Learning Algorithms for Online Principal-Agent Problems (and Selling Goods Online)

Vincent Conitzer CONITZER@CS.CMU.EDU
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213 USA

Nikesh Garera NGARERA@CS.JHU.EDU
Department of Computer Science, Center for Language and Speech Processing, Johns Hopkins University, Baltimore, MD 21218 USA

Abstract

In a *principal-agent problem*, a principal seeks to motivate an agent to take a certain action beneficial to the principal, while spending as little as possible on the reward. This is complicated by the fact that the principal does not know the agent's utility function (or *type*). We study the online setting where at each round, the principal encounters a new agent, and the principal sets the rewards anew. At the end of each round, the principal only finds out the action that the agent took, but not his type. The principal must learn how to set the rewards optimally. We show that this setting generalizes the setting of selling a digital good online.

We study and experimentally compare three main approaches to this problem. First, we show how to apply a standard bandit algorithm to this setting. Second, for the case where the distribution of agent types is fixed (but unknown to the principal), we introduce a new gradient ascent algorithm. Third, for the case where the distribution of agents' types is fixed, and the principal has a prior belief (distribution) over a limited class of type distributions, we study a Bayesian approach.

1. Introduction

In many economic settings, one party (the *principal*) desires that another party (the *agent*) takes a certain action, even though the agent has no direct interest in the action being performed. For instance, the principal may have a certain task that she does not have the resources to perform, or that is more efficiently performed by the agent. As an illustrative example, a homeowner (the principal) may want her neighbor's child (the agent) to shovel her driveway. Unless the agent can be forced to take the desired action, the principal will need to *motivate* him to do so, by promising a reward (monetary or other) for performing the action. This reward typically comes at the cost of the principal's utility. Thus, if the principal knew the agent's utility function, she would promise him a reward that is only just sufficiently high for the agent to want to take the action. However, typically, the agent's utility function is not known, so that a more complex optimization needs to be performed. If the principal has a probability distribution over the agent's utility function (also known as the agent's *type* in this context), she can optimize the promised reward to maximize her expected utility. Such problems are known as *principal-agent* problems (for an overview, see (Mas-Colell et al., 1995)). Applications include deciding on incentives for employees, deciding on the most effective grading system to motivate students, *etc.* Principal-agent problems are closely related to (or subsumed by, depending on the definition) problems of *implementation* and *mechanism design*—topics that are receiving ever-increasing attention among AI researchers (some recent examples include (Porter, 2004; Bartal et al., 2004; Parkes & Schoenebeck, 2004; Blumberg & Shelat, 2004; Conitzer & Sandholm, 2004; Smorodinsky & Tennenholtz, 2004; Bahar & Tennenholtz, 2005; Babaioff et al., 2005)). As we will show, they are also closely related to *pricing* problems, where prices for goods need to be set to maximize the seller's (expected) gain.

In practice, typically not even a distribution over the types is known.[1] However, if the problem repeats itself, the principal may be able to *learn* how to set the rewards. We consider the online model where the principal faces a new agent every round, and gets to set a new reward structure

Appearing in *Proceedings of the 23^{rd} International Conference on Machine Learning*, Pittsburgh, PA, 2006.

[1] From a decision-theoretic standpoint, the principal will have some subjective distribution over the agent's types, but this distribution may or may not be close initially to the distribution of types in the general population of agents.

every round. We assume that the principal only observes the action taken by the agent, and not the agent's type (not even after the round is over). Thus, we cannot infer what the agent would have done for other reward structures. Such problems, in which one of multiple alternatives is to be chosen, and only the result from the chosen alternative is observed, are called *bandit* problems (Auer et al., 1995).

In this paper, we compare three approaches to the online principal-agent problem. First, we take a general bandit algorithm from the online learning literature and show how to apply it to this new domain. We then introduce a new approach, which requires that the agents' types are drawn from a fixed distribution. It also requires that we can add another action to the domain, which is no more useful to the principal than the original action, but less desirable from the agent's perspective (we will argue that for many applications, this can be done). We show how to use this extra action to do a form of gradient ascent on the reward space. In general, this gradient ascent approach can get stuck in a local optimum, but natural distributions typically satisfy a concavity condition that precludes such local optima. We provide experimental results for the fixed-distribution case, showing that the second approach converges much faster and generates less variance in the rewards that are set. Finally, we study the case where the principal has a prior belief over the agents' type distribution, so that a Bayesian approach can be taken. We show that this approach can learn the optimal reward extremely fast when the prior belief places probability on a limited class of type distributions, but may fail to converge to the optimal reward when this class becomes larger. We also discuss how to apply these techniques to online auctions.

2. Definitions

We now present the basic definitions for one round of the online principal-agent problem as studied in this paper.

