Incentive Design for Adaptive Agents

A thesis presented

by

Jerry L. Kung

to

Applied Mathematics

in partial fulfillment of the honors requirements
for the degree of
Bachelor of Arts
Harvard University
Cambridge, Massachusetts

April 1, 2011

Acknowledgments

A big thank you to David Parkes for being an unparalleled mentor. Your genuine dedication and care have been the impetus for my development as a researcher and a person. Your unfailing ability to identify the next plan of attack for a seemingly unsolvable problem will never cease to inspire me. Equal gratitude must be given to Haoqi Zhang for seeing me through these past two years. Thank you for hearing out all of my ideas, for answering all of my questions, and, most of all, for sharing your passion for research with me. Without the both of you, this thesis would certainly not have been possible.

To the members of the EconCS group, especially Yiling Chen and Ariel Procaccia: thank you for all of the helpful conversations, the support, and the camaraderie.

To Terry Ding: thank you for reading drafts of this thesis and for braving through this process with me.

Thank you to John and Wendy for all of the inspiration and confidence that you have given me. You've really made Cambridge my second home! To James and Wanyi (and Gus!), thanks for always reminding me to have fun during my studies. Lastly, thank you to Mom and Dad for your unrelenting love, for putting everything in perspective, and for encouraging me to pursue my dreams.

Incentive Design for Adaptive Agents

Abstract

We consider a setting in which a principal seeks to induce an adaptive agent to select a target action by providing incentives on one or more actions. The agent maintains a belief about the value for each action—which may update based on experience—and selects at each time step the action with the maximal sum of value and associated incentive. The principal observes the agent's selection, but has no information about the agent's current beliefs or belief update process. For inducing the target action as soon as possible, or as often as possible over a fixed time period, it is optimal for a principal with a per-period budget to assign the budget to the target action and wait for the agent to want to make that choice. But with an across-period budget, no algorithm can provide good performance on all instances without knowledge of the agent's update process, except in the particular case in which the goal is to induce the agent to select the target action once. We demonstrate ways to overcome this strong negative result with knowledge about the agent's beliefs, by providing a tractable algorithm for solving the offline problem when the principal has perfect knowledge, and an analytical solution for an instance of the problem in which partial knowledge is available. We also consider settings in between these two extrema in which the principal has some knowledge about the agent's beliefs. We study the problem of how incentives should be provided in these scenarios and the study the empirical performance of a class of algorithms that we devise.

Contents

Chapter 1

Introduction

Many situations arise in which a principal takes an interest in the actions taken by an agent who is learning how to make decisions. As the agent selects a particular action and experiences the realized outcome of that action, he updates his belief about the value associated with that action. Taking these changes into account, the principal wishes to influence the agent to perform certain desirable actions, with the goal of getting the agent to converge to those actions. It is often impossible, however, for a principal to directly force an agent to take a desirable action; instead, it is restricted to the use of incentives.

Consider bottle recycling. The government wants individuals to recycle bottles made of glass and plastic. An individual who is new to the concept of recycling deems the effort of determining the appropriate bin for a recyclable bottle to be too high, so he decides not to bother. In the absence of incentives, the individual would never learn to recycle. However, recycling is a task with high initial cost that decreases through experience. If the government were to provide the individual with incentives for recycling, perhaps through a small reimbursement per bottle to compensate him for the effort spent, then he might give recycling a chance. His cost for recycling would decrease over time, and, eventually, the presence of incentives may not even be necessary to induce the individual to recycle.

Similar examples can be found in a variety of domains. The government wants

individuals to undergo influenza vaccinations. A firm wants consumers to purchase its products. The teaching staff of a computer science course wants students to comment their code. In each of these scenarios, the agent is actively updating his beliefs regarding the utility of certain actions in accordance with realized outcomes. With the proper incentive scheme, the principal may be able to induce more desirable outcomes in both the short term and the long term. Individuals may find that annual flu vaccinations are worth the potential costs. A consumer may sample a particular brand and end up preferring it in the long run. Students may find that commenting their code makes their lives easier. In these cases, incentives may be useful to encourage the agent to explore the values associated with particular actions. While the precise end result will be dependent on a combination of the realized outcomes and the properties of the agent's belief update function, is there anything the principal can do to influence the agent's actions in beneficial ways?

A fundamental difficulty in this problem setting is posed by the agent's preferences. We encounter this difficulty in both the case of automated and human agents. For automated agents, it is costly to completely specify their underlying preference functions; for human agents, it may be impossible to express these preferences at all. For example, it is difficult for an agent to express how the realized outcome resulting from performing a specific action changes his beliefs. Even if these preferences could be expressed, direct questioning may be necessary to obtain this information. This would be invasive of the agent's privacy. Because of the difficulty and cost associated with obtaining the agent's preferences, the principal must make do without this information.

Given these constraints, we consider the principal's **incentive design** problem, which is specified as follows: given only observations about an agent's chosen action during each round, how should limited incentives be provided such that the agent chooses actions desirable to the principal?

1.1 Contributions

In this thesis, we introduce and develop a model that facilitates analysis of the incentive design problem faced by the principal when dealing with adaptive agents. We provide a discussion of aspects of this model and consider the tradeoff between modeling reality and ease of analysis. Using our generalized model for the principal's problem, we first develop an algorithm that is based on the strategy of only incentivizing desired actions. We prove the optimality of this algorithm when information about the agent's update function is known and in a special case when the agent's update function is unknown. We then encounter a result that informs us that, in general, no design strategy can, in the absence of information about the agent's update function, perform within a factor of ϵ of the optimal offline algorithm. From that point on, we consider ways around this impossibility result by assuming that the principal has knowledge about the agent's parameters.

We consider several examples motivated by real life scenarios in which the principal has knowledge about the agent's parameters, as well as the underlying distribution behind each of the actions. Importantly, we maintain some uncertainty in this problem by assuming that the principal cannot observe the realized outcomes experienced by the agent. We provide analytical solutions to the one round and two round cases when the underlying distribution is continuous. For the three round case, we perform simulations to approximate the optimal incentive design strategy. We reason through the actions taken by the simulated optimal solution to better understand the factors at play in our new model. We also formulate a new objective criterion for the principal that helps to express the tradeoff between providing incentives and obtaining selections of the desired action.

To gain traction on the incentive design problem with known agent parameters, we model a setting in which each of the arms has an underlying distribution that is discrete. We make several observations that help to simplify the principal's problem by allowing for tractable representations of all elements of the model. These

simplifying observations prompt us to design a series of algorithms that we believe would work well on the principal's incentive design problem. We perform simulations of these algorithms and analyze the results to better understand scenarios in which sophisticated incentive design strategies are effective. Furthermore, these simulation results allow us to begin comprehending what constitutes good principles of incentive design.

1.2 Related Work

The inspiration for the incentive design problem stems from the **environment design** and **policy teaching** frameworks proposed by Zhang et al. [17–19]. In the environment design problem, an interested party observes the properties of an agent through the agent's repeated interaction with different environment configurations. Over time, the interested party can use these observations to pare down the agent's space of preferences. Once sufficient knowledge about the agent's preferences is elicited, the interested party can design an environment in which the agent performs desirable actions.

The goal of policy teaching is to modify the actions of an agent that is modeled by a Markov Decision Process (MDP). This is done by systematically learning about the agent's reward function. Once the reward function is elicited, the interested party can influence the policy that the agent eventually converges to by perturbing the agent's MDP via the use of external incentives applied to a variety of states.

In both the environment design and policy teaching settings, the agent is assumed to be static. Preferences do not change, and the mapping from preferences to actions are not influenced by external factors such as interacting with changing environments or additional reward associated with states. Our work aims to tie together the goals of these two frameworks while at the same time relaxing their assumptions. This allows us to extend the applicability of our work to human agents, who adapt their preferences as they experience the different realized outcomes of their actions. At

the same time, we maintain the goal of inducing the agent to select desirable actions. This new problem, in which the principal must decide the best way to induce desired behaviors in an adaptive agent, is an instance of **online environment design**. As an agent learns more about the environment, his beliefs about the values associated with certain actions change, and the principal must take this into account.

We draw our model of the agent's interactions with the environment from the **multi-armed bandits** problem [10, 14]. In the classical version of this problem, an agent is faced with a variety of levers, or arms, each with an underlying distribution. At each round, the agent pulls one of these arms and receives a realization of that arm's underlying distribution. The agent is given a total number of pulls and is asked to maximize his payoff at the end of these pulls. One way for an agent to solve this is by computing Gittins indices to keep track of the value associated with each arm [9]. By updating the index of an arm after obtaining a realization of that arm and using the indices to help make his decision at each step, the agent can work toward maximizing his eventual payoff.

While the multi-armed bandit problem has been extensively studied, our work is novel in that it considers a party that is concerned with using external incentives to influence the actions of an agent faced with a multi-armed bandit problem. The closest work involving incentives in multi-armed bandits is studied by Babaioff et al. [1], Bergemann and Välimäki [3], and Cavallo et al. [7], who look at the setting where the center is faced with a multi-armed bandits problem, and multiple selfish agents have private information about the underlying distribution associated with each arm. They consider the problem of designing a mechanism that can truthfully elicit the agents' private information so that the center can maximize his payoff. Our work shifts the focus from the agent solving the multi-armed bandit problem to a principal who is concerned about the specific arms pulled by the agent. Although we can still think about our agent as solving a multi-armed bandit problem, we solve the problem of how to influence the agent to choose desirable actions along the way.

Some more related work that is modeled using a multi-armed bandit problem has been studied in the context of ad-hoc teams by Stone et al. [15, 16]. In this setting, there are two agents, a learner and a teacher. The learner makes his decisions by selecting the arm with the highest empirical mean based on the realized outcomes of both the learner and the teacher. The teacher has access to the underlying distributions behind each arm, and must take actions to maximize the sum of their payoffs. Stone et al. show that it is never optimal for the teacher to select the lowest-paying arm. In a sense, this is similar to some of our results where the principal would only ever want to incentivize the desired action. This logic can be further thought of as follows: there is never a need to demonstrate how bad an action is as an intermediate step; it is better for the agent to obtain this information of his own accord.

Although these results are of a similar flavor, this problem setting differs from our own in that they do not consider the use of incentives in the teaching and learning process. Instead, the learner is only given access to more observations. This does not mean that the teacher can provide information to the learner without cost, however. The teacher faces the tradeoff between always exploiting the highest-paying arm, which they assume the learner is restricted from choosing, and guiding the learner to take more favorable actions while obtaining lower payoff. Additionally, the work on ad-hoc teams assumes that the teacher and the learner are working toward a common goal. In our setting, the principal may not share the same objective as the agent. Furthermore, the agent's preferences and update function may not be provided to the principal.

Like Stone et al., Brafman and Tennenholtz [6] consider settings in which the actions of other participants in a multi-agent system play a role in influencing the behavior of each agent. They study the extent to which an interested party, who is able to manipulate the behavior of a particular agent, can change the behavior of surrounding players. These works are similar to ours in that they consider the ways in which an agent can perturb the actions of others in an indirect fashion. However,

instead of utilizing incentives, they study the effects of participation in the same environment.

At a high-level, our work also resembles the literature on reward shaping [11, 13]. In the reward shaping setting, a human wishes to improve the performance of an automated agent without explicitly specifying a reward function. The human observes the agent's actions and provides positive or negative feedback to influence the agent toward better performance in the long run. While these goals are similar, we make several different modeling assumptions. We study a general class of adaptive agents of which we know no information, while the works dealing with reward shaping assume that the agent is freely programmable. This excludes human agents from the range of applicability. Furthermore, the reward feedback applied by the human agent comes at no cost, so they can be applied as necessary to encourage the agent to move toward desirable behavior.

1.3 Outline

In Chapter 2, we formally introduce our model and discuss various assumptions that facilitate our analysis. In Chapter 3, we consider the incentive design problem in the absence of knowledge about the agent's update function. We present a design strategy based on only providing incentives to the desired action and prove its optimality when there is a per-period budget. For the case in which the principal has a total budget across time, we encounter an impossibility result that forces us to reconsider our assumptions. In Chapter 4, we allow the principal to have knowledge about the agent's update and valuation functions. This problem is still difficult for the principal because he is uncertain about the outcomes that the agent experiences. We begin the chapter by considering arms with continuous distributions and move toward a framework in which arms have discrete distributions. We present and analyze simulation results in the discrete setting. Chapter 7 concludes.

Chapter 2

Preliminaries

In this section, we introduce the aspects of our model and the notational conventions that will be used in subsequent chapters. We illustrate each of the parts of our model using a motivating example and discuss some of the assumptions that we make. Finally, we pose our overarching problem of interest.

2.1 Model

Our model aims to capture the most salient aspects of the problem where a principal would like to influence the actions of an agent making decisions. The agent adapts as he makes decisions; after selecting an action, he gains experience with that action and updates his values accordingly. The principal wants the agent to select a specific action and can act by providing incentives to any subset of the actions, but is limited by a budget constraint. We wish to answer the following question: is there a systematic way in which the principal can assign these incentives so as to maximize the number of times the agent selects the desired action?

2.1.1 Aspects of the Model

We begin with a high-level description of the interactions that are a part of the incentive design problems that we consider in subsequent chapters.

Our inspiration stems from the multi-armed bandit problem [14], in which an agent is faced with a set of arms, each with an underlying value distribution. At each round, the agent selects an arm and observes a realization of the distribution associated with that arm. The agent then uses this information to reformulate his beliefs about that arm. Given only this information about the arms, the agent faces the problem of how to select arms to maximize his payoff.

We use the multi-armed bandit problem to help us formulate the properties of the **agent** and his **decision environment**. The set of arms, each with an underlying value distribution, can be thought of as the decision environment of the agent. We assume that the agent acts as if he is solving a multi-armed bandit problem. Like in the classical version of the multi-armed bandit problem, the agent's beliefs about a specific arm depend only on his previous experiences with that arm.

We now introduce the **principal** to this setting. The principal can interact with the agent by providing incentives on some subset of arms. However, he is limited in that he can only observe the agent's selection, and not the realized outcome or its effect on the agent's beliefs. Furthermore, the principal is limited in the amount of incentives that can be provided. We study the question of how the principal should provide incentives so as to maximize the number of selections of the target arm.

We now delve into each of these aspects of our model.

Properties of the Decision Environment:

- *Current round, t*

  We use $t \in \{1, 2, \dots\}$ to denote the round number. At the beginning of the agent's interaction with the environment, $t = 1$.

- *Arms, or actions, available to the agent, K*

  Suppose there are a total of n arms. $K = \{1, 2, \dots, n\}$ denotes the set of arms available to the agent in the environment. K_{-i} denotes $K \setminus \{i\}$. We assume that all actions are available in all rounds, although this may not be the case

in general.

- *Distribution behind each arm*, O_i^t

  If the agent selects arm $i \in K$ at round t, we can think of the outcome as a realization of a real-valued random variable O_i^t. If the random variable is the same for all rounds, we simply denote it as O_i. Let $o_i^t \in \mathbb{R}$ denote the realized outcome when arm i is selected at round t.

Properties of the Agent:

- *Belief state over all available arms*, $x(t)$

  Let X denote the set of all possible belief states for an individual arm.[1] We denote by $x_i(t) \in X$ the belief state of arm i at round t. $x(t) \in X^n$ denotes the aggregate of the agent's belief states at time t. It can be viewed as a vector of $x_i(t)$ for all $i \in K$.

  To translate from a belief state to a utility value, we use the function $v_i : X \to \mathbb{R}$, which is defined for each $i \in K$. Thus, $v_i(x_i(t))$ corresponds to the agent's perceived value of arm i at round t. This valuation function allows our framework to work with a general class of adaptive agents. For instance, an agent's valuation of an arm may include some noise that is proportional to his uncertainty in that arm. This would promote exploration of arms with which the agent has little experience. In general, any algorithm for the multi-armed bandit problem can be modeled by appropriately specifying these valuation functions v_i.

- *Agent function: how decisions are made at each time step*, $\mathcal{F}^t$

  The agent function takes a vector of belief states and a vector of real-valued incentives provided by the principal, and outputs the action that the agent will select during round t. We denote the agent function at round t by $\mathcal{F}^t$:

[1] We abstract away the precise details of what is included in a belief state. In general, it can be treated as a set of data that represents the agent's beliefs of the distribution behind a specific arm. If viewed in this way, we can simply treat it as a record of all realized outcomes from a specific arm.

$X^n \times \mathbb{R}^n \to K$. In general, the agent function may change between rounds or may be stochastic.

- *Belief update function: how realized outcomes are incorporated into beliefs,* $\mathcal{U}^t$
  The agent's belief update function at round t, which we denote by $\mathcal{U}^t : X^n \times K \times \mathbb{R} \times \mathbb{R}^n \to X^n$, corresponds to the way in which the agent updates his belief states at a given time step. The belief update function takes four inputs: the current vector of belief states, a specific arm, the realized outcome associated with selecting that arm, and the incentives that the principal provided at that round; it outputs an updated vector of belief states.

  Unless otherwise noted, we assume that if the agent selects arm i at round t, then $x_j(t+1) = x_j(t), \forall j \in K_{-i}$. In other words, the belief states of all other arms stay the same. This corresponds to the assumption in the multi-armed bandits setting that the agent only updates his beliefs about an arm that was chosen, and his beliefs for the remaining arms stay the same.

Properties of the Principal:

- *Incentives provided at each round,* $\Delta(t)$
  We denote by $\Delta_i(t)$ the amount of incentive that the principal places on arm i at time t for all $i \in K$. We let $\Delta(t) = (\Delta_1(t), \Delta_2(t), \ldots, \Delta_n(t))$ denote the vector of all incentives provided at time t. Thus, $\Delta = (\Delta(1), \Delta(2), \ldots, \Delta(t), \ldots)$ captures the sequence of all incentives provided to each arm during all time periods. We call Δ the principal's **sequence of interventions**. We restrict all values within Δ to be non-negative.

- *Budgetary limitations,* B
  We consider two kinds of budgetary limitations: a per-round budget and a total budget across all rounds. For the former, the principal is allowed to pay out at most B worth of incentives in round t. For the latter, the principal is given a total budget $\overline{B}$ that he can pay out across all rounds of interaction.

In both cases, we assume that, at each round, the principal must only pay out the incentives placed on the action ultimately chosen by the agent. Thus, we think of incentives as representing contractual agreements with the agent (e.g., I will subsidize your purchase of this item), as opposed to sunk costs (e.g., I advertise the benefits of this particular item).

For the per-round budget, this translates to the following constraint: $\max_{i \in K} \Delta_i(t) \leq B$. Notice that this allows us to simultaneously place up to B worth of incentive on every arm. For the total budget across rounds, the following constraint applies: $\sum_t \Delta_{s_t}(t) \leq B$, where s_t is the action selected at round t.

- *Desired arm, g*

  We denote by $g \in K$ the principal's desired, or target, arm. In general, the agent may desire elements of any proper subset of K to be chosen, but we restrict our analysis to the case in which the principal has only one target arm unless otherwise specified. The principal's goal is to design a sequence of interventions that will maximize the number of times the agent pulls arm g.

Figure 2.1 illustrates how the various aspects of the model interact with each other at round t. Properties of the agent are blue, properties of the principal are green, and properties of the decision environment are red.

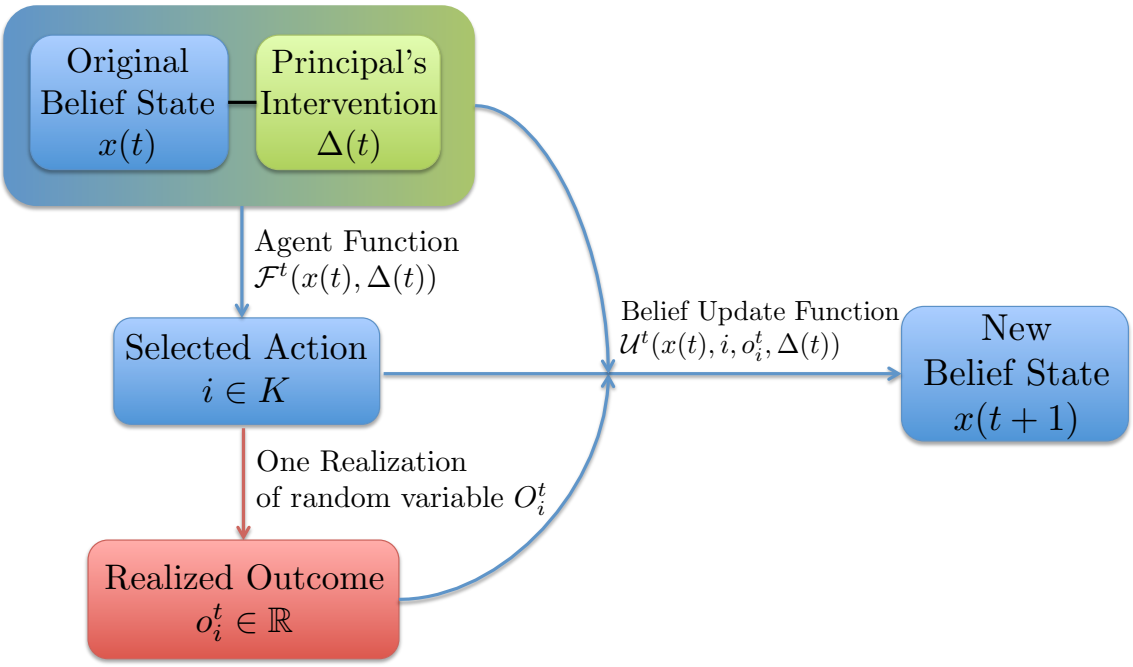

Figure 2.1: Principal and agent interaction at round t

2.1.2 Example: Encouraging Influenza Vaccinations

Now that we have defined our model, we demonstrate how each aspect maps to a real-life situation. We consider the case where the government (the principal) wishes for an individual (the agent) to obtain an influenza vaccination each year in order to decrease the likelihood of an epidemic. Each year, the individual must decide whether or not to get a flu shot.

The individual has initial beliefs over each of these two actions, and the government must decide how to provide the individual with incentives subject to the budget pertaining to that year. Based on his initial beliefs and the provided incentives, the individual decides which action to take during this round. Either action, whether it be getting or eschewing the flu vaccine, has an underlying distribution and provides the agent with a realized outcome. The individual then takes his initial beliefs, the action that he selected along with its realized outcome, and the government's incen-

tives to formulate new beliefs about each of the actions available to him. This process continues in this fashion.

In this scenario, the government must decide what Δ to provide so as to maximize the number of times the individual obtains the flu vaccine.

We can summarize the above model as follows:

Properties of the Environment:

- *Current round*, t

  Each year, the government must decide how to provide incentives to an individual, so this quantity denotes the current year of interaction.

- *Arms, or actions, available to the agent*, K

  At each year, the individual can decide whether or not to get the flu vaccine. Thus, $K = \{1, 2\}$. We use 1 to denote 'deny vaccine', and 2 to denote 'get vaccine'.

- *Distribution behind each arm*, O_i^t

  For the first year, we may want to assume that the random variables look like the following:

$$O_1^1 = \begin{cases} -10 & \text{with probability } .5 \\ 0 & \text{with probability } .5 \end{cases} \qquad O_2^1 = \begin{cases} -15 & \text{with probability } .1 \\ -5 & \text{with probability } .9 \end{cases}$$

  For each of these random variables, the outcomes with lower valuation correspond to scenarios in which the individual ends up getting the flu. Notice that the probability of getting sick is lower when the individual decides to get the shot than when he decides not to. Additionally, the valuations for action 2 are 5 units lower than the valuations for action 1 because this represents the physical and monetary hardship caused by the action of obtaining the shot. For future rounds, we may want to think about the hardship caused by obtaining the shot as decreasing with the number of times the agent chooses action 2.

