\widehat{i}}$, so one can pick a value of $q_{\widehat{i}}$, compute the induced values of q_a, and increase or decrease the value of $q_{\widehat{i}}$ based on whether $\sum_{i=1} q_i$ is larger or smaller than 1 (i.e., binary search). Python code (for expository purposes only, not actual implementation) for solving for p_a using Equation 3 and assuming symmetric confidence bounds is given below.

In some cases, the value of B will be unknown, such as in unpredictable web-environments with time-varying traffic. A natural way to deal with this is to consider the allocation as B goes off to infinity. In this case

$$q_a(1-p_0) = \frac{1}{\left(\left(\frac{c-\widehat{\Delta}_a}{\phi_a}\right)^2 - 1\right)\frac{1}{p_0} + \left(\frac{c-\widehat{\Delta}_a}{\phi_a}\right)^2 \frac{1}{q_{\widehat{i}}(1-p_0)}} \quad \text{for } p_0 \text{ given, as } B \to \infty. \tag{4}$$

and, in particular,

$$\frac{q_a}{q_{\widehat{i}}} = \frac{1}{2\left(\frac{c-\widehat{\Delta}_a}{\phi_a}\right)^2 - 1} \quad \text{for } p_0 = (1-p_0)q_{\widehat{i}}, \text{ as } B \to \infty. \tag{5}$$

The same Python code in the listing can be used to solve Equation 4. The allocation for Equation 5 is trivially solved for by setting $q_{\widehat{i}} = 1$, solving for each q_a, and then normalizing so that $\sum_{i=1}^n q_i = 1$.

3.2 Picking p_0

For any value of p_0, the above section prescribes an allocation for all values of B. Suppose we were going to allocate the same probability to every arm $a \neq 0$ so that $q_a = 1/n$. The variance of $\widehat{\Delta}_a$ would be proportional to $\frac{1}{p_0} + \frac{1}{(1-p_0)/n}$ since the accuracy of arm 0 is relevant to the difference. Minimizing this variance with respect to p_0, we find

$$p_0 = \begin{cases} \frac{\sqrt{n}-1}{n-1} & \text{if } n \geq 2 \\ 1/2 & \text{if } n = 1 \end{cases} \tag{6}$$

The derivation of this value of p_0 assumed that the distribution over the treatments was approximately uniform. To see why this is not an unreasonable assumption, note that the UCB strategy of the algorithm more or less attempts to minimize the maximum absolute confidence bound. That means that $c = \widehat{\Delta}_{\widehat{i}} + \phi_{\widehat{i}} \approx \widehat{\Delta}_a + \phi_a$ for all a. Treating this approximation as equality, $\left(\frac{c-\widehat{\Delta}_a}{\phi_a}\right)^2 = 1$ for all $a \in [n]$, and regardless of the value of p_0 or B, we find that $q_a = q_{\widehat{i}}$ in Equation 3. This motivates using p_0 as defined in Equation 6 for all values of B.

3.3 Experiments

The set $\mathcal{S}$ that is output from Algorithm 1 has controlled FDR at all times *regardless of the sampling procedure*. This means it does not matter whether samples are taken one at a time in a very complicated way, uniformly, or from some other arbitrary sampling procedure, the set $\mathcal{S}$ will always have controlled FDR. Said another way, FDR control and the sampling strategy are *decoupled* so we can explore many different strategies for sampling. In what follows we consider:

- `uniform-max-p0`: samples the control arm and treatment arms $\{0, 1, \ldots, n\}$ uniformly, sampling each arm with probability $1/(n+1)$ for all time t. That is, this method never stops sampling an arm. This is equivalent to "standard" A/B/n testing with FDR control.
- `uniform-auto-p0`: performs just as `uniform-max-p0` except that the control arm probability p_0 is defined as in Equation 6 and $p_i = (1-p_0)/n$ for all treatment arms $i = 1, \ldots, n$.
- `succ-elim-auto-p0`: similar to `uniform-auto-p0` except that it sets $p_i = 0$ for all $i \in \mathcal{S}$. Specifically, it sets p_0 in Equation 6 except with $n \leftarrow n - |\mathcal{S}|$ so that $p_0 = \frac{\sqrt{n-|\mathcal{S}|}-1}{n-|\mathcal{S}|-1}$, $p_i = (1-p_0)/(n-|\mathcal{S}|)$ for $i \in [n] - \mathcal{S}$, and $p_i = 0$ for $i \in \mathcal{S}$. That is, it stops samping arms that it has already declared as significant and then behaves as `uniform-auto-p0`.

