ABSTRACT

Title of dissertation: GENERATION AND ANALYSIS OF STRATEGIES IN AN EVOLUTIONARY SOCIAL LEARNING GAME

James Ryan Carr, Doctor of Philosophy, 2013

Dissertation directed by: Professor Dana Nau
Department of Computer Science

An important way to learn new actions and behaviors is by observing others, and several evolutionary games have been developed to investigate what learning strategies work best and how they might have evolved. In this dissertation I present an extensive set of mathematical and simulation results for Cultaptation, which is one of the best-known such games.

I derive a formula for measuring a strategy's expected reproductive success, and provide algorithms to compute near-best-response strategies and near-Nash equilibria. Some of these algorithms are too complex to run quickly on larger versions of Cultaptation, so I also show how they can be approximated to be able to handle larger games, while still exhibiting better performance than the current best-known Cultaptation strategy for such games. Experimental studies provide strong evidence for the following hypotheses:

1. The best strategies for Cultaptation and similar games are likely to be conditional ones in which the choice of action at each round is conditioned on

the agent's accumulated experience. Such strategies (or close approximations of them) can be computed by doing a lookahead search that predicts how each possible choice of action at the current round is likely to affect future performance.

2. Such strategies are likely to prefer social learning most of the time, but will have ways of quickly detecting structural shocks, so that they can switch quickly to individual learning in order to learn how to respond to such shocks. This conflicts with the conventional wisdom that successful social-learning strategies are characterized by a high frequency of individual learning; and agrees with recent experiments by others on human subjects that also challenge the conventional wisdom.

GENERATION AND ANALYSIS OF STRATEGIES IN AN EVOLUTIONARY SOCIAL LEARNING GAME

by

James Ryan Carr

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2012

Advisory Committee:
Professor Dana Nau, Chair/Advisor
Professor Michele Gelfand
Professor Mohammad Hajiaghayi
Professor Donald Perlis
Professor James Reggia

Acknowledgments

I would like to acknowledge the support and encouragement of many extraordinary people, without whom this work would have been impossible.

I would like to thank Dr. Dana Nau for being a wonderful advisor, mentor, and friend, and especially for teaching me that half of research is arguing, and how to argue well.

I would like to thank my other committee members, Drs. Michele Gelfand, James Reggia, Mohammad Hajiaghayi, Donald Perlis, and V.S. Subrahmanian, for their support, guidance, and patience.

I would like to thank Dr. Marie desJardins for getting me interested in research and encouraging me to pursue a Ph.D.

I would like to thank Eric Raboin and Austin Parker for their encouragement and contributions to our earlier studies of Cultaptation. I would like to thank my fellow graduate students in LCCD, Eric, Austin, Patrick Roos, Kan-Leung Cheng, Ron Alford, Vikas Shivashankar, and Brandon Wilson, for six years of friendship and advice.

This work was supported in part by AFOSR Grant FA95501210021, ARO Grant W911NF1110344, and ONR Grant W911NF0810144. The views expressed are those of the author and do not necessarily reflect the official policy or position of the funders.

Table of Contents

List of Figures

Chapter 1

Introduction

An important way to learn new actions and behaviors is social learning, i.e., learning by observing others. Some social-learning theorists believe this is how most human behavior is learned [1], and it also is important for many other animal species [2, 3, 4]. Such learning usually involves evaluating the outcomes of others' actions, rather than indiscriminate copying of others' behavior [5, 6], but much is unknown about what learning strategies work best and how they might have evolved.

For example, it seems natural to assume that communication has evolved due to the inherent superiority of copying others' success rather than learning on one's own via trial-and-error innovation. However, there has also been substantial work questioning this intuition [7, 8, 9, 10, 11].

Several evolutionary games have been developed to investigate social learning [12, 13, 14, 15]. One of the best-known is Cultaptation, a multi-agent social-learning game developed by a consortium of European scientists [15] who sponsored an international tournament with a €10,000 prize.[1] The rules of Cultaptation are rather complicated (see Section 2.1), but can be summarized as follows:

- Each agent has three kinds of possible actions: innovation, observation, and exploitation. These are highly simplified analogs of the following real-world

[1] NOTE: I am not affiliated with the tournament or with the Cultaptation project.

activities, respectively: spending time and resources to learn something new, learning something by communicating with another agent, and exploiting the learned knowledge.

- At each step of the game, each agent must choose one of the available actions. How an agent does this constitutes the agent's "social learning strategy."

- Each action provides an immediate numeric payoff and/or information about the payoffs of other actions at the current round of the game. This information is not necessarily correct in subsequent rounds because the actions' payoffs may vary from one round to the next, and the way in which they may vary is unknown to the agents in the game.[2]

- Each agent has a fixed probability of dying at each round. At each round, each agent may also produce offspring, with a probability that depends on how this agent's average per-round payoff compares to the average per-round payoffs of the other agents in the game.

A second Cultaptation tournament is scheduled to begin in February 2012. This tournament carries a €25,000 prize and introduces a few new concepts into the game, such as the ability for agents to improve actions they already know, and proximity-based observation. This work does not deal with these additions, although

[2] For my analyses, I assume the payoffs at each round are determined by an arbitrary function (which may be either deterministic or probabilistic), and I analyze how strategies perform given various possible characteristics of that function. In general, such characteristics would not be known to any Cultaptation agent—but my objective is to examine the properties of strategies in various versions of Cultaptation, not to develop a Cultaptation agent *per se*.

I may address them in future work.

My work has had two main objectives: (1) to study the nature of Cultaptation to see what types of strategies are effective; and (2) more generally, to develop ways of analyzing evolutionary environments with social learning.

This work includes the following results. First, it provides a formula for approximating (to within any given error bound $\epsilon > 0$) the *expected per-round utility* (EPRU) of a given strategy (Section 4.2), and a proof that a strategy with maximal EPRU will have the highest frequency in the limit (Section 4.2.2). This provides a basis for evaluating Cultaptation strategies analytically, rather than through simulated games. Next, the work provides a strategy-generation algorithm that can construct a strategy that is within any given error bound $\epsilon > 0$ of a best response to a given set of competing strategies (Section 5.1). It then presents the Cultaptation Strategy Learning Algorithm (CSLA), which uses the strategy-generation algorithm in an iterative self-improvement loop to attempt to find a strategy that is a near-best response to itself, and is therefore a symmetric near-Nash equilibrium (Section 5.2). Finding such a strategy is desirable because it should be able to perform well against any set of competing strategies (i.e., any strategies it plays against in a tournament setting).

The work then provides experimental results outlining the generation of an approximate Nash equilibrium strategy, s_{self}, in a small version of Cultaptation, and a performance comparison (in the smaller environment) between s_{self} and EVChooser, a known good strategy from the Cultaptation tournament (Section 5.3). These experiments show that s_{self} is able to outperform EVChooser, and provide several

insights into the characteristics of good Cultaptation strategies. For example, the experiments show that s_{self} observes and exploits most of the time, but switches quickly to innovation when a structural shock occurs, switching back to observation and exploitation once it has learned how to respond to the shock. This conflicts with the conventional wisdom [16, 7] that successful social-learning strategies are characterized by a high frequency of innovation, but it helps to explain both the results of the Cultaptation tournament [17] and some recent experimental results on human subjects [18].

Finally, the work shows how the previous results (which were run on smaller versions of Cultaptation) can be extended to the larger version used in the tournament (Chapter 6). It first shows how the analysis and experimental results outlined above can be used to identify potential problems in the best strategy from the Cultaptation tournament (i.e., *discountmachine*). Next, it uses the formulae from the analysis to define a new strategy, *relaxedlookahead*, that avoids such weaknesses. Experimental results verify that *relaxedlookahead* is capable of outperforming *discountmachine* in a variety of environments similar to those used in the Cultaptation tournament, and provide an in-depth analysis of the factors that allowed *relaxedlookahead* to perform better.

Taken together, these results provide strong support for the following hypotheses about the best strategies for Cultaptation and similar games: First, the best strategies are likely to be conditional ones in which the choice of action at each round is conditioned on the agent's accumulated experience. Such strategies (or close approximations of them) can be computed by doing a lookahead search that

predicts how each possible choice of action at the current round is likely to affect future performance. Second, it is likely that the best strategies will observe and exploit most of the time, but will have ways of quickly detecting structural shocks, so that they can switch quickly to innovation in order to learn how to respond to such shocks.

In addition to these insights about Cultaptation and social learning in general, the work also explains how the analysis and algorithms outlined above could be adapted to apply to future evolutionary games that, like Cultaptation, are significantly more complex than classical evolutionary games. As researchers in the field of evolutionary game theory continue to study more complex phenomena like social learning, it is likely that these weaker assumptions will be needed more frequently, and so techniques like the ones presented here will be necessary more often.

Chapter 2

Background

2.1 Cultaptation Social-Learning Game

This section gives a more detailed description of the Cultaptation social learning game, adapted from the official description [15]. The game is a multi-agent round-based game, where one action is chosen by each agent each round. There are N agents playing the game, where N is a parameter to the game. No agent knows of any other agent's actions at any point in the game except through the Obs action specified below. The actions available to each agent are innovation (Inv), observation (Obs), and exploitation ($\mathsf{X}_1, \ldots, \mathsf{X}_M$, where M is a parameter to the game). Each Inv and Obs action informs the agent what the utility would be for one of the exploitation actions, and an agent may not use an exploitation action X_i unless the agent has previously learned of it through an innovation or observation action. Here are some details:

Exploitation. Each exploitation action X_i provides utility specific to that action (e.g. X_1 may provide utility 10 and X_2 may provide utility 50). The utility assigned to each action at the beginning of the game is drawn from a probability distribution π, where π is a parameter to the game.

The utility provided by each exploitation action X_i may change on round r, according to a probability c_r. The function c is a parameter to the game, and

specifies the *probability of change* for every round of the game. When the changes occur, they are invisible to the agents playing the game until the agent interacts with the changed action. For instance: if an action's utility happens to change on the same round it is exploited, the agent receives the new utility, and discovers the change when the new utility is received. The new utility for a changed action is determined via the distribution π.

Innovation. When an agent uses the Inv action, it provides no utility, but it tells the agent the name and utility of some exploitation action X_i that is chosen uniformly at random from the set of all exploitation actions about which the agent has no information. If an agent already knows all of the exploitation actions, then Inv is illegal, and indeed undesirable (when there is nothing left to innovate, why innovate?). The agent receives no utility on any round where she chooses an Inv action.

Observation. By performing an Obs action, an agent gets to observe the action performed and utility received by some other agent who performed an exploitation action on the previous round. Agents receive no utility for Obs actions, nor any information other than the action observed and its value: the agent being observed, for instance, is unknown. If none of the other agents performed an exploitation action on the previous round, then there were no X_i actions to observe so the observing agent receives no information. In some variants of the social learning game, agents receive information about more than one action when observing. I do not treat such variants directly in this proposal, but it is straightforward to extend my algorithms to take this difference into account.

Round #	1	2	3	4	5	...	k
I1's action	Inv	X_1	X_1	X_1	X_1	...	X_1
I1's utility	0	3	6	9	12	...	$3(k-1)$
Per round	0	1.5	2	2.25	2.4	...	$3\frac{k-1}{k}$
I2O's action	Inv	Inv	Obs	X_3	X_3	...	X_3
I2O's utility	0	0	0	8	16	...	$8(k-3)$
Per round	0	0	0	2	3.2	...	$8\frac{k-3}{k}$

Table 2.1: Action sequences from Example 1, and their utilities.

Example 1. Consider two strategies: the innovate-once strategy (hereafter I1), which innovates exactly once and exploits that innovated action (whatever it is) for the rest of the game, and the innovate-twice-observe-once strategy (hereafter I2O), which innovates twice, observes once, and exploits the highest valued action of the actions discovered for the rest of the game. For simplicity of exposition, suppose there are only four exploitation actions: X_1, X_2, X_3, and X_4. The values for each of these actions are drawn from a distribution; in this example we will assume that they are chosen to be 3, 5, 8, and 5, respectively. For simplicity, we will assume the probability of change is 0. Suppose there are two agents: one I1 and one I2O. For the first action, I1 will innovate, which we suppose gives I1 the value of action X_1. On every sequential action, I1 will choose action X_1, exploiting the initial investment. If the agent dies k rounds later, then the history of actions and utilities will be that given in Table 2.1; giving a utility of $3(k-1)$ and a per-round utility of $3\frac{k-1}{k}$.

In contrast, I2O will innovate, informing it of the utility of X_3: 8, then it will innovate again, informing it of the utility of X_4: 5, and finally it will observe. On the second round, I1 performed X_1, and since these are the only two agents, this was the only exploitation action performed. Therefore, I2O's observation action on the next round must report that another agent got a utility of 3 from action X_1 last round (if there were multiple possibilities, one would be chosen uniformly at random). On round 4, I2O then knows that actions X_1, X_3, and X_4 have utilities of 3, 8, and 5, respectively. Since the probability of change is 0, the obvious best action is X_3, which I2O performs for the rest of her life. The utility of I2O on round k is $8(k-3)$, making the per-round utility $8\frac{k-3}{k}$. Note that on rounds 2 to 4, I2O will have a worse per-round utility than I1, while after round 4, the utility of I2O will be higher (this is important because reproduction is tied to per-round utility, as I will show shortly).

Formally, everything that an agent α knows about each round can be described

by an *action-percept pair*, $(a, (m, v))$, where $a \in \{\mathsf{Inv}, \mathsf{Obs}, \mathsf{X}_1, \ldots, \mathsf{X}_M\}$ is the action that α chose to perform, and (m, v) is the percept returned by the action. More specifically, $m \in \{\mathsf{X}_1, \ldots, \mathsf{X}_M, \emptyset\}$ is either an exploitation action or a null value, and v is the utility observed or received. While a is chosen by the agent, m and v are percepts the agent receives in response to that choice. If a is Inv or Obs, then v is the utility of exploitation action m. If a is Obs and no agent performed an exploitation action last round, then there is no exploitation action to be observed, hence $m = \emptyset$ and $v = 0$. If a is some X_i, then m will be the same X_i and v will be the utility the agent receives for that action. The *agent history* for agent α is a sequence of such action-percept pairs, $h_\alpha = \langle (a_1, (m_1, v_1)), \ldots, (a_k, (m_k, v_k)) \rangle$. As a special case, the empty (initial) history is $\langle\rangle$.

Example 2. The history for $\mathsf{I2O}$ in Example 1 is:

$$h_{\mathsf{I2O}} = \langle (\mathsf{Inv}, (\mathsf{X}_3, 8)), (\mathsf{Inv}, (\mathsf{X}_4, 5)), (\mathsf{Obs}, (\mathsf{X}_1, 3)), (\mathsf{X}_3, (\mathsf{X}_3, 8)), \ldots \rangle$$

To concatenate a new action-percept pair onto the end of a history, I use the $\circ$ symbol. For example, $h_\alpha \circ (a, (m, v))$ is the history h_α concatenated with the action-percept pair $(a, (m, v))$. Further, for $h_\alpha = \langle p_1, p_2, \ldots, p_k \rangle$, where each p_i is some action-percept pair, I let $h_\alpha[i] = p_i$, and $h_\alpha[i, \ldots, j]$ be the subhistory $\langle p_i, \ldots, p_j \rangle$.

Strategies. The Cultaptation game is ultimately a competition among *strategies*. Here, a strategy is a function from histories to the set of possible actions: $s : h_\alpha \mapsto m$, where h_α is a history of an agent using s and m is Inv, Obs or X_i for some i. Since each strategy may depend on the entire history, the set of possible

strategies is huge;[1] but any particular Cultaptation game is a competition among a much smaller set of strategies **S**, which I will call the *set of available strategies*. For example, if there are n contestants, each of whom chooses a strategy to enter into the game, then in this case,

$$\mathbf{S} = \{\text{the strategies chosen by the contestants}\}. \tag{2.1}$$

Each strategy in **S** may be used by many different agents, and the strategy profile at each round of the game may change many times as the game progresses. When an agent reproduces, it passes its strategy on to a newly created agent, with the per-round utility of each agent determining its likelihood of reproduction. A strategy's success is measured by its average prevalence over the last quarter of the game [15].

The replication dynamics work as follows. On each round, each agent has a 2% chance of dying. As such, I also include a parameter d in my formulation representing the probability of death (d defaults to 0.02). Upon death, an agent is removed from the game and replaced by a new agent, whose strategy is chosen using the *reproduction* and *mutation* mechanisms described below. Mutation happens 2% of the time, and reproduction happens 98% of the time.

Reproduction. When reproduction occurs, the social learning strategy used by the newborn agent is chosen from the strategies of agents currently alive with a probability proportional to their per-round utility (the utility gained by an agent

[1] The number of possible mixed strategies is, of course, infinite. But even if we consider only pure strategies, the number is quite huge. For a 10,000-round Cultaptation game of the type used in the Cultaptation tournament, a loose lower bound on the number of pure strategies is $100^{9.4\times10^{20155}}$ [19]. In contrast, it has been estimated [20] that the total number of atoms in the observable universe is only about 10^{78} to 10^{82}.

divided by the number of rounds the agent has lived). The agent with the highest per-round utility is thus the most likely to propagate its strategy on reproduction.

> **Example 3.** Again looking at the sequences of actions in Table 2.1, we see that both agents would have equal chance of reproducing on round 1. However, on round 2 I1 has a per-round utility of 1.5, while I2O has a per-round utility of 0, meaning I1 gets 100% of the reproductions occurring on round 2. Round three is the same, but on round 4 I1 has a per round utility of 2.25 and I2O has a per-round utility of 2. This means that I1 gets $100 \cdot 2.25/4.25 = 53\%$ of the reproductions and I2O gets $100 \cdot 2/4.25 = 47\%$ of the reproductions on round 4.

Mutation. In Cultaptation, *mutation* does not refer to changes in an agent's codebase (as in genetic programming). Instead, it means that the new agent's strategy s is chosen uniformly at random from the set of available strategies, regardless of whether any agents used s on the previous round. For instance, if there were a Cultaptation game pitting strategies I1 and I2O against one another, then a new mutated agent would be equally likely to have either strategy I1 or I2O, even if there were no living agents with strategy I1.

Game Types. In the Cultaptation tournament [17], two types of games were played: pairwise games and melee games. A *pairwise* game was played with an invading strategy and a defending strategy. The defending strategy began play with a population of 100 agents, while the invading strategy began with none. Mutation was also disabled for the first 100 rounds, to allow the defending strategy time to begin earning utility. After 100 rounds, mutation was enabled and the invader had the challenging task of establishing a foothold in a population consisting entirely of agents using the defending strategy (most of whom would have had time to find several high-payoff actions). Since the pairwise games provide a clear early-

game advantage to the defender, they were typically played twice with the invader and defender swapping roles on the second game. A *melee* game was played with n strategies, for some $n > 2$. Initially, the population of 100 agents was evenly divided between each strategy in the game. Mutation was disabled for the last quarter of the game, so that it would not influence results when strategies had similar fitness.

Scoring. If we have k social learning strategies $s_1, \ldots, s_k$ playing Cultaptation, then on any given round there will be some number n_j of agents using strategy s_j, for $1 \leq j \leq k$. Strategy s_j's *score* for the game is *the average value of* n_j *over the final 2,500 rounds of the game.* The strategy with the highest score is declared the winner.

The only way an agent may affect n_j is through reproduction. I will show in Section 4.2.2 that any strategy maximizing an agent's expected per-round utility (defined in Section 4.1.5) will also maximize its reproduction. I will therefore focus on computing the expected per-round utility.

2.2 Motivating Discussion

The purpose of this section is to explain the motivations for several aspects of my work:

- Sections 2.2.1 and 2.2.2 give examples of types of strategies that seem like they should work well at first glance, but can have unexpectedly bad consequences. The existence of such situations motivate the algorithms described later in this proposal, which perform a game tree search in order to consider strategies'

long-term consequences.

- An important way of getting insight into a game is to examine its best-response strategies; and this approach is at the heart of my formal analysis and my game-tree search algorithms. Section 2.2.3 explains some issues that are important for finding best-response strategies in Cultaptation.

2.2.1 Innovation, Observation, and Structural Shocks

If we want to acquire a new action to exploit, then what is the best way of doing it: to observe, or to innovate? At first glance, observing might seem to be the best approach. If the other agents in the environment are competent, then it is likely that they are exploiting actions that have high payoffs, hence we should be able to acquire a better action by observing them than by innovating. This suggests that an optimal agent will rely heavily on observation actions. However, the following example shows that relying *only* on observation actions can lead to disastrous consequences if there is a structural shock, i.e., a large change in the value of an exploitation action.[2]

> **Example 4. Structural shocks:** Figure 2.1 shows a Cultaptation game in which all agents use the following strategy: each agent begins with a single Obs action, followed by a single Inv action if the Obs action returns $\emptyset$,[3] in order to obtain an exploitation action X_i which the agent will use in all subsequent rounds.
>
> Agent A3 acquires action X_4 by doing an unsuccessful Obs followed by an Inv; and A1 and A2 acquire X_4 by observing A3. At first, X_4 is far

[2] I have borrowed this term from the Economics literature, where it has an analogous meaning (e.g., [21, 22]).

[3] This will generally only happen on the first round of the game, before any agent has obtained an exploitation action.

Round	X_1	X_2	X_3	X_4	A1	A2	A3
1	2	4	1	9	N/A	N/A	$(\mathsf{Obs}, (\emptyset, \cdot))$
2	2	4	1	9	N/A	N/A	$(\mathsf{Inv}, (\mathsf{X}_4, 9))$
3	2	4	1	9	Birth	N/A	$(\mathsf{X}_4, (\cdot, 9))$
4	5	4	1	9	$(\mathsf{Obs}, (\mathsf{X}_4, 9))$	Birth	$(\mathsf{X}_4, (\cdot, 9))$
5	5	2	1	9	$(\mathsf{X}_4, (\cdot, 9))$	$(\mathsf{Obs}, (\mathsf{X}_4, 9))$	$(\mathsf{X}_4, (\cdot, 9))$
6	1	2	1	9	$(\mathsf{X}_4, (\cdot, 9))$	$(\mathsf{X}_4, (\cdot, 9))$	$(\mathsf{X}_4, (\cdot, 9))$
7	1	2	8	9	$(\mathsf{X}_4, (\cdot, 9))$	$(\mathsf{X}_4, (\cdot, 9))$	Death
8	1	2	8	1	$(\mathsf{X}_4, (\cdot, 1))$	$(\mathsf{X}_4, (\cdot, 1))$	N/A
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Figure 2.1: An example of a game in which there is a large structural shock. The columns for the exploitation actions X_i show their values at each round, and the columns for agents A1–A3 show their histories. Note that by round 6, all agents choose action X_4, which has changed to a very low value. Since none of the agents are innovating, none of them can find the newly optimal action X_3.

> better than the other exploitation actions, so all of the agents do well by using it. On round 8, the action X_4 changes to the lowest possible value, but the agents continue to use it anyway. Furthermore, any time a new agent is born, it will observe them using X_4 and will start using it too.

This is a pathological case where the best action has disappeared and the agents are in a sense "stuck" exploiting the suboptimal result. Their only way out is if all agents die at once, so that one of the newly born agents is forced to innovate. Experiments in Section 5.3.1 show that in some cases, situations like these are a big enough risk that a near-best response strategy will choose innovation moves more frequently than observation moves.

2.2.2 Innovation and Observation Versus Exploitation

One might also think that agents should perform all of their innovation and observation actions first, so that they have as many options as possible when choosing an action to exploit. However, as Raboin et al. [23] demonstrate, this intuition

is not always correct. Because the game selects which agents reproduce based on average per-round utility, not total accumulated utility, it is frequently better for newborn agents to exploit one of the first actions it encounters, even if this action has a mediocre payoff (e.g., exploiting an action with value 10 on the second round of an agent's life gives it as much per-round payoff as exploiting an action with value 50 on the tenth round). Once the agent has at least some per-round utility so that it has a nonzero chance of reproducing, it can then begin searching for a high-valued action to exploit for the rest of its lifetime.

2.2.3 Best-Response Strategies in Cultaptation

A widely used technique for getting insight about a game (e.g., see [24]) is to look at the game's best-response strategies. Given an agent α and a *strategy profile* (i.e., an assignment of strategies to agents) $\mathbf{s}_{-\alpha}$ for the agents other than α, α's *best response* is a strategy s_{opt} that maximizes α's expected utility if the other agents use the strategies in $\mathbf{s}_{-\alpha}$.

In Cultaptation, it is more useful to consider a best response to the set of available strategies $\mathbf{S}$, rather than any particular strategy profile. During the course of a Cultaptation game, the strategy profile will change many times as agents die and other agents are born to take their places. Each strategy in $\mathbf{S}$ will be scored based on its average performance over the course of the game; and we will see (in Section 4.2.2) that given $\mathbf{S}$, each strategy's score is independent of the initial strategy profile if the game is sufficiently long.

If G is a Cultaptation game (i.e., a set of values for game parameters such as the number of agents, set of available actions, probability distribution over their payoffs; see Section 4.1 for details), then for any agent α, any set of available strategies $\mathbf{S}$, and any history h_α for α, there is a probability distribution $\pi_{\mathsf{Obs}}(a|h_\alpha, \mathbf{S})$ that gives the probability of observing each action a, given $\mathbf{S}$ and h_α. Given π_{Obs} and G, we can calculate the probability of each possible outcome for each action the agent might take, which will allow us to determine the best response to $\mathbf{S}$. To compute π_{Obs} is not feasible except in general, but it is possible to compute approximations of it in some special cases (e.g., cases in which all of the agents, or all of the agents other than α, use the same strategy). That is the approach used in this proposal.

Chapter 3

Related Work

In this section I will discuss related work on social learning and on computational techniques related to my own.

3.1 Social Learning

The Cultaptation social learning competition offers insight into open questions in behavioral and cultural evolution. An analysis of the competition is provided by Rendell et al. [17]. Of the strategies entered into the competition, those that performed the best were those that greatly favored observation actions over innovation actions, and the top performing strategy learned almost exclusively through observation. This was considered surprising, since several strong arguments have previously been made for why social learning isn't purely beneficial [7, 16]. However, this result is consistent with observations made during my own experiments, in which the ϵ-best-response strategy rarely did innovation (see Section 5.3).

In previous work, Carr et al. showed how to compute optimal strategies for a highly simplified versions of the Cultaptation social learning game [25]. Their paper simplifies the game by completely removing the observation action—which prevents the agents from interacting with each other in any way whatsoever, thereby transforming the game into a single-agent game rather than a multi-agent game.

Their model also assumes that exploitable actions cannot change value once they have been learned, which overlooks a key part of the full social learning game.

Wisdom and Goldstone attempted to study social learning strategies using a game similar to Cultaptation, but using humans rather than computer agents [18]. Their game environment consisted of a group of "creatures," each of which had some hidden utility. The agents' objective was to select a subset of the creatures to create a "team," which was assigned a utility based on the creatures used to create it. Agents had a series of rounds in which to modify their team, and on each round they were allowed to see the teams chosen by other agents on the previous round (and in some cases, the utility of the other agents' teams), and the object of the game was to maximize the utility of one's team. In this game, the acts of keeping a creature on one's team, choosing a creature that another agent has used, and choosing a creature no one has yet used correspond to exploitation, observation, and innovation (respectively) in the Cultaptation game.

The successful strategies Wisdom and Goldstone saw are similar to those used by the strategies found by my algorithm: they keep most of the creatures on their team the same from round to round (which corresponds in Cultaptation to performing mostly exploitation actions), and new creatures are mostly drawn from other agents' teams (which corresponds to preferring observation over innovation in Cultaptation). However, Wisdom and Goldstone highlight these characteristics as interesting because they run contrary to the conventional wisdom for social learning strategies, which suggests that broader exploration should lead to better performance, and therefore that successful strategies should innovate more often [16]. In

this case, analyzing the strategies found by my algorithm allowed me to draw the same conclusions about what works well. This gives more evidence that the conventional wisdom on social learning [7, 16] may be mistaken.

How best to learn in a social environment is still considered a nontrivial problem. Barnard and Sibly show that if a large portion of the population is learning only socially, and there are few information producers, then the utility of social learning goes down [9]. Thus, indiscriminate observation is not always the best strategy, and there are indeed situations where innovation is appropriate. Authors such as Laland have attempted to produce simple models for determining when one choice is preferable to the other [8]. Game theoretic approaches have also been used to explore this subject, but it is still ongoing research [26, 27]. Giraldeau et al. offer reasons why social information can become unreliable. Both biological factors, and the limitations of observation, can significantly degrade the quality of information learned socially [11].

Work by Nettle outlines the circumstances in which verbal communication is evolutionarily adaptive, and why few species have developed the ability to use language despite its apparent advantages [10]. Nettle uses a significantly simpler model than the Cultaptation game, but provides insight that may be useful to understanding social learning in general. In Nettle's model, the population reaches an equilibrium at a point where both individual and social learning occur. The point of equilibrium is affected by the quality of observed information and the rate of change of the environment.

3.2 Related Computational Techniques

The *restless bandit* problem, a generalization of the stochastic multi-armed bandit problem that accounts for probability of change, is cited as the basis for the rules of the Cultaptation tournament [17]. The rules of the Cultaptation game differ from the *restless bandit* problem by including other agents, making observation actions possible and complicating the game significantly. I also show in Section 4.2.2 that maximizing total payoff, the goal of the restless bandit problem, is different from maximizing expected per-round utility (EPRU) of an agent in the Cultaptation tournament.

The *restless bandit* problem is known to be *PSPACE*-complete, meaning it is difficult to compute optimal solutions for in practice [28, 29]. Multi-armed bandit problems have previously been used to study the tradeoff between exploitation and exploration in learning environments [30, 31].

As discussed later in Section 4.1, finding a best-response strategy in Cultaptation is basically equivalent to finding an optimal policy for a Markov Decision Process. Consequently, my algorithm for finding near-best-response strategies has several similarities to the approach used by Kearns *et al.* to find near-optimal policies for large MDPs [32]. Both algorithms use the discount factor of the MDP (which, in the case of Cultaptation, is the probability of death d) and the desired accuracy ϵ to create a horizon for their search, and the depth h_α of this horizon depends on the discount factor and the branching factor, but not on the size of the full state space (unlike conventional MDP algorithms). Thus, both their algorithm

and mine also have running time exponential in $1/\epsilon$ and in the branching factor. However, the algorithm provided by Kearns et al. was designed as an online algorithm, so it only returns the near-optimal action for the state at the root of the search tree. Mine, on the other hand, returns a strategy specifying which action the agent should take for all states that can occur on the first h_α rounds. This means that the exponential-time algorithm only needs to run once to generate an entire strategy, rather than once per agent per round in each game we simulate.

Many algorithms for optimal control of an MDP have been developed, however they all have running time that grows linearly with the size of the state space of the MDP. This makes them intractable for problems like Cultaptation, which have exponentially large state spaces. Several approaches for *near-optimal* control, which produces a policy within some ϵ of optimal, have been developed [32, 33, 34].

Chapter 4

Game Analysis

This chapter presents the formal model of Cultaptation used throughout the rest of the work, as well as the derivation of a formula that, given the parameters of the game environment and the other strategies in the game, will allow us to predict which strategy has an evolutionary advantage.

4.1 Formal Model

In this section I introduce a formal mathematical model of Cultaptation games. A glossary of the notation used in this proposal is provided as Table 4.1.