Action set: The agent has to choose one action from an action set A. We will first concern ourselves with the case where $A = \{a_0, a_1\}$; later in the paper, we will add a third action a_2. The action a_0 can be interpreted as the agent doing nothing. We will assume that associated with each action a, there is a fixed amount of *work*, $w(a)$, that is independent of the agent's type. We will use the shorthand $w_i = w(a_i)$, and assume that $w_0 = 0$.

Reward function: Before the agent chooses his action, the principal must set a reward function $r : A \to R$, where R is the space of possible rewards. $r(a)$ is the reward that the principal pays the agent upon selecting a. We will use the shorthand $r_i = r(a_i)$. We will let $R = \mathbb{R}$, and assume $r_0 = 0$. Thus, when $A = \{a_0, a_1\}$, the principal only needs to set r_1; once we add a_2, she will also need to set r_2.

Agent's utility: The agent's utility depends on the reward he collects, as well as his dislike for the action he performs. This dislike is given by his type t, which is known to the agent but unknown to the principal. The agent is rational and will always choose the action that maximizes his utility, given the reward function. We will assume the agent's utility function can be written as $V(t, a, r) = v(t, a) + r(a) = -tw(a) + r(a)$. That is, the type t represents the agent's dislike for doing a unit of work. The space of possible types is thus $\mathbb{R}$. We note that $v(t, a_0) = 0$ for all t.

Principal's utility: The principal's utility depends on the reward she must award, as well as her appreciation for the action the agent performs. We will assume the principal's utility function can be written $U(a, r) = u(a) - r(a)$. We will use the shorthand $u_i = u(a_i)$. We will assume $u_0 = 0$.

This interaction between the principal and an agent repeats itself each round. At each round, t is drawn anew (we can think of this as the principal facing a different agent every round). The principal's learning objective can be either to perform well relative to the best fixed reward function in hindsight (a standard objective in the online learning literature), or, in case the types are drawn from a fixed distribution, to converge to an expected (principal's) utility-maximizing reward function.

3. Application: Selling a digital good online using posted prices

In this section, we show how the previously studied problem of learning to sell a digital good in an online setting (Bar-Yossef et al., 2002; Blum et al., 2003) can be seen as a special case of the setting that we study. In this problem, we have a seller with an unlimited supply of a good. At each round, the seller faces a new buyer with an unknown value for a unit of the good, and the seller needs to set a price p for a unit of the good for this buyer. The buyer will buy a unit of the good at the posted price if and only if his value exceeds the price.

We can model this as a principal-agent problem as follows. The seller will take the role of the principal, and the buyer will take that of the agent. The seller will give the buyer the choice of two actions: not buying the good (denoted by a_0), and buying the good (denoted by a_1). The seller will set a negative "reward" r_1 for buying the good, which is the negative of the price of the good ($r_1 = -p$). Let the amount of "work" associated with obtaining the good be $w_1 = -1$ (one way to interpret this is that the good may facilitate other tasks for the agent and thereby reduce the agent's workload), and let the agent's type t indicate his utility for having the good (or his dislike for the work that he would not have to do if he had the good). Then, the agent's utility for buying the good is $-tw_1 + r_1 = t - p$. For

the seller, we set $u_1 = u_0 = 0$ (since the seller is indifferent between keeping the good herself and not keeping it), so that the seller's utility for selling the good is $u_1 - r_1 = p$. (More generally, if the seller has a reservation price h for the good, we can set $u_1 = -h$ so that the seller's utility for selling the good is $u_1 - r_1 = p - h$.)

4. How to Apply a General Bandit Algorithm

In this section, we show how to apply a general online bandit learning algorithm to the principal-agent problem.

4.1. Viewing our problem as a bandit problem

In a bandit setting (Auer et al., 1995), we must choose one of a number of "arms," and we then receive a utility that depends on the arm we chose. We do not find out what rewards the other arms would have given us. (This is in contrast with *experts* settings (Cesa-Bianchi et al., 1997; de Farias & Megiddo, 2003), where we do find out what utilities we would have received for following other experts' recommendations.) This process repeats itself for a number of rounds, over which we accumulate utility. The standard goal is to perform well compared to the best (fixed) arm in hindsight.

In the online principal-agent problem, the "arms" of the bandit correspond to the various reward functions the principal can set for the agent. Each reward function will give the principal some utility (depending on the agent's type), but the principal does not find out the utilities that other reward functions would have given her, because by assumption the agent's type remains unknown at the end of the round.[2] This approach requires that we discretize the reward space in order to have a finite number of arms.[3] In our setting, a reward function consists of a single reward r_1, so the arms correspond to different values for r_1.

4.2. Algorithm

The bandit algorithm **Exp3** (Auer et al., 1995) specializes to the following form in our domain. (We refer the reader to the original work for the intuition behind the algorithm and the meaning of its parameters.)

