Exploration Strategies for Model-based Learning in Multi-agent Systems

David Carmel * and Shaul Markovitch
Computer Science Department,
Technion, Haifa 32000, Israel.
{*carmel,shaulm*}*@cs.technion.ac.il*

(Received ..... ; Accepted in final form .....)

Abstract. An agent that interacts with other agents in *multi-agent systems* can benefit significantly from adapting to the others. When performing active learning, every agent's action affects the interaction process in two ways: The effect on the expected reward according to the current knowledge held by the agent, and the effect on the acquired knowledge, and hence, on future rewards expected to be received. The agent must therefore make a tradeoff between the wish to exploit its current knowledge, and the wish to explore other alternatives, to improve its knowledge for better decisions in the future. The goal of this work is to develop exploration strategies for a model-based learning agent to handle its encounters with other agents in a common environment. We first show how to incorporate exploration methods usually used in reinforcement learning into model-based learning. We then demonstrate the risk involved in exploration – an exploratory action taken by the agent can yield a better model of the other agent but also carries the risk of putting the agent into a much worse position.

We present the *lookahead-based exploration strategy* that evaluates actions according to their expected utility, their expected contribution to the acquired knowledge, and the risk they carry. Instead of holding one model, the agent maintains a *mixed opponent model*, a belief distribution over a set of models that reflects its uncertainty about the opponent's strategy. Every action is evaluated according to its long run contribution to the expected utility and to the knowledge regarding the opponent's strategy. Risky actions are more likely to be detected by considering their expected outcome according to the alternative models of the opponent's behavior. We present an efficient algorithm that returns an *almost optimal* exploration plan against the mixed model and provide a proof of its correctness and an analysis of its complexity.

We report experimental results in the *Iterated Prisoner's Dilemma* domain, comparing the capabilities of the different exploration strategies. The experiments demonstrate the superiority of lookahead-based exploration over other exploration methods.

Key words: Model-based learning, Exploration, Multi-agent systems

1. Introduction

Consider a *multi-agent system* (MAS) where agents repeatedly interact with other agents in order to achieve their private goals. The agents may have different preferences, different negotiation skills and different

* Now at IBM - Haifa Research Lab, Haifa 31905, Israel.

levels of knowledge. For such a system an agent should be designed to efficiently interact with other agents who may be potential partners, or alternatively, competing opponents.

Designing an effective interaction strategy for your agent requires prior knowledge about any other agent that might be involved in the system. However,the *agents* are completely autonomous and are not necessarily familiar with the other agents' strategies. One way to deal with this difficulty is to endow the agent with the ability to adapt to the other agents.

Reinforcement learning (RL) is a popular technique used by adaptive agents in MAS for learning interaction strategies with other agents (Sen *et al.*, 1994; Littman, 1994; Weiß and Sen, 1996; Sandholm and Crites, 1995). RL is based on the idea that the tendency to produce an action should be reinforced if it produces favorable results, and weakened if it produces unfavorable results. RL spends little computation resources per example but requires large number of examples.

In previous work we present a *model-based* approach for learning interaction strategies (Carmel and Markovitch, 1996; Carmel and Markovitch, 1998) which reduces the number of examples needed for adaptation, by investing more computational resources in deeper analysis of interaction experience. The learning agent makes use of an explicit model of the other agent's strategy to generate expectations about its behavior. At any stage of the game, the agent updates the opponent model to be consistent with current sample of the opponent's behavior. It then follows the optimal response against the current model.

Models can also be used by RL agents in order to reduce the number of experiences needed for adaptation. Dyna (Sutton, 1990) simultaneously uses experiences to adjust the agent's policy and to build a world model. While the opponent model in our approach is used for computing the best response against the opponent, Dyna uses the world model to empirically adapt a strategy through simulation. Dyna requires an order of magnitude fewer real experiences than model-free RL to arrive to an optimal policy, but consumes about six times more computational resources (Kaelbling *et al.*, 1996).

The model-based approach gives priority to actions with the highest expected utility. But such policy does not consider the effect of the agent's behavior on the learning process, and ignores the contribution of agent's activity to the exploration of the opponent's strategy. In essence, every action affects the interaction process in two ways:

1. The effect on the expected reward according to the current knowledge held by the agent.

2. The effect on the acquired knowledge, and hence, on future rewards expected to be received due to better planning.

The agent therefore must make a tradeoff between the wish to exploit its current knowledge and the wish to explore other alternatives, to improve its knowledge for better decisions in the future. This tradeoff is known in decision-theory as the *exploration versus exploitation dilemma*. Agents that engage in exploration to the exclusion of exploitation pay the cost of experimentation, without gaining benefits of the accumulated knowledge. Conversely, agents that engage in exploitation to the exclusion of exploration, are likely to find themselves trapped in sub-optimal behavior.