Game Definition. Cultaptation requires a number of parameters to determine exactly how it will run. Therefore, I will define the *game parameters*, G, to be a set of values for the following: N, the number of agents; M, the number of exploitation actions in the game; c, the probability that an exploitation action changes its utility each round; π, the probability distribution used to assign a utility value to each exploitation action, both the outset and each time an action's utility changes; and d, the probability of death. In the Cultaptation tournament, only the values of N, M, and d were known ahead of time, but for this analysis I use the values of the other parameters as well.

Recall that in the Cultaptation tournament [15], each evolutionary simulation

N	Number of agents in the environment.
$\mathbf{S}$	The set of available strategies. Agents may only use strategies in $\mathbf{S}$.
r	Number of the current round, ranging from 1 to ∞.
c	The probability of change on all rounds.
d	The probability of death on all rounds.
$(a,(m,v))$	An action-percept pair in which the action a returns the percept (m,v).
h_α	Agent history for α. A sequence of action-percept pairs experienced by agent α.
$h_\alpha[i]$	The i-th action-percept pair in h_α.
$X(h_\alpha)$	Number of exploitable actions given history h_α.
M	Number of exploitation actions in the game.
π	Probability distribution for the new value of any action whose value changed at round r.
$\pi_{\mathsf{Obs}}(m,v\|h_\alpha,\mathbf{S})$	Probability that Obs returns action m with value v, given history h_α and available strategies $\mathbf{S}$.
$\pi_{\mathsf{Inv}}(v\|r)$	Probability that Inv returns an action with value v on round r.
V	The set of potential utility values.
$P(h'_\alpha\|h_\alpha,a,\mathbf{S})$	Probability of transitioning to history h'_α if α performs action a with history h_α and available strategies $\mathbf{S}$.
$L(\|h_\alpha\|)$	Probability that α lives long enough to experience history h_α.
T	Set of all action-percept pairs of the form $(a,(m,v))$.

Table 4.1: A glossary of notation used in this proposal.

was a contest between two or more strategies submitted to the tournament. Thus, there is a fixed set of strategies that are allowed to occur in a given simulation. I will call this the *set of available strategies* $\mathbf{S}$, where $\mathbf{S} = \{s_1, s_2, ..., s_\ell\}$ for some finite ℓ (i.e. in pairwise games $\ell = 2$, in melee games $\ell > 2$). Any strategy profile $\mathbf{s}$ that occurs in the simulation will consist only of strategies in $\mathbf{S}$. When an agent is chosen to be replaced via mutation, its new strategy is selected at random from the strategies in $\mathbf{S}$.

A Cultaptation game can now be defined formally, as follows. A *Cultaptation game* is an ℓ-player game, in which each player receives the game parameters G as

input. Each player then simultaneously chooses a strategy to put into the set of available strategies. Let player i's strategy be s_i, so that $\mathbf{S} = \{s_1, s_2, ..., s_\ell\}$. The pair $(G, \mathbf{S})$ is an *instance* of G. In $(G, \mathbf{S})$, each player i will receive a payoff equal to score(s_i), defined below.

Scoring. The version of Cultaptation used in the tournament continued for 10,000 rounds, and each strategy was assigned a score equal to its *average population* over the last 2,500 rounds. But as is often done in analyses of repeated games, the formal model assumes an *infinite Cultaptation game*, i.e., the game continues for an infinite number of rounds, and the score for strategy s is its average population over the entire game:

$$\text{score}(s) = \lim_{r \to \infty} \frac{\sum_{j=1}^{r} p(s, j)}{r},$$

where $p(s, j)$ is the population size of agents using strategy s on round j. This greatly simplifies the analysis in Section 4.1.5, by allowing us to average out the various sources of noise present in the game.

Actions. The rest of the formal model will be constructed from the perspective of an arbitrary agent, α, in a given infinite Cultaptation game instance $(G, \mathbf{S})$. I use r for the number of a round, and $X(h_\alpha)$ to specify the number of exploitation actions available after history h_α. After all exploitation actions $\mathsf{X}_1, \ldots, \mathsf{X}_M$ have been innovated or observed in a history h_α, then $X(h_\alpha) = M$ and innovation actions become illegal.

I model the payoffs supplied for exploitation actions X_i by a probability distribution π. $\pi(v)$ is the probability of an action having payoff v at the start of the

game instance. $\pi(v)$ is also the probability that, when an action changes its payoff, the new payoff is v. I let V be the set of all action values that may occur with non-zero probability:

$$V = \{v \mid \pi(v) > 0\},$$

where V has finite size.

If we let $\pi_{\mathsf{Inv}}(v|r)$ be the probability that value v is innovated on round r, it can be defined recursively in terms of c and π as:

$$\pi_{\mathsf{Inv}}(v|r) = \begin{cases} \pi(v), & \text{if } r{=}0, \\ c\pi(v) + (1-c)\pi_{\mathsf{Inv}}(v|r-1), & \text{otherwise.} \end{cases}$$

That is, initially the chance that Inv will return an action with value v is determined by the given distribution $\pi(v)$. On later rounds ($r > 0$) the chance that Inv will return an action with value v is the chance that an action's value changed to v on the current round (given by $c\pi(v)$), plus the chance that an action's value was v on the previous round and it did not change this round.

While computing the probability distribution for utilities of actions returned by Inv was fairly straightforward, computing a similar distribution for Obs actions is significantly more difficult. Let α be any agent, and $\mathbf{S}$ be the set of available strategies. From $\mathbf{S}$ we can get a probability distribution over the other agents' actions in any given situation; and from this we can derive $\pi_{\mathsf{Obs}}(m, v|h_\alpha, \mathbf{S})$, the probability that Obs would return the action-percept pair (m, v), given history h_α.

In order to derive π_{Obs}, we must consider each possible strategy profile $\mathbf{s}_{-\alpha}$ for agents besides α, determine how likely that strategy profile is to occur, and then determine what each agent in $\mathbf{s}_{-\alpha}$ will do for every possible sequence of actions they

could have encountered, bounded only by the percepts the agent has received in h_α. As discussed in Section 2.2.3, the number of possible histories alone is astronomically large. Since π_{Obs} is conditioned on each possible history it will be larger still, so in any practical implementation the best we can do is to approximate π_{Obs} (Section 5.1.5 describes how we will do this). But for the theoretical development, I will assume we have an oracle for π_{Obs}, that will tell us exactly how likely we are to observe any given action-utility pair.

In what follows, I will show that given π, π_{Obs}, V, and $\mathbf{S}$, we can calculate the possible outcomes of each action the agent may take, and the probability of each of these outcomes. This allows us to treat an infinite Cultaptation game as a Markov Decision Process (MDP) [35]. Calculating the best response in this case is equivalent to finding an optimal control policy for an MDP.

4.1.1 Transition Probabilities

A transition probability function $P(h'_\alpha|h_\alpha, a, \mathbf{S})$ defines the probability of transitioning from history h_α to history $h'_\alpha = h_\alpha \circ (a, (m, v))$ in the next round if an agent α performs action a. These transition probabilities are for the case where α does not die before reaching h'_α; I introduce functions to account for the probability of death in Section 4.1.2.

There are three cases for what $P(h'_\alpha|h_\alpha, a, \mathbf{S})$ might be, depending on whether a is an innovation, observation, or exploitation action:

- If $a = \mathsf{Inv}$, then

$$P(h_\alpha \circ (\mathsf{Inv}, (m, v))|h_\alpha, \mathsf{Inv}, \mathbf{S}) = \begin{cases} \frac{\pi_{\mathsf{Inv}}(v|r)}{M - X(h_\alpha)} & \text{if } h_\alpha \text{ contains no percepts that} \\ & \text{contain the action } m, \\ 0 & \text{otherwise.} \end{cases} \tag{4.1}$$

- Recall that an agent cannot innovate action m if it has already encountered m by innovating or observing. Observation actions are not subject to the same restriction, so if $a = \mathsf{Obs}$, then

$$P(h_\alpha \circ (\mathsf{Obs}, (m, v))|h_\alpha, \mathsf{Obs}, \mathbf{S}) = \pi_{\mathsf{Obs}}(m, v|h_\alpha, \mathbf{S}) \tag{4.2}$$

where $\pi_{\mathsf{Obs}}(m, v|h_\alpha, \mathbf{S})$ models the exploitation behavior of the other agents in the environment. Obviously, the exact probability distribution will depend on the composition of strategies used by these agents. The above definition is general enough to support a wide range of environments; and in Section 5.1.5 I will discuss one potential way to model this function for a more specific set of environments.

- Finally, if $a = \mathsf{X}_m$, then h_α must contain at least one percept for X_m. Let r be the last round at which the last such percept occurred. For the case where X_m's utility did not change since round r, we have

$$P(h_\alpha \circ (\mathsf{X}_m, (m, v))|h_\alpha, \mathsf{X}_m, \mathbf{S}) = \underbrace{(1-c)^{|h_\alpha|-r}}_{\text{prob. of not changing}} \underbrace{+ c\pi(v) \sum_{j=r}^{|h_\alpha|} (1-c)^{|h_\alpha|-j}}_{\text{prob. of changing back to } v.} \tag{4.3}$$

For the case where X_m's utility did change since round r, we have

$$P(h_\alpha \circ (\mathsf{X}_m, (m, v))|h_\alpha, \mathsf{X}_m, \mathbf{S}) = c\pi(v) \sum_{j=r}^{|h_\alpha|} (1-c)^{|h_\alpha|-j} \tag{4.4}$$

which is similar, but assumes that the value must have changed at least once.

In all other cases, no transition from h_α to h'_α is possible, so $P(h'_\alpha|h_\alpha, a, \mathbf{S}) = 0$.

Probability of Reaching a History

We will frequently be interested in $P(h_\alpha|s, \mathbf{S})$, the probability of history h_α occurring given that the agent is following some strategy $s \in \mathbf{S}$. We will be able to derive $P(h_\alpha|s, \mathbf{S})$ iteratively, calculating the probability of each step of history h_α occurring using the functions derived above.

Specifically, $P(h_\alpha|s, \mathbf{S})$ is the probability that each $h_\alpha[i] = (a_i, (m_i, v_i))$ occurs given the action chosen by the strategy in the history $h_\alpha[1, \ldots, i-1] = (a_1, (m_1, v_1)) \cdot \cdots \cdot (a_{i-1}, (m_{i-1}, v_{i-1}))$, or:

$$P(h_\alpha|s, \mathbf{S}) = \prod_{i=1}^{|h_\alpha|-1} P(h_\alpha[1, \ldots, i] \circ h_\alpha[i+1]|h_\alpha[1, \ldots, i], s(h_\alpha[1, \ldots, i]), \mathbf{S}) \tag{4.5}$$

4.1.2 Accounting for Probability of Death

The probability of an agent living long enough to experience history h_α depends on the probability of death. It is

$$L(|h_\alpha|) = (1-d)^{|h_\alpha|-1}. \tag{4.6}$$

When we calculate the probability of reaching a given history h_α, we will generally multiply it by $L(|h_\alpha|)$ to account for the chance that the agent dies before reaching h_α.

Sometimes we will also be interested in the probability that a randomly-selected agent has history h_α. For this we will need to know the probability that a randomly-selected agent is exactly $|h_\alpha|$ rounds old, which is simply:

$$\frac{L(|h_\alpha|)}{\sum_{i=1}^{\infty} L(i)} = \frac{L(|h_\alpha|)}{\frac{1}{1-(1-d)}} = \frac{L(|h_\alpha|)}{\frac{1}{d}} = d\, L(|h_\alpha|). \tag{4.7}$$

4.1.3 Utility Functions

A utility function $U((a,(m,v)))$ defines the utility gleaned on action-percept pair $(a,(m,v))$:

$$U((a,(m,v))) = \begin{cases} v, & \text{if } \exists i \text{ such that } a = \mathsf{X}_i, \\ 0, & \text{otherwise.} \end{cases} \tag{4.8}$$

Notice that $U(\cdot)$ is only non-zero on exploitation actions.

The *per-round utility (PRU)* of history h_α, where

$$h_\alpha = (a_1,(m_1,v_1)) \circ \cdots \circ (a_{|h_\alpha|},(m_{|h_\alpha|},v_{|h_\alpha|})),$$

is defined to be the sum of the utility acquired in that history divided by the history's length:

$$\mathrm{PRU}(h_\alpha) = \frac{1}{|h_\alpha|} \sum_{i=1}^{|h_\alpha|} U((a_i,(m_i,v_i))) \tag{4.9}$$

4.1.4 Strategy Representation

A strategy s is defined as a function mapping each history $h_\alpha \in H$ to the agent's next action $s(h_\alpha) \in \{\mathsf{Inv}, \mathsf{Obs}, \mathsf{X}_1, \ldots, \mathsf{X}_M\}$. For instance, the strategy $\mathsf{I1}$

from Example 1 is defined by the function:

$$s_{\mathsf{I1}}(h_\alpha) = \begin{cases} \mathsf{Inv}, & \text{if } h_\alpha \text{ is empty,} \\ \mathsf{X}_i, & \text{for } h_\alpha = (\mathsf{Inv}, (\mathsf{X}_i, v)), \ldots \end{cases}$$

In this proposal we will deal with *partially specified* strategies. A partially specified strategy is a mixed strategy (i.e., a probability distribution over a set of pure strategies) that is defined by a finite set Q of history action pairs ($Q \subset H \times \{\mathsf{Inv}, \mathsf{Obs}, \mathsf{X}_1, \ldots, \mathsf{X}_M\}$), in which each $h_\alpha \in H$ appears at most once. Given any history h_α, if there is an action m such that $(h_\alpha, m) \in Q$, then s_Q chooses the action m. Otherwise, s_Q chooses an action arbitrarily from all actions that are legal in h_α. Partially specified strategies have the advantage of being guaranteed to be finitely representable.

4.1.5 Evaluating Strategies

At each round, an agent with history h_α has reproductive fitness $\mathrm{PRU}(h_\alpha)$, and agents are selected to reproduce with probability proportional to their reproductive fitness (i.e., using the replicator equation [36]). Since a strategy's score is a function of its average population over the course of the game, we want some metric that allows us to compare the expected reproductive fitness of two strategies. This will allow us to predict which strategy is more likely to win.

At first glance, it may appear that the way to predict which strategy will have higher expected reproductive fitness is to compare their expected utilities. However, prior work has shown that this is not the case: *in Cultaptation, a strategy's expected reproductive fitness is not necessarily proportional to its expected utility* [23]. I now

Table 4.2: The expected utility and expected reproductive fitness of s_{II} and s_{IEI} from Example 5. s_{IEI} has a higher expected reproductive fitness, and therefore will be likely to win a game against s_{II}, even though s_{II} has a higher expected utility.

	Expected Utility	Expected Reproductive Fitness
s_{II}	**90.25**	65.074
s_{IEI}	88.825	**65.185**

present a simple example that illustrates this phenomenon.

> **Example 5.** Consider an infinite Cultaptation game with no probability of change, no observation actions, probability of death $d = 0.05$, two exploitation actions valued at 65 and 100, and an innovate action that will return either exploitation action with uniform probability. This means that an agent needs to perform at most two innovate actions to have knowledge of the action with value 100, since innovating does not return an action the agent already knows.
>
> Let us compare two strategies: s_{II} and s_{IEI}. Both strategies will perform an innovate as their first action. If the action they learn has value 100, both strategies will exploit that action until the agent dies. If the action learned has value 65, s_{II} will perform a second innovate on its next turn, learning the action with value 100, and will exploit that action until its agent dies. Meanwhile, s_{IEI} will exploit the action with value 65 once, before performing an innovate on its third turn to learn the action with value 100. It then exploits this action until its agent dies.
>
> Since the two strategies are identical when they learn the action with value 100 on their first action, and since this case is equally likely to be encountered by both strategies, we can ignore it for the purposes of comparing them. For the rest of this analysis we will assume the first innovate returns the action with value 65. In this case, we can calculate the expected utility for both strategies using geometric series, and we can calculate their expected reproductive fitnesses using methods described in Section 4.2. Table 4.2 presents these values. While s_{II} has a higher expected utility, since it exploits the action with value 100 more often, s_{IEI} has a higher expected reproductive fitness, since it does not wait as long to begin exploiting. Therefore, s_{IEI} will be the likely winner in a contest between these two strategies.

Since we cannot always use a strategy's expected utility to determine whether it is expected to win, we will instead compute a strategy's expected reproductive fitness directly, by computing its Expected Per-Round Utility.

Definition. The *Expected Per-Round Utility* for a strategy s_α, $\text{EPRU}(s_\alpha \mid G, \mathbf{S})$, is the expected value of $\text{PRU}(h_\alpha)$ over all possible histories h_α for a randomly-selected agent α using strategy $s_\alpha \in \mathbf{S}$ in an infinite Cultaptation game instance $(G, \mathbf{S})$. □

To calculate $\text{EPRU}(s_\alpha \mid G, \mathbf{S})$, we look at each possible history h_α and multiply $\text{PRU}(h_\alpha)$ by the probability that a randomly-chosen agent using s_α has history h_α. This probability is equal to the probability that a randomly-chosen agent is $|h_\alpha|$ rounds old (Equation 4.7) times the probability of reaching history h_α in $|h_\alpha|$ steps using strategy s_α (Equation 4.5). Hence, the EPRU of a strategy is:

$$\text{EPRU}(s_\alpha \mid G, \mathbf{S}) = \sum_{h_\alpha \in H} d\, L(|h_\alpha|) \times P(h_\alpha | s_\alpha, \mathbf{S}) \times \text{PRU}(h_\alpha).$$

Note that for a given environment, the probability of death d is a constant. Hence, in the analysis I will frequently factor it out.

> **Example 6.** Recall the innovate-once strategy, which innovates once to learn an action and then exploits that action until it dies. Suppose this strategy exists in an environment with a probability of death of 0.2 and only one possible exploit action with non-changing value 10. All agents using this strategy will therefore learn the only action on their first round, and then exploit an action with value 10 on all subsequent rounds. Hence, there is only one possible history for a j-round old agent using this strategy, and its per-round utility is $10 \cdot (j-1)/j$. The probability that a randomly-selected agent will be j rounds old will be $0.2 \cdot L(j) = 0.2 \cdot 0.8^{j-1}$. Thus the expected per-round utility achieved by this strategy in this environment is $\sum_{j=1}^{\infty} 0.2 \cdot 0.8^{j-1} \cdot 10 \cdot (j-1)/j$.

4.2 Analysis of EPRU

In this section I examine methods for computing the expected per-round utility of a strategy. First I present a method for computing an approximation to the EPRU

for given a strategy, then I present a proof that a strategy maximizing EPRU will also maximize its average population in an infinite Cultaptation game instance.

4.2.1 Computation of EPRU

I will now define a formula that can be used to compute EPRU exactly for a given strategy s. The definition of EPRU given in Section 4.1.5 used a "backward" view: for every possible history h_α, it looked back through h_α to determine $\text{PRU}(h_\alpha)$. Notice, however, that h_α must have some preceding history h'_α, where $h_\alpha = h'_\alpha \circ t$ for some action-percept pair t. This definition of EPRU must examine h'_α and h_α independently, even though their only difference is the addition of t.

For this reason, it will make more sense computationally to use a "forward" view of EPRU: we will construct a recursive function on s and h_α which, for each possible $h_\alpha \circ t$:

- calculates the per-round utility gained from t, both for history $h_\alpha \circ t$ and for all histories that can be reached from $h_\alpha \circ t$, and then
- recurses on s and $h_\alpha \circ t$.

For the calculation in the first bullet, we will use the formula $\text{EV}_{exp}(r, v)$, which computes the expected amount of per-round utility we gain (on the current round and on future rounds) by exploiting a value v on round r:

$$\text{EV}_{exp}(r, v) = \sum_{j=r}^{\infty} \frac{L(j)v}{j} = v \sum_{j=r}^{\infty} \frac{1}{j}(1-d)^{j-1}. \tag{4.10}$$

Using known properties of infinite series, EV_{exp} can also be expressed as

$$\begin{aligned}\mathrm{EV}_{exp}(r, v) &= v\left(\sum_{i=1}^{\infty}\frac{1}{i}(1-d)^{i-1} - \sum_{i=1}^{r-1}\frac{1}{i}(1-d)^{i-1}\right) \\ &= v\left(\frac{\ln d}{d-1} - \sum_{i=1}^{r-1}\frac{1}{i}(1-d)^{i-1}\right) \end{aligned} \tag{4.11}$$

and is therefore computable.

We can now express the expected per-round utility of a strategy s recursively in terms of the average per-round payoff of an agent.

$$\mathrm{EPRU}_{\mathrm{alt}}(s, h_\alpha \mid G, \mathbf{S}) = \tag{4.12}$$

$$\sum_{t\in T} P(h_\alpha \circ t | h_\alpha, s(h_\alpha), \mathbf{S}) \cdot (\mathrm{EV}_{exp}(|h_\alpha \circ t|, U(t)) + \mathrm{EPRU}_{\mathrm{alt}}(s, h_\alpha \circ t \mid G, \mathbf{S}))$$

where T is the set of all possible action-percept pairs, and $h_\alpha \circ t$ represents a possible history on the next round. Note that the size of T is finite. A proof that $\mathrm{EPRU}(s \mid G, \mathbf{S})/d = \mathrm{EPRU}_{\mathrm{alt}}(s, \langle\rangle \mid G, \mathbf{S})$ is included in [19].

Unfortunately, computing $\mathrm{EPRU}_{\mathrm{alt}}$ is not possible since it suffers from infinite recursion. To handle this, I introduce a depth-limited computation of $\mathrm{EPRU}_{\mathrm{alt}}$, which only computes the portion of the total EPRU contributed by the first k rounds:

$$\mathrm{EPRU}^k_{\mathrm{alt}}(s, h_\alpha \mid G, \mathbf{S}) = \tag{4.13}$$

$$\begin{cases} 0 & \text{If } k = 0 \\ \sum_{t\in T} P(h_\alpha \circ t | h_\alpha, s(h_\alpha), \mathbf{S}) \times & \\ (\mathrm{EV}_{exp}(|h_\alpha|, U(t)) + \mathrm{EPRU}^{k-1}_{\mathrm{alt}}(s, h \circ t \mid G, \mathbf{S})) & \text{otherwise} \end{cases}$$

I prove in Section 5.1.3 that if the search depth k is deep enough, $\mathrm{EPRU}^k_{\mathrm{alt}}(s, h_\alpha \mid G, \mathbf{S})$ will always be within ϵ of $\mathrm{EPRU}_{\mathrm{alt}}(s, h_\alpha \mid G, \mathbf{S})$.

4.2.2 EPRU Corresponds to Reproductive Success

This section provides a proof that if a strategy has the highest EPRU for the given environment, it will also have the optimal expected probability of reproducing. This proof applies only to pairwise games, but the same techniques should apply to arbitrary (finite) numbers of strategies.

Assume we have an infinite Cultaptation game instance $(G, \mathbf{S})$, as defined in Section 4.1, made up of agents using strategies s and s' (i.e. $\mathbf{S} = \{s, s'\}$). Recall from Section 4.1 that the score for strategy s is

$$\text{score}(s) = \lim_{r \to \infty} \frac{\sum_{i=0}^{r} p(s, i)}{r}$$

where $p(s, i)$ is the number of agents using strategy s on round i. Our objective for this section will be to show that $\text{EPRU}(s \mid G, \mathbf{S}) > \text{EPRU}(s' \mid G, \mathbf{S})$ if and only if $\text{score}(s) > \text{score}(s')$.

I begin by defining a reset event, which will help illustrate some interesting properties of infinite Cultaptation.

Definition. Let n and n' be the number of agents using s and s', respectively, on the first round of the game instance, and let $N = n + n'$. A *reset event* occurs when all the agents in the environment die on two consecutive rounds, and on the second round they are replaced (via mutation) by n agents using s and n' agents using s'. The probability of a reset event occurring is $\beta = d^N d^N m^N \binom{n}{N} 0.5^n$. □

In other words, after a reset event occurs the conditions are identical to those that were present on the first round; the game instance has essentially started over. Note that β is the same on every round, and it is always greater than 0.

Since the game instance continues for an infinite number of rounds, there will be an infinite number of reset events. Thus, if we were to run other game instances with $\mathbf{S} = \{s, s'\}$, both strategies would have the same score each time. Therefore, we also know that we can define each strategy's score as a function of its *expected population* at each round, rather than its population for a single game instance. This gives us

$$\lim_{r\to\infty} \frac{\sum_{i=0}^{r} p(s,i)}{r} = \lim_{r\to\infty} \frac{\sum_{i=0}^{r} \mathrm{EP}(s,i)}{r} \tag{4.14}$$

where $\mathrm{EP}(s,r)$ is the expected population of agents using strategy s on round r.

I will also define $\mathrm{EAU}(s,r)$ to be the *expected agent utility* of strategy s on round r; that is, $\mathrm{EAU}(s,r)$ is the expected PRU of a randomly-chosen agent using strategy s on round r. $\mathrm{EP}(s,r)$ can now be defined recursively for each strategy using the mechanics of Cultaptation, as follows. Let $\mathrm{EP}(s,0) = n$ and $\mathrm{EP}(s',0) = n'$. Then, for $r \geq 0$

$$\mathrm{EP}(t, r+1) =$$

$$\underbrace{(1-d)\mathrm{EP}(t,r)}_{\text{Survived from previous round}} + \underbrace{Nd(1-m)\frac{\mathrm{EP}(t,r)\mathrm{EAU}(t,r)}{\mathrm{TU}(r)}}_{\text{New agents from selection}} + \underbrace{Nd\frac{m}{2}}_{\text{New agents from mutation}}$$

where $t \in \{s, s'\}$ and $\mathrm{TU}(r) = \mathrm{EP}(s,r)\mathrm{EAU}(s,r) + \mathrm{EP}(s',r)\mathrm{EAU}(s',r)$ is the expected total utility on round r. Recall from Section 4.1 that N is the total number of agents in the environment, d is the probability of death, and m is the probability of mutation.

I now consider the behavior of $\mathrm{EAU}(s,r)$ as r increases.

Lemma 1. *For any strategy s, $\lim_{r\to\infty} \mathrm{EAU}(s,r) = \gamma$ for some finite γ.*

Proof. Let $u(s, r)$ be the expected utility of a single agent using strategy s when r rounds have passed since the first round *or* the last reset event. For all r, we know that $0 \leq u(s, r) \leq V_{\max}/d$, since agents cannot earn negative utility, and no strategy can have better expected performance than a strategy that exploits the best possible action for its entire expected lifespan of $1/d$ rounds. We can rewrite $\mathrm{EAU}(s, r)$ in terms of $u(s, r)$ as follows.

$$\mathrm{EAU}(s, r) = \beta \left(\sum_{i=0}^{r-1} (1-\beta)^i u(s, r) \right) + (1-\beta)^r u(s, r)$$

Taking the limit of this form gives us

$$\begin{aligned} \lim_{r\to\infty} \mathrm{EAU}(s, r) &= \lim_{r\to\infty} \beta \left(\sum_{i=0}^{r-1} (1-\beta)^i u(s, i) \right) + \lim_{r\to\infty} (1-\beta)^r u(s, r) \\ &= \lim_{r\to\infty} \beta \left(\sum_{i=0}^{r-1} (1-\beta)^i u(s, i) \right). \end{aligned}$$

Since $u(s, i)$ is bounded and $\sum_{i=0}^{r-1}(1-\beta)^i$ is a geometric series,

$$\lim_{r\to\infty} \beta \left(\sum_{i=0}^{r-1} (1-\beta)^i u(s, i) \right)$$

converges absolutely by the comparison test. Hence, $\lim_{r\to\infty} \mathrm{EAU}(s, r) = \gamma$ for some finite γ. □

Lemma 2. *For any strategy s_α and set of available strategies* $\mathbf{S}$*,*

$$\lim_{r\to\infty} \mathrm{EAU}(s_\alpha, r) = \mathrm{EPRU}(s_\alpha \mid G, \mathbf{S})$$

Proof. The expected agent utility $\mathrm{EAU}(s_\alpha, r)$ is defined as the expected PRU of an agent using strategy s_α on round r. As r approaches infinity, the probability that a randomly-selected agent will be i rounds old approaches $L(i)/\sum_{j=0}^{\infty} L(j) = dL(i)$.

The probability of reaching a history h_α is defined in Section 4.1.5 as $P(h_\alpha|s_\alpha, \mathbf{S})$, and as r increases the set of histories a randomly-selected agent may have approaches H, the set of all histories. Thus,

$$\lim_{r\to\infty} \mathrm{EAU}(s_\alpha, r) = \sum_{h_\alpha \in H} dL(|h_\alpha|) \times P(h_\alpha|s_\alpha, \mathbf{S}) \times \mathrm{PRU}(h_\alpha),$$

which is the definition of $\mathrm{EPRU}(s_\alpha \mid G, \mathbf{S})$. □

$\mathrm{EP}(s, r)$ and $\mathrm{EP}(s', r)$ are both functions of $\mathrm{EAU}(s, r)$ and $\mathrm{EAU}(s', r)$, which converge to $\mathrm{EPRU}(s \mid G, \mathbf{S})$ and $\mathrm{EPRU}(s' \mid G, \mathbf{S})$ respectively. Therefore, $\mathrm{EP}(s, r)$ and $\mathrm{EP}(s', r)$ must also converge as r approaches infinity. We will let $\mathrm{EP}(s) = \lim_{r\to\infty} \mathrm{EP}(s, r)$ for $s \in \{s, s'\}$. We can find the value of $\mathrm{EP}(s)$ as follows

$$\mathrm{EP}(s) =$$
$$(1-d)\mathrm{EP}(s) + Nd(1-m)\frac{\mathrm{EP}(s)\,\mathrm{EPRU}(s \mid G, \mathbf{S})}{\mathrm{EP}(s)\,\mathrm{EPRU}(s \mid G, \mathbf{S}) + \mathrm{EP}(s')\,\mathrm{EPRU}(s' \mid G, \mathbf{S})} + Nd\frac{m}{2}.$$

After substituting $\mathrm{EP}(s') = N - \mathrm{EP}(s)$ and rearranging terms, we have

$$\begin{aligned} 0 =& (\mathrm{EPRU}(s \mid G, \mathbf{S})\mathrm{EP}(s)^2 - \mathrm{EPRU}(s' \mid G, \mathbf{S})) \\ &+ N\left((1+\frac{m}{2})\,\mathrm{EPRU}(s' \mid G, \mathbf{S}) - (1-\frac{m}{2})\,\mathrm{EPRU}(s \mid G, \mathbf{S})\right)\mathrm{EP}(s) \\ &- N^2\frac{m}{2}\,\mathrm{EPRU}(s' \mid G, \mathbf{S}). \end{aligned}$$

Assume $\mathrm{EPRU}(s \mid G, \mathbf{S}) > 0$ and $\mathrm{EPRU}(s' \mid G, \mathbf{S}) > 0$, which must be true as long as there are no actions that give negative utility. Also, let $x = \mathrm{EPRU}(s \mid G, \mathbf{S})/\,\mathrm{EPRU}(s' \mid G, \mathbf{S})$. Then we can rewrite the above as

$$0 = (x-1)\mathrm{EP}(s)^2 + N\left(1 + \frac{m}{2} - x(1-\frac{m}{2})\right)\mathrm{EP}(s) - N^2\frac{m}{2}. \tag{4.15}$$

This equation, when subject to the constraint $0 \leq \mathrm{EP}(s) \leq N$, allows us to express $\mathrm{EP}(s)$ as a strictly increasing function of x. It also has the property that when $x = 1$, $\mathrm{EP}(s) = \mathrm{EP}(s') = N/2$.