BANDIT ALGORITHM

$R \leftarrow$ { possible rewards }
for all $r \in R$ {
  $s_r \leftarrow 0$, $m_r \leftarrow 1$
}
repeat {
  with probability γ {
    $r_c \leftarrow$ choose reward uniformly at random
  }
  else {
    $r_c \leftarrow$ choose reward r with probability $\frac{m_r}{\sum_{r' \in R} m_{r'}}$
  }
  set reward $r_1 = r_c$
  if the agent chooses a_1 {
    $s_{r_c} \leftarrow s_{r_c} + \frac{(u_1 - r_c)\gamma}{|R|(\frac{\gamma}{|R|} + (1-\gamma)\frac{m_{r_c}}{\sum_{r' \in R} m_{r'}})}$
    $m_{r_c} \leftarrow (1+\alpha)^{\frac{s_{r_c}}{h}}$
  }
}

This algorithm is effectively identical to the algorithm presented in (Blum et al., 2003) for learning to sell a digital good in an online setting. As such, it is a natural benchmark for the algorithms that we present later in the paper.

4.3. Experimental results

In this subsection, we describe experimental results for the bandit algorithm. Let $u_1 = 1$, $w_1 = 1$. Every round, t is drawn from a fixed distribution: we study both a uniform distribution over $[0, 1]$, and an exponential distribution with $\lambda = 1$. It can be shown analytically that with these distributions, the optimal (expected utility-maximizing) settings for r_1 are 0.5 and 0.44, respectively. We discretize the reward space at intervals of 0.05 to get a finite number of bandit arms, and do not consider rewards greater than 1 (because the optimal rewards clearly lie below 1).

First, we present the results for the best setting of the parameters that we found when holding γ fixed over time (Figures 1 and 2). (We show the chosen r_1 only every 1000th round.)

[2] This is why we require a bandit algorithm instead of an experts algorithm. We note that in our setting it is not true that we cannot make *any* inferences about what would have happened for other reward functions: for example, if the agent did not take an action, we know that the agent would still not have taken this action at a lower reward. Nevertheless, this is not enough to allow us to use an experts algorithm. Moreover, in the setting of selling a digital good online (discussed above), it has been shown theoretically (Blum et al., 2003) that having to use a bandit algorithm does not come at much of a loss relative to being able to use an experts algorithm. Such experts algorithms in turn outperform other methods (Bar-Yossef et al., 2002) designed specifically for this problem (and that use the same information as an experts algorithm). Thus, bandit algorithms constitute the natural baseline for our problem.

[3] Recent work on online convex programming (Zinkevich, 2003) provides an algorithm that does not require discretization for settings with a continuum of experts, but it cannot be applied here because we are in a bandit setting and do not observe the principal's utility for rewards other than the one chosen (which is necessary for a gradient computation in that algorithm). Additionally, the principal's utility function is discontinuous in rewards.

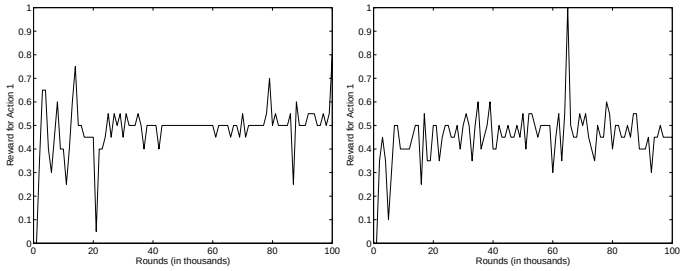

Figure 1. Convergence for uniform (left) and exponential (right) distributions with $\gamma = 0.1$, $\alpha = 0.8$, and $h = 0.2$.

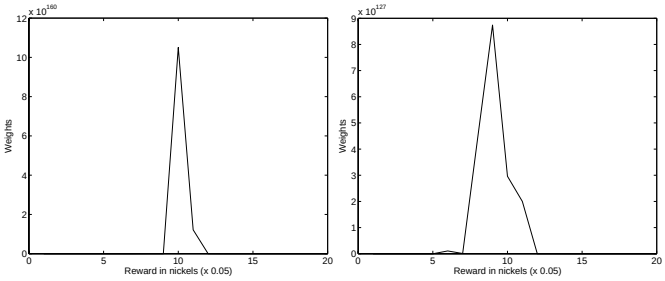

Figure 2. Final weights m_r on rewards for uniform (left) and exponential (right) distributions with $\gamma = 0.1$, $\alpha = 0.8$, and $h = 0.2$. Recall that we have discretized the reward space at intervals of 0.05, hence the reward is in "nickels."

The final weights are quite good in both cases, placing almost all of the probability mass on optimal or near-optimal rewards. The algorithm chooses a random reward with probability $\gamma = 0.1$, leading to "spikes" in the convergence graphs; but disregarding the spikes the algorithm converges quite fast.

The next results (Figures 3 and 4) show that the spikes can be minimized by decreasing γ over time according to a fixed schedule. However, we cannot decrease γ too fast without hurting the algorithm's convergence. (We show the results for the best setting of the parameters that we found.)

