Cooperation in Games

A DISSERTATION

SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL

OF THE UNIVERSITY OF MINNESOTA

BY

Steven Bjorn Damer

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

Doctor of Philosophy

Advisor: Professor Maria Gini

May, 2019

© Steven Bjorn Damer 2019
ALL RIGHTS RESERVED

Acknowledgements

First, I would like to thank my advisor, Maria Gini. She has been consistently encouraging throughout my academic career. Her guidance has been a great aid to me in my struggles with writing and publishing my research, and she has been patient with my topic changes when I've been disappointed with the results of my research. This dissertation would not exist without her support.

Her research group, which I still think of as Outobox, has been a valuable forum for me to present my research and be exposed to the research of others. I would particularly like to thank Paul Schrater, Arindam Banerjee, Wolf Ketter, Brett Borghetti, Timothy Woods, Elizabeth Jensen, Mohammed Elidrisi, William Groves, Julio Godoy, Ernesto Nunes, Marie Manner, Mark Valovage, Libby Ferland, and John Harwell for their friendship and feedback. I would also like to thank them for tolerating my own feedback, which was often more concerned with adversarial considerations than it needed to be.

I am grateful to Jeff Rosenschein for the comments and encouragement he has given me with the Gift Exchange game. My discussions with him have helped me to clarify my thoughts and organize my research.

Finally, I would like to thank my friends and family for their support and encouragement. They helped keep me sane by reminding me of the world outside academia, and giving me a reason to go there.

I would also like to acknowledge the Minnesota Supercomputing Institute (MSI) at the University of Minnesota for providing computational resources to perform simulated annealing on the Gift Exchange game. It is a fantastic resource which I wish I had learned about earlier.

Dedication

To my parents, whose love and support has been invaluable to me.

Thank you for raising a wonderful family to grow up in.

Abstract

This dissertation explores several problems related to social behavior, which is a complex and difficult problem. In this dissertation we describe ways to solve problems for agents interacting with opponents, specifically (1) identifying cooperative strategies, (2) acting on fallible predictions, and (3) determining how much to compromise with the opponent.

In a multi-agent environment an agent's interactions with its opponent can significantly affect its performance. However, it is not always possible for the agent to fully model the behavior of the opponent and compute a best response.

We present three algorithms for agents to use when interacting with an opponent too complex to be modelled. An agent which wishes to cooperate with its opponent must first identify what strategy constitutes a cooperative action. We address the problem of identifying cooperative strategies in repeated randomly generated games by modelling an agent's intentions with a real number, its attitude, which is used to produce a modified game; the Nash equilibria of the modified game implement the strategies described by the intentions used to generate the modified game. We demonstrate how these values can be learned, and show how they can be used to achieve cooperation through reciprocation in repeated randomly generated normal form games.

Next, an agent which has formed a prediction of opponent behavior which may be incorrect needs to be able to take advantage of that prediction without adopting a strategy which is overly vulnerable to exploitation. We have developed Restricted Stackelberg Response with Safety (RSRS), an algorithm which can produce a strategy to respond to a prediction while balancing the priorities of performance against the

prediction, worst-case performance, and performance against a best-responding opponent. By balancing those concerns appropriately the agent can perform well against an opponent which it cannot reliably predict.

Finally we look at how an agent can manipulate an opponent to choose actions which benefit the agent. This problem is often complicated by the difficulty of analyzing the game the agent is playing. To address this issue, we begin by developing a new game, the Gift Exchange game, which is trivial to analyze; the only question is how the opponent will react. We develop a variety of strategies the agent can use when playing the game, and explore how the best strategy is affected by the agent's discount factor and prior over opponents.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Cooperation is not possible in every environment, but when it is possible, cooperation provides a benefit to the agent. Cooperation is acting in conjunction with another agent to mutually improve the scores of both agents. If the other agent is simple and easily predictable, an agent can improve its score by taking advantage of the predictability of the other agent but this is not cooperation.

Cooperation is specially useful when an agent cannot predict the other agent. It is tempting, when confronted by an opponent which the agent is unable to predict, to attempt to improve the prediction algorithm, but this is not always possible, due to limitations in computational resources. We are interested in environments where an agent is interacting with its peers, not its inferiors. Generally it is not reasonable to assume that an agent can predict the behavior of another agent which is just as complicated as the agent itself.

One game which is frequently used to illustrate problems of cooperation is the prisoner's dilemma game [6]. In the prisoner's dilemma, two criminals have been arrested and are being separately questioned by police. They each have the option to implicate their partner or remain silent. If they both implicate their partner they each go to

prison for 5 years. If one implicates their partner and the other doesn't, the one which remained silent goes to prison for 10 years, while the other goes free. If neither implicates their partner, the police will find some other charges to pin on them, and they will each go to prison for 1 year. This is commonly expressed in terms of a normal form game in which the options are cooperate (with the opponent) or defect. The payoffs are shown later in Table 2.1. In a single game of prisoner's dilemma the Nash equilibrium is for both players to defect. When multiple games are played against the same opponent the Nash equilibrium is still to defect (by induction) but an intriguing possibility opens up; players can use the threat of retaliation to enforce cooperation.

Competitions have been performed in which programs play repeated prisoner's dilemma against each other. In the first such competition [6] the most successful strategy was Tit-for-Tat, which cooperates in the first round and copies the opponent action from the previous round in all other rounds. It is very effective in inducing cooperation, but it can only be applied to symmetric games where there is a cooperative move.

1.1 Attitude

The first contribution of this dissertation [19, 20, 21, 22] is a procedure to identify cooperative actions in arbitrary normal form games. Our procedure operates in an environment consisting of a sequence of randomly generated general-sum normal form games. The games must be general-sum to allow for the possibility of cooperative actions. We have chosen to look at randomly generated games because we want our method to generalize to any game. We generate a sequence of games because we do not want our method of cooperation to be contingent on having the same set of actions always available to the agents. In this environment the only strategic considerations for the agent are the details of the current game, and considerations of how the opponent will respond in future games.

Identifying cooperative actions in this type of game is difficult. It is tempting to focus on the results of the opponent's action and if the agent is doing better than expected determine that the opponent is cooperating. This doesn't work because actions can have unintended consequences; the opponent doesn't know what the agent was going to do, so the outcome of the opponent's action is not a reliable indicator of the opponent's intentions.

Our procedure describes the opponent's intentions as a real number called the *attitude* of the opponent. The attitude of an agent describes the relative importance it attaches to its opponent's payoff. An attitude of 0 indicates an indifferent agent. An attitude of 1 indicates an agent which attaches the same importance to the abstract.bblopponent's payoff as its own. Attitudes can be negative; an attitude of -1 indicates a strictly competitive opponent which only cares about the difference between its payoff and its opponent's payoff. Such an agent treats general-sum games as a zero-sum contest, which can be very inefficient.

Given an assignment of attitude values to the player and their opponent we can identify strategies for a given normal form game by creating a modified game and calculating the Nash equilibrium. When these strategies are played in the given game, the outcome reflects the attitude values used to generate them. The primary influence on an agent's payoff is whether or not the opponent has a positive attitude towards them. There is also a smaller effect in which an agent's payoff is slightly increased by adopting an attitude close to 0.

When agents adopt strategies using attitude values we can use a particle filter to detect the attitude values used by the agent. We can create a reciprocating strategy by learning the attitude values used by the opponent, and playing a strategy generated from attitude values in which the agent adopts a slightly higher attitude than the opponent. Agents which reciprocate like this can rapidly arrive at a cooperative outcome with a

reciprocating opponent while avoiding exploitation by an opponent which is unwilling to reciprocate.

1.2 Restricted Stackelberg Response with Safety

abstract.bbl We have developed a strategy capable of reciprocating in a randomly generated game, but our method of reciprocation has a counterintuitive element. We use a particle filter to estimate the attitude values used by the opponent, and so we have a prediction of opponent behavior. However, when our agent responds to the prediction, they respond to the attitude of the opponent, and not the probability distribution over actions which they expect the opponent to adopt. Their action is optimized to signal a cooperative attitude to the opponent instead of being optimized to respond in a cooperative way to the opponent. We have found this approach effective; it does successfully arrive at cooperative outcomes. But it seems wasteful to form a prediction of opponent behavior and then discard it.

On the other hand, simply best-responding to the prediction is also a bad idea. An agent which plays a best-response to a prediction can be extremely vulnerable when the prediction is incorrect. The second main contribution of this dissertation is a method of finding a strategy to respond to a prediction in a general sum game while guarding against a best-responding opponent and providing worst-case guarantees.

We have developed the Restricted Stackelberg Response with Safety (RSRS) [23], a method of calculating a response to a prediction while guarding against the possibility that the prediction may be incorrect. The RSRS uses two parameters which describe different ways of expressing confidence in the prediction. The first parameter is the prediction weight which expresses the probability that the prediction is correct. The second parameter is the risk factor which is the amount the agent is willing to lose if the opponents move is the worst possible move. The RSRS for a given prediction, prediction

weight p, and risk factor r is a probability distribution over actions that maximizes the agent's payoff against an opponent which plays according to the prediction with probability p and best responds to the agent's payoff with probability $1 - p$ subject to the constraint that the agent's payoff against any action of the opponent is within r of the amount the agent can guarantee for itself.

RSRS provides a basis for choosing how to respond to a prediction of opponent behavior but it requires that the agent provide a prediction and choose parameter values. We can use a loss-limiting procedure described in [53] to choose a value for risk factor in a way that guarantees that the average payoff of the agent will be at least the value of the game. For the prediction weight we have developed a procedure to estimate the relative probabilities of a best-responding opponent and an opponent following the prediction; we can use those values to select an appropriate value for the prabstract.bblediction weight parameter. In this work we use a deliberately poor algorithm to generate the prediction: fictitious play. Fictitious play assumes that the opponent is playing according to a static probability distribution and returns the probability distribution which maximizes the probability of the observed actions of the opponent. There are better prediction algorithms available for normal form games, but the point of using RSRS is to perform well when the prediction is inaccurate. Using these methods of parameter selection allows RSRS to perform well even with a relatively simple method of prediction.

Another issue raised by the use of attitude is the question of how much to reciprocate. We have shown that agents which adopt a fixed reciprocating strategy can achieve cooperation, and they arrive at cooperative outcomes more rapidly as they increase their reciprocation factor (how much nicer than the opponent they try to be). However, it is not clear that this is always a desirable strategy. One simple way to take advantage of it is for the opponent to ignore the agent's attempts at cooperation; the opponent

will do slightly better than it would against an indifferent agent at no cost to itself. Another way to attempt to take advantage of an agent using attitude to cooperate is for the opponent to deceive the agent by identifying strategies which benefit the opponent while still appearing to be cooperative, or which appear to be heavily cooperative while not costing the opponent in a particular game.

In general, the question of when and how much to reciprocate is a complicated one, and it is further complicated by the difficulty of discerning the intent of the opponent in a randomly generated normal form game.

1.3 Gift Exchange Game

The third main contribution is the *Gift Exchange* game, a game we have developed to study the problem of how an agent should deal with a self-interested opponent in a general-sum game [25, 24].

When an agent is dealing with an opponent, there are a number of factors which make it difficult to choose a course of action. First, the effects of a particular choice may not be apparent, either due to hidden information which is unavailable to the agent, or due to the effects of a simultaneous action chosen by the opponent. This makes it difficult to determine how to select an appropriate action to further the agenda of the agent and difficult to determine the intent behind the opponent's choice of action. Second, the value of a particular game state may be unknown. In the absence of hidden information or simultaneous moves it may still be the case that the game is too complex for the agent to fully analyze, leading to a situation where the agent knows the exact game state which will result from its choice, but is uncertain about the value of that game state. This also makes it difficult to determine the intent behind the opponent's choice of action. Finally, the agent may be uncertain about the opponent's response to various courses of action, such as what compromises the opponent would

consider acceptable and how would the opponent react if the agent doesn't adopt an acceptable compromise. It is this problem which is relevant to the decision of how much to reciprocate. In repeated normal form games, attempting to address this problem is complicated by the first problem (in more complicated games, the second problem is also relevant).

Attitude provides one way to address the third problem. An agent can use attitude to elicit cooperative behavior from other agents which also use attitude, but this is made more complicated by the uncertainty created by simultaneous moves. RSRS provides a way to address the first problem by identifying strategies which limit the potential outcomes in terms of loss to the opponent and gain to the agent. We want to focus on the third problem without the complications created by simultaneous moves and limitations on achievable outcomes.

We deliberately constructed the Gift Exchange game to make the first two problems trivial; developing strategies for the Gift Exchange game is solely concerned with the third problem: how will the opponent react to the agent's actions, and how can the agent choose actions that trigger a beneficial response from the opponent. In the Gift Exchange game, it is desirable for agents to achieve a pareto-optimal outcome; we will informally refer to such an outcome as 'cooperative' in the sense that agents working together will always achieve a pareto-optimal outcome. Note that pareto-optimality does not necessarily imply that the outcome is fair; an outcome in which one agent receives much more than the other is pareto optimal as long as the other agent can't receive more without reducing the payoff of the first agent.

In the Gift Exchange game the players take turns choosing outcomes from a fixed set. Each outcome consists of a payoff (possibly negative) to each player. The effect of each choice is public knowledge, and there is no persistent game state so the only concern an agent has when selecting outcomes is the immediate payoff of the outcome

to the agent and the effect of choosing that outcome on the future behavior of the opponent.

It is impossible to define a strategy to manipulate the opponent without making some assumptions about the opponent, and the assumptions made determine whether a pareto-optimal outcome is reached, and which one is reached. For example, the assumptions behind the Nash equilibrium do not allow for a pareto-optimal outcome in finitely repeated games (by induction) and in infinitely repeated games nearly any sequence of choices (including but not limited to pareto-optimal outcomes) can be supported as a Nash equilibrium (by the Folk theorem). These are not satisfying assumptions because we want agents to be able to reach a pareto-optimal outcome.

In our work on the Gift Exchange game we begin by looking at how to play against simple opponents. A simple opponent chooses its action based only on the agent's previous move, so to find the best response to a simple opponent the agent only needs to identify which choice of outcome gives the agent the highest combined payoff from its choice and the opponent's choice. We have developed several algorithms based on the UCT algorithm which are able to learn the optimal response to a simple opponent in the Gift Exchange game.

Next we consider the implications of opponents which learn simple models. The best response to such an opponent is an agent which follows a simple model for which the best response to the agent is highly favorable for the agent. This can be accomplished by a strategy in which the agent designates some acceptable choice which provides both the agent and the opponent an amount greater than they could guarantee for themselves. If the opponent makes the acceptable choice the agent will make the same choice, but if the opponent doesn't do so the agent will make a punishing choice. The best response to such a strategy is for the opponent to make the acceptable choice, but the agent can choose an acceptable choice which is biased towards the agent.

In infinitely repeated Gift Exchange games the optimal simple strategy to play against a learning opponent can give the agent a payoff arbitrarily close it its maximum payoff. In finitely repeated games, or in games where the agent has a discount factor, the optimal response to a learning opponent is less clear because the closer the agent's payoff is to its maximum payoff the longer it will take the opponent to learn the best response.

We used simulated annealing to find the best simple strategy to use against a particular learning opponent and discount factor. The more the agent discounts the future the further the payoff of the optimal strategy gets from the agent's maximum possible payoff. The more rapidly the opponent learns, the closer the optimal strategy gets to the agent's maximum possible payoff.

If the opponent follows a simple model, the best strategy for the agent is to use a learning strategy, but if the opponent is using a learning strategy the best strategy for the agent is to use a simple strategy. Furthermore, in self-play both types of strategy perform poorly.

Consider what the agent should do if the opponent is either following a simple strategy or a learning strategy, but the agent doesn't know which one. The best outcome for the agent is if the opponent is using a learning strategy, in which case the agent can adopt a greedy simple strategy and achieve a good outcome. So when the agent is uncertain it should initially adopt a greedy simple strategy. If the opponent learns a best response to that, the agent doesn't need to change its strategy, and if the opponent doesn't learn a best response, the agent can assume that the opponent is following a simple strategy and adopt a learning strategy. We can continue this process - if the agent believes the opponent will eventually switch to a learning strategy the appropriate response is for the agent to play a greedy simple strategy until it is certain the opponent will not switch.

We can characterize strategies of this type as sequential immediately reactive ratio enforcing strategies, which are defined by a function which determines the appropriate simple strategy to adopt as a function of the amount of time the opponent has been playing a greedy strategy. We distinguish between a learning opponent and an opponent playing a greedy strategy by observing how close the opponent gets to the optimal payoff against the simple strategy adopted by the agent.

We have performed simulated annealing to find the optimal strategy to adopt against different distributions of opponents. The strategies we have found generally indicate that agents with higher discount factors compromise more quickly, agents which expect opponents to learn slowly compromise more quickly, and agents which expect their opponent to compromise slowly compromise more quickly.

1.4 Conclusions

This dissertation is concerned with the problem of interacting with an opponent which is a peer. The opponent is a peer in the sense that, although it acts purposefully, the agent is not able to fully model and predict how the opponent will respond. Our general goal is to find a way to cooperate with the opponent by using reciprocation.

The first problem we looked at is how to cooperate with an opponent in the absence of any indication of which actions constitute cooperation. We developed a method to accomplish this in repeated randomly generated normal form games by generating a modified game where the payoffs have been altered to reflect the intentions of the players and playing the Nash equilibrium of the modified game.

The second problem we looked at is how to respond to a prediction of opponent behavior in a general-sum normal form game when that prediction is untrustworthy. We developed a method to accomplish this by making the assumption that the opponent will best-respond to the agent's action when the prediction is incorrect and finding the

strategy that maximizes the agent's payoff in those circumstances.

The final problem we looked at is how much to reciprocate – when should the agent accept a compromise offered by the opponent and when should the agent hold out for more? We have developed a new game, the Gift Exchange game, which is deliberately simplified to focus on the problem of getting the opponent to react in a beneficial way. We have looked at several different ways of playing the Gift Exchange game and used simulated annealing to show how the appropriate strategy for cooperating is dependent on the agent's discount factor and prior beliefs about the opponent.

Chapter 2

Related Work

2.1 Game Theory

We are interested in cooperation – the ability of two agents working together to increase their collective payoff even when their interests do not align perfectly. Cooperation is a valuable ability for any agent acting in any environment where it is possible, and cooperation is frequently possible. Cooperation has been extensively researched in normal form games.

One way to explain cooperation in normal form games is that agents have preferences over the utilities of other agents; agents cooperate because they want other agents to be happy. There are many plausible ways an agent's utility could depend on the opponent's utility which are outlined in [72]. However such a definition of an agent's utility is unstable [33]; if an agent's utility is dependent on its opponent's utility, and its opponent's utility is dependent on the agent's utility, then the agent's utility cannot be uniquely determined without some simplifying functions. A better approach is to split an agent's utility into two parts: the enjoyment they get from consuming goods and the vicarious enjoyment they get from observing the opponent consuming goods.

Note that cooperation in this setting does not necessarily entail a reduction in conflict between the agent and the opponent [31]. Depending on the possible outcomes and the degree to which agents care about their opponents it is possible for agents to compete to force their opponent to accept a greater share of the potential payoff.

In Chapter 3 we discuss how to use the concept of an agent which cares about its opponent's payoff in a linear manner to find cooperative strategies in normal form games. We refer to the amount an agent weights the payoff of its opponent's payoff by as the agent's *attitude*. Variations on this model have been used to try to explain the behavior of people in many different environments. The attitude of a person can be estimated by observing how they play a decomposed game (a game in which one player chooses between 2 outcomes which assign payoffs to both players) [39]. However, it has been found that in this situation the attitude that people exhibit is strongly dependent on the experimental instructions. In the ultimatum and centipede games [51] people will not always play the Nash equilibrium. However, it is possible to find a distribution over attitudes which explain the experimental results. It is possible to extend the notion of attitude to create a model in which agents want opponents which are helpful to do better and opponents which are hostile to do worse [61] instead of unilaterally preferring that the opponent do better or worse regardless of the opponent's attitude. This can be used to explain experimental results in Cournot duopoly games and used car markets [65]. Another possible variant is inequity aversion [30] in which players dislike outcomes in which one player receives significantly more than the other. Depending on the game and the distribution of preferences this can result in cooperative or non-cooperative outcomes, which matches the observations of many experiments. In our system we do not use these variations; the attitude of an agent is treated as a strict preference. To avoid the problem of an agent trying to help an opponent which is trying to hurt it we allow for attitudes to change depending on the behavior of the opponent.

It would seem that adopting a positive attitude towards the opponent would have negative consequences for the agent, but many examples of reciprocation can be observed in the behavior of animals [71] as well as humans. This can be justified on the basis of evolutionary success. There are many different ways reciprocity can help an agent including kin selection, direct reciprocity, indirect reciprocity, network reciprocity, and group selection [57].

Note that attitude is not the only way to explain cooperation in humans, and it is not actually necessary to explain cooperation to be able to cooperate. Instead of using a formal model it is possible to identify associations between how a person plays in one game with cooperative opportunities and how they play in another game [1]. This can be used to increase performance without ever designating a particular action as cooperative or not.

2.2 Agents in Normal Form Games

Many algorithms have been created to play games or operate in environments where cooperation is possible. Such algorithms need to solve a wide variety of problems including analyzing the environment, learning a best response to the opponent, improving social welfare, and avoiding exploitation. Here are some algorithms that we have found interesting.

Fictitious play [34] is a learning algorithm for normal form games with the simplest possible opponent model: it assumes that the opponent is playing a fixed distribution. Fictitious play tracks the number of times the opponent has played each move, and best responds to the assumption that future moves will be drawn from the observed distribution. The problem of crafting a strategy to play against opponents that best respond to an infinite (fictitious play) or bounded history is studied in [26]. Such an opponent induces a Markov decision problem on the underlying game. The authors

describe how to use the best-response properties of the opponent to find a solution to the Markov decision problem. This idea is similar to strategies we propose in Chapter 5, but depends on manipulating observed probabilities instead of observed rewards.

GIGA-WolF [9] learns a best response to the opponent by gradually adjusting its current strategy to better respond to the opponent's actions. It follows the Win-Or-Learn-Fast principle [10] which changes more rapidly when the agent is performing poorly to allow it to spend more time in the more rewarding parts of the search space. GIGA-WolF is guaranteed to converge to a Nash equilibrium in self-play in a two-action game, and it achieves zero average regret. In other words it is guaranteed to converge to the best response to the empirical distribution of play observed from the opponent. GIGA-WolF is not able to cooperate effectively in environments where cooperation is possible, but its capabilities provide a good baseline of desirable capabilities for an agent.

AWESOME [16] is able to improve on the guarantees of GIGA-WolF. It analyzes the opponent's behavior to distinguish between between a stationary distribution, an equilibrium strategy, or some unknown strategy. It plays a best response to a stationary distribution, and plays an equilibrium strategy otherwise. It guarantees a best-response against a stationary player, and can reach a Nash equilibrium in any game in self-play.

TPCM(A) [60] forms a more sophisticated prediction of opponent behavior. In addition to playing a best response to a stationary distribution and an equilibrium in self play, it will detect whether an opponent can be trained using Godfather [52], if it is willing to cooperate to achieve a pareto-efficient outcome, or if it plays a fixed strategy conditional on the last k turns. It is optimal against its target set, pareto-efficient in self-play, and guaranteed to achieve its safety value. CMLeS [13] provides similar guarantees against a larger set of target opponents. It models the opponent as playing a strategy conditional on some number of previous turns, up to a fixed maximum. It

uses the Nash equilibrium as a fallback position against opponents it can't predict.

If the opponent is playing a finite automaton in a repeated normal form game, the agent generally doesn't know the characteristics of the automaton. It is possible to learn a stationary strategy for the opponent using reinforcement learning, however the opponent may not stick to a stationary strategy. This problem can be handled using the R-MAX# algorithm [43] which can detect when the opponent has deviated from the learned strategy and learn to best respond to the new strategy. The strategies we have developed for the Gift Exchange game are intended to take advantage of strategies like R-MAX#[11]; it is easy to find the best response, but the best response to the strategy is beneficial for the agent.

An agent which forms a prediction of opponent behavior must take into account the possibility of that prediction being incorrect. One way to handle this problem in constant-sum games is to use Restricted Nash Response, which estimates a probability that the prediction is incorrect and creates a modified game in which the opponent plays according to the prediction with that probability, and otherwise can play without restriction [44]. The Nash equilibrium of the modified game has been shown to ameliorate risk in two-player limit Texas hold-em. Another approach is to use Safe Policy Selection, which chooses an amount the agent is willing to risk and plays the best response to the prediction subject to the constraint that the worst possible expected payoff to the agent is worse than the value of the game by no more than the amount the agent is willing to risk [53].

In a repeated game an agent can adjust the amount it is willing to risk in a way that guarantees that the average payoff of the agent will approach the payoff of the best response if the opponent is predictable, while also guaranteeing that the agent receives the value of the game in the event that the opponent is not predictable. It is also possible to use Risk What You Won in Expectation to guarantee that the agent's

expected payoff is never lower than the value of the game [37].

RWYWE plays an equilibrium strategy by default, but if the opponent plays a non-equilibrium strategy which gives the agent a higher payoff, it will adjust its strategy to respond better to the opponent's apparent strategy while guaranteeing an expected loss no greater than the already realized expected gain. Wang et al. [74] describe an algorithm similar to Restricted Nash Response which plays a Nash equilibrium in a modified game which constrains the opponent to play a strategy similar to the observed strategy, retreating to minimax if the opponent doesn't play predictably.

In large games it may be computationally intractable to calculate a complex response to a prediction of opponent behavior. DBBR [36] handles this by modifying the prediction to more closely match a pre-calculated equilibrium and then best-responding to the modified prediction. This allows for some resistance to exploitation in environments where it is impossible to calculate a strategy which explicitly guards against exploitation.

Instead of treating the problem of choosing a strategy in repeated normal form games as a sequence of decisions, in [75] the problem is framed as one player selecting a finite automaton to play for them, and the other player selecting an automaton in response. The size of the automatons can be used to represent the bounded rationality of the players. The paper describes how to compute an optimal automaton for the first player to commit to. This is similar (but more complex) to the strategies we have developed to play our Gift Exchange game. We handle the problem of selecting an optimal strategy by introducing a discount factor, while they handle it by introducing limits on the complexity of the automatons.

2.3 Cooperation

The prisoner's dilemma is a normal form game which describes a situation in which the agent and the opponent have an opportunity to cooperate, increasing both their payoffs, but if one player cooperates while the other doesn't, the one that didn't attempt to cooperate gains more than they would've if they had cooperated. A common payoff structure for prisoner's dilemma can be seen in Table 2.1.

	Cooperate	Defect
Cooperate	3 \ 3	0 \ 5
Defect	5 \ 0	1 \ 1

Table 2.1: Common payoffs for prisoner's dilemma

The social welfare maximizing outcome is for both players to cooperate, but the Nash equilibrium is for neither player to cooperate. When people play iterated prisoner's dilemma, they will often cooperate even though induction shows that the Nash equilibrium is still not to cooperate.

It is tempting to dismiss this behavior as irrational, but when a competition of iterated prisoner's dilemma was held [6] the most successful strategy was Tit-for-Tat. An agent playing Tit-for-Tat cooperates initially and copies the opponent's previous action on every subsequent action. Tit-for-Tat will never outscore the opponent in game of iterated prisoner's dilemma, but it was the most successful strategy at convincing the opponent to cooperate.

Other variations on Tit-for-Tat are possible. For example, the strategy Tit-for-Two-Tats is a more forgiving version of Tit-for-Tat; it will cooperate unless the opponent has defected for two rounds in a row. This approach can be generalized by using randomizing strategies to confine the outcome of the game to a bounded objective region [41]. This approach can be successful in a tournament, and also show good performance against a

reinforcement learner.

Reinforcement learning can be used in iterated prisoner's dilemma [67], with game histories being used as states for the reinforcement learner. Complete cooperation rarely occurs but, depending on the parameters, complicated patterns of cooperation and defection can be seen.

Tit-for-Tat does not perform well in noisy environments because it will never forgive an opponent for a betrayal which was caused by noise. On the other hand a forgiving strategy such as Tit-for-Two-Tats is vulnerable to exploitation by an opponent which takes advantage of the forgiveness to defect occasionally. One way to handle that [3] is to learn the opponents policy by deducing deterministic rules from the discounted observed frequencies of the opponents play. Noise is handled by flagging deviations from the policy, and learning a new model if the deviations become too frequent.

The ideas behind Tit-for-Tat can be generalized to other situations. For example, when agents have opportunities to ask other agents for help in performing specialized tasks, agents can achieve efficient cooperation by basing their decision to help on the expected future benefits of helping [64]. The presence of such strategies can increase social welfare.

Cooperation in a single interaction is difficult to justify without reciprocation; the decision to cooperate in a single interaction is more of an expression of the agent's preferences than a strategic decision. To design an agent which is capable of cooperating in a single interaction it is useful to look at focal point theory [49] which discusses methods of developing coordination between two agents which cannot communicate.

A modification of reinforcement learning is described in [32]; the agent updates its policy to optimize performance against a learning opponent instead of a static opponent. The authors show that agents using this update rule are capable of learning to cooperate in the Repeated Prisoner's Dilemma. Furthermore, they show that when the

technique is applied again (i.e., the agent optimizes its policy under the assumption that the opponent is updating *its* policy under the assumption that the agent is a naive reinforabstract.bblcement learner), it does not result in additional gains. In our work we do not explicitly model the learning that agents do as they converge to a cooperative outcome, but [32] suggests that it could be a successful approach.

Cooperation is generally thought of as an interaction between agents with orthogonal goals; agents which could cooperate, but are not required to cooperate. It is also interesting to consider the problem of how to interact with an opponent that shares the goals of the agent, but has not been designed to coordinate with the agent. If the opponent is able to learn by observing the agent, the agent must decide how to balance acting to further its own goals versus acting to teach the opponent a better strategy. This problem is considered in two different contexts [70]: a repeated normal-form game (with both players sharing the same utility function) in which the agent must find the most effective teaching sequence to optimize performance against a bounded memory best-responding opponent, and a shared multi-armed bandit problem where the agent must decide when to pay a cost to demonstrate the optimal choice to the opponent.

As environments become more complex it becomes more difficult to identify cooperative strategies. To study cooperation in Markov games, in [50] a pair of Markov games are presented that have opportunities for cooperation, and a Deep Q-Network is used to learn strategies for those games. By varying parameters of the game, different strategies are learned, which can be designated *cooperate* or *defect* according to the performance of the strategy in self-play. By evaluating the cooperating and defecting strategies against one another, they observe how the structure of the social dilemma of the underlying game varies according to the parameters of the game. The strategies developed implement cooperation or competition in the underlying Markov game, but

they do not attempt to reciprocate. This work approaches the problem of cooperation from the other end—instead of starting from a simple environment and looking at how to decide when to cooperate, they start from a complex environment and look at learning how to cooperate in the first place.

2.4 Different Games

A stochastic game [68] is a repeated set of games between players where the payoffs of each game are determined by the current state, and the outcome of each game affects the subsequent state. A number of approaches have been developed to learn stochastic games (e.g., [69]), but they focus on achieving the best individual payoff and not on cooperation, and they require that the environment consists of a limited number of states. Stochastic games represent a midpoint between repeated play of a single game and our environment, where a game is never seen twice. Algorithms using reinforcement learning for stochastic games only need to know the current state and the payoff received in the previous state. In repeated randomly generated normal form games agents need to know their opponent's payoffs because they do not have the ability to observe the opponent's prior play for the current game; the opponent's payoffs are the only information they can use to try to predict their play.

An algorithm for repeated stochastic games, presented in [28], uses lossy game abstraction [66] to reduce the state space of the game and facilitate learning and adapting rapidly to a non-stationary opponent. They infer opponent intent in stochastic games by clustering paths of play according to that player's payoffs. This allows them to identify desirable coordinated strategies which they can attempt to teach the opponent to use.

The algorithm in [18] reduces the problem of playing repeated stochastic games to a multi-armed bandit problem by generating a handful of expert strategies to use. This

approach simplifies the underlying game to make it a matter of selecting the appropriate expert strategy. The agent selects a strategy with the intent that the opponent will play its part in the selected strategy, and enforces compliance by punishing the opponent when it fails to comply.

The Gift Exchange game has similarities with the Dictator game [40]. Both games involve the active player choosing an outcome that cannot be affected by the opponent; however, the Gift Exchange Game involves an element of social dilemma that is not usually present in the Dictator game, and the Dictator game is not usually a repeated game. In the Dictator game, one player is given an initial endowment, which they may then choose to divide with the other player in any proportion they choose (including the option of keeping the entire endowment for themselves). The game-theoretic analysis of the Dictator game is trivial—the dictator should keep the entire endowment. People playing the Dictator game do not generally play the Nash equilibrium, and the game has been extensively studied to explore the factors that influence how people actually play it. The Dictator game is generally played as a single shot game, but one study has explored the effects on the second player when two games are played in a row with players swapping roles [27]. In that situation the second player is more likely to return the gift received from the first player. Other factors that have been studied include demographics, cultural factors, the value of the endowment, social distance between the players, and how deserving the recipient is. Attention has generally focused on measuring how these factors affect human play [29]. In contrast, our focus in the Gift Exchange Game is on exploring reciprocation as a basis for non-equilibrium play.

The Gift Exchange game also shares some similarities with the Nash Bargaining game [63, 56]. In the Nash bargaining game players take turns proposing outcomes, and an outcome to the game is selected when an offer is accepted. Agents prefer to reach an agreement earlier because they either discount the future or pay a fixed bargaining cost

each round. The only pareto-optimal outcomes to the Gift Exchange game occur when both players select the same action. We can view the progression of a Gift Exchange game as a sequence of offers that resolves when both players take the same action for the remainder of the game. The difference between the Gift Exchange game and the Nash bargaining game is that in the Gift Exchange game the cost paid for not coordinating is dependent on the offers proposed, instead of being a fixed or exponentially decreasing cost.

The Gift Exchange game is similar to Stackelberg games [73][15] in that players act sequentially instead of simultaneously. In Stackelberg games the first player can gain an advantage by committing to a strategy before the second player acts. This is possible because the first player's commitment can restrict the outcomes available to the second player so that the second players best choice benefits the first player. In the Gift Exchange game this dynamic is modified; the effects of players' actions are not dependent on the opponent's action, so the first player doesn't have an advantage. However, either player can gain a Stackelberg-like advantage by acting in a predictable way which encourages the opponent to play a preferred action (see Section 5.4). In Stackelberg games it is possible to handle the problem of an opponent which does not strictly best-respond to the leader's strategy [59]. They explore 3 different approaches, one based on a follower which responds sub-optimally, one based on a follower which responds to an incorrect prediction of the leaders behavior, and one based on maximizing the leader's payoff under the worst possible follower action. They have found that assuming the follower responds sub-optimally provides a performance increase in cases where the follower has a limited number of observations drawn from the leader's strategy.

The Automated Negotiating Agents Competition [7] pits negotiation agents against one another. In each match two or more agents take turns proposing outcomes from a space of possible outcomes, which is selected on a per-match basis. If agents do not

agree on an outcome before the deadline is reached, each agent receives its reserve value. In addition, in some matches there is a discount factor to encourage agents to reach an agreement more rapidly. Unlike in the Gift Exchange game, agents do not know their opponent's utility function—to play the game, they must simultaneously attempt to estimate their opponent's utility for each of the potential outcomes and their opponent's willingness to cooperate or make concessions. Most agents designed for this competition focus on opponent modeling; they attempt to predict which offers their opponent will accept. In competition it has been found that the most successful agents are generally tougher negotiators. This is a consequence of the fact that agents generally reach some agreement (which suggests that most agents are not too rigid). If the community of agents frequently failed to reach an agreement, then more generous agents would be favored. This game is more suited to negotiation than the Gift Exchange Game because only the final offer has an effect on the payoffs received by the agents. In the Gift Exchange Game, being a tough negotiator imposes an immediate opportunity cost, as the agent forgoes the chance to cooperate in that round, and may need to pay a cost to punish the opponent for rejecting the agent's desired outcome.