Lemma 3. $\mathrm{EP}(s) > \mathrm{EP}(s')$ *if and only if* $\mathrm{EPRU}(s \mid G, \mathbf{S}) > \mathrm{EPRU}(s' \mid G, \mathbf{S})$.

Proof. Assume $\mathrm{EPRU}(s \mid G, \mathbf{S}) > \mathrm{EPRU}(s' \mid G, \mathbf{S})$. Then $x > 1$ in Equation 4.15, and therefore $\mathrm{EP}(s) > N/2$, so $\mathrm{EP}(s) > \mathrm{EP}(s')$. Assume $\mathrm{EP}(s) > \mathrm{EP}(s')$. Applying this as an extra constraint to Equation 4.15, we see that only values for x greater than one satisfy the equality. Therefore, $\mathrm{EPRU}(s \mid G, \mathbf{S}) > \mathrm{EPRU}(s' \mid G, \mathbf{S})$.

Hence, $\mathrm{EP}(s) > \mathrm{EP}(s')$ if and only if $\mathrm{EPRU}(s \mid G, \mathbf{S}) > \mathrm{EPRU}(s' \mid G, \mathbf{S})$. □

We now know that the strategy with higher EPRU will have the highest expected frequency in the limit, so all that remains is to show that a strategy with higher frequency in the limit will, in fact, have a higher average population over all rounds (and therefore, a higher score in the game).

Lemma 4. *For all s and s',*

$$\lim_{r \to \infty} \frac{\frac{\sum_{i=0}^{r} \mathrm{EP}(s,i)}{r}}{\frac{\sum_{i=0}^{r} \mathrm{EP}(s',i)}{r}} = \frac{\mathrm{EP}(s)}{\mathrm{EP}(s')}.$$

Proof. First, note that because N, d, and m are all strictly positive, $0 < Nd\frac{m}{2} \leq \mathrm{EP}(s, r) < N$ for all strategies s and rounds r. Therefore, $\sum_{i=0}^{r} \mathrm{EP}(s', i)$ is strictly increasing and unbounded as r increases.

Using the fact that $Nd\frac{m}{2} \leq \mathrm{EP}(s, r) < N$ for all s and r, and the definition

$\mathrm{EP}(s) = \lim_{r\to\infty} \mathrm{EP}(s, r)$, we can obtain the following:

$$\lim_{r\to\infty} \frac{\sum_{i=0}^{r+1} \mathrm{EP}(s, i) - \sum_{i=0}^{r} \mathrm{EP}(s, i)}{\sum_{i=0}^{r+1} \mathrm{EP}(s', i) - \sum_{i=0}^{r} \mathrm{EP}(s', i)} = \lim_{r\to\infty} \frac{\mathrm{EP}(s, r+1)}{\mathrm{EP}(s', r+1)}$$
$$= \frac{\lim_{r\to\infty} \mathrm{EP}(s, r+1)}{\lim_{r\to\infty} \mathrm{EP}(s', r+1)} = \frac{\lim_{r\to\infty} \mathrm{EP}(s, r)}{\lim_{r\to\infty} \mathrm{EP}(s', r)} = \frac{\mathrm{EP}(s)}{\mathrm{EP}(s')}.$$

Therefore, by the Stolz-Cesàro theorem,

$$\lim_{r\to\infty} \frac{\sum_{i=0}^{r} \mathrm{EP}(s, i)}{\sum_{i=0}^{r} \mathrm{EP}(s', i)} = \frac{\mathrm{EP}(s)}{\mathrm{EP}(s')},$$

thus

$$\lim_{r\to\infty} \frac{\frac{\sum_{i=0}^{r} \mathrm{EP}(s,i)}{r}}{\frac{\sum_{i=0}^{r} \mathrm{EP}(s',i)}{r}} = \frac{\mathrm{EP}(s)}{\mathrm{EP}(s')}.$$

□

From Equation 4.14 and Lemmas 3 and 4, we immediately get the following:

Theorem 1. *For all s and s',*

$$\lim_{r\to\infty} \frac{\frac{\sum_{i=0}^{r} n(s,i)}{r}}{\frac{\sum_{i=0}^{r} n(s',i)}{r}} > 1$$

if and only if $\mathrm{EPRU}(s \mid G, \mathbf{S}) > \mathrm{EPRU}(s' \mid G, \mathbf{S})$.

Therefore, the strategy with higher EPRU will have the higher score in a game of infinite Cultaptation.

Irrelevance of the Initial Strategy Profile

From the fact that EPRU is independent of the initial strategy profile $\mathbf{s}$, we also get the following corollary which will help us understand some of the experimental results (see Section 5.3).

Corollary 1. *The initial strategy profile* $\mathbf{s}$ *of an infinite Cultaptation game instance (defined in Section 4.1) does not affect the score of any strategy in* $\mathbf{S}$.

If this seems counterintuitive, consider the following. At the beginning of this section I defined a reset event, in which every agent dies on two consecutive rounds, and all are replaced via mutation so that the population is identical to the initial strategy profile. For each reset event, there will be many similar events in which every agent dies on two consecutive rounds and is replaced via mutation, but in some arrangement different from the initial strategy profile. The probability of this happening is $d^{2N}m^N$, which is greater than 0. In an infinite-length game, such an event will eventually occur with probability 1. After it occurs, the initial strategy profile clearly has no bearing on how the rest of the game plays out, yet there are still an infinite number of rounds left in the game. Since each strategy's score is its average population over the entire game (see Section 4.1), the impact of the initial strategy profile on each strategy's total score is vanishingly small in an infinite-length game.

Application of EPRU **to other Evolutionary Games**

Many of the equations used in calculating EPRU involve concepts particular to Cultaptation, such as innovation, observation, and changing action values. However, the general technique used is to calculate the expected reproductive fitness of an agent on round j, multiply this quantity by the expected proportion of agents that are j rounds old, and sum these quantities to get the expected fitness of an entire population. This should be a useful metric in any evolutionary game in which agents live for more than one generation and reproduce according to the replicator equation, even if the game uses some measure other than per-round utility to determine reproductive fitness. The proofs in this section rely primarily on the

symmetry between 1) the probability that an agent will be alive after k rounds and 2) the expected proportion of a population of agents that are k rounds old on any given round. Thus, any evolutionary game that allows agents to live more than one generation and in which agents die with the same probability on every round should be able to use a metric very similar to EPRU to compare strategies.

Chapter 5

Strategy Generation Algorithms

This chapter describes the algorithms used to generate near-best response strategies for Cultaptation, and presents experimental studies in which near-best response strategies are tested against a known good strategy from the Cultaptation tournament.

5.1 Finding an ϵ-Best Response Strategy

In this section I explain what it means for a strategy to be a best response or near-best response in infinite Cultaptation, and I provide an algorithm for calculating a near-best response to $\mathbf{S}_{-\alpha}$, the available strategies other than our own.

5.1.1 Problem Specification

Now that we have derived EPRU and proved that a strategy's EPRU is directly proportional to its score in an infinite Cultaptation game, we can determine how each strategy in a given set of available strategies $\mathbf{S}$ will perform by evaluating the EPRU of each strategy. Therefore, we can define a best-response strategy in terms of EPRU, as follows.

Recall that in an infinite Cultaptation game (as defined in Section 4.1) there are ℓ players, each of whom selects a strategy to put into the set of available strategies

$\mathbf{S}$. Let $\mathbf{S}_{-\alpha}$ be the set of available strategies other than our own, i.e. $\mathbf{S}_{-\alpha} = \{s_1, ..., s_{\alpha-1}, s_{\alpha+1}, ..., s_\ell\}$. Strategy s_{opt} is a *best response* to $\mathbf{S}_{-\alpha}$ if and only if for any other strategy s',

$$\text{EPRU}(s_{\text{opt}} \mid G, \mathbf{S}_{-\alpha} \cup s_{\text{opt}}) \geq \text{EPRU}(s' \mid G, \mathbf{S}_{-\alpha} \cup s').$$

Computing s_{opt} is not possible due to its prohibitively large size. However, we can compute an ϵ-*best-response strategy*, i.e., a strategy s such that $\text{EPRU}(s \mid G, \mathbf{S}_{-\alpha} \cup s)$ is arbitrarily close to $\text{EPRU}(s_{\text{opt}} \mid G, \mathbf{S}_{-\alpha} \cup s_{\text{opt}})$. This problem can be stated formally as follows: Given game parameters G, error bound $\epsilon > 0$, and the set $\mathbf{S}$ of available strategies other than our own, find a strategy s_α such that $\text{EPRU}(s_\alpha \mid G, \mathbf{S}_{-\alpha} \cup s_\alpha)$ is within ϵ of $\text{EPRU}(s_{\text{opt}} \mid G, \mathbf{S}_{-\alpha} \cup s_{\text{opt}})$.

5.1.2 Bounding EPRU

In games where $0 < d < 1$, an agent could potentially live for any finite number of rounds. However, since the agent's probability of being alive on round r decreases exponentially with r, the expected utility contributed by an agent's actions in later rounds is exponentially lower than the expected utility contributed by earlier rounds. I will use this fact in deriving a bound on $\text{EPRU}_{\text{alt}}(s, h_\alpha \mid G, \mathbf{S})$ for a given strategy and a history h_α of length l.

Recall from Equations 4.10 and 4.11 that:

$$\text{EV}_{exp}(r, v) = v \sum_{i=r}^{\infty} \frac{1}{i}(1-d)^{i-1} = v \left(\frac{\ln d}{d-1} - \sum_{i=1}^{r-1} \frac{1}{i}(1-d)^{i-1} \right) \tag{5.1}$$

where $\text{EV}_{exp}(r, v)$ is the expected contribution to EPRU made by exploiting an action with value v on round r.

Since we know how much any given exploit contributes to EPRU($s_\alpha \mid G, \mathbf{S}$) for a given strategy s_α, we can calculate $G(l, v)$, the amount that exploiting the same action on *all* rounds after l contributes to EPRU($s_\alpha \mid G, \mathbf{S}$), as follows:

$$G(l,v) = \sum_{j=l+1}^{\infty} v \sum_{n=j}^{\infty} \frac{1}{n}(1-d)^{n-1} = v \sum_{j=l+1}^{\infty} \sum_{n=j}^{\infty} \frac{1}{n}(1-d)^{n-1}$$

Expanding the summations yields:

$$\begin{aligned} G(l,v) &= v\left(\frac{1}{l+1}(1-d)^l + \frac{2}{l+2}(1-d)^{l+1} + \cdots\right) \\ &= v \sum_{n=l+1}^{\infty} \frac{n-l}{n}(1-d)^{n-1} \\ &= v\left(\sum_{n=l+1}^{\infty}(1-d)^{n-1} - \sum_{n=l+1}^{\infty}\frac{l}{n}(1-d)^{n-1}\right) \\ &= v\left(\frac{(1-d)^l}{d} - \sum_{n=l+1}^{\infty}\frac{l}{n}(1-d)^{n-1}\right) \end{aligned}$$

Next, we pull l out of the summation and use (5.1) to obtain:

$$G(l,v) = v\left(\frac{(1-d)^l}{d} - \underbrace{l\left(\frac{\ln d}{d-1} - \sum_{n=1}^{l}\frac{1}{n}(1-d)^{n-1}\right)}_{a}\right) \tag{5.2}$$

Note that for $0 < d < 1$, $G(l, v)$ is finite. $G(l, v)$ provides a closed form formula for the eventual contribution of exploiting an action with value v at every round after the l^{th} round. Since the set V of possible action values is finite (see Section 4.1), let $v_{max} = \max(V)$ be the largest of these values. Then $G(l, v_{max})$ is an upper bound on the expected per-round utility achieved after round l (clearly no strategy can do better than making an action with maximal value every action after action l). I use this fact to bound the depth limited expected per-round utility computation.

Theorem 2. *Let v_{max} be the highest possible action utility for game parameters G, and let $\mathbf{S}_{-\alpha}$ be the set of available strategies other than our own. Then for all l and all strategies s_α,*

$$\mathrm{EPRU}_{\mathrm{alt}}(s_\alpha, \langle\rangle \mid G, \mathbf{S}_{-\alpha} \cup s_\alpha) - \mathrm{EPRU}^{l}_{\mathrm{alt}}(s_\alpha, \langle\rangle \mid G, \mathbf{S}_{-\alpha} \cup s_\alpha) \leq G(l, v_{max}).$$

Proof. Since it is not possible for any strategy to gain more utility than v_{max} on any round, this follows from the discussion above. □

Theorem 2 states that $G(l, v_{max})$ is the highest possible contribution to the total expected per-round utility (i.e., $\mathrm{EPRU}_{\mathrm{alt}}(s_\alpha, \langle\rangle \mid G, \mathbf{S})$) made by any strategy s_α after round l. Thus, if we are given an $\epsilon > 0$ and we can find a value of k such that $G(k, v_{max}) \geq \epsilon$, then we know that no strategy can earn more than ϵ expected utility after round k. The next section will show how to find such a k.

5.1.3 Determining How Far to Search

In this section I show how to find a search depth k such that, for any given $\epsilon > 0$, no strategy can earn more than ϵ utility after round k. We first note a bound on $G(l, v)$:

Lemma 5. $G(l, v) \leq v(1-d)^l/d$.

Proof. The lemma follows from noting that part (a) of Equation 5.2 is greater than or equal to zero, since $\frac{\ln d}{d-1} = \sum_{n=1}^{\infty} \frac{1}{n}(1-d)^{n-1}$ and $l < \infty$. Thus $G(l, v) = v(\frac{(1-d)^l}{d} - w) \leq v\frac{(1-d)^l}{d}$, since w is always non-negative. □

Now if we can find a k such that

$$\epsilon = v_{max}(1-d)^k/d,$$

then we can be certain that $\epsilon \geq G(k, v)$. Solving for k in the above equation yields

$$k = log_{(1-d)}\left(\frac{d\epsilon}{v_{max}}\right), \tag{5.3}$$

which has a solution for $0 < d < 1$ and $v_{max} > 0$, both of which will always be true in Cultaptation. This gives us the following theorem.

Theorem 3. *Given $\epsilon > 0$, set of available strategies $\mathbf{S}_{-\alpha}$ other than our own, and game parameters G with maximal utility v_{max}, let $k = log_{(1-d)}\left(\frac{d\epsilon}{v_{max}}\right)$. If s_α has the maximal value of* $\mathrm{EPRU}^k_{\mathrm{alt}}(s_\alpha, \emptyset \mid G, \mathbf{S}_{-\alpha} \cup s_\alpha)$*, then s_α is an* ϵ-best response *to* $\mathbf{S}_{-\alpha}$.

Proof. Let s_{opt} be the strategy with the maximal value of $\mathrm{EPRU}(s_{\mathrm{opt}} \mid G, \mathbf{S}_{-\alpha} \cup s_{\mathrm{opt}})$. By Theorem 2, we know that s_{opt} cannot earn more than ϵ expected utility on rounds after k. Since s_α earns the maximum EPRU possible in the first k rounds, it follows that $|\,\mathrm{EPRU}(s_{\mathrm{opt}} \mid G, \mathbf{S}_{-\alpha} \cup s_{\mathrm{opt}}) - \mathrm{EPRU}(s_\alpha \mid G, \mathbf{S}_{-\alpha} \cup s_\alpha)| \leq \epsilon$. Therefore, s_α is an ϵ-best response. □

5.1.4 Algorithm

I will now present my algorithm for computing the strategy s with the maximal value of $\mathrm{EPRU}^k_{\mathrm{alt}}(s, \emptyset \mid G, \mathbf{S}_{-\alpha} \cup s)$, and show how it can be used to compute an ϵ-best response.

Algorithm 1 returns a 2-tuple with a partially specified strategy s and a scalar U. Strategy s maximizes $\mathrm{EPRU}^k_{\mathrm{alt}}(s, h_\alpha \mid G, \mathbf{S}_{-\alpha} \cup s)$, and U is the value of this

Algorithm 1 Produce strategy s that maximizes $\mathrm{EPRU}^k_{\mathrm{alt}}(s, h_\alpha \mid G, \mathbf{S}_{-\alpha} \cup s)$, given initial history h_α, set of possible utility values V, and $\mathbf{S}_{-\alpha}$, the set of available strategies other than our own.

Strat(h_α,k,V, $\mathbf{S}_{-\alpha}$)

if $k = 0$ **then**
  return 0
end if
Let $U_{\max} = 0$
Let $s_{\max} = null$
for each action $a \in \{\mathsf{Inv}, \mathsf{Obs}, \mathsf{X}_1, \ldots, \mathsf{X}_M\}$ **do**
  Let $U_{\mathrm{temp}} = 0$
  Let $s_{\mathrm{temp}} = \langle h_\alpha, a \rangle$
  for each action $m \in \{1, \ldots, M\}$ **do**
    for each value $v \in V$ **do**
      Let $t = (a, (m, v))$
      Let $p = P(h_\alpha \circ t | h_\alpha, a, \mathbf{S}_{-\alpha})$
      if $p > 0$ **then**
        Let $\{S', U'\} = \mathrm{Strat}(h_\alpha \circ t, k-1, V, \mathbf{S}_{-\alpha})$
        $s_{\mathrm{temp}} = s_{\mathrm{temp}} \cup S'$
        $U_{\mathrm{temp}} = U_{\mathrm{temp}} + p\,(\mathrm{EV}_{exp}(|h_\alpha \circ t|, U(t)) + U')$
      end if
    end for
  end for
  if $U_{\mathrm{temp}} > U_{\max}$ **then**
    $U_{\max} = U_{\mathrm{temp}}$
    $s_{\max} = s_{\mathrm{temp}}$
  end if
end for
return $\{s_{\max}, U_{\max}\}$

expression.

The algorithm performs a depth-first search through the space of strategies that start from the input history h_α, stopping once it reaches a specified depth k. Figure 5.1 provides an example of the kind of tree searched by this algorithm. For each possible action $a \in \{\mathsf{Inv}, \mathsf{Obs}, \mathsf{X}_1, \ldots, \mathsf{X}_M\}$ at h_α, it computes the expected per-round utility gained from performing a, and the utility of the best strategy for each possible history h'_α that could result from choosing a. It combines these

quantities to get the total expected utility for a, and selects the action with the best total expected utility, a_{max}. It returns the strategy created by combining the policy $\langle h_\alpha, a_{max} \rangle$ with the strategies for each possible h'_α, and the utility for this strategy.

Seen another way, $\text{Strat}(h_\alpha, k, V, \mathbf{S}_{-\alpha})$ computes $\text{EPRU}^k_{\text{alt}}(s, h_\alpha \mid G, \mathbf{S}_{-\alpha} \cup s)$ for all possible strategies s, returning the strategy maximizing $\text{EPRU}^k_{\text{alt}}$ as well as the maximal value of $\text{EPRU}^k_{\text{alt}}$.

Proposition 1. *Strat*$(h_\alpha, k, V, \mathbf{S}_{-\alpha})$ *returns* (s, U) *such that*

$$\text{EPRU}^k_{\text{alt}}(s, h_\alpha \mid G, \mathbf{S}_{-\alpha} \cup s) = U = \text{argmax}_{s'}(\text{EPRU}^k_{\text{alt}}(s', h_\alpha \mid G, \mathbf{S}_{-\alpha} \cup s')).$$

A proof of this proposition is presented in [19].

We now have an algorithm capable of computing the strategy with maximal expected utility over the first k rounds. Hence, in order to find an ϵ-best response strategy we need only find the search depth k such that no strategy can earn more than ϵ expected utility after round k, and then call the algorithm with that value of k.

Theorem 4. *Given* $\epsilon > 0$*, available strategies other than our own* $\mathbf{S}_{-\alpha}$*, and a set of values* V *with maximum value* v_{max}*, let* $k = log_{(1-d)}\left(\frac{d\epsilon}{v_{max}}\right)$*. Then Strat*$(\emptyset, k, V, \mathbf{S}_{-\alpha})$ *returns* (s, U) *such that* s *is an* ϵ*-best response to* $\mathbf{S}_{-\alpha}$*.*

Proof. This follows from Theorem 3 and Proposition 1. □

We also have the following.

Corollary 2. *Given available strategies other than our own* $\mathbf{S}_{-\alpha}$ *and a set of values* V*, let* s_k *be the strategy returned by Strat*$(\emptyset, k, V, \mathbf{S}_{-\alpha})$*. Then* $\lim_{k\to\infty} s_k$ *is a best*

response to $\mathbf{S}_{-\alpha}$.

Proof. Let s_{opt} be a best response to $\mathbf{S}_{-\alpha}$. By Lemma 5 and Theorem 3,

$$\text{EPRU}(s_{\text{opt}} \mid G, \mathbf{S}_{-\alpha} \cup s_{\text{opt}}) - \text{EPRU}(s_k \mid G, \mathbf{S}_{-\alpha} \cup s) \leq v_{max}(1-d)^k/d.$$

Since $\lim_{k\to\infty} v_{max}(1-d)^k/d = 0$, it follows that $\lim_{k\to\infty}(\text{EPRU}(s_{\text{opt}} \mid G, \mathbf{S}_{-\alpha} \cup s_{\text{opt}}) - \text{EPRU}(s_k \mid G, \mathbf{S}_{-\alpha} \cup s_k)) = 0$. Therefore, $\lim_{k\to\infty} s_k$ is a best response to $\mathbf{S}_{-\alpha}$. □

5.1.5 Implementation

In this section I discuss modifications that improve the running time of Algorithm 1 without any loss in accuracy. Section 5.1.5 discusses techniques for state aggregation that cut the branching factor of the algorithm in half. Section 5.1.5 discusses the representation of π_{Obs}, and Section 5.1.5 discusses caching and pruning.

State Aggregation

If the pseudocode for Algorithm 1 were implemented verbatim, it would search through each history that can be reached from the starting state. However, there is a significant amount of extraneous information in each history that is not needed for any of the algorithm's calculations. For example, the histories $h_\alpha = \langle(\mathsf{Inv}, (1, 10))\rangle$ and $h'_\alpha = \langle(\mathsf{Inv}, (2, 10))\rangle$ both describe a situation where α innovates once and obtains an action with value 10. The only difference between these histories is the identifier assigned to the action, which does not impact any of the calculations—yet the pseudocode must still search through each of these histories separately. We can

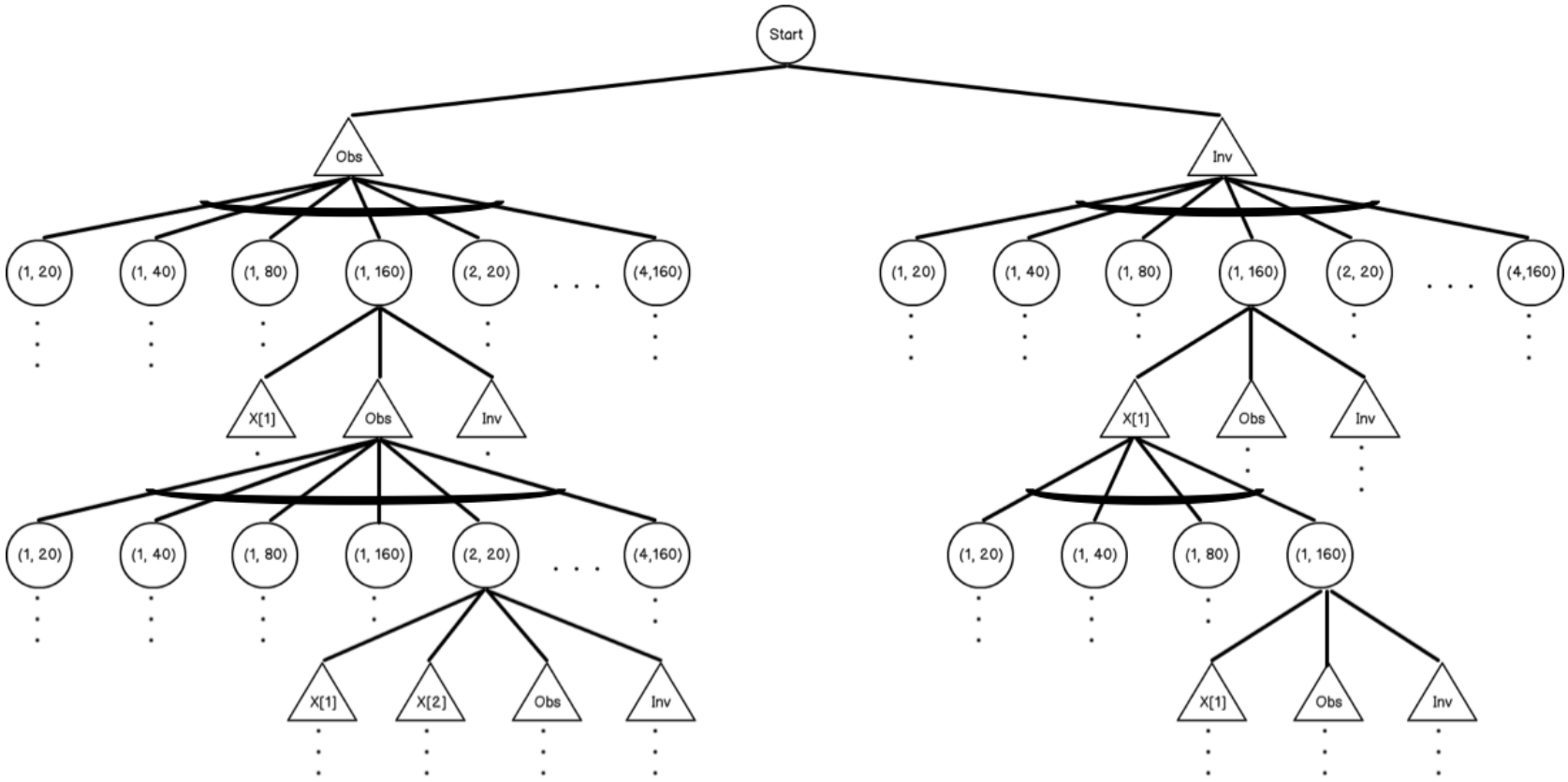


Figure 5.1: An illustration of a portion of the search performed by Algorithm 1 on a very small version of Cultaptation, with four actions and a set of possible action values equal to $\{20, 40, 80, 160\}$. Circle nodes indicate possible observations made by the agent, and triangle nodes indicate possible choices made by the agent (which have a random outcome, indicated by the line through the branches leading to its children). Ellipses indicate branches or subtrees omitted from the figure. The left side of the illustration shows how more actions can become available to the agent as the game progresses, while the right side shows how potentially-changing values of actions must be modeled in the tree. Several techniques outlined in Section 5.1.5 can reduce the size of the tree that needs to be searched in the implementation of this algorithm.

eliminate this redundancy by using *repertoires*, rather than histories, as the states for the algorithm to search through. A repertoire is a record of what the agent knows about each of the actions it has learned, rather than a record of everything that has happened to it.

Making this simple change allows Algorithm 1 to calculate the value of an observation action by combining information it learns when exploring innovate and exploit actions, rather than recursing again. This cuts the branching factor of the search in half. The analysis and details involved in this change, as well as the proof that the version of the algorithm using repertoires returns the same result as the previous version, are included in Appendix B.

Running time analysis.

When Algorithm 1 considers a history h_α, it makes one recursive call for each possible action-percept pair $(a, (m, v))$ that can be executed at h_α. There are $2M$ such pairs for each history; if the agent knows how to exploit j actions, then it can innovate any of the $M - j$ actions it does not know, and it can observe any of the M actions. Each of these actions can also have any of v values. Hence, the number of recursive calls made by the algorithm each action is at most $2Mv$. Since the algorithm recurses to depth k, the running time for Algorithm 1 is $O((2Mv)^k)$. With the state aggregation technique described above, we do not need to perform additional recursions for observation actions. Hence, the number of recursive calls made each action is at most Mv, and the total running time is $O((Mv)^k)$, which improves upon the original running time by a factor of 2^k.

Representing $\pi_{\mathbf{Obs}}$

For the formal proofs, I treated π_{Obs} as a black box that, when given the agent's history and round number, could tell us the exact probabilities of observing each action on the current round. However, since there are an exponential number of possible histories, storing π_{Obs} in this form would require an exponential amount of space, which would severely limit the size of games for which we could compute strategies. Algorithm 2 (introduced in Section 5.2) would also need to run a prohibitively large number of simulations to get enough samples to generate a new π_{Obs} of this type.

Therefore, as an approximation, my implementation assumes that π_{Obs} has a similar structure to π_{Inv}, and remains constant throughout the agent's lifetime. That is, the π_{Obs} used in the experiments returns the probability of an action valued v being observed. While this leads to some loss in accuracy, it is very easy to store and compute. Further, we will see in the experimental results (particularly those dealing with iterative computation in Section 5.3.2) that this form of π_{Obs} is still able to produce good strategies.

Caching and Pruning

Since the implementation uses repertoires rather than histories to represent the agent's set of known actions, and since it is possible for two histories to produce the same repertoire, the algorithm will sometimes encounter repertoires that it has already evaluated. So that the algorithm will not have to waste time evaluating them again, the implementation includes a cache which stores the EPRU of every repertoire it has evaluated. When the algorithm encounters a repertoire whose expected utility is needed, the implementation first checks the cache to see if the

EPRU of the repertoire has been previously computed, and uses the computed value if it exists. Caching is widely used in tree-search procedures, and is analogous to the transposition tables in chess-playing algorithms [37].

I also use another well-known method for avoiding unnecessary evaluation of states, namely branch-and-bound pruning [38, 39], which can be summarised as follows. Before we compute the expected per-round utility of a given action, we check to see if an upper bound on the EPRU of that action would be sufficient to make the given action's utility higher than the best previously computed action. In many situations, the maximal utility that can be achieved for a given action will in fact be less than the utility we know we can achieve via some other action, and therefore we can skip the evaluation of that action (i.e., we can "prune" it from the search tree).

There are no theoretical guarantees on runtime reduction using these techniques, but we will see in Section 5.3.1 that the combination of pruning and caching allows the algorithm to avoid evaluating significant portions of the state space in the environments I tested.

5.2 Cultaptation Strategy Learning Algorithm

Until now I have assumed that Algorithm 1 has access to π_{Obs}, the distribution of observable actions, when it performs its calculations. While the algorithm finds the near-best-response strategy given a particular π_{Obs}, agents playing the real Cultaptation game are not given access to π_{Obs} beforehand, and even estimating

what π_{Obs} looks like can be very difficult while playing the game due to the limited amount of information each agent receives in its lifetime. It is also unclear how exactly an agent's own actions will affect π_{Obs}: by exploiting a particular action, the agent is making that action observable to others who might then exploit it in greater proportion than in the π_{Obs} used to compute the agent's strategy.

To address these difficulties, I developed the Cultaptation Strategy Learning Algorithm (CSLA), which uses a method for creating a strategy and a distribution π_{Obs} *simultaneously* so that (i) π_{Obs} is the distribution created when all agents in a Cultaptation game play the computed strategy and (ii) the computed strategy is a near-best response for π_{Obs} (and other parameters).

This algorithm copes with the lack of information about π_{Obs}, and generates an approximation of a strategy that is a best response to itself. At a high level, the algorithm can be thought of as generating a series of strategies, each an ϵ-best response to the one before it, and stopping when two successive strategies are extremely similar. A more detailed description of this process follows.