A good example for the above dilemma is the *multi-armed bandit* problem (Berry and Fristedt, 1985; Kaelbling, 1993; Gittins, 1989). A machine with k levers independently pays some amount of money each time a lever is pulled according to a fixed unknown distribution. The problem is to develop a strategy that gains the maximum payoff over time by choosing which lever to pull based on previous experience of lever pulling and their associated payoffs. For this problem, exploitation corresponds to choosing the estimated best arm, according to the current knowledge. Exploration corresponds to choosing of another arm for the aim of making estimations more accurate, and making better decisions in the future.

The exploration vs. exploitation problem has received significant attention within the RL research. Several methods were developed for incorporating exploration strategies into RL agents (Sutton, 1990; Thrun, 1992; Kaelbling, 1993; Singh *et al.*, 1998). *Undirected* methods are based on incorporating randomness into the decision procedure by assigning each action a positive probability of being selected and performed. *Directed* methods use the interaction history to evaluate the expected contribution of every action to exploration.

The goal of this work is to develop exploration strategies for model-based agents. We first show how to incorporate RL exploration methods into model-based learning. One problem with the existing exploration methods is that they do not take into consideration the risk involved in exploration. An exploratory action taken by the agent tests unfamiliar aspects of the other agent's behavior which can yield a better model of the other agent. However, such an action carries also the risk of putting the agent into a much worse position. Recently, some utility-based strategies has been suggested to guide exploration by considering the effects of a given plan on the reduction of the agent's uncertainty about its model of the world. Dayan and Sejnowsky (1996) claim that the agent's uncertainty should drive its exploration behavior. They

describe a method that uses *certainty equivalence approximation* that uses the mean values of the state variables, instead of the variables themselves, in expressions that determine the utility of the agent's actions. *Optimal experiments design* (Fedorov, 1972) is concerned with the design of experiments that are expected to minimize variances of the parameterized model, and therefore, to maximize the confidence in the given model. When applying this method to sequential decision making, we estimate how an addition of a new training example is expected to change the computed variance. Cohn (1994) applies this method for selecting examples to train an artificial neural network. Karakoulas (1995) presents a probabilistic algorithm that also applies a similar statistical selection procedure to decide how much exploration is needed for selecting a plan.

In this work we present the *lookahead-based exploration strategy* that drives exploration according to the player's uncertainty about its opponent model, but also considers the effect of exploration on the agent's position. It evaluates actions according to their expected utility, to their expected contribution to the acquired knowledge, and to the risk they carry. Instead of holding one model, the agent maintains a *mixed opponent model*, a belief distribution over a set of models that reflects its uncertainty about the opponent's strategy. Every action is evaluated according to its long run contribution to the expected utility, and to the knowledge regarding the opponent's strategy, expressed by the posterior probabilities over the set of models. Risky actions are detected by considering their expected outcome according to the alternative models of the opponent's behavior.

The rest of the paper is organized as follows: Section 2 describes the basic model of interaction studied in this work. An encounter among agents is described as a *game* and a sequence of encounters as a *repeated game*. For such a game, each agent has a finite set of alternative actions (moves) to choose from, and the game outcome is determined according to the joint moves performed by the agents. We study repeated games among *bounded rational* agents with restricted computational abilities. The agents' strategies are restricted to *regular strategies*, i.e., strategies that can be represented by deterministic finite automata, following Rubinstein's model for bounded rationality in repeated games (Rubinstein, 1986). We briefly describe our earlier work (Carmel and Markovitch, 1996; Carmel and Markovitch, 1998) on *model-based learning* for repeated games.

Section 3 addresses the necessity of exploration for model-based agents, and adjusts several exploration methods originally developed for

RL for model-based learning [1]. In Section 4 we describe the *lookahead-based exploration strategy* that deals with exploration using utility-based criteria usually used in decision-theory.

In Section 5 we show experimentally the necessity of exploration for model-based learning and show the superiority of lookahead-based exploration over uninformed exploration methods. Finally, the conclusions of the work are given in Section 6.