In the Gift Exchange game agents can select outcomes which provide benefits to both players. In an environment where the agent can only take a penalty to give a larger gift to the opponent reciprocation works differently [42]. The decision to cooperate can be treated as evidence of an agents discount factor; an agent which cooperates is presumed to do so because they anticipate that the opponent will reciprocate and the agent's discount factor is low enough that they are willing to pay a present cost for the expected future benefit of the opponent's cooperation. Discount factor is relevant because agents in this environment can be replaced randomly. This is an interesting alternative approach to characterizing an agents behavior, but it is dependent on the assumption of a grim trigger strategy for the opponent.

Chapter 3

Attitude and Belief

Cooperation is the ability of an agent to work together with another agent to achieve an outcome which is better for both agents. It is valuable for an agent to be able to cooperate in any environment where cooperation is possible. However, cooperation can be risky; if the opponent does not also cooperate the agent may be worse off than if it had never attempted to cooperate.

Cooperation plays an important role in evolution, but it is difficult to explain how cooperation developed, since natural selection favors defectors who take advantage of cooperators without paying any cost. One of the mechanisms postulated for evolution of cooperation [57] is direct reciprocity [71], where in repeated encounters two individuals can choose to cooperate or defect. This was formalized in the Iterated Prisoner Dilemma [62, 6, 58] and in the Tit-for-Tat strategy, a strategy which starts with cooperation and then reciprocates whatever the other player has done in the previous round.

3.1 Cooperation

It is possible to achieve cooperation in repeated games using Tit-for-Tat [6] or variants such as win-stay lose-shift [58], but Tit-for-Tat requires that moves are labelled as cooperative or uncooperative. This is not suitable for agents which will be operating in environments which are too large and complex to be analyzed and labelled beforehand.

Another requirement of the Tit-for-Tat strategy is that the actions available to the players cannot change. However this requirement is not fundamental to reciprocation; it is possible to reciprocate by choosing an action which benefits the opponent even if that action is not identical to the action chosen by the opponent to benefit the agent.

We will consider the problem of reciprocation in a game where actions are not labelled as cooperative and game changes between consecutive rounds. In order to create an environment suitable to cooperation and general enough to be adaptable to different situations we considered a number of criteria:

1. Cooperation must be possible. This excludes environments which consist of fixed-sum games, where a gain for one player is necessarily a loss for the other.
2. Exploitation must be possible as well - if there is no danger of exploitation then methods of cooperation might not guard against it, which would make them unsuitable for environments in which exploitation is possible. This excludes environments where players interests are completely aligned, such as if their payoffs are identical.
3. The opportunity for reciprocation is necessary, because reciprocation provides a way to cooperate without becoming vulnerable to exploitation. This means that players must interact multiple times.
4. The environment should have minimal constraints on interactions between agents, so that methods of cooperation will be suitable for other environments.

We have tried to satisfy all these criteria by generating each single interaction of two agents from a probability distribution over a large class of normal form games, where each normal form game is randomly generated and played only once. Since each game is randomly generated, it is extremely unlikely that it will be a fixed sum game or that the payoffs for each player will be equal. Cooperation in this environment is possible, and so is exploitation. Since agents play with each other multiple times, even though the game is different every time, they have the opportunity to reciprocate their opponents cooperative or exploitative moves (the problem of determining which moves are cooperative and which are exploitative is left to the agents).

There are several desirable properties for agents in this environment:

1. They should not be vulnerable to a hostile opponent. Their payoff should not drop lower than the best payoff they could achieve if their opponent had the sole goal of reducing their payoff.
2. They should be able to achieve cooperation. When playing against another agent which is willing to cooperate they should be able to jointly increase their payoffs.
3. They should not be vulnerable to exploitation. They should only cooperate if their opponent is cooperating as well. Unreciprocated cooperation over the short term is reasonable (such as on the first move in Tit-for-Tat), but if the opponent has a history of not cooperating an agent should not continue to cooperate.

Our method of achieving cooperation is based on a parameter driven modification of the original game, where the parameter models the attitude of each agent towards the other agent. For any given game, we construct a modified game using the attitudes of both players. If both players have positive attitudes, a Nash equilibrium of the modified game has the property that when it is played in the original game the expected payoff of each player will be higher than their expected payoff from playing a Nash equilibrium of the original game. (A discussion on how Nash equilibria are computed and selected

is included later).

Since agents generally do not know the attitude of their opponent and since that attitude can change over time, we present a particle filter for an agent to estimate the opponent's attitude, and chose its own attitude accordingly.

3.2 Repeated Randomly Generated Normal Form Games

To study cooperation we chose our environment with a number of goals in mind. Cooperation should be possible, but not the only reasonable course of action. Cooperation should be non-trivial; agents should need to determine which moves are cooperative on their own. Cooperation should be based on reciprocation, so agents should interact repeatedly, but not repeat the same game, so agents will need to cooperate from general strategies, and not just learn how to cooperate in a single game.

The environment we have chosen consists of repeated play of randomly generated normal form games. After exploring a number of alternatives, we have found that 16 move normal form games with payoffs drawn from a uniform distribution between 0 and 1 provide opportunities for cooperation without making cooperation the only reasonable choice. Increasing the number of moves per player causes the environment to become too computationally expensive without changing the nature of the game. Reducing the number of moves per player reduces opportunities for cooperation. We have explored generating payoffs from a normal distribution, but found that this also reduced the opportunities for cooperation. Running 1000 iterations allows sufficient time for agents to adjust to the play of their opponent with a high degree of accuracy. The sequence of play in our environment is as follows:

1. Generate a game by assigning each player 16 possible moves, and drawing a payoff for each player for each combination of moves from a uniform distribution from 0

to 1.

2. Allow both players to observe the game and simultaneously select a strategy for the game which consists of a probability distribution over possible moves.
3. Draw a move for each player from the probability distribution the player provided.
4. Award each player the appropriate payoff for the pair of moves chosen.
5. Inform each player of the move chosen by its opponent.

This is a good environment to study cooperation because players interests are neither diametrically opposed nor identical, so cooperation is possible without being mandatory. Agents in this environment must determine how to cooperate on their own without any environmental cues, and because the games are randomly generated, they must be able to cooperate in a wide variety of situations.

3.3 Attitude

The problem which must be solved to achieve a reciprocating strategy in a repeated randomly generated normal form game is how to identify cooperative strategies. Given a particular normal form game, we identify cooperative strategies by using a modification of the game. Each player selects an *attitude* which reflects the degree to which it is willing to sacrifice its own score to improve its opponent's score. An attitude is a real number, in the range between -1 and 1. An attitude of 1 means that the opponent's payoff is valued as highly as the agent's own payoff. An attitude of 0 means that the agent is indifferent to the opponent's payoff. An attitude of -1 means that the agent is only concerned with how well it does in comparison to its opponent.

The attitude of the agent and its opponent is used to create a modified game in which each player's payoff is equal to their payoff from the original game plus their attitude times the payoff of their opponent in the original game. Let G be a game with

players $\{A, B\}$, action sets M_A and M_B, and utility functions $U_A, U_B : M_A \times M_B \to \mathbb{R}$. Construct the modified game G' with the same players and action sets, and the utility functions modified as follows:

$$U'_A(m_A, m_B) = U_A(m_A, m_B) + \alpha_A * U_B(m_A, m_B)$$
$$U'_B(m_A, m_B) = U_B(m_A, m_B) + \alpha_B * U_A(m_A, m_B)$$

where α_i indicates the attitude of player i and $m_i \in M_i$ indicates the action selected by player i. When players select their actions according to the Nash equilibria of the modified game and play them in the original game the score of each player improves.

This approach to cooperation has been explored as an explanation of the non-Nash behavior of people [34, 33], here we are exploring it as a means of generating non-Nash cooperative behavior. Note that in many games there can be multiple Nash equilibria. If agents play different Nash equilibria, or make false assumptions about their opponent's attitude, then they do not achieve cooperation. Fortunately, we will show later that a player can learn what attitude and method of selecting Nash equilibria is used by the opponent.

Figure 3.1 shows the effect of various combinations of attitude on the payoff of a player. To compute the Nash equilibria we used the Lemke-Howson algorithm as described in [54]. When both players adopt an attitude of 1, they can improve their average payoffs from .80 to .90. Even when they only adopt an attitude of .2 their payoffs improve to .87.

Unsurprisingly, the strongest effect on an agent's payoff is the attitude of the opponent. If the opponent is hostile, the agent will do poorly regardless of the attitude of the agent, although the agent will generally perform better when the agent adopts an attitude close to 0. If both agents adopt a positive attitude there is a *plateau of*

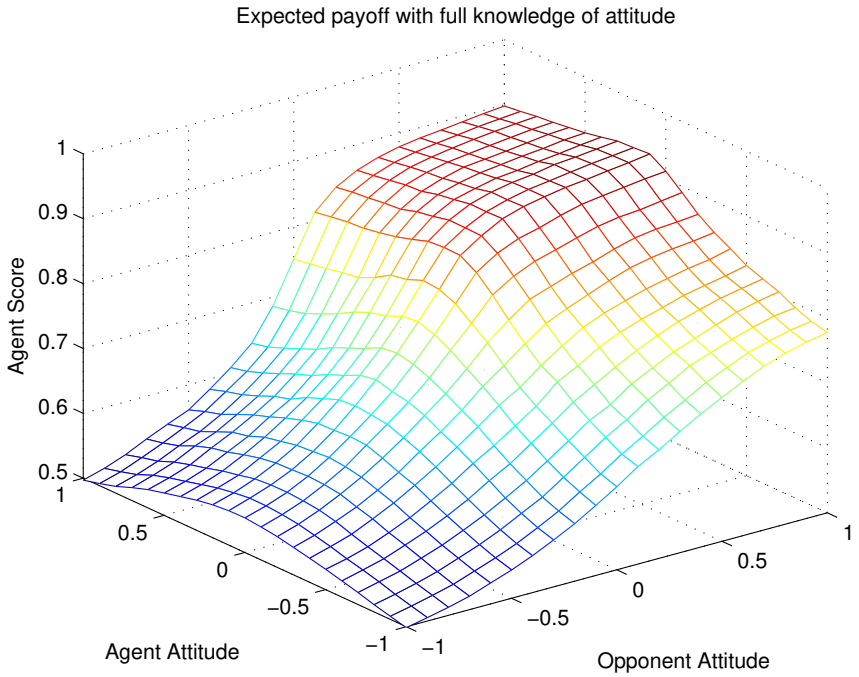

Figure 3.1: This graph shows how the payoff of an agent is affected by the attitudes selected by the agent and its opponent. These are aggregated results over 1000 games with 16 moves per player and payoffs drawn from a uniform distribution between 0 and 1. The results for a single game may be quite different.

cooperation where both agents receive a high payoff. Interestingly, when the opponent's attitude is positive, the agent will achieve a higher payoff with an attitude of .1 than with an attitude of 0, although increasing its attitude above .1 will result in a slight decrease in payoff. When the agent's attitude is even slightly positive the opponent can play a move which aims at a good outcome for both players, confident that the agent will reciprocate instead of picking some outcome which is slightly better for the agent and much worse for the opponent. This opens more opportunities for cooperation, resulting in a higher payoff for an agent with an attitude of .1.

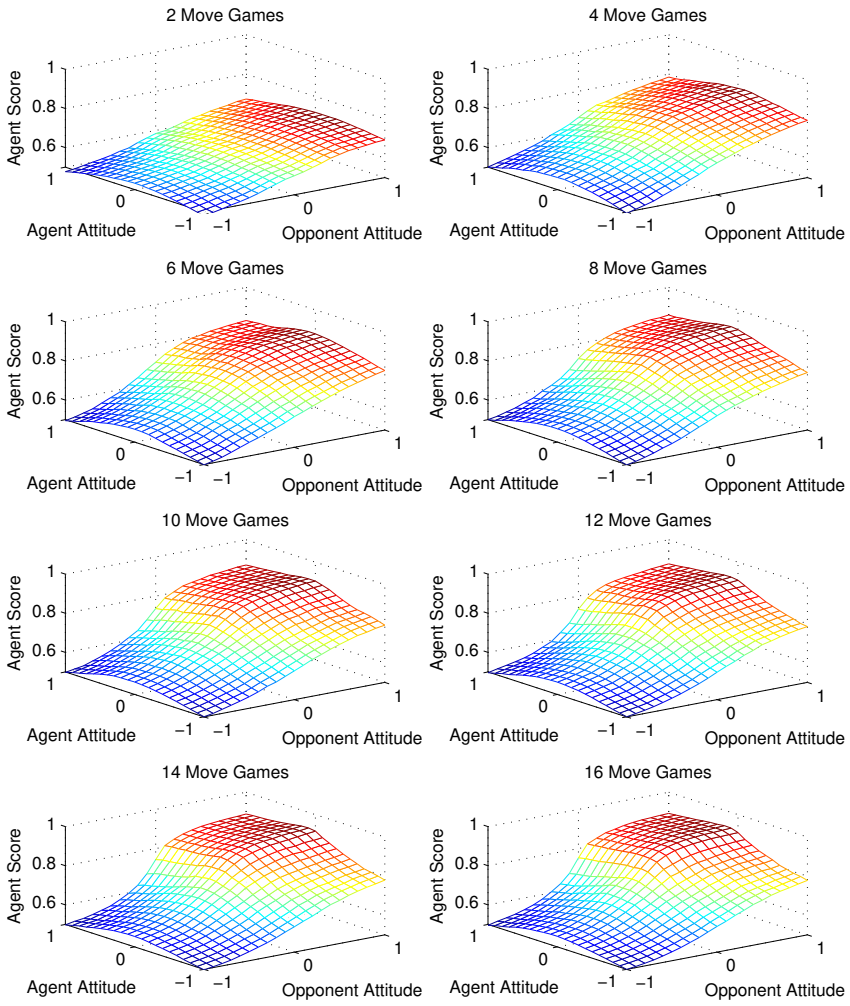

Figure 3.2: This graph shows how the payoff of an agent is affected by the attitudes selected by the agent and its opponent. These are aggregated results over 1000 games with 2 to 16 moves per player and payoffs drawn from a uniform distribution between 0 and 1. The results for a single game may be quite different.

3.3.1 Effects of Attitude

The performance effect of adopting different combinations of attitude values depends on the game which is being played. Figure 3.1 shows the average results for a game with

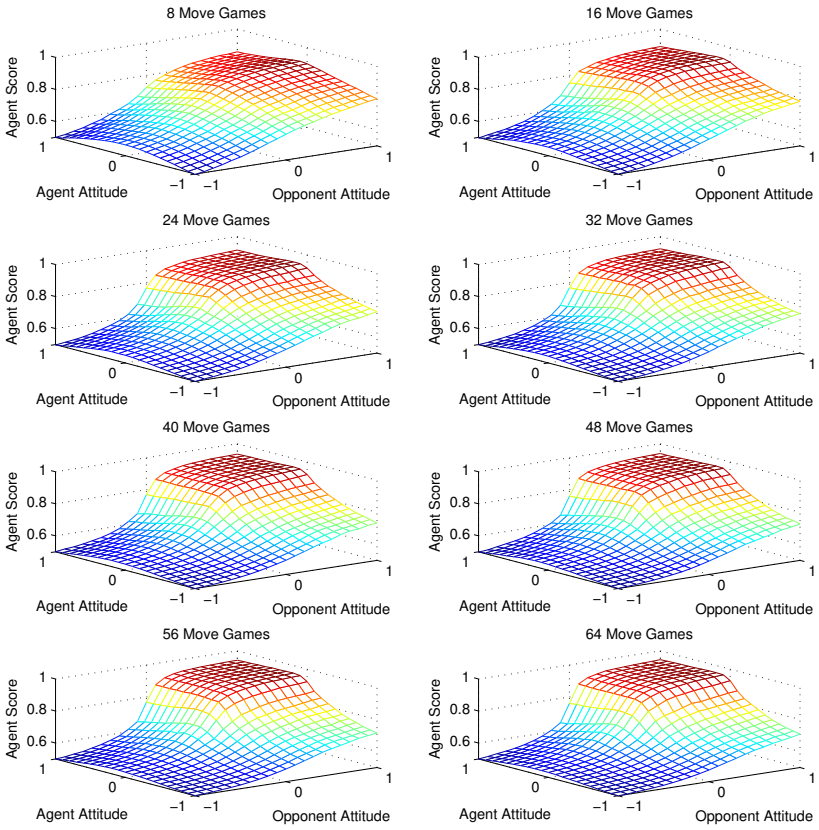

Figure 3.3: This graph shows how the payoff of an agent is affected by the attitudes selected by the agent and its opponent. These are aggregated results over 1000 games with 8 to 64 moves per player and payoffs drawn from a uniform distribution between 0 and 1. The results for a single game may be quite different.

16 moves with payoffs uniformly distributed between 0 and 1. It is also interesting to consider the effects in games with greater or fewer moves or with different probability distributions over payoffs. We will also look at the non-aggregate effects of attitude in specific games.

Figures 3.2 and 3.3 show the effects of attitude on the payoff of the agent for games with 2 to 64 actions per player. When each player only has 2 actions, the primary

influence on an agent's score is the opponent's attitude. It is also beneficial for the agent to adopt an attitude close to 0.

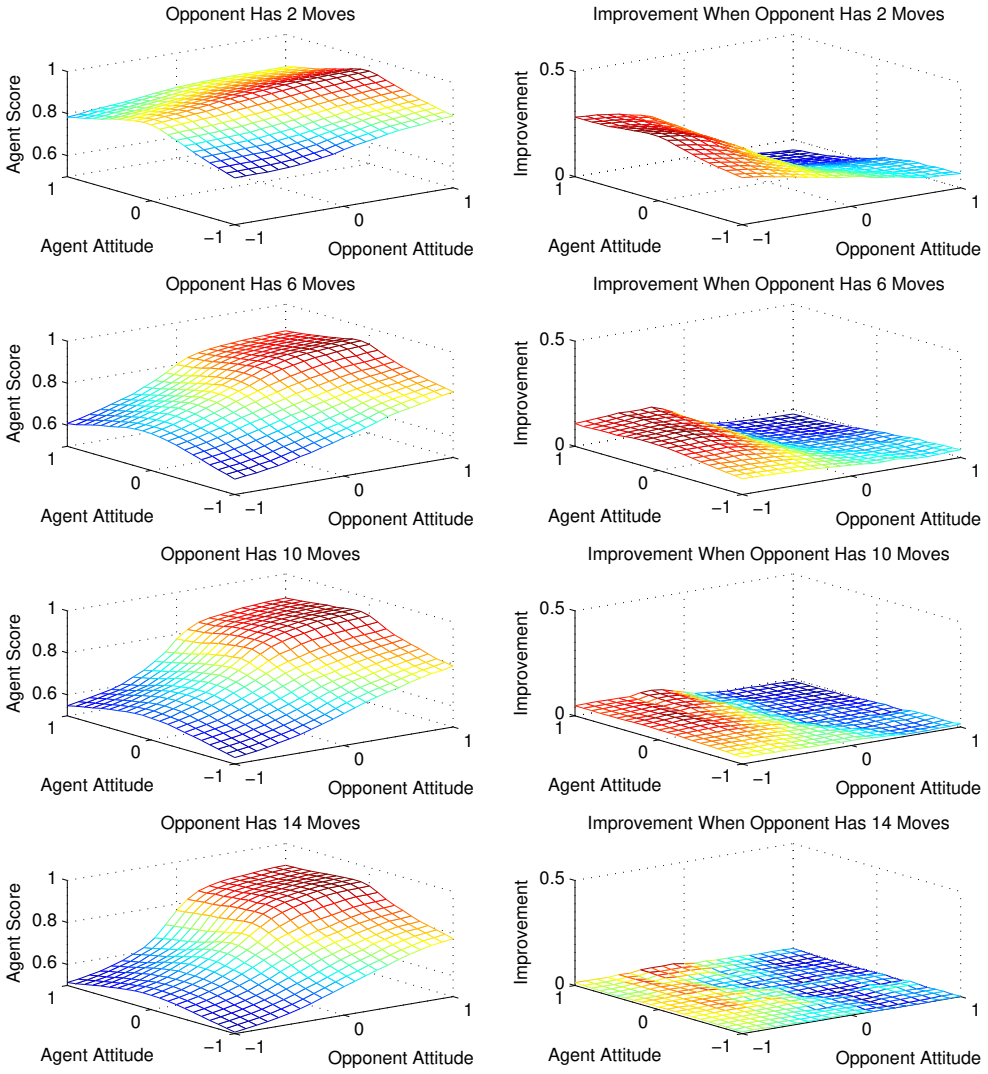

Figure 3.4: This graph shows how the payoff of an agent is affected by the attitudes selected by the agent and its opponent. These are aggregated results over 1000 games with 16 moves available to the agent and 2 to 14 moves available to the opponent. Payoffs are drawn from a uniform distribution between 0 and 1. The results for a single game may be quite different. The graphs on the left show the relative change in payoff for the agent compared to when the opponent has 16 moves available.

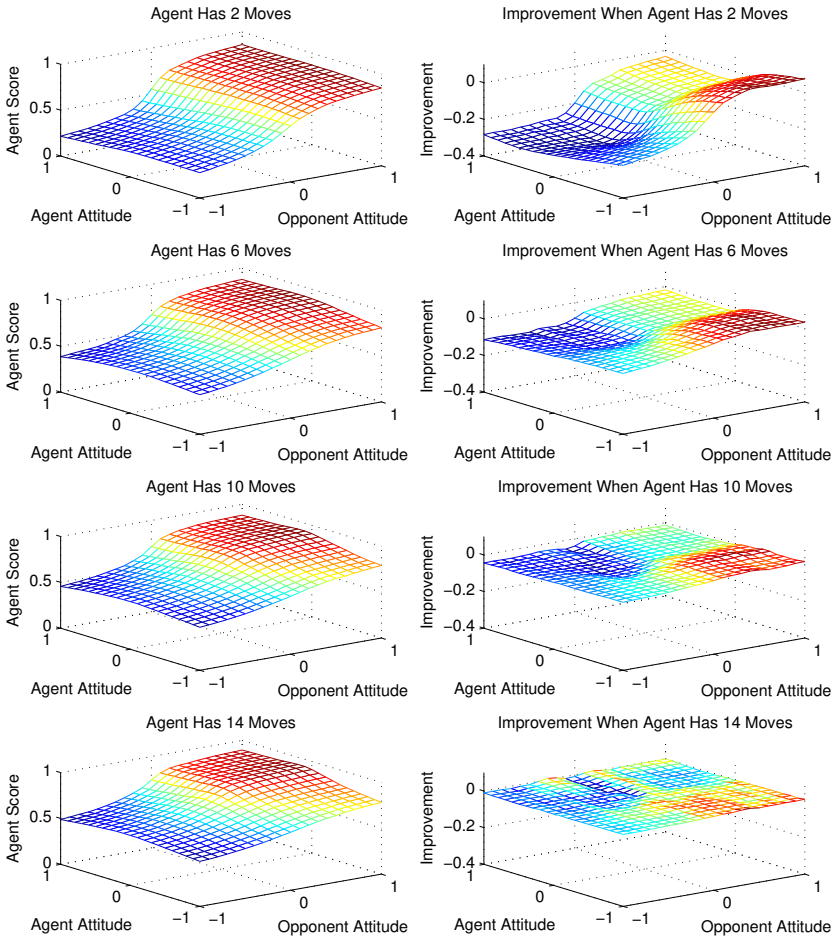

Figure 3.5: This graph shows how the payoff of an agent is affected by the attitudes selected by the agent and its opponent. These are aggregated results over 1000 games with 16 moves available to the opponent and 2 to 14 moves available to the agent. Payoffs are drawn from a uniform distribution between 0 and 1. The results for a single game may be quite different. The graphs on the left show the relative change in payoff for the agent compared to when the agent has 16 moves available.

As the number of moves increases, the payoff to the agent increases, and we begin to see a plateau of cooperation. When the number of moves increases beyond 16, the plateau of cooperation gets higher, because with more combinations of moves available, there are more opportunities to draw cooperative combinations from the payoff distribution.

We have been looking at games in which both agents have the same number of moves available to them. We can give one agent more power than the other by creating a game in which that player has more moves available to it. Figures 3.4 and 3.5 show the effect on an agent's payoff when one of the players has more moves available to it than the other player. When the opponent has more moves available the agent's payoff suffers, particularly when the opponent's attitude is negative. When the opponent's attitude is positive and the agent's attitude is negative, the agent's payoff actually increases, because the agent is less capable of sacrificing its own payoff to hurt the opponent. When the agent has more power, its payoff increases, especially when the opponent's attitude is negative. When the opponent has only 2 moves, the agent's payoff drops a bit when it adopts a cooperative attitude against a cooperating opponent because the reduced number of moves reduces the opportunities for cooperation.

Figure 3.6 shows the effect of attitude on the payoff of the agent when payoffs are drawn from a Gaussian distribution with mean 0 and standard deviation 1. As with a uniform distribution the primary effect on an agent's payoff is the opponent's attitude and an agent's payoff is also improved by the agent adopting an attitude closer to 0. Unlike a uniform distribution, there is no plateau of cooperation; this happens because, unlike a uniform distribution, there is not an upper limit on the payoffs which can be generated by a Gaussian distribution. Therefore when both players adopt a positive attitude, but one player is more generous than the other, it is possible for the players to select an outcome where the other player receives a greater share of the reward.

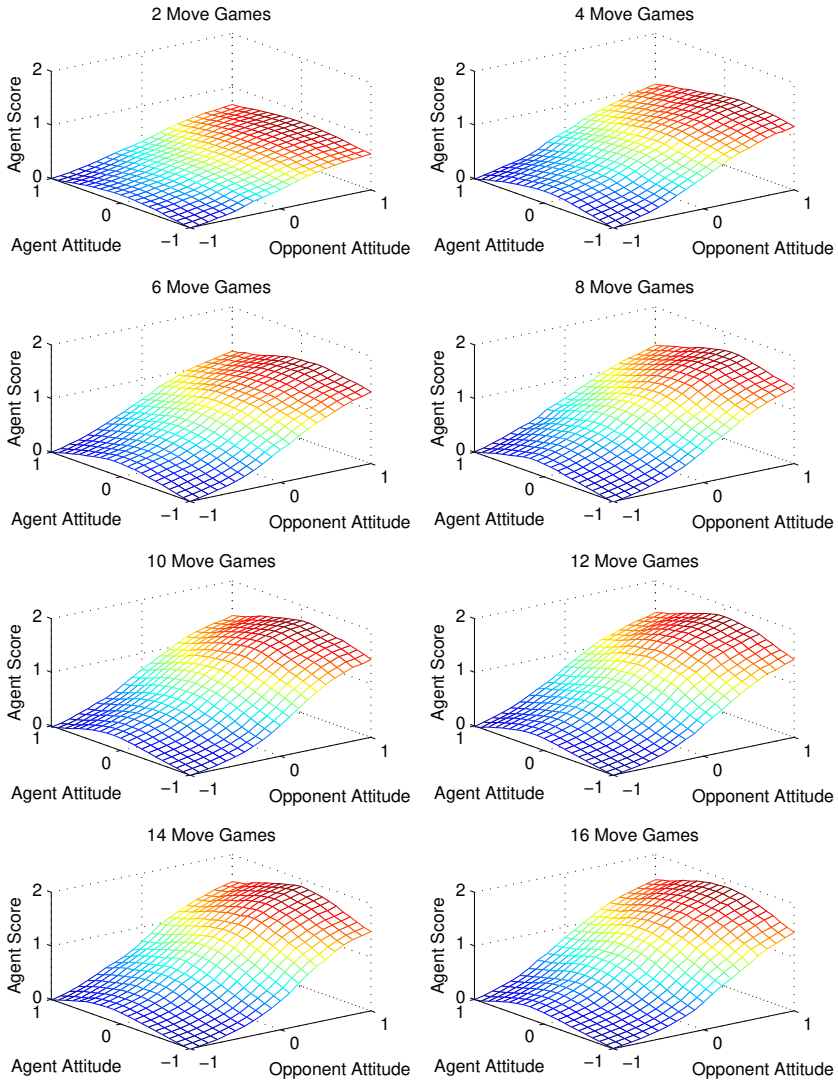

Figure 3.6: This graph shows how the payoff of an agent is affected by the attitudes selected by the agent and its opponent. These are aggregated results over 1000 games with 2 to 16 moves per player and payoffs drawn from a Gaussian distribution with mean 0 and standard deviation 1. The results for a single game may be quite different.

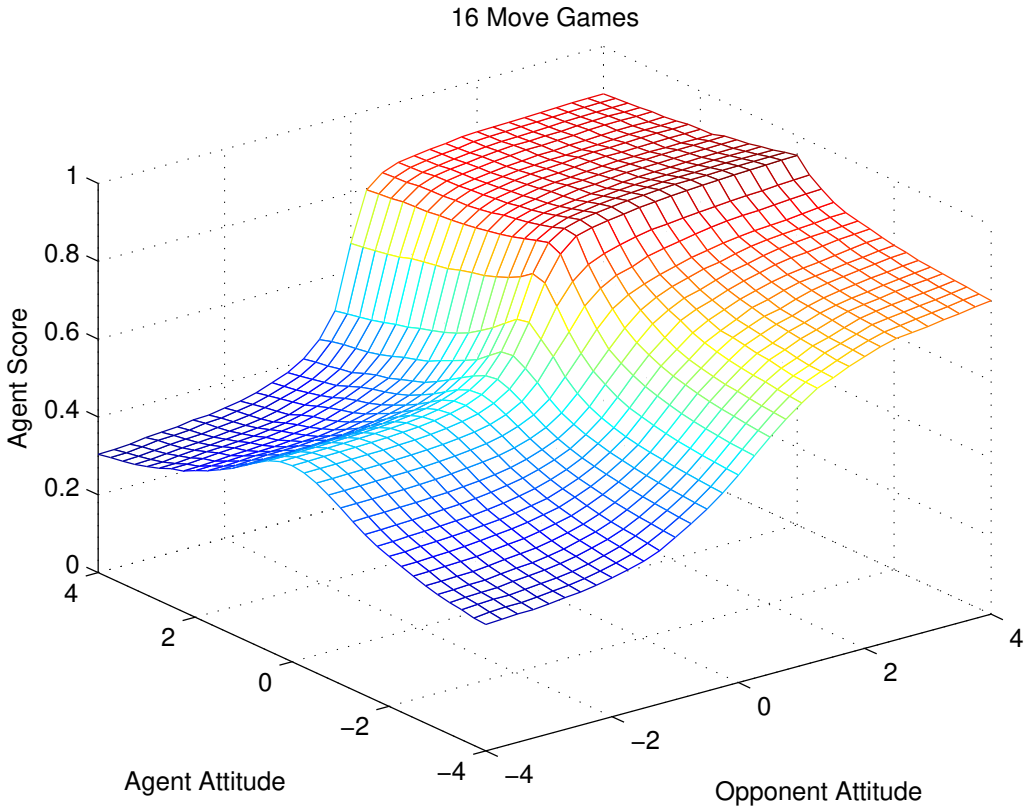

Figure 3.7: This graph shows how the payoff of an agent is affected by the attitudes selected by the agent and its opponent. These are aggregated results over 1000 games with 16 moves per player and payoffs drawn from a uniform distribution from 0 to 1. The results for a single game may be quite different.

Until now we have limited attitude values to the interval $[-1, 1]$. We have done so because adopting an attitude value greater than 1 or less than -1 implies that the agent cares more about the payoff to the opponent than its own payoff. For an indifferent agent, adopting an attitude of 1 can be justified as an attempt to reach a social welfare maximizing outcome, but adopting an attitude above 1 is harder to justify. Figure 3.7 shows the effect of adopting attitude values in the range $[-4, 4]$. As the opponent's attitude drops below -1 the agent's payoff gets worse, and the benefit to the agent of

adopting an attitude of 0 becomes more significant. One interesting thing to note is that as the opponent's attitude increases above 1 an agent with an attitude of .1 no longer receives a higher payoff than an agent with an attitude of 0. One way to view this is to consider an attitude value as expressing a preference for a certain relative weighting of payoffs. A player with an attitude of 1 prefers a 1 : 1 ratio, a player with an attitude of 0 prefers a 1 : 0 ratio, while an opponent with an attitude of 4 prefers a 4 : 1 ratio. When the opponent's attitude is very high, adopting an attitude of .1 doesn't allow for additional cooperative opportunities over an attitude of 0 as frequently because the opponent is more willing to sacrifice its own payoff to give the agent a greater payoff.

In our environment games are generated randomly, so we have been presenting aggregate results to show the general effects of various alterations in the game structure on the performance of combinations of attitude values. However, it is important to note that the effects of attitude do vary according to the specific game. Figures 3.8, 3.9, and 3.10 show the effects of attitude in specific games generated by drawing from a uniform distribution between 0 and 1 with 2(Figure 3.8), 16(Figure 3.9), or 64(Figure 3.10) moves per player.

The effect of attitude in games where players have 2 moves is fairly irregular. Since the games are randomly generated and there are only 4 combinations of actions, it is easily possible for one pair of actions to be a dominant strategy for both players regardless of the attitudes they adopt, as we see in the bottom two examples. When players have 16 moves the general pattern seen in Figure 3.1 begins to become more evident and when the players have 64 moves it begins to look very similar. There is still noise, but with that many moves agents will generally have enough options that the combination of their attitudes results in payoffs which are reflective of those attitudes.

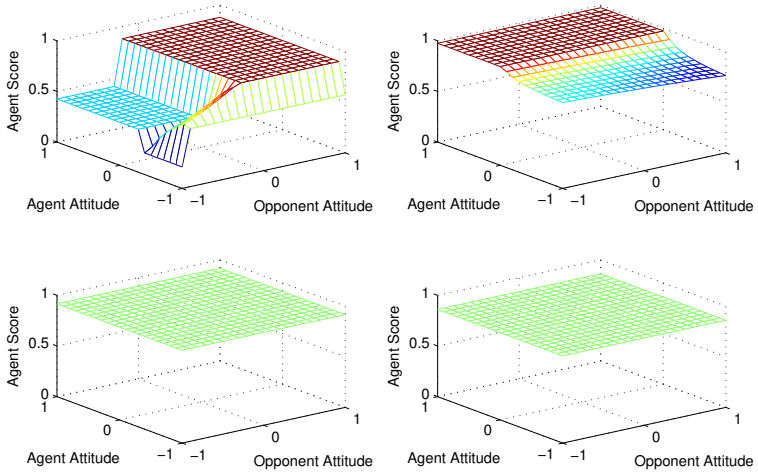

Figure 3.8: This graph shows how the payoff of an agent is affected by the attitudes selected by the agent and its opponent. These are 4 examples of randomly generated games with 2 moves for each player and payoffs drawn from a uniform distribution from 0 to 1.