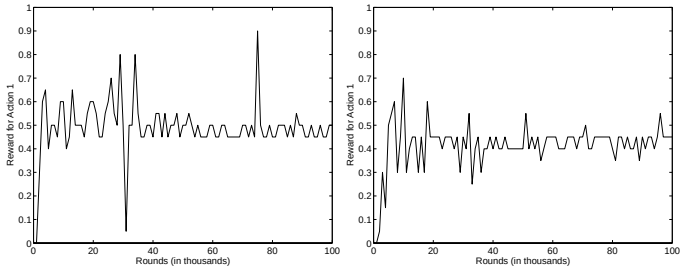

Figure 3. Convergence for uniform (left) and exponential (right) distributions with $\gamma = \frac{0.1}{i^{0.1}}$ in round i, $\alpha = 0.8$, and $h = 0.2$.

The final weights are again quite good, and we do not see as many spikes in the rewards in later rounds.

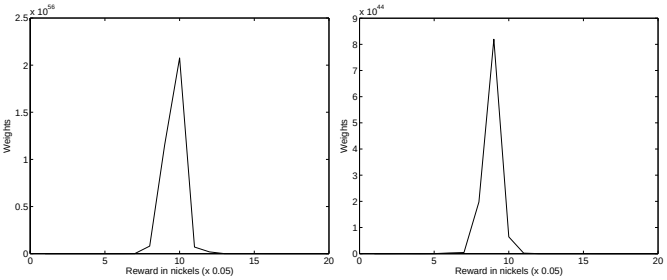

Figure 4. Final weights m_r on rewards for uniform (left) and exponential (right) distributions with $\gamma = \frac{0.1}{i^{0.1}}$ in round i, $\alpha = 0.8$, and $h = 0.2$.

5. Gradient Ascent

We now take a new *gradient ascent*-based approach to the online principal-agent problem.

5.1. Intuition

For the gradient ascent algorithm, we assume that the distribution from which the agents' types are drawn is fixed.[4] As a result, there is some reward structure (some r_1) that maximizes the principal's expected utility. The goal is to find this r_1 using gradient ascent. However, the principal does not actually know the gradient, because the distribution of agents' types is unknown. Instead, we propose a method that closely approximates gradient ascent *in expectation*.

To do so, consider the tradeoffs involved in slightly increasing (or decreasing) r_1. (We will derive exact expressions for these tradeoffs in the next subsection.) On the one hand, increasing r_1 will decrease the principal's utility for any type for which the agent performs a_1, because the principal will have to pay more. On the other hand, increasing r_1 will increase the set of types for which the agent performs action a_1, thereby increasing the probability that the agent performs a_1. Thus, the decision of whether to increase r_1 by a given amount depends on the probability mass on the types for which the agent will perform a_0 with the original reward, but a_1 with the new reward. This mass is difficult to estimate, because the principal does not see what the agent's type was at the end of a round. One approach to estimating this mass is to run with a fixed reward $r_1 = r$ for a long time, then to run with $r_1 = r + \epsilon$ for a long time, and finally to take the difference in the fraction of times that the agent chose to perform a_1 between these two approaches. Of course, to be confident in our estimate, we would have to run for a very long time in both cases.

[4]The bandit algorithm from the previous section does not assume this, even though we did do that section's experiments in the fixed-distribution setting (so that we can compare them with the experiments for this algorithm).

Instead, we take the following approach. We assume that we can add another action a_2 to the action space, with the following properties. First, the principal is indifferent between the agent performing a_1 and a_2—that is, $u(a_1) = u(a_2)$. Second, the agent prefers performing a_1 to performing a_2, because a_2 is slightly more work—w_2 exceeds w_1 slightly. (We note that because we assume the agent's utility function can be written as $V(t, a, r) = v(t, a) + r(a) = -tw(a) + r(a)$, all the work has the same "dislike" multiplier t, and in this sense, a_2 is *more of the same* work as a_1.) For example, if a_1 corresponds to the task of shoveling a large area, then a_2 may correspond to the task of shoveling the same area, plus another small area for which the principal does not care if it is shoveled or not. In the case where we are selling a good online, a_2 may correspond to buying a slightly inferior product—for example, one with a reduced quality or lifetime. (We note that most real-world goods can straightforwardly be made slightly worse in the manner desired. For example, the lifetime of a product can always be reduced by using the product before the sale.)

Naturally, the reward r_2 must be set at least slightly higher than r_1 if the agent is ever to choose a_2. As we will show, the principal can set the reward r_2 to effect the following: the agent will choose a_1 rather than a_2 if and only if his type t was almost high enough to choose a_0. Thus, a_1 *being chosen indicates that the agent is close to being indifferent between all the actions*, that is, t is close to r_1/w_1. Assuming continuity of the probability density function over types, this suggests that there is significant probability mass on types close to r_1/w_1. This suggests that increasing the rewards is likely to decrease the probability of a_0 being selected significantly.

