Trading Off Scientific Knowledge and User Learning with Multi-Armed Bandits

Yun-En Liu[1], Travis Mandel[1], Emma Brunskill[2], and Zoran Popović[1]
[1]Center for Game Science, Computer Science & Engineering, University of Washington
[2]School of Computer Science, Carnegie Mellon University
{yunliu, tmandel, zoran}@cs.washington.edu, ebrun@cs.cmu.edu

ABSTRACT
The rise of online educational software brings with it the ability to run experiments on users quickly and at low cost. However, education is a dual-objective domain: not only do we want to discover general educational principles, we also want to teach students as much as possible. In this paper, we propose an automatic method for allocating experimental samples, based on multi-armed bandit algorithms, that balances between learning each experimental condition's effectiveness and users' test performances. Our algorithm, UCB-Explore, allows the experimenter to explicitly specify the tradeoff these two objectives. We assess the performance of our algorithm in a simulated experiment with parameters drawn from a real-world data. In this simulation, our algorithm is better able to navigate this trade off compared to other multi-armed bandit algorithms such as UCB1 and ϵ-greedy. As an example application, we show how a researcher could use the generated samples to identify strong and weak interaction effects, and confirm these findings on a separately-collected dataset.

Keywords
Multi-armed bandits, automatic experimentation, scientific discovery

1. INTRODUCTION
The rise of online educational software has greatly increased the amount of data available to researchers in the educational data mining community. Indeed, both the games and e-commerce industries have already been transformed by the introduction of A/B testing to understand how users react to different software designs and incentives. These tests are traditionally done in a staged manner: run a test, then choose the best option to deploy to all users. But why separate the test phase and the optimized software? Just as systems like Assistments promise to both educate and assess simultaneously [20], we should both educate and experiment at the same time. With programmatic control of educational material and automatic data collection, such systems could provide ever more effective educational experiences while also generating scientific knowledge as they collect data about the comparative effectiveness of different representations of knowledge or teaching strategies. This knowledge could then be used to inform educational theories, potentially producing new and better options to be investigated by the system.

However, this scientific objective is complicated by the fact that in many educational contexts we are working in a high-stakes domain: users of educational software should learn from the system, so testing ineffectual conditions can cause real harm. We want an approach that allows the experimenter to explicitly specify the relative worth of teaching players (by giving out the best conditions) against the gain of scientific knowledge (by giving out sub-optimal conditions to better assess their worth). Then the algorithm can sample the different experimental conditions, with a bias in favor of finding and exploring the better ones depending on the specified weighting. As the algorithm obtains better estimates of the effects of different conditions, it should eventually converge to placing all students in the most effective condition. This problem fits nicely into a multi-armed bandit formulation, where the arms are conditions and the reward is user learning, so we will attack it from this angle.

Our contributions are as follows. First, we formulate a dual-objective bandit in which we want to optimize a weighted combination of the 95% confidence interval sizes around the condition means, and user test performance. We then introduce UCB-Explore, a modification of an existing multi-armed bandit algorithm that tries to directly optimize for this user-specified tradeoff. Second, we analyze the performance of our algorithm in a 64-condition simulation with parameters learned from a real-world experiment involving different ways of displaying number lines. UCB-Explore better optimizes the weighted objective compared to existing bandit algorithms, and in our simulation appears fairly robust to changes in its parameters. Finally, we show how to use the samples generated from our algorithm to identify likely and unlikely two-way interactions between factors, and validate our hypotheses on a separate dataset.

2. RELATED WORK
2.1 Scientific Discovery, Massive Experiments
Researchers in AI have been working for years to develop systems capable of automatically generating useful scientific knowledge. This field, known as scientific discovery, has generated many such systems [14]. For example, Lee et al. used a feedback loop between the RL rule induction program and

expert knowledge to identify potentially carcinogenic compounds [15]. Perhaps the most comprehensive example of such a system is Robot Scientist Adam, a fully automated robot capable of the full loop of hypothesis generation, experimental design, and data analysis in yeast genomics [10]. These systems are quite general, and try to automate many aspects of scientific behaviors; we are primarily concerned with the automation of certain kinds of experimentation in high-stakes domains.

We can also be considered related to massive (educational) testing, made possible in recent years through web-based software. As noted by Stamper et al., online experiments can be considered a new source of data with fundamentally different properties than ones conducted at the lab or school district level [24]. Experiments run online can have many more users, and better measure true task engagement; however, it may be difficult or impossible to collect rich data such as interview data. As examples, Lomas et al. conducted two large-scale experiments with many conditions on the effect of challenge on motivation and learning in an educational game [18]. Lindsey et al. used Gaussian process regression and an active learning framework to reduce the number of samples required to learn functions of user responses [16]. And in previous work, we proposed a greedy, hierarchical algorithm that ran sequences of multivariate tests in multi-factor settings [17]. In this work, however, we want an algorithm that adapts incoming samples to maximize an experimenter-specified weighted sum user learning and confidence in the estimated condition means.