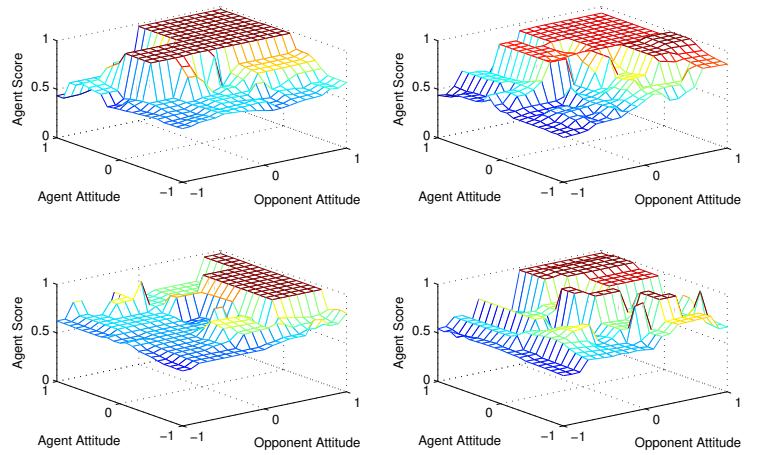

Figure 3.9: This graph shows how the payoff of an agent is affected by the attitudes selected by the agent and its opponent. These are 4 examples of randomly generated games with 16 moves for each player and payoffs drawn from a uniform distribution from 0 to 1.

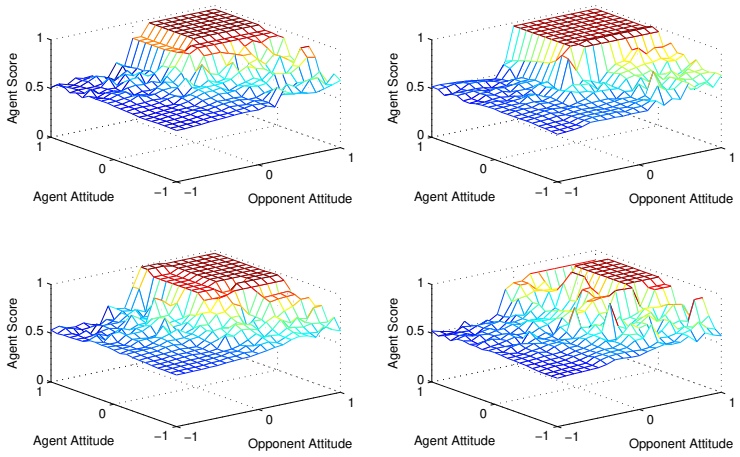

Figure 3.10: This graph shows how the payoff of an agent is affected by the attitudes selected by the agent and its opponent. These are 4 examples of randomly generated games with 64 moves for each player and payoffs drawn from a uniform distribution from 0 to 1.

3.4 Learning Attitude and Belief

When both player's attitudes are public knowledge, attitude can be used to find strategies whose payoffs reflect the combination of attitudes used to generate them. However attitudes are not generally public knowledge, especially in instances where the agent is adopting an attitude to encourage reciprocation, and not because it has any preferences about the payoff received by the opponent. We have explored a number of different approaches an agent can use to determine the opponent's attitude. We call the value an agent uses for an estimate of its opponent's attitude its *belief*.

Figure 3.11 shows the effect of different combinations of attitude values when both players adopt the belief that their opponent is indifferent (has an attitude of 0). The player's beliefs in this case are inaccurate (except at the center of the graph), and as a result the strategies they select perform poorly. Figure 3.12 shows the effect when both players assume that the opponent holds the same attitude they do. This assumption is

slightly better, in that in allows players to identify a cooperative strategy, but it is still inefficient when players assumptions are incorrect.

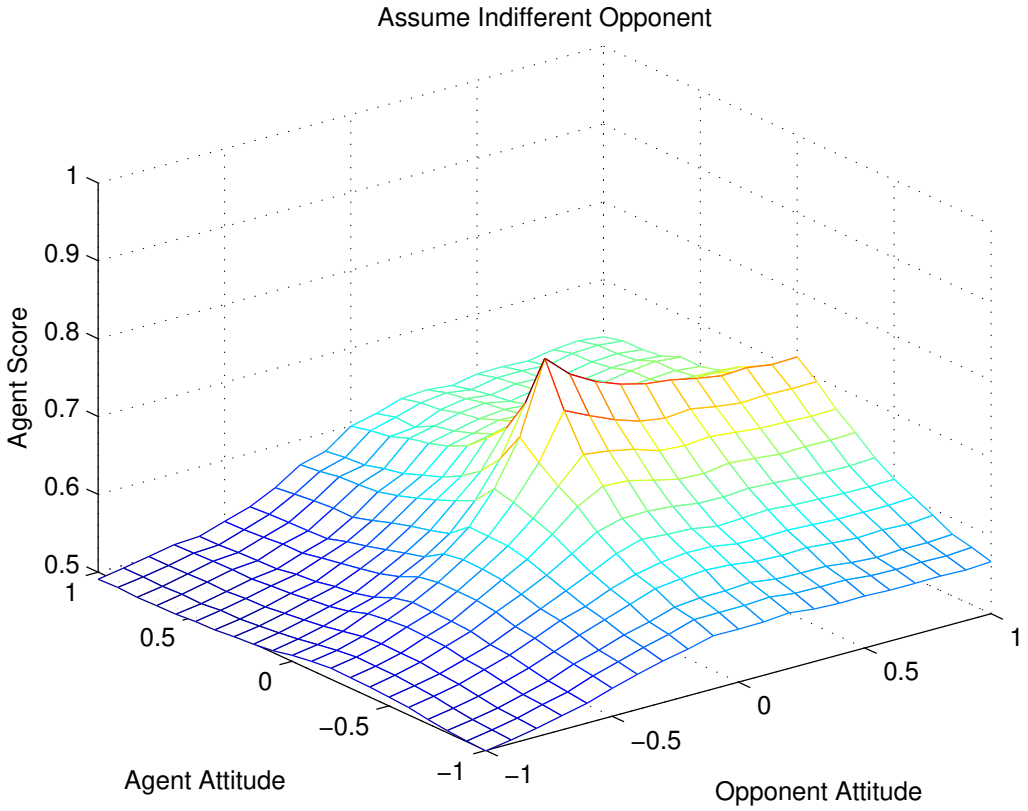

Figure 3.11: This graph shows how the payoff of an agent is affected by the attitudes selected by the agent and its opponent when both players assume that the other is indifferent.

In view of the inefficiencies introduced by incorrect assumptions, it is tempting to have players disclose their attitudes to each other. Unfortunately this does not work. Figure 3.13 shows the effect deception has when an indifferent player adopts a deceptive attitude which is believed by the opponent. We can see that the most effective claim for an indifferent agent to make is to claim an attitude of 1. We can also see that it is nearly always costly for an agent to believe such a claim. The only exception is when the

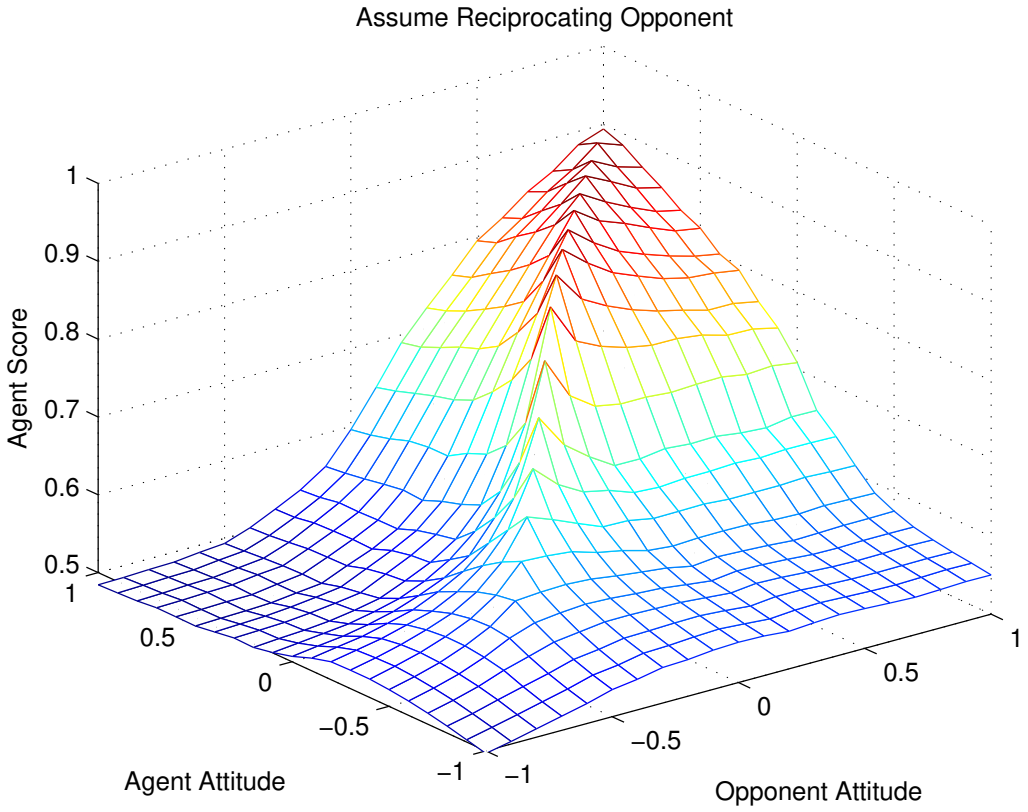

Figure 3.12: This graph shows how the payoff of an agent is affected by the attitudes selected by the agent and its opponent when both players assume that the other agent has adopted the same attitude.

agent has a negative attitude; an indifferent opponent can occasionally achieve a better payoff by claiming to be hostile and then choosing a different strategy. This prevents the agent from sacrificing its own payoff to hurt the opponent.

Since an agent cannot be trusted to honestly disclose its attitude, it is necessary to learn the attitude being used by an agent over repeated interactions. Since an opponent's behavior will be strongly influenced by its belief of the agents attitude, it is also necessary to learn the opponent's belief. This is difficult because the only evidence available is the sequence of moves chosen in previous games. It is further complicated by the possibility

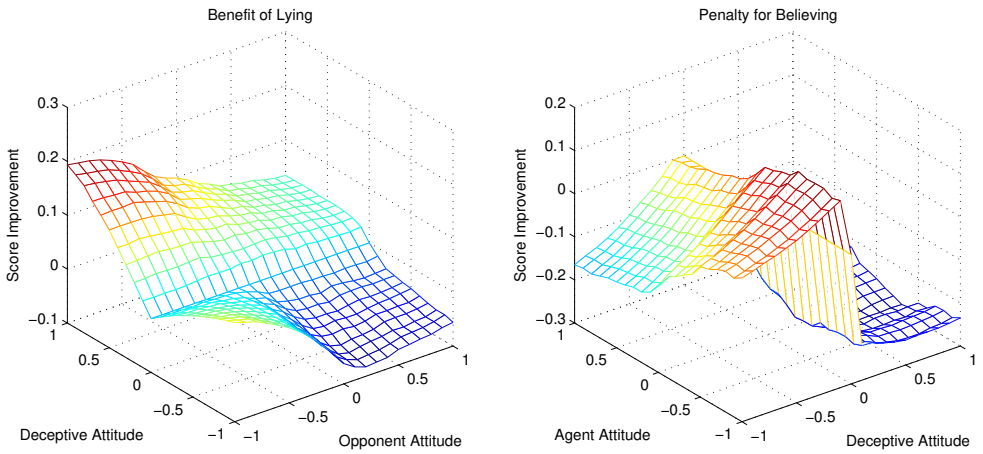

Figure 3.13: These graphs show the effect of one player deceiving the opponent about its attitude. The left graph shows the benefit an indifferent agent can gain by adopting various deceptive attitudes. The right graph shows the cost to the agent of believing an opponent which is adopting various deceptive attitudes.

of an agent changing its attitude and belief, perhaps in response to its perceptions of its opponent's attitude and belief.

The approach we propose uses Monte Carlo methods to represent a probability distribution over values of attitude and belief. We model a probability distribution over attitude and belief values using a set of particles, each of which has a value for attitude and belief. Each particle's combination of attitude and belief will assign a probability to each move of an observed game. Upon observing the move chosen by the opponent, each particle is assigned a weight equal to the probability it assigned to that move. Then the set of particles is resampled with probability proportional to the weights assigned. This procedure is a variation on a particle filter - for more details see [2].

We use a regularized particle filter [55], which resamples from a continuous instead of a discrete distribution. As observations are made, the relative probability of the particles changes. At the extreme, if one particle has all the weight, the distribution is effectively represented by a single particle. To avoid this, when the effective number of

particles drops below a threshold, a new set of particles is drawn by sampling from the existing distribution and adding noise.

Noise is drawn from a Gaussian distribution with 0 mean and standard deviation equal to $N^{-1/6}$ times the standard deviation of the particle set, where N is the number of particles. This is an improvement over the approach in [19] because it doesn't require knowledge of the distribution from which games are drawn and increases accuracy. Method is a discrete value, so it cannot be perturbed with Gaussian noise. Instead, with some probability we change it to a random new method. The optimal probability is found using a technique called Leave-One-Out, where we select the probability that gives the highest likelihood of resampling the current distribution from a distribution created by removing one particle from the current set.

We use 400 particles with attitude and belief drawn from a Gaussian distribution centered at 0 with the identity matrix as a covariance matrix, and method drawn from a uniform distribution over the list of methods under consideration. We assign each particle a weight of .0025 and resample if the effective number of particles goes below 200.

3.5 Cooperation

Once an agent has an estimate of the attitude and belief of its opponent, it can respond by using those values of attitude and belief to generate a modified game and playing a Nash equilibrium of the modified game. This will allow an agent to cooperate with an opponent which is already cooperating, but two agents which only do that will not cooperate because they will each wait for the other to make the first move. We have developed an agent which uses reciprocation to cooperate in self-play while still performing well against opponents which don't reciprocate. After learning the opponent's attitude and belief our cooperative agent reciprocates with an attitude slightly higher than its

estimate of the opponent's attitude, with a maximum value of 1. If the opponent is not cooperative this will not have a large effect on the agent's score. If the agent plays an opponent with a similar method of choosing an attitude they will each increase their attitude to cooperate slightly more than the other until both agents are playing fully cooperatively.

Algorithm 1 *RegularizedParticleFilter*

Initialize N ▷ Number of particles
for $i \in 1..N$ **do** ▷ Generate initial particles
    Create particle p with attitude p_α, belief p_β, and method p_μ drawn from prior
    Assign weight $p_w = 1/N$
end for
while presented with data **do**
    Observe opponent's action M in game G
    Compute effective number of particles $N_{eff} = 1/[\sum_{p \in P} {p_w}^2]$
    if $N_{eff} >$ threshold **then** ▷ Do not need to Resample
        for particle $p \in$ P **do** ▷ Update weights
            Compute probability of opponent's action M in game G, $\mathbb{P}(M|p)$, given p_α, p_β, and p_μ
            Update p_w=$p_w \times \mathbb{P}(M|p)$
        end for
    else ▷ Resample particles
        Compute standard deviation σ_α of p_α and standard deviation σ_β of p_β
        $h = N^{-1/6}$
        Compute perturbation probability $\mathbb{P}(permute)$ for p_μ
        while accepted particles $< N$ **do** ▷ Generate new particles
            Select particle p from P with probability proportional to p_w
            Create new particle p' with $p'_\alpha = p_\alpha + h \times N(0, \sigma_\alpha)$; $p'_\beta = p_\beta + h \times N(0, \sigma_\beta)$
            With probability $\mathbb{P}(permute)$, p'_μ = random method else $p'_\mu = p_\mu$
            Compute $\mathbb{P}(M|p')$= probability of opponent's action M in G given p'_α, p'_β, and p'_μ
            Accept p' with probability $\mathbb{P}(M|p')$
        end while
    end if
end while

Choosing a Nash Equilibrium

A significant difficulty with this approach to cooperation lies in selecting the Nash equilibrium. Games may have multiple Nash equilibria, and games drawn from the distribution we have chosen frequently do. If each agent plays a different Nash equilibrium the payoffs they receive will be effectively random, which will neutralize any attempt at cooperation and result in a significantly worse outcome than if they had both chosen the same Nash equilibrium but not attempted to cooperate.

Many methods of selecting a specific Nash equilibrium have been proposed. A method of selecting a subset of Nash equilibria from the set of all Nash equilibria of a game is known as a refinement of the Nash equilibrium. There is a lot of research on the problem of finding a unique refinement of the Nash equilibrium [38], but there is no consensus on any particular refinement.

Fortunately, it is possible to learn what method an agent uses to select equilibria by extending the method we use to learn attitude and belief. Each particle can be assigned a method of selecting the Nash equilibrium in addition to its values for attitude and belief. The weight for the particle during resampling is then determined by using its method to calculate a Nash equilibrium for the modified game. When particles are perturbed, the estimated error is used to determine a probability of switching the method. In this manner any reasonable method of choosing a Nash equilibrium can be included among the possibilities considered, and detected if the opponent uses it. We use a deterministic algorithm to find Nash equilibria which can return different equilibria depending on a starting parameter. This provides a number of methods to find Nash equilibria.

3.6 Evaluation

3.6.1 Learning a stationary opponent

Figure 3.14 shows the speed at which our algorithm can learn the attitude and belief used by a stationary agent. The agents attitude and belief are randomly drawn from a Gaussian distribution with mean 0 and standard deviation 1. The agent uses a set of starting parameters drawn from a uniform distribution over all values. Note that 100% predictive accuracy is not achieved despite a low level of error. This is because agents which are not fully cooperative tend to use randomization when picking their moves.

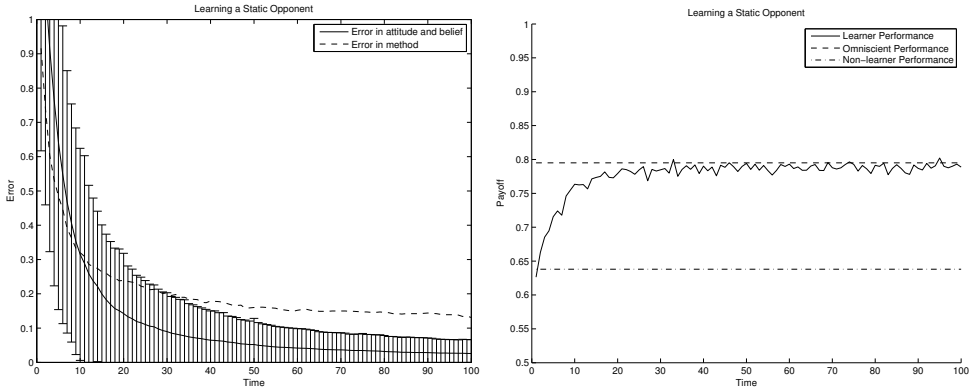

Figure 3.14: These graphs shows the efficiency with which an agent can learn a static attitude and belief. The left graph shows the error in the estimate of the opponent's attitude and belief and the estimate of the opponents method. The error bars show +/- 1 standard deviation. The right graph shows the payoff achieved by the learning agent. These results are aggregated over 100 runs. Agents had to learn their opponents choice of Nash equilibrium.

3.6.2 Achieving cooperation in self play

Figure 3.15 shows the speed at which our algorithm can achieve cooperation in self play. Both agents are simultaneously learning the attitude and belief of the other agent, and then setting their own attitude equal to .1 greater than the other players attitude, with

the constraint that their own attitude must fall between 0 and 1.

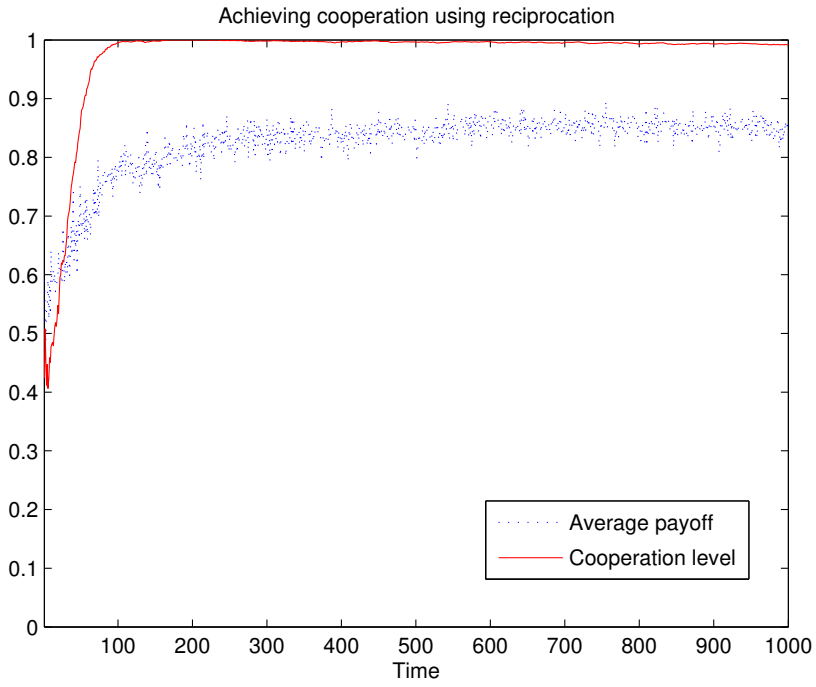

Figure 3.15: This graph shows the speed at which cooperation can be achieved. The solid line shows the level of cooperation between the agents, and the dotted line shows the payoff achieved by the agents. These results are aggregated over 100 runs. Agents had to learn their opponents choice of Nash equilibrium.

3.7 Conclusions and Future Work

This chapter describes an environment to explore cooperation among self-interested agents. It presents an approach which can achieve cooperation in that environment, resists exploitation, and adjusts to changing intentions.

This particle filter based approach to learning opponent strategies is easy to adapt to include any particular strategy, but will fail to learn any strategy which it does not consider. While the size of the space of possible strategies and the large proportion

of irrational strategies in that space suggest that it is not useful to attempt to include every strategy, it would be useful to further investigate various refinements of the Nash Equilibrium.

An advantage of our algorithm is that it can be easily adapted to a situation in which opponents payoffs are not fully known. For example, consider an environment in which the results of interactions are expressed in terms of goods the agents receive, while the agents valuations of goods are unknown. Instead of using attitude/belief, we could define a parameter space which describes how an agent values various combinations of goods, and learn that.

Chapter 4

Restricted Stackelberg Response with Safety

In Chapter 3 we looked at agents which use a particle filter to estimate the opponent's attitude so that the agent can reciprocate appropriately. This implicitly produces a prediction of opponent behavior. However, instead of responding to the prediction, the agent uses attitude values to signal its intent to cooperate. In this chapter we will look at the problem of how an agent should respond to a prediction.

How should an agent respond when given a prediction of opponent behavior in a general-sum two-player normal form game? Selecting the strategy with the highest payoff against the prediction provides optimal performance if the prediction is correct, but can be arbitrarily bad if the prediction is incorrect. Playing a maximin strategy guarantees a payoff equal to the safety value of the game, but at the cost of performance against the prediction. Playing a Nash equilibrium only makes sense if the prediction is incorrect and the opponent also plays that Nash equilibrium. Furthermore, a maximin strategy or a Nash equilibrium don't use a prediction, so there is no reason for an agent using them to make a prediction.

In the previous chapter we have shown how an agent can model the opponent's attitude towards cooperation to pursue opportunities for cooperation while limiting exploitation. In this chapter we introduce *Restricted Stackelberg Response with Safety* (RSRS), a novel method of choosing a mixed strategy for a general-sum game, given a prediction of the opponent's strategy. RSRS uses a prediction weight parameter, w, to determine how much to guard against a best-responding opponent, and a risk factor parameter, r, to determine how much to guard against a worst-case outcome. RSRS provides a way to make a controlled tradeoff between responding to a (possibly flawed) prediction and dealing with a best-responding opponent while also providing a worst-case performance guarantee. We will use fictitious play to make predictions to demonstrate that RSRS can handle flawed predictions, but RSRS can be used with any prediction method that produces a probability distribution over opponent moves.

If the opponent plays a mixed strategy it is impossible to predict their exact action without knowledge of their randomization device, but even if we settle for predicting their mixed strategy any prediction method will only be able to provide predictions for a subclass of all possible opponents. Eventually, it is necessary to accept that one's prediction is as accurate as it can be, and deal rationally with the possibility that it may still be inaccurate.

This leads us to consider the ways in which a prediction can be incorrect, and how we can track and respond to that. In some cases, a prediction may be technically incorrect, but harmlessly so. For example, consider an agent which plays a mixed strategy but uses the digits of π as its randomization device. Such an agent is perfectly predictable in theory (it is playing a fixed sequence of moves), but in practice it is impossible to consider all strategies of that type. In this situation it is reasonable and effective to model the agent as playing the mixed strategy, despite the fact that a more accurate prediction is theoretically possible.

The prediction errors which are of practical concern are those in which the opponent is exploiting the agent's response to an inaccurate prediction, either for their own benefit, or to reduce the payoff to the agent. (Technically speaking it is also a prediction error when the opponent plays to increase the agents score or reduce its own score, but we feel that issue does not represent a significant problem). RSRS can be customized to allow an agent to deal appropriately with those types of opponents by choosing appropriate parameter values.

4.1 Terminology

A game G consists of a set of players $\{A, B\}$, a set of actions for each player $M_A = \{m_A^1 \dots m_A^{n_A}\}, M_B = \{m_B^1 \dots m_B^{n_B}\}$, and a set of utility functions $U_A, U_B : M_A \times M_B \rightarrow \mathbb{R}$. $s_A \in \triangle^{n_A - 1}$ and $s_B \in \triangle^{n_B - 1}$ are the mixed strategies adopted by player A and player B respectively.

We define $U_i(s_A, s_B) = \mathbb{E}_{m_A \sim s_A, m_B \sim s_B}[U_i(m_A, m_B)]$ as the expected outcome for player i when actions are drawn from the distributions s_A and s_B. The Nash equilibrium is a set of strategies s_A, s_B such that $U_A(s_A, s_B) \geq \max_{m_A \in M_A} U_A(m_A, s_B)$ and $U_B(s_A, s_B) \geq \max_{m_B \in M_B} U_B(s_A, m_B)$. The safety value of game G for player i against opponent j is: $V_G^i = \max_{s_i \in \triangle^{n_i - 1}} \min_{m_j \in M_j} U_i(s_i, m_j)$. This is the greatest amount player i can guarantee for herself regardless of the opponent's action. Note that for general-sum games, this value may be lower than the expected payoff of any Nash equilibrium of the game.

If player A is designated as a *Stackelberg leader* [35] for the game, she selects a mixed strategy which is observed by player B before player B selects her strategy.

The Stackelberg equilibrium of a game is a set of strategies s_A, s_B such that

$$\begin{aligned} s_A &= \arg \max_{s \in \triangle^{n_A - 1}} U_A(s, \arg \max_{s' \in M_B} U_B(s, s')) \\ s_B &= \arg \max_{s \in M_B} U_B(s_A, s) \end{aligned}$$

4.1.1 Demonstration Game

The advantages of RSRS can most easily be observed in competitive general sum games where players have some common interests. In more competitive games, such as Rock/Paper/Scissors, performance is similar to other algorithms for risk avoidance. In more cooperative games, such as Battle of the Sexes, faults in the predictor aren't as significant because the opponent has less motivation to play deceptively.

Table 4.1: Payoffs for Rock/Spock/Paper/Lizard/Scissors.

	Rock	Spock	Paper	Lizard	Scissors
Rock	0,0	-.5,1.5	-1.5,.5	.5,-1.5	1.5,-.5
Spock	1.5,-.5	0,0	-.5,1.5	-1.5,.5	.5,-1.5
Paper	.5,-1.5	1.5,-.5	0,0	-.5,1.5	-1.5,.5
Lizard	-1.5,.5	.5,-1.5	1.5,-.5	0,0	-.5,1.5
Scissors	-.5,1.5	-1.5,.5	.5,-1.5	1.5,-.5	0,0

The game we will use to show the properties of our RSRS method is a general-sum modification of Rock/Spock/Paper/Lizard/Scissors – a variant of Rock/Paper/Scissors with 5 moves (Table 4.1).

Rock/Paper/Scissors/Lizard/Spock was presented in the TV show The Big Bang Theory; we have modified it to make it general-sum, and changed the name to reflect the precedence relationship between the moves. Each action beats two other actions, and is beaten in turn by the two remaining actions, as shown in Figure 4.1.

Players receive a payoff of 1 for a win, -1 for a loss, and 0 for a tie. In addition, both players receive .5 when adjacent moves are played and lose .5 when non-adjacent moves

are played. The game has a unique Nash equilibrium at $s_A = s_B = (.2, .2, .2, .2, .2)$. In this game players have conflicting interests but some cooperation is possible, which allows us to distinguish between a best-responding opponent and a worst case outcome. This distinction highlights the properties of our algorithm, which is why we use Rock/Spock/Paper/Lizard/Scissors for a demonstration game.

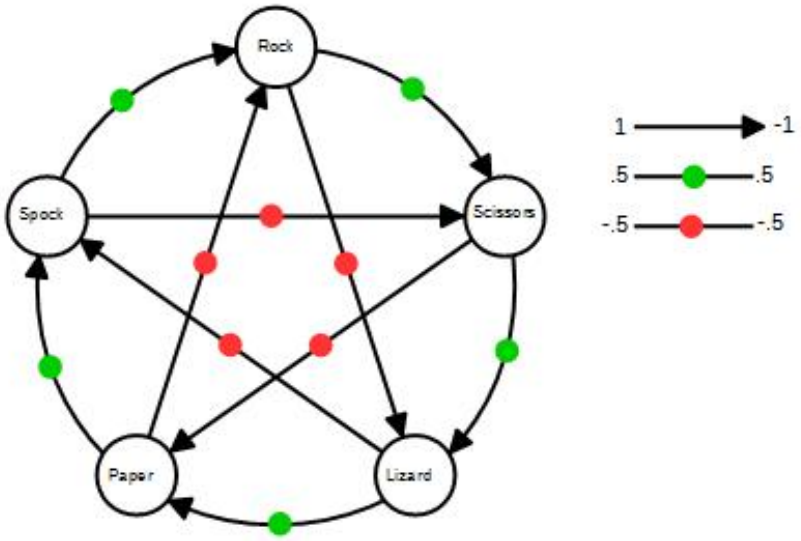

Figure 4.1: Precedence relationships in Rock/Spock/Paper/Lizard/Scissors. Arrows point from winning moves to losing moves. Green dots indicate adjacent moves which receive a bonus. Red dots indicate non-adjacent moves which receive a penalty.

4.2 Background

4.2.1 Safe Policy Selection.

SPS [53] is a method of deciding how much to risk against a potentially stronger opponent. Given a game G with safety value V_G^A, an r-safe strategy s_A^r is one whose worst case payoff is within r of the safety value

$$\min_{s_B \in \triangle^{n_B - 1}} U(s_A^r, s_B) \geq V_G^A - r.$$

SPS selects the r-safe strategy with the best performance against the prediction. Over a series of games SPS adjusts r values to guarantee a payoff close to the value of the game, while also performing well against predictable opponents. It does this by setting r_n (the r value to use in round n) according to $r_n = r_{n-1} + 1/n + U_A(s_A^{n-1}, m_B^{n-1}) - V_G^A$ where s_A^n is the agent's mixed strategy in round n and m_B^n is the opponent's move in round n. Results in Rock/Paper/Scissors demonstrate that SPS can improve the performance of weak players against stronger players.

4.2.2 Restricted Nash Response.

RNR [44, 8] exploits a prediction of opponent behavior in a zero-sum game while avoiding exploitation. It finds a strategy by constructing a modified game, and taking the Nash equilibrium of that game. Results in poker demonstrate that it is possible to find strategies which are very effective against the prediction while remaining resistant to exploitation.

To calculate a RNR to a prediction $s_B \in \triangle^{n_B - 1}$ in a zero-sum game G using a weight $w \in [0, 1]$ construct a modified game G' with $M'_A = M_A$, $M'_B = M_B$, $U'_B = U_B$, and $U'_A(m_A, m_B) = w \times U_A(m_A, s_B) + (1 - w) \times U_A(m_A, m_B)$. RNR returns the Nash equilibrium of G'. Note that although G' is not zero-sum equilibrium selection is not a

problem because, as we will show later, all equilibria of G' are interchangeable.

Although RNR and SPS are generated by different procedures, the set of strategies generated by RNR is a subset of the set of strategies generated by SPS. Johanson et al. [44] demonstrate that for any w value, there will always be an r value for which SPS produces the same strategy as RNR.

We will show that the general-sum modified game created to calculate a RNR for a zero-sum game has a unique Nash equilibrium. We consider two equilibria distinct if each player strictly prefers to play their equilibrium strategy in each equilibrium: $U_A(s_A, s_B) > U_A(s'_A, s_B)$, $U_A(s'_A, s'_B) > U_A(s_A, s'_B)$, $U_B(s_A, s_B) > U_B(s_A, s'_B)$, and $U_B(s'_A, s'_B) > U_B(s'_A, s_B)$. (If the preference is weak then the two equilibria are part of the same connected component and players can play either strategy and achieve the same payoff.)

Theorem 1. *Given a zero-sum game G with utility functions $U_A = U$ and $U_B = -U$ let G' be the modified game created to calculate a RNR to a prediction p, with utility function U' where $U'_A(m_A, m_B) = w \times U(m_A, p) + (1-w) \times U(m_A, m_B)$ and $U'_B = -U$. G' doesn't have two distinct equilibria.*

Proof. Assume G' has two distinct Nash equilibria s and s'. Construct a new game G'' from G' with moves $s_A, s'_A \in \triangle^{n_A - 1}$ and $s_B, s'_B \in \triangle^{n_B - 1}$ and payoffs equal to playing the corresponding strategies in G'. Because s and s' are distinct, we have:

$$\begin{aligned} w \times U(s_A, p) + (1-w) \times U(s_A, s_B) > \\ w \times U(s'_A, p) + (1-w) \times U(s'_A, s_B) \end{aligned} \tag{4.1}$$

$$\begin{aligned} w \times U(s'_A, p) + (1-w) \times U(s'_A, s'_B) > \\ w \times U(s_A, p) + (1-w) \times U(s_A, s'_B) \end{aligned} \tag{4.2}$$

$$-U(s'_A, s'_B) > -U(s'_A, s_B) \tag{4.3}$$

$$-U(s_A, s_B) > -U(s_A, s'_B) \tag{4.4}$$

From 4.1 and 4.3 we get:

$$\begin{aligned} w \times U(s_A, p) + (1-w) \times U(s_A, s_B) > \\ w \times U(s'_A, p) + (1-w) \times U(s'_A, s'_B) \end{aligned} \tag{4.5}$$

From 4.5 and 4.2 we get:

$$\begin{aligned} w \times U(s_A, p) + (1-w) \times U(s_A, s_B) > \\ w \times U(s_A, p) + (1-w) \times U(s_A, s'_B) \end{aligned} \tag{4.6}$$

From 4.6 and 4.4 we get:

$$\begin{aligned} w \times U(s_A, p) + (1-w) \times U(s_A, s_B) > \\ w \times U(s_A, p) + (1-w) \times U(s_A, s_B) \end{aligned} \tag{4.7}$$

This is not possible, so there cannot be two distinct Nash equilibria in the modified game created for RNR. □

4.3 Extension to General-Sum Games

4.3.1 Safe Policy Selection.

SPS was developed for zero-sum games, but it can be extended to general-sum games by treating the value of the game as the amount which the player can guarantee for itself, regardless of the actions of the opponent. In general-sum games SPS may not be an effective method of ameliorating risk. Consider a game in which an opponent has a punishing move which causes the agent to receive a bad outcome regardless of the action the agent chooses. In this case, regardless of the risk value chosen, the algorithm

will play a best-response to the prediction, because it will do no worse than any other strategy if the opponent selects the punishing move. We don't use SPS in such games.