The algorithm begins by assuming $\pi_{\mathsf{Obs}} = \pi_{\mathsf{Inv}}$. The algorithm then proceeds iteratively; at each iteration it generates s, the ϵ-best response strategy to the current π_{Obs}, then simulates a series of Cultaptation games in which s plays itself, and extracts a new π_{Obs} from the actions exploited in these games.

At the end of each iteration, the algorithm compares s to s_{old}, the strategy produced by the previous iteration, using the stratDiff function. $\text{stratDiff}(s, s_{\text{old}})$ computes the probability that an agent using s would perform at least one different action before dying than the same agent using s_{old}. For instance, $\text{stratDiff}(s, s_{\text{old}}) =$

Algorithm 2 Produce an approximation of a strategy that is an ϵ-best response to itself.

CSLA($\pi_{\mathsf{Inv}}, \tau, k$)

1: Let $\pi_{\mathsf{Obs}} = \pi_{\mathsf{Inv}}$.
2: $s = \emptyset$.
3: **repeat**
4: Let $s_{\text{old}} = s$.
5: Let $V = \{\pi_{\mathsf{Inv}}, \pi_{\mathsf{Obs}}\}$.
6: $s = \text{Strat}(\emptyset, k, V, \mathbf{S})$
7: Simulate a series of Cultaptation games in which s plays itself, and action utilities are initially drawn from π_{Inv}, recording all actions exploited in the last quarter of this game.
8: Use records of exploited actions to generate a new distribution π_{Obs} (i.e. $\pi_{\mathsf{Obs}}(v)$ = fraction of the time v was exploited in the records).
9: **until** $\text{stratDiff}(s, s_{\text{old}}) < \tau$
10: **return** s.

1.0 means that the two strategies will always perform at least one different action (i.e. the actions they choose on the first round are different), while $\text{stratDiff}(s, s_{\text{old}}) = 0.0$ means that s is identical to s_{old}.

When $\text{stratDiff}(s, s_{\text{old}})$ is found to be below some threshold τ, CSLA terminates and returns s, the strategy computed by the last iteration. The formal algorithm is presented as Algorithm 2.

Properties of the strategy. CSLA as presented here is a "best-effort" algorithm in the following sense: If CSLA converges to a strategy s, we know that it is an approximation of a symmetric Nash equilibrium strategy, but we do not know (i) whether or not CSLA will converge for a given environment, or (ii) how close to the true Nash equilibrium s is. The improved version of CSLA proposed in Chapter 5 will address these issues.

In my experimental studies (see Section 5.3), the strategies produced by CSLA in any given game were all virtually identical, even when a random distribution

(rather than π_{Inv}) was used to initialize π_{Obs}. This strongly suggests (though it does not prove) that the strategy profile consisting of copies of s_{self} is a symmetric near-Nash equilibrium.

Furthermore, there is reason to believe that s is evolutionarily stable. Consider an environment in which all agents use the strategy s, and suppose a small number (say, one or two) other strategies are introduced as invaders. Because s was an near-best response to the environment that existed before the opponent's agents are introduced, and because the introduction of one or two invaders will change this environment only slightly, agents using s will still be using a strategy that is close to the best response for the current environment, and they will also have some payoff they have accumulated on previous rounds when their strategy was still an near-best response. Thus, the invaders should have a difficult time establishing a foothold in the population, hence should die out with high probability. This suggests (but does not prove) that s is evolutionarily stable.[1]

5.2.1 Implementation Details

We have created a Java implementation of CSLA. Here I briefly discuss two issues dealt with during implementation.

Representing π_{Obs}

My implementation of CSLA uses the same representation of π_{Obs} as my implementation of Algorithm 1 does. In other words, it assumes π_{Obs} has the same

[1] Among other things, a formal proof would require a way to calculate the payoffs for s and any invading strategy. Accomplishing this is likely to be complicated, but I hope to do it in my future research.

form as π_{Inv}, and remains constant throughout the game. Ideally we would be able to condition π_{Obs} on the agent's history, but in practice this would require too much space (since there are an exponential number of possible histories), and we would need to run too many simulations in step 7 to get an accurate distribution for each history.

Training

In the Machine Learning literature, the process of improving an agent's performance on a given task is often referred to as "training." In Algorithm 2, strategy s is trained by playing against itself in a series of simulated games in step 7. However, in the implementation of CSLA the agents involved in the games in step 7 are a parameter to the algorithm. This means that CSLA can also produce a strategy that is trained by playing in an environment consisting of itself and one or more given strategies. The intuition behind this approach is that a strategy trained by playing against itself and strategy s' may perform better when playing against s' than a strategy trained against itself alone. I test this hypothesis experimentally, in Section 5.3.2.

5.3 Experimental Results

In this section I present my experimental results.

Section 5.3.1 examines the performance of the implementation of the ϵ-best response algorithm. I find that the optimizations allow the algorithm to find strategies within 1% of the best response 1,000 times faster than the unoptimized algorithm.

Section 5.3.1 examines the strategies found by the ϵ-best response algorithm when presented with different environments and strategy profiles, and the results give us an idea of what kinds of circumstances are necessary for the near-best-response strategy to prefer innovation over observation.

Section 5.3.2 presents a series of experiments comparing two strategies generated with my Cultaptation Strategy Learning Algorithm to a known good strategy used in the international Cultaptation tournament. I find that the strategies generated with CSLA are able to beat the known good strategy, even when the environment is different than the one CSLA used to learn the strategies (Sections 5.3.2 and 5.3.2). Finally, I perform an in-depth qualitative analysis of all three strategies and highlight the differences in behavior that give my learned strategies an advantage (Section 5.3.2).

5.3.1 Experiments with ϵ-Best Response Algorithm

In this section I present the experiments involving an implementation of Algorithm 1, which generates ϵ-best-response strategies for a given set of game parameters and strategy profile.

Implementation Performance

My first set of experiments was designed to study the accuracy and running time of my implementation, and the effectiveness of the methods developed to improve its performance. I first examined the effect of ϵ on running time and on the expected per-round utility computed by Algorithm 1. I ran the experiments in

several different environments; first I examined the uniform1 environment. In this environment, π_{Inv} is a uniform distribution over $\{33.33, 66.67, 100, 133.33, 166.67\}$, and $\mathbf{S}$ contains the innovate-once (I1) strategy from Example 1, so π_{Inv} is identical to π_{Obs}. The probability of change in uniform1 is 1%, and the probability of death is 40%.

I also introduced several variations on the uniform1 environment to study the effect of different probabilities of change. They are uniform10, uniform20, uniform30, and uniform40, which have the respective probabilities of change of 10%, 20%, 30%, and 40%.

In Table 5.1, we see the EPRU computed for various values of epsilon in these environments. As a point of reference, strategy I1 can be analytically shown to achieve an EPRU of about 38.56. We can see that an upper bound on achievable EPRU in the uniform1 environment is 40.5, since the EPRU of an ϵ-best response to $\mathbf{S}$ is 40.1 when ϵ is 0.4. Also, note that the algorithm finds lower EPRUs as the probability of change increases. This is as expected: in a rapidly changing environment, one cannot expect an agent to do as well as in a static environment where good actions remain good and bad actions remain bad. The ϵ-best-response strategies computed generally innovate as the first action, then exploit that value if it is not the lowest value available (in this case 33.33). Otherwise, the strategies tend to innovate again in an attempt to find an action with a value bigger than 33.33. This is how they manage to achieve a higher EPRU than the innovate-once strategy.

As part of the experiment in the uniform1 environment, I kept track of the

Table 5.1: Expected per-round utility of the ϵ-best response strategy computed by Algorithm 7, for eight different values of ϵ in various environments.

	$\epsilon = 0.4$	$\epsilon = 0.8$	$\epsilon = 1.2$	$\epsilon = 1.6$	$\epsilon = 2.0$	$\epsilon = 2.4$	$\epsilon = 2.8$	$\epsilon = 3.2$
uniform1	40.1	40.0	39.6	39.6	39.6	38.9	38.9	38.9
uniform10	39.3	39.1	38.8	38.8	38.8	38.1	38.1	38.1
uniform20	38.7	38.5	38.2	38.2	38.2	37.7	37.7	37.7
uniform30	38.7	38.5	38.2	38.2	38.2	37.7	37.7	37.7
uniform40	38.7	38.5	38.2	38.2	38.2	37.7	37.7	37.7

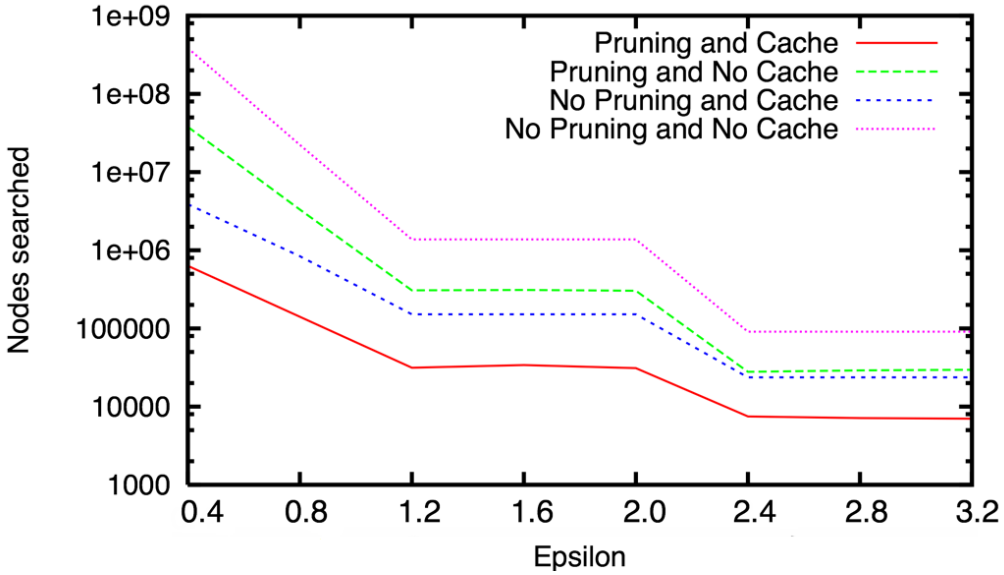


Figure 5.2: Number of nodes searched in the uniform1 environment, with different combinations of caching and pruning.

number of nodes searched by four variations of the algorithm. In the first variation, I ran Algorithm 1 without optimizations. I also examined the algorithm's performance with the pruning and caching optimizations described in Section 5.1.5.

In Figure 5.2 we see that employing both caching and pruning allows the algorithm to compute strategies within 1% of the best response about 1,000 times faster. The search time required for 80,000-node search is around 15 seconds on a 3.4GHz Xeon processor.

Effects of Varying π_{Inv} and π_{Obs} on the ϵ-Best-Response

The objective of this experiment was to study how near-best-response strategies (as computed by Algorithm 1) change as I varied the mean and standard deviation of π_{Inv} and π_{Obs} (which I will call μ_{Inv}, σ_{Inv}, μ_{Obs} and σ_{Obs}, respectively). If we assume that the other agents in the game are rational and not trying to deceive us by intentionally exploiting low-utility actions, then one should expect that $\mu_{\mathsf{Obs}} \geq \mu_{\mathsf{Inv}}$. It may seem natural, then, to conclude that an agent should choose to observe rather than innovate whenever possible, since the average action returned by observing will have higher utility than one returned by innovating. However, previous work has suggested that the standard deviation of these distributions may also play a role in determining which is better [23]. Also, as discussed in Section 2.2.1, it is possible to imagine pathological scenarios where a population that relies too heavily on observation can become stuck exploiting a low-value action. I designed this experiment to test the hypothesis that, even if I let $\mu_{\mathsf{Obs}} > \mu_{\mathsf{Inv}}$, the standard deviations of these distributions can still be varied such that the ϵ-best-response strategy computed by $\mathrm{EPRU}^k_{\mathrm{alt}}$ will choose to innovate rather than observe. My methods and results are presented below.

I used the repertoire-based algorithm $\mathrm{Strat}(R, r, k, V)$ (Algorithm 1) to compute ϵ-best-response strategies for Cultaptation games with several different parameter settings, then analyzed the strategies to determine how often they would observe or innovate. In this experiment, the agents died with 40% probability on each round ($d = 0.4$) and there were 5 potential exploitation actions. These games are smaller than the Cultaptation game used in the tournament, to ensure they can

Table 5.2: The portion of innovation actions (calculated as $n_{\mathsf{Inv}}/(n_{\mathsf{Inv}} + n_{\mathsf{Obs}})$) in the ϵ-best-response strategy when the standard deviations of π_{Inv} and π_{Obs} are as specified. In all cases, $\mu_{\mathsf{Inv}} = 100$ and $\mu_{\mathsf{Obs}} = 110$.

	$\sigma_{\mathsf{Obs}} = 10$	$\sigma_{\mathsf{Obs}} = 300$
$\sigma_{\mathsf{Inv}} = 10$	2.80×10^{-9}	3.40×10^{-10}
$\sigma_{\mathsf{Inv}} = 300$	**0.995**	0.215

be solved in a reasonable amount of time. In these games, I used distributions π_{Obs} and π_{Inv} with means of 110 and 100 respectively.

Table 5.2 shows the results for four combinations of parameter settings: $\sigma_{\mathsf{Inv}} \in \{10, 300\}$ and $\sigma_{\mathsf{Obs}} \in \{10, 300\}$. When $\sigma_{\mathsf{Inv}} = 10$, the near-best-response strategy will observe almost exclusively (innovating only in rare cases where observation returns several low-quality moves in a row). However, in the environment where $\sigma_{\mathsf{Inv}} = 300$ the near-best-response strategy includes significantly more Innvoates; when $\sigma_{\mathsf{Obs}} = 10$ it will innovate 99.5% of the time, and even when $\sigma_{\mathsf{Obs}} = 300$ it still innovates 21.5% of the time.

This experiment lets us conclude that the means of π_{Inv} and π_{Obs} are *not sufficient to determine if innovation or exploitation is better.* In particular, if the standard deviation of innovated values is high, then innovation becomes more valuable because multiple innovations tend to result in a higher valued action than multiple observations.

An interesting strategy emerges when π_{Inv} and π_{Obs} both have high standard deviations. Even though the mean value of innovated actions is lower than the mean value of observed actions, the ϵ-best-response strategy in these cases innovates initially, then, if the value innovated is high, exploits that value. If the innovated

value is not high, an observation action is performed to ensure the agent has a reasonably-valued action available to exploit until it dies.

5.3.2 Experiments with the Cultaptation Strategy Learning Algorithm

The objective of my second experiment was to examine the performance of strategies produced by the Cultaptation Strategy Learning Algorithm (Algorithm 2 in Section 5.2), and the importance of the environment (see Section 5.2.1) used to train these strategies. Specifically, I was interested in—

- examining whether the strategies produced with CSLA were capable of beating a strategy that is known to do well;
- examining whether strategies produced by CSLA were able to perform well in environments different from those they were trained in;
- comparing how well a strategy that is trained only against itself (i.e., all agents in the simulated game in Step 7 of the CSLA algorithm use strategy s) can do at repelling an invader, versus how well a strategy trained against the invader (i.e. the invading strategy is included in the population of agents at Step 7) can do at repelling the invader.

For the previous experiments, I assumed the algorithm had an oracle for π_{Obs}. For the rest of this section I will be running experimental simulations, so the oracle will observe what the agents do in the simulations and construct π_{Obs} from this, as

described in Section 5.2.

For the known good strategy I used an algorithm called EVChooser, which performs a few innovation and observation actions early in the game and uses the results of these actions (along with a discount factor) to estimate the expected value of innovating, observing, and exploiting, making the action with the highest expected value. It placed 15th out of over 100 entries in the Cultaptation tournament [15]. We chose EVChooser because (1) it has been shown to be a competitive strategy, (2) its source code was readily available to me (unlike the other successful strategies from the Cultaptation tournament), and (3) it could be tuned to perform well in the Cultaptation environments I used (which, in order to accommodate CSLA's exponential running time, were much smaller than those used in the international Cultaptation tournament).

For games as small as the ones in my experiments, I believe EVChooser is representative of most of the high-performing strategies from the tournament. Nearly all of the strategies described in the tournament report [15] spend some time trying to figure out what the innovate and observe distributions look like, and afterwards use some heuristic for choosing whether to innovate, observe, or exploit their best known action on any given round. This heuristic often involves some type of expected-value computation; for instance, the winning strategy *discountmachine* used a discount factor to compare the utility gained by exploiting the current best-known action to the utility of possibly learning a better action and exploiting it on all future rounds,

which is exactly what EVChooser does.[2] Unlike my CSLA algorithm, none of the strategies in the tournament conducted lookahead search.

For this experiment, I used an environment where π_{Inv} was a uniform distribution over the actions $\{20, 40, 80, 160\}$, probability of change was 1%, and probability of death was 25%. Due to the exponential running time of my strategy generating algorithm, this is the largest environment (i.e., smallest probability of death, highest number of actions and action values) for which the algorithm could compute full strategies in a reasonable amount of time.

Convergence and Consistency of CSLA

As part of this work, I have developed a Java implementation of Algorithm 2 that allows one to specify the type of game to be used for the simulation in Step 7, and created two strategies: s_{self} and s_{EVC}. The training process for both strategies began with s_0, the best-response to a random π_{Obs} distribution, and continued by constructing a strategy s_{i+1} as a best-response to the π_{Obs} generated by simulating games involving s_i. When training s_{self} the simulated games consisted solely of agents using s_i, but while training s_{EVC} they consisted of a population of agents using s_i being invaded by EVChooser. In both cases, 100 games were simulated at each step of the iteration, to limit the amount of noise in the π_{Obs} that was extracted from the simulations.

[2] *discountmachine* differs from EVChooser largely because it modifies the expected value of Observing using a machine-learned function that accounts for observe actions being unreliable and returning multiple actions, neither of which are possible in my version of the game

While there are no theoretical guarantees that the strategies produced by Algorithm 2 will converge, the algorithm's similarity to policy iteration [35] led me to suspect that the they would converge. Also, since CSLA is greedy, i.e., it selects the best response strategy at each step of the iteration, I was interested in seeing whether the strategy it found represented a local maximum or a global one.

I designed a simple experiment to see how these issues would play out when generating s_{self} and s_{EVC}: I modified the program to use a randomly-generated distribution for the initial value of π_{Obs}, rather than always initially setting $\pi_{\text{Obs}} = \pi_{\text{Inv}}$ as is done in Algorithm 2, and I used this modified program to generate 100 alternate versions of s_{self} and s_{EVC}. I then compared these alternates to the original s_{self} and s_{EVC} using stratDiff. In the case of s_{self}, I found that all 100 alternate versions were identical to the original. In the case of s_{EVC}, I found that 58 alternate versions were identical to the original, and the rest exhibited a stratDiff of no more than 1.08×10^{-4}. This means that an agent using an alternate version of s_{EVC} would choose all the same actions as one using the original s_{EVC} at least 99.989% of the time. This tells us that not only does CSLA converge for the environment I am testing it in, it converges to the same strategy each time it is run. This suggests that the algorithm is finding a globally-best solution for this environment, rather than getting stuck in a local maximum.

Finally, to estimate how different s_{self} and s_{EVC} are, I ran stratDiff$(s_{\text{self}}, s_{\text{EVC}})$ and found it to be 0.27. This means that training a strategy against an external, fixed strategy in Algorithm 2 does produce significantly different results than training a strategy against itself. For a more in-depth look at where s_{self} and s_{EVC} differ,

Table 5.3: Win percentages of s_{self} and s_{EVC} when playing against EVChooser over 10,000 games as both Defender and Invader.

	Win percentage	
	Defending vs. EVChooser	Invading vs. EVChooser
s_{self}	70.65%	70.16%
s_{EVC}	69.92%	69.92%

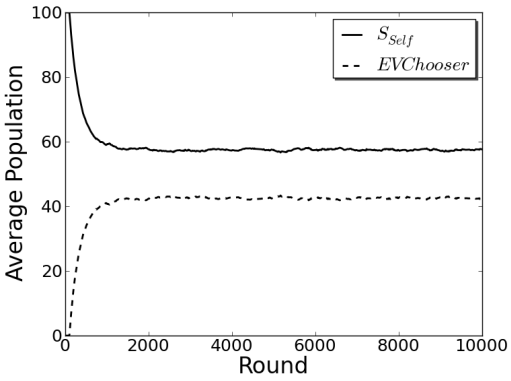
see Section 5.3.2.

Pairwise Competitions: s_{self} vs. EVChooser and s_{EVC} vs. EVChooser

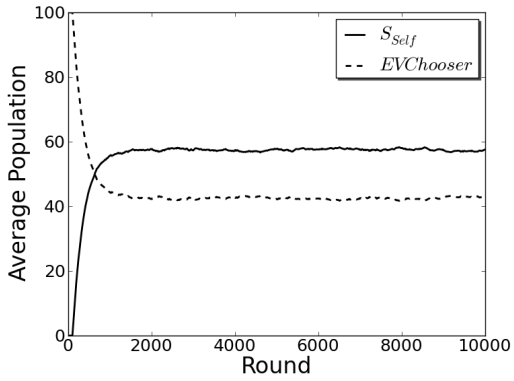
I played both of the generated strategies, s_{self} and s_{EVC}, against EVChooser for 20,000 games – in 10,000 games, the generated strategy was defending against an invading population of EVChooser agents, and in 10,000 games the roles were reversed, with the generated strategy invading and EVChooser defending. I recorded the population of each strategy on every round, as well as the winner of every game.[3] The populations in an individual game were extremely noisy, as seen in Figure 5.3(e), however by averaging the populations over all 10,000 games we can see some trends emerge. These average populations for each strategy in all four match-ups are presented in Figure 5.3(a–d), while the win rates for each match-up are presented in Table 5.3.

In Figure 5.3 we see that, on average, the strategies generated by Algorithm 2 control roughly 57% of the population for the majority of the game in all four match-ups. Interestingly, both s_{self} and s_{EVC} are able to reach this point in roughly the same amount of time whether they are invading or defending. It is also worth noting

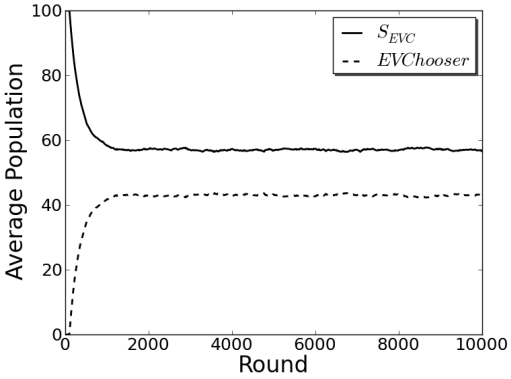
[3]Recall that the winner of a Cultaptation game is the strategy with the highest average population over the last quarter of the game.



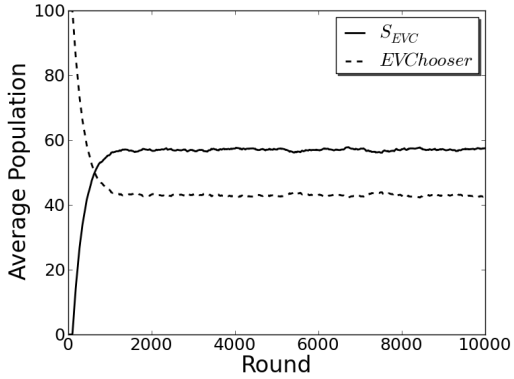
a) EVChooser invading s_{self}



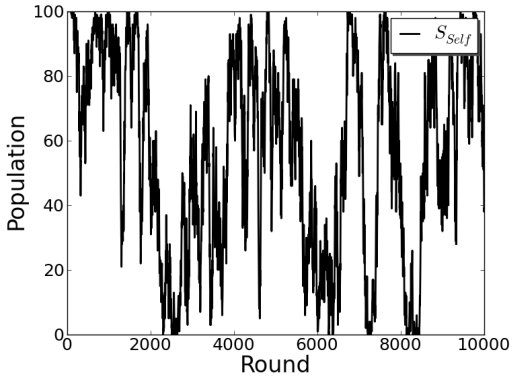
b) s_{self} invading EVChooser



c) EVChooser invading s_{EVC}



d) s_{EVC} invading EVChooser



e) Population of s_{self} at each round, in a single game against EVChooser

Figure 5.3: Average populations of both strategies for each round, in match-ups between s_{self} and EVChooser (parts a and b) and between s_{EVC} and EVChooser (parts c and d), over 10,000 games. From round 2000 onwards, s_{self} or s_{EVC} control 57% of the population on average, regardless of whether EVChooser was invading or defending. Since mutation is enabled from round 100 onwards, populations in an individual game (exhibited in part e) are highly mercurial and do not converge. Therefore, we must run a large number of trials and average the results to get a good idea of each strategy's expected performance.

Table 5.4: Percentage of games won (out of 10,000) by s_{self}, s_{EVC}, and EVChooser in a melee contest between all three.

	s_{self}	s_{EVC}	EVChooser
Melee win percentage	38.78%	37.38%	23.84%

that, even though I showed above that s_{self} and s_{EVC} have significant differences, they performed almost identically against EVChooser in terms of population and win percentages

Melee Competition: s_{self} vs. s_{EVC} vs. EVChooser

My next experiment was to run s_{self}, s_{EVC}, and EVChooser against one another in a melee contest to see how the three strategies would interact in an environment where none of them originally had the upper hand. All three strategies had an initial population of 33 agents at the start of each game. I used the same π_{Inv}, probability of change, and probability of death as in Experiment 2. Mutation was disabled for the final 2,500 rounds of each melee game, as was done in the Cultaptation tournament to allow the population to settle. I ran 10,000 games in this manner, and percentage of wins for each strategy are shown in Table 5.4.

In the table we can see that s_{self} has a slight edge over s_{EVC}, and both these strategies have a significant advantage over EVChooser. In fact, I observed that in the first 100 rounds of most games (before mutation begins) EVChooser nearly died out completely, although it is able to gain a foothold once mutation commences. Mutation is also turned off after 7500 rounds in Cultaptation melee games; this caused the population to quickly become dominated by one of the three strategies in all 10,000 games played.

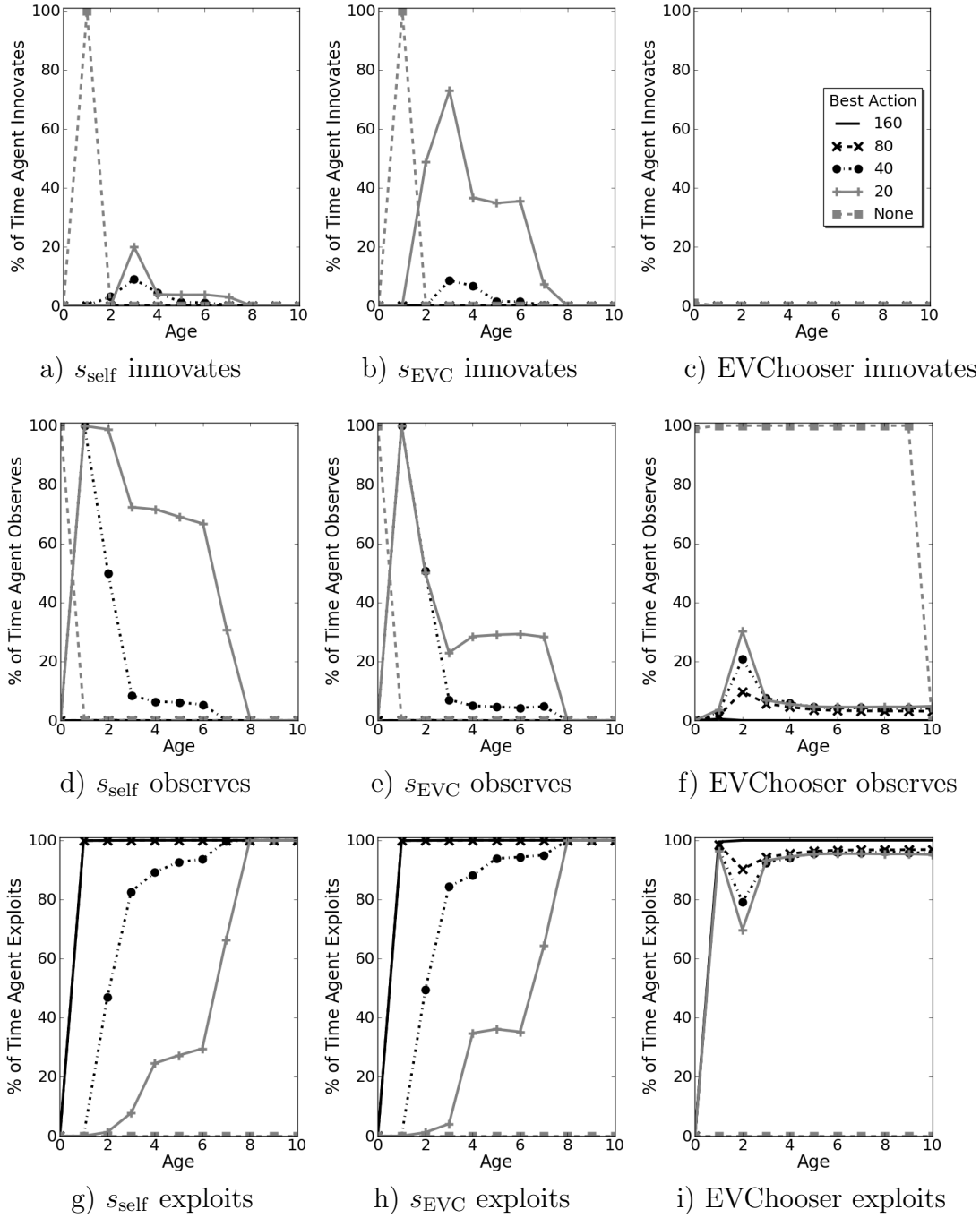
Performance Analysis of s_{self}, s_{EVC}, and EVChooser

In the experiments in Section 5.3.2, we saw that the strategies found by CSLA consistently outperform EVChooser in environments similar to the ones they were trained in. In order to get a better idea of why this happens, I ran two experiments to compare the performance of s_{self} and EVChooser in more detail. The first was designed to show the kinds of situations in which the two strategies chose different actions, while the second was designed to show how well the two strategies were able to spread good actions through their population.

Action Preferences

The objective of this experiment was to identify the kinds of situations in which s_{self}, s_{EVC}, and EVChooser made different choices. To this end, I allowed s_{self} to play against itself for five games, in an environment identical to the one used for the previous experiments in Section 5.3.2 (note that this is the same environment s_{self} was trained in). On each round, for each agent, I recorded the number of rounds the agent had lived, the value of the best action in its repertoire,[4] and whether the agent chose to innovate, exploit, or observe on that round. Since there are 100 agents alive at any given time and each game lasts 10,000 rounds, this yielded a total of five million samples. Figures 5.4(a), (d), and (g) show the observed probability that s_{self} would innovate, observe, or exploit (respectively) for its first ten rounds and for each possible best action value. I then repeated this process for s_{EVC} and EVChooser, allowing each strategy to play against itself for five games and recording the same

[4]This could be 20, 40, 80, 160, or None if the agent had not yet discovered an action



a) s_{self} innovates b) s_{EVC} innovates c) EVChooser innovates

d) s_{self} observes e) s_{EVC} observes f) EVChooser observes

g) s_{self} exploits h) s_{EVC} exploits i) EVChooser exploits

Figure 5.4: The observed probability that s_{self}, s_{EVC}, and EVChooser will innovate, observe, or exploit when they are a given number of rounds old (on the x-axis) and with a given value of the best action in the agent's repertoire. These results were observed by allowing each strategy to play itself for five games of 10,000 rounds each with 100 agents alive on each round, generating a total of 5,000,000 samples. All graphs in this figure share the same legend, which is included in graph c) and omitted elsewhere to save space.

data. The results for s_{EVC} and EVChooser may be found in Figures 5.4(b), (e), and (h), and Figures 5.4(c), (f), and (i), respectively.