2.2 Adaptive Trials

Clinical trials are another example of high-stakes domains where it may be undesirable or unethical to assign patients to certain experimental conditions. Over the years, researchers have developed different methods for minimizing patient harm while trying to identify the best treatments. Some examples include play-the-winner, drop-the-losers, sample size re-estimation, adaptive treatment-switching, and so on; for a review, see [8] or [4]. The adaptive randomization designs are closest in spirit to our work: they bias the randomization in favor of successful conditions and away from failed ones [29] [26].

These strategies are often heuristic and offer no guarantees. However, clinical trials can also be formulated as multi-armed bandit problems, for which algorithms with theoretical performance guarantees are known. The closest work in this space is perhaps by Kuleshov et al., who propose the use of existing multi-armed bandit algorithms for the allocation of users to experimental conditions and show simulations suggesting that more patients are successfully treated [12]. Similar empirical investigations of bandits have been undertaken in the domain of web content retrieval by Vermorel et al. [25]. They focus on the standard bandit formulation in which the only objective is to maximize reward; in this paper, we also care about scientific knowledge.

3. MULTI-ARMED BANDITS

Throughout this paper, we will be using data drawn from an experiment carried out in the educational game Treefrog Treasure, seen in Figure 1. We give out different types of number lines to players and study how accurately they answer a randomized test number line; the full experimental design is described in Section 5.1. Importantly, we have two competing objectives: teach as much as possible to players in the game (by only giving out the best number lines), but identify the effectiveness of different fraction representations, hinting systems, or other number line properties (by testing different kinds of number lines). The relative weight of these two objectives will vary by application, so we will later allow the experimenter to set the weight explicitly.



Figure 1: A screenshot of Treefrog Treasure, our source of users. Players navigate through a physics-based world, solving number line problems along the way. Notice that the number line has full tick marks, pie chart labels on the line, and a symbolic (ex. $\frac{a}{b}$) target representation. In our experiment, these are a few of the parameters we allow our system to automatically explore to determine which types of number lines lead to maximal near-transfer.

If our goal were solely to maximize player learning, then this problem reduces to a multi-armed bandit (MAB). MAB problems consist of K probability distributions, $D_1, \ldots, D_K$, with expected values $\mu_1, \ldots, \mu_K$. The D_i and μ_i are not known at the start. These distributions are classically viewed as wins or losses (1 or 0) from various arms of a slot machine, but in continuous formulations can be any (bounded) user-defined function. In our scenario, we can think of the arms as different types of numberlines to be given as practice, and the reward as player success on a randomized test number line.

In a bandit problem, the experimenter tries to pull arms in order to collect as much reward as possible (e.g. assign players to conditions to maximize test performance). At each turn, $t = 1, 2, ...$, we select an arm $j(t)$ and receives some reward from that arm's reward distribution $r(t) \sim D_{j(t)}$. If the D_i were known, the optimal strategy would be to choose the arm with the highest expected reward, $j(t) = \arg\max_i \mu_i$: that is, pick the most effective number line and give it to every player. Unfortunately, the D_i and μ_i are hidden. Successful bandit algorithms must navigate an *exploration-exploitation* tradeoff to discover information about the D_i while also generating high reward.

In general, we will not be able to give everyone the best intervention. Define the *total expected regret* at some fixed timestep T as the loss of reward from playing non-optimally,

$R^T = T \max_i \mu_i - \sum_{t=1}^{T} \mu_{j(t)}$. Lai and Robbins show that regret must grow at least logarithmically in time [13], developing a lower bound of $R^T = \Omega(log(T))$.

Not all strategies that work well in practice meet this bound. One such heuristic strategy is ϵ-greedy, which for any $0 < \epsilon < 1$ plays a random arm with ϵ probability and otherwise plays the arm with the highest empirical mean. This strategy has linear regret because it has a constant chance to play suboptimal arms. One could consider allowing ϵ to decrease over time to eliminate this linear regret term; however, this adds another parameter and does not always help in practice [25].

A different, popular class of theoretically-motivated strategies which meet the logarithmic regret bound are the Upper Confidence Bound strategies (UCB) [3]. These algorithms exemplify the principle of *optimism under uncertainty* by pulling the arm which has the highest estimated upper confidence bound on its mean. The simplest, UCB1, works in the following way. Assume that all rewards are in the range $[0, 1]$. To initialize, pull each arm once. Then at each subsequent timestep, if the number of times an arm i has been pulled is n_i, choose the arm $j(t) = \arg\max_i \hat{\mu_i}^t + c\sqrt{\frac{2 \ln t}{n_i^t}}$. In this formula, the exploitative first term is the estimate of the arm mean, while the exploratory second term represents an uncertainty that grows slowly as other arms are pulled but decreases sharply when this arm is pulled. UCB1 provably incurs logarithmic regret when $c = 1.0$, though c is often set to be smaller for better empirical performance.

In this paper, we will focus on ϵ-greedy and UCB; for other algorithms, see [6]. Note,however, that all of these algorithms are focused on maximizing reward: bandit strategies will try to only allocate enough samples to sub-optimal arms to tell that they are indeed sub-optimal. Unfortunately, this may leave uncertainty about the exact values of each arm, which was our second goal. Furthermore, algorithms such as ϵ-greedy and UCB can be quite sensitive to the settings of their parameters.