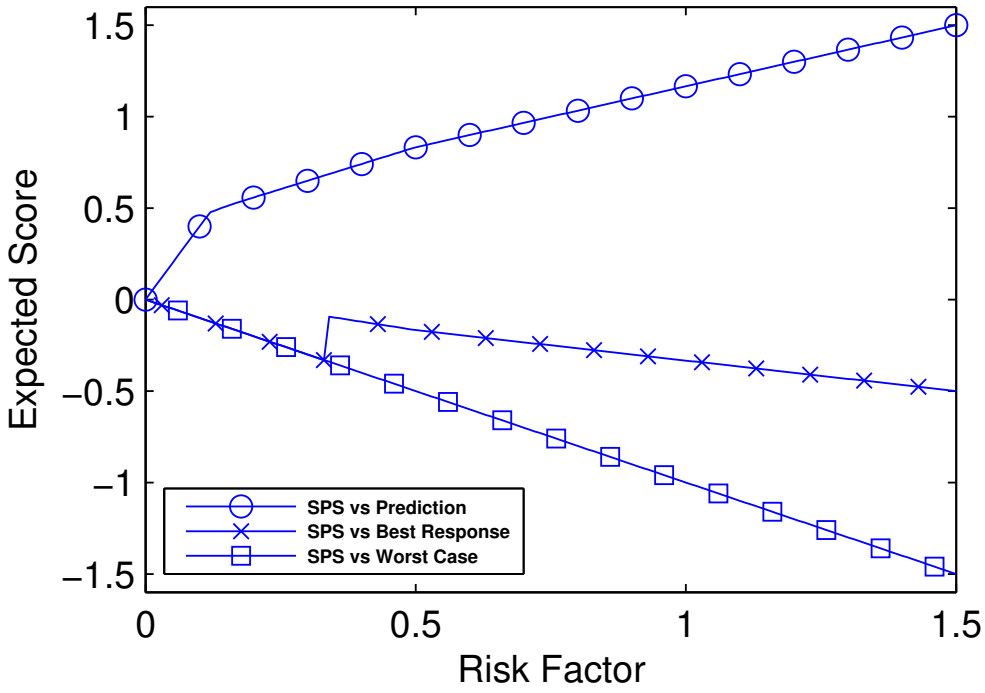

Figure 4.2: Payoffs of r-safe strategies with a prediction of Rock in the game Rock/-Spock/Paper/Lizard/Scissors for different values of r.

Figure 4.2 shows how the r value affects the performance of SPS in Rock/Spock/-Paper/Lizard/Scissors. Performance is measured against the prediction, against a best responding opponent which maximizes its own payoff given the agent's strategy, and against a worst case opponent which minimizes the agent's payoff given the agent's strategy. When $r = 0$ the generated strategy is the maximin strategy $(.2, .2, .2, .2, .2)$. When $r = 1.5$ the generated strategy is $(0, 1, 0, 0, 0)$, which is the best response to the prediction. Intermediate values cause the generated strategy to vary continuously between those two extremes.

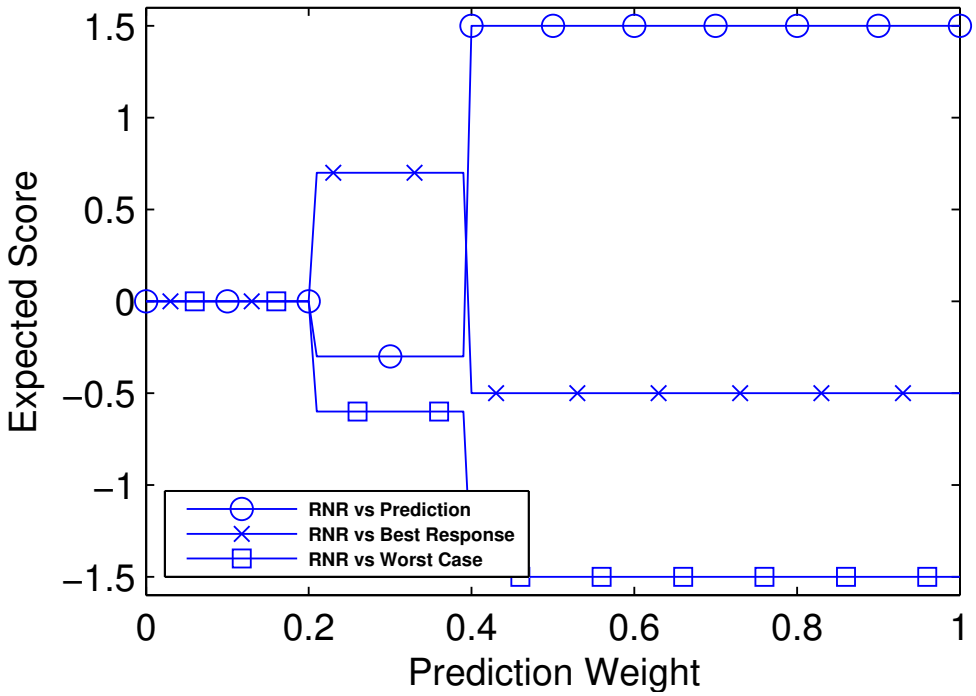

Figure 4.3: Payoffs of RNR to a prediction of Rock in Rock/Spock/Paper/Lizard/-Scissors for different values of w. The chosen strategy changes at weight values .2 and .4, where the changing parameter disrupts the current equilibrium.

4.3.2 Restricted Nash Response

RNR can be extended to general sum games by providing a method of selecting an equilibrium when there are multiple equilibria, such as choosing the equilibria with the highest payoff. RNR does not generate multiple equilibria for Rock/Spock/Paper/-Lizard/Scissors. Figure 4.3 shows the effect of the w parameter. The agent predicts Rock and performance is measured against the prediction, against a best responding opponent which maximizes its own payoff given the agent's strategy, and against a worst case opponent which minimizes the agent's payoff given the agent's strategy. When $w = 1$, RNR will play a best response to the prediction, When $w = 0$, RNR will play a Nash equilibrium of the original game. Intermediate values will cause the strategy to

abruptly change when increasing the w value prevents the opponent from playing its strategy in the current equilibrium. In general-sum games, increasing the weight value can reduce performance against the prediction when the opponent strategy in the new equilibrium is beneficial to the agent.

4.4 Restricted Stackelberg Response with Safety

We define the Restricted Stackelberg Response with Safety (RSRS) for player A in game G with prediction $p \in \triangle^{n_B-1}$, prediction weight $w \in [0,1]$, and risk factor $r \in \mathbb{R}+$ to be the mixed strategy for player A that maximizes its expected payoff given the assumption that, with probability w, the opponent will play according to the prediction p and, with probability $1-w$, it will best-respond to the agent, subject to the constraint that its expected payoff when played against any opponent action is at least $V_G^A - r$. The Restricted Stackelberg Response (RSR) is identical except that it has no constraint using r (it has no safety factor).

The RSRS is calculated by constructing a new payoff function for player A which reflects the assumption that the opponent will play the prediction with probability w:

$$U'_A(m_A, m_B) = w \times U_A(m_A, p) + (1-w) \times U_A(m_A, m_B)$$

The RSRS for player A is the probability distribution $s_A \in \triangle^{n_A-1}$, which maximizes the expected value of U'_A under the assumption that player B will best respond, subject to the constraint

$$U_A(s_A, m_B) \geq V_G^A - r \text{ for all } m_B \in M_B.$$

Assuming that the opponent is best-responding to the action of player A is equivalent to designating player A as a Stackelberg leader. The game does not have a Stackelberg

leader; that assumption is a convenient way to handle the possibility of a best-responding opponent.

We compute the RSRS using a modification of the technique in [14]. For each opponent action we find a mixed strategy to which the opponent action is a best response and which satisfies the safety value constraint. Then we select the option which performs best against a weighted combination of the prediction and a best responding opponent. More formally, for each $m_B \in M_B$ maximize over $s_A^{m_B} \in \triangle^{n_A-1}$:

$$s_A^{m_B} = \text{argmax}_{s_A \in \triangle^{n_A-1}} U_1'(s_A, m_B)$$

$$\text{subject to:} \quad \forall m_B' \in M_B, U_B(s_A^{m_B}, m_B) \geq U_B(s_A^{m_B}, m_B')$$

$$\text{and} \quad \forall m_B' \in M_B, U_A(s_A^{m_B}, m_B') \geq V_G^A - r$$

Solving this set of equations for each opponent action will give us at least 1 and up to n mixed strategies for player A. The mixed strategy s_A with the highest expected value in U_A' against the opponent's best response is the RSRS. The complexity of calculating the RSRS is polynomial in the number of moves in the game.

The values chosen for probability weight (w) and risk factor (r) control the trade off between performance against the prediction, performance against a best-responding opponent (w), and performance against a worst-case opponent (r).

For a fixed risk factor there are many probability weights that produce the same strategy; changes in w either produce no change in strategy or a discontinuous jump to a new strategy. Jumps occur when confidence in the prediction becomes high enough to justify the additional vulnerability to a best-responding opponent. In contrast, changes to r produce a continuous variation between a minimax strategy and a prediction exploiting strategy. The effect of r dominates the effect of w; if $r = 0$, the w value has no

effect. This allows us to provide the same guarantees as SPS by selecting a value for r as SPS does. We select a value for w by calculating the relative posterior probability of the opponent following the prediction and the opponent best-responding.

Note that the way r dominates w is an effect of how we have chosen to define those parameters. It is instructive to consider an alternative in which w dominates r; all that is necessary is to alter the constraint involving r to:

$$\forall m'_B \in M_B, U'_A(s_A, m'_B) \geq V_G^A - r$$

This guarantees that the loss will be less than r in the modified game instead of the original game. We have chosen not to calculate Restricted Stackelberg Response with Safety this way because it eliminates the performance guarantee provided by selecting r values according to Safe Policy Selection.

Figure 4.4 shows the effects of w on performance, which is measured against the prediction, against a best responding opponent which maximizes its own payoff given the agent's strategy, and against a worst case opponent which minimizes the agent's payoff given the agent's strategy. $w < .6$ produces a Stackelberg equilibrium, $w > .6$ produces a best response to the prediction. The strategy abruptly changes at .6 because at that point a threshold is passed where the increased payoff against the prediction justifies a reduced payoff against a best-responding opponent. Like SPS, r produces a continuous variation of performance (see Figure 4.2), ranging from the maximin strategy ($r = 0$) to a best response ($r = 1.5$).

We can characterize the change in performance produced by a change in w for a fixed r in terms of the trade-off between performance against the prediction and performance against a best-responding agent. For example, in Rock/Spock/Paper/Lizard/Scissors, with a prediction of Rock and $r = 1.5$ RSRS produces $(0, .6, 0, 0, .4)$ when $w < .6$ and

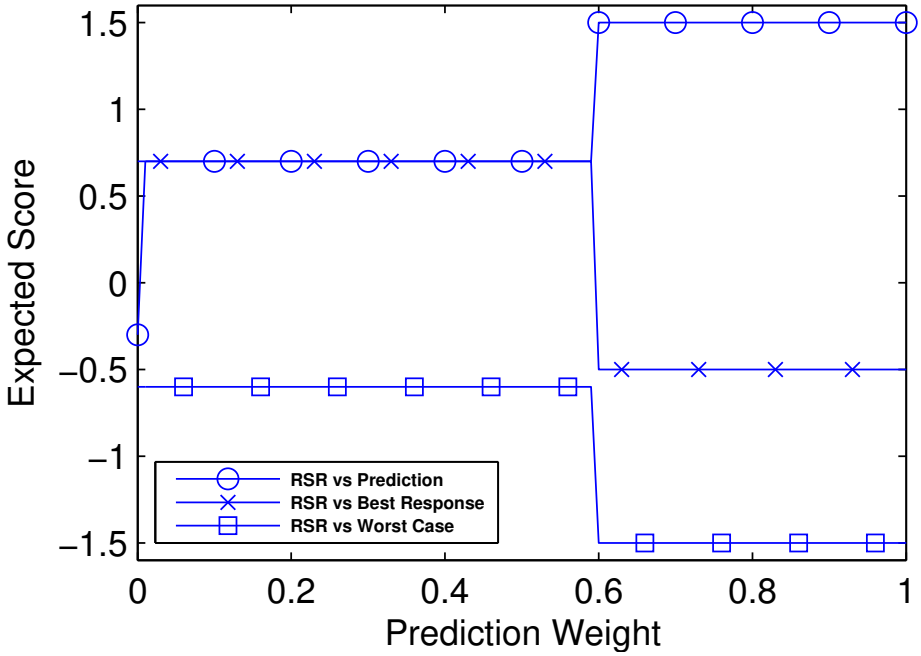

Figure 4.4: Payoffs of RSRS to a prediction of Rock in the game Rock/Spock/Paper/-Lizard/Scissors with w varying from 0 to 1, and r fixed at 1.5.

$(0, 1, 0, 0, 0)$ when $w \geq .6$. The first strategy produces a payoff of .7 when played against the prediction or a best-responding opponent. The second equation produces a payoff of 1.5 when played against the prediction, and a payoff of $-.5$ when played against a best-responding opponent.

When w changes from below .6 to above .6 RSRS gains .8 in expected payoff against the prediction and loses 1.2 against a best-responding opponent. The expected gain is 2/3 as much as the expected loss. This matches .4/.6, the relative probability of those events expressed by a w value of .6. This relationship holds for any game and value of w.

We are interested in values of w between two regions where the RSRS for a fixed r does not change. For those w values we denote with $rsrs_{w+}$ and $rsrs_{w-}$ respectively

the RSRS for the region with weight values higher or lower than w. We denote with br_{w+} and br_{w-} the best responses to those strategies. Assume we are given a game G, a prediction $p \in \triangle^{n_B-1}$, and a w where the RSRS changes. If there is a δ such that for all $0 < \epsilon < \delta$ the RSRS with weight $w + \epsilon$ is the same for all ϵ, and the RSRS with weight $w - \epsilon$ is the same for all ϵ, and $rsrs_{w+} \neq rsrs_{w-}$, we will prove:

Theorem 2. *The ratio of the performance gain against the prediction to the performance loss against a best-responding opponent is* $\frac{1-w}{w}$*, i.e.,*

$$\frac{U_A(rsrs_{w+}, p) - U_A(rsrs_{w-}, p)}{U_A(rsrs_{w-}, br_{w-}) - U_A(rsrs_{w+}, br_{w+})} = \frac{1-w}{w}$$

We will begin by showing that reducing w can only improve performance against a best-responding opponent. This may seem obvious, but it doesn't hold for RNR in general-sum games.

Lemma 1. $U_A(rsrs_{w+}, br_{w+}) < U_A(rsrs_{w-}, br_{w-})$

Proof. Consider the quantities $U_A(rsrs_{w+}, p) - U_A(rsrs_{w-}, p)$ and $U_A(rsrs_{w+}, br_{w+}) - U_A(rsrs_{w-}, br_{w-})$, which represent the performance gain of $rsrs_{w+}$ relative to $rsrs_{w-}$ against the prediction and against a best-responding opponent respectively. If both are positive or both are negative then $rsrs_{w+}$ or $rsrs_{w-}$ would be strictly superior to the other, which contradicts the fact that that they were generated as payoff-maximizing distributions.

Let $U^{w+\epsilon}$ be the utility function for player A in the modified game created with prediction p and weight $w + \epsilon$. Because $rsrs_{w+}$ was found by maximizing performance in $U^{w+\epsilon}$ against a best-responding opponent, we know that $U^{w+\epsilon}(rsrs_{w+}, br_{w+}) >$

$U^{w+\epsilon}(rsrs_{w-}, br_{w-})$. From the definition of $U^{w+\epsilon}$ this gives us

$$\begin{aligned}(w+\epsilon)U_A(rsrs_{w+}, p) + (1-w-\epsilon)U_A(rsrs_{w+}, br_{w+}) > \\ (w+\epsilon)U_A(rsrs_{w-}, p) + (1-w-\epsilon)U_A(rsrs_{w-}, br_{w-})\end{aligned} \tag{4.8}$$

Similarly, for $rsrs_{w-}$ we have

$$\begin{aligned}(w-\epsilon)U_A(rsrs_{w-}, p) + (1-w+\epsilon)U_A(rsrs_{w-}, br_{w-}) > \\ (w-\epsilon)U_A(rsrs_{w+}, p) + (1-w+\epsilon)U_A(rsrs_{w+}, br_{w+})\end{aligned} \tag{4.9}$$

We can manipulate Eq. 4.8 to get

$$\begin{aligned}&(w-\epsilon)U_A(rsrs_{w+}, p) + (1-w+\epsilon)U_A(rsrs_{w+}, br_{w+}) \\ &+2\epsilon((U_A(rsrs_{w+}, p) - U_A(rsrs_{w-}, p)) \\ &-(U_A(rsrs_{w+}, br_{w+}) - U_A(rsrs_{w-}, br_{w-}))) \\ &> (w-\epsilon)U_A(rsrs_{w-}, p) + (1-w+\epsilon)U_A(rsrs_{w-}, br_{w-})\end{aligned}$$

For this and Eq. 4.9 to be true, we must have

$$\begin{aligned}2\epsilon((U_A(rsrs_{w+}, p) - U_A(rsrs_{w-}, p)) > \\ (U_A(rsrs_{w+}, br_{w+}) - U_A(rsrs_{w-}, br_{w-})))\end{aligned} \tag{4.10}$$

We know that $U_A(rsrs_{w+}, br_{w+}) - U_A(rsrs_{w-}, br_{w-})$ and $U_A(rsrs_{w+}, p) - U_A(rsrs_{w-}, p)$ have different signs. If the first term is positive and the second negative, then Eq. 4.10 will be false, so it must be the case that $U_A(rsrs_{w+}, p) - U_A(rsrs_{w-}, p)$ is positive and $U_A(rsrs_{w+}, br_{w+}) - U_A(rsrs_{w-}, br_{w-})$ is negative. □

Using Lemma 1, we can prove Theorem 2.

Proof. $rsrs_{w+}$ is calculated by maximizing payoff in the modified game, so

$$U_A^{w+\epsilon}(rsrs_{w+}, br_{w+}) > U_A^{w+\epsilon}(rsrs_{w-}, br_{w-})$$

where $U_A^{w+\epsilon}(rsrs_{w+}, br_{w+})$ is the expected value of playing $rsrs_{w+}$ against a best-responding opponent in the modified game $U^{w+\epsilon}$. Similarly

$$U_A^{w-\epsilon}(rsrs_{w-}, br_{w-}) > U_A^{w-\epsilon}(rsrs_{w+}, br_{w+})$$

From how U^w is constructed we have

$$\begin{aligned}(w+\epsilon)U_A(rsrs_{w+}, p) + (1-w-\epsilon)U_A(rsrs_{w+}, br_{w+}) > \\ (w+\epsilon)U_A(rsrs_{w-}, p) + (1-w-\epsilon)U_A(rsrs_{w-}, br_{w-})\end{aligned}$$

$$\begin{aligned}(w-\epsilon)U_A(rsrs_{w-}, p) + (1-w+\epsilon)U_A(rsrs_{w-}, br_{w-}) > \\ (w-\epsilon)U_A(rsrs_{w+}, p) + (1-w+\epsilon)U_A(rsrs_{w+}, br_{w+})\end{aligned}$$

By rearranging terms we have:

$$\frac{U_A(rsrs_{w+}, p) - U_A(rsrs_{w-}, p)}{U_A(rsrs_{w-}, br_{w-}) - U_A(rsrs_{w+}, br_{w+})} > \frac{1-w-\epsilon}{w+\epsilon}$$

and

$$\frac{U_A(rsrs_{w+}, p) - U_A(rsrs_{w-}, p)}{U_A(rsrs_{w-}, br_{w-}) - U_A(rsrs_{w+}, br_{w+})} < \frac{1-w+\epsilon}{w-\epsilon}$$

These last two equations provide the lower and upper bounds. By taking the limit as $\epsilon \longrightarrow 0$ we prove the theorem. □

4.5 Learning Weight Values

An agent using RSRS to generate strategies needs to select values for the parameters. The risk factor can be selecting using the same method as SPS, but there isn't a comparable method to select the prediction weight. RSRS assumes that the opponent will best respond if the prediction is incorrect, so to compute w we estimate the relative probabilities that the opponent played according to the prediction or played a best response in previous rounds. Computing the probability the opponent has played according to the prediction is trivial. A naive method of estimating the probability that the opponent played a best response would assign a probability of 1 or 0 (either the opponent played a best response in every previous game or not). This is easy to deceive – for example, an opponent which consistently plays the second-best response would not be considered to be best-responding.

We adopt a model in which the opponent plays according to an exponential response function to their expected payoff against the agent's chosen strategy. Under this model, it is more likely for the opponent to play moves which perform well against the agent's strategy, but it is still possible for them to play sub-optimal moves. Given an agent strategy $s_A \in \triangle^{n_A-1}$, the opponent's exponentially weighted response is

$$\mathbb{P}(m_B^i) = \frac{e^{\lambda U_B(s_A, m_B^i)}}{\sum_{m_B^j \in M_B} e^{\lambda U_B(s_A, m_B^j)}}$$

where λ describes how responsive the opponent is to higher payoffs. $\lambda = 0$ describes an opponent that plays uniformly at random. The higher the λ value, the stronger the opponent's preference for higher expected payoff moves.

We calculate λ value being used by the opponent by finding the value with the maximum likelihood for the prior actions of the opponent. If all the observations have been best responses, the probability maximizing value will be ∞, so we introduce a

smoothing observation (see Algorithm 2). We then use λ to compute the probability that a best-responding opponent played the observed move. This allows us to compute the relative probability of a best-responding opponent vs. an opponent playing according to the prediction. We use that value to determine the probability weight to use with RNR and RSRS (see Algorithm 2).

Algorithm 2 Estimate relative probability of prediction and best-responding opponent

Initialize $t = 1$ ▷ Current Round
Initialize $\{chosen_1 = .9\}$ ▷ List of values the opponent has chosen
Initialize $options_1 = \{(0, .9, 1)\}$ ▷ List of sets the opponent has chosen values from
Initialize $P(Prediction) = .5$, $P(BestResponse) = .5$ ▷ Initial estimate
while the game continues **do** ▷ Make observations from the previous round
    Set $s'_A \in \triangle M_A$, the strategy played by the agent
    Set $s'_B \in \triangle M_B$, the predicted strategy for the opponent
    Set $m'_B \in M_B$, the observed opponent move
    Set $u'_B = U_B(s'_A, m'_B)$, the opponents expected utility
    ▷ Calculate the expected payoff of the opponent's moves
    for $m^i_B \in M_B$ **do**
        Set $u^i_B = U_B(s'_A, m^i_B)$
    end for
    Increment t
    Set $chosen_t = u'_B$
    Set $options_t = u^1_B..u^n_B$
    Find λ maximizing $\prod_{i=1}^{t} \frac{e^{\lambda chosen_i}}{\sum_{j=1}^{n} e^{\lambda options_{i,j}}}$ using gradient descent
    ▷ Update the estimated probabilities
    $P(Prediction) = P(Prediction) \times s'_B(m'_B)$
    $P(BestResponse) = P(BestResponse) \times \frac{e^{\lambda u'_B}}{\sum_{i=1}^{n} e^{\lambda u^i_B}}$
    Renormalize $P(Prediction)$ and $P(BestResponse)$
    Set prediction weight to $\frac{P(Prediction)}{P(Prediction)+P(BestResponse)}$
end while

4.6 Results

We report results obtained in Rock/Spock/Paper/Lizard/Scissors, Traveller's Dilemma, and a simple pursuit/evasion game. In more competitive games all approaches are

broadly successful since a best-responding opponent behaves like a worst-case opponent. In more cooperative games, using the Stackelberg equilibrium provides a significant gain against best-responding opponents.

In the experiments agents play a sequence of 100 games. Results are averaged over 100 repetitions. We show the performance of six approaches:

1. a best response to the prediction,
2. SPS,
3. RNR using the calculated weight,
4. RSR using the calculated weight (with no risk factor),
5. RSRS using the calculated weight and SPS to determine risk factors, and
6. GIGA-WolF.

All, except GIGA-WolF, predict the opponent using fictitious play: they assume a stationary opponent playing a distribution drawn from a uniform Dirichlet distribution, and predict using the expected value of that distribution given the observed moves. This flawed prediction technique is chosen to show that RSRS can handle inaccurate predictions.

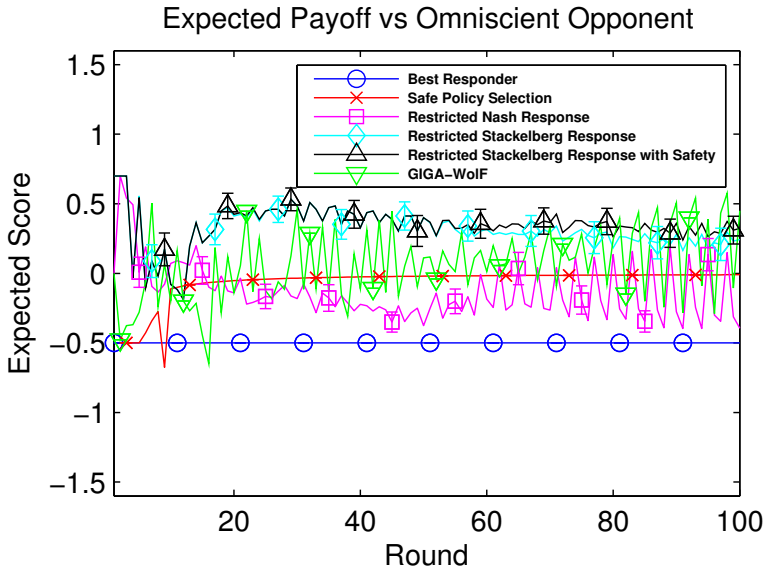

Figure 4.5: Expected payoff of a player playing different strategies in the game Rock/Spock/Paper/Lizard/Scissors against a best-responding opponent, over 100 games. Error bars show 95% confidence interval.

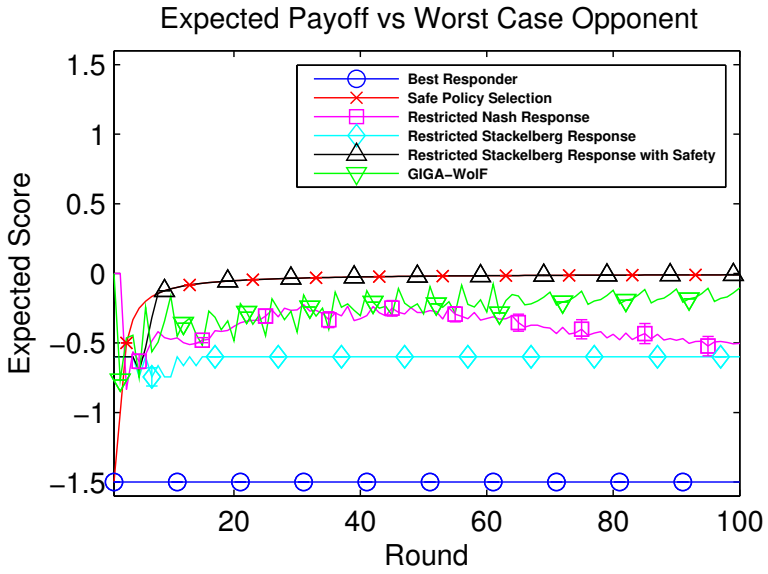

Figure 4.6: Expected payoff of a player playing different strategies in the game Rock/Spock/Paper/Lizard/Scissors against a worst case opponent, over 100 games.

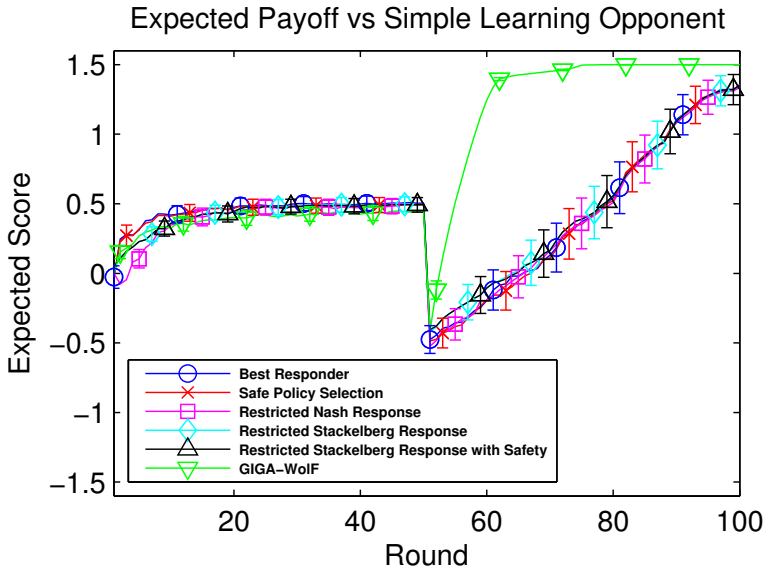

Figure 4.7: Expected payoff of a player playing different strategies in the game Rock/Spock/Paper/Lizard/Scissors against a simple learning opponent, over 100 games. Error bars show 95% confidence interval.

Figure 4.5 shows the performance of the six strategies against a best-responding opponent which knows the agent's mixed strategy and plays to maximize its own payoff. Approaches based on the Stackelberg response perform well; it takes 10-20 observations to learn that the opponent is best-responding, after which the agent takes advantage of that trait. SPS treats the situation as worst case and achieves the value of the game. GIGA-WolF is worse than RSRS and RSR, but still outperforms the safety value. RNR uses the same weight value as RSRS but achieves a worse outcome. Both RSR and RNR are trying to find a strategy which performs well when the opponent is best-responding, but RSR achieves a better outcome because it does not require its own strategy to be a best-response to the opponent strategy.

Figure 4.6 shows the performance of the six strategies against a worst-case opponent which knows the agent's mixed strategy and plays to minimize the agent's payoff. Approaches which include SPS quickly detect a worst-case opponent and play the maximin

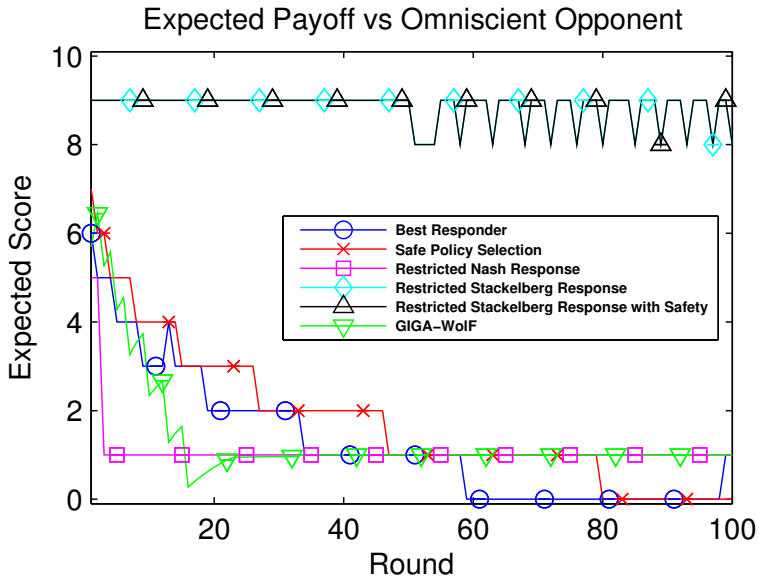

Figure 4.8: Expected payoff in Traveller's Dilemma against a best-responding opponent, over 100 games.

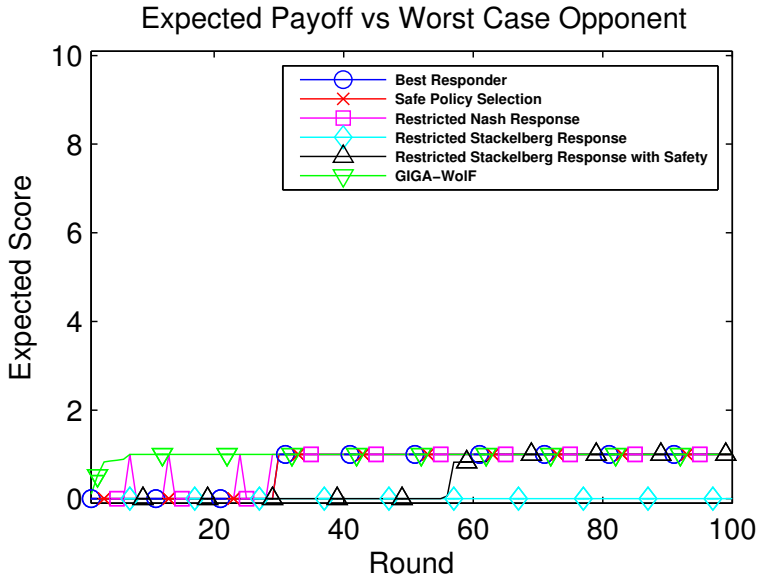

Figure 4.9: Expected payoff in Traveller's Dilemma against a worst case opponent, over 100 games.

strategy (which is the best possible outcome in this situation). Of the agents that don't use SPS, GIGA-WolF approaches the safety value, and RNR does better than RSR (without safety) because assuming a best-responding opponent is inaccurate.

Figure 4.7 shows the performance of the six strategies against a simple learning opponent which observes for 50 rounds, and then switches to playing a best response to the strategy the agent played in the first 50 rounds. This opponent reveals what happens when the predictor adjusts to a changing opponent more rapidly than the parameter values. 50 rounds of observations of successful play against the initial strategy builds up a very high risk factor, and a high certainty that the opponent is not best-responding. As a result strategies dependent on those parameters don't adjust to the new strategy until the predictor does. GIGA-WolF reacts faster to the switching opponent because it doesn't use fictitious play as a predictor.

Traveller's Dilemma is a general-sum game in which both players choose a payoff from 1 to 10. Each player receives the lowest payoff, and if one player chose a lower payoff than the other, that player receives a bonus of 1, while the other receives a penalty of 1. The Nash equilibrium of Travellers Dilemma is for both players to chose the minimum payoff, but the social welfare maximizing strategy is for them both to chose the maximum payoff.

Figures 4.8 and 4.9 shows the performance of best-responding, SPS, RNR, RSR, and RSRS in the game Traveller's Dilemma. Against an omniscient best-responding opponent methods which use the Stackelberg equilibrium perform well, SPS, GIGA-WolF, and simple best-response gradually converge to the Nash equilibrium, and unsurprisingly, RNR rapidly arrives at the Nash equilibrium. Against the worst case opponent, all agents, except RSR, eventually arrive at the Nash equilibrium (which is also the minimax strategy).

Figures 4.10-4.13 show the performance of best-responding, GIGA-WolF, SPS, RNR,

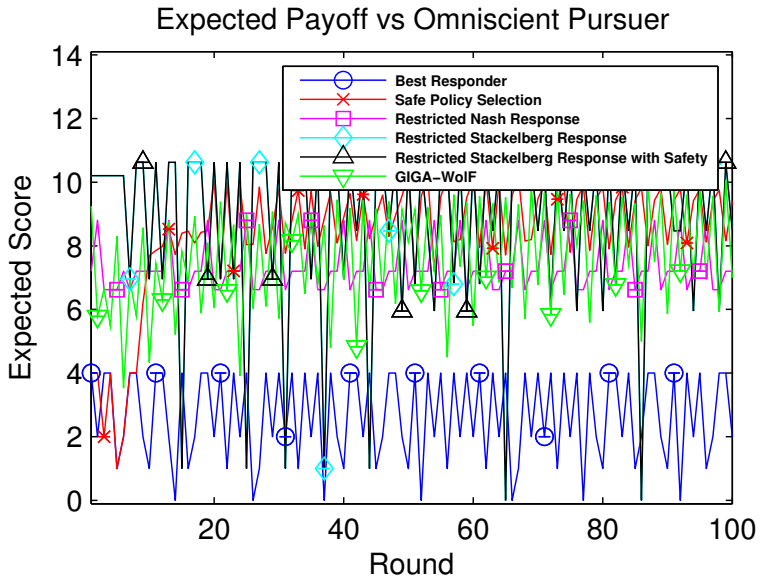

Figure 4.10: Expected payoff of the evader in the pursuit/evasion game against a best-responding opponent, over 100 games.