2. The Basic Framework

To formalize the notion of interacting agents we consider a framework where an encounter between two agents is represented as a *two-player game* and a sequence of encounters as a *repeated game*. A *two-player game* is a tuple $G = \langle R_1, R_2, u_1, u_2 \rangle$, where R_1, R_2 are finite sets of alternative moves for the players (called *pure strategies*), and $u_1, u_2 : R_1 \times R_2 \rightarrow \Re$ are utility functions that define the utility of a joint move (r_1, r_2) for the players. For example, the *Prisoner's dilemma* (PD) is a two-player game, where each player has two actions, cooperate (c) and defect(d), $R_1 = R_2 = \{c, d\}$. The utility functions, u_1, u_2, are described by the payoff matrix shown in Figure 1.

		II c	II d
I c		3 / 3	5 / 0
I d		0 / 5	1 / 1

Figure 1. The payoff matrix for the Prisoner's dilemma game. Lower left numbers are the utilities for player I. Upper right numbers are the utilities for player II.

A sequence of encounters among agents is described as a repeated game, $G^{\#}$, based on the repetition of G an indefinite number of times. At any stage t of the game, the players choose their moves, $(r_1^t, r_2^t) \in R_1 \times R_2$, simultaneously. A history, $h(t)$ of $G^{\#}$, is a finite sequence of joint moves chosen by the agents up to the current stage of the game, $h(t) = \left\langle (r_1^0, r_2^0), (r_1^1, r_2^1), \ldots, (r_1^{t-1}, r_2^{t-1}) \right\rangle$. λ is the null history. $H(G^{\#})$ is the set of all finite histories for $G^{\#}$.

A *strategy* for player i, $s_i : H(G^{\#}) \rightarrow R_i, i \in \{1, 2\}$, is a function that takes a history and returns an action. $\mathcal{S}_i$ is the set of all possible strategies for player i in $G^{\#}$. A pair of strategies (s_1, s_2) defines an infinite sequence of joint moves (a path), $g_{(s_1,s_2)}(t)$, while playing the game $G^{\#}$.

[1] An earlier version of this section appeared in (Carmel and Markovitch, 1997).

$$\begin{aligned} g_{(s_1,s_2)}(0) &= \lambda \\ g_{(s_1,s_2)}(t+1) &= g_{(s_1,s_2)}(t) || \left(s_1(g_{(s_1,s_2)}(t)), s_2(g_{(s_1,s_2)}(t)) \right) \end{aligned} \quad (1)$$

The *discounted-sum* function is a utility function commonly used for analysis of repeated games:

$$U_i^{ds}(s_1, s_2) = \sum_{t=0}^{\infty} \gamma^t u_i(s_1(g_{(s_1,s_2)}(t)), s_2(g_{(s_1,s_2)}(t))) \quad (2)$$

$0 \leq \gamma < 1$ is a discount factor for future payoffs of player i. It can also be thought of as the estimation of player i for the probability that the game will be allowed to continue after the current move. It is easy to show that $U^{ds}(s_1, s_2)$ converges for any $\gamma < 1$.

Definition 1. A strategy $s^{opt}(s_j, U_i)$ will be called *best response* for player i with respect to strategy s_j and utility U_i, iff $\forall s \in \mathcal{S}_i$, $[U_i(s^{opt}(s_j, U_i), s_j) \geq U_i(s, s_j)]$.

The Iterated Prisoner's Dilemma (IPD) is an example of a repeated game based on the PD game that attracts significant attention in the game-theory literature. Tit-for-tat (TFT) is a simple well known strategy for IPD that has been shown to be very successful in IPD tournaments (Axelrod, 1984). It begins with cooperation and imitates the opponent's last action afterwards. The *best-response* against TFT with respect to U^{ds} depends on the discount parameter γ (Axelrod, 1984):

$$s^{opt}(\text{TFT}, U^{\gamma}) = \begin{cases} \text{all-c} & \frac{2}{3} \leq \gamma \\ \text{all-d} & \gamma \leq \frac{1}{4} \\ \text{“Alternate between c and d”} & \text{otherwise} \end{cases}$$

In many cases there is more than one *best response* strategy. For example, for $\gamma \geq \frac{2}{3}$, TFT is also a *best response* against itself as well as many other cooperative strategies.

One of the basic factors affecting the behavior of agents in MAS is the knowledge that they possess about each other. In this work we assume that each player is aware of the other player's set of actions, i.e., R_1, R_2 are *common knowledge*, while the players' preferences, u_1, u_2, are *private*. In such a framework, while the history of the game is *common knowledge*, each player predicts the future course of the game differently. The prediction of player i is based on the player's strategy s_i and on the player's belief about the opponent's strategy, $\hat{s}_j$. $\hat{s}_j$ will be called an *opponent model*.

How can a player acquire a model of its opponent's strategy? One source of information available for the player is the history of the game.