The most obvious difference among the three strategies is that EVChooser almost never innovates,[5] a property it shares with the strategies that did well in the Cultaptation tournament [17]. On the other hand, s_{self} and s_{EVC} have conditions under which they innovate and conditions under which they do not. For instance, both s_{self} and s_{EVC} always innovate if their first action (which is always an observation) returns no action. Also, s_{EVC} frequently innovates if it is stuck with the worst action after several observes, and s_{self} also innovates (although less frequently; see next paragraph) in this case. Another sharp contrast between EVChooser and the generated strategies is in their exploitation actions. EVChooser spends nearly all of its time exploiting, even if it has a low-value action, and only observes with significant probability on round two. On the other hand, s_{self} and s_{EVC} will begin exploiting immediately if they have one of the two best actions, but otherwise will spend several rounds observing or innovating to attempt to find a better one, and the number of rounds they spend searching for a better action increases as the quality of their best known action decreases.

The main difference between s_{self} and s_{EVC} that can be seen in Figure 5.4 is in the way they handle being stuck with the lowest-value action after several rounds. In these circumstances, s_{self} prefers observation while s_{EVC} prefers innovation. Here we see the most obvious impact of the differing environments used to generate these two strategies. s_{self} prefers observation in these cases because it was trained in an

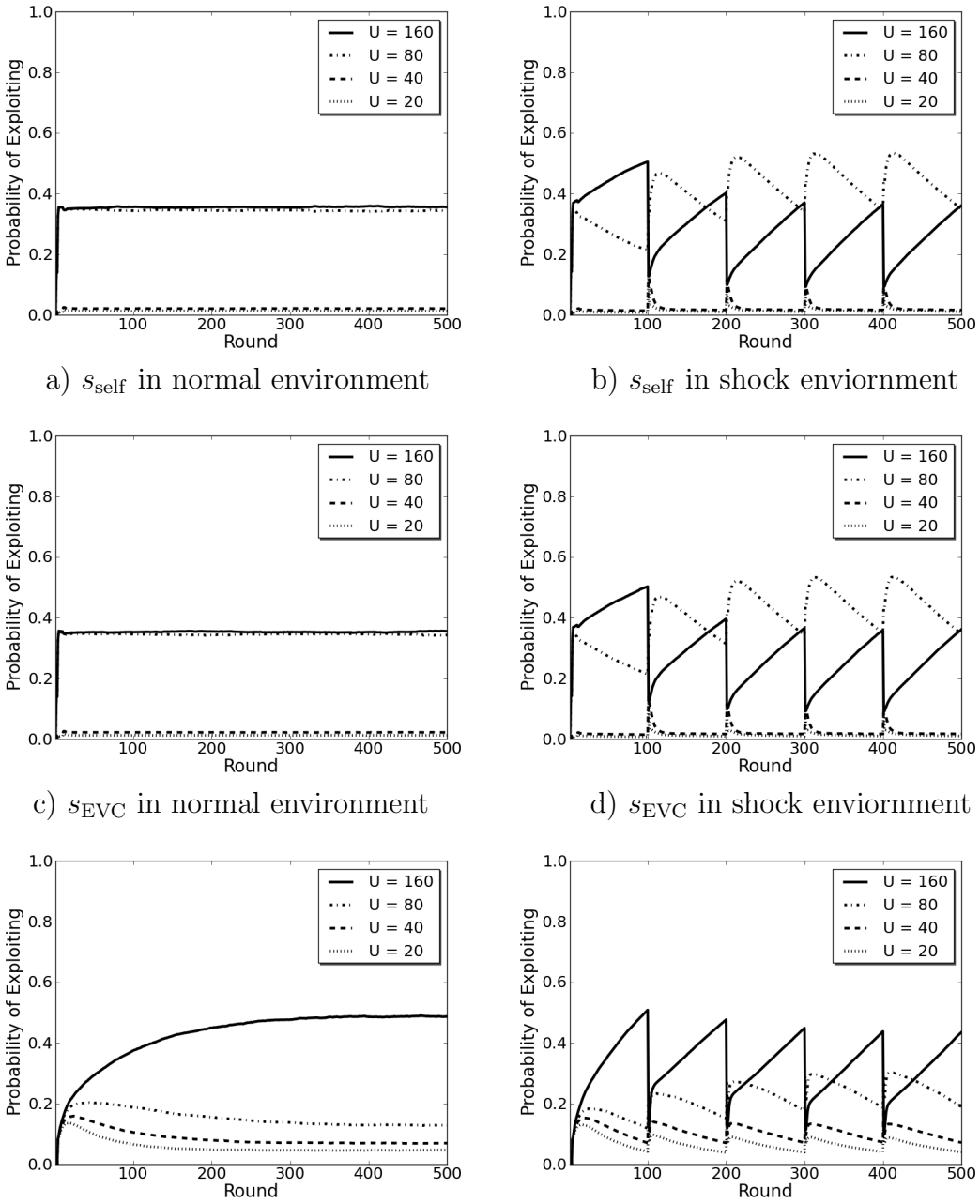
[5]EVChooser innovates 1% of the time on its first round.

environment where all agents are willing to perform innovation. Therefore, if an s_{self} agent is stuck with a bad action for more than a few rounds it will continue to observe other agents, since if a better action exists, it is likely that it has already been innovated by another agent and is spreading through the population. On the other hand, s_{EVC} prefers innovation in these situations because it has been trained with EVChooser occupying a significant portion of the population, and we have seen that EVChooser almost never innovates. Therefore, if s_{EVC} is stuck with a bad action after several rounds, it will attempt to innovate to find a better one, since it is less likely that another agent has already done so.

Spreading High-value Actions

The objective of this experiment was to measure the rate at which s_{self}, s_{EVC}, and EVChooser were able to spread high-valued actions through their populations. To measure this, I again played s_{self} against itself in the same environment used in the previous experiment (which I will refer to as the "normal" environment in this section), and on each round I recorded the number of agents exploiting actions with each of the four possible values (20, 40, 80, and 160). To account for the noise introduced by changing action values, I ran 10,000 games and averaged the results for each round. I then repeated this process, playing s_{EVC} and EVChooser against themselves. The results for s_{self}, s_{EVC}, and EVChooser may be found in Figures 5.5(a), (c), and (e) respectively.

This experiment lets us see what the steady state for these strategies looks like, and how quickly they are able to reach it. However, I am also interested in seeing how they respond to structural shocks [21, 22] (i.e., how quickly the strategies



a) s_{self} in normal environment

b) s_{self} in shock enviornment

c) s_{EVC} in normal environment

d) s_{EVC} in shock enviornment

e) EVChooser in normal environment

f) EVChooser in shock enviornment

Figure 5.5: The average number of agents exploiting an action with value U in two environments. The "normal" environment in parts a, c, and e shows how quickly s_{self}, s_{EVC}, and EVChooser spread actions through their population under normal circumstances when they control the entire population. The "shock" environment in parts b, d, and f shows how quickly each strategy responds to periodic structural shock. The "normal" environment is the same as in the rest of Section 5.3.2, and the "shock" environment is similar except that actions with value 160 are forced to change every 100th round and held constant all other rounds. Each data point is an average over 10,000 games.

are able to recover when a good, widely-used action changes values). To this end, I created a "shock" environment, which is identical to the normal environment with one modification: actions with value 160 have a probability of change equal to 0 except on rounds divisible by 100, in which case they have probability of change equal to 1. All other actions use the normal probability of change for this environment, 0.01. This modification creates a shock every 100 rounds, while still keeping the expected number of changes the same for all actions. I then repeated the experiment above with the shock environment, running 10,000 games for s_{self}, s_{EVC}, and EVChooser and averaging the results, which are presented in Figures 5.5(b), (d), and (f) respectively.

In Figure 5.5 we can see that s_{self} and s_{EVC} exhibit nearly identical performance in both the normal and shock environments. In the normal environment, they are able to reach their steady state in only a few rounds, and the steady state consists of a roughly equal number of agents exploiting the best and second-best action. In the shock environment, we see that s_{self} and s_{EVC} respond to external shock by drastically increasing the number of agents exploiting the second-best action over the course of a few rounds, and returning to their steady states at a roughly linear rate over the next 100 rounds. The number of s_{self} and s_{EVC} agents exploiting the two worst actions remains extremely low except for small spikes immediately after each shock.

Compared to the generated strategies, EVChooser's performance appears to be less stable, and less robust to structural shock. In the normal environment, we see that EVChooser takes hundreds of rounds to reach its steady state. While

EVChooser's steady state does include more agents exploiting the best action than s_{self} and s_{EVC}, it also includes a significant number of agents exploiting the two worst actions. In the shock environment, we see that changes to the best action result in significant increases to the number of EVChooser agents exploiting the other actions, including the two worst ones. We can also see that populations of EVChooser agents take a lot longer to return to normal after an external shock than populations of s_{self} and s_{EVC}. These results help us account for the superior performance of s_{self} and s_{EVC} over EVChooser in previous experiments, and indicate that there is plenty of room for improvement in EVChooser and strategies like it.

Chapter 6

Improving Full-scale Cultaptation Strategies

So far, the objective of this work has been to analyze Cultaptation and see what types of strategies work well. However, in order to find strategies that were provably good using the algorithms I was able to develop, it was necessary to use game environments that were much smaller than the ones used in the real Cultaptation tournament. In this chapter, I will show how to take the observations and analysis used in the previous chapters and use them to develop a strategy for full-size Cultaptation that can outperform the current best strategy from the Cultaptation tournament.

6.1 The Tournament Winner: *discountmachine*

The first Cultaptaton tournament was won by the strategy *discountmachine*, by Dan Cownden and Tim Lillicrap of Queen's University [15]. According to the tournament organizers, *discountmachine* performed significantly better than all other contestants; its population was about 50% higher, on average, than the second-place strategy [17].

Pseudocode for *discountmachine* is provided as Algorithm 3. It can be summarized as follows: On the first round, it always plays Observe. If it's the second round and the first Observe returned no action, it will always play Innovate as a

bootstrapping measure. Otherwise, it begins by estimating the probability of change ($\hat{c}$), the average value of the payoff distribution ($\hat{\mu}_\pi$), the average value of the observe distribution ($\hat{\mu}_{\mathsf{Obs}}$), and the current value of its best-known action ($\hat{v^*}$). The values $\hat{v^*}$ and $\hat{\mu}_{\mathsf{Obs}}$ are then multiplied by a discount factor meant to reflect the value of exploiting the actions until either the action changes or the agent dies.

In order to prevent the agent from missing out on high-value actions in environments with low probability of change, the strategy includes a heuristic that will cause it to Observe at least once every 20 rounds if $\hat{c} < 0.05$ and its current action is not significantly better than any it has seen before. Otherwise, it decides which action to perform based on the output of a pre-trained neural network, which takes as input discounted versions of $\hat{v^*}$ and $\hat{\mu}_{\mathsf{Obs}}$, along with an estimate of the degree of noise present in Observation.[1]

While the source code for *discountmachine* used in the tournament has been made public on the organizers' website [15], that code includes a neural network with pre-trained weights, and the code used to train the neural network was not made public. However, a comment in the source code explains, "We trained this function by having it try and match the estimate made by a (strategy) with perfect knowledge of what could be observed and (the amount of noise present for Observe moves)."

[1]The strategy also defines a rare subset of values for some parameters for which the neural network performed poorly, and in these cases the strategy simply returns Observe if $\hat{v^*} < \hat{\mu}_{\mathsf{Obs}}$ and Exploits the best-known action otherwise. For ease of exposition I have omitted this from the pseudocode.

Algorithm 3 Simplified pseudocode for the *discountmachine* strategy, which won the first Cultaptation tournament. Returns the action selected by an agent, given the agent's current history h. Full source code available from the Cultaptation website [15].

discountmachine(History h)

if $|h| == 0$ **then**
  return Observe.
end if
if $|h| == 1$ AND No actions are known **then**
  return Innovate.
end if
Let $\hat{c}$ and $\hat{\mu}_\pi$ be the probability of change and average new action value, respectively, experienced in history h.
Let v^* be the value of the best known action, and r be the number of rounds since that action was seen in h.
$\hat{v^*} = (1 - \hat{c})^r \times v^* + (1 - (1 - \hat{c})^r) \times \hat{\mu}_\pi$.
Let $\hat{\mu}_{\mathsf{Obs}}$ be the mean value of all observed actions in h.
Let $f = (1 - \hat{c})(1 - 0.02)$. This is the "discount factor."
$\hat{v^*} = \hat{v^*} \times \frac{1}{(1-f)}$
$\hat{\mu}_{\mathsf{Obs}} = \hat{\mu}_{\mathsf{Obs}} \times \frac{f}{1-f}$.
if $\hat{c} < 0.05$ AND $|h| > 20$ AND Last 20 actions have been Exploit **then**
  if $\hat{v^*} < 3 + \mathrm{PRU}(h)$ **then**
    return Observe.
  else
    return Exploit the best-known action.
  end if
end if
return The action selected by a pre-trained neural network with inputs $\hat{v}_{\max}$, $\hat{\mu}_{\mathsf{Obs}}$, and an estimate of the amount of noise present in Observation.

6.1.1 Potential Problems with *discountmachine*

It's clear from the results of the Cultaptation tournament that *discountmachine* is a significantly better strategy than the other contestants. However, the analysis and experiments presented in Chapters 4 and 5 suggest that it has some aspects that could still be improved.

First, *discountmachine* only ever innovates as a bootstrapping mechanism on the second round of the game. While the tournament organizers found that this was

a universal characteristic among the top-performing strategies in the tournament, Sections 2.2.1 and 5.3.2 present theoretical and empirical evidence that, in at least some circumstances, Innovating can be advantageous. Therefore, a strategy that could recognize these situations would have an advantage over *discountmachine*.

Second, the formulae used to obtain the total value of exploiting the best-known action (line 12) and the expected result of an observe (line 13) are incomplete. These formulae compute the expected total payoff the agent will earn by exploiting each of these actions until either the action changes value or the agent dies. As we have seen in Example 5 and Theorem 1, the expected total payoff is not always the same as the EPRU, and EPRU is the metric that should be used to determine the quality of a strategy (or, in this case, a potential course of action). Furthermore, the formula on line 13 neglects the fact that the information returned by an Observe pertains to the *round before* the Observe was performed. This means that, if the newly-observed action is Exploited on the following round, it has had *two* opportunities to change values, not one, and the expected value of the action should be updated to reflect this.

Since the source code used to train the neural net in *discountmachine* is not provided, it's difficult to characterize the function approximated by the network. However, given the issues with the formulae used to compute the inputs to the network, it seems unlikely that it is perfectly accurate in determining whether Observing or Exploiting is a better choice.

Finally, the experimental results from Section 5.3.2 suggest that the choice of whether to exploit a given action should be determined by the action's value and

the agent's age, and that the preference towards exploiting should increase monotonically as the agent gets older and as the value of the best known action increases. However, the heuristic used on lines 14 through 19 causes *discountmachine* to consistently violate this rule once every 20 rounds in environments with low probability of change. While this does cause the agent to have a better idea of what actions exist to be observed, I have been unable to find any analytical support for the notion that this will typically increase the agent's EPRU by more than the amount that is lost by not exploiting as much as possible when the agent already has a high-value action.

6.2 A Full-scale Cultaptation Player: *relaxedlookahead*

In order to show how the analysis and experiments presented in Chapters 4 and 5 can be used to create a player for the Cultaptation tournament, I developed *relaxedlookahead*. The main insights that inspired *relaxedlookahead*'s design were:

1. Actions should be evaluated based upon the estimated amount of EPRU they provide the agent, since EPRU is proportional to reproductive success (Theorem 1). While this may seem intuitive, I know of no current Cultaptation player that uses EPRU or an equivalent metric (e.g., *discountmachine* estimates expected total payoff, see above).

2. Innovating should be regarded as a worthwhile choice in at least some circumstances. Even though the top-performing strategies from the tournament tended to eschew Innovate moves, we have seen that there are some circum-

stances in which it is preferable. With an accurate formula for estimating the value of innovation, the agent should be able to identify such circumstances.

3. The agent should choose its move based upon its entire history; in other words, hard and fast rules such as the one used by *discountmachine* on lines 14 through 19 should be avoided. This will allow the agent to have behavior that smoothly transitions according to the value of its best action and age, as s_{self}'s behavior does in Figure 5.3.2, which gave it a significant performance advantage over EVChooser.

4. Due to computational constraints present in the Cultaptation tournament, it will not be possible to perform a full lookahead search to a depth sufficient to achieve a small ϵ. However, a *relaxed* lookahead search, which considers a few different possible outcomes of each action and their effects over the course of many rounds, should still be sufficient to allow the agent to estimate which of three choices (Exploit the best known action, Observe, or Innovate) has the highest EPRU.

The *relaxedlookahead* strategy is defined in Algorithm 4, and can be summarized as follows. Like *discountmachine*, it always observes on its first round and, if that observe fails to return a value, it innovates as a bootstrapping measure. Otherwise, the strategy begins by creating estimates of several quantities based on the information contained in h; it estimates the probability of change ($\hat{c}$), average value of the payoff distribution ($\hat{\mu}_\pi$), current value of the best known action ($\hat{v^*}$), and the mean and standard deviation of the distribution of observe moves ($\hat{\mu}_{\mathsf{Obs}}$, $\hat{\sigma}_{\mathsf{Obs}}$). It

then estimates the value of three possible moves: exploiting the best known action, observing, and innovating. These estimates are created using a relaxed lookahead search, which is a very rough approximation of the search conducted in Chapter 5.

The relaxed lookahead search is conducted slightly differently for each of the three possible moves. For the observe move, the strategy models the observe distribution π_{Obs} as a normal distribution[2] with mean $\hat{\mu}_{\mathsf{Obs}}$ and standard deviation $\hat{\sigma}_{\mathsf{Obs}}$, and calculates the probability that the observed action will have a higher payoff than $\hat{v^*}$. The search then considers six different branches. In the first branch, the observed action has a payoff lower than the current best action, so the agent's best known action is still $\hat{v^*}$. The other five branches assume the observed action is better than the current best action; candidate payoffs are generated by using the inverse CDF of π_{Obs} to find five values appropriately distributed among the set of values greater than $\hat{v^*}$. For example, if $\hat{v^*}$ was found to be in the 80th percentile of π_{Obs}, the five candidate values would be in the 82nd, 86th, 90th, 94th, and 98th percentiles respectively. For each branch, the strategy assumes the agent will exploit its new best known action for 100 rounds, and it estimates the EPRU of the branch using formulae based on those from Chapter 4. Finally, the EPRU of the observe move is computed by combining the EPRU of each branch with its estimated likelihood.

[2]The normal distribution was selected as the most likely shape of π_{Obs} using a process similar to the one in CSLA; the first version of *relaxedlookahead* did not assume any shape to π_{Obs} and only considered its average value. This version was then played against itself in simulated games, and the shape of the observe distribution from the simulations appeared to be gaussian. I then changed *relaxedlookahead* to model the observe distribution as a normal distribution, repeated the process, and found that the observe distribution from the simulations was still gaussian.

The EPRU of innovating is estimated in a similar manner, but the strategy models the payoff distribution π as an exponential distribution[3] rather than a normal one. Finally, the EPRU of exploiting the current best action is estimated by simply assuming that the agent exploits on the current round and the next 100 rounds. Once the EPRUs for observing, innovating, and exploiting have all been estimated, *relaxedlookahead* simply makes the move with the highest estimate.

6.3 Experiments: RLA vs. discountmachine

In order to test the hypothesis that *relaxedlookahead* would have an advantage over *discountmachine*, I conducted several experiments to compare their performance.

6.3.1 Overall Performance

The first experiment examines the overall performance of both strategies in a variety of environments. I created two different versions of the action distribution π. The first, π_{Geo}, is a geometric distribution with probability of success $p = 0.1$. The second, π_{Gam}, is a Gamma distribution with shape $k = 2$ and scale $\theta = 5$. Both these distributions have a mean value of 10 and conform to the tournament organizers' specifications of generating a larger number of small-value actions with

[3]The exponential distribution was selected due to the tournament organizers' statement that the move distribution would typically contain large numbers of low-value actions and small numbers of high-value actions [15]. The exponential distribution is generally representative of distributions that have this shape, and can be easily constructed since its shape depends only on its mean.

Algorithm 4 Simplified pseudocode for the *relaxedlookahead* strategy. Returns the action selected by an agent, given the agent's current history h.

relaxedlookahead(History h)

if $|h| == 0$ **then**
  return Observe.
end if
if $|h| == 1$ AND No actions are known **then**
  return Innovate.
end if
Let $\hat{c}$ and $\hat{\mu}_\pi$ be the probability of change and average new action value, respectively, experienced in history h.
Let v^* be the value of the best known action, and r be the number of rounds since that action was seen in h.
$\hat{v^*} = (1-\hat{c})^r \times v^* + (1-(1-\hat{c})^r) \times \hat{\mu}_\pi$.
Let $\hat{\mu}_{\mathsf{Obs}}$ and $\hat{\sigma}_{\mathsf{Obs}}$ be the mean value and standard deviation, respectively, of all observed actions in h.
U_O = evaluateObs($\hat{\mu}_{\mathsf{Obs}}$, $\hat{\sigma}_{\mathsf{Obs}}$, $\hat{v^*}$, $\hat{\mu}_\pi$, $\hat{c}$, $|h|$)
U_I = evaluateInv($\hat{v^*}$, $\hat{\mu}_\pi$, $\hat{c}$, $|h|$)
U_X = estimateEPRU($\hat{v^*}$, $\hat{\mu}_\pi$, $\hat{c}$, $|h|$)
if $U_X \geq \max(U_O, U_I)$ **then**
  return Exploit the best known action.
else if $U_O \geq U_I$ **then**
  return Observe.
else
  return Innovate.
end if

estimateEPRU(v, $\bar{v}$, c, r)

$U = 0$
EPRU $= 0$
for $i = 0 \rightarrow 100$ **do**
  $U = U + (1-c)^i \times v + (1-(1-c)^i) \times \bar{v}$
  EPRU $=$ EPRU $+ 0.98^i \times \frac{U}{r+i}$
end for
return EPRU

a few high-value actions [15]. However, π_{Geo} largely matches the assumptions *relaxedlookahead* makes about the shape of π while π_{Gam} does not. It features a large "hump" around the low-value actions, with the most likely value being 5, and a lower probability of high-value actions than π_{Geo}.

Algorithm 5 Pseudocode for the functions used to evaluate Observation and Innovation by *relaxedlookahead*.

evaluateObs(μ, σ, v, $\bar{v}$, c, r)

Let π_{Obs} be a normal distribution with mean μ and standard deviation σ.
Let p_{hi} be the probability that a value drawn from π_{Obs} is greater than v.
$U_{\text{hi}} = 0$
for $i = 0 \to 4$ **do**
    $p = 1 - p_{\text{hi}} \times \frac{1+2i}{10}$
    Let v_i be the value of the inverse CDF of π_{Obs} at percentile p.
    $v_i = (1-c)^2 \times v_i + (1 - (1-c)^2) \times \bar{v}$
    $U_{\text{hi}} = U_{\text{hi}} + \text{estimateEPRU}(v_i, \bar{v}, c, r+1)$
end for
$U_{\text{hi}} = U_{\text{hi}}/5$
$U_{\text{low}} = \text{estimateEPRU}(v, \bar{v}, c, r+1)$
return $p_{\text{hi}}U_{\text{hi}} + (1 - p_{\text{hi}})U_{\text{low}}$

evaluateInv(μ, v, $\bar{v}$, c, r)

Let π_{Inv} be an exponential distribution with mean μ.
Let p_{hi} be the probability that a value drawn from π_{Inv} is greater than v.
$U_{\text{hi}} = 0$
for $i = 0 \to 4$ **do**
    $p = 1 - p_{\text{hi}} \times \frac{1+2i}{10}$
    Let v_i be the value of the inverse CDF of π_{Inv} at percentile p.
    $v_i = (1-c) \times v_i + c \times \bar{v}$
    $U_{\text{hi}} = U_{\text{hi}} + \text{estimateEPRU}(v_i, \bar{v}, c, r+1)$
end for
$U_{\text{hi}} = U_{\text{hi}}/5$
$U_{\text{low}} = \text{estimateEPRU}(v, \bar{v}, c, r+1)$
return $p_{\text{hi}}U_{\text{hi}} + (1 - p_{\text{hi}})U_{\text{low}}$

I simulated 200 pairwise Cultaptation games between *relaxedlookahead* and *discountmachine* (100 with *relaxedlookahead* invading and 100 with *discountmachine* invading) in each of 14 different environments. The first seven environments used π_{Geo} as the action distribution, and had probability of change c equal to 0.001, 0.005, 0.01, 0.05, 0.1, 0.2, and 0.4, respectively. The other environments used π_{Gam} as the action distribution. This set of values for c is the same set used by the Cultaptation tournament organizers for their first round of evaluations [15].

Table 6.1: Performance of *relaxedlookahead* when played against *discountmachine* in a variety of environments. Results are based on 200 simulated games for each environment (100 with *relaxedlookahead* invading, 100 with *discountmachine* invading). Results in bold or italics are statistically significant with $\rho < 0.01$; bold results represent an advantage for *relaxedlookahead*, italicized results represent an advantage for *discountmachine*.

	Action distribution π_{Geo}		Action distribution π_{Gam}	
	Win percentage	Average score	Win percentage	Average score
$c = 0.001$	**.96**	82.02	**.97**	83.96
$c = 0.005$	**.78**	66.29	**.84**	70.86
$c = 0.01$	**.65**	58.20	**.72**	63.77
$c = 0.05$	.48	47.70	.55	52.21
$c = 0.1$	.53	52.97	**.59**	55.24
$c = 0.2$	.44	47.32	*.41*	47.38
$c = 0.4$	.55	53.95	.42	45.04

The results of this experiment are presented in Table 6.1. *relaxedlookahead* has a statistically significant advantage ($\rho < 0.01$) over *discountmachine* in seven environments: $c = 0.001, 0.005$, and 0.01 for both action distributions, and $c = 0.1$ for π_{Gam}. It has a statistically significant disadvantage for one environment, $c = 0.2$ with distribution π_{Gam}.

These results suggest that *relaxedlookahead* has a significant advantage over *discountmachine* for environments with low probability of change, and that its advantage shrinks and eventually disappears as the probability of change grows to moderate and high levels. Curiously, the shape of π did not seem to make much difference in *relaxedlookahead*'s performance, and in fact it performed slightly better in the environment which did *not* match its assumptions. We will investigate the reasons for this in the following section.

It should be noted that, in general, it will be much more difficult to obtain a significant advantage in environments with high probabilities of change, since as long as both players are exploiting as much as possible their choice of action makes little difference. The tournament organizers have acknowledged this and, appropriately, put significantly more emphasis on low-change environments in all rounds of their evaluation. Therefore, since *relaxedlookahead* has a clear advantage over *discountmachine* in such environments, it seems reasonable to conclude that *relaxedlookahead* would be more likely to win in a tournament setting.

6.3.2 Performance Analysis

The next round of experiments was designed to duplicate the ones presented in Section 5.3.2, in order to get a more detailed look at the differences between *relaxedlookahead* and *discountmachine*.

Action Preferences

The objective of this experiment, like the first one in Section 5.3.2, was to get an idea of the kinds of situations in which *relaxedlookahead* and *discountmachine* made different choices. To that end, I allowed each strategy to compete against itself in 50 full-length games using four separate environments: $c = 0.001$ and 0.05 with action distribution π_{Geo}, and $c = 0.001$ and 0.05 with action distribution π_{Gam}.[4] On each round, I recorded the actions taken by all 100 agents and the payoff of their best-known action, yielding a total of 50×10^6 data points for each

[4]The values $c = 0.001$ and $c = 0.05$ were chosen because the first reflects the largest advantage for *relaxedlookahead* in the previous experiment, while the second represents the lowest value at which both strategies appear to be evenly matched.

strategy in each environment.

Figures 6.1 through 6.4 present the results of this experiment, showing the probability that each strategy would innovate, observe, or exploit given its age and the value of its best known action. There are two obvious differences between the strategies in all four environments.

First, *discountmachine*, like EVChooser, never innovates except as a bootstrapping mechanism on the second round of the game. Meanwhile, *relaxedlookahead* is willing to innovate on any round, under the right circumstances. In fact, if its best known action is between 5 and 15, *relaxedlookahead* appears to *prefer* innovating to observing. The fact that it does so in environments in which it handily defeats *discountmachine* is evidence that innovating should not be completely overlooked, as the top-performing strategies in the tournament tended to do.

Second, the observe and exploit graphs for *discountmachine* exhibit large spikes at regular intervals. These are due to the clause on line 14 of Algorithm 3, which causes *discountmachine* to observe at least once every 20 rounds unless the probability of change is high. The authors of the strategy explained that this is done to make sure that an agent doesn't miss a high-value action that most other agents are exploiting. Meanwhile, the same graphs for *relaxedlookahead* also exhibit spikes, but their size and the interval between them is correlated with the probability of change in the environment and the value of the best-known action. There is no code in *relaxedlookahead* that explicitly causes this behavior, so I can only conclude that this is emergent behavior and that certain rounds are more likely than others to be opportune times for exploring rather than exploiting. Whatever the cause, this is

evidence that arbitrary rules like the one on line 14 of Algorithm 3 are not necessary if the values of observing and exploiting can be estimated accurately.

Spreading High-value Actions

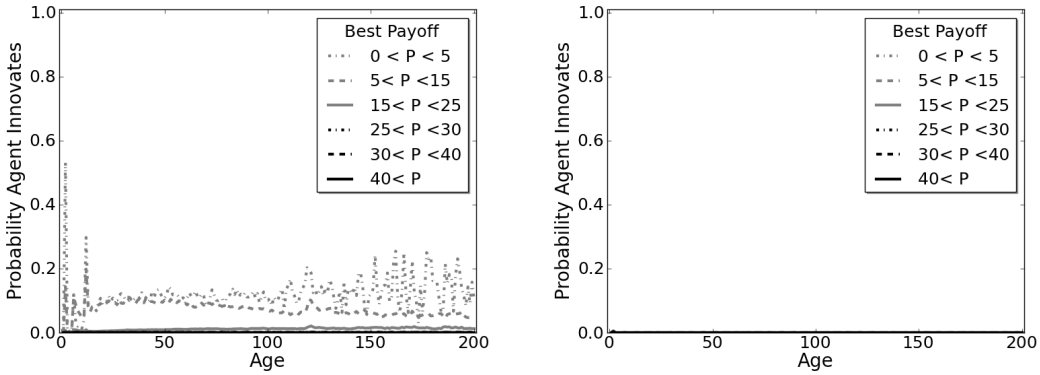
The objective of the next experiment, like the second one in Section 5.3.2, was to measure the rate at which *relaxedlookahead* and *discountmachine* were able to spread high-value actions through their populations. To measure this, I re-ran the simulations used for the previous experiment,[5] but this time recorded the value of each action exploited on each round of the game. Since there are 100 agents in each game, this yielded 5,000 data points for each round.