Furthermore, the probability of getting sick with the flu and the severity of the illness may vary from year to year.

Properties of the Individual (Agent):

- *Belief state over all available arms,* $x(t)$

  Perhaps at the first year, the individual does not like getting vaccinations, so he overestimates the monetary and physical hardship that will result from the vaccination. Thus, $v_2(x_2(1)) < v_1(x_1(1))$. These beliefs are updated from round to round via the belief update function.

- *Agent function: how decisions are made at each time step,* $\mathcal{F}^t$

  At round t, the individual takes his belief state $x(t)$ of all of the actions and the government's provided incentive $\Delta(t)$ and decides how to act. If we assume that the individual is myopic with respect to future incentives (i.e., he does not reason about what the principal will do in the future), the agent function would look like the following:

  $$\mathcal{F}^t(x(t), \Delta(t)) \in \arg\max_{i \in K} [v_i(x_i(t)) + \Delta_i(t)]$$

  In general, the agent function may utilize more sophisticated reasoning to determine how the individual should act. For example, the individual may intentionally forgo the vaccine if he believes that the government will become desperate after observing this behavior for several rounds and provide an abundance of incentives.[2]

- *Belief update function: how realized outcomes are incorporated into beliefs,* $\mathcal{U}^t$

  In general, after selecting an action i at round t, the individual receives a realization o_i^t of the random variable O_i^t. He uses the action i that he chose, the realization o_i^t, and $\Delta(t)$ to formulate his new beliefs $x(t+1)$ based on his original

[2]Note that depending on how v_i is defined for all $i \in K$, the above agent function does not preclude the individual from utilizing any algorithm based on the exploration-exploitation tradeoff. In particular, Gittins index policies [9] are still applicable.

beliefs $x(t)$. If we assume that the individual takes the empirical mean of all realized outcomes of an action as his belief about the value of that action, then we can say that $\mathcal{U}^t$ acts as follows:

$$x(t+1) = \mathcal{U}^t(x(t), i, o_i^t, \Delta(t)) \text{ such that } v_i(x_i(t+1)) = \frac{\sum_{j=1}^{k} r_i^j}{k}$$

where k is the number of times that action i has been selected before, and r_i^j is the realized outcome of action i the j^{th} time it was selected. Note that all of this information can be encapsulated in the belief state $x_i(t+1)$.

Properties of the Government (Principal):

- *Incentives provided at each round*, $\Delta(t)$

  At each year, the government must decide which actions to incentivize and by how much. For instance, if the government subsidizes flu vaccinations at time t, then one possible way to denote this would be $\Delta(t) = (0, 5)$.

- *Budgetary limitations*, B

  At the beginning of each fiscal year, the government sets a budget B for incentivizing the individual to get the flu vaccine. Thus $\max(\Delta_1(t), \Delta_2(t)) \leq B$ for all rounds t. Unused budget does not have any value to government officials and does not carry over into future years.

- *Set of desired arms*, g

  The government wants the individual to get the flu vaccine, so $g = 2$.

2.2 Assumptions

Although the model that we introduce in section 2.1.1 strives to be as general as possible, in moving forward we make the following assumptions to help us gain traction on the principal's incentive design problem.

- **The agent is myopic with respect to future incentives.**

  We assume that the agent is myopic insofar as he does not reason about incentives provided at future rounds. This precludes scenarios in which the agent discovers the principal's desired action and design strategy and intentionally manipulates his choices to exploit this information. This assumption does not mean that the agent is myopic with regard to his utility at each round. In particular, we make no assumptions about the agent's valuation functions v_i, which could be defined to encapsulate exploration-exploitation tradeoffs as well as Gittins index policies [9].

- **The agent function is fixed and known to the principal**

  We assume that $\mathcal{F}^t = \mathcal{F}$ for all t. We also assume that the agent treats the principal's incentives to a certain arm as being additive to the utility valuation of that arm. We further assume that the agent selects the action which maximizes his value over the next round; combined with the assumption that the agent is myopic, this gives us the following agent function:

  $$\mathcal{F}(x(t), \Delta(t)) \in \arg\max_{i \in K} [v_i(x_i(t)) + \Delta_i(t)]$$

  We assume that the principal has knowledge of this agent function.[3] However, it is important to note that this does not give the principal full knowledge about the agent. In particular, the principal does not know the agent's valuations of the arms, nor does he know the agent's belief update function.

- **The agent views incentives as being extrinsic to the arms.**

  We assume that the agent observes the incentives as separate from the arms. That is, the agent does not include the incentives provided by the principal in its update function. Thus we can write the agent's belief update function as $\mathcal{U}^t : X^n \times K \times \mathbb{R} \to X^n$, and we no longer need to view the incentive as an input

[3] We assume that the agent breaks ties in favor of g when g is included in the candidate set of actions. This assumption stems from the fact that the principal can always provide some additional ϵ of incentive to g to tip the scales in its favor.

to the agent's belief update function. In other words, if the agent selects arm i at round t, $x(t+1)$ will always be the same, provided we fix the realized outcome from the random variable O_i^t. It will not depend on $\Delta(t)$. This assumption precludes the possibility of the agent internalizing the provided incentives.

- **The agent's belief update function is stationary.**

  We assume that $\mathcal{U}^t = \mathcal{U}$ for all t. This allows us to say that the agent's belief update is independent of the total number of rounds passed. Along with the previous assumption of incentives being external to the agent's valuation update and the independence of the arms that we inherit from the multi-armed bandits setting (i.e., a selection on arm i does not affect the belief state on any arm in K_{-i}), this assumption tells us that the agent's valuation for an arm i depends only on the number of times that it has been chosen before, along with the sequence of realizations of that arm. It does not depend on selections of arms in K_{-i}.

- **The underlying value distribution for each arm is static.**

  We assume that $O_i^t = O_i$ for all t. This allows us to equate pulls of arm i that were performed during different time periods. Thus, we can fully specify an agent's valuation for an arm based on the number of times that arm was chosen before. For ease of notation, let x_i^k denote the agent's belief state for arm i after it has been selected k times. Let $v_i(x_i^k)$ denote the agent's belief about the value of arm i after it has been selected k times.

- **Incentives over each of the arms are equally costly.**

  We assume that it is equally costly for the principal to provide 1 unit of incentive to every arm. This assumption is not strictly necessary for our results to hold, since a simple rescaling of the units on the agent's valuation for each arm can yield the same result. It is simply for ease of exposition and analysis.

These assumptions still allow us to capture much of the richness in the principal's incentive design problem. In particular, it still allows us to deal with a general class of adaptive agents. This is because aside from our assumptions that the agent's belief update function is stationary and does not take the principal's incentives into account, we make no claims about exactly what the function looks like. Moreover, we make no assumptions about the agent's v function, which gives the agent a valuation over each of the arms given a belief state. However, since our restrictions on $\mathcal{U}$ make it such that the state of an arm only depends on how many times that arm has been chosen before, this necessarily means that $v_i(x_i^k)$ depends only on how many times arm i has been selected before.

Thus, even with our modeling assumptions, our framework is still capable of addressing the principal's incentive design problem for adaptive agents. Of special interest is that our model is capable of dealing with agents utilizing Gittins indices [9] to make decisions at each round, as well as any agents following any algorithms that consider the tradeoff between exploration and exploitation.

2.3 Problem Specification

In this section, we pose two different models of computation to which we will repeatedly refer in future chapters.

2.3.1 Online Model

In the online model [4], we assume that the principal knows none of the agent's valuations in advance. As the principal interacts with the agent, he can only observe which action the agent selects as it is happening. The principal cannot go back and change his decision based on what was observed. We thus phrase the **online** version of the principal's incentive design problem as follows: given only observations about the agent's selections as they happen, what is the best way to assign Δ such that g is chosen as many times as possible? To begin our analysis of the online problem, we

first wish to identify an upper bound. We thus refer to an offline model.

2.3.2 Offline Model

In the offline model, we assume that the sequence of all of the agent's valuations are known in advance. For instance, if the principal interacts with the agent for m rounds, then $v_i(x_i^j)$ is known to the principal for all values of $j \in \{0, \ldots, m-1\}$ for all $i \in K$. The **offline** version of the principal's incentive design problem is now phrased as follows: given information about the sequence of the agent's valuations for all arms and a total number of rounds, what is the best way to assign Δ such that g is chosen as many times as possible? By identifying an efficient offline algorithm, we can provide an upper bound on the performance of any online algorithm. Furthermore, the design motivations of our offline algorithms fuel our understanding of the online model. Ultimately, we seek to determine whether or not we can develop algorithms for the online model that are competitive with optimal offline algorithms.

Chapter 3

Incentive Design Without Knowledge of Agent Parameters

In this chapter, we begin with a motivating example that demonstrates the richness of the model introduced in Chapter 2 even after all of our assumptions are factored in. We then state the incentive design problem for a principal with a per-period budget. Using intuition obtained while trying to solve this problem, we consider a simple design strategy that only provides incentives to the desired arm. We prove that incentive schemes based on this design strategy are optimal when the principal has a per-period budget. After that, we consider the same problem for a principal with a total budget across time. We conclude this chapter with a discussion about the implications of our results.

3.1 Example: Promoting Cereal

3.1.1 No Principal

We first look at an example in which there is no principal present. Thus, we can think of the agent as solving a multi-armed bandit problem to maximize total payoff.

After moving to a new town, Alice is faced with a variety of new choices. One

[0]Many of the results in this chapter appear in the following: Y. Chen, J. Kung, D. Parkes, A. Procaccia, H. Zhang. Incentive Design for Adaptive Agents. In *Proceedings of the Tenth International Joint Conference on Autonomous Agents and Multiagent Systems*, 2011, forthcoming.

choice that she must make is the type of cereal to consume. The local supermarket carries three types of cereal: Apple Loops, Berry Bites, and Choco Flakes. Alice has never heard of these cereals before, and makes her first decision based on a small sample of each provided by the store.

The first time Alice goes to the supermarket, she decides that Berry Bites would be a good choice. She tries the cereal for a few weeks and decides that she hates it. As a result, she revises her valuation about Berry Bites downward. The next time she goes to the store, she decides to try Apple Loops. This time, Alice finds that she really enjoys eating the cereal. Based on this experience, she formulates a new belief about Apple Loops. For the next few visits to the supermarket, Alice decides to purchase Apple Loops over her other two options, since she believes this will maximize her utility. Over time, her value for Apple Loops converges, and she chooses it forever after.

Figure 3.1 demonstrates one way of explaining Alice's trajectory:

At $t = 1$

$v_i(x_i^k)$	$k = 0$	1	2	3	...
Apple Loops	**4**	(6)	(3)	(10)	(10)
Berry Bites	**5**	(2)	(4)	(6)	(1)
Choco Flakes	**2.5**	(7)	(5)	(3)	(6)

At time $t = 1$, Alice selects Berry Bites because it has the highest perceived value.

At $t = 2$

$v_i(x_i^k)$	$k = 0$	1	2	3	...
Apple Loops	**4**	(6)	(3)	(10)	(10)
Berry Bites	5	**2**	(4)	(6)	(1)
Choco Flakes	**2.5**	(7)	(5)	(3)	(6)

With her new valuation for Berry Bites, Alice selects Apple Loops at $t = 2$.

At $t = 3$

$v_i(x_i^k)$	$k = 0$	1	2	3	...
Apple Loops	4	**6**	(3)	(10)	(10)
Berry Bites	5	**2**	(4)	(6)	(1)
Choco Flakes	**2.5**	(7)	(5)	(3)	(6)

Alice then selects Apple Loops three more times, and her value converges.

At $t \geq 6$

$v_i(x_i^k)$	$k=0$	1	2	3	...
Apple Loops	4	6	3	10	**10**
Berry Bites	5	**2**	(4)	(6)	(1)
Choco Flakes	**2.5**	(7)	(5)	(3)	(6)

She then chooses it in all subsequent rounds.

Figure 3.1: Alice's valuations and selections in the absence of a principal

Notice how the bandit property is applied in this scenario. Because Alice selects Berry Bites at $t = 1$, her valuations for Apple Loops and Choco Flakes carry over into the next round. This is because we assume that the "arms" are orthogonal; information about Berry Bites does not shift Alice's beliefs about the remaining cereals. To illustrate further, given that we have strict uncertainty about the agent's parameters, we are agnostic to how Alice's belief update and valuation functions are specified. This model still holds even if Alice is computing Gittins indices or updating her valuations via Bayesian inference. Along the same lines, we are also agnostic to the realized outcomes that Alice observes.

3.1.2 With Principal

We now add the principal to this problem. The company that produces Choco Flakes would like to increase its market share. It thus sets a weekly budget for the purpose of incentivizing consumers to purchase Choco Flakes. Unused budget does not carry over into future rounds and does not provide any utility to the company. The company thus faces an **incentive design** problem: how should incentives be provided to the three cereals to incentivize Alice to buy Choco Flakes as often as possible?

The company decides to place the maximum allowable amount of incentive per round on Choco Flakes. To do so, they mail Alice a coupon for Choco Flakes each week. Each time she visits the supermarket, Alice takes this coupon into account when deciding which cereal to purchase. We now consider what Alice's decisions would be if she knew about the coupon from the start.

At the first round, Alice decides that the coupon is not enough to convince her to try Choco Flakes. She decides to purchase Berry Bites, which was her choice in the absence of incentives. Like before, her first experience with Berry Bites is a negative one. When she returns to the store, only her value for Berry Bites has changed. During the second round, Alice finds that the coupon is still not enough to induce her to choose Choco Flakes; she decides to try Apple Loops. She has a generally positive experience and decides to give Apple Loops another try in the third round. This time, however, she becomes unsatisfied. The next time she goes to the store, she decides that Choco Flakes, after the coupon is applied, might be a marginally better choice. Upon trying the cereal, she is hooked. She becomes a Choco Flakes customer for life.

Figure 3.2 utilizes the same values from before, except that an incentive is applied to Choco Flakes:

	$v_i(x_i^k) + \Delta_i$	$k = 0$	1	2	3	...
At $t = 1$	Apple Loops	**4**	(6)	(3)	(10)	(10)
	Berry Bites	**5**	(2)	(4)	(6)	(1)
$\Delta_c = 1$	Choco Flakes	**2.5 + 1**	(7 + 1)	(5 + 1)	(3 + 1)	(6 + 1)

At time $t = 1$, Alice selects Berry Bites because it has the highest perceived value.

	$v_i(x_i^k) + \Delta_i$	$k = 0$	1	2	3	...
At $t = 2$	Apple Loops	**4**	(6)	(3)	(10)	(10)
	Berry Bites	5	**2**	(4)	(6)	(1)
$\Delta_c = 1$	Choco Flakes	**2.5 + 1**	(7 + 1)	(5 + 1)	(3 + 1)	(6 + 1)

With her new valuation for Berry Bites, Alice selects Apple Loops at $t = 2$.

	$v_i(x_i^k) + \Delta_i$	$k = 0$	1	2	3	...
At $t = 3$	Apple Loops	4	**6**	(3)	(10)	(10)
	Berry Bites	5	**2**	(4)	(6)	(1)
$\Delta_c = 1$	Choco Flakes	**2.5 + 1**	(7 + 1)	(5 + 1)	(3 + 1)	(6 + 1)

Alice selects Apple Loops again, but she has a bad experience.

At $t = 4$, $\Delta_c = 1$

$v_i(x_i^k) + \Delta_i$	$k = 0$	1	2	3	...
Apple Loops	4	6	**3**	(10)	(10)
Berry Bites	5	**2**	(4)	(6)	(1)
Choco Flakes	**2.5 + 1**	(7 + 1)	(5 + 1)	(3 + 1)	(6 + 1)

Her bad experience induces her to try Choco Flakes.

At $t \geq 8$, $\Delta_c = 1$

$v_i(x_i^k) + \Delta_i$	$k = 0$	1	2	3	...
Apple Loops	4	6	**3**	(10)	(10)
Berry Bites	5	**2**	(4)	(6)	(1)
Choco Flakes	2.5 + 1	7 + 1	5 + 1	3 + 1	**6 + 1**

She is hooked. She purchases Choco Flakes in all subsequent rounds.

Figure 3.2: Alice's valuations and selections when principal provides $\Delta_c = 1$

This example demonstrates the potential of incentives in influencing the agent's actions. In the absence of a principal, Alice converges to purchasing Apple Loops in the long run. When the principal is present, she instead converges to Choco Flakes. This example is also useful because it helps to illustrate and motivate some of our assumptions from the previous chapter.

In this example, we assume that Alice's agent function is fixed and known. For some valuation on each arm, she takes the action that leads to the maximum of the valuation plus any provided incentive. *Note that this does not mean that the principal knows Alice's valuations over each of the arms*; knowledge of the agent function is simply knowledge about how the provided incentives interact with Alice's valuations, whatever they are. The valuation tables here are used to illustrate how Alice makes her decisions. In general, we assume that the principal does not have knowledge of these tables.

Notice that the unrealized valuations (parenthesized in the tables above) all remain the same. This is a result of the assumptions that we made in the previous chapter; the valuations $v_i(x_i^k)$ of an arm i depend only on k, the number of times it has been selected before. The valuations of an arm i do not depend in any way on the sequence

of selections of arms in K_{-i}. Nor do these intrinsic valuations depend on the incentives provided by the principal. Rather, the incentives instantaneously modify the total realized value of Alice for a choice. The motivation is as follows: Alice judges each cereal based on her experiences with the cereal itself. She views the coupon as external to the cereal and does not take into account any earlier coupon when determining her own valuation in future rounds. Nor does Alice reason about how decisions she makes now might affect future incentives offered by the principal.

Now that we have seen the power of incentives, we consider what incentive schemes are effective in inducing the agent to select the target arm in various scenarios.

3.2 Per-round Budget

In this section, we consider two different problems for a principal with a limited per-round budget. In the first problem, the principal would like to induce the agent to select the target arm once. We then consider the case in which the principal would want to maximize the total number of pulls of the target arm in his interactions with the agent.

3.2.1 Induce Target Once

When the goal is to induce the target arm once, the principal would like to determine how to get the first selection of the target arm as soon as possible. We adopt a framework of competitive analysis to study the following problem.

Problem 1 (INDUCE-ONCE)**.** *Given a per-round budget B, how should incentives be provided to minimize the time t at which the agent first selects target arm g? That is, how should Δ be assigned to minimize the time t at which $\mathcal{F}(x(t), \Delta(t)) = g$ for the first time?*

We first note that there are instances in which it may be impossible to induce the target arm g even once. As an example, consider an agent who can select from two

arms, but never updates his valuations. If the agent's value for the target arm is less than his value for the non-target arm by an amount greater than B, then no incentive policy will induce the agent to ever select the target arm. In this case, $t = \infty$, and no incentive scheme that induces the target arm once exists.

We now restrict our analysis to cases in which it is possible for the target arm to be induced once. To begin, let the agent's first selection of g occur at round t. What conditions must be in place for this to be possible? If the agent selects g at round t, then it must be the case that $B \geq \max_{i \in K_{-g}} v_i(x_i(t)) - v_g(x_g^0)$. This is because the principal can simply apply $\Delta_g(t) = B$ and $\Delta_i(t) = 0, \forall i \in K_{-g}$ to induce the agent to select g at round t. Rearranging the terms in our condition, we obtain the following:

$$v_g(x_g^0) + B \geq \max_{i \in K_{-g}} v_i(x_i(t))$$

This allows us to think about the problem in terms of an **inducible threshold**. That is, as long as all non-target arms fall under a certain threshold value, then the principal can induce the agent to select the target arm as its next action. We thus introduce the following definition.

Definition 1. *A threshold T for inducing action g is met at time t if and only if $v_i(x_i(t)) \leq T, \forall i \in K_{-g}$.*

As we shall see, this concept will prove useful in providing intuition and driving results for a variety of problems. By setting our inducible threshold appropriately, we can obtain the strongest lower bound on the number of rounds necessary before our goal is achieved. For Induce-Once, it will be useful to set our threshold as $T_{\text{once}} = v_g(x_g^0) + B$. Now we can say that if T_{once} is first met at time t, then the principal can induce the agent to select g at that round.

From the inducible threshold, we can deduce that it will never be beneficial for the principal to provide incentives to non-target arms. Although initially we might think that the principal should incentivize non-target arms so that their values may drop significantly after being chosen, it turns out that this reasoning does not help

in inducing the target arm any sooner. In fact, while it may be possible to bring the value of a specific non-target arm, say h, below the inducible threshold by inducing it via incentives, we see that such a selection would have been necessary anyway as a means of meeting the threshold. In other words, the action that leads to h falling below the threshold must be taken at some point if the threshold is to be met. Sooner or later, it would have happened, even if the principal provided no incentives to any non-target arms. Thus, the time it takes T_{once} to be met cannot be accelerated via incentives on non-target arms. It may even be the case that poorly applied incentives on non-target arms may delay the time at which the inducible threshold is met.

Consider the following scenario. The principal, unaware of the agent's valuations, decides to provide incentives to a non-target arm h. The agent's value for arm h has already fallen below the inducible threshold; that is, $v_h(x_h(t)) \leq B + v_g(x_g^0)$. Nevertheless, the principal decides to provide $\Delta_h(t) > 0$ such that $\max_{i \in K} v_i(x_i(t)) + \Delta_i(t) = v_h(x_h(t)) + \Delta_h(t)$, which means that h is selected at this round. Notice that this is an unnecessary action because the value for h was already below the inducible threshold. This superfluous selection could have potentially catastrophic effects! For example, it may be the case that the agent's value for h never again drops below the inducible threshold, making it impossible for the principal to ever get the target arm to be chosen.

We formalize the intuitions gained from the inducible threshold using the following lemma, which we will refer to as the **threshold lemma**.

Lemma 1 (Threshold Lemma)**.** *Given a threshold T, let k_i denote the number of pulls of arm necessary to bring a non-target arm i below the threshold T for the first time. More precisely, $k_i = \min\{k : v_i(x_i^k) \leq T\}, \forall i \in K_{-g}$. Assume that all of these k_i exist. Any incentive policy Δ that assigns $\Delta_i(t) = 0$ for all $i \in K_{-g}$ and $\Delta_g(t) \geq 0$ at all times t satisfies the following:*

(a) At any time t before T is met, $x_i(t) = x_i^{m_i}$ such that $m_i \leq k_i, \forall i \in K_{-g}$.

(b) If the T is first met at time t, then $x_i(t) = x_i^{k_i}, \forall i \in K_{-g}$.

Proof. We prove part (a) by contradiction. Assume that at some time t before the threshold is met, there is an arm $i \in K_{-g}$ with $x_i(t) = x_i^{m_i}$ such that $m_i > k_i$. But then this means that i was selected at some time $t' < t$ when $x_i(t') = x_i^{k_i}$. Since the threshold has not yet been met at time t', we know that there exists an arm $j \in K_{-g}$ such that $j \neq i$ and $v_j(x_j(t')) > T$. But by our assumption, then, the following must have been true:

$$v_i(x_i^{k_i}) = v_i(x_i^{k_i}) + \Delta_i(t') > \max_{j \in K_{-g}, j \neq i} v_j(x_j(t')) + \Delta_j(t') = \max_{j \in K_{-g}, j \neq i} v_j(x_j(t')) > T$$

where the first equality is due to $\Delta_i(t) = 0, \forall t, \forall i \in K_{-g}$, the first inequality is due to the selection of i over all other non-target arms, and the final inequality is because the threshold has not yet been met. But then this means that $v_i(x_i^{k_i}) > T$, which is a contradiction. Thus, we can conclude that at all times t before the threshold is met, all arms $i \in K_{-g}$ have $x_i(t) = x_i^{m_i}$ such that $m_i \leq k_i$.