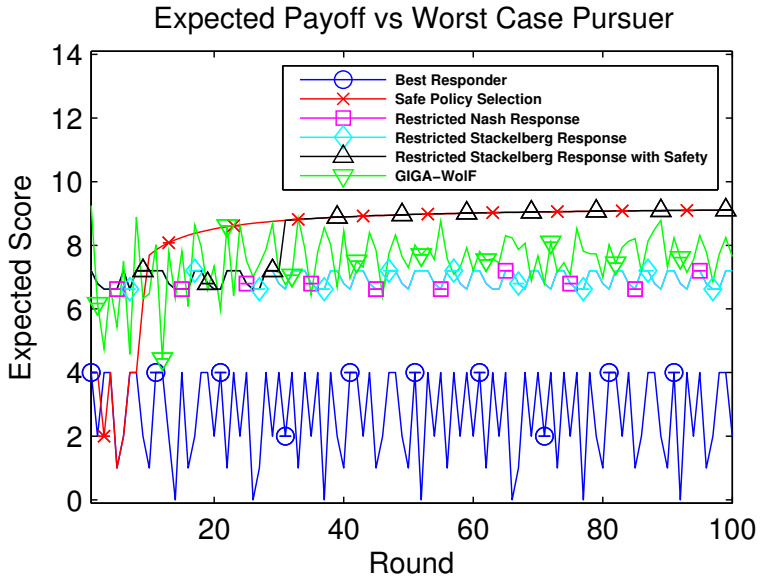

Figure 4.11: Expected payoff of the evader in the pursuit/evasion game against a worst case opponent, over 100 games.

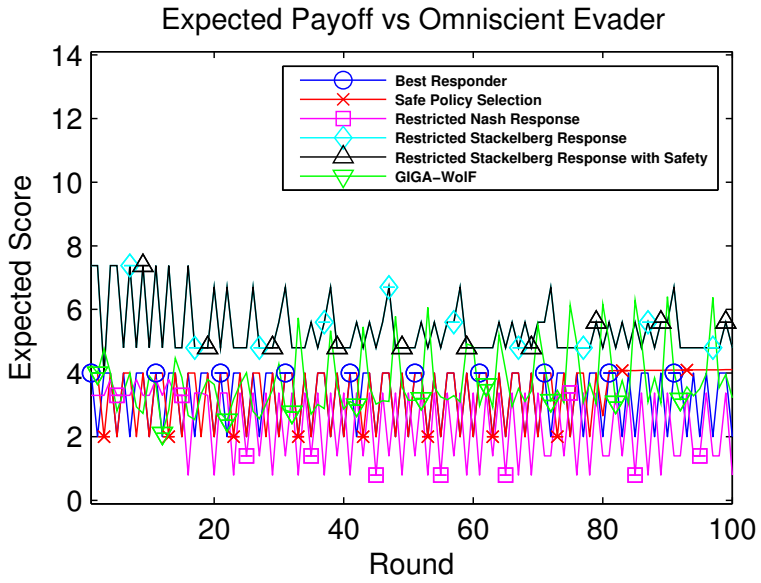

Figure 4.12: Expected payoff of the pursuer in the pursuit/evasion game against a best-responding opponent, over 100 games.

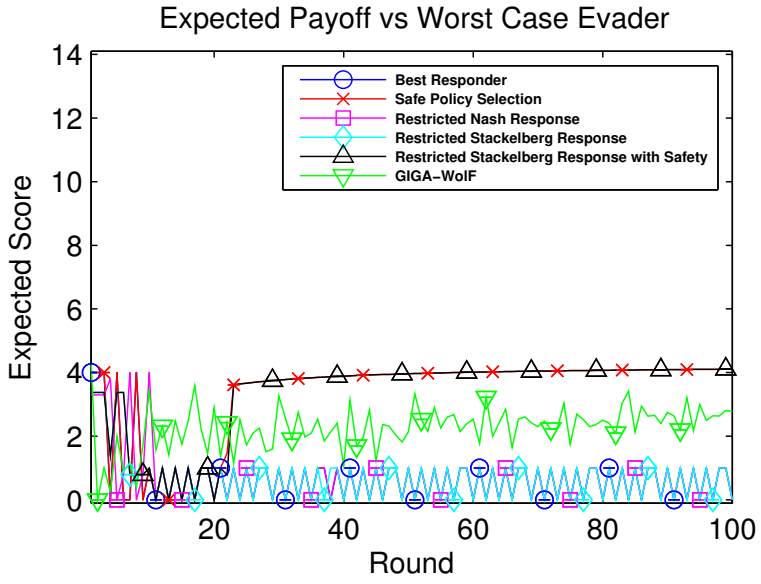

Figure 4.13: Expected payoff of the pursuer in the pursuit/evasion game against a worst case opponent, over 100 games.

RSR, and RSRS in a simple pursuit/evasion game. There are 4 locations, with associated payoffs from 0 to 4. Each player receives the payoff of the location they chose. In addition the pursuer receives a payoff of 10 for choosing the same location as the evader, while the evader receives a payoff of 10 for choosing a different location from the pursuer. The Nash equilibrium of the game is $(.075, .175, .275, .475)$ for the pursuer (checking preferred locations more often), and $(.425, .325, .225, .025)$ for the evader (staying away from the preferred locations).

An evader which best-responds to its prediction does particularly poorly regardless of the opponent (Figures 4.10 and 4.11). For all agents there is considerable instability because minor changes in the prediction can result in relatively large changes in the response. Against a worst case opponent (Figures 4.11 and 4.13), agents which use a risk factor parameter perform best. Note that RSRS is slower to reach the minimax strategy because its performance prior to reaching that strategy is better, so it takes longer for the initial risk factor to deplete. A pursuer is best-off using RSR or RSRS against a best-responding opponent. Against a worst-case opponent, RSRS and SPS are the only strategies which find the minimax solution.

4.7 Conclusions

We have presented RSRS, a new method for choosing a strategy in a general-sum normal form game. that takes advantage of a prediction of opponent behavior while guarding against exploitation, and shown experimentally that it is effective. RSRS provides a useful basis for acting on a prediction in a general-sum environment.

RSRS deals well with two dangerous opponent types when the prediction is inaccurate, but there are other possible opponents. For example, an opponent which is mildly hostile to the agent will restrict RSRS to the value of the game, but a better strategy could be to offer such an opponent a higher incentive to cooperate. Future work will

explore a larger variety of opponents, making more general assumptions about their behavior.

Chapter 5

The Gift Exchange Game

We have been looking at the problem of repeatedly playing against an opponent in a normal form game. We can create a reciprocating strategy by defining cooperation as adopting a particular attitude and belief and playing the resulting Nash equilibrium. However this reciprocating strategy, while effective, is inflexible in terms of the relative benefits to each of the agent, and it requires significant analysis of the underlying games. In this chapter we will look at the problem of influencing the opponent's behavior more generally; we will present a simplified environment which allows us to focus on the problem of how to influence the opponent's behavior in isolation.

It is valuable for the agent to influence the behavior of its opponent. In games where the interests of the players are diametrically opposed, this is not possible, because the opponent will act in strict opposition to the agent, so the agent should choose its strategy by solving or approximating the Nash equilibrium. However, in general-sum games it may be possible to influence the behavior of the opponent to increase the payoff to the agent.

If the opponent is following a sufficiently simple strategy it may be possible for the agent to calculate an optimal response to that strategy. However, in cases where

the opponent can be as complex as the agent, explicitly modelling the opponent and finding a best response is generally computationally infeasible, unless the opponent has been constructed to provide a simple best response [60]. Because it is not possible to best respond to arbitrary opponents, we will attempt to construct strategies that can respond effectively to self-interested opponents.

Reciprocation is one way to motivate the opponent to select the actions the agent prefers. The main difficulty in using reciprocation to motivate opponents to select preferred actions is the problem of identifying preferred actions, both for the opponent (so the agent can identify when reciprocation is appropriate) and the agent (so the agent can identify how to reciprocate). The complexity of the game is the largest factor in identifying preferable options. In many games it is difficult to identify exactly how a specific action will affect the outcomes for the agent and the opponent; this makes it difficult to determine whether the opponent was attempting to cooperate. Another complicating factor present in many games is simultaneous actions. Simultaneous actions introduce two problems: predicting the effect of the agent's actions now requires a prediction of the opponent's next action as well, and secondly, while it may be possible to estimate the opponent's intent from their action, their intent may change as a result of the agent's most recent action. Finally, it is clear that rational agents should prefer pareto-optimal outcomes, but it is not clear how agents should select a particular pareto-optimal outcome when there are multiple pareto-optimal outcomes available.

5.1 Gift Exchange Game

To explore how an agent's action may affect the opponent's future actions, we have developed the *Gift Exchange* game. In the game, agents take turns choosing actions, where each action consists of a choice of outcomes from a set of potential outcomes. Each outcome is an assignment of (potentially negative) payoffs to the agent and to

its opponent. We will focus on Gift Exchange games where the set of outcomes is the unit circle; one player receives the x-coordinate of the chosen point and the other player receives the y-coordinate of the chosen point.

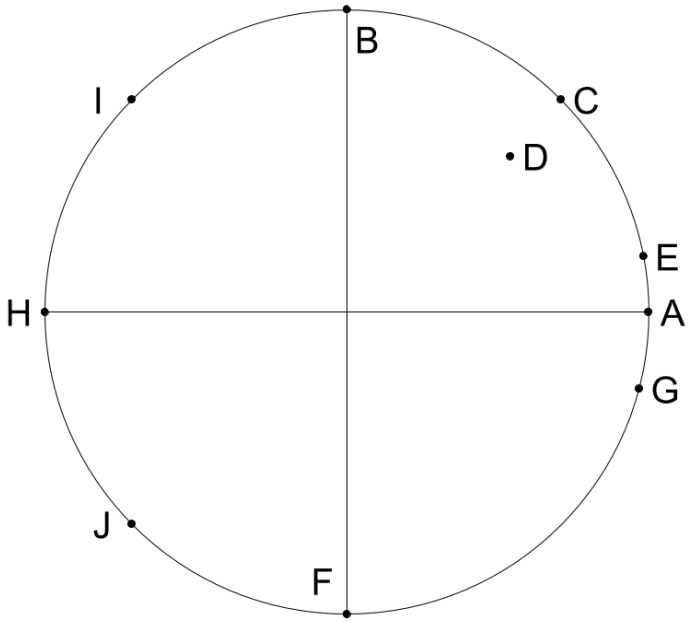

Figure 5.1: The choice set for the unit circle Gift Exchange game. All points in the circle are possible choices for either player (but not necessarily rational). The x-axis is the agent's payoff and the y-axis is the payoff of its opponent.

We choose the unit circle for several reasons. First, its symmetry places both players in an identical position. Second, it presents opportunities for cooperation (so, for example, it is possible for an agent to present to the opponent an option that is beneficial for both agents). Finally, because it is convex there is no need to use randomization to achieve any particular desired expected payoff. If we were to use a diamond shape instead, the optimal strategy would be the same for all opponents because there would be no benefit to cooperation. The best strategy for an agent would be to assign itself the maximum payoff. If we were to use a square shape, the motivation to cooperate

increases, because in this case cooperation imposes no cost on the agent. This still presents an interesting problem, but for now we are focusing on costly cooperation. If we were to use a shape which is not convex, then there will be outcomes that can only be achieved by using strategies that require randomization or multi-round coordination. If we were to use an asymmetric shape, it would be more difficult to evaluate an agent's performance, because the agent's performance will depend heavily on the set of payoffs available to it.

Figure 5.1 shows the set of choices with some noteworthy options highlighted. The *greedy choice* for the agent is A, i.e. $(1, 0)$, and the greedy choice for the opponent is B, i.e. $(0, 1)$. The *social welfare maximizing choice* is C, i.e. $(\sqrt{2}/2, \sqrt{2}/2)$. D, i.e. $(.5, .5)$, is also cooperative, but not pareto-optimal since both agents could make more by playing C instead. E, which is $(.98, .2)$, favors the agent, but is slightly beneficial to the opponent. F, which is $(0, -1)$, is the *maximally punishing choice* for the opponent. G, i.e. $(.98, .2)$, is a slightly punishing choice which also gives the agent a nearly optimal result. H, which is $(-1, 0)$, is the most punishing choice for the agent. I, i.e. $(-\sqrt{2}/2, \sqrt{2}/2)$, would be played by a competitive opponent that seeks to maximize the difference in scores instead of maximizing its own score. Finally J, which is $(-\sqrt{2}/2, -\sqrt{2}/2)$, is an irrational choice that neither player would choose, because it penalizes both. If both players are rational, the agent will only choose points along the arc $B - A - F$ and the opponent points along the arc $H - B - A$. In this paper we will restrict our attention to agents which play strategies along those arcs.

The decision to have agents act sequentially instead of simultaneously is intended to allow agents to reason about opponent intent more easily. Because agents act sequentially and there is no hidden information it is not necessary for an agent to attempt to estimate the opponent's beliefs about the state of the game; the state of the game is public information. However, it would be possible to play the game simultaneously as

well. The effect of an agent's action is independent of the opponent's action, so agents don't need to estimate the opponent's action. An agent playing the Gift Exchange game simultaneously only needs to estimate the opponent's action if the agent's valuation of outcomes depends on the opponent's action. For example, the agent may be willing to play C, but only if the opponent is also playing C this round, otherwise the agent would prefer to play A. Strategies for the sequential game can easily be extended to the simultaneous game by treating the opponent's simultaneous action as if it happened after the agent's action. In this paper we focus on sequential play.

5.1.1 Formal Definitions

Definition of Gift Exchange game. The game is described by a set of two players $P = \{A, B\}$ and a choice set $U \subset \mathbb{R}$. In round i the current player chooses an outcome $c_i = (c_{i,A}, c_{i,B}) \in U$ with player A choosing in odd numbered rounds and player B choosing in even numbered rounds. After each choice $(c_{i,A}, c_{i,B})$ player A receives payoff $c_{i,A}$ and player B receives payoff $c_{i,B}$. Note that these payoffs may be negative, in which case they represent a loss to that player. The game may be played for a fixed or an indeterminate number of rounds. By convention we will assume the agent is player A and the opponent is player B.

We will focus on two choice sets: the unit circle choice set $U^\circ = \{(c_A, c_B) \in \mathbb{R}^2 | c_A^2 + c_B^2 \leq 1\}$ and the discretized perimeter choice set $U_n^* = \{(\sin 2\pi\frac{i}{n}, \cos 2\pi\frac{i}{n}) \forall i \in [1..n]\}$ where n is any number divisible by 4. The unit circle choice set represents a continuous choice set; the advantage of a continuous choice set is that any pareto-optimal outcome can be achieved in pure strategies, which makes it easier to determine the intended outcome of the opponent's action. The discretized perimeter choice set is discrete, which makes it easier to define and learn simple opponent strategies. The disadvantage of a discrete choice set is that it provides a motivation for agents to randomize, which

makes designing strategies more complicated.

When using the unit circle choice set we will focus on strategies where players assign themselves non-negative payoffs and only choose outcomes on the border of the choice set. This is because a rational agent would never select a point inside the circle and get a smaller payoff. For player A this means selecting outcomes from $U_A = \{(c_A, c_B) \in U | (c_A^2 + c_B^2 = 1) \wedge (c_A^2 \geq 0)\}$ and similarly for player B. We will refer to these as *rational* choices.

Definition of history. We define a history as a sequence of choices, $h = \{c_1 \ldots\}$ with $c_i \in U$, where c_i is the choice made in round i with $c_{i,A}$ being the payoff assigned to player A and $c_{i,B}$ being the payoff assigned to player B. We define $h_{:i}$ as the subsequence of choices in h up to and including round i and h_{-1} as the final choice in h for histories with finite length. H_A is the set of histories in which player A chooses next, and H_B is the set of histories in which player B chooses next.

Definition of Strategies. The set of all strategies for player A is $S_A : H_A \to U$, the set of functions mapping H_A to U. Similarly, S_B is the set of all strategies for player B. The combination of a strategy for player A, $s_A \in S_A$, and a strategy for player B, $s_b \in S_B$, will produce a specific history $Outcome(s_A, s_B) \in H$ with the property that $c_i = s_A(h_{:i-1})$ when i is odd and $c_i = s_B(h_{:i-1})$ when i is even. We refer to generic strategies $s \in S_A \cup S_B = S$ when we don't wish to specify which player we are referring to. Strategies which include randomization can be represented as a probability distribution over S_A or S_B. The outcome of two randomized strategies is a distribution over H.

Definition of Strategy Types.

A strategy s is a *constant* strategy if $s(h) = c \in U$ for some constant c. A strategy s is *non-reactive* if the opponent's choices never depend on the agent's choices. Formally, $s(h) = s(h')$ whenever $length(h) = length(h')$. Note that non-reactive does not imply

that the strategy is stationary. A *reactive* strategy is one for which there exist histories h and h' with $length(h) = length(h')$ for which $s(h) \neq s(h')$.

A strategy s is *immediately reactive* if it only responds to the opponent's previous action without regard to the history before it. Formally, $s(h) = s(h')$ whenever h and h' have the same length, $length(h) = length(h')$, and the same last move $h_{-1} = h'_{-1}$. An immediately reactive strategy can be characterized by a *response function* $r : U \to U$, where in response to an opponent move (x, y) the agent plays $r((x, y))$. A *rational immediately reactive strategy* is an immediately reactive strategy which only makes choices on the convex hull of the choice set which cannot increase the agent's payoff without changing the opponents payoff and which treats all opponent moves which give it the same payoff the same. If the choice set is U° a rational immediately reactive strategy can be characterized by a response function $r : [-1, 1] \to [-1, 1]$ which takes as input the amount the opponent gave to the agent and returns the amount the agent will give to the opponent. In response to an opponent move $(c_{i,A}, c_{i,B})$ the agent plays $(c_{i+1,A}, c_{i+1,B})$ where $c_{i+1,A} = \sqrt{1 - r(c_{i,A})^2}$ and $c_{i+1,B} = r(c_{i,A})$, which maximizes its own payoff under the constraint that the outcome is selected from U and opponent receives $r(c_A)$.

Definition of Payoff. The payoff of a history through time t is the sum of payoffs of the moves $\mathit{Payoff}_t(h) = (\sum_{i=1}^{t} c_{i,A}, \sum_{i=1}^{t} c_{i,B})$. The average payoff is $\overline{\mathit{Payoff}}_t(h) = (\sum_{i=1}^{t} c_{i,A}/t, \sum_{i=1}^{t} c_{i,B}/t)$. In games played for an indefinite period, where the stopping point is unknown or there is no stopping point, it is trickier to evaluate performance since the sum of the payoffs can diverge. It is tempting to use the limit of the average payoff, $\overline{\mathit{Payoff}}_\infty(h) = \lim_{t\to\infty} \overline{\mathit{Payoff}}_t(h)$ as a performance measure, but there are histories for which that limit doesn't converge. An example of a strategy which produces a non-converging history is one that alternates between playing $(0, 1)$ and $(1, 0)$, switching

with decreasing frequency as the game goes on.

$$s(h) = \begin{cases} (0,1) & if\ \lfloor \log_{10} length(h) \rfloor \cong 0 \mod 2 \\ (1,0) & if\ \lfloor \log_{10} length(h) \rfloor \cong 1 \mod 2 \end{cases} \tag{5.1}$$

If both players follow this strategy, average payoffs will oscillate between $(.9, .1)$ and $(.1, .9)$ and never converge. Combinations of strategies which produce histories for which the limit doesn't converge do not generally perform well, as they involve cycling between different choices infinitely often, which is not Pareto-optimal, but we still want to be able to evaluate such strategies.

We will generally use the discounted average payoff to evaluate performance, where δ is the *discount factor*:

$$\overline{Payoff}^{\delta} = \lim_{t\to\infty}(\sum_{i=1}^{t}(1-\delta)^i \times c_{i,A} \times \delta, \sum_{i=1}^{t}(1-\delta)^i \times c_{i,B} \times \delta)$$

The discount factor is used to describe an agent which doesn't value future payoffs as highly as immediate payoffs. It is often used to describe a situation in which the game can end after any round with probability δ. Using the discounted average payoff has the advantage that the limit always exists, but doesn't entirely solve the problem of evaluating non-converging strategies. For example, with a discount factor of .01, the oscillating strategy (Equation 5.1) we defined earlier has a discounted average payoff of $(.452, .548)$, but with a discount factor of .001 the discounted average payoff is $(.546, .454)$. In other words, the discounted average payoff can give odd results for irrational strategies.

In situations where using a discount factor is not appropriate we will use the limit of the minimum average payoff: $\lim_{t\to\infty} \min_{x>t} \overline{Payoff}_x(h)_p$ where p is the player. If $\overline{Payoff}_\infty(h)$ exists, then the limit will be equal to $\overline{Payoff}_\infty(h)_p$ for both players. If it doesn't exist, then the limit will be pessimistic; for the oscillating strategy described

earlier the limit of the minimum average payoff will be $(.1, .1)$.

5.2 A Taxonomy of Opponent Models

The optimal strategy of an agent that plays the Gift Exchange game depends on the opponent's strategy (or more accurately, on the agent's beliefs about the opponent's strategy). Our taxonomy of opponent types is outlined in Figure 5.2.

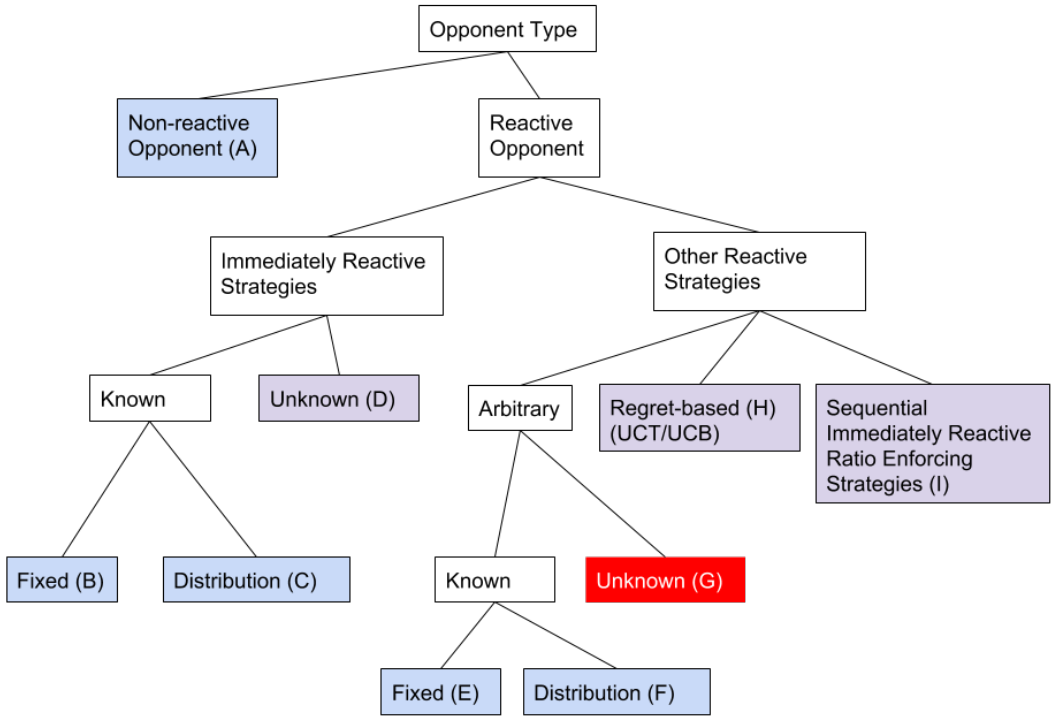

Figure 5.2: A taxonomy of opponent types in the Gift Exchange game. Light blue squares indicate opponent types for which we briefly discuss the optimal solution. Light purple squares indicate opponent types which we discuss in greater depth. The red square indicates an opponent type for which there is no optimal solution.

The space of possible strategies is huge. Since the Nash equilibrium in the Gift

Exchange game is non-reactive, it is worth considering why an agent would adopt a reactive strategy. We will consider two primary motivations to adopt a reactive strategy. Firstly, the agent may believe its opponent's strategy can be learned, and so it reacts to choose the optimal response to the learned strategy. Secondly, the agent may believe its opponent is attempting to learn its strategy. In this case the agent should select the strategy for which the opponent's learned response is best for the agent.

The first distinction we will draw is between non-reactive strategies and reactive strategies. The optimal response to a non-reactive strategy (A) is simple: choose the outcome $(1, 0)$ which maximizes the agent's payoff. If the opponent is non-reactive, the agent's total payoff can be viewed as the sum of the payoff received from the opponent's actions and the payoff received from the agent's actions. The payoff received from the opponent's actions can't be affected, and the payoff received from the agent's actions can by maximized by always assigning a payoff of 1 to the agent.

Against reactive strategies, the problem is more complicated. We can divide reactive opponents into ones which are immediately reactive and more complex opponents. Immediately reactive opponents are simpler to play against, so we will discuss them first. Against a fixed, known, immediately reactive strategy (B) the optimal response is to choose a fixed outcome. This outcome can be found simply by examining the representation of the strategy. Against a distribution over known immediately reactive strategies (C) finding the best response is more complex. If the distribution is over a finite number of strategies, the optimal response will be a decision tree in which each node contains an outcome to try and the child nodes are indexed by possible opponent responses to that outcome. The leaves of the tree represent histories in which the opponent has been narrowed down to a set of immediately reactive strategies which all have the same best response. If the opponent is known to be immediately reactive, but the agent doesn't know the distribution over opponents (D), then it is impossible to find an

optimal response. In this situation it is reasonable to play a no-regret strategy, which is guaranteed to achieve sub-linear regret against an immediately reactive opponent [12]. A no-regret strategy will eventually converge to the best-response of the opponent. We will discuss the implementation of no-regret strategies for the Gift Exchange game in Section 5.3.

Playing against opponents with reactive strategies that are not immediately reactive is more complicated. If the opponent's strategy is known (E) and the opponent can be described as a finite automaton, the problem of finding a best response is equivalent to solving a Markov decision process. If the agent can describe the set of possible opponents as a distribution over finite automata (F), then finding a best response is equivalent to solving a partially observable Markov decision process [45].

If the opponent is reactive, but the agent has no information about the opponent's strategy (G), then it is impossible to find a best response. However there are some properties which an agent can take advantage of. If the opponent is playing a strategy which guarantees sub-linear regret vs. an immediately reactive opponent (H), an agent can take advantage of that by playing an immediately reactive strategy to which the best response gives the agent a good outcome. We will discuss the process of selecting an appropriate immediately reactive strategy in more detail in Section 5.4.

Regret based algorithms can provide good performance against immediately reactive strategies, but in self-play they perform poorly. The final type of strategy we will look at are Sequential Immediately Reactive Ratio Enforcing Strategies (I), which can perform well against immediately reactive strategies, against regret based strategies, and in self play. These strategies play a sequence of immediately reactive strategies which perform well against regret based strategies and adjust over time to allow for good performance in self-play, or against immediately reactive strategies. We will describe these strategies in Section 5.5.

5.3 Responding to Immediately Reactive Strategies

If the opponent is immediately reactive, but the exact strategy is unknown and the distribution from which the strategy is drawn is also unknown, the best approach to take is to minimize the agent's regret. In this context an agent's regret is defined as the difference between the agent's payoff and the maximum possible payoff achievable when playing against the opponent. For an agent playing strategy s_A against an opponent playing an immediately reactive strategy s_B with response function r, the agent's regret through time t is $t \times (r(c^*)_A + c^*_A) - \mathit{Payoff}_t(\mathit{Outcome}(s_A, s_B))$ where $c^* \in U$ is the best response to the opponent's strategy. In this situation it is impossible to achieve an optimal response to the opponent, but if the agent can achieve sub-linear regret then its average payoff will approach the optimal payoff over time. Note that no-regret strategies can find a best response even if the opponent randomly selects an immediately reactive strategy from a unknown fixed distribution at every round.

5.3.1 Discrete Choice Sets (U^*)

One advantage of the discrete version of the Gift Exchange game is that it is easier to define specific classes of opponents and the best responses to them. In this section we will discuss best responses to opponents which only consider the previous choice of the agent when making their choice under a variety of circumstances.

Opponents which are non-reactive are trivial to best-respond to. The best response to a non-reactive opponent is to always select the choice which maximizes the agent's payoff.

Theorem 3. *Given a non-reactive strategy $s \in S_B$, the strategy $s'(h) = \arg\max_{u \in U} u_A$ maximizes the payoff of player A.*

Proof. Let $h = \mathit{Outcome}(s', s)$. The payoff of player A through round t is $\mathit{Payoff}_{t,A}(h) =$

$\sum_{i=1}^{t} h_{i,A}$. This can be split into the rounds where player A chose and the rounds where player B chose $\sum_{i=1}^{t/2} h_{2i-1,A} + \sum_{i=1}^{t/2} h_{2i,A}$. which is $= \sum_{i=1}^{t/2} s'(h_{2i-2})_A + \sum_{i=1}^{t/2} s(h_{2i-1})_A$. The first sum is maximal by the definition of s' and the second sum is constant because s is a non-reactive strategy. □

If the opponent is playing a fixed immediately reactive strategy, we consider these different cases:

Maximize average payoff with no time horizon. If the goal is to maximize the average payoff with no time horizon ($\overline{Payoff}_\infty(h)_A$), the agent should make every choice once to learn the opponent strategy s^B, and then play the constant strategy which gives the best payoff $s(h) = \arg\max_{u \in U} u_A + s^B(u)_A$. The order in which options are checked doesn't matter because the goal is to maximize average payoff over an infinite time horizon.

Maximize total payoff over a fixed horizon. If the opponent is immediately reactive, and the goal is to maximize the total payoff over some fixed time horizon which is much longer than the total number of options available $t >> |U|$, then the optimal strategy is similar, except that options should be checked in descending order of u_A, and no further options should be checked when the payoff of choosing the current best option for the rest of the game is greater than the maximum possible payoff of the remaining unchecked options. Let $s_i : U_i' \to U$ be the incomplete function learned from observing the opponent's behavior up to round i where U_i' is the set of choices for which responses have been observed up to round i. Let u_i^{max} be the best observed option in round i, $u_i^{max} = \arg\max_{u \in U_i'} u_A + s_i(u)_A$

Theorem 4. *The optimal sequence of choices $\{u_1, u_2, ..., u_n\}$ to maximize the total payoff assuming a uniform distribution over immediately reactive opponents is in descending order of preference $u_{i,A} \geq u_{i+1,A}$.*

Proof. Sketch. Consider any two sequences which are identical except for two adjacent choices, $C = \{u_1, ...u, u', ...u_n\}$ and $C' = \{u_1, ...u', u, ...u_n\}$. Assume without loss of generality $u_A > u'_A$. When neither u nor u' is the maximizing choice, the expected payoff of ordering guesses according to either sequence is identical. When one of u and u' is the payoff maximizing choice, the expected payoff of guessing u first is higher because when the agent guesses u first the immediate payoff is higher, and the agent is able to rule out more unchecked options. Therefore, the only sequence for which the expected payoff can't be improved is one in which options are checked in descending order of preference. □

Maximize discounted average payoff. If the opponent is immediately reactive and the goal is to maximize the discounted average payoff, then the agent should guess in descending order of expected value $\mathbb{E}(u_A + s(u)_A)$, but stop guessing when the expected payoff of guessing the next value is lower than the value of choosing the best option found so far for the rest of the game $\frac{u^{max}_{i,A} + s(u^{max}_i)_A}{1-\delta} > u_{i+1,A} + \delta\mathbb{E}(s(u_{i+1,A})) + \frac{\delta^2}{1-\delta}\mathbb{E}(max(u^{max}_{i,A} + s(u^{max}_i)_A), u_{i+1,A} + s(u_{i+1})_A)$ where δ is the discount factor of the agent. Note that the expectation over opponent strategies can be uniform, but is not required to be.

All the strategies described up to now have assumed that the opponent is playing a fixed immediately reactive strategy. However the opponent may be using randomization. A randomizing immediately reactive opponent will respond to an agent choice u by drawing from a random distribution over immediately reactive strategies, which is functionally identical to a multi-armed bandit. Each choice of outcome u represents a separate arm, with a payoff of $u_A + \mathbb{E}(s(u))$ where s is drawn from the opponent's distribution over strategies. The problem of finding the optimal action can be solved using the Upper Confidence Bound (UCB) [4] algorithm.

5.3.2 Choice Set U°

When the choice set is U^* the Upper Confidence Bounds (UCB) [4] can be used to limit regret, but UCB is dependent on a discrete action space. To find the best response to an immediately reactive strategy when the choice set is U° we have adapted the Upper Confidence Bounds for Trees (UCT) [47] algorithm.

UCT was created to solve MDPs with large state spaces, but it can be adapted to an environment with continuous choices by treating the problem of selecting a point in the interval $[-1, 1]$ as a sequential process of narrowing down the interval. Algorithm 3 describes the UCT algorithm. Define a binary tree in which each node represents a sub-interval of $[-1, 1]$. The children of a node divide the nodes sub-interval in two. The problem of finding a maximum value of a function on $[-1, 1]$ can be treated as identifying the set of nodes in the tree which contain the global maximum of the function.

Note that it is not possible to achieve sub-linear regret against all immediately reactive opponents: consider the set of immediately reactive strategies with a response function which returns $(1, 0)$ if the agent's move gives the opponent x, for some arbitrarily chosen $x \in [-1, 1]$, and $(-1, 0)$ otherwise. There is an uncountable set of possible values for x, and an agent can only make a countable number observations, so the probability of stumbling on the correct value for x is 0, and therefore the expected regret is linear. However, for immediately reactive strategies with continuous response functions no-regret learning will work. We can define the maximum slope of a response function r as $max_{c,c' \in U} |\frac{r(c)_A - r(c')_A}{c_B - c'_B}|$. This quantity measures the maximum change in the agent's payoff for a given change in the amount the agent assigns the opponent. We can express the regret limits in terms of this maximum slope.