On the opposite extreme, researchers have also studied the case where the only goal is to learn something about the arm distributions D_i, such as their means, the μ_i. Antos et al. introduce an algorithm for this problem, GAFS-MAX, which attempts to minimize squared error of the worst estimated μ_i by sampling the "most under-sampled" arm or the arm with greatest empirical variance [2].

Our definition of "scientific knowledge" is similar: we wish to minimize the sizes of our estimates of the 95% confidence intervals around the arm means. To the best of our knowledge, algorithms exist only for the extreme cases where we maximize only for reward or only for estimation of arm properties. In this paper, we propose allowing the experimenter to set a tradeoff between these goals, and introduce an algorithm which smoothly interpolates between these extremes by maximizing for the specified tradeoff.

4. UCB-EXPLORE

Our algorithm, called UCB-Explore, is a variant of UCB1. As noted earlier, changing the scaling factor c on UCB1's confidence bounds often leads to improved performance in practice: thus, the key idea behind UCB-Explore is to self-adjust c in response to mistakes. Our algorithm takes as input a set of arms with unknown reward distributions D_i, a function CI to calculate confidence interval sizes, a weight w controlling the tradeoff between reward and confidence interval sizes around the arm means, and a multiplier m that controls how quickly c changes. Good choices of CI in general depend on the shape of the reward distributions D_i, though methods such as the centered percentile bootstrap [23] allow reasonable estimates in most situations. In our example, the arms are Bernoulli processes generating "success" or "failure" depending on whether the student answers a test question correctly, so a reasonable choice for CI is the Wilson score interval [27].

Let N be the number of arms, r^t be the reward received on pull t, and Δ_j^t be the size of the 95% confidence interval of arm j on round t. For any timestep T, our goal is to maximize the expression $w \sum_{t=1}^{T} r^t - (1 - w) \sum_{j=1}^{N} \Delta_j^t$. That is, we want to maximize the total reward received, but minimize the sizes of the 95% confidence intervals sizes, with some weight w between both goals. This type of goal makes sense if the experimenter is able to assign "worth" to confidence interval sizes, in terms of reward. For example, an educational institution might be given funding based on students' standardized test scores, and be willing to pay a certain amount of money for smaller confidence intervals about certain educational interventions. As an alternative interpretation, note that reward grows without bound while the confidence interval sizes cannot go below 0, so w can be thought of as a rough "switching point" after which the algorithm will aim primarily to gather more reward. Say that the experimenter knows a reasonable reward is 0.6, would roughly like the reward term to dominate after n pulls, and calculates that after this many pulls the confidence intervals can be expected to decrease by about 0.3 per pull. Then setting $w = 0.33$ causes the reward term to overtake the confidence intervals after about n pulls.

Note that this objective is difficult to optimize directly. This can be seen by considering the case $w = 1.0$, where we are focused on reward: it is computationally intractable to optimally pull arms to optimize the objective [19]. It must therefore be similarly intractable to optimize in the general case. As such, we propose UCB-Explore as a heuristic algorithm which lacks guarantees but seems to work well in our scenario. Our algorithm is shown in Algorithm 1. Let $c^1 = 1.0$ be our initial scaling factor, as in UCB1. We first pull each arm once; at subsequent times t, we choose the arm $j(t) = \arg\max_i \hat{\mu_i}^t + c^t\sqrt{\frac{2 \ln t}{n_i^t}}$. When c^t is very large, $\hat{\mu_i}^t$ has little effect, and we will tend to choose arms that have fewer pulls (more exploratory). When c^t is very small, $\hat{\mu_i}^t$ dominates, so that we tend to pull only the arms with highest empirical means (more exploitative). So far, then, our algorithm is simply a tuned variant of UCB1.

The key change in our algorithm is that we increase or decrease c if we choose the wrong arm to pull. Say that arm i has been pulled at times $t_1, t_2, \ldots$. Then the rewards of all pulls of arm i up until this time are $R_i^t = r_i^{t_1}, r_i^{t_2}, \ldots$. Let $s_i^t = w\hat{\mu_i}^t + (1 - w)(CI(R_i^t) - \mathbb{E}_{r_i^t}[CI(R_i^t + r_i^t)])$: that is,

Algorithm 1 UCB-Explore

Require: a tradeoff w, multiplier m, bandit arms $A_{1,\ldots,N}$
  $c = 1.0$
  for $j = 1$ to N **do**
    $r_j = \text{PULL}(A_j)$
    $\mu_j = r_j$
    $n_j = 1$
    $R_j = \{r_j\}$
  end for
  for $t = N + 1$ to ∞ **do**
    for $k = 1$ to N **do**
      $b_k = \mu_k + c\sqrt{\frac{2 \ln t}{n_k}}$
      $c_j = \text{CALCULATECI}(R_j)$
      $e_j = \text{CALCULATEEXPECTEDNEXTCI}(R_j)$
      $s_k = w\mu_i + (1 - w)(c_j - e_j)$
    end for
    $u = \arg\max_i b_i$
    $v = \arg\max_{i \neq u} b_i$
    if $s_v > s_u$ **then**
      if $\mu_u > \mu_v$ **then**
        $c = mc$
      else
        $c = \frac{c}{m}$
      end if
    end if
    $r_j = \text{PULL}(A_u)$
    $\mu_j = n_j\mu_j + r_j$
    $n_j = n_j + 1$
    $R_j = R_j \cup r_j$
  end for

s_i^t is the weighted combination of the expected reward and the expected decrease in confidence interval size if we pull arm i. The calculation of $\mathbb{E}_{r_i^t}[CI(R_i^t + r_i^t)]$ in full generality requires a posterior estimate of D_i; since we are working with Bernoulli trials, we can estimate $p(r_i^t = 1) = \hat{\mu_i}^t$ and calculate the expectation directly.