We proceed to part (b). Suppose that the threshold T is first met at some time t. Suppose that an arm, say $i \in K_{-g}$ was selected at time $t-1$. We can say that i was selected exactly $k_i - 1$ times before round $t-1$. Furthermore, we have that for all other non-target arms $j \in K_{-g}, j \neq i$, j was selected at least k_j times by round t. By part (a), we have that all non-target arms were selected at most k_i times by round t. This means that $x_j(t) = x_j^{k_j}$. Adding in the fact that $x_i(t) = x_i^{k_i}$ concludes the proof. □

Notice that the threshold lemma does not assume any particular threshold value. It merely states that given a threshold value, one way to meet that threshold is to only provide incentives to the target arm and never to a non-target arm. By doing so, the agent will only select non-target arms the minimum number of times necessary to meet the threshold. It is also important to note that the threshold lemma does not guarantee that the threshold will be met. Indeed, in some scenarios and threshold values, it may be impossible to satisfy the threshold conditions.

Using the insights from the threshold lemma, we now present a simple incentive design strategy that the principal could use in solving INDUCE-ONCE. As we shall see, its acronym is especially apt.

Definition 2. *The 'only provide to target' (OPT) incentive policy provides* $\Delta_g(t) = B$ *and* $\Delta_i(t) = 0, \forall i \in K_{-g}$ *at each time step* t*.*

Before continuing, we note that OPT has several desirable properties. The principal does not need to know any information about the agent's current values for any of the arms. Similarly, he does not need to know how the values on the arms change. OPT is also very easy for the principal to implement since no computation is required in order to determine his incentive policy. We now demonstrate that OPT solves INDUCE-ONCE.

Theorem 1. *When used in the online model with a per-period budget, OPT always provides performance equivalent to the optimal offline solution to* INDUCE-ONCE.

Proof. Let the threshold T_{once} be set as follows: $T_{\text{once}} = v_g(x_g^0) + B$. As in the threshold lemma, we can define $k_i = \min\{k : v_i(x_i^k) \leq T_{\text{once}}\}$ for all $i \in K_{-g}$. Now we must consider two cases. If at least one k_i does not exist, then OPT cannot get the target arm to be induced. The same is true for any offline algorithm. Now we need only consider the second case: all of the non-target arms drop below the threshold value after some number of pulls.

First consider the performance of the optimal offline algorithm. Before this algorithm can get the target arm to be selected once, all arms $i \in K_{-g}$ must have been selected at least k_i times. If it is optimal, then all arms $i \in K_{-g}$ will have been selected precisely k_i times before the target arm is selected. Thus, the optimal offline solution tells us that $\sum_{i \in K_{-g}} k_i$ rounds must pass before the threshold can be met. After the threshold has been met, an additional round is necessary for the agent to select the target arm after the threshold has been met.

By the threshold lemma, we know that OPT will get the threshold to be met as quickly as possible. In fact, it will meet the threshold at the same time as the optimal offline algorithm. Furthermore, it guarantees that the target arm will be chosen in the round directly after the threshold has been met. Thus, its performance is equivalent to the optimal offline algorithm, as desired. □

This result is inspiring. It tells us that even in the absence of knowledge about the agent's current and future valuations, the principal is still capable of using a computationally simple approach to induce the desired action in as few rounds as possible. Its generality lies at the core of its significance: OPT works for any general class of adaptive agents that satisfy the bandit property (i.e., independence of arms) and are myopic with respect to provided incentives. Moreover, the principal needs none of this information in order to induce the agent to select the desired action as soon as possible. The main takeaway from this analysis is that no incentive policy can get the agent to meet the threshold faster than OPT can. We continue by addressing the case in which the principal would like to induce the target arm more than once.

3.2.2 Induce Target Multiple Times

In many of our motivating examples, the principal would like to get the agent to choose the desired arm as many times as possible within a given time frame. Often this is because the principal only has a limited number of rounds during which it can interact with the agent. For instance, the government decides that it wants to maximize the rate at which an individual recycles over the next decade, but it is restricted in the amount of money it can provide per bottle returned. The teaching staff of a computer science course only has one semester to encourage students to comment their code, but they can only give out one gold star per week. A firm wants to increase sales for the next fiscal year, but its discount campaigns are restricted during each quarter. We formalize this problem for principals who have a per-round budget.

Problem 2 (INDUCE-MULTI)**.** *Given a per-round budget B and a total number of rounds R, how should incentives be provided to maximize m where $x_g(R) = x_g^m$?*

This problem asks us maximize the number of times the target arm is selected by time R. In working toward a solution, we will consider the minimum number of rounds required to induce the target arm m times, for a given m. As we shall see, a solution to this related problem can be converted to a solution for INDUCE-MULTI.

Suppose that $m = 1$. By the threshold lemma, we know that OPT will minimize the time t such that $x_g(t) = x_g^m$. Can we iteratively apply this reasoning to values of $m \geq 2$ to get the desired result? It turns out that we cannot.

We only know that, from any initial belief state that the agent can have, OPT minimizes the amount of time before the desired arm is selected. However, this minimum amount of time depends on the agent's initial belief state. Consider the following scenario. The principal uses OPT and sees the first selection of g at time t_1 and the second selection of g at time t_2. Now, assume the principal uses a different incentive policy Δ'. Assume that this incentive policy induces the first selection of g at time $t_1' > t_1$ and the second selection at time t_2'. Because we have no way of comparing $t_2' - t_1'$ and $t_2 - t_1$, we cannot conclude that $t_2' > t_2$. We only know that Δ' took longer to induce the target arm once. But it may be the case that in taking longer to induce it the first time, it led to a favorable agent belief state that decreased the time between the first and second pulls. We thus require more sophisticated reasoning to show that OPT will minimize the time before m pulls for $m \geq 2$.

Lemma 2. *When used in the online model with a per-period budget, OPT minimizes the time t at which $x_g(t) = x_g^m$, for any $m \geq 2$.*

Proof. We begin by setting the threshold T_{multi} appropriately. We want the threshold value to be the minimum value that the target arm g takes in the process of getting pulled m times. Let $w = \arg\min_{0 \leq \ell < m} v_g(x_g^\ell)$. Set $T_{\text{multi}} = v_g(x_g^w) + B$. As before,

let $k_i = \min\{k : v_i(x_i^k) \leq T_{\text{multi}}\}$ for all $i \in K_{-g}$. We restrict our analysis to the case in which it is possible to induce g at least m times, since the performance of OPT is trivially equivalent to the optimal offline algorithm in the other case.

Consider the performance of the optimal offline algorithm. It is a necessary condition for each arm $i \in K_{-g}$ to be selected at least k_i times before g can be selected a total of m times, based on the way we defined k_i. Thus, the optimal offline algorithm will select each non-target arm i precisely k_i times. This algorithm will spend a total of $\sum_{i \in K_{-g}} k_i + m$ rounds to get the target pulled m times.

We now consider the performance under OPT. By part (b) of the threshold lemma, we know that OPT takes the fewest number of rounds possible before the threshold is met. Assume that the desired arm is selected q times before the threshold is met. This means that it takes $q + \sum_{i \in K_{-g}} k_i$ rounds to meet the threshold. After the threshold has been met, however, OPT ensures that the desired arm will be chosen in each of the next $m - q$ rounds based on the definition of T_{multi}. This means that OPT uses a total of $q + \sum_{i \in K_{-g}} k_i + m - q = \sum_{i \in K_{-g}} + m$ rounds. Thus, its performance is equivalent to that of the optimal offline algorithm. □

Notice how the proper definition of the threshold is the main idea behind this proof. By setting T_{multi} as the minimum of the valuations attained by g before it is selected m times, we can come up with a tight lower bound on the number of selections of non-target arms before g can be selected m times. Once we have this bound, we simply need to take care of a few details to obtain the desired result. The takeaway from this proof is similar to that of Theorem 1: OPT will induce only the requisite number of selections of non-target arms. Once a non-target arm falls below the threshold, it will not be selected again. We are now ready to provide a solution to Induce-Multi.

Theorem 2. *When used in the online model with a per-period budget, OPT always provides performance equivalent to the optimal offline solution to* Induce-Multi.

Proof. Let m denote the number of selections of the target arm that are achieved in R rounds under OPT. Assume for a contradiction that the optimal offline solution Δ' leads to $m' > m$ selections of the target arm in R rounds. But then this means that Δ' achieves m selections of the target arm in less than R rounds. This is a contradiction, since by Lemma 2 we know that OPT induces m selections in the fewest number of rounds possible. Thus, no such Δ' can exist, and so OPT is equivalent to the optimal offline solution. □

It turns out that OPT is quite powerful and can deal with all of our problems of interest when the principal has a per-round budget. An important point to note is that one of the main reasons OPT returns a solution that is equivalent to the optimal offline algorithm even without knowledge of the agent's valuations is because the principal has no utility for leftover budget in rounds. OPT will generally overshoot the required amount of incentives to get g chosen during specific rounds. For instance, if the difference between g and the current highest arm is within the budget, the optimal offline algorithm would only need to provide this difference. OPT has no knowledge of this, but still induces the target arm because it applies more incentives than what is strictly necessary. This intuition will turn out to be important as we move on to consider a total budget across time.

3.3 Total Budget Across Rounds

In this section, we consider a principal who has a total budget across all rounds, and must decide how to allocate incentives such that the target arm g is induced once or multiple times. We can think about our budget constraint when we are given a total budget $\overline{B}$ and incentive policy Δ as follows. Since the principal need not pay out incentives that were applied to arms that were not chosen, we let $i(t)$ denote the arm that the agent selects at time t under incentive policy Δ. Δ is within budget if and only if $\sum_{t=1}^{\infty} \Delta_{i(t)}(t) \leq \overline{B}$. As usual, we assume that all incentives provided are

non-negative.

Is this problem more difficult than the case in which the principal has a per-round budget? It seems so. This is because the principal is hurt when he overshoots an application of an incentive. Furthermore, since the principal has no knowledge of the agent's valuations, he does not know when his budget would be spent most effectively to induce the target arm. For instance, it could be the case that the target arm's value is significantly lower than the highest arm for the first few rounds. After a few pulls of those other arms, the value of the highest arm becomes only slightly higher than that of the target arm. The principal has no idea that this is going on. Compounded with the fact that his incentive policy in earlier rounds affects what he can do in later rounds, this problem becomes more difficult.

It turns out that if the principal's goal is to induce the target arm once, then a simple algorithm is still optimal. For scenarios in which the goal is to induce the target arm multiple times, the outlook is bleak.

3.3.1 Induce Target Once

We study the problem of inducing the target arm once with a total budget $\overline{B}$ across rounds for similar reasons as before. We consider Induce-Once, except this time we assume that the principal is given a total budget $\overline{B}$ across rounds.

Theorem 3. *When used in the online model with a total budget $\overline{B}$ across rounds, OPT always provides performance equivalent to the optimal offline solution to* Induce-Once.

Proof. We first address the case in which the target arm cannot be induced once with the given incentive. Since no algorithm can induce a selection of g, then OPT's performance is trivially equivalent to the offline optimal. We now turn our attention to the case in which the target arm can be induced once.

It is important to remember that the principal needs to pay out incentives from

his budget if and only if he provided incentives to an arm that was chosen. Thus, we can set the threshold $T_{\text{once}} = v_g(x_g^0) + \overline{B}$. If the principal uses OPT by providing $\Delta_g(t) = \overline{B}$ and $\Delta_i(t) = 0, \forall i \in K_{-g}$, then the following occurs. At all rounds before the threshold is met, the agent selects an arm $i \in K_{-g}$. Thus, the principal does not need to pay out any incentive. Immediately after the threshold is met, the principal uses OPT to induce one pull of the target arm. This demonstrates part (b) of the threshold lemma. The optimal offline algorithm can do no better because it is impossible to satisfy the threshold in fewer rounds than OPT. □

With Theorem 3, we see that OPT can induce the target arm once as soon as possible, even when the principal has a total budget across rounds. This can be explained by the fact that the principal does not care if he overshoots the amount of incentive that is necessary to induce g once, remaining budget would be of no use to him. We now consider the case in which the principal would like to induce the target arm multiple times.

3.3.2 Induce Target Multiple Times

Although OPT provides performance equivalent to the optimal offline algorithm when a principal with a total budget would like to induce the target arm once, we shall see that these optimality results do not work for scenarios in which the principal would like to induce the target arm multiple times. We consider Induce-Multi for a principal with a total budget. Given a total number of rounds, how should the principal assign incentives to maximize the number of pulls of the target arm? We begin our analysis with an illustrative example of why OPT fails to work in this setting. Consider the following two cases:

Case 1. Suppose there are two arms: target arm g and non-target arm h. Suppose $v_g(x_g^i) = 1, \forall i \geq 0$ and $v_h(x_h^0) = 2, v_h(x_h^j) = 10, \forall j \geq 1$. The value on g is always 1, and the initial value on h is 2, but then permanently jumps up to 10 after it is selected once. Let $\overline{B} = 1$. In this scenario, the optimal offline algorithm would place

the entire budget on the target arm and get it selected once. Any other incentive policy would fail to ever get the target arm chosen.

Case 2. Now again consider a case with two arms, except this time $v_g(x_g^i) = 1$ and $v_h(x_h^i) = 1 + \epsilon, \forall i \geq 0$. The values on both of the arms are always $\epsilon > 0$ apart and never change. Thus, the optimal offline algorithm would place ϵ worth of incentives on the target arm and achieve $\min(R, \frac{1}{\epsilon})$ pulls. Notice that as the total number of rounds R increases, we can decrease ϵ to further increase the total number of pulls that the optimal offline algorithm achieves.

We see that OPT would match the performance of the optimal offline algorithm in Case 1, but would only get one pull of the target arm in Case 2. This is because the principal must pay out his entire budget just to obtain the first selection of the target arm. After he does this, he cannot induce the target arm again in the remaining rounds. We now use this example to introduce the notion of competitive ratio from the online algorithms literature [4]. We say that an online algorithm is α-competitive if the ratio of the performance of the optimal offline algorithm to the performance of the online algorithm is at most α. To illustrate how this notion is used, we can say that in the case of a per-round budget, OPT is 1-competitive (i.e., performs the same as the optimal offline algorithm) for both Induce-Once and Induce-Multi. In the above cases, we see that OPT is 1-competitive in Case 1, but has no bounded competitive ratio in Case 2. This is because we can always make OPT perform arbitrarily badly (i.e., increase α) on Case 2 by increasing the total number of rounds R and decreasing ϵ appropriately.

The above example is illustrative for all deterministic algorithms for designing incentive policies. We can think about an adversary who, given knowledge that the principal is utilizing a certain deterministic algorithm for designing incentive policies Δ, creates agents with valuation sequences that exploit the weaknesses of Δ. Since the adversary has knowledge of precisely how Δ is assigned by the algorithm, he is able to create instances on which Δ performs arbitrarily badly in a manner similar to

what was done with OPT above. While we do not present a formal proof specifically geared toward deterministic algorithms here, we introduce a result that implies this observation.

We turn our attention to randomized algorithms, or algorithms that harness the power of randomness. It seems that a randomized algorithm may be able to fend off an adversary that is trying to exploit its weaknesses. Before we begin our analysis, we must carefully define how we can measure the performance of a randomized algorithm.

We measure the worst-case performance of a randomized algorithm in the online framework in the following manner. The principal first selects his choice of randomized algorithm. The adversary, observing this choice, presents an instance of an agent to the principal. The principal then applies his randomized algorithm on the given agent instance. *Notice that the adversary does not get to see the realizations of the incentives provided by the randomized algorithm before he creates the agent!* This is important because it draws a distinction between how we measure performance for deterministic and randomized algorithms. Using this framework, we present the following result.

Theorem 4. *In the online model with a total budget across rounds, no randomized algorithm can achieve a bounded competitive ratio for* Induce-Multi*, even if it is given knowledge of the agent's current valuations.*

Proof. We begin with a high level overview of our plan of attack. We use the term **agent input** to signify the sequence of valuations of a particular agent's arms. This is because we can view the randomized algorithm as taking the agent's current valuations as inputs when it decides how much incentive should be provided. We create a series of closely related agent inputs and we derive a series of conditions that must hold if a randomized algorithm with a bounded competitive ratio were to exist. We then use these conditions to demonstrate a contradiction, which implies that no randomized algorithm with a bounded competitive ratio exists for our example, completing the proof.

Assume that the total budget $\overline{B} = 1$. Note that this is without loss of generality as we can simply scale all of the valuations appropriately. Let $k \in \mathbb{N}, \epsilon = \frac{1}{10k}$. Assume that the agent has a choice of two actions: target arm g and non-target arm h. Assume also that the number of rounds is sufficiently large. We now present a series of agent inputs, each of which fully specifies a sequence of values for both g and h. Let $\mathcal{I}_0, \mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_j, \ldots$ denote a series of agent inputs. For all of these agent inputs, $v_g(x_g^t) = 0$ for all t. For any $j \geq 0$, the values $v_h(x_h^0), v_h(x_h^1), \ldots$ specified by $\mathcal{I}_j$ are as follows:

$$1, \epsilon, \epsilon^2, \ldots, \epsilon^j, 2, 2, \ldots, 2, \ldots$$

Thus we see that for any given $\mathcal{I}_j$, after the non-target arm has been selected $j+1$ times, the principal can never induce the target arm to be chosen. We assume that the principal can see the agent's valuations at the current round. Notice that for $p \leq j$, the randomized algorithm cannot differentiate between $\mathcal{I}_p$ and $\mathcal{I}_j$ before h has been selected $p+1$ times. This provides some intuition as to why this example might be difficult for a randomized algorithm; when faced with this problem, the algorithm has the choice of exploiting the current observed value of h or taking a risk by letting h proceed to its next value. It may be the case that the value of h drops significantly, or that it will increase so much as to make any further selections of g impossible.

We adopt the following terminology. Given a run of the randomized algorithm on an input $\mathcal{I}_j$, we use *phase* p to refer to the actions taken by the agent while the value of h is ϵ^p. Note that all of the actions before the last action in a given phase will be selections of g. This is because once h is selected in phase p, its value will become ϵ^{p+1}, thus initiating phase $p+1$. Notice that one pull of g in phase p costs the principal ϵ^p.

Note that in the absence of incentives, the agent will always choose h. Thus, we need not consider incentive strategies that place incentives on h. Taking this into account, let Z_p^j be a random variable that represents the amount of budget spent (paid out, not just applied) at round p on a given input $\mathcal{I}_j$. This random variable

suffices to capture the randomness utilized by the randomized algorithm. We now present the following necessary condition regarding a randomized algorithm that has a bounded competitive ratio.

Lemma 3. *For every $j \in \{0\} \cup \mathbb{N}$ and every $p \in \{0, 1, \ldots, j\}$, if a randomized algorithm has competitive ratio $< k$, then $\mathbb{E}[Z_p^j] \geq \epsilon$*

Proof. Assume for a contradiction that there exists a randomized algorithm with competitive ratio smaller than k such that there exist $j \in \{0\} \cup \mathbb{N}$ and $p \in \{0, 1, \ldots, j\}$ such that $\mathbb{E}[Z_p^j] < \epsilon$. Because we are using an online model, the algorithm cannot differentiate between $\mathcal{I}_p$ and $\mathcal{I}_j$ as noted earlier. Thus, if such j and p exist, we can say that the algorithm $\mathbb{E}[Z_p^j] = \mathbb{E}[Z_p^p]$. In particular, we can now say that $\mathbb{E}[Z_p^p] < \epsilon$ when the randomized algorithm is given agent input $\mathcal{I}_p$. This means that the number of selections in phase p has an upper bound of $\frac{\epsilon}{\epsilon^p} = \epsilon^{1-p} = (10k)^{p-1}$. We now seek to establish an upper bound on the performance of of the randomized algorithm. Since no more selections of the target arm can occur after phase p, we need only an upper bound on the selections in phases 0 through $p-1$. Adding this upper bound to our upper bound on the number of selections of g in phase p will give us an upper bound on the total performance of the algorithm. Because we only desire an upper bound, let us allow the principal to apply a total incentive of 1 at each phase. This yields the following for the summation for the aggregate performance of the algorithm in phases 0 through p:

$$1 + \frac{1}{\epsilon} + \frac{1}{\epsilon^2} + \cdots + \frac{1}{\epsilon^{p-1}} + \frac{1}{\epsilon^{p-1}} = 1 + 10k + (10k)^2 + \ldots (10k)^{p-1} + (10k)^{p-1}$$

If we treat the first p terms of the summation as numbers in base $10k$, then we see that this summation is $\leq 3 \cdot (10k)^{p-1}$, which is our upper bound for the performance of the randomized algorithm.

We now wish to consider the performance of the optimal offline algorithm when given agent input $\mathcal{I}_p$. Because it is cheapest to incentivize the target arm during phase p, the optimal offline algorithm will spend ϵ^p during each round of phase p to

get the target arm chosen a total of $\frac{1}{\epsilon^p} = (10k)^p$ times. Now we can compute the competitive ratio of the randomized algorithm:

$$\alpha = \frac{(10k)^p}{3 \cdot (10k)^{p-1}} = \frac{10k}{3}$$

But this contradicts our assumption that the randomized algorithm has competitive ratio smaller than k, so the lemma must be true. □

Lemma 3 tells us that $\mathbb{E}[Z_p^j] \geq \epsilon$ for all $j \in \{0\} \cup \mathbb{N}$ and every $p \in \{0, 1, \ldots, j\}$ is a necessary condition for any randomized algorithm with a bounded competitive ratio. This condition turns out to be problematic, as we shall soon see.

Suppose the principal selects a randomized algorithm that satisfies this condition. The adversary selects agent input $\mathcal{I}_{j^*}$ such that $j^* \in \mathbb{N}, j^* > \frac{1}{\epsilon}$. Because we know that this randomized algorithm satisfies $\mathbb{E}[Z_p^{j^*}] \geq \epsilon$ for all $p \in \{0, 1, \ldots, j^*\}$, we know that the total performance of the randomized algorithm can be described by the following:

$$\mathbb{E}\left[\sum_{p=0}^{j^*} Z_p^{j^*}\right] = \sum_{p=0}^{j^*} \mathbb{E}[Z_p^{j^*}] \geq j^* \cdot \epsilon > 1$$

Note that the first equality is a result of the linearity of expectation, and the first inequality is due randomized algorithm satisfying the necessary condition. The second inequality is a result of how we defined j^*. This result says that the mean amount of incentives that the randomized algorithm will pay out is greater than its budget $\overline{B}$. This is strange!

More formally, we know that the total amount of incentives paid is $\sum_{p=0}^{j^*} Z_p^{j^*}$. Since the principal's budget is 1, this summation must takes values on the closed interval $[0, 1]$. Thus, we know that $\mathbb{E}[\sum_{p=0}^{j^*} Z_p^{j^*}]$ cannot lie outside of this interval. In particular, it must be less than or equal to 1.

Thus, we have a contradiction, and so the necessary condition prescribed in Lemma 3 cannot be satisfied. We can conclude that no randomized algorithm achieves a bounded competitive ratio, even when knowledge about current valuations is given to the algorithm. □

Note that because deterministic algorithms are a proper subset of randomized algorithms (deterministic algorithms can be thought of as randomized algorithms that do not utilize the randomness provided to them), we can say that Theorem 4 implies that no deterministic algorithm in the online model has a bounded competitive ratio.