We note that, unlike in the case of the bandit algorithm from the previous section, we do not need to discretize the reward space. The following subsection will make the algorithm more precise, and show how to turn this observation into an online gradient ascent algorithm.

5.2. Analysis and Algorithm

The principal's objective is to maximize her expected utility. The agent will switch from a_0 to a_1 when $V(t, a_0, r_0) = V(t, a_1, r_1)$, or equivalently, $t = \frac{r_1}{w_1}$. Because the principal's utility for the agent choosing a_1 is $u_1 - r_1$, the principal's expected utility is given by $U = (\int_{t=-\infty}^{\frac{r_1}{w_1}} p(t)dt)(u_1 - r_1)$. We emphasize that this is the expression for her expected utility *when action a_2 does not come into play*. Our goal in this section will be to let r_1 converge to a value that maximizes this expression, rather than to set r_1 and r_2 to maximize the expected utility of the three-action setting. This makes our analysis easier, and it is reasonable because: 1. a_2 is helpful for learning the distribution over types, but for any *single* round, there is no purpose in using it. 2. In our algorithm, we will keep r_1 and r_2 very close to each other, and thus the above expression is a close approximation of the actual expected utility for the principal in any given round.

Taking the derivative of U, we get $\frac{dU}{dr_1} = \frac{p(\frac{r_1}{w_1})}{w_1}(u_1 - r_1) - \int_{t=-\infty}^{\frac{r_1}{w_1}} p(t)dt$. To use this to do gradient ascent, we will need to estimate both $p(\frac{r_1}{w_1})$ and $\int_{t=-\infty}^{\frac{r_1}{w_1}} p(t)dt$. The latter is easy to estimate without introducing an extra action, by the fraction of times that a_1 is selected. However, to estimate the other term, we will need to introduce another action, a_2. In the following, we describe how to do this.

We recall that the agent will switch from a_0 to a_1 when $t = \frac{r_1}{w_1}$. He will switch from a_1 to a_2 when $V(t, a_1, r_1) = V(t, a_2, r_2)$, or equivalently, $t = \frac{r_2 - r_1}{w_2 - w_1}$.

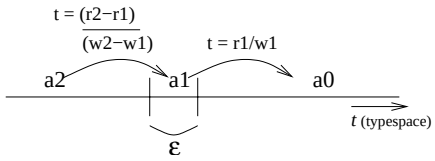

Figure 5. Switching points between actions.

The purpose of the new action is to get an estimate of the probability mass on types for which the agent is close to being indifferent between the actions. What is considered "close" is defined by the distance between the two switching points in the type space; we can determine this distance ourselves by setting the rewards appropriately. Suppose we want the distance between the switching points to be ϵ. Then we must set $\frac{r_2 - r_1}{w_2 - w_1} = \frac{r_1}{w_1} - \epsilon$, or equivalently $r_2 = \frac{w_2}{w_1} r_1 + \epsilon(w_1 - w_2)$. Given that ϵ is small, we can say that the probability of a_1 being selected is approximately $\epsilon p(\frac{r_1}{w_1})$, and the probability of a_2 being selected is approximately $\int_{t=-\infty}^{\frac{r_1}{w_1}} p(t)dt$—corresponding to the terms we were seeking to estimate. Now consider the following algorithm:

GRADIENT ASCENT ALGORITHM

initialize r_{c1}, r_{c2}
repeat {
  set rewards $r_1 = r_{c1}, r_2 = r_{c2}$
  if the agent chooses a_0
    do nothing
  if the agent chooses a_1
    increase r_{c1} by $\frac{u_1 - r_{c1}}{\epsilon w_1} K$
  if the agent chooses a_2
    decrease r_{c1} by K
  update $r_{c2} \leftarrow \frac{w_2}{w_1} r_{c1} + \epsilon(w_1 - w_2)$
}

With this algorithm, the *expected change* to r_1 in any round is approximately:

$$\epsilon p(\tfrac{r_1}{w_1})\tfrac{u_1-r_1}{\epsilon w_1}K - \int_{t=-\infty}^{\frac{r_1}{w_1}} p(t)dtK = K(p(\tfrac{r_1}{w_1})\tfrac{u_1-r_1}{w_1} - \int_{t=-\infty}^{\frac{r_1}{w_1}} p(t)dt) = K\tfrac{dU}{dr_1}.$$

It follows that (in expectation) we are (approximately) doing gradient ascent, and the step size is proportional to K. Thus, if we initialize the rewards within the support of the distribution, K is set appropriately over time, and the objective function has no local optima other than the global optimum, then we will converge to the optimal setting for r_1. We note that for the uniform and exponential distributions, $\frac{dU}{dr_1} = \frac{p(\frac{r_1}{w_1})}{w_1}(u_1 - r_1) - \int_{t=-\infty}^{\frac{r_1}{w_1}} p(t)dt$ is decreasing in r_1 because the probability density function is nonincreasing; hence, the objective function is concave and has no local optima other than the global optimum.