Listing 1: Procedure for solving for the probability allocations for B samples

```
def __ucb_batch_allocation(self, B, mu_hat, phi, T, p0):
    """
    Returns probability vector prescribing proportion to sample arms

    Input:
        B: int or float('inf'), number of samples to be sampled
        mu_hat: (n+1) array, empirical means of arms (arm 0 is control)
        phi: (n+1) array, confidence bound s.t. |mu_hat-mu| < phi with high probability
        T: (n+1) array, number of times arms pulled
        p0: float, probability that control arm will be sampled in final output

    Output:
        p: (n+1) array, probability vector with p[0]=p0
    """
    n = len(mu_hat)-1
    p = np.zeros(n+1)
    p[0] = p0

    Delta_hat = abs(mu_hat-mu_hat[0])
    Delta_hat[0] = -float('inf')
    i_hat = np.argmax(Delta_hat + phi)
    c = Delta_hat[i_hat] + phi[i_hat]

    a = 0.
    b = 1.
    while (b-a)>EPS_TOL:
        p[i_hat] = (a+b)/2.
        for i in range(1, n):
            if i != i_hat:
                alpha = (c-Delta_hat[i])/phi[i]
                den = (alpha*alpha-1)/(T[0]/B+p[0]) + alpha*alpha/(T[i]/B+p[i_hat])
                p[i] = max(0, 1./den - T[i]/B)
        if sum(p) > 1.:
            b = (a+b)/2.
        else:
            a = (a+b)/2.

    return p/sum(p)
```

- `Bandit-B-auto-p0`: takes B as input, sets $p_0 = \frac{\sqrt{n-|\mathcal{S}|}-1}{n-|\mathcal{S}|-1}$, and samples each arm a with probability $p_a = (1-p_0)q_a$ where q_a and p_0 is defined in Equation 3.
- `Bandit-Binf-auto-p0`: samples each arm a with probability $p_a = (1-p_0)q_a$ where q_a and p_0 is defined in Equation 4.
- `Bandit-Binf-max-p0`: samples each arm a with probability $p_a = (1-p_0)q_a$ where q_a and p_0 is defined in Equation 5.
- `Bandit-auto-p0`: samples control arm 0 with probability $p_0 = \frac{\sqrt{n-|\mathcal{S}|}-1}{n-|\mathcal{S}|-1}$ and the arm prescribed by Algorithm 1 (i.e. $\widehat{i} = \arg\max_i \widehat{\mu}_i + \phi_i$) with probability $1-p_0$.

Something to keep in mind is that as $B \to \infty$ we have `Bandit-B-auto-p0` $\to$ `Bandit-Binf-auto-p0`. And as $B \to 0$ we have `Bandit-B-auto-p0` $\to$ `Bandit-auto-p0`. Thus, if one uses `Bandit-B-auto-p0` even with an incorrectly specified value of B, its performance should not be too far from `Bandit-B-auto-p0` or `Bandit-Binf-auto-p0`.

Figure 4 shows the first way adaptivity can benefit over non-adaptive methods, when the means in $\mathcal{H}_1$ are all very diverse, as described in Section 1.1. Here the control arm has probability μ_0 and each treatment arm has mean $\mu_i = \mu_0 + \Delta 10^{-(i-1)/(n-1)}$ so that $\mu_i \in [\Delta, \Delta/10]$ for all $i \in \mathcal{H}_1$. We observe that `succ-elim-auto-p0` and the `Bandit` methods behave well, with the exception of `Bandit-auto-p0`. All the methods that are behaving well here are optimal up to log factors that account for the subtle statistical effects that we study next.

Figures 5, 6 study the second way adaptivity can benefit over non-adaptive methods, when the means in $\mathcal{H}_1$ are all roughly equal. Here the control arm has probability μ_0 and the treatment size $\mu_i - \mu_0 = \Delta$ for $i \in \mathcal{H}_1$ (by definition, $\mu_i = \mu_0$ for $i \in \mathcal{H}_0$). The different panels of the figures study the effect of the size of $|\mathcal{H}_1|$, Δ, and n. We see that the non-adaptive methods are outperformed by the `Bandit` methods which are all comparable with the exception of the performance of `Bandit-auto-p0` which wanes as B gets very large (which is expected since it is pulling just one treatment arm per batch). We also note that in these situations there is almost no difference between `succ-elim-auto-p0` and the non-adaptive methods.