This result may seem discouraging at first, but it is reassuring upon closer inspection. For all of the problems we have studied so far, INDUCE-MULTI for a principal with a total budget across rounds is the only one that cannot be solved optimally using OPT. However, Theorem 4 demonstrates that no matter how sophisticated the principal's incentive design strategy is, he cannot provide any worst-case performance guarantees. We find this result reassuring because it means that OPT is just as good as anything else in the worst case. This is desirable because OPT requires no computation and no knowledge of the agent's valuations any time step.

The result of Theorem 4 encourages us to turn our attention to settings with a more informed principal, which we consider in the next chapter.

3.3.3 Induce Target Multiple Times (Offline)

We now take a brief detour from the online model to consider INDUCE-MULTI with a total budget in the offline model. We define the problem in the offline model as follows. The principal is given complete knowledge of the agent input $\mathcal{I}$. That is, he knows the entire sequence of valuations for each of the arms. Given this knowledge, how should the principal assign incentives at each round to maximize the number of pulls of the target arm within a certain number of rounds R? We note that we could obtain a solution via brute force by trying all of the possibilities, but this method is computationally intractable.

It is interesting to ask whether this problem is tractable in an offline model.

Theorem 5. *Given knowledge of the agent input $\mathcal{I}$, the offline optimal solution to* INDUCE-MULTI *for a principal with a total budget across rounds can be computed in*

polynomial time.

Proof. In this proof, we will devise a non-trivial incentive policy that can be efficiently computed. Note that the offline optimal algorithm will be non-trivial because it must determine how the total budget should be split across rounds such that it is used the most effectively. We demonstrate how to compute the this incentive policy and prove that it indeed returns the optimal solution. To begin our analysis, we consider the following subproblem.

Problem 3. *Given a total number of rounds $\bar{t} > 1$, a desired number pulls of the target arm $m > 1$, and a total budget $\overline{B}$, find an incentive policy Δ such $v_g(x_g(\bar{t})) = v_g(x_g^{\ell})$ where $\ell \geq m$, when such an incentive exists.*

We see that if we can obtain a solution to Problem 3, then we can input $\bar{t} = R$ and repeatedly run the solution with different values of m. We can use a binary search to figure out the maximal m that is achievable under the budget $\overline{B}$ and the number of rounds R. Now all we need is an efficient solution to Problem 3.

Given $\bar{t}$ and m, if the agent selects the target arm exactly m times after $\bar{t}$ rounds, then he necessarily spent $\bar{t} - m$ rounds selecting non-target arms. One effective way of assigning incentives would be to figure out when it is cheapest to provide incentives to get the target arm to be selected. Because the principal has knowledge of the agent input $\mathcal{I}$, this is possible. We use a threshold to formalize this idea.

Taking the above observations into account, we devise the following algorithm. Simulate $\bar{t} - m$ rounds of the agent on all arms $i \in K_{-g}$ when no incentive is applied. Set the inducible threshold as the following value

$$\underline{v} = \min_{1 \leq t \leq \bar{t}-m+1} \max_{i \in K_{-g}} v_i(x_i(t))$$

$\underline{v}$ represents the point at which it is cheapest to provide incentives to the target arm to get it selected. Notice that the quantity that we aim to minimize is the value of the highest non-target arm, or the arm against which the target is competing.

Importantly, given the agent input $\mathcal{I}$, this simulation can be done in polynomial time. Drawing inspiration from OPT, we define the following incentive strategy.

Definition 3. *The 'only provide to target when cheap' (OPTc) incentive policy assigns* $\Delta_i(t) = 0$ *for all times* t *before the threshold* $T = \underline{v}$ *is met, and provides* $\Delta_g(t') = \max\{0, \underline{v} - v_g(x_g(t'))\}$ *in all subsequent rounds* t'*, while there is enough budget remaining. Otherwise,* $\Delta_g(t') = 0$*. No incentives are ever provided to non-target arms.*

Example 1. To illustrate how OPTc works, we return to the cereal purchasing example, except this time we change Alice's sequence of valuations on Choco Flakes to make our example more informative. Choco Flakes is still the desired action, and $\overline{B} = 10$. Suppose the principal is given knowledge of Alice's agent input:

$v_i(x_i^k)$	$k=0$	1	2	3	...
Apple Loops	4	6	3	10	10
Berry Bites	5	2	4	6	1
Choco Flakes	7	5	2	3	1

The principal would like to solve Problem 3 with $\bar{t} = 20, m = 7, \overline{B} = 10$. To use OPTc, he would simulate Alice's actions forward $20 - 7 = 13$ steps on only the non-target actions: Apple Loops and Berry Bites. To obtain the correct threshold $T = \underline{v}$, the principal looks at the value of the action selected at each time step during the simulation. He then sets $\underline{v}$ to be the minimum over those values.

In this example, if the principal simulates for 13 steps over Apple Loops and Berry Bites with no incentive at each round, he sees the following pulls in order (where i $(n + k)$ denotes that i was selected, had value intrinsic value n and incentives k provided to it at the time of selection):

$$b\ (5) \to a\ (4) \to a\ (6) \to a\ (3) \to a\ (10) \to a\ (10) \to \dots$$

Taking the minimum of these values gives us $\underline{v} = 3$. Notice that even though calculation of $\underline{v}$ looks computationally intensive because we are desire the minimum of the maxima, this simulation idea allows us to find $\underline{v}$ quickly.

Now that the principal has identified the proper threshold, he executes OPTc as follows: provide no incentive until the value of Apple Loops is 3. This results in the following pulls. $c\ (7) \rightarrow c\ (5) \rightarrow b\ (5) \rightarrow a\ (4) \rightarrow a\ (6)$. Now, the threshold is met because the value of Apple Loops is 3. To proceed with OPTc, we consider what incentives we should provide.

If the value of Choco Flakes is less than or equal to 3, provide 0 incentive. Otherwise, provide exactly the difference between 3 and the value of Choco Flakes. Continue until either the budget is exhausted or if the total number of rounds has elapsed. This results in the following pulls: $c\ (2+1) \rightarrow c\ (3) \rightarrow c\ (0+3) \rightarrow c\ (0+3) \rightarrow c\ (0+3)$. Now, our budget $\overline{B} = 10$ has elapsed, and we can provide no more incentives. So far, there have been 10 rounds of interaction. For the remaining 10 rounds of interaction, Choco Flakes will never be selected again. Nonetheless, we see that in this case $m = 7$ is feasible.

If we can prove that OPTc does indeed yield the optimal solution, then we simply need to run a binary search over the values of m. Since this can be done in polynomial time, proving that OPTc yields the optimal solution also proves Theorem 5.

Lemma 4. *When used in the offline model under a fixed budget across rounds, OPTc solves Problem 3.*

Proof. We begin by showing that the threshold $\underline{v}$ is the cheapest point at which we can provide incentives to the target over at most $\bar{t} - m$ selections of non-target arms. We denote this cheapest point v^* as follows:

$$v^* = \min_{m_{-g}} \max_{i \in K_{-g}} v_i(x_i^{m_i}(t))$$

$$\text{s.t.} \sum_{i \in K_{-g}} m_i \leq \bar{t} - m$$

v^* represents the cheapest point at which to provide incentives given at most $\bar{t} - m$ selections of non-target arms, where we define $m_{-g} = (m_1, \ldots, m_{g-1}, m_{g+1}, \ldots, m_n)$.

We wish to show that $\underline{v} = v^*$. To do this, we show that $\underline{v} \geq v^*$ and $\underline{v} \leq v^*$. To begin, we know that $\underline{v} \geq v^*$ because $\underline{v}$ is included in the space over which v^* is minimized. We now show that $\underline{v} \leq v^*$.

Assume for a contradiction that $\underline{v} > v^*$. We use $m_j^*, j \in K_{-g}$ to denote the number of pulls of each of the non-target arms that minimize v^*. We run a simulation over the non-target arms for a total of $\sum_{i \in K_{-g}} m_i^*$ rounds. Let ℓ_i for $i \in K_{-g}$ denote the number of selections of non-target arm i that result during this simulation. It must be the case that $\sum_{i \in K_{-g}} \ell_i = \sum_{i \in K_{-g}} m_i^*$. If $\ell_i = m_i^*, \forall i \in K_{-g}$, then we can conclude that $\underline{v} = v^*$. Otherwise, we know that $\ell_i > m_i^*$ for some $i \in K_{-g}$ (and $\ell_j < m_j^*$ for some $j \in K_{-g}, j \neq i$). We can now state the following:

$$v_i(x_i^{m_i^*}) \leq v^* < \underline{v}$$

where the first inequality is by the definition of v^*, and the second inequality is by assumption. However, since $\ell_i > m_i^*$, it must be the case the case that arm i is selected at least one more time before the threshold $\underline{v}$ can be met. But this means that $v_i(x_i^{m_i^*}) > \underline{v}$, and so we have a contradiction. Thus, it must be the case that $v^* = \underline{v}$.

Now, we have shown that OPTc uses $\underline{v} = v^*$ as the threshold T. This corresponds to the cheapest point at which incentives can be provided. Notice that OPTc satisfies the conditions of the threshold lemma. Now, we can break this down into two cases: if the threshold is not met and if it is met. If the threshold is not met by time $\bar{t}$, then by part (a) of the threshold lemma we know that there were fewer than $\bar{t} - m$ selections of non-target arms. This means that there were more than m selections of target arms. Now we only need to consider the case where the threshold is met. This proceeds similarly to the final part of the proof for Lemma 2. We simply need to note that OPTc will never waste any incentive before the threshold is met, so that it conserves its budget to ensure that if m selections of the target arm are possible, then it will be achieved. Otherwise, it is impossible given the budget and time constraints. $\square$

Now that we have shown that OPTc, when given parameter m, achieves at least that number of pulls if possible, we can determine the largest feasible m using a binary search. Running OPTc with the maximum feasible value provides the desired optimal incentive policy. This entire process can be completed in polynomial time. □

Theorem 5 provides a counterpoint to the negative result found in Theorem 4. While it is impossible to design any randomized algorithm with a bounded competitive ratio in the online model, there is a computationally tractable algorithm that provides the optimal solution in the offline model. Theorem 4 suggests that if we wish to make progress in designing incentive policies in the general case, we must consider average case analysis or appeal to empirical results. Theorem 5 also encourages us to move in this direction by providing us with an efficient means of calculating the baseline performance, given any agent input $\mathcal{I}$.

3.4 Discussion

In this section, we reflect on the implications of our findings in this chapter. We also consider ways in which we can move past the negative result found in Theorem 4.

3.4.1 OPT

Our results are summarized in Table 3.1.

	Induce-Once	Induce-Multi
Per-period	OPT optimal (Thm 1)	OPT optimal (Thm 2)
Total across rounds	OPT optimal (Thm 3)	Unbounded ratio (Thm 4)

Table 3.1: Summary of our results in the online model for the two objective criteria under different budget models

In this table, the rows correspond to the different budget models that we use, and the columns correspond to the different objective criteria that we consider. Having structured our results in this manner, we see a striking structural element. If we begin

by considering INDUCE-ONCE under a per-period budget, we are allowed to change either the budget model or the objective criterion and OPT still returns the optimal solution. However, if we change both the budget model and the objective criterion at the same time, then we end up with a problem for which no randomized algorithm can give any performance guarantees in the worst case. From a worst-case analysis, this is reassuring to us because it informs us that no algorithm can do better than OPT in the worst case.

The power of OPT is even more remarkable when we consider the fact that the principal can apply it without having to perform any computations whatsoever. Furthermore, because we are in an online model, OPT provides these performance guarantees without any knowledge of the agent's current or future valuations over the arms. We also note that OPT returns the optimal solution to INDUCE-MULTI under a per-period budget even without knowledge of how many pulls of the target arm are desired. For any $k \in \mathbb{N}$, OPT will always get k pulls of the target arm as soon as possible.

One important takeaway from our analysis is that, in these cases, the principal would never want to place incentives on non-target arms. This is because there is no way to change the learning process over the non-target arms as a means of speeding up selections of the target arm; it is necessary for the agent to learn that non-target arms are bad (i.e., letting them fall below a desired inducible threshold) before the desired number of selections of the target arm can occur. While certain incentive schemes can get specific individual non-target arms to fall below the threshold sooner than OPT, such incentive schemes can never satisfy a threshold condition over the aggregate of non-target arms sooner than OPT, by the threshold lemma. We can think about this as follows: an agent must reach a certain part of his learning process over each of the non-target arms before a desired objective criterion can be satisfied. This point in the learning process over all of the non-target arms will occur the quickest in the absence of incentives on non-target arms. In fact, incentives on non-target arms could

potentially lengthen the time it takes for the objective to be satisfied. In some cases, it could even make a feasible problem impossible.

3.4.2 Next Steps

In moving forward, we refer to the results provided by Theorems 4 and 5 for guidance. Theorem 4 is a strong negative result, which we can attribute to the fact that the allowable set of agent inputs $\mathcal{I}$ is unrestricted. On the other hand, Theorem 5 is a strong positive result that tells us that computationally tractable algorithms exist when the agent input is fully specified to the principal. We seek to study problems that lie within the gap posed by Theorem 4 and 5. This takes the following form: given some, but not complete, knowledge of the agent input, what incentive policies can the principal use to maximize his objective criterion?

Chapter 4

Incentive Design with Knowledge of Agent Parameters

As we discovered in the previous chapter, there exists a gap between what the principal can achieve in the absence of knowledge about the agent's belief update process and what he can achieve with complete knowledge. In this chapter, we consider what can be done when the principal has more knowledge about the agent. Indeed, in most realistic scenarios, we expect the principal to have some knowledge about how the agent updates his beliefs and determines his perceived value of the arms.

Note that knowledge of the agent's update and valuation function does not imply knowledge of the agent's input, i.e., the value received in each round. This is because the underlying distribution behind each of the arms is a property of the environment. In particular, even if the principal has knowledge of the specific distribution that lies behind each arm, we assume that he cannot observe the realization that the agent experiences. Thus, the principal's problem still involves some uncertainty, and the problem is not fully "offline."

We begin this chapter with a motivating example that helps to illustrate some of the techniques and reasoning that will be used in the analysis of this new setting. We consider a class of agent parameters that a principal might reasonably expect to be faced with in real-life scenarios. Along with these agent parameters, we assume that the principal has knowledge of the underlying distribution behind each of the arms

in the agent's decision environment. We define a new objective criterion that is able to capture the tradeoff between the principal's utility over incentives provided and activations of the target arm and devise several algorithms that aim to maximize the principal's objective criterion.

4.1 Example: A New Competitor

On his way home from work one day, Bob notices that a new Italian restaurant called Al's is opening up in Boston. Italian being his favorite cuisine, Bob is curious how this new entrant stacks up against Bart's, his current favorite option.

Because Al's is new in town, the management would like to establish its name in the area via word of mouth. They decide that the best way to do this is to offer its food free of charge to all customers on the day of its grand opening. Bob, an Italian food connoisseur, attends and obtains some information about the quality of the food served by Al's restaurant.

Unfortunately for Al's, the chefs at Bart's have perfected all of their classic recipes throughout the years. Furthermore, the customers in the area already have a good idea about the quality of the food served by Bart's. In contrast, the chefs at Al's have yet to pin down a set of reliable suppliers for their ingredients, making the quality of their food highly variable. They are unable to observe the quality of their ingredients. Concerned by this, the management decides that it is necessary to provide coupons if they wish for Bob to return. Given their limited operating budget for the first month of operation, how should the management provide coupons to Bob so that Bob will return to their restaurant?

4.1.1 Two Round Case

The management assumes the following about Bob: every two weeks, Bob will visit the Italian restaurant with the highest perceived value plus incentives provided, where

his perceived value is based on the empirical mean of the outcomes obtained during all previous visits.[1] Furthermore, the management estimates that Bob always sees a realized outcome of 1 from Bart's. In comparison, they estimate that, including the day of their grand opening, Bob sees an outcome that varies uniformly on the continuous interval $[0, 1]$ on each visit. The total amount of coupons that Al's can provide corresponds to about 1 unit of Bob's value. Since Bob will make two choices over the next month, how should they provide incentives over each of the two weeks to maximize the expected number of times that Bob will visit their restaurant?

We model this problem as follows. The management has a total budget $\overline{B} = 1$ to be split over two rounds. Recall that they need only pay out incentives if Bob selects the action on which those incentives were placed (i.e., they keep their entire budget into the second round if Bob selects Bart's during the first round, since their incentives only apply to Al's in this case). Let a_i denote each of a sequence of values that Bob obtained from selecting Al's, beginning with a_0. We can think of Bob's belief state at any given time t as an array of all his observed values a_i. Each value a_i can be thought of as a single realization of the random variable O_a that is continuously uniform over the interval $[0, 1]$. Bob's valuation function v_a takes the current belief state and returns the average of all the values within the array. $v_b = 1$ at all times t.

At time $t = 1$, Bob's belief state $x_a(1)$ on Al's is an array that contains one value a_0, which represents his realized outcome when he attended the grand opening at Al's. Thus, his initial valuation is distributed uniformly between the interval $[0, 1]$. Let A_i be the event that Bob selects Al's at time i. Since we seek to maximize the expected number of times that Bob selects Al's over the next two rounds, we wish to maximize the following quantity:

$$2P(A_1)P(A_2|A_1) + 1P(A_1)P(\neg A_2|A_1) + 1P(\neg A_1)P(A_2|\neg A_1)$$

$$= P(A_1)[2P(A_2|A_1) + P(\neg A_2|A_1)] + P(\neg A_1)P(A_2|\neg A_1)$$

[1] Because they respect his privacy, the management at Al's cannot observe Bob's value even on the nights he visits.

$$= P(A_1)[1 + P(A_2|A_1)] + P(\neg A_1)P(A_2|\neg A_1)$$

Notice that we can simplify the last term as $P(\neg A_2)$ in this case, since if Bob does not select Al's during round 1, the management can simply put their entire incentive value of 1 on Al's for the second week to guarantee a selection, which translates to $P(A_2|\neg A_1) = 1$. This allows us to simplify the entire expression into $P(A_1) + P(A_1)P(A_2|A_1) + P(\neg A_1) = P(A_1, A_2) + 1$, which is the expected number of "pulls" of Al's that Bob will take. We wish to find incentives that maximize this quantity.

Let δ be the amount of incentive that the principal provides to Al's at round 1. Since we only care about the probability that Al's is selected during both rounds, we assume that Bob chooses Al's during the first round, which means that the principal provides the remaining $1 - \delta$ during the next round. Recall that a_0 is Bob's initial valuation of Al's. Thus, we have the following:

$$P(A_1, A_2) = P(\delta + a_0 \geq 1, \frac{a_0 + a_1}{2} + 1 - \delta \geq 1) = P(1 - \delta \leq a_0 \leq 1, \frac{a_0 + a_1}{2} \geq \delta)$$

Given the above, we can obtain $P(A_1, A_2)$ as a function of δ by integrating over the probability density function of the uniform distribution over the valid regions based on the value of δ. Because we must be careful not to integrate outside of the valid regions, we must separate the integrals appropriately We defer a detailed analysis of this process to Appendix A. After completing the integration over a_0 and a_1, we obtain $P(A_1, A_2)$ as a piecewise function δ that can be specified as follows:

$$P(A_1, A_2) = \begin{cases} \delta & \text{if } \delta \leq \frac{1}{3}. \\ -\frac{9}{2}\delta^2 + 4\delta - \frac{1}{2} & \text{if } \frac{1}{3} < \delta \leq \frac{1}{2}. \\ -\frac{5}{2}\delta^2 + 2\delta & \text{if } \frac{1}{2} < \delta \leq \frac{2}{3}. \\ 2\delta^2 - 4\delta + 2 & \text{if } \frac{2}{3} < \delta \leq 1. \end{cases}$$

We see that the rate of change of this function varies according to δ. This means that a small shift in δ may have different consequences for what happens to $P(A_1, A_2)$

depending on the value of δ. To better understand this function, we consider its graph in Figure 4.1.

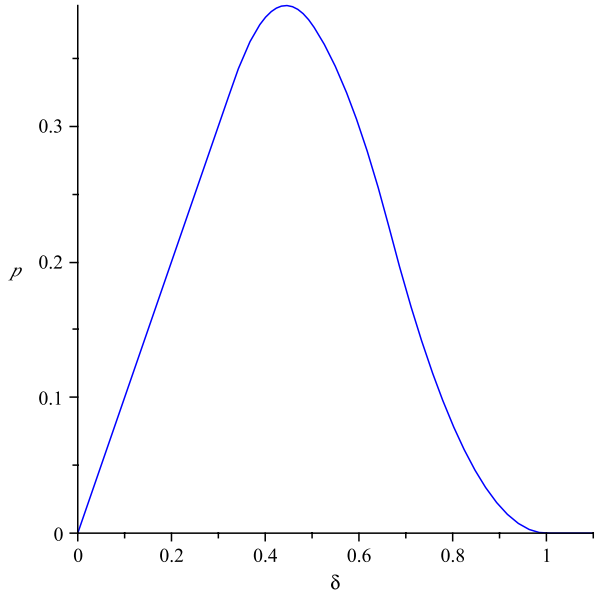

Figure 4.1: $P(A_1, A_2)$ as a function of δ

Taking the derivative of the piecewise function and examining the critical points, we see that the maximum occurs at $\delta = \frac{4}{9}$, with a value of $P(A_1, A_2) = \frac{7}{18}$. Thus, we see that the maximum number of expected number of selections of Al's is $P(A_1, A_2) + 1 = \frac{25}{18}$. To obtain this, the management would provide $\Delta_a(1) = \frac{4}{9}$ in the first round. If Bob selects Al's, then they would provide $\Delta_a(2) = \frac{5}{9}$ in the second round. Otherwise, they would provide $\Delta_a(2) = 1$.

How should we interpret this result? Intuitively, the management should conserve some of his budget so that they can still induce Bob to select Al's in the second time period. On the other hand, if they are too conservative with the amount provided during the first round, then they might risk missing out on a selection during the first round. With this in mind, $\delta = \frac{4}{9}$ seems like it would be a good choice. But why is it a better choice than $\delta = .5$?

Suppose the management provides $\Delta_a(1) = .5$. Then $P(A_1) = .5$, since this just

corresponds to the probability that $a_0 \geq .5$. Given that A_1 occurred, we can provide $\Delta_a(2) = .5$. Now, we would like to calculate the probability of $P(A_2|A_1)$, which is the probability that $\frac{a_0+a_1}{2} \geq .5$, given that $0.5 \leq a_0 \leq 1$. By taking the integral, we see that this comes out to $\frac{3}{4}$.

Suppose the management provides $\Delta_a(1) = \frac{4}{9}$. Then $P(A_2) = \frac{4}{9}$ and we know that $\frac{5}{9} \leq a_0 \leq 1$. Given that A_1 occurred, we can provide $\Delta_a(2) = \frac{5}{9}$. To obtain the probability of $P(A_2|A_1)$, we need to compute the probability that $\frac{a_0+a_1}{2} \geq \frac{4}{9}$, given that $\frac{5}{9} \leq a_0 \leq 1$. We see that this comes out to $\frac{7}{8}$ upon taking the integral.