5.3. Experimental results

In this subsection, we describe experimental results for the gradient ascent algorithm. Let $u_1 = u_2 = 1$, $w_1 = 1$, $w_2 = 1.001$. Every round, t is drawn from a fixed distribution: again, we study both a uniform distribution over $[0, 1]$, and an exponential distribution with $\lambda = 1$. We recall that the optimal (expected utility-maximizing) settings for r_1 are 0.5 and 0.44, respectively. We let $\epsilon = 0.02$. We note that for the gradient ascent algorithm, we do not need to discretize or cap the reward space (as we did for the bandit algorithm).

First (Figure 6), we present the results obtained when holding K fixed over time at the best value that we found for it. (We show the bandit algorithm's results with fixed γ in the background of the graphs as a dotted line.)

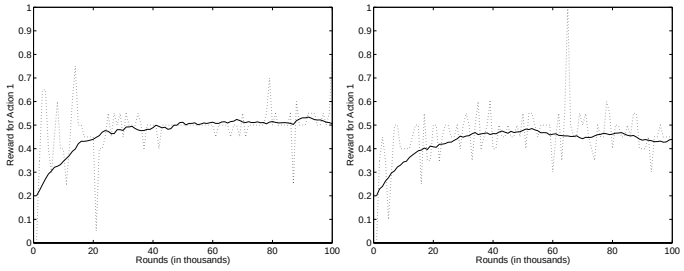

Figure 6. Convergence for uniform (left) and exponential (right) distributions with $K = .00005$.

Convergence is reasonably fast, but because K is fixed, the rewards fluctuate around the optimal reward. (It is possible to obtain much faster convergence, at the cost of much greater fluctuation, by increasing K.) The solution is, of course, to decrease K over time.

We next (Figure 7) present results where K decreases over time according to a fixed schedule. (We show the bandit algorithm's results with a changing γ in the background of the graphs as a dotted line.)

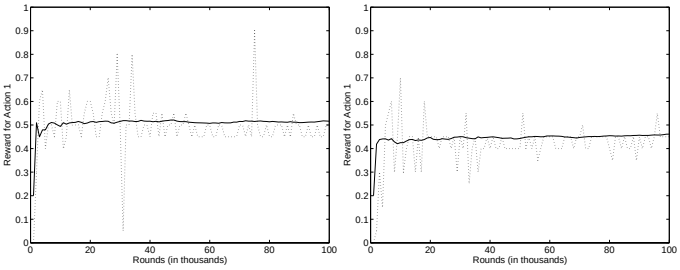

Figure 7. Convergence for uniform (left) and exponential (right) distributions with $K = \frac{0.1}{i^{0.8}}$ in round i.

Convergence is now almost immediate, and fluctuation is minimized because K is decreased over time.

6. A Bayesian approach

In this section, we take a Bayesian approach. We continue to assume that agents' types are drawn from a fixed distribution, but we will no longer require the availability of an extra action.[5] We do, however, require that the principal has a prior belief (distribution) over the distribution of agents' types. Each round, the principal chooses the reward that maximizes her expected utility, given her beliefs over type distributions; then, the principal updates these beliefs based on the observed result, using Bayes' rule. (We omit the mathematical details due to space constraint, but they are straightforward.)

Again, we run experiments on both a uniform distribution over $[0, 1]$, and an exponential distribution with $\lambda = 1$. We must now also specify the principal's prior beliefs, and we do so as follows. In each case, the principal knows the type of distribution (uniform over $[0, k]$, or exponential with parameter λ), but is uncertain about the parameter in the distribution (k or λ, respectively). Specifically, the principal initially believes that the parameter (k or λ, respectively) is equally likely to be any member of $\{0.00, 0.01, \ldots, 10.00\}$. The results are in Figures 8 and 9.

[5]The Bayesian approach can be extended to make use of such an extra action; however, insofar as the Bayesian approach can perform well without it, it is of course preferable not to require this assumption. We will show shortly that the Bayesian approach can indeed perform well without the extra action.

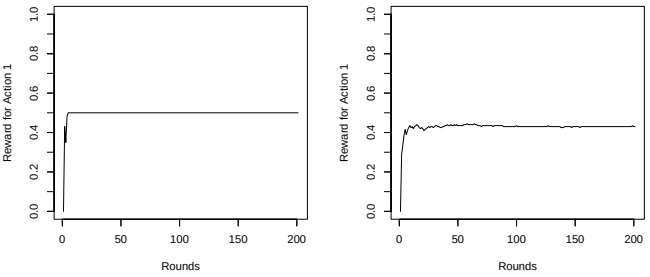

Figure 8. Convergence for uniform (left) and exponential (right) distributions under the Bayesian approach, when the prior places probability only on distributions of the correct type (uniform or exponential).