When should we adjust c? In UCB-EXPLORE, we ask whether we should have picked the arm with the second-highest upper confidence bound. Let $b_i^t = \hat{\mu_i}^t + c\sqrt{\frac{2 \ln n}{n_i^t}}$ for each arm i. Without loss of generality, assume that $b_1^t >= b_2^t >= b_i^t$, $i = 3, \ldots, K$. If $s_2^t > s_1^t$, then our algorithm has made an error: it could have (greedily) obtained a better tradeoff respecting the researcher's decision of w by pulling the second best arm. If $\hat{\mu_1}^t > \hat{\mu_2}^t$, then the algorithm was exploiting too much, so we set $c^{t+1} = mc^t$. If $\hat{\mu_2}^t \leq \hat{\mu_1}^t$, then the algorithm was exploring too much, so we set $c^{t+1} = c^t/m$. We then pull the arm $j(t)$ and continue. It is important to note that this algorithm is heuristic in nature, but seems to work well in our simulation. It may be possible to develop a more theoretically-motivated algorithm to maximize for this weighted goal, which we leave to future work. In either case, if the algorithm respects w in its behavior and seems relatively robust to the choice of m, then we will have achieved our goal. We will see in Section 6 that this is the case in our scenario.

5. EXAMPLE APPLICATION

We will examine the performance of our algorithm with a 64-arm simulation whose parameters are drawn from real-world data. This will demonstrate the feasibility of our approach in a real-world situation. In this simulation, we will try to identify how the appearance of a "practice" number line affects player performance on a randomized "test" number line, a particularly challenging problem since we expect the effect sizes to be small given that the intervention is one number line long. We choose number lines as they are a well-studied and commonly-used pedagogical tool, and a fair amount of evidence suggests that much whole and rational number knowledge is organized around mental number lines [1], [22]. We will first describe the game from which we collected our data, as well as the factors that vary between number lines.

5.1 Treefrog Treasure

Treefrog Treasure is a platformer game that involves jumping through a jungle world and solving number line problems to reach an end goal. Number line problems serve as barriers that the player must solve by hitting the correct target location, as shown in Figure 1. It has been played by over 10 million players worldwide on various websites; data for this experiment is drawn from BrainPOP [5], an educational website aimed at school-aged children. Our dataset consists of 34,197 players, who played from June 3, 2013 to June 20, 2013.

We consider each player as a sequence of many pairs of number lines, and treat each pair as an experimental unit. This gives us 361,738 pairs. This potentially violates independence assumptions in classical statistical tests, but greatly increases the amount of available data we can use to estimate our arm reward distributions. We will strictly adhere to the correct assumptions when we attempt to generate hypotheses and validate them on a new dataset, later.

The appearance of the first number line in each pair constitutes the experimental condition - the full set of factors is specified in Table 1, with illustrations in Figure 2. We care primarily about Ticks, Animations, Backoff Hints, Target Representation, and Label Representation; the Fraction and Initial Labels are randomly chosen, so that our results are meant to generalize for different settings of these factors. This gives us 64 separate conditions, one for each combination of factor settings.

There are additional complexities in the sampling distributions in this dataset that are not relevant to this work; for a more thorough explanation, refer to [17]. The important point is that we can obtain an unbiased estimate, for each experimental condition, of the probability that a player receiving a number line with those parameters will reach and solve the randomized next "test" number line correctly on the first try. For our simulation we will assign each arm the associated mean estimated from our data, and draw simulated samples from the arms by flipping coins with the specified probability of success. These probabilities range from 0.38 to 0.47, with the vast majority falling in $[0.41, 0.45]$. Since the arms are Bernoulli in nature and the probabilities are close to 0.5, the variance is nearly the maximum possible for distributions in $[0, 1]$.

Parameter	Settings	Interpretation
Fraction	Any $\frac{a}{b} \in (0,1), b \leq 9$	The target fraction the player must hit
Initial Labels	[0,1]	For target $\frac{a}{b}$, the proportion of labels of $\frac{n}{b}$ fractions shown at the start.
Target Representation	Symbolic, Pie	How the target fraction is displayed.
Label Representation	Symbolic, Pie	How fraction labels on the number line are displayed.
Ticks	Present, Absent	For target $\frac{a}{b}$, we can display tick marks for each fraction $\frac{n}{b}$.
Animations	Present, Absent	If the player misses a target $\frac{a}{b}$, they might receive an lengthy pie chart animation showing how to divide up the number line into b parts.
Backoff Hints	1, 2, 3, 4	The number of misses for target $\frac{a}{b}$ before the progressive hinting system fills in all labels for $\frac{n}{b}$ and displays the correct answer.