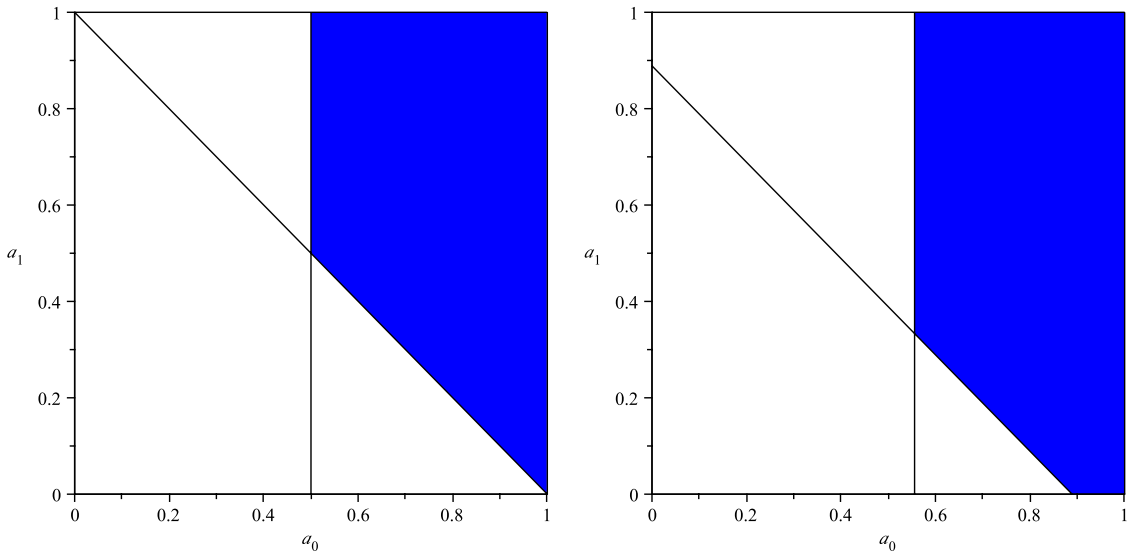

Figure 4.2: Values of a_0, a_1 that allow for two selections; left: $\delta = 0.5$, right: $\delta = \frac{4}{9}$

We can think of the conditional probability $P(A_2|A_1)$ as the fraction of the area to the right of $a_0 = 1 - \delta_1$ that is shaded blue in Figure 4.2. In the graph on the left in Figure 4.2 where $\delta = .5$, this conditional probability is $\frac{3}{4}$. In the graph on the right where $\delta = \frac{4}{9}$, this conditional probability is $\frac{7}{8}$. Even though the principal may end up sacrificing a small amount of $P(A_1)$ to obtain this higher value, the product $P(A_1)P(A_2|A_1) = P(A_1, A_2)$ can be increased by moving from $\delta = 0.5$ to $\delta = \frac{4}{9}$. This is because the slight sacrifice in the probability of selection during the first round allows for a greater increase in the probability of obtaining a selection during the

following round. By using $\delta = \frac{4}{9}$ as opposed to $\delta = 0.5$, the expected number of selections increases from 1.375 to $1.3\bar{8}$.

Notice that the graphs illustrate the tradeoff between the probability of selection during the first round and the probability of selection during the second round. As δ increases, the probability of getting a selection in the first round, $P(A_1)$, increases linearly. However, the same cannot be said of the probability of a selection in the second round given a selection in the first. This is because $P(A_2|A_1)$ depends on the ratio of the area above the line $\frac{a_0+a_1}{2} \geq \delta$ to the total area in the region $1-\delta \leq a_0 \leq 1$. We can see this play out geometrically. As δ decreases from $\frac{4}{9}$, we see that the vertical line corresponding to $a_0 = 1-\delta$ will shift toward the right, while the line corresponding to $\frac{a_0+a_1}{2} = \delta$ will shift downward. Eventually, they will meet at some point where $a_1 = 0$. To see this, consider two lines $\frac{a_0+a_1}{2} = \delta, a_0 = 1-\delta$. Setting $a_1 = 0$ yields

$$\frac{a_0}{2} = \delta, a_0 = 1 - \delta$$

Substituting for a_0 yields $\delta = \frac{1}{3}$. At that point, we see have the graph in Figure 4.3.

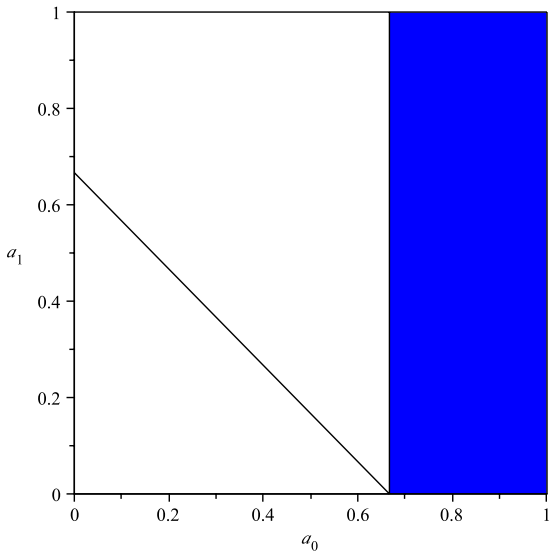

Figure 4.3: Graph illustrating $P(A_2|A_1)$ when $\delta = \frac{1}{3}$

Once δ decreases further past this point, the probability $P(A_2|A_1)$ is guaranteed

to be equal to 1. Suppose $\delta \leq \frac{1}{3}$. If Al's was selected in the first round, then $a_0 \geq \frac{2}{3}$. This means that in the worst case where $a_1 = 0$, the average of the two quantities is still at least $\frac{1}{3}$. Since there is still at least $1 - \frac{1}{3} = \frac{2}{3}$ of the budget remaining, a selection in the second round is guaranteed. Thus, $P(A_2|A_1) = 1$. These points at which the geometric structure of $P(A_1, A_2)$ demonstrate why it is a piecewise function of δ.

This example is illustrative. It demonstrates that the principal, through his observations of the agent, can deduce certain properties about the realized outcomes that the agent observes. Armed with only the distributions that lie behind each of the arms and observations about the agent's selections, the principal can pare down the agent's state space. For instance, if the management provided $\delta = \Delta_a(1) = 0.5$, and Bob does not select Al's during the first round, then we can conclude that Bob's value for Al's is lower than 0.5. We see in this example that reasoning about the conditional probabilities allows the principal to achieve better results that he would otherwise. In this case, when he makes a decision about how set δ, he is implicitly making a decision about how much he would be able to provide in the second round. He is able to improve his performance by reasoning about this tradeoff and using the information obtained by observing the agent's actions. This example hints at the potential for more sophisticated algorithms in this general problem setting.

4.1.2 Three Round Case

An interesting problem to consider would be how the management should divide up its budget when there are a total of three rounds of interaction. Because the analytical solution is difficult to reason about, we analyze some of the results that we obtained via simulation. The simulation iterates over all possible incentive provisions in the following way: because there are three rounds of interaction and a total budget, the principal must decide how much incentive to provide during the first round and how much incentive to provide during the second round. However, the principal may

decide to provide a different amount during the second round, depending on the agent's observed action in the first round. Once these values have been determined, the amount of incentive provided at the third round has no degrees of freedom, because the principal would just use all of the remaining budget.

We iterate over each of the possible incentive provisions with steps of .01 and perform a Monte Carlo simulation for each to estimate its expected number of selections of the target arm. Our simulation results suggest that the optimal incentive design strategy is to provide 0.37 in the first period. If the target arm is selected during the first round, then 0.27 should be provided in the second period. Otherwise, the principal should provide 0.54.

These results corroborate our analysis of the two round case. When there are three rounds, the principal should be somewhat conservative with the amount of incentive provided during the first round, so as to preserve the remaining budget for use over the next two rounds. After applying 0.37 in the first round and seeing a selection, the principal can be certain that $a_0 \geq .63$. Thus, $0.315 \leq \frac{a_0+a_1}{2}$.

Having observed a selection of the target arm in the first round, the principal can be fairly certain that the agent has a high average value for the target arm. Thus, he spends approximately $\frac{.27}{.63}100 \approx 43\%$ of his remaining budget during round two. This is reminiscent of the $\frac{4}{9}100 = 44.\bar{4}\%$ spent in the first round of the two round case. *However, we caution that the problem faced by the principal during the second round of the three round case is distinct from the problem he faces during the first round of the two round case.* There are a variety of reasons for this. The remaining budget in the former is lower, while there is more information available. Furthermore, the update dynamics upon viewing the third realization of an arm differs from those upon viewing the second second realization of the arm. This is because the principal assumes that the agent is updating in this case. Thus, the expected magnitude of the shift in the agent's valuation decreases as the number of observations increases.

If the agent did not select the target arm with the 0.37 incentives provided on

the target arm during the first round, the principal knows that $a_0 \leq .63$. In this case, we essentially have the problem faced by a principal in the first round of a two round case. This is because we assume that the non-target arm will always have a realized value of 1. Thus, the agent's information state does not change as it did in the previous case when the target was selected during the first round. Unlike in the two round problem, however, there is extra information available to the principal. As a quick sanity check, we add the fact that $a_0 \leq .63$ and solve for the value of $\Delta_a(2)$ that maximizes the expected number of pulls of the target arm in the last two rounds. It turns out that adding the information yield $\Delta_a(2) = .548$ as the optimal incentive to provide (refer to Appendix A for the calculation). The simulation results of providing 0.54 corroborate this analysis, and we can attribute the slight difference to the coarseness of our stepping interval.

4.1.3 Discussion

By reasoning through this example, we have gained some insight into how a principal might want to design incentives so as to maximize the number of pulls of the target arm given a certain number of rounds and a total budget. We can carry out this analysis for similar cases in which the values of the target arm are drawn from a continuous uniform distribution and the non-target arm is stationary.

However, our analysis also demonstrates that analytic reasoning about the optimal incentive design quickly becomes a difficult problem. One of the reasons for this is that there are exponentially many cases to consider. Even with just the three round case, the optimal amount of incentives to provide at the second round depended on the action that the agent selected in the first round. Furthermore, our analysis shows that as additional information is obtained, the decision about how to provide incentives can be further refined. Since each sequence of pulls (along with provided incentive) may lead to a unique information state, we would have to reason about an exponential number of cases to arrive at the optimal incentive design strategy

analytically. This also poses a barrier in terms of obtaining a good design strategy via Monte Carlo sampling. Because we need to simulate over all possible sequences of actions to determine the best amount of incentives to provide at each step, we would have to explore over a state space that is exponential in the total number of rounds.

While this result may be disappointing, it encourages us to think about a different goal. Instead of demanding the optimal solution in these scenarios, are there ways in which we can learn about good ways to intervene such that we can still perform close to optimal? For instance, our intuition is that we should provide half of the incentives during the first round of the two arm case leads to performance that is within 1% of the optimal number of expected pulls. Are there simple ways for us to obtain good design strategies in general?

4.2 A Continuous Model

Further extending the previous example, we consider a problem setting in which the agent is faced with two arms, each of which has an underlying continuous uniform distribution. Suppose the target arm g is distributed uniformly on the interval $[\underline{g}, \overline{g}]$, and the non-target arm h is distributed uniformly on the interval $[\underline{h}, \overline{h}]$. Can we say anything about how incentives should be provided in this case? We focus on this question in this section.

We begin by considering how the principal should decide what to do in the first round of an interaction with an agent that spans multiple rounds. Because it is difficult to think about how the principal can incorporate information obtained during the first round to analyze future rounds in this setting, we restrict our attention to the one round case. We consider the following question: are there certain incentives that are better to provide than others? That is, can the principal maximize the effect of his incentives so that he can conserve some for future use?

We begin our analysis by looking at a graphical representation. Let the value

of h be represented on the x-axis, and the value of g be represented on the y-axis. Notice that the rectangular region $R = \{(x, y)|\underline{h} \leq x \leq \overline{h}, \underline{g} \leq y \leq \overline{g}\}$ represents the outcome space, and the fraction of R that lies above the line $y = x$ is the probability that the target item will be induced. The principal can decide how much incentive δ to provide to the target item g. We can think about this incentive δ as shifting the uniform distribution upward by δ units such that the value of the goal arm plus incentive is drawn uniformly from the continuous interval $[\underline{g} + \delta, \overline{g} + \delta]$. Graphically, this can be seen in two different ways: we can shift R vertically upward δ units along the y-axis, or we can leave R in place and shift the line of induction to be $y = x - \delta$ (equivalent to a right-ward shift of δ units along the x-axis).

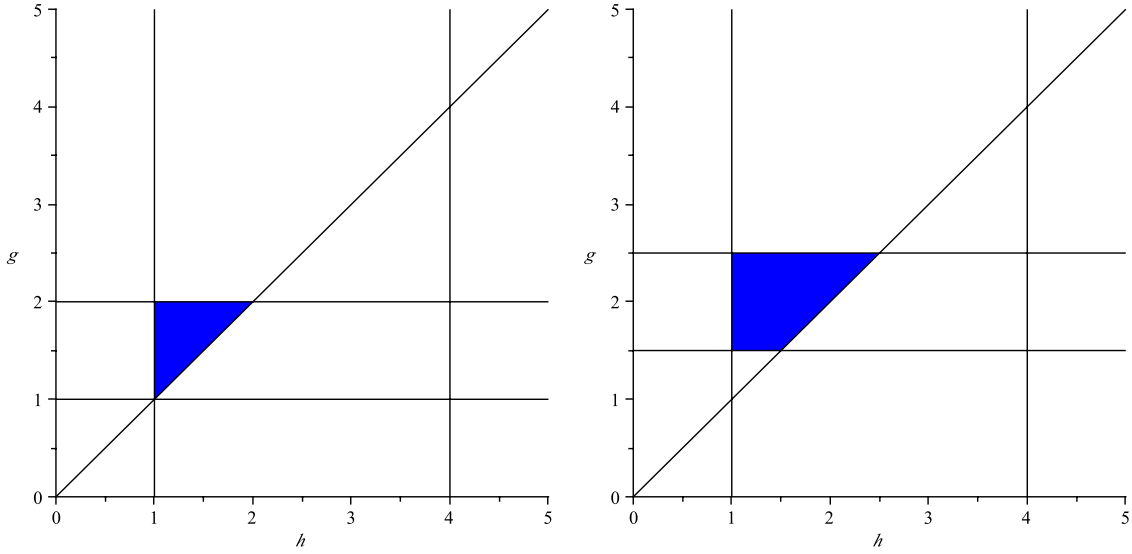

Figure 4.4: An example where g drawn from $U[1, 2]$ and h is drawn from $U[1, 4]$; left: $\delta = 0$, right: $\delta = .5$

Using the graphical representation, we can think about four cases that, together,

can help us solve the problem analytically:

$$\underline{g} + \delta \leq \underline{h} \text{ and } \overline{g} + \delta \leq \overline{h} \tag{1}$$

$$\underline{g} + \delta \geq \underline{h} \text{ and } \overline{g} + \delta \leq \overline{h} \tag{2}$$

$$\underline{g} + \delta \leq \underline{h} \text{ and } \overline{g} + \delta \geq \overline{h} \tag{3}$$

$$\underline{g} + \delta \geq \underline{h} \text{ and } \overline{g} + \delta \geq \overline{h} \tag{4}$$

We can compute the following integrals to calculate the probability of inducing the target item:

$$\frac{1}{(\overline{h} - \underline{h})(\overline{g} - \underline{g})} \int_{\underline{h}}^{\overline{g}+\delta} \int_{\underline{h}}^{y} \mathrm{d}x\mathrm{d}y = \frac{1}{(\overline{h} - \underline{h})(\overline{g} - \underline{g})} \left(\frac{(\overline{g} + \delta)^2 - \underline{h}^2}{2} - \underline{h}(\overline{g} + \delta - \underline{h}) \right) \tag{1}$$

$$\frac{1}{(\overline{h} - \underline{h})(\overline{g} - \underline{g})} \int_{\underline{g}+\delta}^{\overline{g}+\delta} \int_{\underline{h}}^{y} \mathrm{d}x\mathrm{d}y = \frac{1}{(\overline{h} - \underline{h})(\overline{g} - \underline{g})} \left(\frac{(\overline{g} + \delta)^2 - (\underline{g} + \delta)^2}{2} - \underline{h}(\overline{g} - \underline{g}) \right) \tag{2}$$

$$\begin{aligned} &\frac{1}{(\overline{h} - \underline{h})(\overline{g} - \underline{g})} \left(\int_{\underline{h}}^{\overline{h}} \int_{\underline{h}}^{y} \mathrm{d}x\mathrm{d}y + \int_{\overline{h}}^{\overline{g}+\delta} \int_{\underline{h}}^{\overline{h}} \mathrm{d}x\mathrm{d}y \right) \\ &= \frac{1}{(\overline{h} - \underline{h})(\overline{g} - \underline{g})} \left(\frac{\overline{h}^2 - \underline{h}^2}{2} - \underline{h}(\overline{h} - \underline{h}) + (\overline{h} - \underline{h})(\overline{g} + \delta - \underline{h}) \right) \end{aligned} \tag{3}$$

$$\begin{aligned} &\frac{1}{(\overline{h} - \underline{h})(\overline{g} - \underline{g})} \left(\int_{\underline{g}+\delta}^{\overline{h}} \int_{\underline{h}}^{y} \mathrm{d}x\mathrm{d}y + \int_{\overline{h}}^{\overline{g}+\delta} \int_{\underline{h}}^{\overline{h}} \mathrm{d}x\mathrm{d}y \right) \\ &= \frac{1}{(\overline{h} - \underline{h})(\overline{g} - \underline{g})} \left(\frac{\overline{h}^2 - (\underline{g} + \delta)^2}{2} - \underline{h}(\overline{h} - \underline{g} - \delta) + (\overline{g} + \delta - \overline{h})(\overline{h} - \underline{h}) \right) \end{aligned} \tag{4}$$

Let us work through an example. Suppose g is drawn from $U[0, 0.5]$ and h is drawn from $U[0.5, 1.5]$. Notice that if $\delta \geq 1.5$, then the probability of inducing the target is 1. We must now find the values of $\delta \leq 1.5$ for which changes in the cases occur. We see that if $0 \leq \delta \leq 0.5$ is case (1), $0.5 \leq \delta \leq 1$ is case (2), and $1 \leq \delta \leq 1.5$ is case (4). Plugging in the appropriate values, we find the following:

$$P(g \text{ selected}) = \begin{cases} \delta^2 & \text{if } 0 \leq \delta < 0.5 \\ \delta - 0.25 & \text{if } 0.5 \leq \delta < 1 \\ -\delta^2 + 3\delta - 1.25 & \text{if } 1 \leq \delta \leq 1.5 \end{cases}$$

We can graph this piecewise function to see how the probability of inducing the target arm changes as the principal applies more incentives δ. The red curve in Figure 4.4 is the piecewise function; a linear graph in blue is shown for comparison:

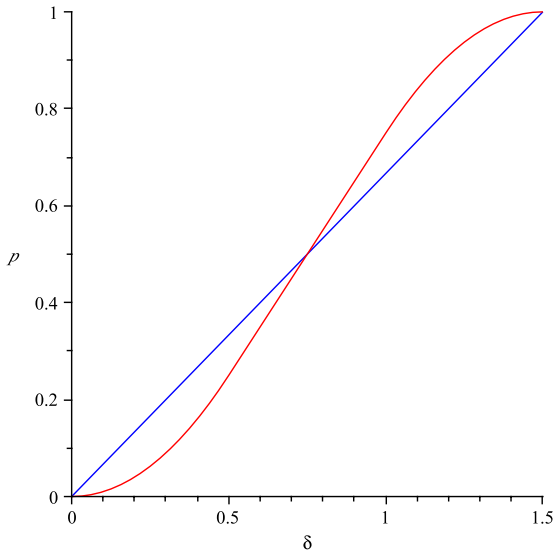

Figure 4.5: Red: $P(g$ selected) v. incentives δ, blue: linear baseline

In this example, we see that when $\delta \leq 0.75$ the principal's influence is less than linear in the amount of incentives; when $\delta \geq 0.75$, the effect is greater than linear. This suggests that as the principal provides more incentive, the effect of the incentives begins to increase. We can think about why the effect of the incentives is non-linear in the probability of inducing the target item. In case (1), as δ increases, the area of the triangle corresponding to the inducible region increases. Because this involves both the height and the width, the change is quadratic. In case (2), as δ changes, we are simply increasing the height of a rectangle with constant width; thus it grows linearly. In case (4), as δ changes, we again see a quadratic growth rate, but this time we are adding trapezoids of decreasing area (the triangular part is getting smaller in both the height and the width).

If the principal wished to conserve incentives to be used in future rounds, he might think about how to effectively provide incentives during the first round. One

potentially good way of assigning incentives in this scenario would be to maximize the ratio of the principal's influence (i.e., increase in probability of inducing the target arm) to the amount of incentives spent. In the case above, this would correspond to $\delta = \frac{7}{6}$.

4.3 A New Objective Criterion

When both the target and non-target arm are varying uniformly over continuous intervals, the problem turns out to be analytically difficult. In the previous section, we begin to consider this question by reasoning about how the principal might wish to provide incentives during the first round in order to maximize the number of pulls of the target arm. In some sense, the principal wishes to get the most out of the incentives that he provides. Since the probability that the agent will select the target arm is a function of the amount of incentives that the principal provides, the principal may want to maximize ratio of probability gained to incentives provided.

Drawing inspiration from this observation, we can reformulate our model to better capture the principal's tradeoff between incentives provided and selections of the target arm. Instead of providing the principal with a total budget over which he has no utility for remaining incentives, we can think of the principal as providing incentives so as to maximize some utility function $u : (\mathbb{R}^+ \cup \{0\}) \times (\mathbb{N} \cup \{0\}) \to \mathbb{R}$ that is based on the total amount of incentives provided and the number of selections of the target arm. This function is decreasing in the total amount of incentives provided and increasing in the number of selections of the target arm. We can fully specify a utility function with just one extra parameter w_g, which we use to denote the value of a selection of the target arm to the principal. Thus, if d is the total amount of incentives that were used to induce k selections of the target arm, the principal's utility is $kw_g - d$. Note that we only need one extra parameter because we can think of scaling incentives to be one unit, making the principal indifferent between w_g units of incentives and one pull of the target arm.

From now on, we assume that the principal wishes to maximize his utility with no budget restriction. This allows us to better capture the tradeoff between providing incentives and inducing a selection of the target arm. Notice that because the budget restriction is lifted, the principal will never be in a scenario where he can no longer provide incentives. Thus, he can always provide incentives when he believes he can benefit by providing them and is not restricted in what he can do by incentives provided earlier. Still, we will see that identifying the optimal policy for the principal remains a complicated challenge.

4.4 A Discrete Model

Based on our experience with the example in the Section 4.1, we see that problem settings involving even the simplest of continuous distributions become analytically and computationally challenging quite rapidly. In the cases that we considered, having just the target arm vary uniformly over a continuous interval led to an analytically intractable problem. Our experience with Section 4.2 further demonstrated the difficulty of tackling settings in which both target and non-target arms are modeled by continuous distributions.

In this section, we introduce a model in which the underlying distributions behind each of the arms is modeled by stationary discrete distributions. We model the principal's incentive design problem as a Partially-Observable Markov Decision Process (POMDP). We discuss some candidate algorithms that we propose would deliver good performance on the principal's problem and run simulations to observe their performance over a variety of problems. Finally, we present the results of these simulations and discuss the insights into good design strategies that we can glean from these results.

4.4.1 Elements of the Model

In our new model, we assume that the following properties hold true (any properties not listed below remain the same as in the original model):

Properties of the Decision Environment:

- *Arms, or actions, available to the agent, K*

  There are only 2 arms, a target arm denoted by g and a non-target arm denoted by h.