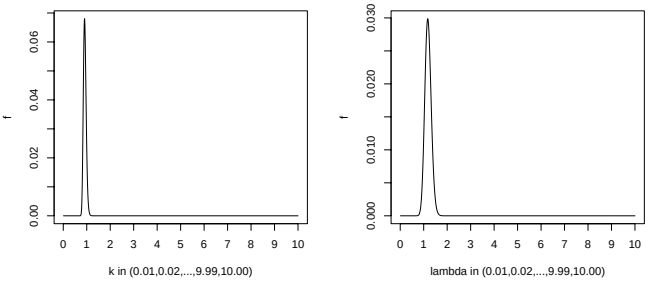

Figure 9. Final principal beliefs over parameter values for uniform (left) and exponential (right) distributions under the Bayesian approach, when the prior places probability only on distributions of the correct type (uniform or exponential).

The Bayesian approach converges much faster than the approaches studied earlier in the paper. This is perhaps not too surprising, because the principal now has significant information at the outset about the distribution (namely, the type of distribution). For example, in both cases, if the principal knows the value of the cumulative distribution at any given reward (*i.e.* what the true probability is that an agent will accept that reward), then the principal can immediately infer the entire true distribution.

To see what happens if the principal initially has slightly less information about the distribution of agents' types, we ran the same experiments again with the following modification: all the principal knows initially is that the distribution is uniform *or* exponential—she does not know which of the two is true. She believes that both types of distribution are equally likely, and given the type of distribution, that each parameter value in $\{0.00, 0.01, \ldots, 10.00\}$ is equally likely. The results are in Figures 10, 11, and 12.

The Bayesian approach still converges fast, but it no longer converges to the optimal reward (for the uniform distribution, it converges to 0.45 instead of 0.5, and for the exponential distribution, it converges to 0.48 instead of 0.44). Moreover, while the approach is still able to eliminate most parameter values, the principal is unable to distinguish

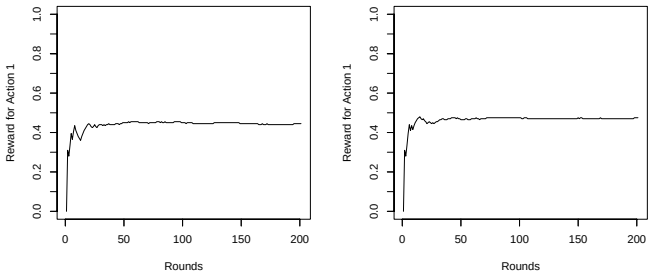

Figure 10. Convergence when the true distribution is uniform (left) and exponential (right) under the Bayesian approach, when the prior places probability on both types of distribution.

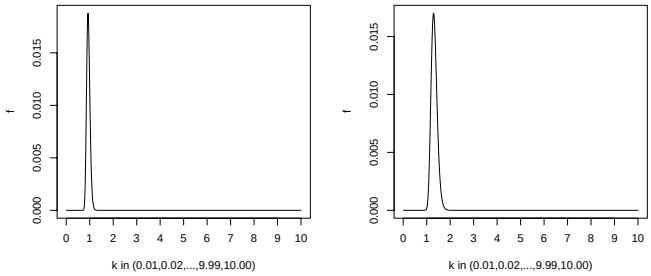

Figure 11. Final principal beliefs for values of k (the uniform distribution parameter) when the true distribution is uniform (left) and exponential (right) under the Bayesian approach, when the prior places probability on both types of distribution.

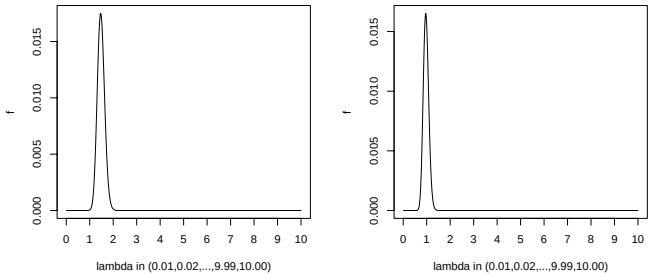

Figure 12. Final principal beliefs for values of λ (the exponential distribution parameter) when the true distribution is uniform (left) and exponential (right) under the Bayesian approach, when the prior places probability on both types of distribution.

whether the distribution is uniform or exponential. This can be explained as follows. The principal ends up placing all the probability mass on or close to one value for λ, and one value for k, such that the optimal reward to set under this belief has the following property: the probability that the agent will accept given the value for λ is the same as the probability that the agent will accept given the value for k. Therefore, the agent's decision tells us nothing about the relative probabilities of these two parameter values, and we remain stuck at the same reward value, which is suboptimal with respect to the true parameter value. Put another way, the problem is that the principal's focus on exploita-

tion prevents her from distinguishing type distributions that produce the same behavior under the current optimal reward. It may be possible to force the principal to do some limited exploration to prevent this problem.