We conclude that the `Bandit-B-auto-p0` and `Bandit-Binf-auto-p0` methods are the overall best-performers. Conveniently, these are the same algorithm, just with different values of input B (finite if known, infinite if not).

$$\mu_i = \mu_0 + \Delta 10^{-(i-1)/(n-1)},\ \mu_0 = 0.01,\ \Delta = 0.01$$

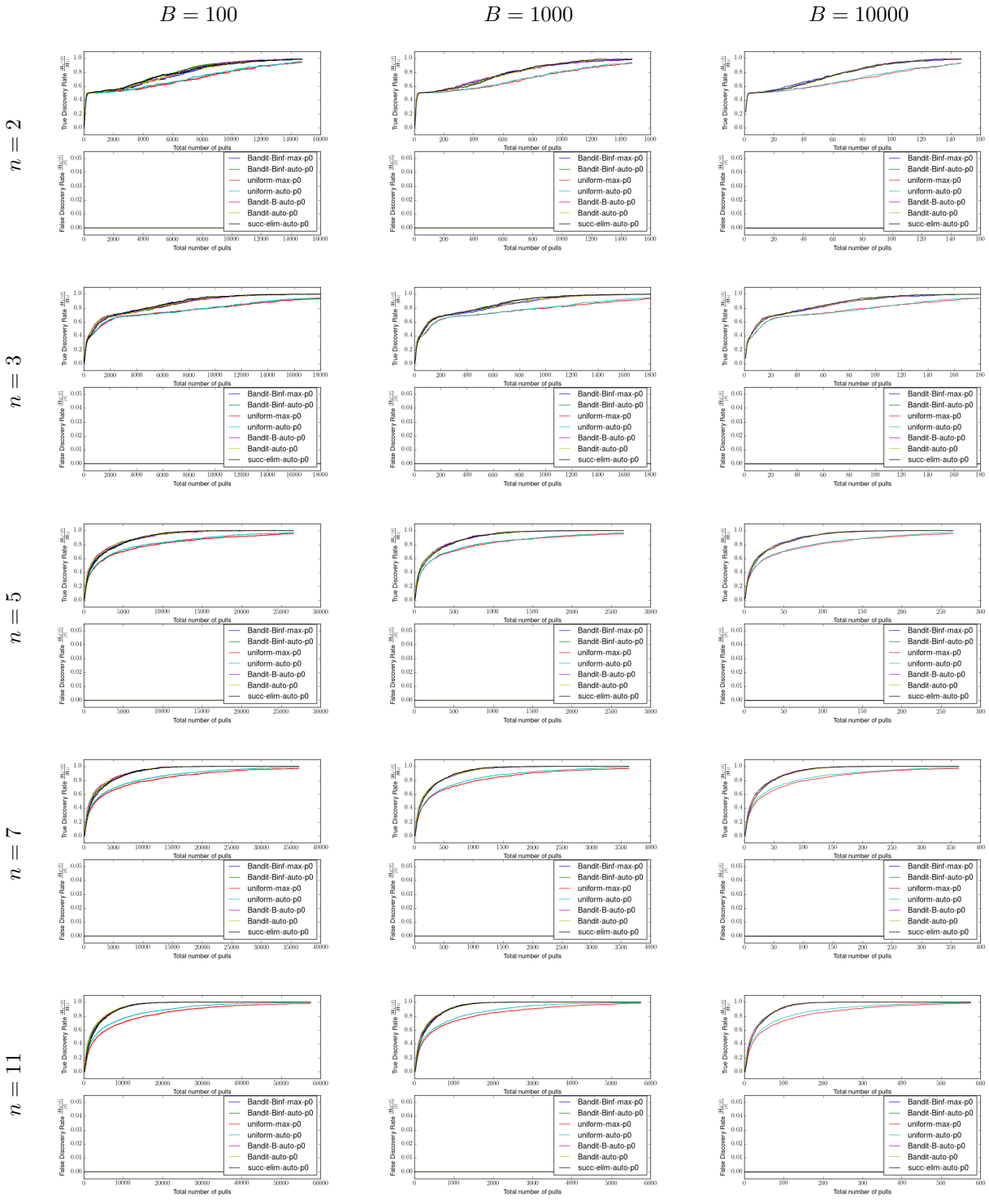

Figure 4: Diverse gap experiment from Section 3. A "pull" here is a single allocation of B samples.