- *Distribution behind each arm, O_i*

  The underlying distributions associated with each of these arms is modeled by a discrete distribution consisting of two point masses, corresponding to high and low draws. We let p_i denote the probability of a high draw on on arm i. $1 - p_i$ corresponds to the probability of a low draw. More precisely,

  $$O_g = \begin{cases} \overline{g} & \text{with probability } p_g \\ \underline{g} & \text{with probability } 1 - p_g \end{cases} \qquad O_h = \begin{cases} \overline{h} & \text{with probability } p_h \\ \underline{h} & \text{with probability } 1 - p_h \end{cases}$$

  We assume that the principal knows the distribution behind each of the arms. He can also observe which arm was pulled by the agent in a given round, but does not see whether the agent experienced a high draw or a low draw. For ease of notation, we let g_i denote the realized value of arm g the i^{th} time it was selected. This is equivalent to saying $g_i = o_g^t$ if the i^{th} selection of g occurred at time t. We use the same notation for h_i.

Properties of the Agent:

- *Belief state over all available arms, $x(t)$*

  The agent's belief state over g and h can be thought of as consisting of sequences of values g_i and h_i that the agent previously experienced. In this setting, we assume that each agent's belief state over g and h is initialized to be arrays

representing one outcome from their respective distributions. Thus, $x_g(1)$ can be thought of as an array consisting of only g_0. We assume that the agent's valuation function for each of the arms takes the belief state and outputs the **empirical mean** of the values in the array.

- *Agent function: how decisions are made at each time step,* $\mathcal{F}^t$

  To emphasize, we assume that the agent function $\mathcal{F}$ is specified as follows for all times t:

  $$\mathcal{F}(x(t), \Delta(t)) \in \arg\max_{i \in \{g,h\}} [v_i(x_i(t)) + \Delta_i(t)]$$

  Thus, the agent will always choose the arm with the highest perceived value plus added incentive.[2]

- *Belief update function: how realized outcomes are incorporated into beliefs,* $\mathcal{U}^t$

  As we noted when specifying the belief state, if the agent selected arm i at time t, then the belief function appends the outcome o_i^t to the current array specified by the belief state $x_i(t)$.

Before we introduce the properties of the principal in our new model, we discuss some of the benefits that we enjoy in moving to a discrete model. First, notice that we can enumerate all of the possible beliefs an agent could have at any given time. As an example, since we assume that the agent is initialized with values g_0 and h_0, the principal knows that at time $t = 1$ the agent could be in one of four possible belief states:

$$x(1) = (x_g(1), x_h(1)) = \begin{cases} ([\overline{g}], [\overline{h}]) & \text{with probability } p_g p_h \\ ([\underline{g}], [\overline{h}]) & \text{with probability } (1 - p_g) p_h \\ ([\overline{g}], [\underline{h}]) & \text{with probability } p_g (1 - p_h) \\ ([\underline{g}], [\underline{h}]) & \text{with probability } (1 - p_g)(1 - p_h) \end{cases}$$

[2] As before, we assume that ties are broken in favor of the target arm g. We can continue thinking about this as the principal always placing an extra ϵ on the target arm to break the tie.

In comparison to the continuous model, this greatly simplifies the principal's information at the beginning of his interaction with the agent. Importantly, this will allow the principal to enumerate the different incentive policies that he should consider. We now define the principal's properties more precisely.

Properties of the Principal:

- *Vector of the agent's possible belief states, $Y(t)$*

  The principal maintains a vector of the agent's possible belief states $Y(t)$. This vector is indexed by $i \in \mathbb{N}$. Each element of $Y(t)$, denoted $y_i(t)$, is a tuple consisting of a possible belief state that the agent could have at time t, denoted $\hat{x}^i(t)$, and the probability p_i that the agent is in state $\hat{x}_i(t)$. To illustrate, we think of $Y(1)$ as follows:

i	1	2	3	4
$\hat{x}^i(1)$	$([\overline{g}], [\overline{h}])$	$([\underline{g}], [\overline{h}])$	$([\overline{g}], [\underline{h}])$	$([\underline{g}], [\underline{h}])$
p_i	$p_g p_h$	$(1-p_g)p_h$	$p_g(1-p_h)$	$(1-p_g)(1-p_h)$

  Note that we would like the **belief state vector** $Y(t)$ to contain the agent's actual belief state at time $x(t)$ for all times t. That is, we require that there exists some i such that $\hat{x}^i(t) = x(t)$. We can ensure that this will always be the case by updating each of the states and probabilities appropriately. To demonstrate, suppose the agent selected g in the first round in the absence of any incentives. Observing this, the principal's new belief state vector $Y(2)$ is as follows:

i	1	2	3	4	5
$\hat{x}^i(2)$	$([\overline{g}, \overline{g}], [\overline{h}])$	$([\overline{g}, \underline{g}], [\overline{h}])$	$([\underline{g}, \overline{g}], [\overline{h}])$	$([\underline{g}, \underline{g}], [\overline{h}])$	$([\overline{g}, \overline{g}], [\underline{h}])$
p_i	$p_g p_h p_g$	$p_g p_h (1-p_g)$	$(1-p_g)p_h p_g$	$(1-p_g)p_h(1-p_g)$	$p_g(1-p_h)p_g$

i	6	7	8
$\hat{x}^i(2)$	$([\overline{g}, \underline{g}], [\underline{h}])$	$([\underline{g}, \overline{g}], [\underline{h}])$	$([\underline{g}, \underline{g}], [\underline{h}])$
p_i	$p_g(1-p_h)(1-p_g)$	$(1-p_g)(1-p_h)p_g$	$(1-p_g)(1-p_h)(1-p_g)$

  Notice that the potential number of states in the next round doubled because for each potential state $\hat{x}^i(1)$, the agent could have received a high draw from O_g

or a low draw from O_g. Since we began with all possible states and our update process ensures that we take every possibility into account, we know that the agent's actual belief state must be contained in the vector. Furthermore, given a sequence of selections of length t, the number of belief states in the vector $Y(t)$ is bounded above by 2^{t+1}. Later we will discuss how the principal can keep his belief state vector a manageable size, while still ensuring that it contains the agent's actual belief state.

- *Incentives provided at each round, $\delta \in D(t)$*

  Given a belief state vector $Y(t)$, the principal can enumerate a candidate set of incentives $D(t)$ that he should consider providing during round t. We think of this as follows. For each belief state $y_i(t)$ of the state vector, we can compute the difference between the agent's value for the target arm and the non-target arm if he were in that state. Let this quantity be denoted d_i. It is computed as $v_h(\hat{x}_h^i(t)) - v_g(\hat{x}_g^i(t))$. Now, we can go through our belief state vector $Y(t)$ and compute d_i for each of the belief states i. Non-negative values of d_i can be thought of as potential incentives to be placed on the target arm, while negative values of d_i can be thought of as potential incentives to be placed on the non-target arm.[3] We set our candidate list of incentives $D(t)$ to be all values of d_i along with 0 and use δ to refer to the incentive that the principal assigns at the specified round. Because we only have two arms, we use $\delta < 0$ to refer to incentives placed on the non-target arm. We will later demonstrate that the principal need not consider any other incentives aside from those on the candidate list.

- *Value for selections of the target arm, w_g*

  The principal views a selection of the target arm as being just as valuable as

[3]Note that we have to be slightly careful when we set our incentive $\delta = d_i < 0$ since we have assumed up until this point that ties are broken in favor of the target arm. We assume that if the principal is providing incentive to the non-target arm, then he provides $|d_i| + \epsilon$ so as to break the tie in favor of the non-target arm. This will only apply for negative values of d_i that we consider in our candidate list of incentives.

w_g units of incentive.

As we can see, this model greatly simplifies the principal's decision problem by creating a set of effective incentives $D(t)$ at each time step. For example, regardless of what the parameter values are, the principal's action at the first step will be selected from a candidate list of at most five different incentives (one for each of the states and 0). This is much more tractable than in either of the continuous models that we considered earlier.

This model continues to capture much of the richness of our problem setting. For instance, if we return to the restaurant example from earlier in this chapter, we can think of Al's as obtaining fresh ingredients from a reputable supplier with some probability and Al's obtaining spoiled ingredients from questionable suppliers with some other probability. This would still correspond to a realistic scenario for which we might want to determine the optimal way in which incentives should be provided over a series of rounds.

4.4.2 Updating the Belief State Vector

In this subsection, we discuss how the principal can pare down the size of belief state vector $Y(t)$ that represents all possible belief states that the agent could be in at time t, given the sequence of observations. We first consider a different way to think about the agent's belief states that helps us collapse multiple belief states into one belief state. We then incorporate the information that the principal gains from observing the actions that the agent selects at each round. Finally, we study how these methods make the principal's belief state vector tractable.

First, notice that we can fully represent the agent's belief state by a 4-tuple consisting of the total number of high draws from g and h as well as the total number of pulls of each of the arms. Consider this representation:

$$x(t) = (m_g, m_h, n_g, n_h)$$

where m_i and n_i represent the number of high draws and the total number of observations from arm i, respectively. From this representation, we can calculate $v_i(x_i(t)) = (\overline{i} - \underline{i})\frac{m_i}{n_i} + \underline{i}$. This corresponds to the empirical mean. Since the principal is only concerned with the values of the arms in each of the belief states and how they might update in future rounds, he can collapse all potential belief states with the same 4-tuple into a single belief state.[4] This process can be applied to all belief states in the vector $Y(t)$. Notice that this disregards the order in which the high and low selections were obtained, since the sequence of pulls does not change the agent's current perceived value, which is based on the empirical mean.

We now discuss how the principal can incorporate the information obtained from observing the agent's selections at each round. Consider the following example, which begins with $Y(1)$ that is specified as follows:

i	1	2	3	4
$\hat{x}^i(1)$	$(1,1,1,1)$	$(0,1,1,1)$	$(1,0,1,1)$	$(0,0,1,1)$
p_i	$p_g p_h$	$(1-p_g)p_h$	$p_g(1-p_h)$	$(1-p_g)(1-p_h)$
d_i	$\overline{h} - \overline{g}$	$\overline{h} - \underline{g}$	$\underline{h} - \overline{g}$	$\underline{h} - \underline{g}$

Assume that the principal does not provide any incentives during round 1 and observes that the target arm g was selected. What information can the principal extract from this? Because we know that the agent selects the arm with the highest perceived value plus added incentive, we can conclude that $v_g(x_g(t)) \geq v_h(x_h(t))$. Since we know that the agent's true belief state $x(t)$ has this property, we can go back to $Y(1)$ and eliminate all entries i where $\hat{x}^i$ does not satisfy this condition. Recall that $d_i = v_h(\hat{x}^i_h(t)) - v_g(\hat{x}^i_g(t))$. Thus, this elimination operation can easily be done by removing any entries i where $d_i > 0$. We can do this because we know that the agent's true belief state was not captured in any of these entries, because he could not have selected g under the belief states associated with them. After each observation, the principal can use this process to decrease the number of potential

[4]Note that this does not mean that we should collapse all states with the same d_i. This is because the principal is still concerned about the update dynamics, which are different depending on the specific values of the 4-tuple, since the agent is taking the empirical mean.

belief states in his belief state vector $Y(t)$.

As another example, consider the case in which the principal provides incentive δ to the target arm during round 1 and selects the target arm g. From this observation, we know that $v_g(x_g(t)) + \delta \geq v_h(x_h(t))$, i.e., $v_h(x_h(t)) - v_g(x_g(t)) \leq \delta$. Thus, we can go ahead and eliminate all entries i in which $d_i > \delta$ from $Y(1)$.

Notice that these methods of updating the belief state vector will never eliminate the agent's true belief state from the vector. This is because belief states are only eliminated if they are inconsistent with the agent's observed action.

Once we eliminate an entry i from the belief state vector, the probability p_i associated with that state before the inference (and thus elimination) is eliminated. Once we have removed all of the inconsistent belief states, we must rescale the probabilities p_j on all of the remaining entries j. This can be done by dividing each entry by the summation $\sum_j p_j < 1$, which corresponds to the correct scaling of the probability estimates. We can think of this as a conditional probability. Given that all of the entries i are invalid, what is the probability that the agent's belief state corresponds to $\hat{x}^j$?

To see how well these methods work in practice, we performed the principal's belief state vector update over all valid sequences of twenty rounds with and without the incorporation of this inference. We set $\underline{g} = 0, \overline{g} = .75, \underline{h} = .25, \overline{h} = 1$ and $p_g = p_h = 0.5$. In this setting, no incentives are being provided. In Figure 4.6, we use box plots to compare the number of states as a function of the number of rounds of interaction.

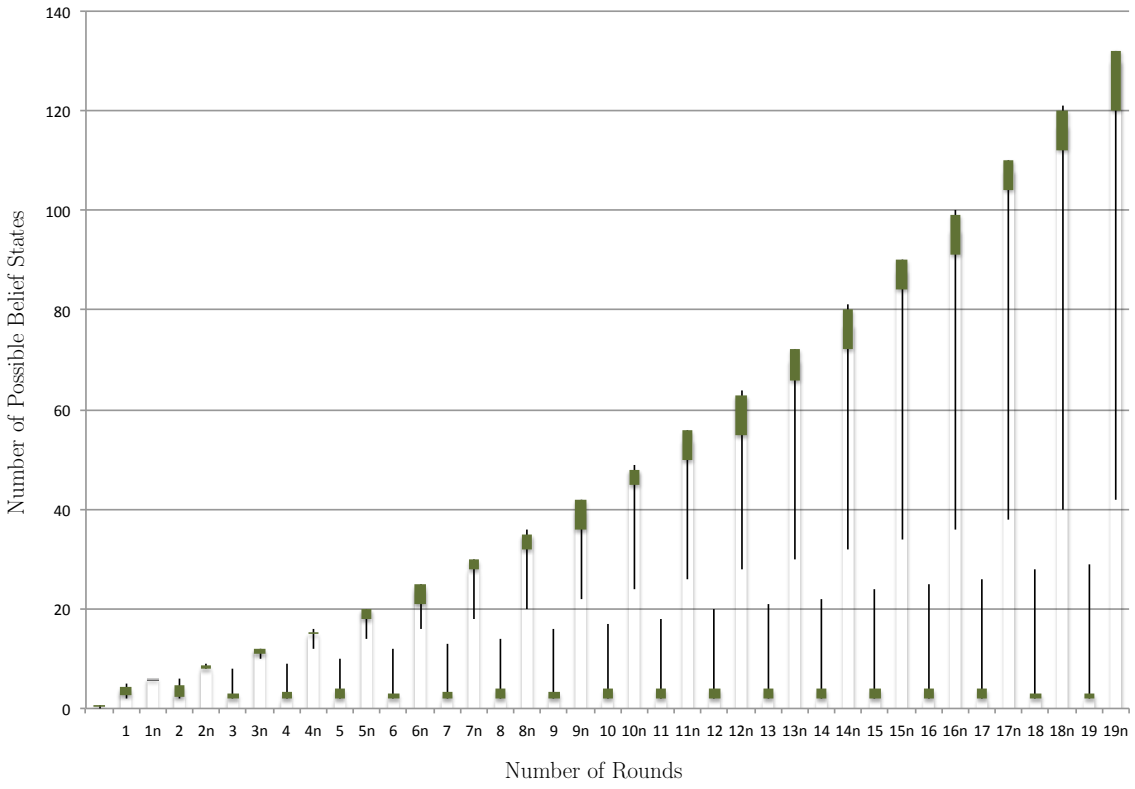

Figure 4.6: Box plots comparing the number of states with inference and without inference

We have already established that the total number of belief states is 2^{t+1} at round t if we do not collapse belief states or perform inference about the agent's belief states. The box plot above demonstrates the decrease in the number of possible belief states when moving from only collapsing similar belief states but no inference (denoted n) to collapsing belief states and inference. We see that in this case, the number of belief states when we perform inference remains quite tractable, with more than 75% of the valid sequences having fewer than 5 states during all rounds measured. In contrast, when we do not perform inference, we see that the number of belief states begins to increase, with approximately 75% of the valid sequences leading to belief state vectors of size at least 120 after 19 rounds have passed.

This result is promising. It means that the principal can tractably keep track of the agent's potential belief states. Based on our results, more than the 75% of the

time, the principal will only need to keep track of fewer than 5 states. In particular, the number of states at the 75^{th} percentile does not seem to grow. This means that after a given number of rounds, the principal can, with high probability, maintain a small belief state vector.

It is important to note that with changes in the values of the parameters and incentives provided, the size of the principal's belief state vector may be larger or smaller than in this example. Furthermore, we demonstrated that the number of states can also depend on the incentives that the principal provides at each round. As an extreme example, suppose the principal provides $\delta = \overline{h} - \underline{g}$ worth of incentive at each round. Because this value will induce the agent to select g at each round, regardless of his belief state, the principal will obtain no information about the agent's actual belief state.

This example encourages us to consider how to design algorithms that might perform well in practice on the principal's incentive design problem. In particular, it opens up the possibility of a class of algorithms that perform a limited-depth look ahead into possible future belief state vectors.

4.4.3 Candidate List of Incentives

In this section we will prove that the principal need only consider actions taken from the candidate list of incentives, $D(t)$, which he can calculate when given a belief state vector $Y(t)$. The intuition behind this proof is that regardless of the agent's current belief state, any incentive $\delta \in D(t)$ will lead to the same information state as any other incentive $\delta' \notin D(t)$ while providing weakly greater utility, regardless of the action that the agent ends up taking.

Proposition 1. Given an incentive policy $\Delta = (\delta_1, \delta_2, \ldots, \delta_t)$ where at least one $\delta_i \notin D(i)$, the principal can perform at least as well on the same agent input by using some incentive $\Delta' = (\delta'_1, \delta'_2, \ldots, \delta'_t)$ where $\delta'_i \in D(i), \forall i$.

Proof. It suffices to show that given any incentive policy $\Delta = (\delta_1, \delta_2, \ldots, \delta_t)$, where at least one $\delta_i \notin D(i)$, the principal can perform at least as well by replacing one such entry with $\delta_i' \in D(i)$. To do this, we must show that the principal gains the same information with δ_i' as he does with δ_i, and that he provided less incentives in doing so.

Given $\delta_i \notin D(i)$, we will exhibit a $\delta_i' \in D(i)$ that satisfies both properties stated above. Assume that have $Y(i)$ so we can construct $D(i)$. If $\delta_i \geq 0$, then we let δ_i' be equal to the largest element in $D(i)$ such that $\delta_i' < \delta_i$. If $\delta_i < 0$, then we let δ_i' be equal to the smallest element in $D(i)$ such that $\delta_i' > \delta_i$. Note that because $0 \in D(i)$, this means that δ_i' must be non-negative. It is easy to see that such δ_i' satisfies our second criteria above.

Now we show that δ_i' will lead to the same information state as δ_i. Assume that we are given $Y(i)$. We let $Y'(t)$ denote the principal's belief state vector at times t after having provided δ_i' instead of δ_i. We show that $Y'(t) = Y(t)$. Suppose not. Then there exists some $j > i$ such that $Y'(j) \neq Y(j)$. But this cannot be the case because at time i, the agent must choose the same action regardless of whether or not δ_i or δ_i' was applied, since δ_i' was defined as the least incentive that will prompt the agent to choose the same action. Thus at time i, we have $Y(i+1) = Y'(i+1)$, because no extra information is gained. Since the remaining incentives are all the same, we can conclude that $Y'(j) = Y(j)$ for all $j > i$. □

The intuition behind this proof is that when we perform an update on the belief state vector, we are doing it such that we incorporate *all* of the information obtained from the agent during that round. Information about round i cannot help during round $j > i$ because it has been completely integrated into the belief state vector for all times after i.

4.4.4 Partially-Observable Markov Decision Process

We can use the framework of Partially-Observable Markov Decision Processes (POMDPs) to help think about the principal's problem [5, 12]. The principal maintains a vector of possible belief states that the agent can have, along with all of probabilities that each belief state represents the agent's actual beliefs. The agent's actual belief state represents the underlying state of the POMDP.

The principal can take actions in this environment by providing some amount of incentive at each round. Assume that he provides incentive δ. Based on the value of δ, the agent will take an action, thus providing the principal with information about the agent's belief state. Furthermore, the principal will receive a reward equal to $w_g - \delta$ if g is selected or 0 otherwise.

The principal, faced with this POMDP, must decide what policy would maximize his utility. Thus, the problem at hand can be formulated as a finite-horizon POMDP with no discounting. However, POMDPs are intractable in general. In order to make progress on solving the principal's problem, we will need to utilize some of the problem's structural properties.

4.4.5 Candidate Algorithms

We now consider several algorithms that might provide good performance on the principal's incentive design problem.

Myopic

One natural algorithm that is computationally efficient involves the principal maximizing his expected utility at each time step. He does this by iterating through his candidate list of incentives and computing the expected value for each of them. If we assume for now that the principal only considers placing incentives on the target arm, then we see that if the agent selects the non-target arm, the principal's utility

does not change since he gains no utility from the non-target arm and does not have to pay incentives. Suppose that the agent selects the target arm with probability p^δ when the principal provides incentive δ on the target arm. Thus, his expected utility for providing incentive δ is just $p^\delta(w_g - \delta)$. If the principal wishes to maximize his utility during the current round, he can select δ such that $p^\delta(w_g - \delta)$ is maximized. We present the algorithm more precisely as follows:

Algorithm 1 MYOPIC($Y(t)$)

Given belief state vector $Y(t)$ with all d_i calculated, w_g
Sort $Y(t)$ in terms of decreasing d_i, with largest d_i on the left (if there is a tie, then collapse the two states together and give the resultant state the sum of the two probabilities)
Relabel the indices of entries accordingly (d_1 is the highest)
$\delta^* \leftarrow$ NULL (initialize incumbents)
$u^* \leftarrow -\infty$
$p^\delta \leftarrow 1$ (p^δ denotes the probability of inducing the target given δ)
for entries i in $Y(t)$ such that $d_i > 0$, and once more with $d_i = 0$ **do**
    $\delta \leftarrow d_i$
    $u \leftarrow p^\delta(w_g - \delta)$
    if $u > u^*$ **then**
        $\delta^* \leftarrow d_i$ (update incumbents)
        $u^* \leftarrow u$
    end if
    $p^\delta \leftarrow p^\delta - p_i$ (update p^δ for next incentive value)
end for
return (δ^*, u^*)

We call this algorithm MYOPIC because the principal is only maximizing over the next time step. He treats the current round as the last step and would simply like to assign the incentive that maximizes his utility over this one round. Given this, we present the following proposition.

Proposition 2. MYOPIC will never provide incentives to a non-target arm.

Proof. Consider $\delta \in D(t)$ such that $\delta < 0$. This corresponds to $|\delta|$ worth of incentive placed on the non-target arm. We show that at time t, providing $\delta' = 0$ will provide

at least as much expected utility as providing δ. Because $\delta' \in D(t)$, we know that MYOPIC always has a better choice than δ.

Case 1. Suppose the agent selects the non-target arm. δ leads to some utility $u < 0$ because the principal paid out $|\delta|$, and the non-target arm has no utility to the principal. In this case $\delta' = 0$ leads to utility $u' = 0$.

Case 2. Suppose the agent selects the target arm. Both δ, δ' lead to utility w_g.

Notice that Case 1 will occur at least as frequently under δ as it does under δ', since the principal is incentivizing the non-target arm by applying δ. Since the utility under Case 2 is the same, and the $u < u'$ under Case 1, we see that the expected utility when providing $\delta' = 0$ is at least as high as when providing $\delta = 0$. □

Thus, MYOPIC can stop once it sees values of $D(t)$ that are negative. Still, we note that the bottleneck for this algorithm is in the sorting. If we assume that the belief state vector given to the algorithm is sorted, then this algorithm runs in $O(n)$ time, where n is the number of entries in the belief state vector.