Figures 6.5 through 6.8 present the results of this experiment. They show the probability of each strategy exploiting an action in each of six possible value ranges: 0 to 5, 5 to 15, 15 to 25, 25 to 30, 30 to 40, and more than 40.[6]

These figures make it clear why *relaxedlookahead* has an advantage over *discountmachine*, particularly in environments with low probability of change. Over the first few generations of agents, *discountmachine* does an excellent job of quickly spreading actions in the best value category that can reliably be found (e.g., values over 40 in Figure 6.5, values between 15 and 25 in Figure 6.6). However, in all four environments, the proportion of agents exploiting such actions peaks at about round 200 and then begins to steadily decline. Meanwhile, the proportion of *relaxedlookahead* agents exploiting the same actions increases more slowly, but eventually reaches a steady state with more agents exploiting high-value actions

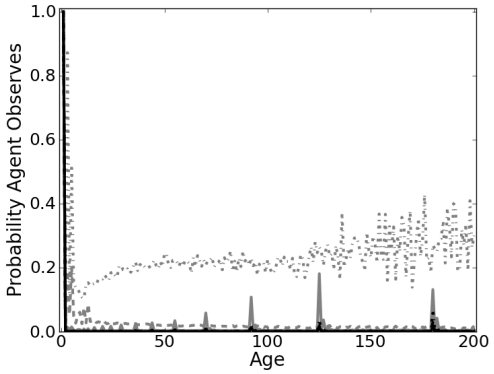
[5]Each strategy played against itself in 50 full-length Cultaptation games in each of four environments, with $c = 0.001$ or 0.05 and π_{geo} or π_{gam} as the payoff distribution.

[6]In both π_{geo} and π_{gam}, the mean action value is 10.

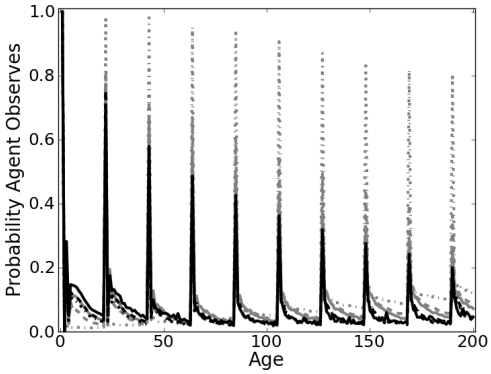


a) *relaxedlookahead* Innovates

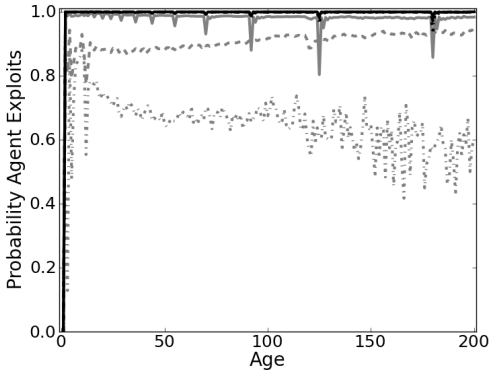
b) *discountmachine* Innovates



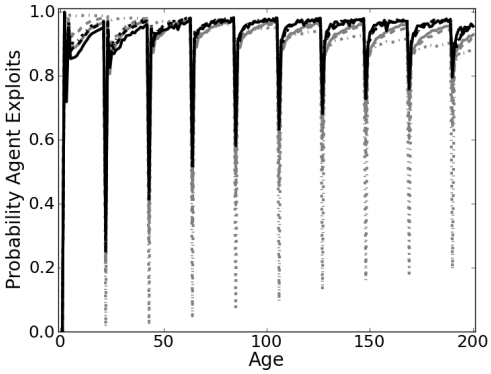
c) *relaxedlookahead* Observes



d) *discountmachine* Observes

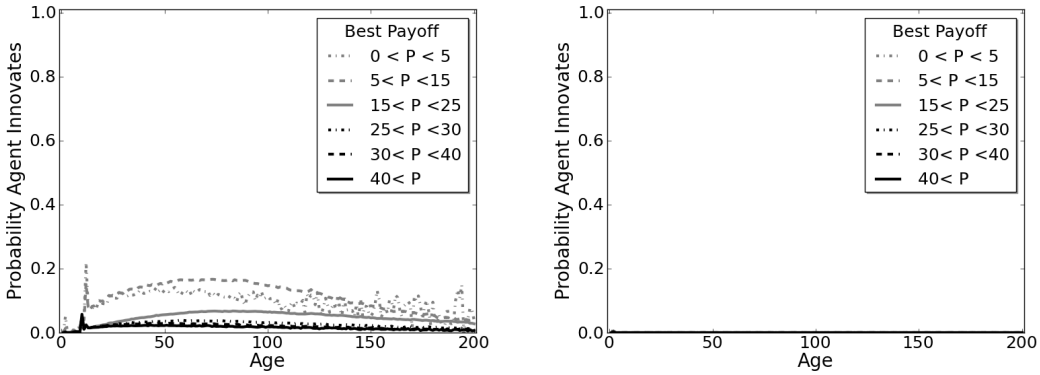


e) *relaxedlookahead* Exploits



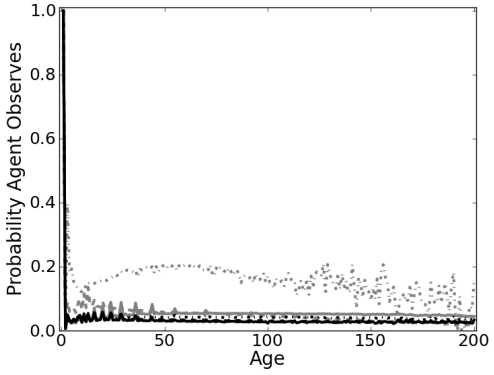
f) *discountmachine* Exploits

Figure 6.1: Frequency that an agent using *relaxedlookahead* (left) or *discountmachine* (right) chose to Innovate, Observe, or Exploit, given its age and best known action, in an environment with the geometrically-distributed action distribution π_{Geo} (with mean 10) and probability of change $c = 0.001$.

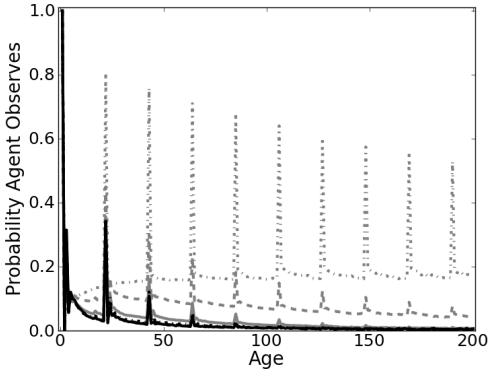


a) *relaxedlookahead* Innovates

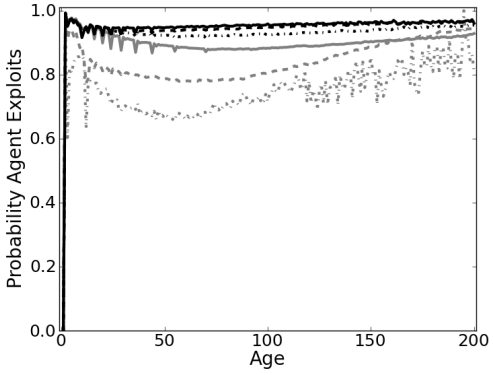
b) *discountmachine* Innovates



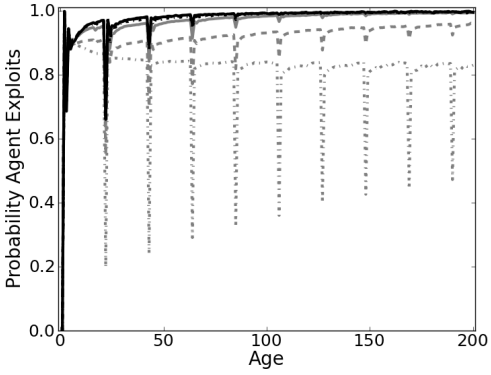
c) *relaxedlookahead* Observes



d) *discountmachine* Observes

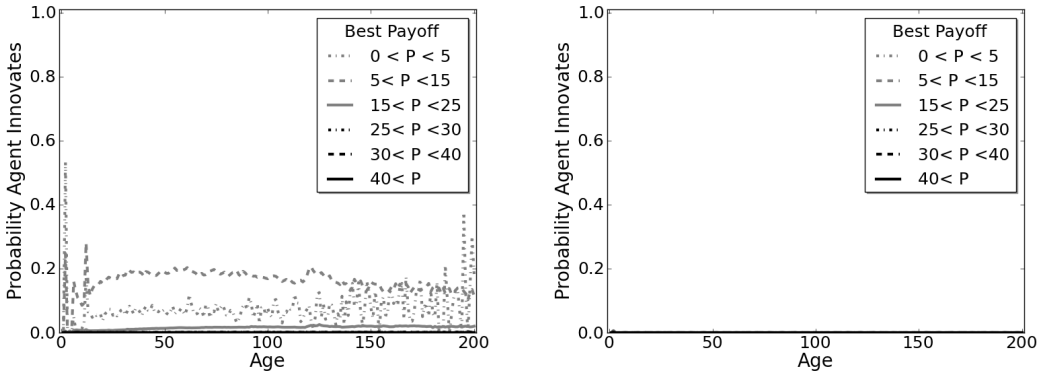


e) *relaxedlookahead* Exploits



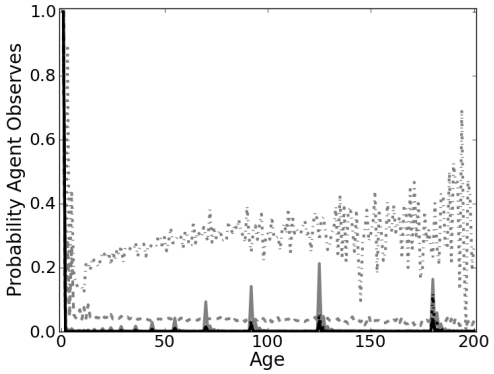
f) *discountmachine* Exploits

Figure 6.2: Frequency that an agent using *relaxedlookahead* (left) or *discountmachine* (right) chose to Innovate, Observe, or Exploit, given its age and best known action, in an environment with the geometrically-distributed action distribution π_{Geo} (with mean 10) and probability of change $c = 0.05$.

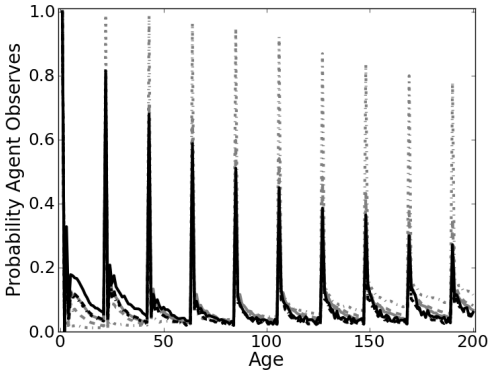


a) *relaxedlookahead* Innovates

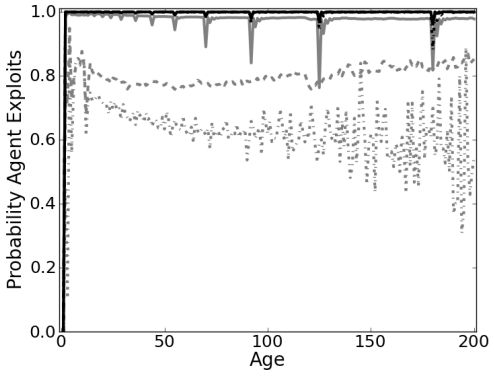
b) *discountmachine* Innovates



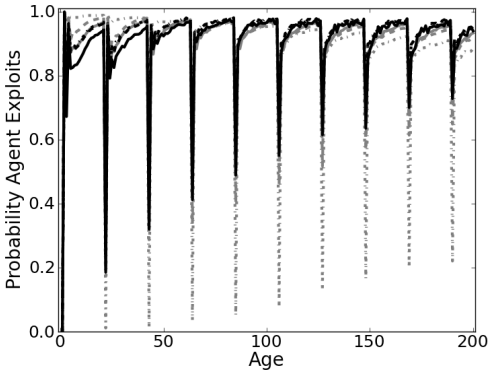
c) *relaxedlookahead* Observes



d) *discountmachine* Observes

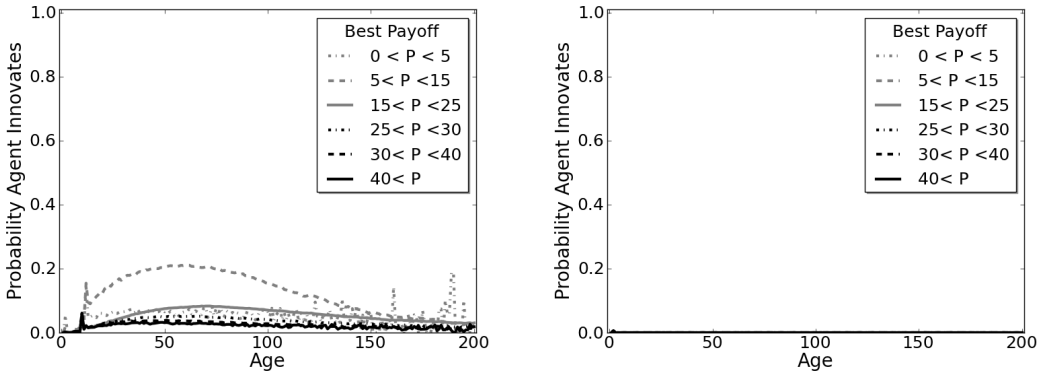


e) *relaxedlookahead* Exploits



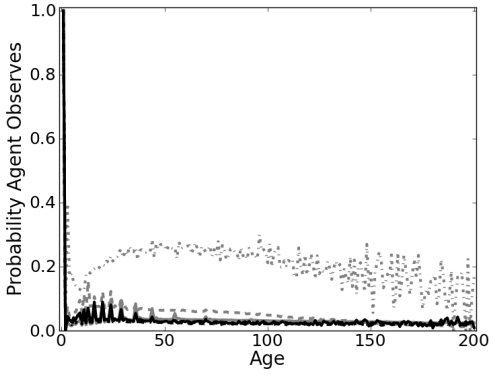
f) *discountmachine* Exploits

Figure 6.3: Frequency that an agent using *relaxedlookahead* (left) or *discountmachine* (right) chose to Innovate, Observe, or Exploit, given its age and best known action, in an environment with the action distribution π_{Gam} (i.e. a gamma distribution with shape 2 and scale 5) and probability of change $c = 0.001$.

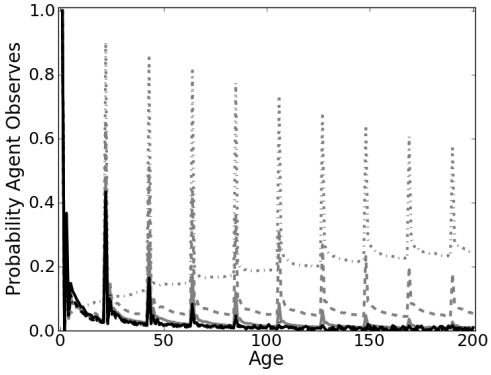


a) *relaxedlookahead* Innovates

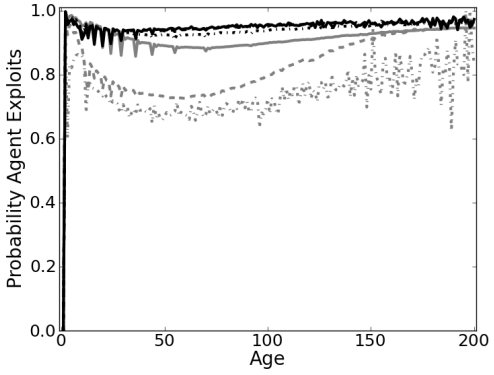
b) *discountmachine* Innovates



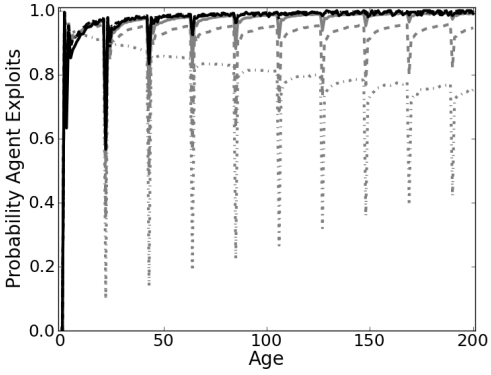
c) *relaxedlookahead* Observes



d) *discountmachine* Observes



e) *relaxedlookahead* Exploits



f) *discountmachine* Exploits

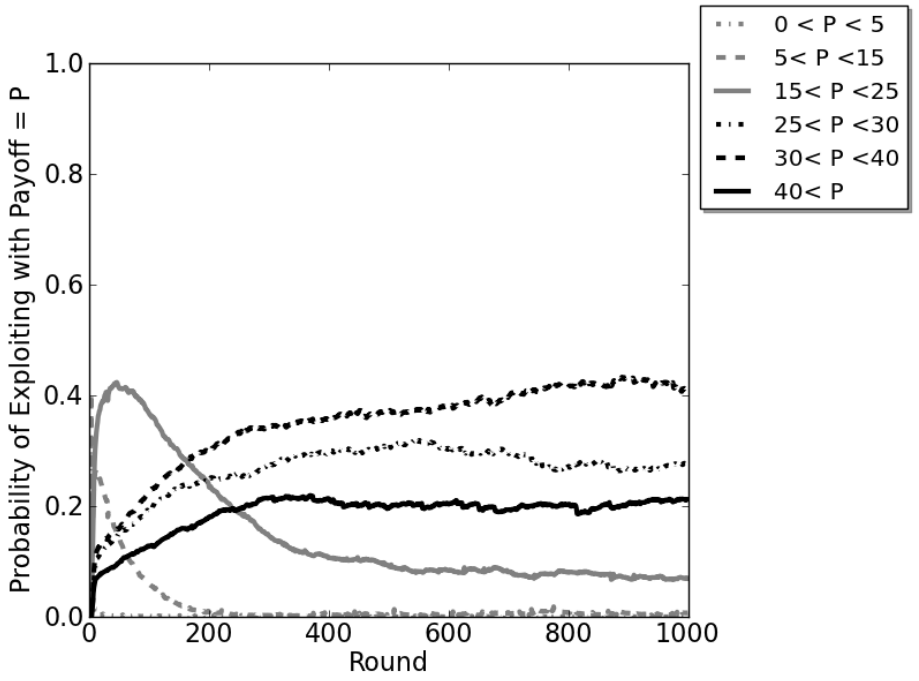
Figure 6.4: Frequency that an agent using *relaxedlookahead* (left) or *discountmachine* (right) chose to Innovate, Observe, or Exploit, given its age and best known action, in an environment with the action distribution π_{Gam} (i.e. a gamma distribution with shape 2 and scale 5) and probability of change $c = 0.001$.

than *discountmachine*.

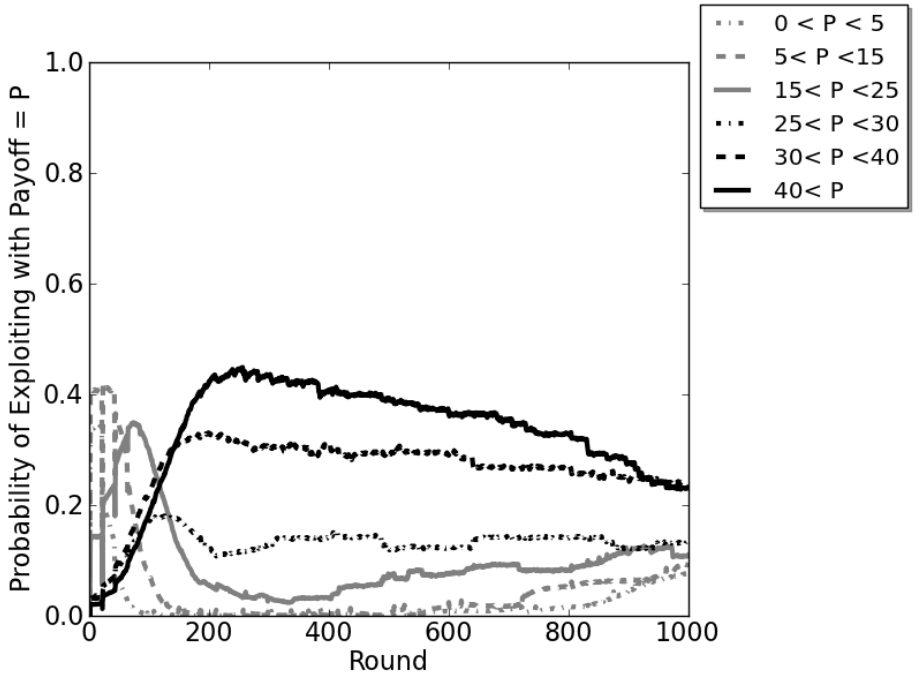
This is the consequence of only innovating as a bootstrapping measure, as *discountmachine* does. Over time, agents will observe actions that are better than the ones they currently know, and begin exploiting them. In environments with low probability of change, the best action in the environment will take a long time to change, which makes it likely that almost all of the agents will be exploiting this action when it does change. This creates a structural shock (see Section 2.2.1), and leaves strategies like *discountmachine* with knowledge of only a very small subset of the actions in the environment, and no way to learn more without innovating. Strategies like *relaxedlookahead*, however, can recover by innovating and finding other actions that have changed to have higher value. This phenomenon appears to be the biggest reason why *relaxedlookahead* has such a large advantage in environments with low probability of change.

These figures also help to explain why, in Table 6.1, *relaxedlookahead* was more likely to win in many environments using π_{Gam}, even though π_{Geo} more closely matches its assumptions about the shape of π. The reason appears to be fairly simple: *relaxedlookahead*'s performance is about the same, while *discountmachine* appears to do worse in the environments using π_{Gam}. Comparing *relaxedlookahead*'s performance as the distribution changes (i.e. comparing Figure 6.6a to 6.8a), we can see that it is more likely to exploit actions between 15 and 25, and less likely to exploit actions with higher value than 25. This is to be expected, however, since π_{Gam} is characterized by a large hump around the mean and a narrower tail than π_{Geo}, so there will be fewer actions with very high values in these environments.

Meanwhile, comparing Figure 6.6b to 6.8b, we can see that *discountmachine* has much more trouble finding actions with above-average payoff in the environment using π_{Gam}, and after round 500 most of its exploits have values between 5 and 15 (recall that the mean of π_{Gam} is 10). This explains why *relaxedlookahead* was able to win against *discountmachine* seven percent more often in the environment used in Figure 6.8.

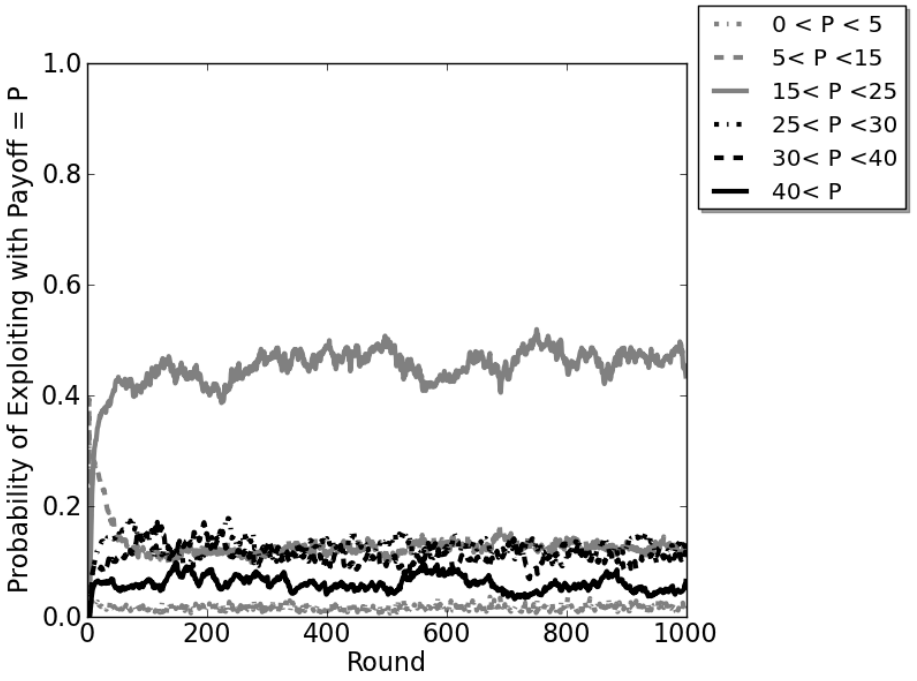


a) *relaxedlookahead*

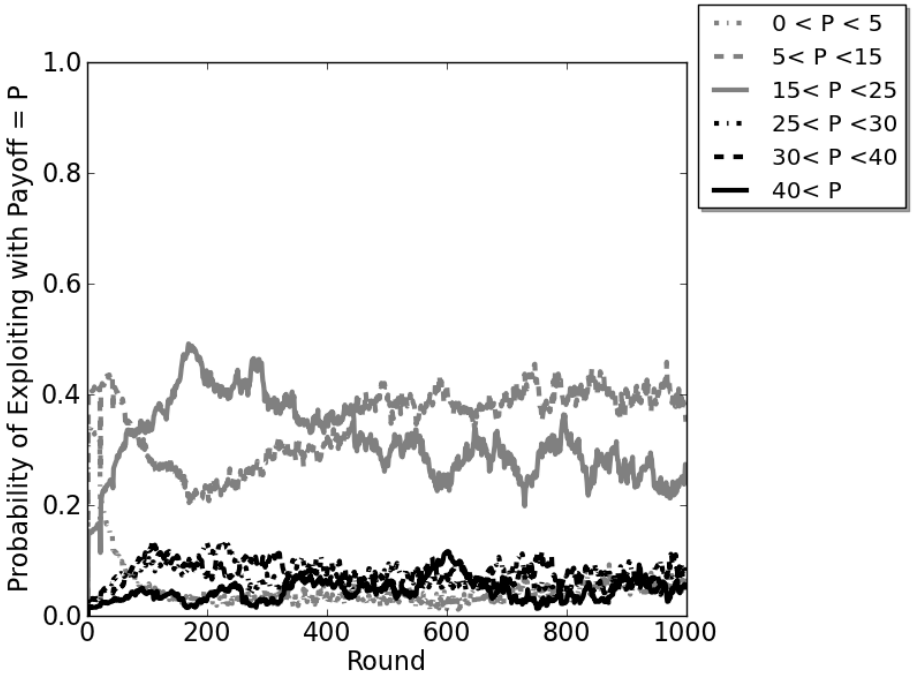


b) *discountmachine*

Figure 6.5: Frequency with which *relaxedlookahead*(top) and *discountmachine*(bottom) exploited an action with payoff P, for six different ranges of values for P, over the first 1000 rounds of a game. Results were obtained by allowing each strategy to play a 50 games against itself, in an environment with action distribution π_{Geo} and probability of change $c = 0.001$, and averaging the exploits observed on each round.

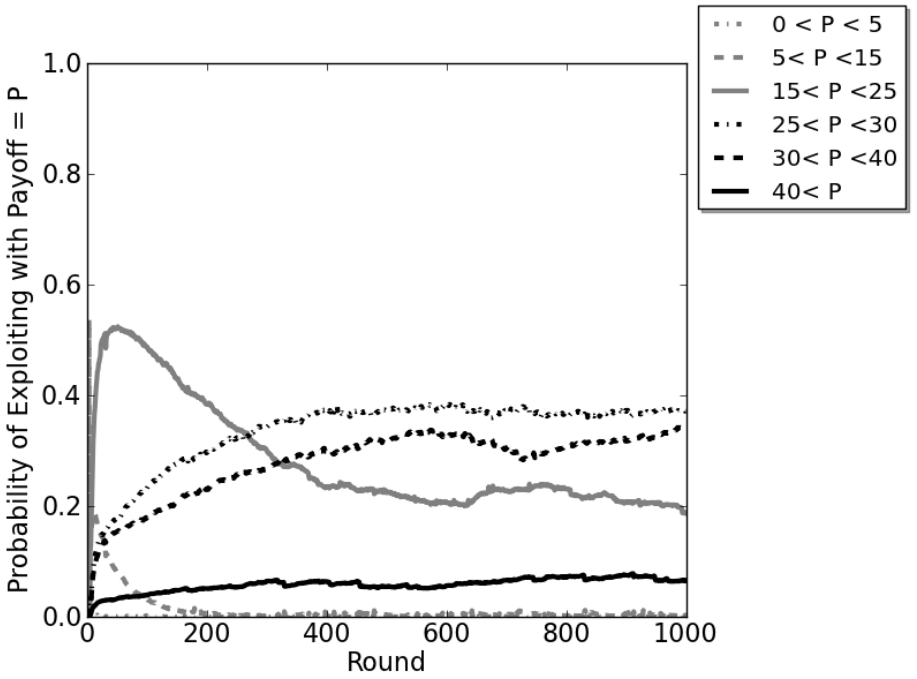


a) *relaxedlookahead*

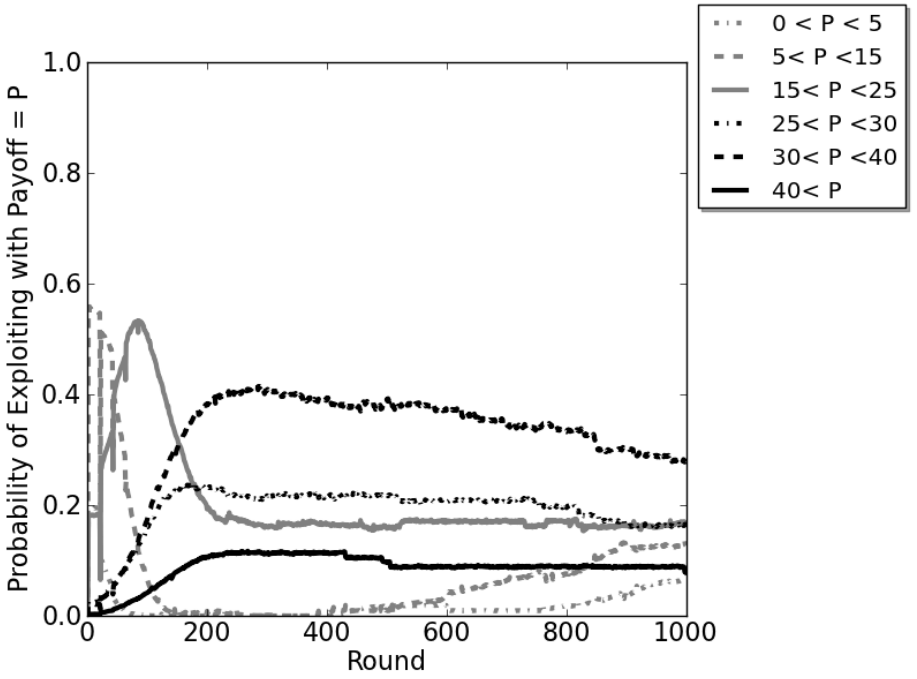


b) *discountmachine*

Figure 6.6: Frequency with which *relaxedlookahead*(top) and *discountmachine*(bottom) exploited an action with payoff P, for six different ranges of values for P, over the first 1000 rounds of a game. Results were obtained by allowing each strategy to play a 50 games against itself, in an environment with action distribution π_{Geo} and probability of change $c = 0.01$, and averaging the exploits observed on each round.