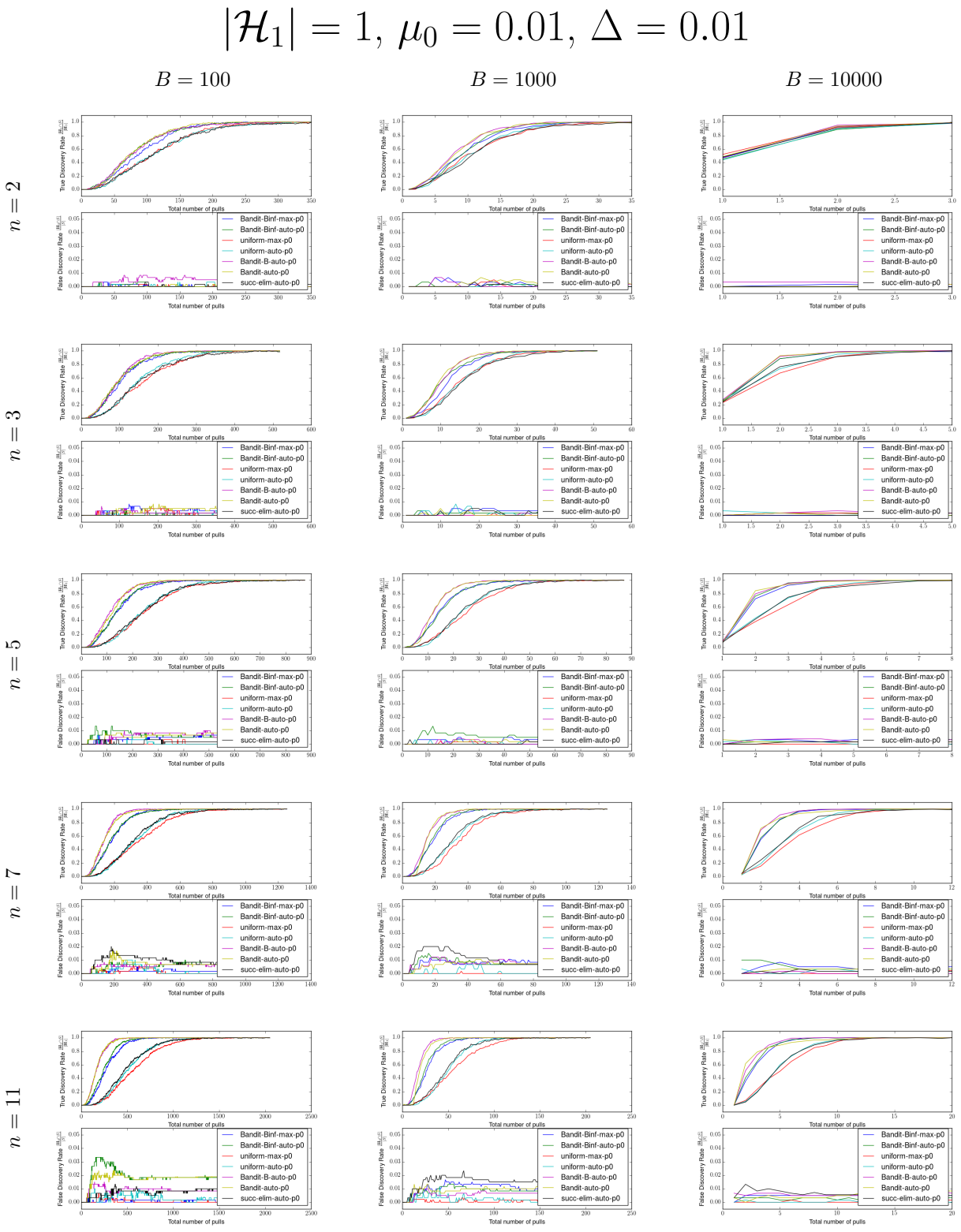

Figure 5: Fixed gap experiment from Section 3. A "pull" here is a single allocation of B samples.