Table 1: The parameters controlling number lines in our experiment. Bolded parameters are factors we are interested in studying; non-bolded parameters are selected randomly.

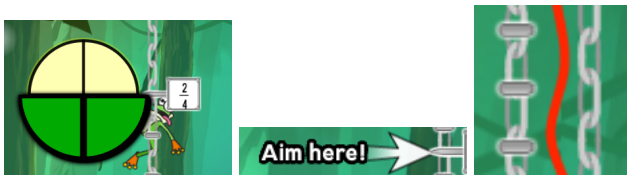


Figure 2: The animation condition on the left shows the player how to divide up the number line. The backoff condition in the middle gradually more direction about where to hit. The ticks condition either divides up the number line into segments when ticks are present, or leaves it empty besides the 0 and 1 labels when ticks are absent.

6. SIMULATION

Empirical MAB research such as [12] and [25] indicates that MAB algorithm performance is very sensitive to the exact values of parameters, and that tuned simple algorithms, such as ϵ-greedy, outperform theoretically-motivated algorithms such as UCB1. One reason that tuning affects reward is that different settings of these parameters can result in a tradeoff between identifying the best mean and exploiting the current best. Although these parameters do not explicitly optimize the tradeoff between confidence interval size and reward, it is often the case that more exploratory parameter settings will do a better job of minimizing confidence interval size. But it is not immediately obvious how to set these parameters for any particular tradeoff - what ϵ should we choose if we want to weight confidence interval size and reward equally? In contrast, UCB-Explore allows us to explicitly set this trade off and optimize for it more directly. We will compare how UCB-Explore trades off reward and scientific knowledge compared to MAB algorithms ϵ-greedy, UCB1, and UCB1-Tuned, ignoring the fact that the UCB-Explore parameter is given by the objective while the other parameters may be more difficult to choose.

ϵ-greedy is a simple and straightforward method for balancing the dual goal of learning about the arm means and also maximizing reward. It has the additional advantage of ϵ being easy to interpret: the proportion of players who will be devoted to exploring the non-optimal arms. UCB1 is also capable of trading off between learning and reward by scaling the bounds: making them very large causes the algorithm to prefer exploration, while making them small causes the algorithm to focus on the highest empirical mean and prefer exploitation. UCB1-Tuned replaces the loose bounds of UCB1 with ones that depend on empirical variance of the arms, which usually works better in practice [3].

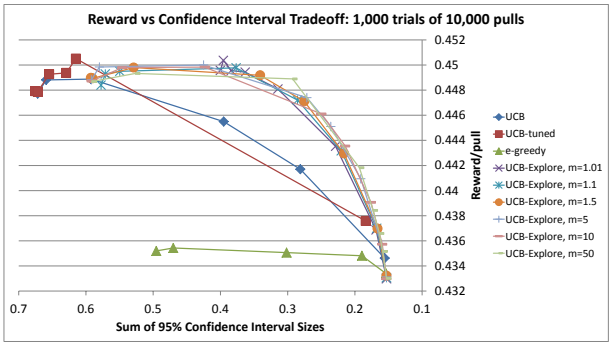


Figure 3: Reward vs confidence interval sizes. Up and to the right is ideal; we see that UCB-Explore is typically better at generating high reward and learning the various arm means than other algorithms. ϵ-greedy performs poorly overall.

Parameter	Values (right to left)
ϵ-greedy, ϵ	0.3, 0.03, 0.01, 0.001, 0.0001
UCB1, c	1.0, 0.2, 0.15, 0.1, 0.03, 0.01
UCB1-Tuned, c	1.0, 0.2, 0.15, 0.1, 0.03, 0.01
UCB1-Explore, w	0, 0.01, 0.03, 0.06, 0.1, 0.3, 0.6, 1

Table 2: The parameter settings generating the tradeoff graphs in Figure 3. All UCB1-Explore variants in the graph, which have different values of the scaling multiplier m, are generated from the same group of w.

We generate a tradeoff curve between average reward per pull and the sum of the sizes of the 95% confidence intervals around each estimated arm mean, shown in Figure 3. Each point for each algorithm is the average reward and interval size for 1000 trials of 10,000 pulls, for the parameters shown in Table 2. Points that are up and to the right are better. We see both that UCB-Explore tends to have superior performance, especially when one does not care entirely about reward, and also that ϵ-greedy appears much worse than both strategies once we scale the bounds calculated by UCB1. In addition, the different UCB-Explore curves are generated by different values of m - in our problem, it appears that the choice of m has little impact within a wide range, with perhaps the exception of $m = 1.01$.