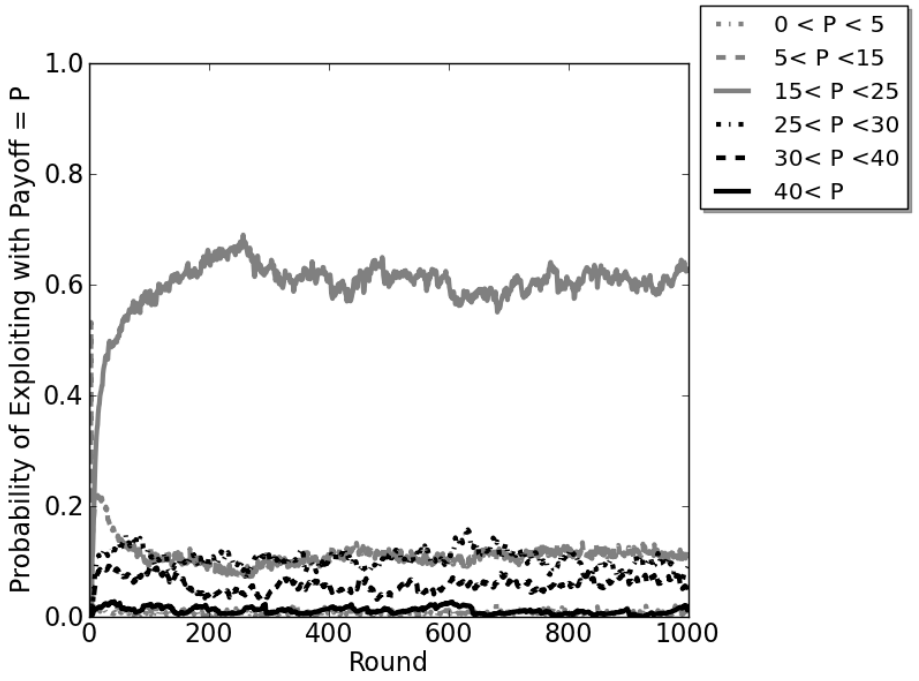


a) *relaxedlookahead*

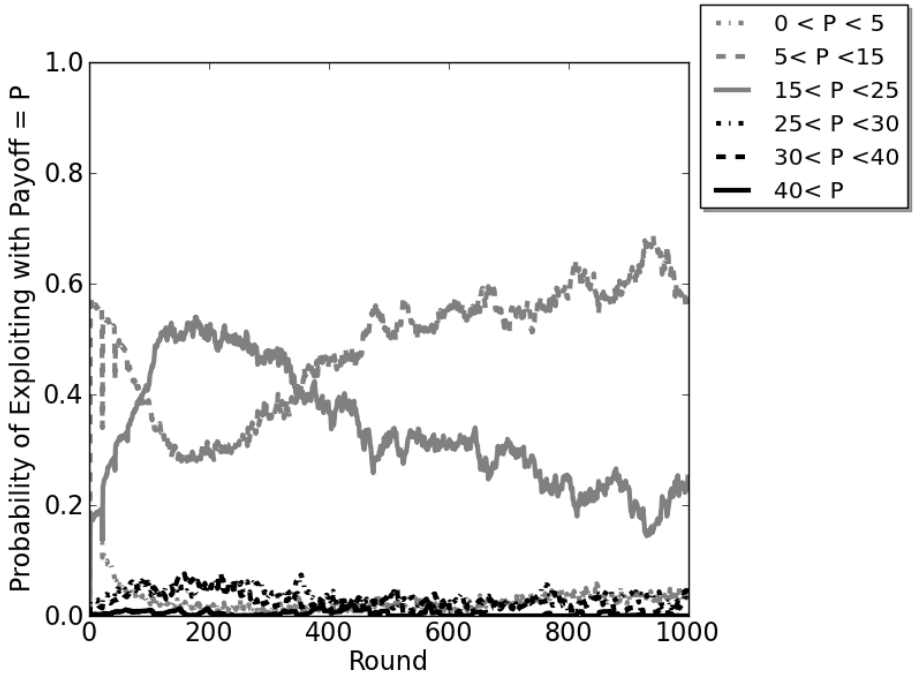


b) *discountmachine*

Figure 6.7: Frequency with which *relaxedlookahead*(top) and *discountmachine*(bottom) exploited an action with payoff P, for six different ranges of values for P, over the first 1000 rounds of a game. Results were obtained by allowing each strategy to play a 50 games against itself, in an environment with action distribution π_{Gam} and probability of change $c = 0.001$, and averaging the exploits observed on each round.



a) *relaxedlookahead*



b) *discountmachine*

Figure 6.8: Frequency with which *relaxedlookahead*(top) and *discountmachine*(bottom) exploited an action with payoff P, for six different ranges of values for P, over the first 1000 rounds of a game. Results were obtained by allowing each strategy to play a 50 games against itself, in an environment with action distribution π_{Gam} and probability of change $c = 0.05$, and averaging the exploits observed on each round.

Chapter 7

Conclusion

This dissertation has presented a variety of techniques for analyzing Cultaptaiton, a complex evolutionary game designed to explore the phenomenon of social learning. Furthermore, the work has demonstrated how to find strategies that are provably good, and how to analyze such strategies to draw conclusions about how good social learning strategies will operate. Thus, this work has advanced the state of the art in two ways.

First, it provides theory and analysis that support many of the empirical results found by the first Cultaptation tournament, and identifies some aspects of the tournament results that are likely due to experimental error (e.g., the lack of individual learning in any of the top-performing strategies). This helps strengthen our understanding of social learning in general, and should help inform future studies of this phenomenon.

Second, it demonstrates techniques that can be used to analyze evolutionary games that are significantly more complex than classical evolutionary games, i.e., games that have a finite number of agents, rather than an infinitely large, well-mixed population; games that last a finite, rather than infinite, number of generations; and games that allow agents to live for multiple generations and condition their actions on accumulated experience, rather than replacing the population every generation

and preventing agents from accumulating experience in the first place. One of the reasons Cultaptation is so complex is that early evolutionary models of social learning made very strong assumptions about social learning mechanics and strategies [16, 7], and the conclusions drawn from studying such models generated a controversial challenge to social learning's role in evolutionary fitness that has taken decades to fully address [7, 13, 14, 18]. As researchers in the field of evolutionary game theory continue to study more complex phenomena like social learning, it is likely that these weaker assumptions will be needed more frequently, and so techniques like the ones presented here will be necessary more often.

In summary, this dissertation has provided the following contributions:

1. Analyzing strategies' reproductive success. Given a Cultaptation game G and a set $\mathbf{S}$ of available strategies for G, the work presents a formula for approximating (to within any $\epsilon > 0$) the expected per-round utility, $\text{EPRU}(s \mid G, \mathbf{S})$, of each strategy in $\mathbf{S}$. The work shows that a strategy with maximal expected per-round utility will have the highest expected frequency in the limit, independent of the initial strategy profile. These results provide a basis for evaluating highly complex strategies such as the ones described below.

Generalizability: These results can be generalized to other evolutionary games in which agents live more than one generation, with a fixed probability of death at each generation, and reproduction is done using the replicator dynamic.

2. Computing near-best-response strategies. The work provides a strategy-generation algorithm that, given a Cultaptation game G and a set of available strategies $\mathbf{S}$, can construct a strategy s_α that is within ϵ of the a response to

S.

Generalizability: The strategy-generation algorithm performs a finite-horizon search, and is generalizable to other evolutionary games in which there is a fixed upper bound on per-round utility and a nonzero lower bound on the probability of death at each round.

3. Approximating symmetric Nash equilibria. The work provides CSLA, an iterative self-improvement algorithm that uses the strategy-generation algorithm in Section 5.1 to attempt to find a strategy s_{self} that is a near-best response in a Cultaptation game in which the other players are all using s_{self}. Hence a strategy profile composed entirely of instances of s_{self} is a symmetric near-Nash equilibrium.

Generalizability: An iterative self-improvement algorithm similar to CSLA should be able to find a near-Nash equilibrium for any game in which the strategies are complex enough that computing a best (or near-best) response is not feasible by analyzing the strategies directly, but is feasible using information from a simulated game between strategies in the profile. Games of this type will typically have a high branching factor but relatively simple interactions between agents.

4. State aggregation. To make its algorithms fast enough for practical experimentation, the work provides a state-aggregation technique that speeds them up by an exponential factor without any loss in accuracy. The experimental results in Section 5.3 demonstrate the practical feasibility that this provides: in these experiments, CSLA always converged in just a few iterations.

Generalizability: The state-aggregation technique is generalizable to other evo-

lutionary games in which the utilities are Markovian.

5. Experimental results. In the experimental studies, the near-Nash equilibria produced by CSLA in any given game were all virtually identical, regardless of the starting values that were used. That strongly suggests (though it does not prove) that the strategy profile consisting of copies of s_{self} approximates an optimal Nash equilibrium, and possibly even a unique Nash equilibrium.

Consequently, s_{self}'s characteristics provide insights into the characteristics of good Cultaptation strategies. For example, the experiments show that s_{self} relies primarily on observation and exploitation, but switches quickly to innovation when a structural shock occurs, switching back to observation and exploitation once it has learned how to respond to the shock. This conflicts with the conventional wisdom [16, 7] that successful social-learning strategies are characterized by a high frequency of innovation, but it helps to explain both the results of the Cultaptation tournament [17] and some recent experimental results on human subjects [18].

6. Improvement on the best tournament strategy. While the algorithms described above are fast enough for experimentation on smaller variants of the Cultaptation game, they would be intractable for use on the more complex variant used in the Cultaptation tournament. The work shows two approaches that can extend the above results to larger environments such as these. First, it shows how the analysis and experimental results outlined above can be used to identify potential problems in the best strategy from the Cultaptation tournament (i.e. *discountmachine*). Second, it uses the formulae from the analysis to define a new strategy, *relaxedlookahead*, that avoids such weaknesses. Experimental results verify

that *relaxedlookahead* is capable of outperforming *discountmachine* in a variety of environments similar to those used in the Cultaptation tournament, and provide an in-depth analysis of the factors that allowed *relaxedlookahead* to perform better.

The analysis showed that refusing to innovate except as a bootstrapping measure (as *discountmachine* and all of the top-performing strategies from the tournament did) makes it much more difficult to recover from structural shocks, especially in environments with low probability of change. Strategies that are willing to innovate when a structural shock is detected, as *relaxedlookahead* is, are able to avoid this problem. The analysis also showed that heuristics instructing a strategy to explore with some minimum frequency, like the one used by *discountmachine*, are unnecessary, since *relaxedlookahead* exhibits emergent behavior in which it explores at intervals dependent upon the parameters of the environment, without being specifically programmed to do so.

One possible avenue for future work would be to identify computationally feasible techniques capable of approximating the Nash equilibrium strategy in these larger versions of Cultaptation, preferably with provable bounds on the difference between the performance of such techniques and the Nash equilibrium strategy. In classical games, a regret minimizing strategy[1] is typically not computationally intensive and has been shown to have performance quite close to that of a Nash strategy in several large classes of repeated games [40]. Therefore, it may be fruitful

[1] in short, a regret minimizing strategy seeks to minimize the expected difference in payoff between the result of its chosen action and the best possible result if it had selected a different action.

to attempt to extend the concept of regret minimization to evolutionary game theory; such an extension would need to account for the conceptual difference between payoff in classical games and fitness in evolutionary games, perhaps involving an idea of "lost fitness minimization," in which the player compares the fitness gained by its chosen action to the highest amount of fitness it would have obtained if it had selected a different action.

7. Implications. These results provide strong support for the following hypotheses about the best strategies for Cultaptation and similar games:

- *What they are like, and how they can be computed.* The best strategies are likely to be conditional ones in which the choice of action at each round is conditioned on the agent's accumulated experience. Such strategies (or close approximations of them) can be computed by doing a lookahead search that predicts how each possible choice of action at the current round is likely to affect future performance.

- *How they are likely to behave.* It is likely that the best strategies will observe and exploit most of the time, but will have ways of quickly detecting structural shocks, so that they can switch quickly to innovation in order to learn how to respond to such shocks.

Appendix A

Proofs

Proposition 2. $\mathrm{EPRU}(s \mid G, \mathbf{S})/d = \mathrm{EPRU}_{\mathrm{alt}}(s, \langle\rangle \mid G, \mathbf{S})$.

Proof.

First, we will show by induction that $\mathrm{EPRU}_{\mathrm{alt}}(s, \langle\rangle \mid G, \mathbf{S})$ equals the summation of $P(h_\alpha|s_\alpha, \mathbf{S})\,\mathrm{EV}_{exp}(|h_\alpha|, U(h_\alpha[|h_\alpha|]))$ for all histories h_α. Then we will show that this equals the summation of $L(|h_\alpha|)P(h_\alpha|s_\alpha, \mathbf{S})\,\mathrm{PRU}(h_\alpha)$ for all h_α.

We will begin with the definition of $\mathrm{EPRU}_{\mathrm{alt}}$ in Equation 4.12, and note that $P(\langle\rangle \circ t|\langle\rangle, s_\alpha(\langle\rangle), \mathbf{S}) = P(\langle\rangle \circ t|s_\alpha, \mathbf{S})$ for histories of length one. This gives us a base case of

$$\mathrm{EPRU}_{\mathrm{alt}}(s_\alpha, \langle\rangle \mid G, \mathbf{S}) =$$

$$\sum_{t \in T} P(\langle\rangle \circ t|s_\alpha, \mathbf{S})\,\mathrm{EV}_{exp}(1, U(t)) + \sum_{t \in T} P(\langle\rangle \circ t|s_\alpha, \mathbf{S})\,\mathrm{EPRU}_{\mathrm{alt}}(s_\alpha, \langle\rangle \circ t \mid G, \mathbf{S}).$$

For the inductive case, we will again start from Equation 4.12, this time noting that $P(h_\alpha|s_\alpha, \mathbf{S})P(h_\alpha \circ t|h_\alpha, s_\alpha(h_\alpha), \mathbf{S})$ simplifies to just $P(h_\alpha \circ t|s_\alpha, \mathbf{S})$. Thus, for all h_α we can rewrite $P(h_\alpha|s_\alpha, \mathbf{S})\,\mathrm{EPRU}_{\mathrm{alt}}(s_\alpha, h_\alpha \mid G, \mathbf{S})$ in terms of histories one round longer than h_α, as follows:

$$P(h_\alpha|s_\alpha, \mathbf{S})\,\mathrm{EPRU}_{\mathrm{alt}}(s_\alpha, h_\alpha \mid G, \mathbf{S}) = \sum_{t \in T} P(h_\alpha \circ t|s_\alpha, \mathbf{S})\,\mathrm{EV}_{exp}(|h_\alpha \circ t|, U(t)) + \sum_{t \in T} P(h_\alpha \circ t|s_\alpha, \mathbf{S})\,\mathrm{EPRU}_{\mathrm{alt}}(s_\alpha, h_\alpha \circ t \mid G, \mathbf{S})).$$

Therefore, by induction we have

$$\mathrm{EPRU}_{\mathrm{alt}}(s_\alpha, \langle\rangle \mid G, \mathbf{S}) = \sum_{h_\alpha \in H} P(h_\alpha | s_\alpha, \mathbf{S}) \, \mathrm{EV}_{exp}(|h_\alpha|, U(h_\alpha[|h_\alpha|])),$$

where H is the set of all possible histories.

The proof then proceeds arithmetically:

$$\begin{aligned}
\mathrm{EPRU}_{\mathrm{alt}}(s_\alpha, \langle\rangle \mid G, \mathbf{S}) &= \sum_{h_\alpha \in H} P(h_\alpha | s_\alpha, \mathbf{S}) \, \mathrm{EV}_{exp}(|h_\alpha|, U(h_\alpha[|h_\alpha|])) \\
&= \sum_{h_\alpha \in H} P(h_\alpha | s_\alpha, \mathbf{S}) \sum_{i=|h_\alpha|}^{\infty} \frac{L(i) U(h_\alpha[|h_\alpha|])}{i} \\
&= \sum_{h_\alpha \in H} \sum_{i=|h_\alpha|}^{\infty} \frac{L(i) P(h_\alpha | s_\alpha, \mathbf{S}) U(h_\alpha[|h_\alpha|])}{i} \\
&= \sum_{i=1}^{\infty} \sum_{h_\alpha \in H(\leq i)} \frac{L(i) P(h_\alpha | s_\alpha, \mathbf{S}) U(h_\alpha[|h_\alpha|])}{i} \\
&= \sum_{i=1}^{\infty} \sum_{h_\alpha \in H(i)} L(i) P(h_\alpha | s_\alpha, \mathbf{S}) \, \mathrm{PRU}(h_\alpha) = \mathrm{EPRU}(s_\alpha \mid G, \mathbf{S})/d
\end{aligned}$$

Where $H(\leq i)$ is the set of all histories of length less than or equal to i, and $H(i)$ is the set of all histories exactly of length i. □

Proposition 3. *Strat*$(h_\alpha, k, V, \mathbf{S})$ *returns* (s_α, U) *such that*

$$\mathrm{EPRU}^k_{\mathrm{alt}}(s_\alpha, h_\alpha \mid G, \mathbf{S}) = U = max_{s'}(\mathrm{EPRU}^k_{\mathrm{alt}}(s', h_\alpha \mid G, \mathbf{S})).$$

Proof. Let s' be a strategy maximizing $\mathrm{EPRU}^k_{\mathrm{alt}}(s', h_\alpha \mid G, \mathbf{S})$ and let $\{s_\alpha, U\}$ be the strategy and value returned by $\mathrm{Strat}(h_\alpha, k, V, \mathbf{S})$. We will show by induction on k that

$$\mathrm{EPRU}^k_{\mathrm{alt}}(s_\alpha, h_\alpha \mid G, \mathbf{S}) = U = \mathrm{EPRU}^k_{\mathrm{alt}}(s', h_\alpha \mid G, \mathbf{S}).$$

In the base case, $k = 0$ and clearly $\mathrm{EPRU}^0_{\mathrm{alt}}(s', h_\alpha \mid G, \mathbf{S}) = 0$ for any s', therefore $\mathrm{EPRU}^k_{\mathrm{alt}}(s_\alpha, h_\alpha \mid G, \mathbf{S}) = \mathrm{EPRU}^k_{\mathrm{alt}}(s', h_\alpha \mid G, \mathbf{S}) = 0 = U$ as required.

For the inductive case, suppose that for k, $\mathrm{Strat}(h_\alpha, k, V, \mathbf{S})$ returns $\{s_\alpha, U\}$ such that $\mathrm{EPRU}^k_{\mathrm{alt}}(s_\alpha, h_\alpha \mid G, \mathbf{S}) = U = max_{s'}(\mathrm{EPRU}^k_{\mathrm{alt}}(s', h_\alpha \mid G, \mathbf{S}))$. We must then show that $\mathrm{Strat}(h_\alpha, k+1, V, \mathbf{S})$ returns $\{s_\alpha, U\}$ such that

$$\mathrm{EPRU}^{k+1}_{\mathrm{alt}}(s_\alpha, h_\alpha \mid G, \mathbf{S}) = U = max_{s'}(\mathrm{EPRU}^{k+1}_{\mathrm{alt}}(s', h_\alpha \mid G, \mathbf{S})).$$

Let s_{temp} be the strategy constructed in lines 8–19 of the algorithm. First we show that on line 20,

$$\mathrm{EPRU}^{k+1}_{\mathrm{alt}}(s_{\mathrm{temp}}, h_\alpha \mid G, \mathbf{S}) = \tag{A.1}$$

$$\sum_{t \in T} P(h_\alpha \circ t | h_\alpha, s_{\mathrm{temp}}(h_\alpha), \mathbf{S}) \left(\mathrm{EV}_{exp}(|h_\alpha|, U(t)) + \mathrm{EPRU}^k_{\mathrm{alt}}(s_{\mathrm{temp}}, h_\alpha \circ t \mid G, \mathbf{S})\right)$$

$$= U_{\mathrm{temp}}.$$

This follows because the t on line 11 iterates over all possible $t \in T$ (due to the `for` loops on lines 6, 9, and 10), meaning that the eventual value of U_{temp} is

$$\sum_{t \in T} P(h_\alpha \circ t | h_\alpha, s_{\mathrm{temp}}(h_\alpha) | \mathbf{S}) \left(\mathrm{EV}_{exp}(|h_\alpha \circ t|, U(t)) + U'\right).$$

By the inductive hypothesis, $U' = \mathrm{EPRU}^k_{\mathrm{alt}}(s_{\mathrm{temp}}, h_\alpha \circ t \mid G, \mathbf{S})$, sufficing to show that (A.1) holds.

Now we show that

$$\mathrm{EPRU}^{k+1}_{\mathrm{alt}}(s_\alpha, h_\alpha \mid G, \mathbf{S}) = \mathrm{EPRU}^{k+1}_{\mathrm{alt}}(s', h_\alpha \mid G, \mathbf{S}).$$

Clearly $\mathrm{EPRU}^{k+1}_{\mathrm{alt}}(s_\alpha, h_\alpha \mid G, \mathbf{S}) \leq \mathrm{EPRU}^{k+1}_{\mathrm{alt}}(s', h_\alpha \mid G, \mathbf{S})$, since s' is assumed to

have maximal $\text{EPRU}_{\text{alt}}^{k+1}$ for h_α, so it suffices to show that

$$\text{EPRU}_{\text{alt}}^{k+1}(s_\alpha, h_\alpha \mid G, \mathbf{S}) \geq \text{EPRU}_{\text{alt}}^{k+1}(s', h_\alpha \mid G, \mathbf{S}).$$

Since s_α maximizes

$$\sum_{t \in T} P(h_\alpha \circ t | h_\alpha, s_\alpha(h_\alpha), \mathbf{S})(\text{EV}_{exp}(|h_\alpha \circ t|, U(t)) + U'),$$

where $U' \geq \text{EPRU}_{\text{alt}}^{k}(s', h_\alpha \circ t \mid G, \mathbf{S})$ by the inductive hypothesis, there can be no action a such that

$$\sum_{t \in T} P(h_\alpha \circ t | h_\alpha, a, \mathbf{S})(\text{EV}_{exp}(|h_\alpha \circ t|, U(t)) + \text{EPRU}_{\text{alt}}^{k}(s', h_\alpha \circ t \mid G, \mathbf{S}))$$

$$> \sum_{t \in T} P(h_\alpha \circ t | h_\alpha, s'(h_\alpha), \mathbf{S})(\text{EV}_{exp}(|h_\alpha \circ t|, U(t)) + U').$$

Therefore $\text{EPRU}_{\text{alt}}^{k+1}(s_\alpha, h_\alpha \mid G, \mathbf{S}) \geq \text{EPRU}_{\text{alt}}^{k+1}(s', h_\alpha \mid G, \mathbf{S})$. This concludes the inductive argument.

Thus for all k, $\text{EPRU}_{\text{alt}}^{k}(s_\alpha, h_\alpha \mid G, \mathbf{S}) = U = \max_{s'} \text{EPRU}_{\text{alt}}^{k}(s', h_\alpha \mid G, \mathbf{S})$. $\square$

Appendix B

Converting from Histories to Repertoires

In this appendix, we will formally define a repertoire, explain how to transform histories into repertoires, and show that the number of possible repertoires is substantially smaller than the number of possible histories, while maintaining the property that any best-response action for a given repertoire is also a best resonse for any history associated with that repertoire (Theorem 5). Finally, we will present a modified version of Algorithm 1, which uses repertoires rather than histories, and we will show how this simple change cuts the branching factor of the algorithm in half (Algorithm 7).

B.1 Repertoire Definition

A repertoire tells the last value and *age* of each action an agent "knows," where an action's age is the number of rounds that have passed since the agent last obtained information about it. Since at any given point in a game, each known action has a unique age, we label exploitation actions by their value and age, leaving off the action number (e.g. if we discovered an action with value 4 last round and an action with value 26 three rounds ago, then the repertoire will be $\{\langle 4, 1\rangle, \langle 26, 3\rangle\}$ where $\langle 4, 1\rangle$ denotes the existence of an action with value 4 discovered 1 round ago, and $\langle 26, 3\rangle$ denotes the existence of an action with value 26 discovered 3 rounds

Algorithm 6 Creates and returns a repertoire for history h_α.

CreateRepertoire($h_\alpha = (a_1, (m_1, v_1)), \ldots, (a_r, (m_r, v_r))$)
  Let $M = \{m_i | i = 1, \ldots, r\}$.
  Let $R = \emptyset$ {R will be the repertoire.}
  for $m \in M$ **do**
    Let $i = \max_{\{1,\ldots,r\}} (m_i = m)$
    Add $\langle v_i, r - i \rangle$ to R.
  end for
  return R

ago). Formally, a repertoire is defined to be a set of pairs, where the first value in each pair represents the knowledge of an action with the given value, while the second value in the pair represents the number of rounds since that knowledge was last updated.

Definition. Let $v_1, \ldots, v_m \in V$ be action values and $\gamma_1, \ldots, \gamma_m \in \mathbb{Z}^+$ (the positive integers) be action ages. A repertoire R is a set of action value/action age pairs $R = \{\langle v_1, \gamma_1 \rangle, \ldots, \langle v_m, \gamma_m \rangle\}$. We denote the set of all repertoires as $\mathcal{R}ep$, and the set of all repertoires where all $\gamma_i \leq j$ as $\mathcal{R}ep_j$. □

$\mathcal{R}ep$ has unbounded size, but $\mathcal{R}ep_j$ has finite size. We show how to create a repertoire R from a history h_α using the CreateRepertoire function in Algorithm 6.

Repertoires change based on the action performed. For example, repertoire

$$R = \{\langle 4, 1 \rangle, \langle 26, 3 \rangle\}$$

can change to repertoire

$$R' = \{\langle 4, 2 \rangle, \langle 26, 4 \rangle, \langle 27, 1 \rangle\}$$

after an innovation action where an action with value 27 is innovated. Notice that all actions in R', apart from the newly-innovated action with age 1, are one round older

than they were in R. This aging process occurs often enough for us to introduce a function which ages a repertoire $R = \{\langle v_i, \gamma_i \rangle\}$:

$$\text{age}(\{\langle v_i, \gamma_i \rangle\}) = \{\langle v_i, \gamma_i + 1 \rangle\}$$

Finally, we will introduce two functions to represent the two ways our repertoire can change when we perform an action.

The first, newaction, returns a repertoire with a new action added to it:

$$\text{newaction}(R, v) = \text{age}(R) \cup \{\langle v, 1 \rangle\}$$

The second, updaction, returns a repertoire with updated information on action m:

$$\text{updaction}(R, v, m) = \text{age}(R \setminus \{\langle v_m, \gamma_m \rangle\}) \cup \{\langle v, 1 \rangle\}$$

B.2 Transition Probabilities

We can now define the probability of transitioning between repertoires on round r. We will call the transition probability functions $P^{Rep}(R'|R, r, a, \mathbf{S})$ for $a \in \{\mathsf{Inv}, \mathsf{Obs}, \mathsf{X}_i\}$. In general, these functions will mirror the $P(h'|h, a, \mathbf{S})$ functions defined in Section 4.1, with some extra clauses added to ensure that if it is not possible to go from repertoire R to repertoire R' using the given action, then the transition probability is 0.

Innovation actions For innovation actions, the function is:

$$P^{Rep}(R'|R, r, \mathsf{Inv}, \mathbf{S}) = \begin{cases} 0 & \text{if } |R| = M \ \vee \ \nexists v : R' = \text{newaction}(R, v) \\ \pi(v) & \text{if } R' = \text{newaction}(R, v) \end{cases}$$

The first clause ensures that if all possible actions are already in R, or if it is not possible to go from R to R' in one innovation action, then the transition probability is 0. The second clause simply tells us the probability of innovating an action with value v, given that it is possible to go from R to R' in one innovation action.

Observation actions In Section 4.1 we assumed the existence of a distribution π_{Obs} that, when given the current history, would tell us the probability of observing an action with a given value. Here, we will make the following assumptions about π_{Obs}:

- When given a repertoire and round number, $\pi_{\mathsf{Obs}}(v|R, r, \mathbf{S})$ tells us the probability of observing an action that has value v and is not already known by R.
- When given a repertoire and round number, $\pi_{\mathsf{Obs}}(m, v|R, r, \mathbf{S})$ tells us the probability of observing an action that has value v, and was previously in R at position m. The value of this action may have changed.
- π_{Obs} can make its predictions without using any information lost when converting from a history to a repertoire.

These assumptions are all satisfied by the π_{Obs} used in our implementation, and we expect them to hold for other practical implementations as well, since a distribution conditioned on entire histories would be impractically large.

With this in mind, the transition probability function for observation actions is

$$P^{Rep}(R'|R,r,\mathsf{Obs},\mathbf{S}) = \begin{cases} \pi_{\mathsf{Obs}}(m,v|R,r,\mathbf{S}) & \text{if } \langle v'_m,\gamma_m\rangle \in R \ \wedge \\ & \quad R' = \text{updaction}(R,v,m) \\ \pi_{\mathsf{Obs}}(v|R,r,\mathbf{S}) & \text{if } R' = \text{newaction}(R,v) \\ 0 & \text{Otherwise.} \end{cases}$$

The first clause gives us the probability of observing an action already in our repertoire, while the second gives us the probability of observing a new action.

Exploitation actions Let $\langle v_i,\gamma_i\rangle$ be the value and age of exploitation action X_i. Then

$$P^{Rep}(R'|R,r,\mathsf{X}_i,\mathbf{S}) = \begin{cases} 0 & \text{if } |R| \neq |R'| \\ 0 & \text{if } \forall v' \in V, R' \neq \text{updaction}(R,v',i) \\ \prod_{j=r-\gamma_i}^{r}(1-c(j)) + \sum_{j=r-\gamma_i}^{r} c(j)\pi(v_i,j)\left(\prod_{i=j}^{r}(1-c(i))\right) & \\ \quad \text{if } R' = \text{updaction}(R,v_i,i) & \\ \sum_{j=r-\gamma_i}^{r} c(j)\pi(v'_i,j)\left(\prod_{i=j}^{r}(1-c(i))\right) & \\ \quad \text{if } R' = \text{updaction}(R,v'_i,i) \text{ and } v_i \neq v'_i \end{cases}$$

The first two clauses check that we can, in fact, transition between R and R' by exploiting. The third clause gives us the probability that the action we exploited has not changed since we last saw it, while the fourth clause gives us the probability that the action we exploited has changed.

B.3 Consistency between P and P^{Rep}

Later in this section, we will show that using a repertoire-based algorithm to compute ϵ-best-response strategies returns the same results as using the history-based Algorithm 1. To do this, we will use the notion of *consistency* between the P and P^{Rep} equations.

Definition. Let M be the set of actions known to an agent with history h_α. The P and P^{Rep} equations are *consistent* for h_α if, for all $a \in \{\mathsf{Inv}, \mathsf{Obs}, \mathsf{X}_i\}$ and $v \in V$:

$$\sum_{m \in M} P(h \circ (a,(m,v)) \mid h, a, \mathbf{S}) = \sum_{m=1}^{|R|} P^{Rep}(\text{updaction}(R,v,m) | R, r, a, \mathbf{S}) \quad \text{(B.1)}$$

and

$$\sum_{m \in \{1,\ldots,M\} \setminus M} P(h \circ (a,(m,v)) \mid h, a, \mathbf{S}) = P^{Rep}(\text{newaction}(R,v) | R, r, a, \mathbf{S}) \quad \text{(B.2)}$$

where $R = \text{CreateRepertoire}(h)$ and $r = |h_\alpha|$. □

Lemma 6. *The P and P^{Rep} equations are consistent for all $h \in H$.*

Proof. We can prove this by using the definition of P, found in Section 4.1, and the definition of P^{Rep} found above. We will simply consider what happens for arbitrary h_α and v when performing innovation, observation, and exploitation actions.