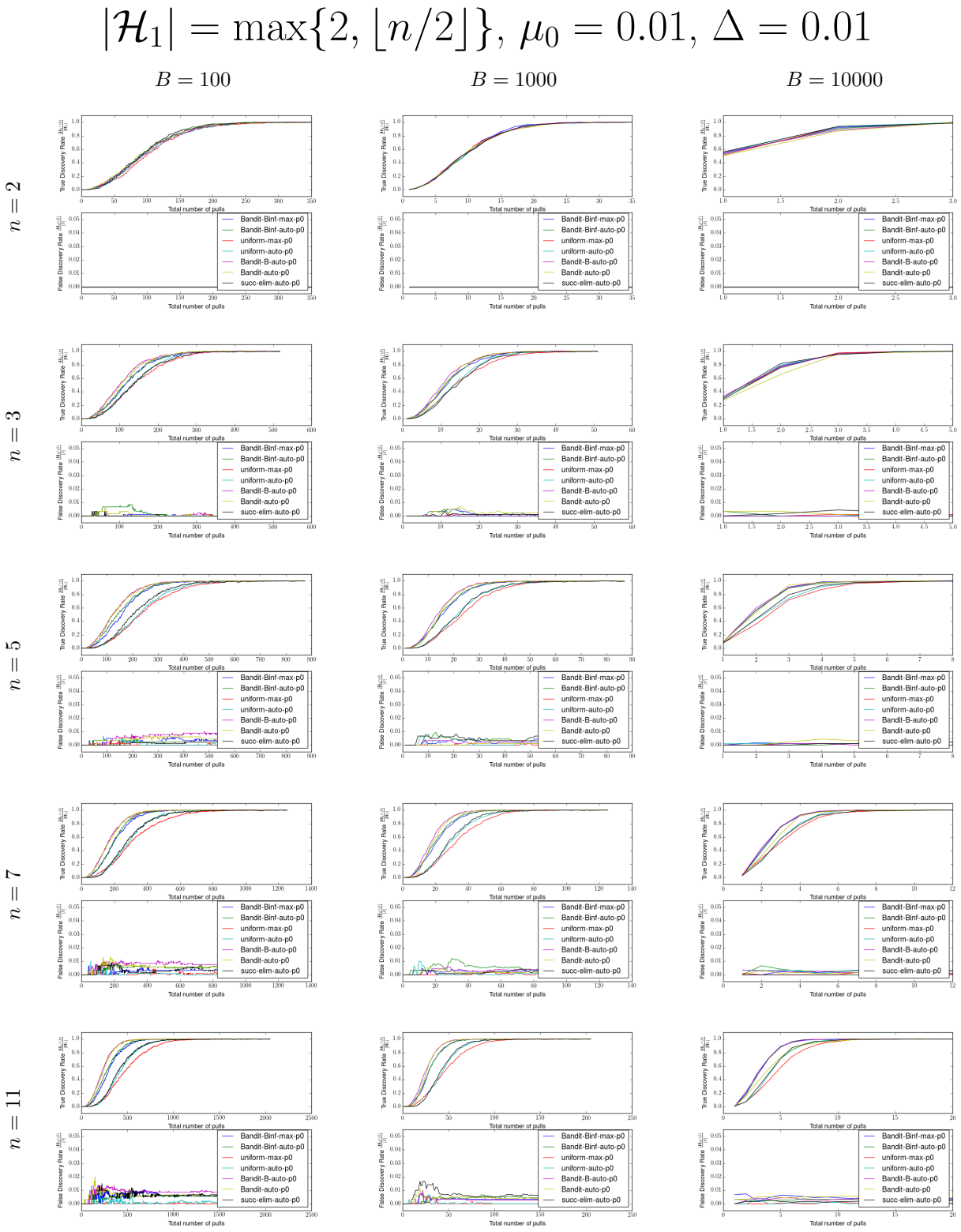

Figure 6: Fixed gap experiment from Section 3. A "pull" here is a single allocation of B samples.

References

[1] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

[2] Max Simchowitz, Kevin Jamieson, and Benjamin Recht. The simulator: Understanding adaptive sampling in the moderate-confidence regime. *Conference on Learning Theory (COLT)*, 2017.

[3] Alexandra Carpentier, Alessandro Lazaric, Mohammad Ghavamzadeh, Rémi Munos, and Peter Auer. Upper-confidence-bound algorithms for active learning in multi-armed bandits. In *International Conference on Algorithmic Learning Theory*, pages 189–203. Springer, 2011.

[4] Kevin Jamieson, Matthew Malloy, Robert Nowak, and Sébastien Bubeck. lilucb: An optimal exploration algorithm for multi-armed bandits. In *Conference on Learning Theory*, pages 423–439, 2014.

[5] Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the royal statistical society. Series B (Methodological)*, pages 289–300, 1995.

[6] Yoav Benjamini and Daniel Yekutieli. The control of the false discovery rate in multiple testing under dependency. *Annals of statistics*, pages 1165–1188, 2001.

[7] Kevin Jamieson. On the sample complexity of null rejection under adaptive sequential experimental design and false discovery rate control. *Unpublished*, 2017.

[8] Kirthevasan Kandasamy, Akshay Krishnamurthy, Jeff Schneider, and Barnabas Poczos. Asynchronous parallel bayesian optimisation via thompson sampling. *arXiv preprint arXiv:1705.09236*, 2017.

[9] Thomas Desautels, Andreas Krause, and Joel W Burdick. Parallelizing exploration-exploitation tradeoffs in gaussian process bandit optimization. *The Journal of Machine Learning Research*, 15(1):3873–3923, 2014.