Look Ahead

When formulating MYOPIC, we only considered the principal's utility during that round. In doing so, we neglected to consider the information included in the resultant belief state vector. We thus consider a look ahead algorithm in which the principal simulates all possible sequences of incentives for a depth of k rounds. We can think about this as follows: the algorithm takes the current belief state vector and treats it as the root of a tree of depth k. The nodes of this tree are belief state vectors. The algorithm seeks to give a value to each of the nodes within this tree, starting with the leaves. The value that the algorithm assigns to a specific node corresponds to the maximum expected utility that the algorithm believes is possible from that node. For now, we will assume that the algorithm sets the value of the leaves to be 0. The algorithm then assigns values to nodes by proceeding upward from the leaves. We

specify the algorithm more precisely as follows:

Algorithm 2 LOOK-AHEAD-$k(Y(t), k, R)$

1: Given belief state vector $Y(t)$ with all d_i calculated, w_g, look ahead depth k, remaining number of rounds R
2: **if** $k > R$ **then**
3: return LOOK-AHEAD-$R(Y(t), R, R)$
4: **end if**
5: **if** $k = 1$ **then**
6: return MYOPIC$(Y(t))$
7: **else**
8: Let D be the set of all candidate incentives
9: $\delta^* \leftarrow$ NULL (initialize incumbents)
10: $u^* \leftarrow -\infty$
11: **for** $\delta \in D$ **do**
12: Compute $Y_g^\delta(t+1)$, $Y_h^\delta(t+1)$, and p^δ
13: $u_g \leftarrow$LOOK-AHEAD$(Y_g^\delta(t+1), k-1)[1] + w_g$
14: $u_h \leftarrow$LOOK-AHEAD$(Y_h^\delta(t+1), k-1)[1]$
15: **if** $\delta \geq 0$ **then**
16: $u_g \leftarrow u_g - \delta$
17: **else**
18: $u_h \leftarrow u_h - \delta$
19: **end if**
20: $u \leftarrow p^\delta u_g + (1 - p^\delta) u_h$
21: **if** $u > u^*$ **then**
22: $\delta^* \leftarrow \delta$ (update incumbents)
23: $u^* \leftarrow u$
24: **end if**
25: **end for**
26: return (δ^*, u^*)
27: **end if**

In line 12, we use $Y_i^\delta(t+1)$ to denote the updated belief state vector in the next time step if the principal provided incentive δ at time t and observed a selection of arm i. p^δ denotes the probability that the target arm g will be selected when the principal provides incentive δ to the target arm at time t. Notice that the conditional in line 15 allows us to differentiate between incentives placed on the target arm and incentives placed on the non-target arm. Furthermore, notice that LOOK-AHEAD-1 is equivalent to MYOPIC because we assumed that the values at each of the leaves is

0. This corresponds to the first step of valuations propagating up our tree of belief state vectors. It associates a value and action with each node at depth $k-1$ in the tree of belief state vectors.

At the next level up, the algorithm labels each node at depth $k-2$ in the tree of belief state vectors with the action that yields the maximum expected utility as well as the numerical value that corresponds to that expected utility. This is done by extracting the action at that round that leads to the maximum expected value and taking the expectation of the children associated with that action. This process repeats until all of the nodes, including the root, have been labeled. LOOK-AHEAD-k thus returns the first incentive that the principal should provide so as to maximize expected utility over the next k rounds.

At every time step, the principal can run this algorithm to return the action he should take if he wishes to maximize his utility over the next k steps. Notice that we do not have to worry about overstepping the total number of rounds R that is given to us because the algorithm rescales to make sure that it never looks past the total number of allowable rounds. Thus, we can be sure that the principal will be providing the optimal amount of incentives within the last k rounds.

Note that in practice we would want k to be small. This is because LOOK-AHEAD-k requires an amount of memory exponential in the number of actions in the principal's candidate list of incentives. As a counterpoint, we note that if we could run a LOOK-AHEAD-R on a problem with a total number of steps R, then the algorithm would search through all possible incentive provision policies and return the optimal incentive at each step. For large R, this is intractable, and so we must reduce our look ahead depth in order to gain some traction.

In our simulations, we will test the performance of this algorithm on different values of k. Intuitively, we expect that larger values of k will yield better performance than smaller values, at the cost of being more computationally intensive. This is because increasing the depth allows the algorithm to see further ahead and be aware

of a larger number of possible outcomes.

Look Ahead with Myopic Markov Chain Leaf Evaluation

In LOOK-AHEAD-k, we valued all of the leaves at depth k of the tree of belief state vectors as 0. In this section, we propose a heuristic which we will later test empirically.

Instead of valuing each of the leaves in the tree as 0, which disregards the information captured by the belief state vectors at the leaves, we instead return the expected utility if the principal were to follow MYOPIC for all rounds between the leaf and the final horizon. We can do this by creating a Markov Chain where at each time step we simulate what would happen if the principal chose the myopic action. We then set the value of the leaf as the expected utility if the principal were to use MYOPIC for all of the remaining rounds.

Note that this Markov Chain simulation is still exponential in the number of remaining rounds. However, the base of the exponent in this case is 2, since we are assuming the action that the principal takes is known. This is more tractable in practice than increasing the depth of the look ahead, for which the base of the exponent is twice the number of actions at each step.

We can construct this new algorithm LOOK-AHEAD-MYOPIC-MARKOV-$k(Y(t), k, R)$ by changing lines 5 and 6 in Algorithm 2 to the following:

if $k = 0$ **then**

return MYOPIC-MARKOV$(Y(t), R - k)$

where MYOPIC-MARKOV$(Y(t))$ returns the myopic action at that round and the expected utility if MYOPIC was used for all remaining rounds. Note that we can make this algorithm somewhat more tractable by restricting the depth of the Markov Chain simulation to some parameter that is independent of the remaining number of rounds. This would still provide some of the benefit of the entire Markov Chain while greatly increasing the tractability of the simulation.

4.4.6 Walkthrough of Algorithms

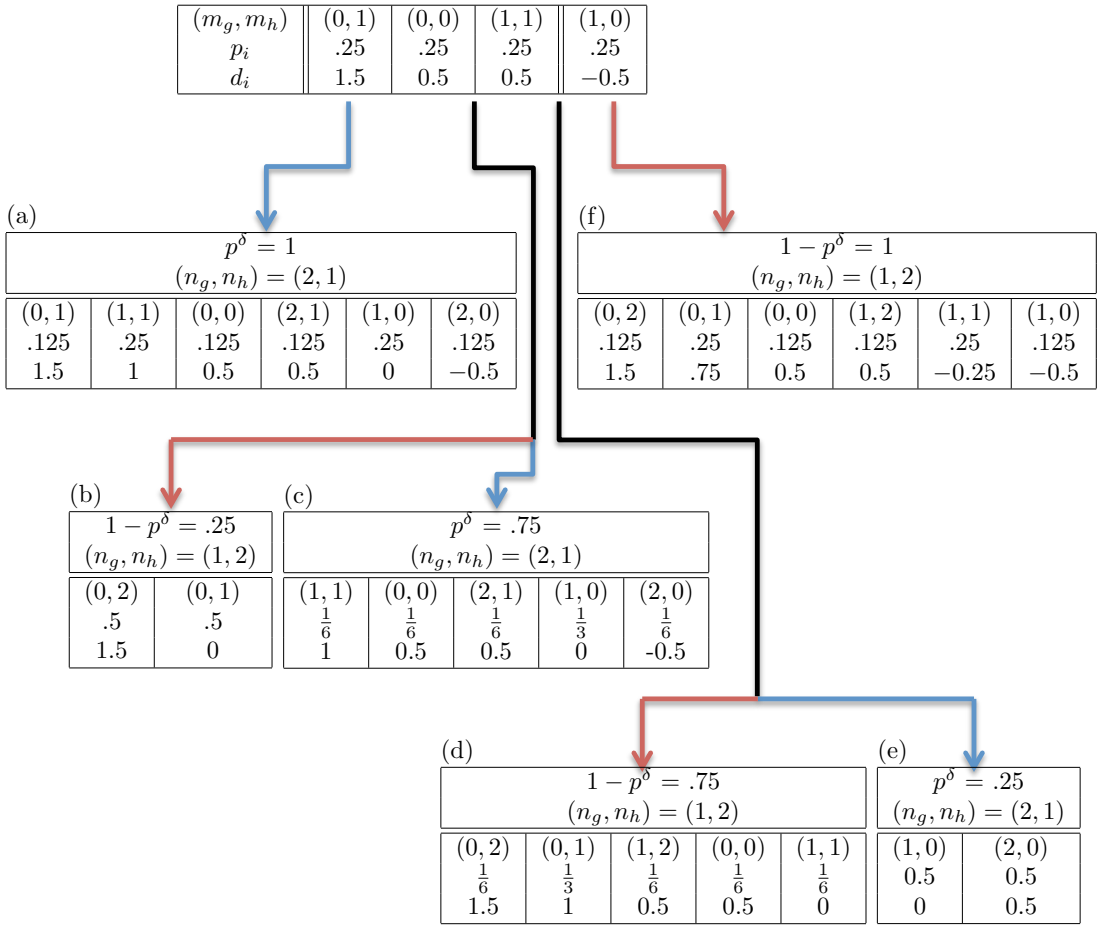

Figure 4.7: One step of the look ahead algorithm with the following parameters: $\underline{g} = 0, \overline{g} = 1, \underline{h} = 0.5, \overline{h} = 1.5, p_g = p_h = 0.5$; red arrows correspond to selections of the non-target arm h and blue arrows correspond to selections of the target arm g.

We will walk through both MYOPIC and LOOK-AHEAD-2 on the example shown in Figure 4.7. We assume that we are solving the principal's incentive design problem for a horizon of two steps; that is, the principal can provide incentives at two rounds and wishes to provide incentives to maximize his utility. We will also assume that the principal's value for selecting the target arm is $w_g = 0.7$.

Myopic

We begin with MYOPIC. MYOPIC begins by looking at the belief state vector $Y(1)$, which is the root in Figure 4.7, and considers the expected utility for each of the incentives in the candidate list. Here, we can see that the candidate list $D = \{1.5, 0.5, 0, -0.5\}$ by reading off the d_i in $Y(1)$ and including 0. Since we established that MYOPIC would never consider placing incentives on the non-target arm, we only need to consider the non-negative values $\delta \in D$. Starting from the left-most entry of $Y(1)$, MYOPIC computes $p^{\delta}(w_g - \delta)$ for all non-negative $\delta \in D$. Thus, MYOPIC considers the following incentive-utility pairs:

δ	Expected Utility
1.5	$w_g - 1.5$
.5	$.75(w_g - 0.5)$
0	$.25w_g$

Substituting $w_g = 0.7$, we see that MYOPIC will return $\delta = 0$, which maximizes the expected utility. With this incentive, the agent will select the non-target arm with probability 0.75 and the target arm with probability 0.25. This can be seen in Figure 4.7, with the former case corresponding to (d) and the latter case corresponding to (e). We consider each of these two cases.

Case 1. Suppose $\delta = 0$ at round 1. Assume the agent selected the non-target arm h at round 1. The principal observes this, and performs MYOPIC on (d), which considers the following incentive-utility pairs:

δ	Expected Utility at (d)
1.5	$w_g - 1.5$
1	$\frac{5}{6}(w_g - 1)$
0.5	$.5(w_g - 0.5)$
0	$\frac{1}{6}w_g$

In this case, MYOPIC selects $\delta = 0$, which again corresponds to the maximum expected utility. Thus the expected utility in Case 1 would be $.11\bar{6}$ during the second round. Since the target arm was not chosen in the first round, the first round utility is 0. Thus, the total expected utility in Case 1 is $.11\bar{6}$.

Case 2. If instead the agent selected the target arm g at round 1, then the principal's new belief state vector would be (e) at round 2. The principal performs MYOPIC on (e). The principal has no choice here as to what incentive he should provide during round 2, since he is certain that the agent will select the target arm g in the absence of incentives. Thus, his total utility in Case 2 is $w_g = 0.7$ from the first round and $w_g = 0.7$ from the second round.

Since we know that Case 1 happens with probability .25 and Case 2 happens with probability .75, we say that MYOPIC returns expected utility equal to $.75(.11\bar{6}) + .25(1.4) = .4375$ to this problem. As we consider our next algorithm, it will be important to keep in mind that the first action selected by MYOPIC was $\delta = 0$.

Look-Ahead-2

Because LOOK-AHEAD-2 is recursive, it is easiest to illustrate what the algorithm is doing by first computing LOOK-AHEAD-1 over all possible belief state vectors during the second round. Notice that because the total time horizon is 2, we can extract no more value from the leaf nodes. Thus, regardless of how we decide to evaluate the leaf nodes for this algorithm, all of the values at each of the leaves of depth two from the root node will be zero[5], and LOOK-AHEAD-1 is equivalent to Myopic.

Thus, we look at all of the possible belief states at round two ((a)-(f)) and consider what MYOPIC would do at each of them. We enumerate the tables in Appendix B and note the results here:

Belief State Vector	Optimal δ	Expected Utility
(a)	0	.2625
(b)	0	.35
(c)	0	.35
(d)	0	$.11\bar{6}$
(e)	0	.7
(f)	0	.2625

[5]This observation includes the myopic Markov Chain simulation, as it would be simulation for $2 - 2 = 0$ rounds.

LOOK-AHEAD-2 takes these expected utilities and assigns them to the appropriate nodes in the tree of belief state vectors that can be seen in Figure 4.7. It now considers the following table:

δ	p^δ	Expected Utility if g	Expected Utility if h	Total Expected Utility
1.5	1	(a) = .2625	n/a	$0.7 - 1.5 + (a) = -0.5375$
0.5	.75	(c) = 0.35	(b) = 0.35	$.75((c) + .7 - .5) + .25(b) = 0.5$
0	.25	(e) = 0.7	(d) = $.11\overline{6}$	$.25((e) + .7) + .75(d) = 0.4375$
-0.5	0	n/a	(f) = .2625	$0 - 0.5 + (f) = -.2375$

Now, it selects the incentive δ that maximizes the total expected utility over the next two rounds. To maximize total expected utility, he should provide $\delta = 0.5$ at the first step. Notice that this is different from what MYOPIC would have done! This is very promising because it shows that even in small examples, there is a potentially large benefit to providing incentives in a sophisticated manner. Also notice that in computing the table above, we also computed the expected value of the incentive strategy that MYOPIC ended up using.

For this problem, we know that LOOK-AHEAD-2 will return the optimal way to design incentives because we have limited the total horizon of the problem to two rounds of interaction. Thus, the algorithm searches through all possible incentive strategies and returns the first action that the principal should take in order to maximize his expected utility. In general, we note that LOOK-AHEAD-k will provide the optimal way to design incentives in a setting with k steps. It is important to remember, however, that this would require an amount of space exponential in the number of actions at each step. Thus, we cannot optimally solve the principal's incentive design problem in general.

This example does not illustrate the difference between the types of leaf evaluation that LOOK-AHEAD can utilize. This is because we assume that round 3 is the end, and the principal has no value over the information state at which he ends up because he does not care about future rounds.

4.5 Experiments

In this section, we discuss the considerations that went into the design of our simulations. We then look at the results that we obtain.

4.5.1 Experiment Setup and Design

We sought design experiments that would help us answer the following questions:

- How much, if any, of a performance benefit does MYOPIC yield over the trivial incentive policy where the principal never intervenes?
- In what cases can the more computationally intensive algorithms such at LOOK-AHEAD and LOOK-AHEAD-MYOPIC-MARKOV provide significant increases in performance over MYOPIC?
  - In these cases, what sorts of design decisions are driving this performance increase?
  - Are there bad decisions that MYOPIC seems to be making? Is it overspending or underspending?
- Is there some way to characterize scenarios in terms of how effective sophisticated design strategies might be just by looking at the parameter values?

In determining what problems to consider, we drew our inspiration from real life scenarios. We first decided to look at the case where both the target and the non-target arms had identical underlying distributions. This corresponds to the case in which the agent must choose between two decisions that are similar in quality. This would model settings in which, in the absence of a principal, the agent might be indifferent between two types of cereal or two restaurants.

In the second problem, we consider the case in which the target arm is at a severe disadvantage compared to the non-target arm. This models settings in which

the target arm is an underdog and must be boosted above the competition via the use of incentives. This could potentially correspond to cases in which a company would like to make up for an inferior product through external means. For instance, an automobile company might want to compensate for its lack of R&D funding by providing incentives to consumers.

We set up our experiments as follows:

Experiment 1. We fix the following parameters: $\underline{g} = \underline{h} = 0, \overline{g} = \overline{h} = 1, p_g = p_h = 0.5$. The total number of rounds is $R = 12$. We vary the parameter w_g using values from the following set: $\{0.3, 0.5, 0.7\}$.

Experiment 2. We fix the following parameters: $\underline{g} = 0, \overline{g} = 1, \underline{h} = 0.5, \overline{h} = 1.5, p_h = 0.5, w_g = .5$. The total number of rounds is $R = 12$. We vary the parameter p_g using values from the following set: $\{0.3, 0.5, 0.7\}$.

Experiment 3. We fix the following parameters: $\underline{g} = 0, \overline{g} = 1, \underline{h} = 0.5, \overline{h} = 1.5, p_g = p_h = 0.5$. The total number of rounds is $R = 12$. We vary the parameter w_g using values from the following set: $\{0.3, 0.4, 0.45, 0.5, 0.55, 0.6, 0.7\}$.

For each of these experiments, we create a set of 100 randomly generated agent inputs that abide by the parameters specified. For each experiment, we fix the set of agent inputs that will be used in each. We then simulate Myopic, Look-Ahead-2, Look-Ahead-3, Look-Ahead-Myopic-Markov-2, and Look-Ahead-Myopic-Markov-3 on each agent input in this set and output the average utility that the principal obtained after 12 rounds. We also do the same for the trivial incentive policy which provides no incentive at any round.

We implemented all of the elements of our model, as well as Myopic, Look-Ahead-k, Look-Ahead-Myopic-Markov-k in Python and ran these simulations on the remote server equipped with 24GB of RAM and 6 cores, each running at 2.4GHz.

4.5.2 Simulation Results

Experiment 1. Figures 4.8 and 4.9 summarize our results from Experiment 1.

w_g	No incentive	MYOPIC	LA-2	LA-3	LAMM-2	LAMM-3
0.3	1.91	2.00	2.10	2.11	2.12	2.12
0.5	3.18	3.53	3.82	3.83	3.78	3.81
0.7	4.45	5.45	5.55	5.53	5.49	5.56

Figure 4.8: Average utility over the same set of 100 agent inputs for Experiment 1

w_g	LA-2	LA-3	LAMM-2	LAMM-3
0.3	5.11%	5.60%	6.21%	6.11%
0.5	8.25%	8.42%	6.94%	8.03%
0.7	1.78%	1.50%	0.71%	2.00%

Figure 4.9: Percentage increase in utility over MYOPIC baseline for Experiment 1

For each of the values of w_g with which we conducted our simulation, we see that the more sophisticated LOOK-AHEAD and LOOK-AHEAD-MYOPIC-MARKOV do not yield substantial increases in performance. Why is this the case? For $w_g = 0.5$, we see that on average, MYOPIC spends a total of 1.25 worth of incentive over 12 rounds for each agent input, while the more sophisticated algorithms pay out between 1.50 and 1.70 worth of incentive over 12 rounds. It seems that the algorithms that employ look ahead in this scenario would rather spend more incentive in order to increase the probability that the target arm g is chosen at any given step. Even though each of the look ahead algorithms end up paying more incentives than MYOPIC, the resulting increase in the probability of a selection of the target arm allows these algorithms to provide slight increases in empirical performance.

These results make sense given the parameters of our experiment. Because we set $\underline{g} = \underline{h} = 0, \overline{g} = \overline{h} = 1$, and $p_g = p_h = 0.5$, we can think about the expected value of both arms as 0.5. Thus, varying $w_g \in \{0.3, 0.5, 0.7\}$ means that the principal overvalues selections of the target arm compared to incentives. In the case of MYOPIC,

he would always select an incentive $\delta \in [0, w_g]$ to make $p^\delta(w_g - \delta) \geq 0$. Since the expected values of both of the arms are the same in this case, MYOPIC will end up incentivizing the target arm very often. The same holds true for the look ahead algorithms because of the high value associated with a selection of a target arm. Because w_g is so much greater than the difference between the expected values of each of the arms, the principal's tradeoff between incentives and selections of the target arm is downplayed. Since the look ahead algorithms evaluate this tradeoff to improve their performance, the performance increase over MYOPIC is difficult to observe given these parameters.

Experiment 2. Figures 4.10 and 4.11 summarize our results from Experiment 2.

p_g	No incentive	MYOPIC	LA-2	LA-3	LAMM-2	LAMM-3
0.3	0.180	0.249	0.261	0.579	0.560	0.566
0.5	0.775	0.895	1.309	1.303	1.264	1.296
0.7	1.405	1.665	2.110	2.107	2.096	2.134

Figure 4.10: Average utility over the same set of 100 agent inputs for Experiment 2

p_g	LA-2	LA-3	LAMM-2	LAMM-3
0.3	4.889%	132.9%	124.9%	127.6%
0.5	46.18%	45.61%	41.15%	44.77%
0.7	26.79%	26.58%	25.95%	28.23%

Figure 4.11: Percentage increase in utility over MYOPIC baseline for Experiment 2

Experiment 3. Figures 4.12 and 4.13 summarize our results from Experiment 3.

w_g	No incentive	MYOPIC	LA-2	LA-3	LAMM-2	LAMM-3
.3	0.465	0.477	0.514	0.508	0.508	0.512
.4	0.620	0.678	0.730	0.930	0.960	0.973
.45	0.698	0.788	0.860	1.073	1.095	1.115
.5	0.775	0.895	1.309	1.303	1.264	1.296
.55	0.853	1.019	1.481	1.491	1.480	1.476
.6	0.930	1.197	1.664	1.679	1.687	1.724
.7	1.085	1.659	2.165	2.220	2.270	2.193

Figure 4.12: Average utility over the same set of 100 agent inputs for Experiment 3

w_g	LA-2	LA-3	LAMM-2	LAMM-3
0.3	7.688%	6.429%	6.349%	7.198%
0.4	7.746%	37.21%	41.59%	43.52%
0.45	9.182%	36.25%	39.04%	41.53%
0.5	46.18%	45.61%	41.16%	44.77%
0.55	45.28%	46.26%	45.18%	44.80%
0.6	38.99%	40.29%	40.96%	44.01%
0.7	30.53%	33.81%	36.82%	32.18%

Figure 4.13: Percentage increase in utility over MYOPIC baseline for Experiment 3

Because Experiments 2 and 3 are similar, we will discuss the results together. In these experiments, we see that the look ahead algorithms provide significant increases in performance. Why is this the case?

For both of these experiments, we set $\underline{g} = 0, \overline{g} = 1, \underline{h} = 0.5, \overline{h} = 1.5$. Drawing on our intuition from Experiment 1, we see the following from Experiment 2: when $p_g = 0.3$, the expected value of g is 0.3, while the expected value for h remains at 1. The difference in expected value is greater than the value that is prescribed to a selection of the target arm $w_g = 0.5$. This means that more sophisticated reasoning about the tradeoff between incentives and selections of the target arm is necessary. Indeed, the look ahead algorithms of sufficient depth are empirically able to obtain more than double the performance of MYOPIC. As the probability p_g increases, so does the expected value of g. Thus, we expect that reasoning about the tradeoff between incentives and selections of the target arm becomes less crucial for good

performance, and thus the percentage increase in performance for the look ahead algorithms decreases as the expected value of the two arms converges.