Recall that $X(h)$ returns the number of exploit moves available to an agent with history h. For ease of exposition, we will assume without loss of generality that the first action learned by h_α has label 1, the second has label 2, etc. Thus $M = \{1, \ldots, X(h)\}$, while $\{1, \ldots, M\} \setminus M = \{X(h)+1, \ldots, M\}$.

Innovation actions An innovation action always returns information on a new action, so both sides of Equation B.1 are clearly 0 in this case. If $X(h) = |R| = M$, both sides of Equation B.2 are also 0 since no new actions can be innovated. Thus, we will assume $X(h) = |R| < M$. We now have

$$\sum_{m=X(h)+1}^{M} P(h \circ (\mathsf{Inv}, (m, v))|h, \mathsf{Inv}, \mathbf{S}) = (M - X(h))\frac{\pi(v)}{M - X(h)} = \pi(v)$$

and

$$P^{Rep}(\text{newaction}(R, v)|R, r, \mathsf{Inv}, \mathbf{S}) = \pi(v)$$

which are clearly equivalent. Hence, P and P^{Rep} are consistent on h_α when $a = \mathsf{Inv}$.

Observation actions This section of the proof is mostly trivial, given the assumptions we have made about π_{Obs}. The left side of Equation B.1 is

$$\sum_{m=1}^{X(h)} P(h \circ (\mathsf{Obs}, (m, v))|h, \mathsf{Obs}, \mathbf{S}) = \sum_{m=1}^{X(h)} \pi_{\mathsf{Obs}}(m, v|h, \mathbf{S})$$

which tells us the probability of observing one of the actions already seen in our current history. The right side is

$$\sum_{m=1}^{|R|} P^{Rep}(\text{updaction}(R, v, m)|R, r, \mathsf{Obs}, \mathbf{S}) = \sum_{m=1}^{|R|} \pi_{\mathsf{Obs}}(m, v|R, r, \mathbf{S})$$

Since we assume that $\pi_{\mathsf{Obs}}(m, v|R, r, \mathbf{S})$ tells us the probability of observing action m in the repertoire, $\sum_{m=1}^{|R|} \pi_{\mathsf{Obs}}(m, v|R, r, \mathbf{S})$ also tells us the probability of observing any of the actions we have already seen. Therefore, Equation B.1 holds for observation actions.

For Equation B.2, the left side is

$$\sum_{m=X(h)+1}^{M} P(h \circ (\mathsf{Obs}, (m, v))|h, \mathsf{Obs}, \mathbf{S}) = \sum_{m=X(h)+1}^{M} \pi_{\mathsf{Obs}}(m, v|h, \mathbf{S})$$

which tells us the probability of observing an action we have not yet seen. The right side is

$$P^{Rep}(\text{newaction}(R, v)|R, r, \mathsf{Obs}, \mathbf{S}) = \pi_{\mathsf{Obs}}(v|R, r, \mathbf{S})$$

Since we assume that $\pi_{\mathsf{Obs}}(v|R, r, \mathbf{S})$ gives us the probability of observing a new action, Equation B.2 also holds for observation actions. Hence, P and P^{Rep} are consistent on h_α when $a = \mathsf{Obs}$.

Exploitation actions Exploiting an action never gives us information about a new action, so both sides of Equation B.2 are 0 when we exploit. Thus, we need only consider Equation B.1.

We will consider two cases. In the first case, the action we choose to exploit has changed since we last saw it, so v is a new value. We then have

$$\sum_{m=1}^{X(h)} P(h \circ (\mathsf{X}_m, (m, v))|h, \mathsf{X}_m, \mathbf{S}) = \sum_{m=1}^{X(h)} \sum_{j=last_m}^{r} c(j)\pi(v, j) \prod_{i=j}^{r} 1 - c(i)$$

where $last_m$ is the last round number on which we obtained any information about the m-th action.

Similarly, since exploiting never increases the size of a repertoire, we have

$$\sum_{m=1}^{|R|} P^{Rep}(\text{updaction}(R, v, m)|R, r, \mathsf{X}_m, \mathbf{S}) = \sum_{m=1}^{|R|} \sum_{j=r-\gamma_m}^{r} c(j)\pi(v, j) \prod_{i=j}^{r} (1 - c(i))$$

Since $|R| = X(h)$ and $last_m = r - \gamma_m$ by definition, Equation B.1 holds when the action we exploit changes.

Next, we consider the case where the action we choose to exploit has *not* changed. In this case, we have

$$\sum_{m=1}^{X(h)} P(h \circ (\mathsf{X}_m, (m, v))|h, \mathsf{X}_m, \mathbf{S}) =$$
$$\sum_{m=1}^{X(h)} \left(\prod_{j=last_m}^{r} (1 - c(j)) + \sum_{j=last_m}^{r} c(j)\pi(v, j) \prod_{i=j}^{r} 1 - c(i) \right)$$

and

$$\sum_{m=1}^{|R|} P^{Rep}(\text{updaction}(R, v, m)|R, r, \mathsf{X}_m, \mathbf{S}) =$$
$$\sum_{m=1}^{|R|} \left(\prod_{j=r-\gamma_m}^{r} (1 - c(j)) + \sum_{j=r-\gamma_m}^{r} c(j)\pi(v, j) \prod_{i=j}^{r} (1 - c(i)) \right)$$

which are also equivalent. Therefore, Equation B.1 also holds when the action we exploit does not change.

We have now shown that P and P^{Rep} are consistent on arbitrary h_α when $a \in \{\mathsf{Inv}, \mathsf{Obs}, \mathsf{X}_i\}$ and for arbitrary v. Therefore, P and P^{Rep} are consistent for all h_α. □

B.4 Repertoire-Based Strategies

Repertoires can be used to more compactly define a strategy. We let a *repertoire-based strategy* s be a function from repertoires to actions. Such a strategy can be represented more compactly than the history-based strategies used earlier in this work, since there are fewer possible repertoires than there are possible histories. In any history h_α, a repertoire-based strategy s chooses the action associated with repertoire CreateRepertoire(h).

We can use the P^{Rep} functions to define a formula that determines the EPRU for any repertoire-based strategy s. $\text{EPRU}_{\text{alt}}(s, R, r \mid G, \mathbf{S})$ is a recursive function

for calculating the expected per-round utility of s:

$$\mathrm{EPRU}_{\mathrm{alt}}(s, R, r \mid G, \mathbf{S}) =$$

$$\sum_{a \in A} \sum_{v \in V} [P^{Rep}(\mathrm{newaction}(R, v) | R, r, a, \mathbf{S}) \times$$

$$(\mathrm{EV}_{exp}(r, U((a, (-, v)))) + \mathrm{EPRU}_{\mathrm{alt}}(s, \mathrm{newaction}(R, v), r + 1 \mid G, \mathbf{S}))$$

$$+ \sum_{m=1}^{|R|} P^{Rep}(\mathrm{updaction}(R, v, m) | R, r, a, \mathbf{S}) \times$$

$$(\mathrm{EV}_{exp}(r, U((a, (-, v)))) + \mathrm{EPRU}_{\mathrm{alt}}(s, \mathrm{updaction}(R, v, m), r + 1 \mid G, \mathbf{S}))]$$

where A is the set of possible actions and V is the set of possible action values.

However, like $\mathrm{EPRU}_{\mathrm{alt}}(s, h \mid G, \mathbf{S})$, $\mathrm{EPRU}_{\mathrm{alt}}(s, R, r \mid G, \mathbf{S})$ contains infinite recursion and is therefore not computable. We will deal with this problem as we did in Section 4.2.1, by introducing a depth-limited version. For ease of exposition we will introduce two "helper" functions

$$\mathrm{EPRU}^{k}_{\mathrm{alt}_{\mathrm{new}}}(s, R, r, a, v \mid G, \mathbf{S})$$

$$= P^{Rep}(\mathrm{newaction}(R, v) | R, r, a, \mathbf{S}) \times$$

$$[\mathrm{EV}_{exp}(r, U((a, (-, v)))) + \mathrm{EPRU}^{k-1}_{\mathrm{alt}}(s, \mathrm{newaction}(R, v), r + 1 \mid G, \mathbf{S})]$$

and

$$\mathrm{EPRU}^{k}_{\mathrm{alt}_{\mathrm{upd}}}(s, R, r, a, v \mid G, \mathbf{S})$$

$$= \sum_{m=1}^{|R|} P^{Rep}(\mathrm{updaction}(R, v, m) | R, r, a, \mathbf{S}) \times$$

$$[\mathrm{EV}_{exp}(r, U((a, (-, v)))) + \mathrm{EPRU}^{k-1}_{\mathrm{alt}}(s, \mathrm{updaction}(R, v, m), r + 1 \mid G, \mathbf{S})]$$

Now we can define

$$\mathrm{EPRU}^k_{\mathrm{alt}}(s, R, r \mid G, \mathbf{S}) = \begin{cases} 0, & \text{if } k = 0, \\ \sum_{a \in A} \sum_{v \in V} (\mathrm{EPRU}^k_{\mathrm{alt}_{\mathrm{new}}}(s, R, r, a, v \mid G, \mathbf{S}) + \\ \qquad \mathrm{EPRU}^k_{\mathrm{alt}_{\mathrm{upd}}}(s, R, r, a, v \mid G, \mathbf{S})), & \text{otherwise.} \end{cases}$$

A proof that this formulation is equivalent to the version of $\mathrm{EPRU}^k_{\mathrm{alt}}$ from Section 4.2.1 follows.

Theorem 5. *For all histories h_α, all repertoire-based strategies s, and all $k \geq 0$, if s' is a function from histories to actions where $s'(h) = s(CreateRepertoire(h))$ and $r = |h_\alpha|$, then* $\mathrm{EPRU}^k_{\mathrm{alt}}(s, R, r \mid G, \mathbf{S}) = \mathrm{EPRU}^k_{\mathrm{alt}}(s', h \mid G, \mathbf{S})$.

Proof. We can prove this by using induction on k. For our base case, we will use $k = 0$, since $\mathrm{EPRU}^0_{\mathrm{alt}}(s, R, r \mid G, \mathbf{S}) = \mathrm{EPRU}^0_{\mathrm{alt}}(s', h \mid G, \mathbf{S}) = 0$ by definition.

For the inductive step, we will assume that Theorem 5 holds for some $k \geq 0$, and show that it also holds for $k + 1$. Recall from Section 4.2.1 that, in this case

$$\mathrm{EPRU}^{k+1}_{\mathrm{alt}}(s', h \mid G, \mathbf{S}) = \sum_{t \in T} P(h \circ t | h, s(h), \mathbf{S})(\mathrm{EV}_{exp}(r, U(t)) + \mathrm{EPRU}^k_{\mathrm{alt}}(s', h \circ t \mid G, \mathbf{S}))$$

Since T is simply the set of all action-percept pairs, we can instead write this as

$$\mathrm{EPRU}^{k+1}_{\mathrm{alt}}(s', h \mid G, \mathbf{S}) =$$
$$\sum_{a \in A} \sum_{v \in V} [\sum_{m=1}^{M} P(h \circ (a, (m, v)) | h, s(h), \mathbf{S})$$
$$(\mathrm{EV}_{exp}(r, U((a, (m, v)))) + \mathrm{EPRU}^k_{\mathrm{alt}}(s', h \circ (a, (m, v)) \mid G, \mathbf{S}))]$$

where A is the set of possible actions and V the set of possible values. We also have[1]

$$\begin{aligned}
&\mathrm{EPRU}^{k+1}_{\mathrm{alt}}(s, R, r \mid G, \mathbf{S}) = \\
&\sum_{a \in A} \sum_{v \in V} [P^{Rep}(\mathrm{newaction}(R, v) | R, r, a, \mathbf{S}) \\
&\qquad (\mathrm{EV}_{exp}(r, U((a, (-, v)))) + \mathrm{EPRU}^{k}_{\mathrm{alt}}(s, \mathrm{newaction}(R, v), r + 1 \mid G, \mathbf{S})) \\
&\qquad + \sum_{m=1}^{|R|} P^{Rep}(\mathrm{updaction}(R, v, m) | R, r, a, \mathbf{S}) \\
&\qquad (\mathrm{EV}_{exp}(r, U((a, (-, v)))) + \mathrm{EPRU}^{k}_{\mathrm{alt}}(s, \mathrm{updaction}(R, v, m), r + 1 \mid G, \mathbf{S}))]
\end{aligned}$$

Recall from Lemma 6 that P and P^{Rep} are *consistent* on h_α. We can combine this with our inductive hypothesis to show that the bracketed portions of the two equations above are equal.

Recall that when a repertoire encounters a new action, it does not store the action number m for that action. Thus, for any pair m and m' that are not already in h_α, we know that CreateRepertoire$(h \circ (a, (m, v))) =$ CreateRepertoire$(h \circ (a, (m', v))) =$ newaction(R, v). Therefore, by our inductive hypothesis

$$\mathrm{EPRU}^{k}_{\mathrm{alt}}(s', h \circ (a, (m, v)) \mid G, \mathbf{S}) = \mathrm{EPRU}^{k}_{\mathrm{alt}}(s, \mathrm{newaction}(R, v), r \mid G, \mathbf{S})$$

for all m not already in h_α. Notice that, by definition, there are $M - X(h)$ values of m that are not already in h_α. If we assume without loss of generality that the first action learned in h_α has label 1, the second has label 2, etc., then we can define β_{new}

[1]Recall that function U simply calculates the utility of performing the given action, and does not depend on the action number. Thus $U(a, m, v) = U(a, -, v)$ for any legal m.

to be the quantity P and P^{Rep} are multiplied by when we learn something new:

$$\begin{aligned}\beta_{new} &= \mathrm{EV}_{exp}(r, U((a,(-,v)))) + \mathrm{EPRU}^k_{\mathrm{alt}}(s', h \circ \langle a, X(h)+1, v\rangle \mid G, \mathbf{S}) \\ &= \mathrm{EV}_{exp}(r, U((a,(-,v)))) + \mathrm{EPRU}^k_{\mathrm{alt}}(s', h \circ \langle a, X(h)+2, v\rangle \mid G, \mathbf{S}) \\ &\ldots \\ &= \mathrm{EV}_{exp}(r, U((a,(-,v)))) + \mathrm{EPRU}^k_{\mathrm{alt}}(s', h \circ (a,(M,v)) \mid G, \mathbf{S}) \\ &= \mathrm{EV}_{exp}(r, U((a,(-,v)))) + \mathrm{EPRU}^k_{\mathrm{alt}}(s, \mathrm{newaction}(R,v), r \mid G, \mathbf{S})\end{aligned}$$

Similarly, the inductive hypothesis also tells us that

$$\mathrm{EPRU}^k_{\mathrm{alt}}(s', h \circ (a,(m,v)) \mid G, \mathbf{S}) = \mathrm{EPRU}^k_{\mathrm{alt}}(s, \mathrm{updaction}(R,v,m), r \mid G, \mathbf{S})$$

for all m that are already in h_α. Thus, we can also define β_m to be the quantity P and P^{Rep} are multiplied by when we update our information on action m:

$$\begin{aligned}\beta_m &= \mathrm{EV}_{exp}(r, U((a,(-,v)))) + \mathrm{EPRU}^k_{\mathrm{alt}}(s', h \circ (a,(m,v)) \mid G, \mathbf{S}) \\ &= \mathrm{EV}_{exp}(r, U((a,(-,v)))) + \mathrm{EPRU}^k_{\mathrm{alt}}(s, \mathrm{updaction}(R,v,m), r \mid G, \mathbf{S})\end{aligned}$$

for $m = 1, \ldots, X(h)$.

We can now rewrite $\mathrm{EPRU}^{k+1}_{\mathrm{alt}}(s', h \mid G, \mathbf{S})$ and $\mathrm{EPRU}^{k+1}_{\mathrm{alt}}(s, R, r \mid G, \mathbf{S})$ as

$$\begin{aligned}\mathrm{EPRU}^{k+1}_{\mathrm{alt}}(s', h \mid G, \mathbf{S}) = \sum_{a \in A}\sum_{v \in V}[&\sum_{m=X(h)+1}^{M} P(h \circ (a,(m,v))|h, s(h), \mathbf{S})\beta_{new} \\ &+ \sum_{m=1}^{X(h)} P(h \circ (a,(m,v))|h, s(h), \mathbf{S})\beta_m]\end{aligned}$$

and

$$\begin{aligned}\mathrm{EPRU}^{k+1}_{\mathrm{alt}}(s, R, r \mid G, \mathbf{S}) = \sum_{a \in A}\sum_{v \in V}[&P^{Rep}(\mathrm{newaction}(R,v)|R, r, a, \mathbf{S})\beta_{new} \\ &+ \sum_{m=1}^{|R|} P^{Rep}(\mathrm{updaction}(R,v,m)|R, r, a, \mathbf{S})\beta_m]\end{aligned}$$

Equations B.1 and B.2 tell us that regardless of the values of a and v,

$$\sum_{m=X(h)+1}^{M} P(h \circ (a,(m,v))|h, s(h), \mathbf{S}) = P^{Rep}(\text{newaction}(R,v)|R,r,a,\mathbf{S})$$

and

$$\sum_{m=1}^{X(h)} P(h \circ (a,(m,v))|h, s(h), \mathbf{S}) = \sum_{m=1}^{|R|} P^{Rep}(\text{updaction}(R,v,m)|R,r,a,\mathbf{S})$$

Therefore, $\text{EPRU}^{k+1}_{\text{alt}}(s', h \mid G, \mathbf{S}) = \text{EPRU}^{k+1}_{\text{alt}}(s, R, r \mid G, \mathbf{S})$. This completes the induction. □

B.5 Repertoire-Based Algorithm

Now that we have a formula for computing the EPRU of a repertoire-based strategy, and we know that using repertoires rather than histories to calculate EPRU gives us the same results, we can update our algorithm to use repertoires. The new algorithm will be almost identical to Algorithm 1, except that using repertoires rather than histories will allow us to reduce our number of recursive calls by half. Let R'_{Obs} be the set of repertoires for which $P^{Rep}(R'_{\mathsf{Obs}}|R,r,\mathsf{Obs}) > 0$, and define R'_{Inv} and R'_{X_i} similarly. Note that, if R contains m different actions,

$$R'_{\mathsf{Obs}} \subseteq R'_{\mathsf{Inv}} \cup \bigcup_{i=1}^{m} R'_{\mathsf{X}_i} \tag{B.3}$$

In other words, since repertoires do not need to remember what actions our agent performed, choosing X_1 produces the same repertoire as choosing Obs and observing action 1. Similarly, choosing Inv and Obs can also produce the same repertoires, if both actions happen to tell us about the same action. However, there

is no action we could encounter through observing that we could not encounter through either innovating or exploiting. Therefore, if we save the results of the recursive calls to calculate the utility of Inv and $\mathsf{X}_1, \ldots, \mathsf{X}_m$, we can compute the utility of Obs without any additional recursion. This cuts the branching factor of our algorithm in half, from $(2m + 2)v$ to $(m + 1)v$, which reduces the size of the search tree by a factor of 2^k for search depth k, without any impact on accuracy. Algorithm 7 is the complete algorithm.

Algorithm 7 Produce strategy s that maximizes $\mathrm{EPRU}^{k}_{\mathrm{alt}}(s, R \mid G, \mathbf{S})$, given initial repertoire R, and set of possible utility values V.

Strat(R,r, k,V,$\mathbf{S}$)

1: **if** $k = 0$ **then**
2: **return** 0
3: **end if**
4: Let $U_{\max} = 0$
5: Let $s_{\max} = null$
6: Let $U_{\mathsf{Obs}} = 0$
7: **for** each action $a \in \{\mathsf{X}_1, \cdots, \mathsf{X}_M, \mathsf{Inv}\}$ **do**
8: Let $U_{\mathrm{temp}} = 0$
9: Let $s_{\mathrm{temp}} = null$
10: **for** each value $v \in V$ **do**
11: Let $t = \langle v, 1 \rangle$
12: **if** $\exists i : (a = \mathsf{X}_i)$ **then** Let $R' = \mathrm{age}(R \setminus \{\langle v_i, \gamma_i \rangle\}) \cup t$ and $p = P^{Rep}(R' | R, r, \mathsf{X}_i, \mathbf{S})$
13: **else** Let $R' = \mathrm{age}(R) \cup t$ and $p = P^{Rep}(R' | R, r, \mathsf{Inv}, \mathbf{S})$
14: Let $p_{\mathsf{Obs}} = P^{Rep}(R' | R, r, \mathsf{Obs}, \mathbf{S})$
15: **if** $p + p_{\mathsf{Obs}} > 0$ **then**
16: Let $\{S', U'\} = \mathrm{Strat}(R', r+1, k-1, V, \mathbf{S})$
17: **if** $p_{\mathsf{Obs}} > 0$ **then** $U_{\mathsf{Obs}} = U_{\mathsf{Obs}} + p_{\mathsf{Obs}} \cdot U'$
18: **if** $p > 0$ **then**
19: $s_{\mathrm{temp}} = s_{\mathrm{temp}} \cup S'$
20: **if** $\exists i : (a = \mathsf{X}_i)$ **then** $U_{\mathrm{temp}} = U_{\mathrm{temp}} + p \cdot (\mathrm{EV}_{exp}(r, v) + U')$
21: **else** $U_{\mathrm{temp}} = U_{\mathrm{temp}} + p \cdot U'$
22: **end if**
23: **end if**
24: **end for**
25: **if** $U_{\mathrm{temp}} > U_{\max}$ **then**
26: $U_{\max} = U_{\mathrm{temp}}$
27: $s_{\max} = s_{\mathrm{temp}} \cup \langle R, a \rangle$
28: **end if**
29: **end for**
30: **if** $U_{\mathsf{Obs}} > U_{\max}$ **then**
31: $U_{\max} = U_{\mathsf{Obs}}$
32: $s_{\max} = s_{\mathrm{temp}} \cup \langle R, \mathsf{Obs} \rangle$
33: **end if**
34: **return** $\{s_{\max}, U_{\max}\}$

Appendix C

The Number of Pure Strategies for Cultaptation

In Cultaptation as defined on the tournament web site, each game includes 10,000 rounds, 100 agents, 100 exploitation actions, and the actions Inv and Obs. Let S be the set of all pure Cultaptation strategies, and S' be the set of all strategies such that the first 100 moves are Inv, and all subsequent moves are exploitation actions. Then any lower bound on S' is a loose lower bound on S.

Suppose an agent uses a strategy in S'. If it survives for the first 100 rounds of the game, it will learn values for all 100 of the exploitation actions. There are 100! different orders in which these actions may be learned, and for each action there are 100 possible values; hence there are 100^{100} possible combinations of values. Thus after 100 Inv moves, the number of possible histories is $100! \times 100^{100}$. All subsequent moves by the agent will be exploitations; and it is possible (though quite unlikely!) that the agent may live for the remaining $9,900$ rounds of the game. Thus each of the above histories is the root of a game tree of height $2 \times 9,900$. In this game tree, each node of even depth is a choice node (each branch emanating from the node corresponds to one of the 100 possible exploitation actions), and each node of odd depth is a value node (each branch emanating from the node corresponds to one of the 100 different values that the chosen action may return). Since there are

$100! \times 100^{100}$ of these game trees, the total number of choice nodes is

$$100! * 100^{100} \sum_{d=0}^{9899} (100^2)^d > 9.3 \times 10^{39953}.$$

If we use the conventional game-theoretic definition that a pure strategy s must include a choice of action at each choice node, regardless of whether the choice node is reachable given s, then it follows that

$$|S'| > 100^{9.3 \times 10^{39953}}.$$

If we use the definition used by game-tree-search researchers, in which a pure strategy only includes a choice of action at each choice node that is *reachable* given s, then the number of reachable choice nodes given s is

$$100! * 100^{100} \sum_{d=0}^{9899} 100^d > 9.4 \times 10^{20155},$$

so

$$|S'| > 100^{9.4 \times 10^{20155}}.$$

Bibliography

[1] A. Bandura. *Social Learning Theory*. General Learning Press, New York, 1977.

[2] B.G. Galef and K.N. Laland. Social learning in animals: Empirical studies and theoretical models. *Bioscience*, 55:489–499, 2005.

[3] T.R. Zentall. Imitation: Definitions, evidence, and mechanisms. *Animal Cognition*, 9:335–353, 2006.

[4] L. G. Rapaport and G. R. Brown. Social influences on foraging behavior in young nonhuman primates: Learning what, where, and how to eat. *Evolutionary Anthropology*, 17(4):189–201, 2008.

[5] G. Duffy, T. Pike, and K.. Laland. Size-dependent directed social learning in nine-spined sticklebacks. *Animal Behaviour*, 78(2):371–375, August 2009.

[6] J. Kendal, L. Rendell, T. Pike, and K. Laland. Nine-spined sticklebacks deploy a hill-climbing social learning strategy. *Behavioral Ecology*, 20(2):238–244, 2009.

[7] R. Boyd and P.J. Richerson. Why does culture increase human adaptability? *Ethology and Sociobiology*, 16(2):125–143, 1995.

[8] K.N. Laland. Social learning strategies. *Learning and Behavior*, 32:4–14, 2004.

[9] C.J. Barnard and R. M. Sibly. Producers and scroungers: A general model and its application to captive flocks of house sparrows. *Animal Behavior*, 29:543–550, 1981.

[10] D. Nettle. Language: Costs and benefits of a specialised system for social information transmission. In J. Wells and et al., editors, *Social Information Transmission and Human Biology*, pages 137–152. Taylor and Francis, London, 2006.

[11] L. A. Giraldeau, T. J. Valone, and J. J. Templeton. Potential disadvantages of using socially acquired information. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, 357(1427):1559–1566, 2002.

[12] J Noble, P M Todd, and E Tuci. Explaining social learning of food preferences without aversions: an evolutionary simulation model of norway rats. *Proc. Biol Sci.*, 268(1463):141–149, January 2001.

[13] T. Kameda and D. Nakanishi. Cost–benefit analysis of social/cultural learning in a nonstationary uncertain environment: An evolutionary simulation and an experiment with human subjects. *Evolution and Human Behavior*, 23:373–393, September 2002.

[14] R. McElreath, M. Lubell, P. Richerson, T. Waring, W. Baum, E. Edsten, C. Efferson, and B. Paciotti. Applying evolutionary models to the laboratory study of social learning. *Evolution and Human Behavior*, 26(6):483–508, November 2005.

[15] R. Boyd, M. Enquist, K. Eriksson, M. Feldman, and K. Laland. Cultaptation: Social learning tournament, 2008. `http://www.intercult.su.se/cultaptation`.

[16] A. R. Rogers. Does biology constrain culture? *American Anthropologist*, 90(4):819–831, 1988.

[17] L. Rendell, R. Boyd, D. Cownden, M. Enquist, K. Eriksson, M. W. Feldman, L. Fogarty, S. Ghirlanda, T. Lillicrap, and K. N. Laland. Why copy others? insights from the social learning strategies tournament. *Science*, 328(5975):208–213, April 2010.

[18] T. Wisdom and R. Goldstone. Social learning and cumulative innovations in a networked group. In *Advances in Social Computing: Third International Conference on Social Computing, Behavioral Modeling, and Prediction, SBP 2010*, pages 32–41, 2010.

[19] R. Carr, E. Raboin, A. Parker, and D. Nau. Theoretical and experimental analysis of an evolutionary social-learning game. *Technical Reports from UMIACS, UMIACS-TR-2012-05*, 2012.

[20] J. C. Villaneuva. Atoms in the universe. *Universe Today*, July 2009. `http://www.universetoday.com/36302/atoms-in-the-universe/`.

[21] D. Friedman. On economic applications of evolutionary game theory. *Journal of Evolutionary Economics*, 8(1):15–43, 1998.

[22] E. Guse. Expectational business cycles. Money Macro and Finance (MMF) Research Group Conference 2004 97, Money Macro and Finance Research Group, September 2004.

[23] E. Raboin, R. Carr, A. Parker, and D. Nau. Balancing innovation and exploitation in a social learning game. In *AAAI Fall Symposium on Adaptive Agents in Cultural Contexts*, November 2008.

[24] K. Leyton-Brown and Y. Shoham. *Essentials of Game Theory: A Concise Multidisciplinary Introduction*. Morgan & Claypool, 2008.

[25] R. Carr, E. Raboin, A. Parker, and D. Nau. When innovation matters: An analysis of innovation in a social learning game. In *Second International Conference on Computational Cultural Dynamics (ICCCD)*, September 2008.

[26] J. Henrich and R. McElreath. The evolution of cultural evolution. *Evolutionary Anthropology*, 12:123–135, 2003.

[27] M. Enquist, S. Ghirlanda, and K. Eriksson. Critical social learning: A solution to rogers's paradox of nonadaptive culture. *American Anthropologist*, 109(4):727–734, 2007.

[28] C. Papadimitriou and J. Tsitsiklis. The complexity of optimal queuing network control. *Mathematics of Operations Research*, 24(2):293–305, May 1999.

[29] S. Guha, K. Munagala, and P. Shi. Approximation algorithms for restless bandit problems. In *SODA '09: Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 28–37, Philadelphia, PA, USA, 2009. Society for Industrial and Applied Mathematics.

[30] K.H. Schlag. Why imitate, and if so, how?, : A boundedly rational approach to multi-armed bandits. *Journal of Economic Theory*, 78:130–156, 1998.

[31] D.E. Koulouriotis and A. Xanthopoulos. Reinforcement learning and evolutionary algorithms for non-stationary multi-armed bandit problems. *Applied Mathematics and Computation*, 196(2):913 – 922, 2008.

[32] M. Kearns, Y. Mansour, and A. Y. Ng. A Sparse Sampling Algorithm for Near-Optimal Planning in Large Markov Decision Processes. *MACHINE LEARNING*, 49:193–208, 2002.

[33] D. Koller and R. Parr. Computing factored value functions for policies in structured MDPs. In *International Joint Conference on Artificial Intelligence*, volume 16, pages 1332–1339. Citeseer, 1999.

[34] C. Boutilier, R. Dearden, and M. Goldszmidt. Exploiting structure in policy construction. In *International Joint Conference on Artificial Intelligence*, volume 14, pages 1104–1113. LAWRENCE ERLBAUM ASSOCIATES LTD, 1995.

[35] R.A. Howard. *Dynamic programming and Markov process*. MIT Press, 1960.

[36] J. Hofbauer and K. Sigmund. Evolutionary game dynamics. *Bulletin of the American Mathematical Society*, 40(4):479, 2003.

[37] A. Plaat, J. Schaeffer, W. Pijls, and A. de Bruin. An algorithm faster than negascout and sss* in practice. In *Computer Strategy Game Programming Workshop*. 1995.

[38] T. Ibaraki. Theoretical comparision of search strategies in branch and bound. *International Journal of Computer and Information Sciences*, 5:315 344, 1976.

[39] E. Horowitz and S. Sahni. *Fundamentals of Computer Algorithms*. Computer Science Press, Potomac, MD, 1978.

[40] A. Blum, M.T. Hajiaghayi, K. Ligett, and A. Roth. Regret minimization and the price of total anarchy. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 373–382. ACM, 2008.