Consider a function on $[-1, 1]$ with values in the range $[-2, 2]$. We can convert it into a binary tree of depth d where each leaf is an interval of width 2^{1-d}. If the agent plays at a leaf by selecting a value uniformly at random from the leaf, then UCT will select

non-optimal leaves with vanishing probablity [48]. If the depth of the tree is limited this conversion does not imply that the leaf containing the maximum value will be found; it is possible that the expected value of the leaf containing the maximum value is lower than the expected value of some other leaf. For example, consider the function:

$$f(x) = \begin{cases} .9 & if \ -1 \leq x < 0 \\ -5x + .9 & if \ 0 \leq x < .18 \\ 0 & if \ .18 \leq x < .8 \\ 5x - 4 & if \ .8 \leq x \leq 1 \end{cases}$$

If this function is converted into a binary tree of depth 3, the leaves $[-1, -.75)$, $[-.75, -.5)$, $[-.5, -.25)$, and $[-.25, 0)$ will all have an expected value of .9, while the leaf $[.75, 1)$, which contains the maximum value, will have an expected value of .4. Consider a function with maximum slope L, where $f(x^*)$ is the maximum value of the function and $f(x')$ is the largest local maximum less than $f(x^*)$. The minimum possible expected value of a bucket of width 2^{1-d} containing $f(x^*)$ is $f(x^*) - L \times 2^{-d}$. The maximum possible expected value of a bucket containing $f(x')$ is $f(x')$. Therefore the bucket containing the maximum value is only guaranteed to be the optimal bucket if $f(x^*) - L \times 2^{-d} > f(x')$. This will be true when $d > \log_2 \frac{L}{f(x^*) - f(x')}$. In the case of our example function, this will be true when the depth of the tree is 6 or more. Since the UCT algorithm will increase the depth of the tree indefinitely,

The other algorithm we have modified to play the Gift Exchange game is based on UCB. UCB has been modified to allow for a continuous action space in [5] and [46]. This approach selects a fixed discretization which divides the action space into intervals, and uses the standard UCB algorithm to select an interval each round. Play over an indefinite number of rounds is handled by playing a sequence of epochs, where successive

epochs use finer discretizations. No information is saved between epochs. Our modified version, Split Bucket UCB, handles an indefinite number of rounds without discarding any data. Split Bucket UCB is an adaptation of the flattened UCT algorithm described in [17], modified to handle the continuous action space.

Split Bucket UCB (Algorithm 4) works similarly to the UCT algorithm. Instead of selecting the next node by a sequence of binary choices, we ignore the tree structure and select the interval with with the highest upper confidence bound on the average expected payoff. When the number of observations for an interval exceeds the splitting threshold, the interval is split in two. There are two options for populating data for the new interval. Either we can explicitly track all observations for an interval, and when an interval is split, assign each observation to the appropriate sub-interval (less computationally efficient, but a very rapid learning approach), or we can track aggregate data about the observations and assign that data to both sub-intervals (more computationally efficient, but slower learning).

5.3.3 Experimental Evaluation

We will evaluate the performance of these learning strategies in two ways; we will examine their performance against ratio-enforcing immediately reactive strategies in the Gift Exchange game, and we will look at their total regret against a class of functions to show how regret depends on the maximum slope of the target function.

The first class of target functions is shown in Figure 5.3. These functions are rational immediately reactive strategies which enforce a chosen ratio in the payoffs received by the agent and the opponent. These target functions generally perform well when playing against a learning opponent. Figure 5.4 shows the regret of UCT, Split Bucket UCB, and a variant of Split Bucket UCB which doesn't track individual observations. Regret was measured for each algorithm against each threshold function over a period of 6400

rounds. Each run was repeated 10 times, and the 95% confidence interval was calculated. First, note that tracking individual observations makes a huge difference in the total regret; Split Bucket UCB has a lower regret by about an order of magnitude. The ratio enforced by the opponent has two effects on the regret. Firstly, the higher the threshold (and hence the ratio) is, the lower the possible per-round regret is, because the optimal response is relatively poor for the agent. Secondly, the target function generated by higher ratios is flatter, and thus more difficult to learn. In the experiment the UCT learner achieved the lowest regret (255) against an opponent that enforced a ratio of 1 between the agent's payoff and its payoff. The Split Bucket UCB learner achieved its lowest regret (303) against the opponent which enforced a ratio of 19.975, but the opponent which enforced a ratio of 3.045 is within the confidence interval. Split Bucket UCB with observation tracking achieved the lowest regret (12.2) against the opponent which enforced a ratio of 3.045, but opponents which enforce a ratio of 19.975 are within the confidence interval. In general, regret is higher against opponents which demand a higher percentage of payoff, which suggests that the difficulty of learning to best respond to a greedier opponent outweighs the lower maximum regret an agent can have when playing against such opponents.

The second class of target functions were deliberately constructed to be difficult to learn, given a specific maximum slope. The form of the function is

$$f(x) = \begin{cases} 1-\epsilon & if \ -1 \leq x < 0 \\ 1-\epsilon-L\times x & if \ 0 \leq x < \frac{1-\epsilon}{L} \\ 0 & if \ \frac{1-\epsilon}{L} \leq x < 1-\frac{1}{L} \\ 1-L(1-x) & if \ 1-\frac{1}{L} \leq x \leq 1 \end{cases}$$

where L is the maximum slope of the function and ϵ is the difference between the global

maximum and the highest local maxima less than the global maximum. The optimal value of x is 1, but learning algorithms may spend a significant amount of time exploring $x < 0$ before they find the true maximum.

Figure 5.5 shows how the performance of Split Bucket UCB with observation tracking is affected by the exploration factor and the maximum slope of the target function when played for 10000 iterations. The optimal exploration factor is dependent on the maximum slope of the target function. The left side of the graphs shows the regret caused by settling on a local maximum instead of the global maximum, which happens when the exploration factor is too low. The right side of the graph shows the regret caused by spending too many actions exploring, which happens when the exploration factor is too high. The maximum slope affects the difficulty of learning the target function. If the difference between the global and local maximum is high enough, there is an exploration factor that achieves low regret by identifying and playing the optimal value. If the difference is too low 10000 rounds may be insufficient to discover and exploit the optimal value.

Algorithm 3 Upper Confidence Trees

$root \leftarrow$ binary tree node ▷ Initialize tree
$root.min \leftarrow -1$ ▷ Initial range is full interval
$root.max \leftarrow 1$
$root.N \leftarrow 0$ ▷ Number of observations
$root.total \leftarrow 0$ ▷ Total payoff from playing in nodes range
$root.leftchild, root.rightchild, root.parent \leftarrow None$ ▷ Initial tree is single node
$lastmove \leftarrow None$
while $c_A, c_B \leftarrow getOpponentMove()$ **do**
    if lastmove is not None **then** ▷ Update the tree to reflect the new observation
        $updatenode \leftarrow lastnode$
        while updatenode is not None **do**
            $updatenode.total \leftarrow updatenode.total + c_A + \sqrt{1 - lastmove^2}$
            $updatenode.N \leftarrow updatenode.N + 1$
            $updatenode \leftarrow updatenode.Parent$
        end while
    end if
    $node \leftarrow root$ ▷ Get next action
    while *node* has no unexplored children **do**
        $node \leftarrow child$ which maximizes $child.total + \sqrt{\frac{2 \ln node.N}{child.N}}$
    end while
    Select an unexplored child of *node*
    Create a new node, *child*
    $child.parent \leftarrow node$
    $child.leftchild, child.rightchild \leftarrow None$
    $child.N \leftarrow 0$
    $child.total \leftarrow 0$
    if the left child is selected **then**
        $child.min \leftarrow node.min$
        $child.max \leftarrow \frac{node.min+node.max}{2}$
    else
        $child.min \leftarrow \frac{node.min+node.max}{2}$
        $child.max \leftarrow node.max$
    end if
    $lastnode \leftarrow child$
    $lastmove \leftarrow$ uniform distribution from lastnode.min to lastnode.max
    $playMove(\sqrt{1 - lastmove^2}, lastmove)$
end while

Algorithm 4 Split Bucket UCB

$e \leftarrow 1$ ▷ Exploration factor
$buckets \leftarrow$ list of buckets ▷ Buckets store observations
▷ Each bucket covers an interval
$lastbucket \leftarrow None$
while $c_A, c_B \leftarrow getOpponentMove()$ **do**
    if $lastbucket$ is not None **then** ▷ Update buckets with last observation
        $UpdateModel(lastbucket, lastmove, c_A)$
    end if
    $lastbucket \leftarrow ChooseBucket()$
    $lastvalue \leftarrow$ uniform distribution from $lastbucket.min$ to $lastbucket.max$
    $PlayMove(\sqrt{1 - lastvalue^2}, lastvalue)$ ▷ Only make pareto-optimal choices
end while

function ChooseMove
    if any bucket is empty **then**
        $result \leftarrow$ randomly chosen empty bucket
    else
        $result \leftarrow$ bucket which maximizes
$\frac{bucket.total}{bucket.N} + e \times \sqrt{\frac{2 \ln N}{bucket.N}}(bucket.max - bucket.min)$
    end if
    return result
end function

function UpdateModel(lastbucket,lastmove,c_A)
    $lastbucket.N \leftarrow lastbucket.N + 1$
    $lastbucket.total \leftarrow lastbucket.total + c_A + \sqrt{1 - lastmove^2}$
    if $lastbucket.N > 4$ **then** ▷ Split the bucket
        Split $lastbucket$ in two
        Split the observations in $lastbucket$ into the new buckets
    end if
end function

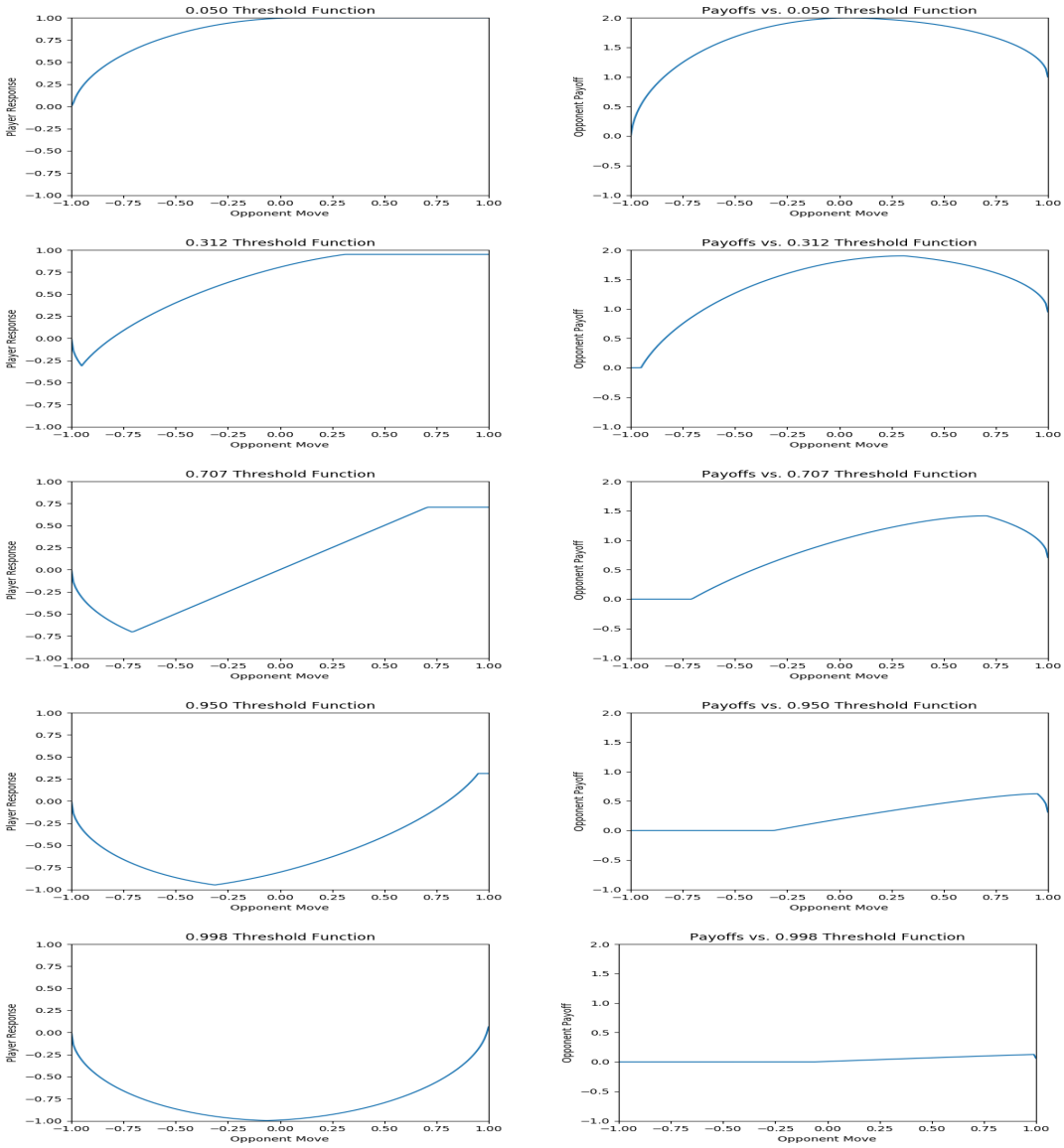

Figure 5.3: Target strategies for the adapted UCT and split bucket UCB algorithms. The response functions are shown on the left, and the payoffs received by the learning agent for each action are shown on the right.

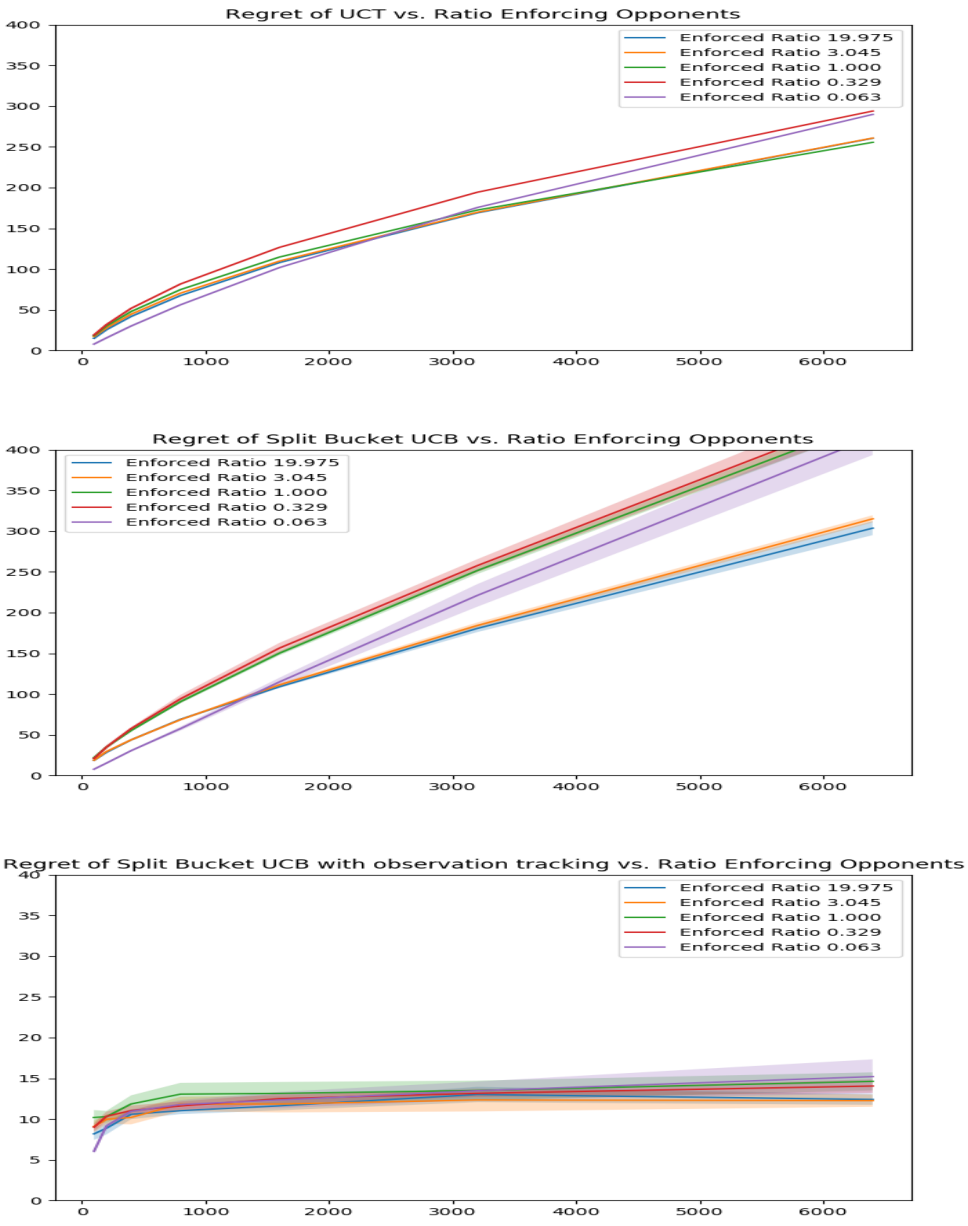

Figure 5.4: Total regret achieved by UCT (top), Split Bucket UCB (middle), and Split Bucket UCB with observation tracking (bottom) playing against the target strategies. The shaded area shows the 95% confidence interval.

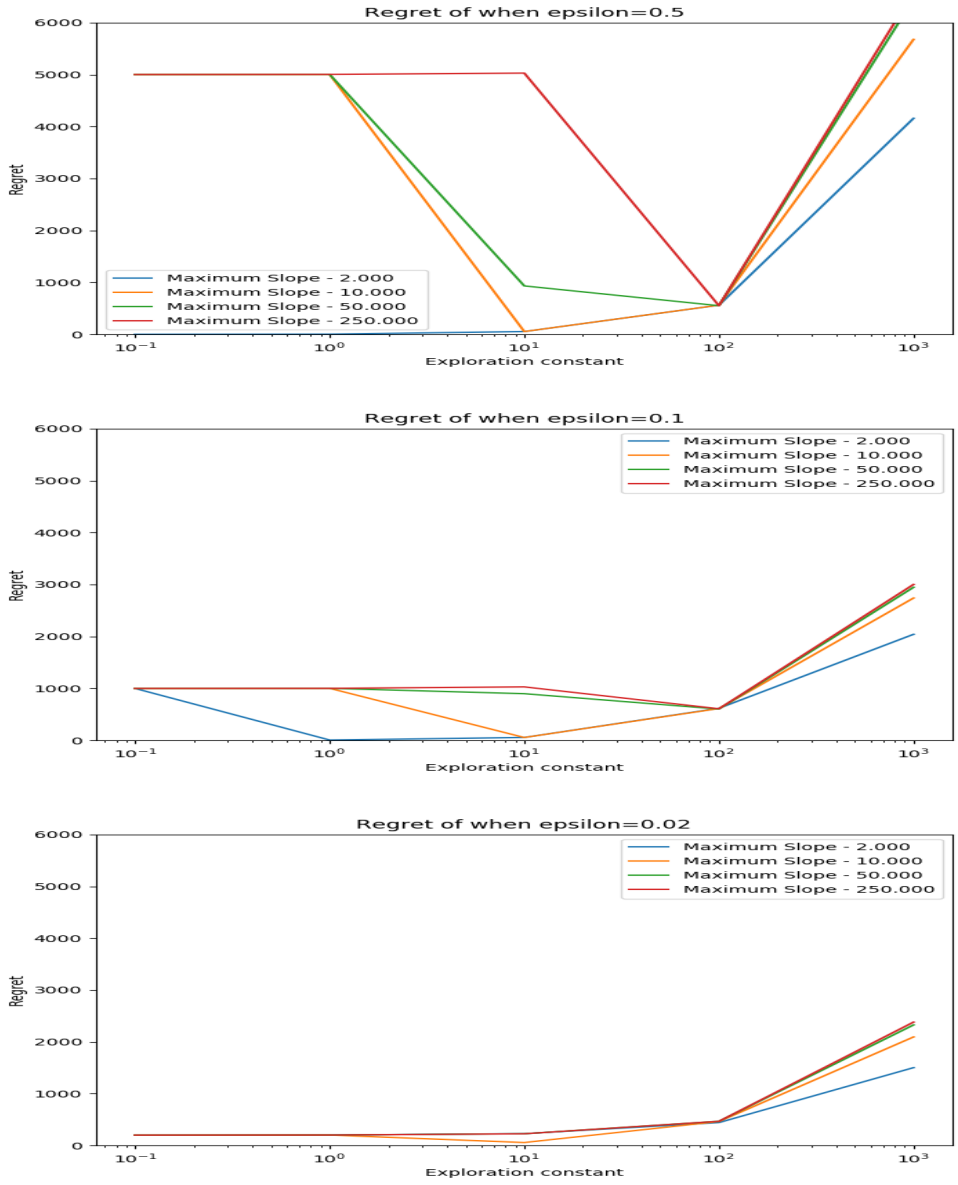

Figure 5.5: Total regret achieved by Split Bucket UCB with observation tracking playing against the target strategies with varying maximum slopes. The difference between the global maximum and the next highest local maximum is .5 (top), .1 (middle), and .02 (bottom). The x-axis is logarithmically scaled.

5.4 Choosing Immediately Reactive Strategies

If the opponent is using a no-regret strategy, it will likely be beyond the computational capabilities of the agent to compute an exact best-response. However, it is possible to take advantage of the guarantees provided by a no-regret strategy to create a strategy which provides the agent with an average payoff arbitrarily close to the average payoff achieved by the best response to the opponent's strategy. This can be done by adopting a reciprocating strategy, similar to Tit-for-Tat in Prisoner's Dilemma. An immediately reactive agent can introduce a positive correlation between its payoff and the opponent's payoff by conditioning its choice on the opponent's previous choice. This can be used to cause a no-regret opponent to select a choice which gives the agent a good payoff.

Algorithm 5 Simulated Annealing Algorithm

```
1: Generate a population of N candidates
2: σ ← σ^initial                                   ▷ Level of noise to add
3: while σ > σ^final do
4:     for each candidate do
5:         Generate m perturbed candidates
6:     end for
7:     for each perturbed candidate do
8:         Set the score of the candidate to Payoff^δ(Outcome(candidate, opponent))
9:     end for
10:    Set the population of candidates to the N highest scoring perturbed candidates
11:    σ ← σ × Δ
12: end while
13: return current population
```

5.4.1 Immediately Reactive Strategies for the Choice Set U^*

There are a number of ways to take advantage of the strategies described in the section 5.3.1 which find a best-response to an immediately reactive opponent. One method is to violate the assumptions of the learning strategies which assume a fixed immediately reactive opponent. An agent can take advantage of those strategies by playing its

initial responses to deceive the opponent and then, once the opponent has settled on what it thinks is the best response, switching to playing the choice which maximizes its own payoff. A more interesting way to take advantage of learning opponents is to play a strategy which follows the assumption of the learning strategy, but is structured such that the best response to that strategy is beneficial to the agent. Essentially, such a strategy is using the constraints on the agent's behavior to gain an advantage over the opponent in a manner similar to a Stackelberg [73] leader.

If the opponent assumes the agent is playing an immediately reactive strategy, we can consider these different cases:

Maximize average payoff vs. fixed strategy: If the opponent is attempting to maximize its average payoff, then the best immediately reactive strategy for the agent is the one which maximizes the agent's payoff subject to the constraint that the opponent's payoff is greater than the maximum amount it can guarantee itself. Let $u_B^{max} \in U$ be the choice which maximizes the opponent's payoff, and $u_B^{min} \in U$ be the choice which minimizes the opponent's payoff. Let $u, u' \in U$ be the choices which maximize $u_A + u'_A$ subject to the constraint that $u_B + u'_B > u_B^{max} + u_B^{min}$. Depending on the choices in U it is possible that $u = u'$; for example, if $u_B^{max} + u_B^{min} = 0$ and $U = \{(1,0), (-1,0), (0,1), (0,-1), (.7,.7), (.7,-.7), (-.7,.7)\}$ then u and u' will be $(1,0)$ and $(.7,.7)$, but if we change $(1,0)$ to $(1,.1)$ then $u = u' = (1,.1)$. u is the choice that the agent would prefer; u' is the offer that the agent needs to make to ensure that the opponent is better off accepting the deal. Define the agent's optimal strategy s as $s(u) = u'$ and $s(u') = u$; for any other choice $v \in U$ let $s(v) = \arg\max_{v' \in U} v'_A$ subject to the constraint that $v_B + v'_B < u_B + u'_B$. s is a best response to any opponent learning strategy which assumes the agent is playing a fixed immediately reactive strategy.

As an example of this kind of strategy, consider the optimal fixed immediately

reactive strategy for an agent playing a Gift Exchange game with discretized perimeter choice set U_n^* against a learning opponent. The optimal strategy for the agent is:

$$s_n(u) = \begin{cases} (\cos\frac{\pi}{2n}, \sin\frac{\pi}{2n}) & if\ u = (1,0) \\ (1,0) & if\ u_B < 0\ or\ u = (\cos\frac{\pi}{2n}, \sin\frac{\pi}{2n}) \\ (u_B, u_A) & if\ u_A < 0 \\ (u_A, -u_B) & otherwise \end{cases}$$

This strategy limits the opponent to receiving a payoff of 0 unless it chooses a preferred outcome for the agent in which case it will receive an average payoff of $\frac{1}{2}\sin\frac{\pi}{2n}$.

Maximize discounted average payoff vs. fixed strategy: the agent's best strategy is constructed in a similar manner, except that the preferred choice must be one of the choices the opponent will check.

Maximize average payoff vs. randomizing strategy: If the opponent assumes that the agent is playing a randomizing immediately reactive strategy and is attempting to maximize its average payoff, this allows the agent to improve its performance by using a randomizing strategy. This is because the ability to randomize allows the agent to make choices with an expected payoff on the convex hull of the set of choices, which allows it to give the opponent a choice even more beneficial to it. Given a choice set U we can construct an randomizing immediately reactive strategy to take advantage of a learning opponent as follows. Let $u_B^{max} + u_B^{min}$ be the amount the opponent can guarantee itself by always playing u_B^{max}. Let u^* be the choice which maximizes u_A subject to the constraint that $u_B > \frac{u_B^{max}+u_B^{min}}{2}$. Let

$u^{*\prime}$ be the choice adjacent to u^* on the convex hull which maximizes u_A; if both neighbors of u^* have a lower payoff for the agent, then $u^{*\prime} = u^*$. Pick a probability p such that $p \times u_B^* + (1-p) \times u_B^{*\prime} > u_B^{max} + u_B^{min}$ Then play the following strategy: if the opponent plays u^* play u^* with probability p otherwise play $u^{*\prime}$, otherwise play $\arg\max_{u' \in U} u_A$ subject to the constraint $u_B + u'_B < u_B^*(1+p) + u_B^{*'}(1-p)$. This strategy will ensure that the opponent's best response is to play u^* while maximizing the agent's payoff.

When the opponent is maximizing the average payoff against a randomizing strategy, the optimal strategy for the agent gives the opponent a payoff very slightly greater than the amount the opponent can guarantee for itself. The closer the opponent's payoff is to the amount it can guarantee for itself, the better the agent's payoff is. However, the closer the opponent's payoff is to the amount it can guarantee itself, the longer it will take the opponent to learn the optimal response. In infinitely repeated games this is irrelevant because the agent's average payoff will be $\frac{u_A^*(1+p)+u_A^{*\prime}(1-p)}{2}$, but in finitely repeated games or games with discounting it is significant.

We use simulated annealing to find good strategies for finitely repeated games or games with discounting. Algorithm 5 provides a brief overview of how simulated annealing finds randomizing immediately reactive strategies. Strategies are limited to choosing rational outcomes in U_n^*; they will only assign themselves a non-negative value. A strategy is represented by an array which describes the response to each possible opponent's action. The response of a strategy to a value is given as an index of the value to return plus a probability of giving the opponent the next higher amount. This can represent any randomizing immediately reactive strategy which only randomizes between adjacent options. This limitation is reasonable because randomizing between non-adjacent options is inefficient – it will select points from the interior of the convex hull of possible payoffs.

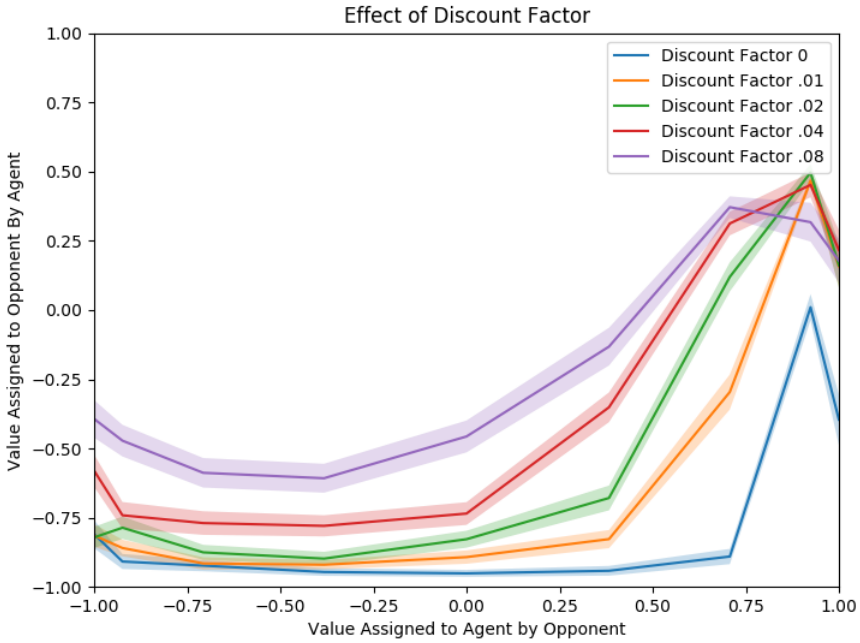

Figure 5.6: Strategies found using simulated annealing in U_{16}^* against a UCB opponent. The discount factor of the agent varies from 0.0 to 0.08. The shaded region shows the 95% confidence interval.

We have used simulated annealing to find effective randomizing immediately reactive strategies for the choice set U^*. We look at the effects of discount factor, number of choices in the choice set, and the exploration factor of the opponent.

Figure 5.6 shows how the best-performing strategies vary as they are optimized for different discount factors. When the discount factor is 0 the strategy is extremely punitive and the opponent's best response is to give the agent .923; the agent will occasionally reciprocate with a gift of 0.382 to the opponent. When the discount factor increases, the agent will accept lower amounts from the opponent, and punish non-compliant choices less severely. This occurs because punishment is costly and with a high discount factor the agent is unwilling to incur those upfront costs to coerce the

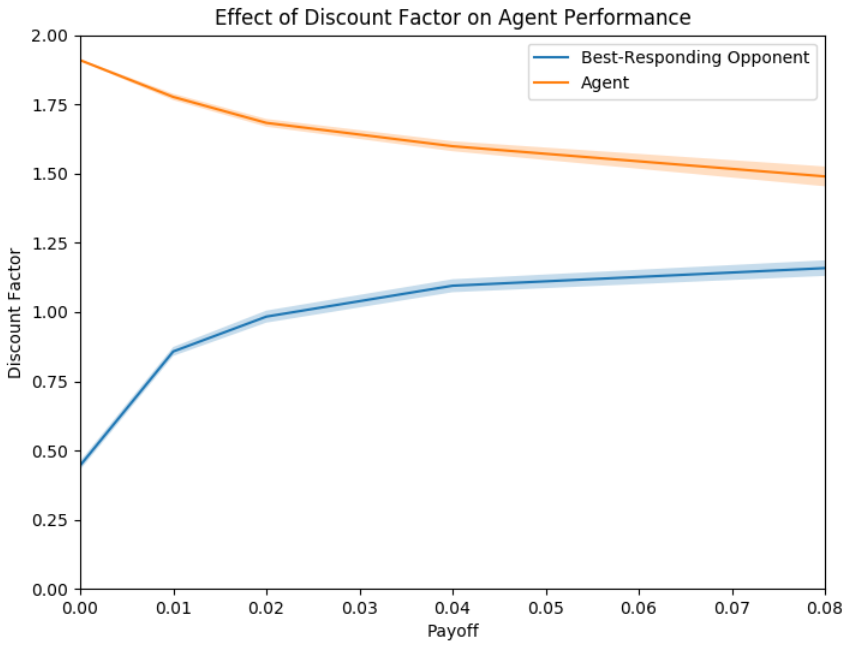

Figure 5.7: Performance of strategies found using simulated annealing in U_{16}^{*} against a UCB opponent as the discount factor varies. The shaded region shows the 95% confidence interval.

opponent into a better long-term strategy. Figure 5.7 shows how the performance of the agent and a best-responding opponent is affected by the discount factor. Note that as the discount factor increases, the agent is offering the opponent a nearly equal split of the potential payoff.

Figure 5.8 shows how the best-performing strategies vary as the number of choices varies. Choices are evenly distributed around the unit circle with 8, 16, or 32 choices. In this case, increasing the number of choices allows the agent to be more precise in its demands. All of the strategies punish at approximately the same level. Strategies with access to more choices are able to be more precise in the amount they demand from the opponent, but they all demand about the same amount.

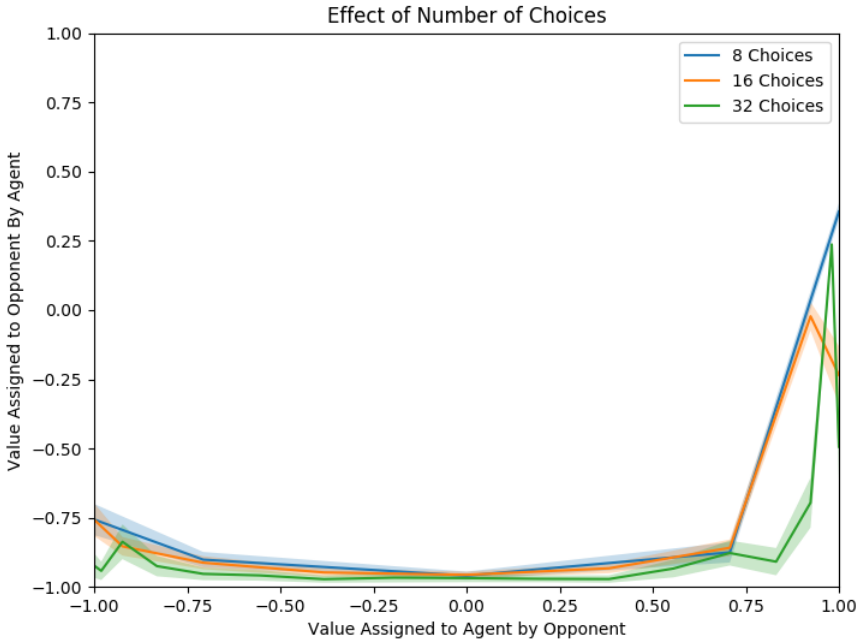

Figure 5.8: Strategies found using simulated annealing in U_8^*, U_{16}^*, and U_{32}^* against a UCB opponent. The shaded region shows the 95% confidence interval.

Figure 5.9: Performance of strategies found using simulated annealing in in U_8^*, U_{16}^*, and U_{32}^* against a UCB opponent. The shaded region shows the 95% confidence interval.

Figure 5.9 shows how the performance of the agent is affected by the number of moves available. Since the discount factor is 0, the agent adopts a very greedy strategy. Increasing the number of moves causes the agent's payoff to go up slightly, but the effect is not as large as that of the discount factor. Unlike the other parameters we have looked at, both the agent's and the opponent's payoff increase as the number of moves rises; this implies that the performance increase is due to more efficient cooperation.