7. Conclusions

In a *principal-agent problem*, a principal seeks to motivate an agent to take an action beneficial to the principal, while spending as little as possible on the reward. Typically, this is complicated by the fact that the principal does not know the agent's utility function (or *type*). If no (accurate) distribution of the agent's type is known beforehand, then the principal needs to learn over time how to set the rewards.

We studied the following model. At each round, the principal encounters a new agent, and the principal sets the rewards anew. At the end of each round, the principal only finds out the action that the agent took, but not his type. We showed that this setting generalizes the setting of selling of a digital good online. We first applied a standard bandit algorithm to this setting. Then, we made two assumptions: 1. agents are drawn from a fixed distribution, 2. the principal can add an extra action to the domain that is equally desirable from the principal's viewpoint, but less desirable from the agent's viewpoint. We showed how this allows the principal to do a form of gradient ascent on the reward space. Experimental results show that the gradient ascent approach converges much faster than the general-purpose bandit algorithm (and generates much less variance in the rewards that are set). Finally, we made the assumption that the principal has a prior belief (distribution) over the distribution of agents' types. Under this assumption, we studied a Bayesian approach, under which the principal chooses the reward that maximizes her expected utility (with respect to her belief), and updates her belief based on the action that the agent takes. Experimental results show that the Bayesian approach can converge extremely fast if the prior belief places probability only on a limited class of type distributions, but it can also fail to converge to the optimal reward if the class of type distributions that the prior belief places probability on is larger.

There are numerous avenues for future research, including at least the following questions. Can we make the Bayesian approach converge to the optimal reward if the prior places probability on a large class of varied type distributions? Presumably, this requires forcing the principal to explore other rewards, and there are many conceivable ways of doing so (random exploration, taking future utilities into account, *etc*). Another interesting question is the following: how powerful is the assumption that an additional action is available (which we used for the gradient ascent algorithm)? Prior research has shown lower bounds on the value of knowing the type distribution (for example, those presented in (Kleinberg & Leighton, 2003)), but these may no longer hold given the possibility of an additional action. Another possible line of future research is to extend these results to richer domains with more actions, different reward spaces, multiple agents that act simultaneously, *etc.* (although the model in this paper appears to capture at least the most natural applications).

Acknowledgements

This material is based on work performed while Garera was at CMU. Conitzer is supported by an IBM Ph.D. Fellowship, by the National Science Foundation under ITR grants IIS-0121678 and IIS-0427858, and by a Sloan Fellowship awarded to Tuomas Sandholm. The authors thank Avrim Blum for helpful comments.

References

Auer, P., Cesa-Bianchi, N., Freund, Y., & Schapire, R. E. (1995). Gambling in a rigged casino: The adversarial multi-arm bandit problem. *FOCS* (pp. 322–331).

Babaioff, M., Lavi, R., & Pavlov, E. (2005). Mechanism design for single-value domains. *AAAI* (pp. 241–247).

Bahar, G., & Tennenholtz, M. (2005). Sequential-simultaneous information elicitation in multi-agent systems. *IJCAI* (pp. 923–928).

Bar-Yossef, Z., Hildrum, K., & Wu, F. (2002). Incentive-compatible online auctions for digital goods. *SODA* (pp. 964–970).

Bartal, Y., Gonen, R., & Mura, P. L. (2004). Negotiation-range mechanisms: Exploring the limits of truthful efficient markets. *ACM-EC* (pp. 1–8).

Blum, A., Kumar, V., Rudra, A., & Wu, F. (2003). Online learning in online auctions. *SODA* (pp. 202–204).

Blumberg, A., & Shelat, A. (2004). Searching for stable mechanisms: Automated design for imperfect players. *AAAI* (pp. 8–13).

Cesa-Bianchi, N., Freund, Y., Haussler, D., Helmbold, D. P., Schapire, R. E., & Warmuth, M. K. (1997). How to use expert advice. *Journal of the ACM, 44*, 427–485.

Conitzer, V., & Sandholm, T. (2004). Self-interested automated mechanism design and implications for optimal combinatorial auctions. *ACM-EC* (pp. 132–141).

de Farias, D. P., & Megiddo, N. (2003). How to combine expert (or novice) advice when actions impact the environment? *NIPS*.

Kleinberg, R., & Leighton, T. (2003). The value of knowing a demand curve: Bounds on regret for on-line posted-price auctions. *FOCS* (pp. 594–605).

Mas-Colell, A., Whinston, M., & Green, J. R. (1995). *Microeconomic theory*. Oxford University Press.

Parkes, D., & Schoenebeck, G. (2004). GROWRANGE: Anytime VCG-based mechanisms. *AAAI* (pp. 34–41).

Porter, R. (2004). Mechanism design for online real-time scheduling. *ACM-EC* (pp. 61–70).

Smorodinsky, R., & Tennenholtz, M. (2004). Sequential information elicitation in multi-agent systems. *UAI* (pp. 528–535).

Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. *ICML* (pp. 928–936).