To gain some intuition about how UCB-Explore behaves and how one should choose w, it is useful to examine the behavior of the scaling factor c that controls the size of the bounds. This is shown in Figure 4. As $w \to 0$, the algorithm

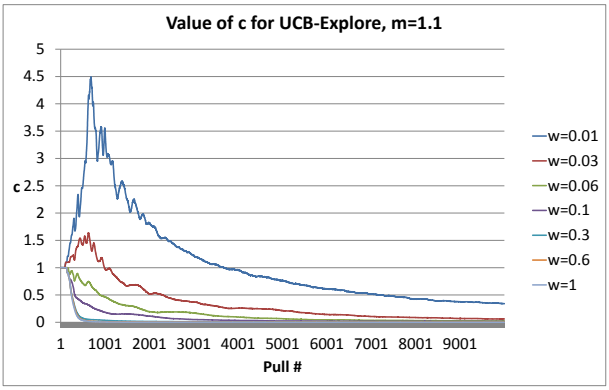


Figure 4: The value of the scale factor on the confidence bounds in UCB-Explore as the algorithm pulls more arms. When $w = 0$, c increases quickly and does not stop, reflecting the fact that the algorithm cares only about exploration.

cares less and less about generating reward, so we expect it to explore more: as expected, c increases. Furthermore, the gains from reward are constant over time, but the gains from reducing estimated confidence interval sizes shrink as we continue to sample. Thus as time passes we expect UCB-Explore to focus more and more on reward as long as $w > 0$, and indeed this is the behavior we observe by the shrinking values of c. In the case of $w = 1.0$, c actually shrinks so fast that the algorithm exploits too quickly, explaining the small dips in performance of both UCB-Explore and UCB1 when tuned to be very exploitative.

7. VALIDATION

7.1 Interaction testing

Our algorithm is a method of allocating samples that respects the tradeoff between encouraging player learning and getting more accurate estimates of experimental condition means. Had this experiment been run online, a researcher could directly analyze the data gathered. However, we have been running our algorithm in a simulated environment with parameters drawn from real-world data, leaving us vulnerable to overfitting. To demonstrate how one might use data gathered from our algorithm, we will instead use these samples to generate hypotheses to validate on a separate dataset: this is similar in principle to how a researcher might run multifactorial experiments in an online setting to find promising hypotheses to test in a more focused setting, as suggested by Stamper et al. [24].

UCBExplore can be thought of as a biased method of drawing samples from different experimental conditions. One natural analysis would be to ask if one condition is significantly better than another. Many of our factors are specific implementation choices in our game, so such comparisons may not lead to very generalizable insights. Instead, we will attempt to identify likely interactions using the samples generated by our proposed method. In particular, we will search for two-way interactions which seem relatively large or small; we study interactions instead of main effects because we already examined main effects in previous work [17]. Then we will use the likelihood ratio test to determine if models learned with these interaction terms fit our validation data significantly better than models where these terms are set to zero. We stress that samples generated from ϵ-greedy and UCB1 could also be used in the same way and would likely lead to the same results, though in light of our simulation results either more samples would be required or more damage would be done to players in that case.

	Target: Pie	Target: Symbolic
Label: Pie	0.431	0.410
Label: Symbolic	0.388	0.422

Table 3: Proportion of players in the validation set able to reach and answer the randomized test number line correctly on the first try. Our simulation results suggested that these parameters might interact; in fact, they interact very strongly.

To do this, we run UCB-Explore with m=1.1, w=0.001, and 100,000 pulls. We then calculate all main effects and two-way factor interactions as is done in the ANOVA test [28]. Our data is not normally distributed and the variances are unequal, violating ANOVA assumptions, but we can still consider which interaction effects seem relatively large or small. In our case, for each pair of factors, we can calculate the average magnitude of the interaction effects between all combinations of settings for those factors. We see that Target and Label have the largest average magnitude at 0.007, while Animation and Ticks have the smallest average magnitude at 0.0003. Thus, we suspect that Target and Label are much more likely to interact than Animation and Ticks.

To test these hypotheses, we will use a held-out validation dataset. This dataset consists of 9,675 players of Treefrog Treasure from June 20, 2013 to July 9, 2013. Unlike the dataset used to estimate parameters of our simulation in the previous section, we will consider only the first three number lines for each player: the first two are treated as the intervention, and the independently and randomly chosen third as the assessment of learning. For any given pair of factors, we can attempt to fit a model with only main effects, or a model with main effects and two-way interaction effects. Since these models are nested, we can use the likelihood ratio test if the interaction model is a significantly better fit, given the increase in degrees of freedom.

For Target and Label, we have that $\chi^2(1, N = 9675) = 7.555, p < 0.006$. For Animation and Ticks, we have that $\chi^2(3, N = 9675) = 0.204, p = 0.977$. Thus, it is very likely that the effects of Target and Label representation on numberlines should not be considered separately, while we have no evidence that our Animation and Ticks hinting systems need to be modeled simultaneously.

7.2 Target/Label representation

In this paper, our goal is to advocate the creation of algorithms which allow experimenters to trade off user learning and scientific knowledge, and the introduction of such an algorithm. The validation is meant to show that this approach generates samples that can lead to interesting hypotheses, so we do not claim them to be mature educational results. We will, however discuss them briefly.

The presence of significant interaction terms means that the factors involved should only be interpreted together. As a

reminder, the Target factor refers to the representation of the fraction the player is asked to hit on the number line, while the Label factor refers to the representation of the fractions on the number line itself. The proportions of successful players for the different representation combinations can be seen in Table 3. The nature of the interaction is immediately apparent: players are more likely to reach and answer the next number line correctly if the target and label have the same representation, and Pie chart targets and Symbolic labels are much worse than other conditions. Even when we ignore players who quit before reaching the second number line, these effects persist.