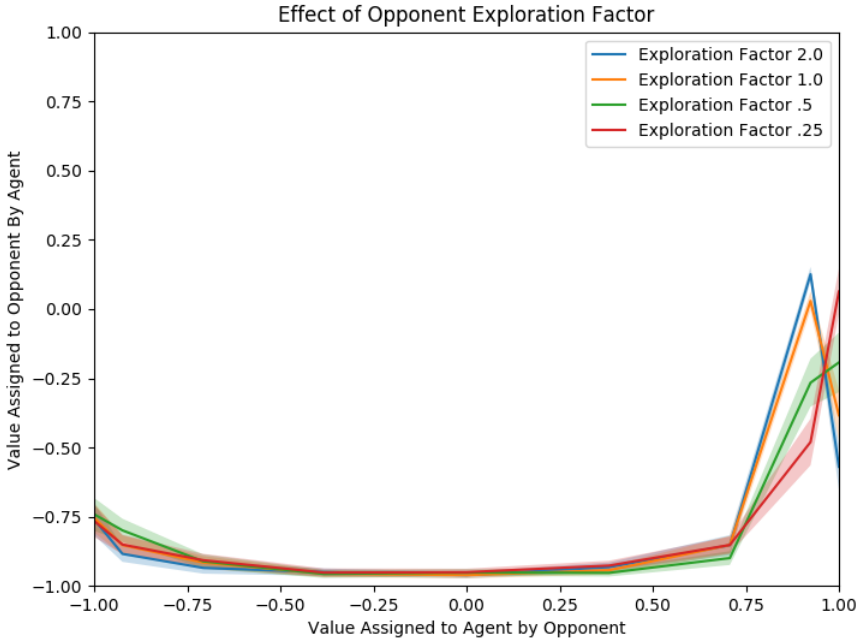

Figure 5.10: Strategies found using simulated annealing in U_{16}^{*} against a UCB opponent with exploration factors 2.0, 1.0, 0.5, and 0.25. The shaded region shows the 95% confidence interval.

Figure 5.10 shows how the best-performing strategies vary as the opponent explores less. The level of exploration has the smallest effect of all the parameters we have looked at. With higher levels of exploration, the agent will select costly punitive choices more frequently, so the optimal strategy is slightly more moderate in its demands of

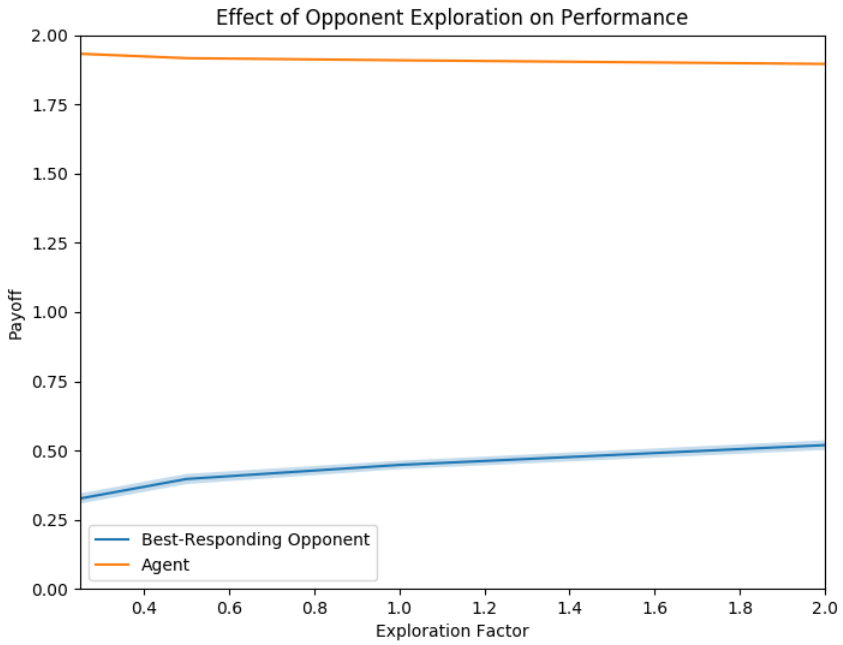

Figure 5.11: Performance of strategies found using simulated annealing in U_{16}^{*} against a UCB opponent as the exploration factor varies. The shaded region shows the 95% confidence interval.

the opponent. However, the effect is far smaller than the effect of varying the discount factor of the agent. Figure 5.11 shows that as the exploration of the opponent increases the performance of the agent decreases while the performance of the opponent increases.

5.4.2 Immediately Reactive Strategies for the Choice Set U°

Consider the arbitrarily chosen immediately reactive strategy for the choice set U° represented by the response function shown in Figure 5.12. The agent's strategy is to play $(.9, \sqrt{.19})$ whenever the opponent plays any move (x, y) with $x \geq .9$, and otherwise to give the opponent $-.9(.9\sqrt{1-x^2} - x\sqrt{.19}) + \sqrt{.19 - .19(.9\sqrt{1-x^2} - x\sqrt{.19})^2}$. This guarantees that over any sequence of (opponent move, agent move) the ratio between

the agent's average payoff and the opponent's average payoff will be at least $.9/\sqrt{.19}$. This may result in a payoff of 0 over both moves if the opponent plays a move with $x < -\sqrt{.19}$, but in no case will the ratio between the agent's payoff and the opponent's payoff drop below $.9/\sqrt{.19}$.

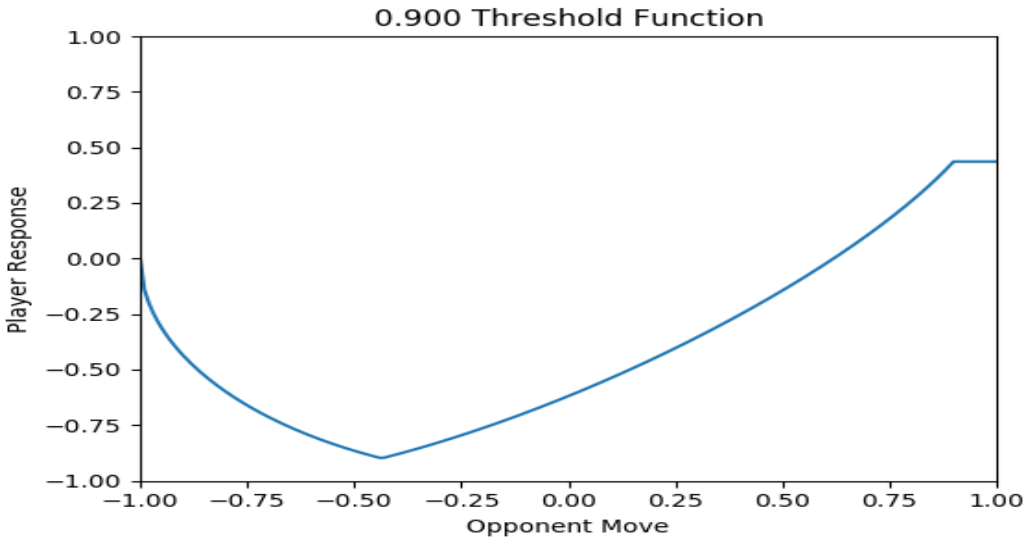

Figure 5.12: Sample response function for a rational immediately reactive strategy.

Consider the performance of a no-regret opponent playing against such a strategy. The best possible average score the opponent can receive is $\sqrt{.19}$. If the opponent receives a total score of S, we know the agent will receive a total score of at least $\frac{.9}{\sqrt{.19}}S$ (the agent can exceed that ratio when the opponent plays a move which gives the agent a payoff greater than .9). The opponent's regret will be $T\sqrt{.19} - S$, where T is the number of rounds played. Let's define the agent's *pseudo-regret* as $.9 \times T - A$ where A is the total score received by the agent. Note that $A \geq \frac{.9}{\sqrt{.19}}S$ so the agent's pseudo-regret will be less than or equal to $.9 \times N - \frac{.9}{\sqrt{.19}}S$ which is $\frac{.9}{\sqrt{.19}}$ of the opponent's regret. If the opponent's regret is bounded by $c\sqrt{T}$ for some constant c the agent's pseudo-regret will be bounded by $\frac{.9c}{\sqrt{.19}}\sqrt{T}$. Therefore, if the opponent's average score S/T converges

to the optimal score $\sqrt{.19}$ then the agent's average score A/T will converge to .9.

This can be generalized to use an arbitrary threshold instead of .9. For any threshold $h \in (0, 1)$ if the agent plays $(h, \sqrt{1-h^2})$ whenever the opponent plays (x, y) with $x \geq h$ and $-hv + \sqrt{h^2v^2 - h^2 - v^2 + 1}$ where $v = h\sqrt{1-x^2} - x\sqrt{1-h^2}$ otherwise, then the agent's average score will converge to h whenever the opponent's average score converges to its optimal payoff of $\sqrt{1-h^2}$. This means that the optimal strategy to play against an opponent playing a no-regret strategy is to adopt a threshold of $1 - \epsilon$ for some arbitrarily small ϵ.

However, adopting a higher threshold does have a disadvantage; the threshold affects the speed of convergence of the opponent. If the agent adopts a threshold of $1 - \epsilon$ the opponents maximum per-round regret will be $\sqrt{2\epsilon - \epsilon^2}$. This affects the speed at which the opponent can learn the optimal response.

If the agent does not discount the future, the speed of convergence is irrelevant, but if the agent discounts future payoffs they will prefer a lower threshold which allows the opponent to learn more rapidly. A constant discount factor models a situation in where the game has a fixed probability of ending after every move.

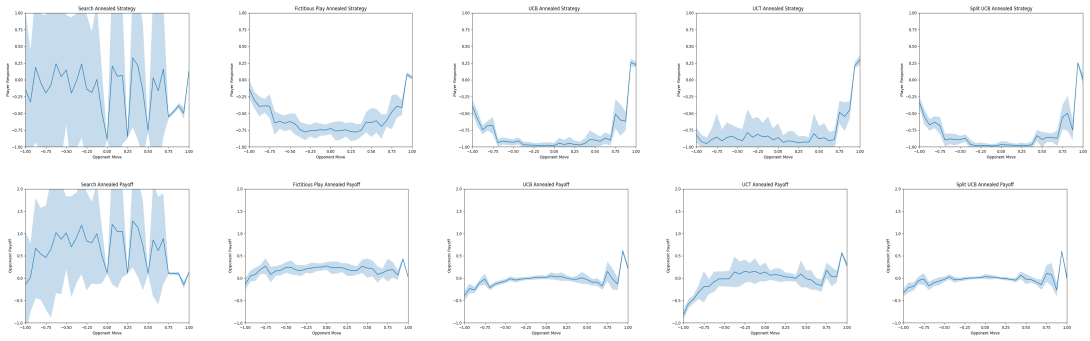

Figure 5.13: The response functions (top) and available payoffs (bottom) of the strategies found by simulated annealing against a opponent using the simple search, greedy, discretized UCB, modified UCT, and the split bucket version of UCB, from left to right. The shaded area shows the 95% confidence interval.

Algorithm 6 Search algorithm which assumes a single-peaked opponent

$gifts \leftarrow (-1, -.5, 0, .5, 1)$ ▷ Set of values to try
$responses \leftarrow (None, None, None, None, None)$ ▷ Payoffs received for those values
loop
  $i \leftarrow$ Random index where $responses_i = None$
  Play $(\sqrt{1 - gifts_i^2}, gifts_i)$ ▷ Make pareto-optimal choices
  $c_A, c_B \leftarrow getOpponentMove()$
  $responses_i \leftarrow c_A + \sqrt{1 - gifts_i^2}$
  if $None \notin responses$ **then** ▷ Identify the interval containing the max value
    $peak \leftarrow \arg\max_i responses_i$
    $newmin = max(1, peak - 1)$
    $newmax = min(5, peak + 1)$
    $gifts \leftarrow 5$ values from $gifts_{newmin}$ to $gifts_{newmax}$
    $newresponses \leftarrow (None, None, None, None, None)$
  end if
end loop

If we confine ourselves to immediately reactive strategies we can use simulated annealing to find the best immediately reactive strategy for a given situation. The optimal strategy is dependent on the discount factor of the agent, and the opponent. Algorithm 5 gives a brief overview of the simulated annealing algorithm we use. We generate a set of N initial candidates, where each candidate is a randomly generated immediately reactive strategy. Each round we generate m perturbed candidates for every initial candidate by adding noise with magnitude σ. We keep the top N candidates and reduce the level of noise by a factor of Δ. From our experiments we have found that the settings $N = 10$, $m = 8$, $\sigma^{inital} = .5$, $\sigma^{final} = .01$, and $\Delta = .99$ produce consistent and effective results.

To perform simulated annealing on immediately reactive strategies in U° we represent response functions as a sequence of evenly spaced values which indicate the agent's response to that move by the opponent. Responses to intermediate moves are found by linear interpolation between the two nearest values. After exploring a range of possible sequence lengths, we settled on 33 values because it performed the best. We permute

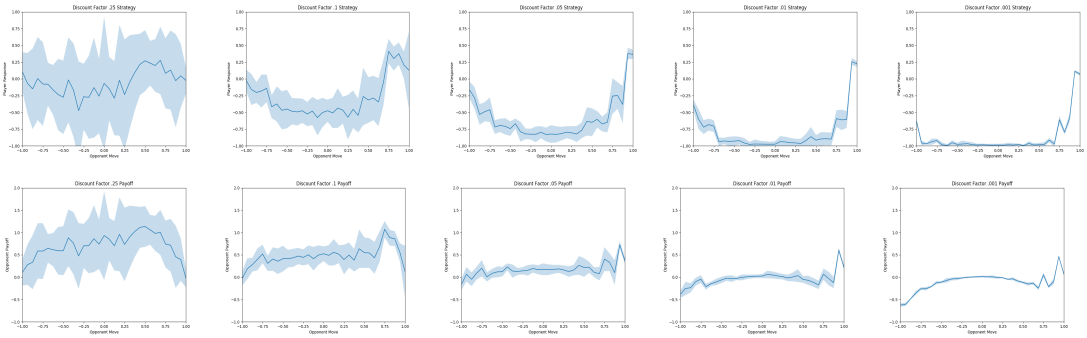

Figure 5.14: The response functions (top) and available payoffs (bottom) of the strategies found by simulated annealing against a opponent using UCB, when the agent has discount factors of .25, .1, .05, .01, and .001 from left to right. The shaded area shows the 95% confidence interval.

them each by adding normally distributed noise with 0 mean and σ standard deviation. We evaluate each copy by observing its performance against the target opponent 10 times.

Figure 5.13 shows the functions found by simulated annealing for five different learning opponents: a simple search algorithm (Algorithm 6), a greedy algorithm (UCB without exploration), discretized UCB without splitting, the modified version of UCT (Algorithm 3), and the split bucket version of UCB (Algorithm 4). Simulated annealing doesn't produce a coherent strategy against an opponent using the search algorithm because the randomly generated response functions do not conform to the search algorithms assumption that the opponent is playing a single-peaked response function. However, note the three low-payoff, low-variance points in the graph – those are the only points checked by the search algorithm before it focuses on the moves which give a higher payoff to the agent. Once it reaches that region the optimal response function for the search strategy converges to an extremely greedy strategy. The greedy algorithm resulted in the next greediest strategy; that may seem odd, but because it updates its observations as it plays it actually ends up exploring more than the search algorithm.

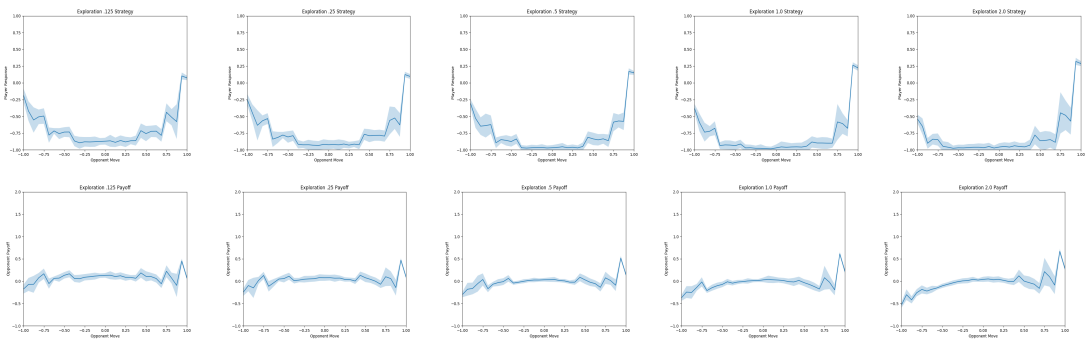

Figure 5.15: The response functions (top) and available payoffs (bottom) of the strategies found by simulated annealing against a opponent using UCB, when the opponent uses exploration constants of .125, .25, .5, 1.0, and 2.0 from left to right. The shaded area shows the 95% confidence interval.

The three variants on UCB that we tried resulted in largely similar strategies. Note that contrary to our expectations, the response functions found are not single-peaked.

Figure 5.14 shows the response functions against an opponent using UCB found by simulated annealing as the discount factor changes. With a higher discount factor there is more noise in the annealed strategies, because performance is evaluated across fewer interactions. As the discount factor drops, the response functions become greedier; with a discount factor of .25 the maximum achievable payoff for the opponent is around 1—when the discount factor is .001 the maximum achievable payoff is around .5.

Figure 5.15 shows the response functions against an opponent using UCB found by simulated annealing as the exploration value of the opponent changes. As the opponent explores more, the amount of punishment used by the response function increases (the agent's response to a 0 gift from the opponent gets closer to -1). As the opponent explores less, the response function gets greedier, although this effect is not as apparent as the effect of the discount factor.

Figure 5.16 shows the performance of the annealed strategies under different discount factors. The best strategy for each discount factor is the strategy optimized for that

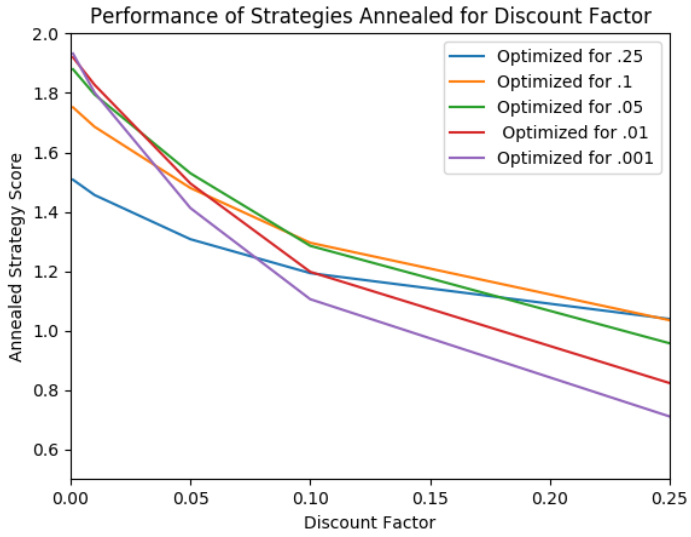

Figure 5.16: Discounted average payoff of strategies optimized for different discount factors when used by agents with other discount factors.

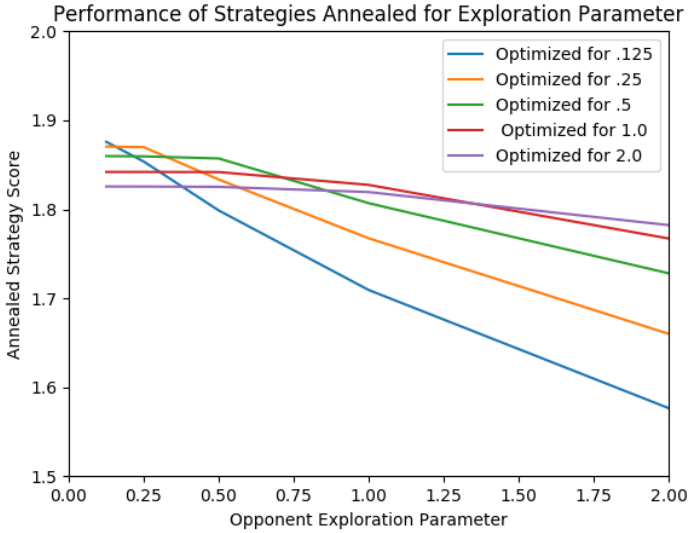

Figure 5.17: Discounted average payoff of strategies optimized for opponents with different exploration factors against opponents with different exploration factors.

	Search	Greedy	UCB	UCT	Split UCB
Search	1.9477	0.7376	0.7765	0.8031	0.8939
Greedy	0.7234	1.8843	1.3907	1.4724	1.5314
UCB	1.8569	1.8425	1.8274	1.8277	1.8058
UCT	1.8727	1.8350	1.8129	1.8363	1.7912
Split UCB	1.5467	1.8617	1.7907	1.8082	1.8098

Table 5.1: Discounted average payoff of the strategies created by simulated annealing vs. each opponent. Each row shows the scores achieved by the strategy against the different opponents.

discount factor, with performance gradually degrading as the discount factor gets further from the value it was optimized for.

Figure 5.17 shows the performance of the annealed strategies against opponents with different exploration values. The higher the exploration value of the opponent, the less efficient learning is, so all strategies generally perform worse as the opponent explores more.

Table 5.1 shows the performance of the annealed strategies against each opponent. Each annealed strategy is the best performer against the strategy it was trained against, but we can see that strategies developed against opponents which explore less are less robust against different opponents.

5.5 Sequential Immediately Reactive Ratio Enforcing Strategies

We have discussed how to use no-regret strategies to find the best response to immediately reactive strategies, and how to design an immediately reactive strategy to perform well against no-regret strategies, but both types of strategies still have a common flaw – poor performance in self-play. When two no-regret strategies play against each other, they generally perform at the level of the Nash equilibrium or slightly better, but far

short of a pareto-optimal outcome. When two immediately reactive strategies play against each other they might find a pareto-optimal outcome, but they can also perform arbitrarily badly, possibly getting average payoffs as low as 0. In addition, when there is uncertainty about the strategy of the opponent, neither strategy performs well.

Consider how an agent should play when there is a 50% chance the opponent is playing an immediately reactive strategy and a 50% chance they are playing a no-regret strategy. In this situation, the best strategy is to initially play an immediately reactive strategy to take advantage of a no-regret playing opponent, and switch to the best response to the immediately reactive strategy when it becomes clear that the opponent is not playing a no-regret strategy. In this section we will describe a class of strategies which can encapsulate this decision, and discuss how the optimal strategy is affected by the discount factor of the agent and the distribution over opponents.

The strategies we have developed are called *sequential immediately reactive ratio enforcing strategies*; they can respond effectively to both immediately reactive strategies and to no-regret strategies. They are based on playing a sequence of immediately reactive strategies with descending thresholds, and stopping when the opponent matches the threshold.

The main idea of sequential immediately reactive strategies is to play a sequence of immediately reactive strategies, each with a threshold which is the best response for the opponent. The threshold is gradually lowered, with a speed proportional to the losses of the opponent (compared to the best result the opponent could have received against the threshold). This strategy will eventually reach a point where its threshold is compatible with the threshold of the opponent. On its own, this would not be sufficient – for example, if both agents are playing with a threshold of $\sqrt{2}/2$ and the last move of the opponent gave the agent a payoff of 0, the agent's reply will give the opponent a payoff of 0, and no progress would be made. The agent needs to do some exploration to

ensure that cooperation will occur when thresholds are compatible. We allow for this by introducing a forgiveness factor, which is an amount by which the agent will deviate from its current threshold.

Algorithm 7 Sequential Immediately Reactive Strategy

1: $round \leftarrow 1$
2: initialize opponent loss $o \leftarrow 0$
3: initialize forgiveness factor $\theta \in \mathbb{R}^+$
4: initialize threshold function $F : \mathbb{R}^+ \rightarrow [0, 1]$
5: **while** $(c_A, c_B) \leftarrow getOpponentMove()$ **do**
6: $\quad t \leftarrow F(o)$ ▷ get threshold value to use
7: $\quad c'_A \leftarrow c_A + \theta/round$ ▷ Forgive opponent slightly
8: $\quad c''_B \leftarrow \mathbb{T}(t, c'_A)$ ▷ Calculate response to opponent move
9: $\quad o \leftarrow o + 2\sqrt{1-t^2} - \sqrt{1-c_A^2} - c''_B$ ▷ Update opponent loss
10: $\quad round \leftarrow round + 1$
11: $\quad$ select the outcome $(\sqrt{1-c''^2_B}, c''_B)$ ▷ Make pareto-optimal choices
12: **end while**

The response function used takes a threshold τ, and the amount assigned to the agent by the opponent's last move (c_A, c_B), and returns an amount for the agent to give the opponent. We use the following response function:

$$\mathbb{T}(\tau, c_A) = \begin{cases} -\tau \times x + \sqrt{\tau^2 x^2 - \tau^2 - x^2 + 1} & if\ c_A < \tau \\ \sqrt{1-\tau^2} & if\ c_A \geq \tau \end{cases}$$

where $x = \tau\sqrt{1-c_A^2} - c_A\sqrt{1-\tau^2}$. The agent's response to an opponent move (c_A, c_B) will be $(\sqrt{1-\mathbb{T}(\tau, c_A)^2}, \mathbb{T}(\tau, c_A))$. This function enforces a minimum ratio of $\frac{\tau}{\sqrt{1-\tau^2}}$ between the payoff of the agent and the payoff of the opponent. If the opponent does not cooperate, this may be achieved by forcing the payoff of both to be zero.

The forgiveness factor determines how quickly a pareto-optimal outcome will be reached when both players have compatible thresholds. In order to detect compatible thresholds, the agent will need to do some exploration. In order to avoid spending

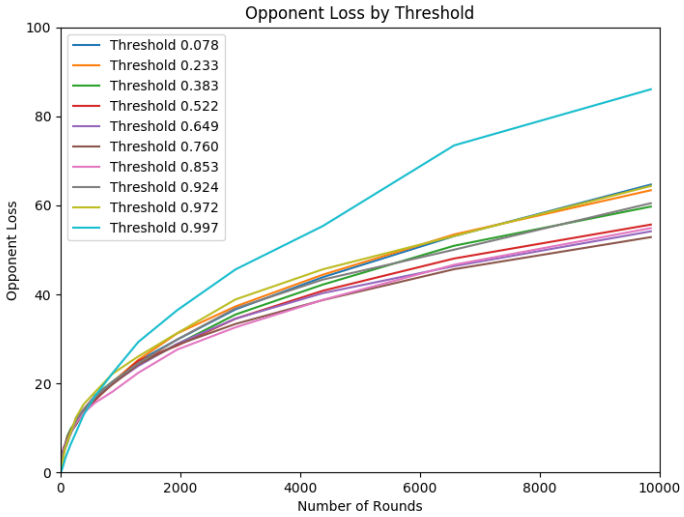

Figure 5.18: Total loss (regret) of Split-Bucket UCB playing against various fixed thresholds.

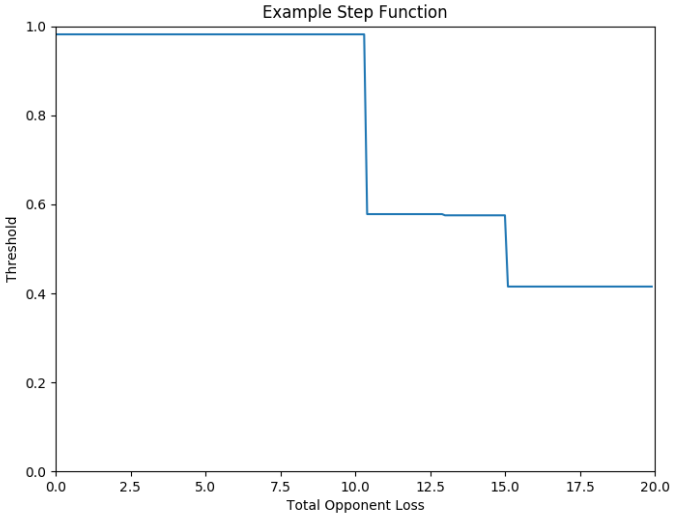

Figure 5.19: Example of a step threshold function. The y-axis shows the threshold and the x-axis shows the range of total opponent losses to which that threshold is applicable.

too much on exploration we use the harmonic sequence $(\frac{1}{1}, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, ...)$ to determine the magnitude of the deviation from the response function. Because the sequence converges

to 0, the effect of the exploration converges to zero. Because the sum of the subsequence starting at any point is infinite, regardless of how long it takes for two strategies to arrive at compatible thresholds, when they arrive at compatible thresholds, they will reach a pareto-optimal outcome.

The threshold function determines how rapidly the strategy will compromise with an opponent which fails to learn the optimal response to the agent's threshold. The function uses the opponent's loss to determine the appropriate threshold in order to ensure that the opponent cannot manipulate the threshold value without sacrificing its own payoff. Figure 5.18 shows the loss (regret) of a Split-Bucket UCB opponent playing against various constant thresholds. The loss is approximately logarithmic in the number of rounds, which is not surprising given the regret bounds of the learning algorithm.

We use simulated annealing (Algorithm 5) to find the best threshold function for a given combination of prior and discount factor. We represent each threshold function as a step function containing 4 thresholds. Figure 5.19 shows a sample threshold function. The general structure of a threshold function is a monotonically decreasing function where the agent gradually lowers its threshold as it becomes clear that the opponent will not play the best response to a higher threshold.

Figure 5.20 shows how the threshold function depends on the probability that the opponent is playing UCB. It shows the threshold function found with simulated annealing against an opponent which either plays UCB, or plays an immediately reactive function to which the best response is .8. When it is more likely that the opponent is using UCB, the optimal function found by simulated annealing holds out longer at the initial high threshold.

Figure 5.21 shows how the threshold function is affected by the amount of exploration done when the opponent is using UCB. It shows the functions found with simulated

annealing against an opponent which is equally likely to be using UCB or an immediately reactive function to which the best response is .6. When the opponent explores more the threshold function waits for more loss before it abandons its preferred solution, and it adopts an initial threshold which is slightly less greedy.

Figure 5.22 shows how the optimal threshold function is affected by the threshold of the opponent. It shows the functions found with simulated annealing against an opponent which is either using UCB or is an immediately reactive function as the threshold of the immediately reactive function changes. In this case, the difference between the optimal threshold functions is only in the threshold they adopt after abandoning the possibility that the opponent is using UCB.

Figure 5.23 shows how an agent using a threshold function can adapt when the opponent is definitely not using UCB. It shows the functions found with simulated annealing against an immediately reactive opponent which is using one of two different thresholds. For this class of opponent there is no point in waiting to see if the opponent will learn. The opponent either has a low threshold, so it will immediately accept the agent's current threshold and incur no further loss, or it has a high threshold, in which case the agent should immediately adjust to it. We look at the effect of varying the relative probabilities of a low or high threshold and the figure doesn't show a significant effect. This is because the opponent will rapidly incur loss against any incompatible threshold, so the threshold function will rapidly adjust.

Figure 5.24 shows how the optimal threshold function deals with an opponent which is either using a threshold function or using an immediately reactive function. It shows the functions found with simulated annealing against an opponent which starts out with an initial threshold of .8, but has a chance of lowering it to .5 after its opponent has lost 5 against the .8 threshold. The figure shows the effect of varying the probability of the opponent lowering its threshold. When the opponent is more likely to switch to a

lower threshold, the agent will maintain a high threshold for a longer period of time.

Figure 5.25 shows how the optimal threshold function deals with an opponent which is either using a threshold function or using an immediately reactive function. It shows the functions found with simulated annealing against a variety of initial threshold values. When the initial threshold is higher, the agent will hold out for a greater opponent loss before adopting a best response to the initial threshold. This is because a higher initial threshold will adopt a more expensive punishment strategy against the agent.

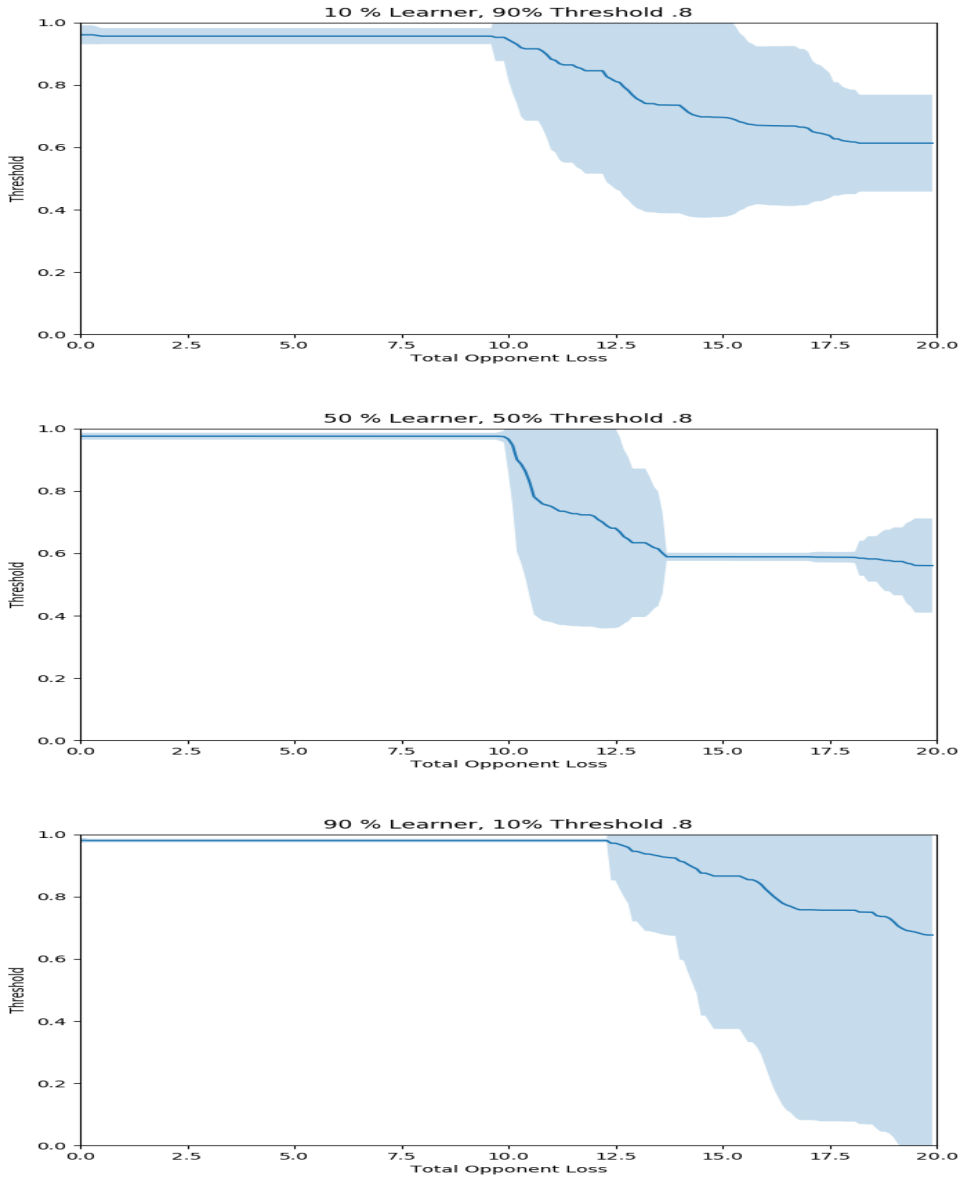

Figure 5.20: Strategies found using simulated annealing against a distribution over opponents including a UCB opponent, and a threshold opponent with a threshold of .8. The relative likelihood of the two opponents varies from 90%-10% (left) 50%-50% (center) 10%-90% (right). The shaded region shows the 95% confidence interval.