Furthermore, we believe that the look ahead algorithms benefit greatly when the probability p_g is far away from .5. Intuitively, this is because the probability mass for a given belief state vector becomes concentrated at a few specific belief states in the principal's belief state vector. The look ahead algorithms can use this more refined information to their advantage when determining what incentives to provide. This effect serves to accentuate the difference in performance between MYOPIC and the look ahead algorithms.

In many of our results from Experiment 3, we see that some of the look ahead algorithms of depth 3 are outperformed by the same algorithm of depth 2. We believe that this might be a result of the lack of discounting in the formulation of our look ahead algorithm. Because we do not discount the values of nodes further away from the root node in our algorithm, too much weight is being given to low probability outcomes. This, along with the fact that we only take the average of 100 simulations would lead to decreased empirical performance of the look ahead algorithms of greater depth.

As a quick sanity check for our results from Experiment 3, we see that the peak increase in performance occurs at around $w_g = 0.5$ or 0.55. As w_g decreases or increases from that point, the percentage increase in performance begins to drop off. If $w_g = 0$, then all algorithms would provide 0, so the look ahead algorithms would provide no benefit over MYOPIC. Similarly, if $w_g \to \infty$, then all algorithms would always provide incentives to guarantee that the target arm is selected.

Looking over the results to Experiment 3, we see some inconsistencies with the observations we obtained from Experiments 1 and 2. First, when $w_g = 0.3$, we see that the performance increase is much lower. Based on what we said earlier, should it not be the case that the look ahead algorithms will perform significantly better due to the tradeoff between incentives and target arm selections? Here, we provide

two explanations. The first is that because $p_g = 0.5$, the look ahead algorithms do not have the ability to use the concentrated probability masses to their advantage. Moreover, there must be a point at which the value of the target arm becomes too low for any algorithm to provide better performance than any other. Take, for instance, the extreme case when $w_g = 0$, for which all algorithms of our algorithms will return 0.

The results from Experiment 3 show that when $p_g = 0.5$, the cases that allow for the most significant increases in performance over MYOPIC via the use of look ahead algorithms are where the value for the incentives and a selection of the target arm are about the same. In this case, we see that the peak seems to occur at around $w_g = 0.5$. This makes sense because more sophisticated algorithms might be able to make better decisions about how to reason about the tradeoff in comparison to MYOPIC, which only maximizes for the current time step.

To better understand what the look ahead algorithms are doing in the case where $w_g = 0.5$ for Experiment 3, we plot the average utility per round in Figure 4.14 and the average amount of incentives provided to the target arm in Figure 4.15.

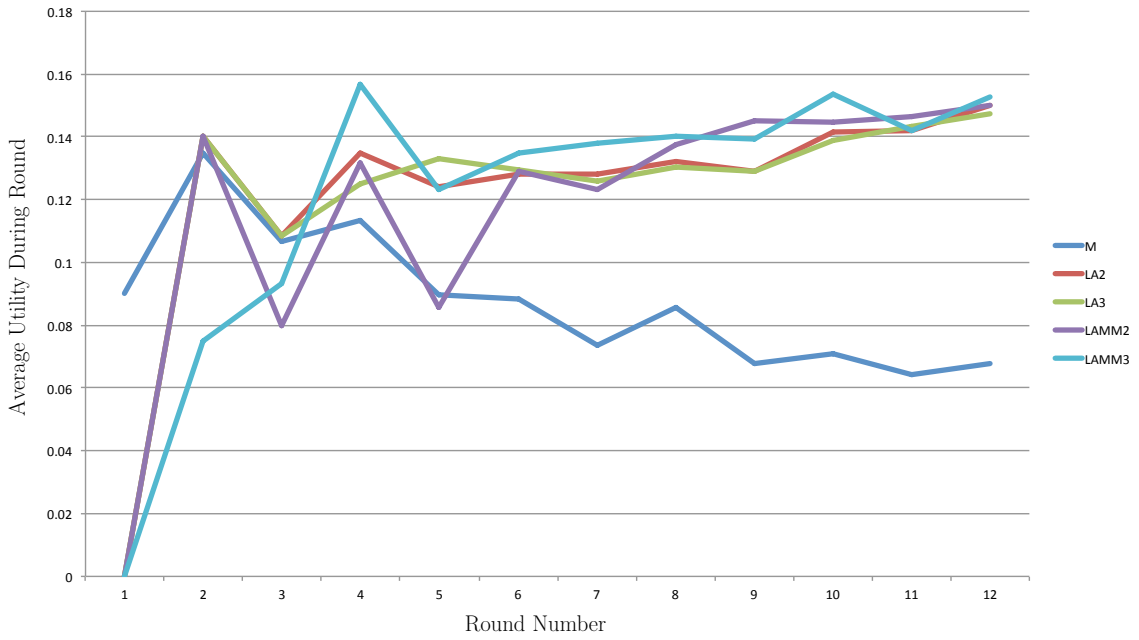

Figure 4.14: Average utility in each round for Experiment 3, $w_g = 0.5$

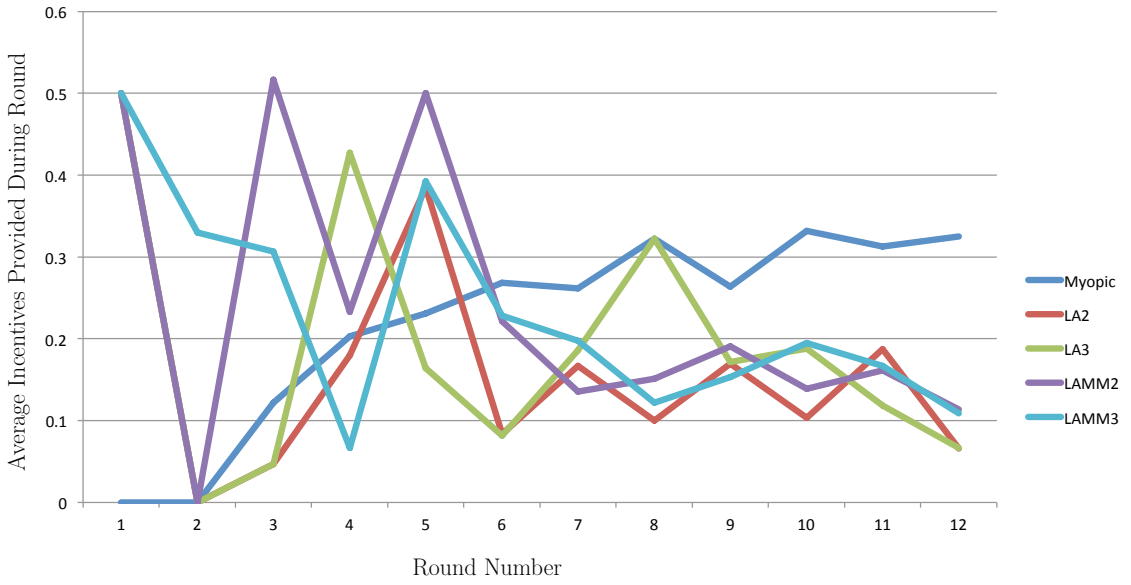

Figure 4.15: Average incentives provided in each round for Experiment 3, $w_g = 0.5$

We see that in the first three rounds, MYOPIC performs at least as well as any

of the look ahead algorithms. However, starting from round 4 onward, the average utility per round decreases drastically. Most notably, by round 12, MYOPIC is consistently getting about .08 less utility per round on average than all of the look ahead algorithms, which begin to perform similarly after round 7. We also see that the incentives that MYOPIC continues to provide after round 8 is about 50% more than the incentives provided by the look ahead algorithms. This can perhaps be explained by the fact that the look ahead algorithms have a better idea that the agent's belief states are converging than MYOPIC. The look ahead algorithms can use this convergence information to help obtain a higher expected utility.

4.6 Discussion

The motivation for this chapter was drawn from the interplay between Theorem 4 and Theorem 5. The negative result in Theorem 4 prompted us to consider scenarios in which the agent's parameters are known to the principal. Even though we assume that the principal has knowledge of the agent's parameters, an element of uncertainty remains in that the principal cannot observe the outcomes experienced by the agent.

We began by considering scenarios in which the underlying distribution behind each of the arms was continuous. Due to the analytical difficulty of these problems, we were prompted a new objective criterion that better captures the tradeoff between incentives and selections of the target arm that the principal faces. We also moved to a discrete setting in an attempt to gain some traction on the problem. Although the problem seemed intractable at first, we were able to reason about the properties of the discrete setting to collapse the state space. We discussed ways for the principal to easily update his belief state vector given observations about the agent's behavior and demonstrated that the principal need only consider a candidate set of incentives.

To further simplify the principal's decision problem, we conjecture that it may be the case that the principal would never want to provide incentives to the non-target arm in the setting. This conjecture is reminiscent of OPT, and, if true, could greatly

speed up the look ahead process by potentially halving the number of incentives that need to be considered at each step. However, reasoning about this conjecture is difficult. Incentives placed on the non-target arm may provide extra information about the agent's belief state. Because we do not have a means of quantifying the value of this extra information, we cannot compare its benefits against the cost of the extra incentive required to obtain the information. Although we allowed our algorithms to provide incentives to the non-target arm, we did not see this occur in any of our simulations.

Nonetheless, our other simplifying observations encouraged us to consider the problem of designing algorithms that would return good empirical performance on the principal's incentive design problem. We drew up several algorithms and ran simulations to see their performance in comparison to the myopic policy. We used the results from our simulation to reason about which problems pose interesting incentive design challenges and allow for the effective incentive strategies to play a significant role in influencing the decisions of the agent. By studying the simulation results, we were able to better understand the way in which the algorithms provide incentives.

Our simulation results are promising. They demonstrate that a principal solving an incentive design problem with known agent parameters can utilize simple, tractable algorithms to obtain performance increases over a myopic incentive design strategy. The similarity between the results of varying look ahead depths suggest that we do not need to simulate too far forward in order to obtain good performance in practice.

Chapter 5

Conclusion

At the beginning of this thesis, we set out to explore the following question: how can a principal with incomplete information provide incentives so as to encourage an adaptive agent to perform desirable actions?

Our main motivations were drawn from real life scenarios. The government wants citizens to conserve energy and natural resources. An elementary school wants its students to utilize hand sanitizer to prevent the propagation of communicable diseases. A firm wants to promote its product over the competition. A coach wants his students to adopt a particular technique. Local healthcare centers want individuals to vaccinate their children as soon as possible.

In each of these settings, the way in which external incentives are designed play a crucial role in determining the outcome. When provided with a small reimbursement for their effort, citizens are more likely to recycle plastic bottles. If provided with candy for their good habits, students might be more likely to make use of sanitizing stations. A coupon might entice a consumer to consider a previously uninteresting option. Providing a bag of lentils upon completion of an immunization program might ensure greater penetration of a vaccine in rural villages [2].

In this thesis, we laid down a generalized framework to analyze how incentives could be provided to produce the desired result. In doing so, we uncovered the power and limitations of incentives in settings that approximate everyday scenarios.

5.1 Brief Review

In Chapter 1, we developed the motivation for the questions behind this thesis in great detail. We followed existing work on policy teaching and environment design and formulated our question in terms of an online model of environment design. We explored the literature in search of related works and came across the multi-armed bandit problem [14], which served as the foundation for the model in our analysis found in later chapters.

In Chapter 2, we laid the framework for a generalized model of incentive design by specifying the properties of the agent, the principal, and the decision environment. We specified various assumptions that we made about each of these properties and the motivation behind each of these assumptions. We then formally stated the principal's incentive design problem, in which the principal must decide how to provide incentives to as to maximize a specific objective criterion.

In Chapter 3, we considered the principal's incentive design problem when the agent parameters are unknown. We introduced OPT, a powerful incentive design strategy based on the notion of only providing incentives to the desired target action. In settings with a per-round budget, we proved that OPT is optimal, regardless of whether the objective criterion is to induce the target arm to be selected once or multiple times. In settings with a total budget, we proved that OPT is optimal when the principal would like the target arm to be induced once, but in the case in which the principal would like to induce the target arm multiple times, we demonstrated that no randomized algorithm can provide a bounded competitive ratio. Given this result, we were encouraged to consider settings in which the principal has knowledge of the agent's parameters in order to move forward with our analysis.

In Chapter 4, we analyzed the principal's incentive design problem when agent parameters are known, but the outcomes of the arms cannot be observed. Although we were quickly overwhelmed in our analysis of settings with continuous distributions

behind the arms, we were still able to make several insightful observations through the use of simulations. Our analysis of continuous models also motivated us to reconsider our objective criterion to better capture the tradeoff between incentives paid out and selections of the target arm.

To gain traction on the problem, we moved to a discrete model in which the underlying distributions behind each arm were modeled as two point masses. We demonstrated a way to pare down the principal's decision space and exhibited a tractable means of incorporating information obtained from the agent's action at each round. Having gained some traction in this scenario, we considered possible algorithms that might lead to good performance. We implemented each of these algorithms and conducted simulations in order to learn more about when sophisticated incentive design strategies could be useful. Through these experiments, we were able to determine scenarios in which the principal could make a significant difference in influencing the decisions of the agent.

5.2 Reflections and Future Work

Many of the future directions of this work can be identified by considering ways in which we can make our model better approximate reality. This can be done by relaxing the modeling assumptions that we discussed in Chapter 2 and by considering more realistic continuous and discrete distributions. We discuss a few directions for future work below:

- We currently assume that the agent deterministically chooses the action with the highest perceived value plus incentive. In practice, the agent might make a mistake in his selection and accidentally select another arm at random. How does the principal's incentive design problem change if the agent function has some element of randomness? Are there specific design strategies that might help to mitigate the effects of randomness?

- In all of our problem settings, we assume that the bandit property (i.e., independence of the arms) holds. In many real life settings, this is not the case because a pull from one arm may reveal information about multiple arms. For instance, if the agent is deciding between what flavor donut he should purchase at each round, then the information that he obtains from a realization of a chocolate donut might shift the perceived value that he associates with a chocolate donut with sprinkles. Is there a clean way to model decision environments that do not satisfy the bandit property? How can the principal think about the incentive design problem in these settings?

- The continuous distributions that we have considered in this thesis have been based mainly on the uniform distribution, for ease of analysis. Even so, the analysis becomes quite difficult even in small cases. Is it possible to perform a similar analysis if the outcomes behind each of the arms were normally distributed? Along the same lines, can we easily extend our discrete model to account for multiple point masses? If these discrete models remain tractable to simulate, would it be possible to approximate continuous distributions using sufficiently rich discrete models?

- In our work we assume that the principal always has the same objective criterion. In practice, the principal may have some richer set of criteria. For instance, the principal may attach a high utility to a specific sequence of pulls. Along the same lines, the principal's objective criterion might vary over time and between arms.

- Our algorithms in Chapter 4 assume that the principal has perfect knowledge of the agent parameters. It may be the case that this information is noisy and thus the principal only has approximations of these parameters. How robust are these algorithms to inaccuracies in the input parameters? The answer to this question would also be helpful in understanding the set of parameters that would lead to a change in the principal's incentive design strategy.

- A key assumption in our work is that the agent does not reason about the incentives that the principal provides. One interesting question to consider is how a strategic agent who knows what algorithm the principal is using to provide incentives might take actions so as to minimize the principal's payoff.

5.3 Conclusion

Our work offers insight into the power and limitations of incentive design. On the one hand, we have a surprisingly negative result that tells us that we cannot bound our worst-case performance in the general case when nothing about the agent is known. On the other hand, when we have a good model of the agent's parameters, we can run tractable algorithms to provide non-trivial increases in performance. These results are promising. They demonstrate that even with some uncertainty that is characteristic of real life scenarios, it is still possible to encourage desired behaviors through the use of incentives.

Appendix A

Obtaining $P(A_1, A_2)$ as a Function of δ

We seek to evaluate the joint probability $P(1-\delta \leq a_0 \leq 1, \frac{a_0+a_1}{2} \geq \delta)$. The straightforward way of computing this value is incorrect because we end up integrating over invalid values of a_1 (values that are less than 0 and greater than 1). As a result, to compute this, we break things up into cases based on δ. Consider the following straightforward integral:

$$\int_{1-\delta}^{1} \int_{2\delta-a_0}^{1} \mathrm{d}a_1 \mathrm{d}a_2$$

Case 1. The first case is where $a_2 \geq 1-\delta \rightarrow \frac{a_0+a_1}{2} \geq 2\delta$. To compute this value, we can check for when $1-\delta \geq 2\delta$. This happens when $\delta \leq \frac{1}{3}$. As a result, when $\delta \leq \frac{1}{3}$, the probability we are interested in just the probability that $a_2 \geq 1-\delta$, which is just δ.

Case 2. Now consider when $\alpha > \frac{1}{3}$. Since we do not wish to integrate over negative values in the inner integral, we need to ensure that $2\delta - a_0 > 0$. For $1/3 < \delta < 1/2, 2\delta < 1$, so we need to enforce this limit. Thus, for $1/3 < \delta < 1/2$, the value we seek can be expressed as:

$$\int_{1-\delta}^{2\delta} \int_{2\delta-a_0}^{1} \mathrm{d}a_1 \mathrm{d}a_0 + (1-2\delta) = \frac{-1+8\delta-9\delta^2}{2}$$

The $(1-2\delta)$ comes from the fact that if $a_0 > 2\delta$ on its own, it does not matter what a_1 is, and the probability of this occurring is exactly $1-2\delta$.

Case 3. For $\delta > 1/2$, we do not need to restrict the upper bound of the outer integral since $2\delta - 1$ will always be positive. However, we do reach a point where the lower bound of the inner integral becomes larger than one (since no value of a_1 can cause $a_0 + a_1$ to exceed 2δ. We can check for this break point by computing δ where $1 - \delta + 1 < 2\delta$, which occurs at $\delta = 2/3$. For $\delta < 2/3$, this is not an issue, and we just get the basic integral shown above:

$$\int_{1-\delta}^{1}\int_{2\delta-a_0}^{1} \mathrm{d}a_1\mathrm{d}a_0 = 2\delta - \frac{5\delta^2}{2}$$

Case 4. Now we run into the case where we need to increase the lower bound of the outer integral so that the lower bound of the inner integral is < 1. We can make sure that there are values of a_1 that make the sum greater than 2δ by setting the lower bound of the outer integral to $2\delta - 1$.

$$\int_{2\delta-1}^{1}\int_{2\delta-a_0}^{1} \mathrm{d}a_1\mathrm{d}a_9 = 2 - 4\delta + 2\delta^2$$

The union of all of these equations gives us our piecewise function:

$$P(A_1, A_2) = \begin{cases} \delta & \text{if } \delta \leq \frac{1}{3}. \\ -\frac{9}{2}\delta^2 + 4\delta - \frac{1}{2} & \text{if } \frac{1}{3} < \delta \leq \frac{1}{2}. \\ -\frac{5}{2}\delta^2 + 2\delta & \text{if } \frac{1}{2} < \delta \leq \frac{2}{3}. \\ 2\delta^2 - 4\delta + 2 & \text{if } \frac{2}{3} < \delta \leq 1. \end{cases}$$

When we know that $a_0 \geq .63$, we can evaluate Case 2 as

$$\int_{1-\delta}^{2\delta}\int_{2\delta-a_0}^{0.63} \mathrm{d}a_1\mathrm{d}a_0 + (1-2\delta) = -.72 + 2.74\delta - \frac{5}{2}\delta^2$$

Taking the derivative, we see that the critical point occurs at $\delta = \frac{2.74}{5} = 0.548$.

Appendix B

Myopic Tables for Figure 4.6

δ	Expected Utility at (a)
1.5	$w_g - 1.5$
1	$\frac{7}{8}(w_g - 1)$
0.5	$\frac{5}{8}(w_g - 0.5)$
0	$\frac{3}{8}w_g$

δ	Expected Utility at (b)
1.5	$w_g - 1.5$
0	$.5w_g$

δ	Expected Utility at (c)
1	$w_g - 1$
0.5	$\frac{5}{6}(w_g - 0.5)$
0	$.5w_g$

δ	Expected Utility at (d)
1.5	$w_g - 1.5$
1	$\frac{5}{6}(w_g - 1)$
0.5	$.5(w_g - 0.5)$
0	$\frac{1}{6}w_g$

δ	Expected Utility at (e)
0	w_g

δ	Expected Utility at (f)
1.5	$w_g - 1.5$
0.75	$\frac{7}{8}(w_g - 0.75)$
0.5	$\frac{5}{8}(w_g - 0.5)$
0	$\frac{3}{8}w_g$

References

[1] Moshe Babaioff, Yogeshwer Sharma, Aleksandrs Slivkins. Characterizing Truthful Multi-Armed Bandit Mechanisms. In *Proceedings of the Tenth ACM Electronic Commerce Conference*, pp. 79–88, 2009.

[2] Abhijit Vinayak Banerjee, Esther Duflo, Rachel Glennerster, Dhruva Kothari. Improving Immunisation Coverage in Rural India: Clustered Randomised Controlled Evaluation of Immunisation Campaigns With and Without Incentives. *BMJ* 2010; 340:c2220.

[3] Dirk Bergemann, Juuso Välimäki. The Dynamic Pivot Mechanism. *Econometrica*, 78:771–789, 2010.

[4] Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[5] Craig Boutilier. A POMDP Formulation of Preference Elicitation Problems. In *Proceedings of the Seventeenth AAAI Conference on Artificial Intelligence*, pp. 239–246, 2002.

[6] Ronen I. Brafman and Moshe Tennenholtz. On Partially Controlled Multi-Agent Systems. *Journal of Artificial Intelligence Research*, 4:477–507, 1996.

[7] Ruggiero Cavallo, David C. Parkes, and Satinder Singh. In *Proceedings of the Twenty-second Conference on Uncertainty in Artificial Intelligence*, pp. 55–62, 2006.

[8] Jon Doyle. Prospects for Preferences. In *Computational Intelligence.*, 20:111–136, 2004.

[9] John C. Gittins. *Multi-Armed Bandit Allocation Indices.* Wiley-Interscience Series in Systems and Optimization, 1989.

[10] Robert D. Kleinberg. *Online Decision Problems with Large Strategy Sets.* Ph.D. thesis, MIT, 2005.

[11] W. Bradley Knox and Peter Stone. Interactively Shaping Agents via Human Reinforcement: The TAMER Framework. In *The Fifth International Conference on Knowledge Capture*, pp. 9–16, 2009.

[12] George E. Monahan. A Survey of Partially Observable Markov Decision Processes: Theory, Models, and Algorithms. In *Management Science*, 28:1–16, 1982.

[13] Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, 1999.

[14] Herbert Robbins. Some Aspects of the Sequential Design of Experiments. *Bulletin of the American Mathematical Society*, 58:527–535, 1952.

[15] Peter Stone, Gal A. Kaminka, Sarit Kraus, and Jeffrey S. Rosenschein. Ad Hoc Autonomous Agent Teams: Collaboration without Pre-Coordination. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence.* 2010.

[16] Peter Stone and Sarit Kraus. To Teach or Not to Teach?: Decision Making Under Uncertainty in Ad Hoc Teams. In *Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems.* 2010.

[17] Haoqi Zhang, Yiling Chen, and David C. Parkes. A General Approach to Environment Design with One Agent. In *Proceedings of the Twenty-first International Joint Conference on Artificial Intelligence*, pp. 2002–2008, 2009.

[18] Haoqi Zhang and David C. Parkes. Value-Based Policy Teaching with Active Indirect Elicitation. In *Proceedings of the Twenty-third AAAI Conference on Artificial Intelligence*, pp. 208–214, 2008.

[19] Haoqi Zhang, David C. Parkes, and Yiling Chen. Policy Teaching Through Reward Function Learning. In *Proceedings of the Tenth ACM Electronic Commerce Conference*, pp. 295–304, 2009.