We do not know why this is the case. In our game, player ability to answer number line questions correctly is mostly a function of knowing where to hit, as the game informs the player where they will intersect the number line before they click to jump. In addition, the representation has no effect on game mechanics. Thus, the difference is most likely due to how players perceive the different fraction representations. One possibility is that number lines in classrooms are generally presented with symbolic labels only, so that the mix of familiar and new combinations of representations is particularly confusing to players. As with nearly all online experiments, we do not have access to players' thought processes, only their actions, so a more carefully designed study or a think-aloud in a classroom might be profitable. Regardless, our algorithm was able to generate samples that we could analyze for interesting factor interactions, suggesting that it is a viable method of adaptively randomizing experimental conditions.

8. LIMITATIONS

While our results seem promising, there are some limitations. It is extremely unlikely that UCB-Explore is ideal for all bandit arm configurations; it seems to perform well on many-arm Bernoulli distributions with similar means, but some simulations suggest that it may not perform as favorably in very different cases. One example where our approach fails outright, as do ϵ-greedy and UCB1, is in the case that there are more arms than subjects - the algorithm expends all its subjects on the initialization phase when pulling each arm once. This problem occurs most obviously in the presence of continuous factors, in which case there are an infinite number of arms. It is less likely to occur in standard categorical experimental designs, though the limit can still be reached if researchers want to study something like the exponential space of all possible problem sequences.

Furthermore, while the reward portion of our dual objectives can be any measurable function of subject behavior, there may be other ways to define "scientific knowledge" or navigate the tradeoff between the two. For example, "scientific knowledge" might be the probability that the ordering of arms is correct. Or instead of assigning some weighting between reward and knowledge, an experimenter might have constraints of the form "maximize reward subject to at least x knowledge." Some of these are relatively easy to incorporate into our framework. The example constraint might be handled by forcing the scale factor to stay at 1.0 until enough information has been collected, for instance. Other types of constraints or tradeoffs may require entirely different algorithms: knowing the number of subjects in advance, for example, leads to very different bandit algorithms than the infinite horizon variant we have presented here.

9. FUTURE WORK

UCB-Explore appears to outperform UCB1, UCB1-Tuned, and ϵ-greedy in our problem for most tradeoffs between reward and knowledge. However, our modifications remove any theoretical performance guarantees. It may be possible to alter the algorithm in a principled way to maintain its good performance and guarantee logarithmic regret. Extensive simulation on other problems with more or fewer arms and different reward variances would also be useful to understand when it or another method of allocating samples is preferable. We would also like to test if this algorithm is robust to unusual continuous reward distributions.

In addition, there are other problem formulations that would be interesting to investigate. In many practical cases (such as delayed rewards), fully online learning is infeasible; for these, we could adapt Bayesian techniques such as probability matching [21], which do not depend on online performance. Also, in cases where there is a finite budget of users, the algorithm will need to be modified based on work in mortal bandits to exploit more as the experiment draws to a close [7]. Lastly, in reality the arm distributions may be nonstationary, which might require adaptation of work in dynamically changing bandits [9].

More generally, UCB-Explore only makes sense in situations where we have enough users to get substantial information about each condition. If we have many factors, we may not be able to get information about each specific condition, but may still be able to determine the best settings of the most important factors. A similar problem arises when we want to explore sequences of interventions, in which case techniques from Monte Carlo Tree Search may be most applicable. And if one of the factors is continuous, the algorithm will not be able to make progress: here, it could be useful to adapt work on bandits in general metric spaces [11], or modifying a function approximation scheme as in [16] to incorporate the reward-knowledge tradeoff.

10. CONCLUSION

The rise of online educational software with massive numbers of users promises to change the experimental paradigm in educational research. With access to so many users and individualized control over what educational experiences they receive, it is now possible to automatically run complicated, multi-factor experiments quickly and at relatively low cost. However, education is a high-stakes domain: in many situations we have the ability to cause harm by placing students in sub-optimal conditions. Because of this we want to automatically put less students into harmful conditions while simultaneously discovering which are harmful and which are beneficial.

In this paper, we propose allowing researchers to explicitly weight subject welfare against the amount of generalizable knowledge gained from the experiment. We show how the problem of allocating subjects to experimental conditions can be thought of as a multi-armed bandit problem with a dual objective of gaining maximum reward and minimizing the sizes of 95% confidence intervals around the arm means.

We propose a new algorithm, UCB-Explore, which takes a user-specified weight on the relative value of reward and confidence interval size, and adaptively adjusts its optimistic bound estimates to explore or exploit more when it makes a mistake with regards to this weight function. We analyze the behavior of our algorithm and compare it to tuned versions of other common bandit algorithms in a 64-arm simulation with parameters drawn from real-world data, showing that our algorithm is able to interpolate between these two goals much more effectively than standard algorithms. We use the simulated results of running our algorithm to generate some hypotheses about factor interactions, and confirm these results on a separate validation dataset, showing that the generated samples are useful from a research perspective.

11. ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship grant No. DGE-0718124, as well as the Office of Naval Research grant N00014-12-C-0158, the Bill and Melinda Gates Foundation grant OPP1031488, the Hewlett Foundation grant 2012-8161, Adobe, and Microsoft.

12. REFERENCES

[1] D. Ansari. Effects of development and enculturation on number representation in the brain. *Nature Reviews Neuroscience*, 9(4):278–291, 2008.

[2] A. Antos, V. Grover, and C. SzepesvÃąri. Active learning in multi-armed bandits. In Y. Freund, L. GyÃűrfi, G. TurÃąn, and T. Zeugmann, editors, *Algorithmic Learning Theory*, volume 5254 of *Lecture Notes in Computer Science*, pages 287–302. Springer Berlin Heidelberg, 2008.

[3] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

[4] D. A. Berry. Adaptive clinical trials in oncology. *Nature reviews Clinical oncology*, 9(4):199–207, 2012.

[5] BrainPOP. http://www.brainpop.com/.

[6] S. Bubeck and N. Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *arXiv preprint arXiv:1204.5721*, 2012.

[7] D. Chakrabarti, R. Kumar, F. Radlinski, and E. Upfal. Mortal multi-armed bandits. In *NIPS*, pages 273–280, 2008.

[8] S.-C. Chow, M. Chang, et al. Adaptive design methods in clinical trials-a review. *Orphanet J Rare Dis*, 3(11), 2008.

[9] A. Garivier and E. Moulines. On upper-confidence bound policies for non-stationary bandit problems. *arXiv preprint arXiv:0805.3415*, 2008.

[10] R. D. King, J. Rowland, S. G. Oliver, M. Young, W. Aubrey, E. Byrne, M. Liakata, M. Markham, P. Pir, L. N. Soldatova, A. Sparkes, K. E. Whelan, and A. Clare. The automation of science. *Science*, 324(5923):85–89, 2009.

[11] R. Kleinberg, A. Slivkins, and E. Upfal. Multi-armed bandits in metric spaces. In *Proceedings of the 40th ACM Symposium on Theory of Computing*, STOC '08, pages 681–690, New York, NY, USA, 2008. ACM.

[12] V. Kuleshov and D. Precup. Algorithms for the multi-armed bandit problem. *Journal of Machine Learning*, 2000.

[13] T. L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

[14] P. Langley. *Scientific discovery: Computational explorations of the creative processes*. MIT press, 1987.

[15] Y. Lee, B. Buchanan, and J. Aronis. Knowledge-based learning in exploratory science: Learning rules to predict rodent carcinogenicity. *Machine Learning*, 30(2-3):217–240, 1998.

[16] R. Lindsey, M. Mozer, W. J. Huggins, and H. Pashler. Optimizing instructional policies. In *Advances in Neural Information Processing Systems*, pages 2778–2786, 2013.

[17] Y.-E. Liu, T. Mandel, E. Brunskill, and Z. Popović. Towards automatic experimentation of educational knowledge. In *CHI*, 2014.

[18] D. Lomas, K. Patel, J. L. Forlizzi, and K. R. Koedinger. Optimizing challenge in an educational game using large-scale design experiments. In *CHI*, 2013.

[19] W. B. Powell and I. O. Ryzhov. *Optimal learning*, volume 841. John Wiley & Sons, 2012.

[20] L. Razzaq, M. Feng, G. Nuzzo-Jones, N. T. Heffernan, K. Koedinger, B. Junker, S. Ritter, A. Knight, E. Mercado, T. E. Turner, R. Upalekar, J. A. Walonoski, M. A. Macasek, C. Aniszczyk, S. Choksey, T. Livak, and K. Rasmussen. Blending assessment and instructional assisting. In *Proceedings of the 2005 Conference on Artificial Intelligence in Education: Supporting Learning Through Intelligent and Socially Informed Technology*, pages 555–562, Amsterdam, The Netherlands, The Netherlands, 2005. IOS Press.

[21] S. L. Scott. A modern bayesian look at the multi-armed bandit. *Applied Stochastic Models in Business and Industry*, 26(6):639–658, 2010.

[22] R. S. Siegler, C. A. Thompson, and M. Schneider. An integrated theory of whole number and fractions development. *Cognitive Psychology*, 62(4), 2011.

[23] K. Singh and M. Xie. Bootstrap: A statistical method, 2008.

[24] J. C. Stamper, D. Lomas, D. Ching, S. Ritter, K. R. Koedinger, and J. Steinhart. The rise of the super experiment. In *EDM*, pages 196–200, 2012.

[25] J. Vermorel and M. Mohri. Multi-armed bandit algorithms and empirical evaluation. In *Machine Learning: ECML 2005*, pages 437–448. Springer, 2005.

[26] L. J. Wei and S. Durham. The randomized play-the-winner rule in medical trials. *Journal of the American Statistical Association*, 73(364):pp. 840–843, 1978.

[27] E. B. Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927.

[28] B. J. Winer. *Statistical principles in experimental design*. McGraw-Hill Book Company, 1962.

[29] M. Zelen. Play the winner rule and the controlled clinical trial. *Journal of the American Statistical Association*, 64(325):pp. 131–146, 1969.