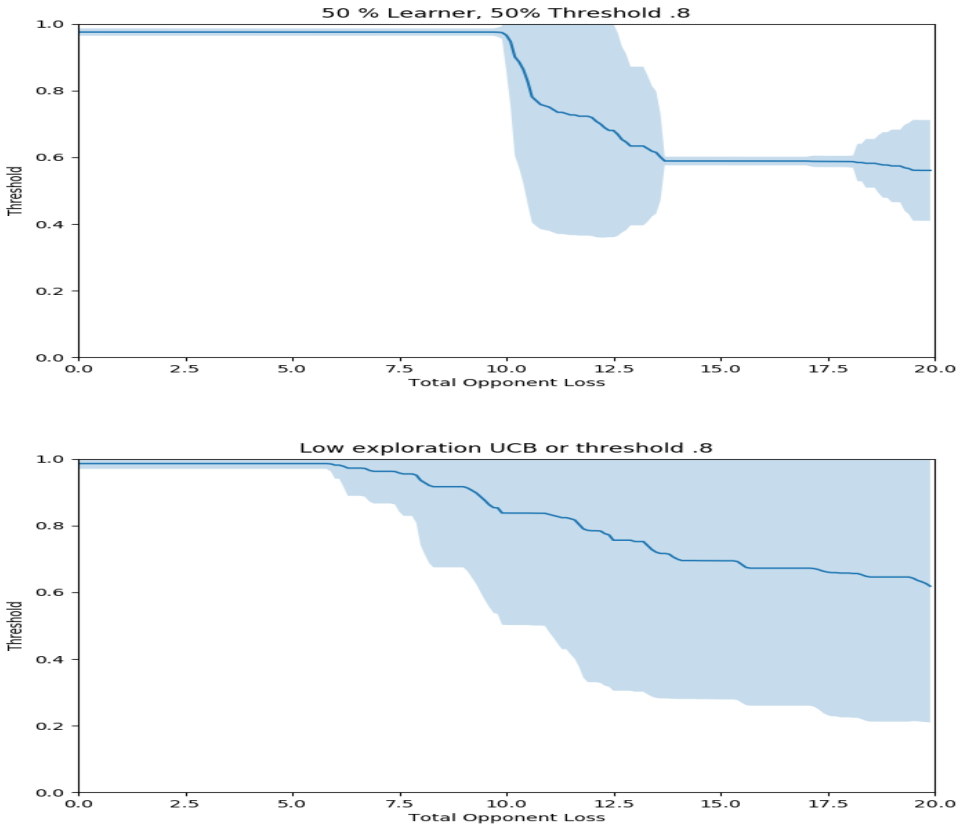

Figure 5.21: Strategies found using simulated annealing against a distribution over opponents including a UCB opponent, and a threshold opponent with a threshold of .8. The exploration factor of the UCB varies from 1.0 (top) to .125 (bottom). The shaded region shows the 95% confidence interval.

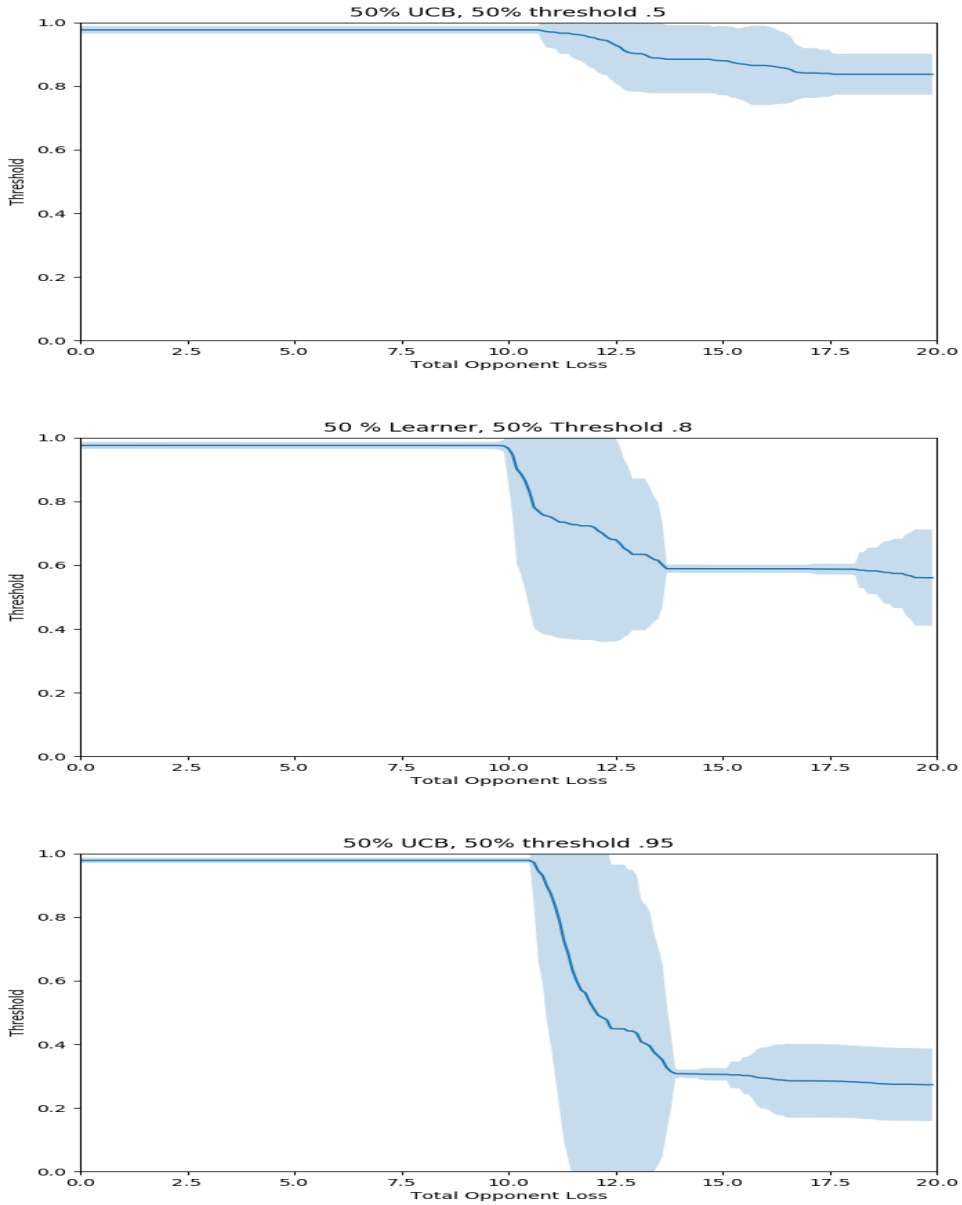

Figure 5.22: Strategies found using simulated annealing against a distribution over opponents including a UCB opponent, and a threshold opponent with a threshold of .5 (top), .8 (middle), and .95 (bottom). The shaded region shows the 95% confidence interval.

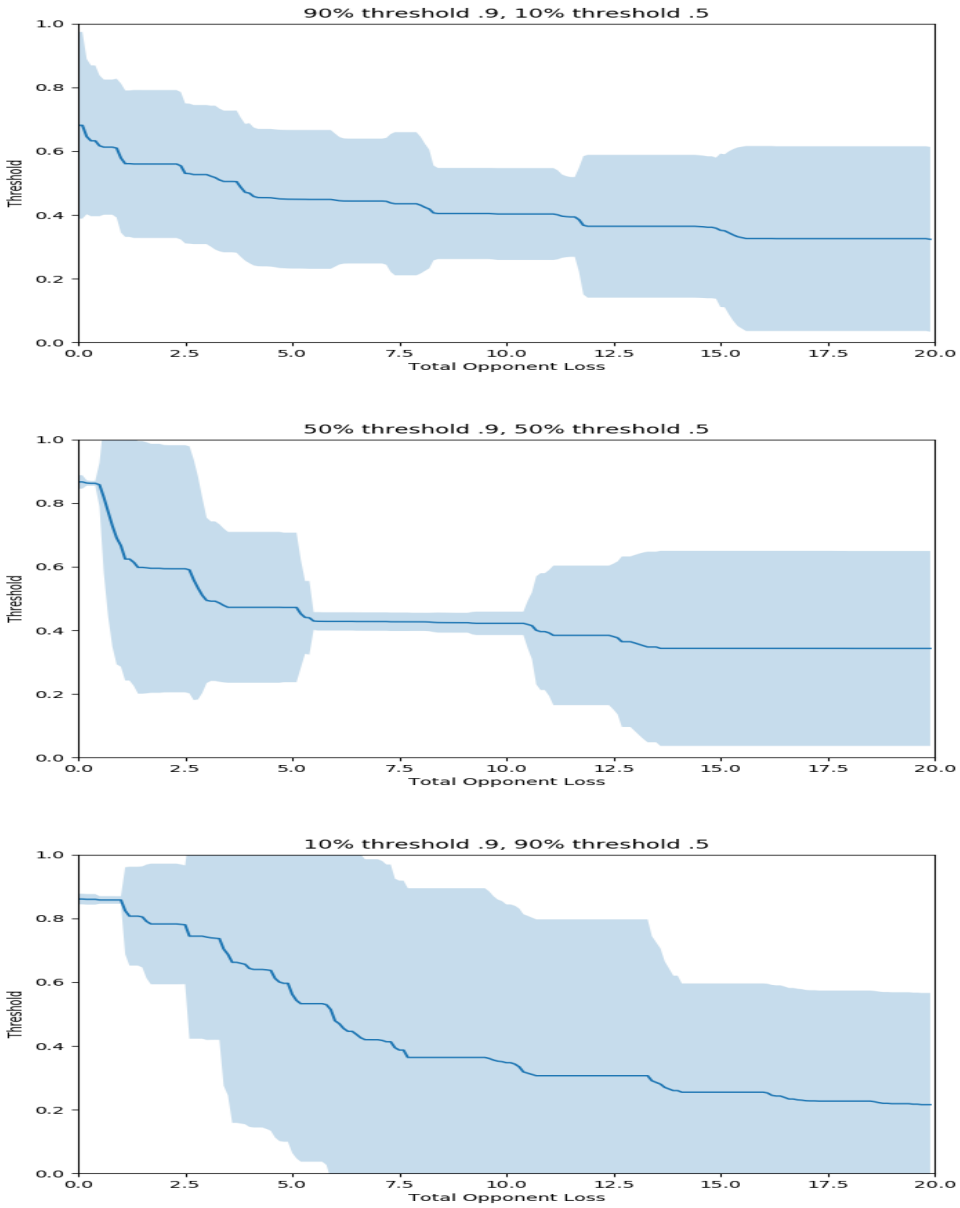

Figure 5.23: Strategies found using simulated annealing against a distribution over opponents with 2 different thresholds, .95 and .5. The relative probability varies from biased towards the high threshold (top) to biased towards the low threshold (bottom).

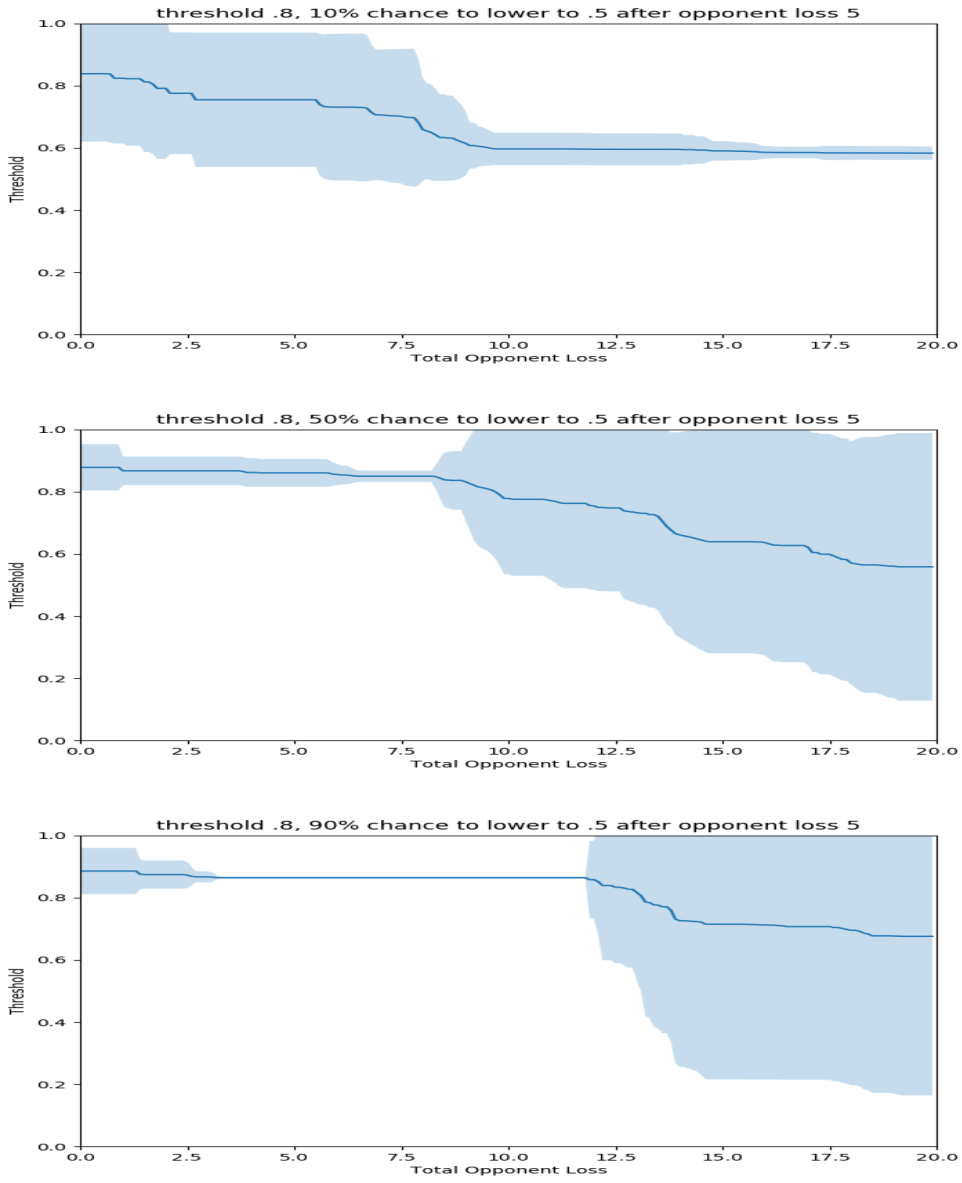

Figure 5.24: Strategies found using simulated annealing against an opponent with an initial threshold of .8, with a 10% (top), 50% (middle), or 90% (bottom) chance of switching to a threshold of .5 after an opponent loss of 5. The shaded region shows the 95% confidence interval.

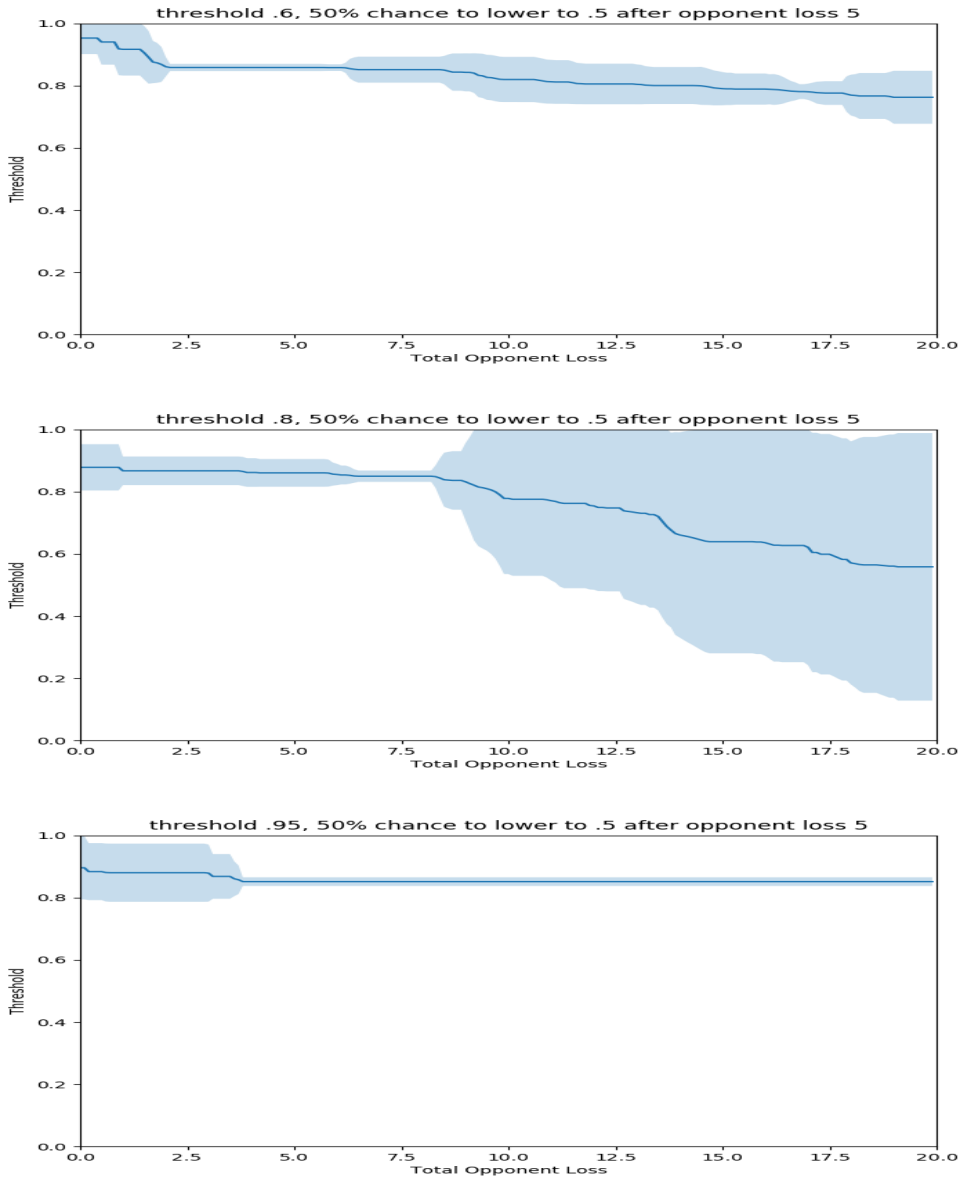

Figure 5.25: Strategies found using simulated annealing against an opponent with an initial threshold of .6 (top), .8 (middle) or .95 (bottom) which has a 50% chance to drop to a threshold of .5 after observing a total opponent loss of 5. The shaded region shows the 95% confidence interval.

5.6 Conclusions and Future Work

The Gift Exchange game is a suitable platform to study problems of cooperation and negotiation. A fundamental problem when dealing with an opponent is how the agent's actions will affect the future behavior of the opponent. The Gift Exchange game allows us to focus on that problem without needing to analyze complexities in the game, because the effect of an action on the outcome of the game is immediately evident. Furthermore, there is no motivation for the opponent to select any outcome other than the outcome they want to achieve because every outcome that can be achieved is achievable with a single action.

An agent should be able to find a best response to a simple opponent. We have described how to adapt no regret algorithms to the Gift Exchange game to find a best response to an agent which plays an immediately reactive strategy. We have described how such a strategy can be taken advantage of by choosing an appropriate immediately reactive strategy. Finally, we have described a class of strategies which can perform well against immediately reactive strategies, no-regret strategies, and in self-play. The parameters chosen for these strategies form an implicit representation of the discount factor and prior distribution of the agent.

The results from the Gift Exchange game suggest that in a general-sum environment with repeated interactions with an opponent which is too complex to explicitly predict an effective strategy is to enforce a ratio between the opponent's performance and the agent's performance and adjust the target ratio according to the loss experienced by the opponent.

Future work on the Gift Exchange game is divided into two areas: exploring the effect of alterations to the players or the choice sets on the dynamics of the game, and adapting strategies developed for the Gift Exchange to other games.

In the Gift Exchange game players interact in a symmetric unchanging environment,

but it is easy to alter the parameters to explore more options. By assigning players different choice sets we can introduce a power differential between the players. We can vary the size of the choice set each round, which would require players using reciprocation to track some notion of debt. We can vary the efficiency of cooperation by altering the shape of the choice set. We can use asymmetric choice sets to cause one player to care more about the outcome in a particular round than the other. Finally, it would be quite easy to introduce additional players.

The second main area of future work lies in extending strategies developed for the Gift Exchange game to other games. If the agent can estimate the expected impact of an action on its payoff and the opponent's payoff, it can treat the opponent's choice of action as a choice of payoff in a Gift Exchange game. It is important to form the estimate using only public information to ensure that the estimated intent reflects the opponent's actual intent. The agent can then respond using a strategy from the Gift Exchange game to inform its level of cooperation.

Chapter 6

Conclusions and Future Work

This dissertation is focused on the problem of cooperation between self-interested agents. In environments where agent's interests are diametrically opposed cooperation is impossible, and the agent can safely assume that the opponent will take the worst possible action for the agent. In the event that the agent can predict the behavior of the opponent there are some things the agent can do to take advantage of the prediction [53] [44] but performance in that situation is less critical because the agent has an advantage because the opponent is predictable. In environments where the agent's interests are aligned with that of the opponent, cooperation is inevitable – the problem is one of coordination. This is still a difficult problem in many environments, but there is no need to attempt to determine the intentions of the opponent. We are primarily concerned with situations where neither cooperation nor competition is inevitable and the opponent cannot be predicted - determining the intentions of the opponent and manipulating them is the main problem for the agent.

The first main group of contributions in this dissertation are concerned with identifying strategies in general-sum games which indicate and implement cooperative intent. We introduced repeated randomly generated games as an environment to look at the

problem of identifying cooperative strategies. We showed how attitude can be used to identify cooperative strategies and explore the effects of various parameters of the environment on the performance of cooperative strategies generated by adopting attitude values. We showed how the attitude values used by the opponent can be learned using a particle filter. Finally we showed how agents can use attitude to implement reciprocating strategies to achieve cooperation in repeated randomly generated normal form games.

When using attitude values to cooperate the agent uses the Nash equilibrium of the modified game. Using the Nash equilibrium means that the agent discards any prediction it makes of opponent behavior; to avoid doing so we have created a method of using the prediction more directly. Best-responding to the prediction means that the agent is vulnerable to errors in the prediction, and leads to instability when attempting to use attitude. We have developed Restricted Stackelberg Response with Safety (RSRS), a method of responding to a prediction in a general-sum normal form game that guards against a best responding opponent as well as the worst case outcome. In the process of developing RSRS we created a proof that Restricted Nash Response [44] creates a general-sum game with a unique equilibrium. We developed a technique to generate parameter values for the prediction weight parameter Finally, we show the performance of RSRS compared to a number of different algorithms against a variety of opponents.

Finally, we looked at the problem of how much to cooperate. When we developed reciprocation using attitude, we created agents which reciprocated a fixed amount with the goal of reaching a fair outcome. However, there may be multiple pareto-optimal outcomes and the agents may value them differently, which leads to the problem of how agents should select an outcome. More generally, this can be phrased as the problem of how to manipulate the opponent into taking actions that benefit the agent, especially when it is too complex to calculate a best response to the opponent. The problem of

how to manipulate an opponent is often complicated by the complexity of analyzing the environment or game in which the interaction takes place. We have developed the Gift Exchange game as a platform to study the problem of how to manipulate the opponent without the distraction of analyzing the environment. We have developed a modification of the UCB algorithm, Split-Bucket UCB, which an agent can use to find a best response to an immediately reactive opponent in the continuous version of the Gift Exchange game. In the discrete game we have described optimal learning strategies against an immediately reactive opponent, and the optimal strategies to take advantage of those learning strategies. We have used simulated annealing to find optimal immediately reactive strategies for a particular opponent and discount factor and run experiments to see how those factors affect the optimal strategy. Finally, we have developed a class of strategies, Sequential Immediately Reactive Ratio Enforcing Strategies, which can perform well against a wide variety of priors over opponent strategies and discount factors.

References

[1] Alon Altman, Avivit Bercovici-Boden, and Moshe Tennenholtz. Learning in one-shot strategic form games. In *Proc. European Conf. on Machine Learning*, pages 6–17. Springer, 2006.

[2] M. Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A tutorial on particle filters for online nonlinear/non-gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, 2002.

[3] Tsz-Chiu Au and Dana S. Nau. Accident or intention: That is the question (in the noisy iterated prisoner's dilemma). In *Proc. Int'l Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 561–568, 2006.

[4] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.

[5] Peter Auer, Ronald Ortner, and Csaba Szepesvári. Improved rates for the stochastic continuum-armed bandit problem. In *International Conference on Computational Learning Theory*, pages 454–468. Springer, 2007.

[6] R. M. Axelrod. *The evolution of cooperation*. Basic Books, 1984.

[7] Tim Baarslag, Mark J. C. Hendrikx, Koen V. Hindriks, and Catholijn M Jonker. Learning about the opponent in automated bilateral negotiation: a comprehensive

survey of opponent modeling techniques. *Journal of Autonomous Agents and Multi-agent Systems*, 30(5):849–898, 2016.

[8] Nolan Bard, Michael Johanson, Neil Burch, and Michael Bowling. Online implicit agent modelling. In *Proc. Int'l Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 255–262, 2013.

[9] Michael Bowling. Convergence and no-regret in multiagent learning. In *Advances in Neural Information Processing Systems (NIPS)*, volume 17, pages 209–216, 2005.

[10] Michael Bowling and Manuela Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136:215–250, 2002.

[11] R. I. Brafman and M. Tennenholtz. R-MAX a general polynomial time algorithm for near-optimal reinforcement learning. *The Journal of Machine Learning Research*, 3:213–231, 2003.

[12] Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge University Press, 2006.

[13] Doran Chakraborty. Convergence, targeted optimality and safety in multiagent learning. In *Sample Efficient Multiagent Learning in the Presence of Markovian Agents*, pages 29–47. Springer, 2014.

[14] Vincent Conitzer and Tuomas Sandholm. Computing the optimal strategy to commit to. In *Proc. Conference on Electronic Commerce*, pages 82–90, 2006.

[15] Vincent Conitzer and Tuomas Sandholm. A technique for reducing normal-form games to compute a Nash equilibrium. In *Proc. Int'l Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 537–544, 2006.

[16] Vincent Conitzer and Tuomas Sandholm. AWESOME: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. *Machine Learning*, 67(1–2):23–43, 2007.

[17] Pierre-Arnaud Coquelin and Rémi Munos. Bandit algorithms for tree search. In *Proc. Conf. on Uncertainty in Artificial Intelligence (UAI)*, UAI'07, pages 67–74, 2007.

[18] Jacob W Crandall. Robust learning for repeated stochastic games via meta-gaming. In *Proc. Int'l Joint Conf. on Artificial intelligence (IJCAI)*, pages 3416–3422, 2015.

[19] Steven Damer and Maria Gini. Achieving cooperation in a minimally constrained environment. In *Proc. of the AAAI Conf. on Artificial Intelligence*, pages 57–62, July 2008.

[20] Steven Damer and Maria Gini. Learning to cooperate in normal form games. Technical report, AAAI Technical Report WS-10-03, July 2010.

[21] Steven Damer and Maria Gini. Extended abstract: Friend or foe? detecting an opponent's attitude in normal form games. In *Proc. Int'l Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2011.

[22] Steven Damer and Maria Gini. Cooperation without exploitation between self-interested agents. In *Proc. Intelligent Autonomous Systems (IAS)*, 2012.

[23] Steven Damer and Maria Gini. Using predictions while avoiding exploitation in general-sum normal form games. In *Proc. Int'l Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS)*, May 2017.

[24] Steven Damer, Maria Gini, and Jeffrey S. Rosenschein. Discrete gift exchange game: Effects of limited choices on opponent interaction). In *GAIW: Games, Agents and Incentives Workshop at AAMAS*, 2019.

[25] Steven Damer, Maria Gini, and Jeffrey S. Rosenschein. The gift exchange game: Managing opponent actions (extended abstract). In *Proc. Int'l Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 1913–1915, 2019.

[26] Enrique Munoz De Cote and Nick Jennings. Planning against fictitious players in repeated normal form games. In *Proc. Int'l Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 1073–1080, 2010.

[27] Andreas Diekmann. The power of reciprocity: Fairness, reciprocity, and stakes in variants of the dictator game. *Journal of Conflict Resolution*, 48(4):487–505, 2004.

[28] Mohamed Elidrisi, Nicholas Johnson, Maria Gini, and Jacob Crandall. Fast adaptive learning in repeated stochastic games by game abstraction. In *Proc. Int'l Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 1141–1148, 2014.

[29] Christoph Engel. Dictator games: A meta study. *Experimental Economics*, 14(4):583–610, 2011.

[30] Ernst Fehr and Klaus M. Schmidt. A theory of fairness, competition and cooperation. *Quarterly Journal of Economics*, 114:817–68, 1999.

[31] Bruce D. Fitzgerald. Self-interest or altruism. *Journal of Conflict Resolution*, 19:462–479, 1975.

[32] Jakob Foerster, Richard Y Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. In *Proc.*

Int'l Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS), pages 122–130, 2018.

[33] N. Frohlich. Self-Interest or Altruism, What Difference? *Journal of Conflict Resolution*, 18(1):55–73, 1974.

[34] Drew Fudenberg and David K. Levine. *The Theory of Learning in Games*. MIT Press, 1998.

[35] Drew Fudenberg and Jean Tirole. *Game theory*. The MIT Press, 1991.

[36] Sam Ganzfried and Tuomas Sandholm. Game theory-based opponent modeling in large imperfect-information games. In *Proc. Int'l Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 533–540, 2011.

[37] Sam Ganzfried and Tuomas Sandholm. Safe opponent exploitation. *ACM Transactions on Economics and Computation*, 3(2):8:1–8:28, April 2015.

[38] Srihari Govindan and Robert Wilson. Refinements of Nash equilibrium. In Steven N. Durlauf and Lawrence E. Blume, editors, *The New Palgrave Dictionary of Economics, 2nd Edition*. Palgrave Macmillan, 2008.

[39] D. Griesinger and J. Livingston. Towards a model of interpersonal motivation in experimental games. *Behavioral Science*, 18:173–188, 1973.

[40] Werner Güth, Rolf Schmittberger, and Bernd Schwarze. An experimental analysis of ultimatum bargaining. *Journal of Economic Behavior & Organization*, 3(4):367–388, 1982.

[41] Dong Hao, Kai Li, and Tao Zhou. Payoff control in the iterated prisoner's dilemma. In *Proc. Int'l Joint Conf. on Artificial intelligence (IJCAI)*, pages 296–302, 2018.

[42] Christopher J Hazard. Por favor? favor reciprocation when agents have private discounting. In *Workshop on Coordination, Organizations, Institutions and Norms (COIN) at AAAI*, pages 9–16, 2008.

[43] Pablo Hernandez-Leal, Yusen Zhan, Matthew E. Taylor, L. Enrique Sucar, and Enrique Munoz de Cote. An exploration strategy for non-stationary opponents. *Journal of Autonomous Agents and Multi-agent Systems*, 31(5):971–1002, 2017.

[44] Michael Johanson, Martin Zinkevich, and Michael Bowling. Computing robust counter-strategies. In *Advances in Neural Information Processing Systems (NIPS)*, pages 721–728, 2007.

[45] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1):99–134, 1998.

[46] Robert D Kleinberg. Nearly tight bounds for the continuum-armed bandit problem. In *Advances in Neural Information Processing Systems (NIPS)*, pages 697–704, 2005.

[47] Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo planning. In *European Conference on Machine Learning*, pages 282–293. Springer, 2006.

[48] Levente Kocsis, Csaba Szepesvári, and Jan Willemson. Improved Monte-Carlo search. *Univ. Tartu, Estonia, Tech. Rep*, 1, 2006.

[49] Sarit Kraus, Jeffrey S. Rosenschein, and Maier Fenster. Exploiting focal points among alternative solutions: Two approaches. *Annals of Mathematics and Artificial Intelligence*, 28(1-4):187–258, 2000.

[50] Joel Z. Leibo, Vinicius Zambaldi, Marc Lanctot, Janusz Marecki, and Thore Graepel. Multi-agent reinforcement learning in sequential social dilemmas. In *Proc. Int'l Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 464–473, 2017.

[51] David K. Levine. Modeling altruism and spitefulness in experiments. *Review of Economic Dynamics*, 1:593–622, 1998.

[52] M.L. Littman and P. Stone. Leading best-response strategies in repeated games. In *Workshop on Economic Agents, Models, and Mechanisms at IJCAI*, 2001.

[53] Peter McCracken and Michael Bowling. Safe strategies for agent modelling in games. In *AAAI Fall Symposium on Artificial Multi-agent Learning*, October 2004.

[54] Richard D McKelvey and Andrew McLennan. Computation of equilibria in finite games. *Handbook of computational economics*, 1:87–142, 1996.

[55] C. Musso, N. Oudjane, and Franç Legland. Improving regularized particle filters. In A. Doucet, N. de Freitas, and N. Gordon, editors, *Sequential Monte Carlo Methods in Practice*, pages 247–271. Springer-Verlag, New York, 2001.

[56] John F Nash Jr. The bargaining problem. *Econometrica: Journal of the Econometric Society*, pages 155–162, 1950.

[57] Martin A. Nowak. Five rules for the evolution of cooperation. *Science*, 314:1560–1563, 2006.

[58] Martin A. Nowak and Karl Sigmund. A strategy of win-stay, lose-shift that outperforms Tit for Tat in the Prisoner's Dilemma game. *Nature*, 364:56–58, 1993.

[59] James Pita, Manish Jain, Milind Tambe, Fernando Ordóñez, and Sarit Kraus. Robust solutions to Stackelberg games: addressing bounded rationality and limited observations in human cognition. *Artificial Intelligence*, 174(15):1142–1171, 2010.

[60] R. Powers, Y. Shoham, and T. Vu. A general criterion and an algorithmic framework for learning in multi-agent systems. *Machine Learning*, 67(1–2):45–76, 2007.

[61] Matthew Rabin. Incorporating fairness into game theory and economics. *The American Economic Review*, 83(5):1281–1302, 1993.

[62] A. Rapoport and A. Chammah. *Prisoner's dilemma: A study in conflict and cooperation*. University of Michigan Press, 1970.

[63] Ariel Rubinstein. Perfect equilibrium in a bargaining model. *Econometrica: Journal of the Econometric Society*, pages 97–109, 1982.

[64] Sabyasachi Saha, Sandip Sen, and Partha Sarathi Dutta. Helping based on future expectations. In *Proc. Int'l Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 289–296, 2003.

[65] David F. Sally. Two economic applications of sympathy. *The Journal of Law, Economics, and Organization*, 18:455–487, 2002.

[66] T. Sandholm and S. Singh. Lossy stochastic game abstraction with bounds. In *Prod. of the 13th ACM Conf. on Electronic Commerce*, pages 880–897, 2012.

[67] T. W. Sandholm and R. H. Crites. Multiagent reinforcement learning and the iterated Prisoner's Dilemma. *Biosystems Journal*, 37:147–166, 1995.

[68] Lloyd S. Shapley. Stochastic games. *Proceedings of the NAS*, 39:1095–1100, 1953.

[69] Yoav Shoham, Rob Powers, and Trond Grenager. Multi-agent reinforcement learning: a critical survey. Technical report, Stanford University, 2003.

[70] Peter Stone, Gal A. Kaminka, Sarit Kraus, Jeffrey R. Rosenschein, and Noa Agmon. Teaching and leading an ad hoc teammate: Collaboration without pre-coordination. *Artificial Intelligence*, 203:35–65, 2013.

[71] T. Trivers. The evolution of reciprocal altruism. *Quarterly Review of Biology*, 36:35–57, 1971.

[72] Stefan Valavanis. The resolution of conflict when utilities interact. *The Journal of Conflict Resolution*, 2(2):156–169, 1958.

[73] H. von Stackelberg. *Market Structure and Equilibrium*. Springer, 2011. Translation to English.

[74] Z Wang, A Boularias, K Mülling, and J Peters. Balancing safety and exploitability in opponent modeling. In *Proc. of the AAAI Conf. on Artificial Intelligence*, pages 1515–1520, August 2011.

[75] Song Zuo and Pingzhong Tang. Optimal machine strategies to commit to in two-person repeated games. In *Proc. of the AAAI Conf. on Artificial Intelligence*, pages 1071–1078, 2015.