Another possible source of information is observed games between the opponent and other agents. Note that any history of length t provides a sample, $E_j(h(t)) = \{(h(k), r_j^k) | 0 \leq k \leq t\}$, of $t+1$ examples of the opponent's behavior, since any prefix of $h(t)$ is also a history of the game.

Given a learning algorithm L_i that infers an opponent model based on a sample of its behavior, and a utility function U_i, we can define the strategy $s_i^{U_i,L_i}$ of a model-based learning agent as

$$s_i^{U_i,L_i}(h(t)) = s_i^{opt}\left(L_i\left(E_j(h(t))\right), U_i\right)(h(t))$$

The above definition yields a model-based player (MB-agent) that adapts its strategy during the game. A MB-agent begins the game with an arbitrary opponent model $\hat{s}_j^0$, finds the *best response* $s_i^0 = s_i^{opt}(\hat{s}_j^0, U_i)$, and plays according to the *best response*, $r_i^0 = s_i^0(\lambda)$, where λ is the null history. At any stage t of the game, the MB-agent infers an updated opponent model by applying its learning algorithm L_i to the current sample of the opponent's behavior, $\hat{s}_j^t = L_i(E_j(h(t)))$. It then finds the best response against the current model, $s_i^t = s_i^{opt}(\hat{s}_j^t, U_i)$, and plays according to the best response $r_i^t = s_i^t(h(t))$. Figure 2 illustrates the general architecture of an on-line model-based learning agent for repeated games.

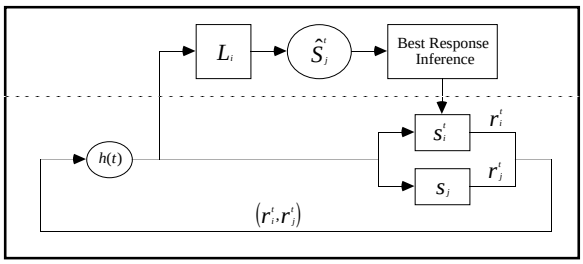

Figure 2. A general architecture for a model-based learning agent in repeated games.

The efficiency of this adaptive strategy mainly depends on the two main processes involved:

1. The finding of the best response against a given model.
2. The learning process of the opponent model.

Generally, computing the *best response* against a given model is too complicated for *bounded rational agents* (agents with limited computational resources). There is an extensive line of research in game theory on the problem of playing repeated games against computationally

bounded opponents (Papadimitriou, 1992; Fortnow and Whang, 1994; Freund *et al.*, 1995). See Kalai (1990) for a survey on *bounded rationality* in repeated games. The typical approach is to assume that the opponent's strategy is a member of some natural class of computationally bounded strategies. A natural measure of the complexity of a given strategy is the amount of computational resources needed for implementation. In this work we adopt a common restriction that the opponent uses *regular strategies*, i.e. strategies that can be represented by deterministic finite automata (DFA) (Rubinstein, 1986). The complexity of a regular strategy can be measured by the number of states of the minimal automaton implementing the strategy.

A DFA (*Moore machine*) is defined as a tuple $M = (Q, \Sigma_{in}, q_0, \delta, \Sigma_{out}, F)$, where Q is a non-empty finite set of states, Σ_{in} is the machine input alphabet, q_0 is the initial state, and Σ_{out} is the output alphabet. $\delta : Q \times \Sigma_{in} \rightarrow Q$ is a transition function. Σ_{in}^* is the set off all strings over Σ_{in}. δ is extended to the range $Q \times \Sigma_{in}^*$ in the usual way: $\delta(q, \lambda) = q$, $\delta(q, s\sigma) = \delta(\delta(q, s), \sigma)$. $F : Q \rightarrow \Sigma_{out}$ is the output function. $M(s) = F(\delta(q_0, s))$ is the output of M for a string $s \in \Sigma_{in}^*$. $|M|$ denotes the number of states of M.

A strategy for player i against opponent j is represented by a DFA, M_i, where $\Sigma_{in} = R_j$ and $\Sigma_{out} = R_i$. Given a history $\langle (r_1^0, r_2^0), (r_1^1, r_2^1), \ldots, (r_1^{t-1}, r_2^{t-1}) \rangle$, the move selected by M_i is $M_i(r_j^0 r_j^1 \ldots r_j^{t-1})$. For example, Figure 3 shows a DFA that implements the strategy TFT for the IPD game. The player's actions, (Σ_{out}), appear inside the machine states. The opponent's actions, (Σ_{in}), change the DFA states and appear next to the corresponding transitions. The initial state is marked by an arrow.

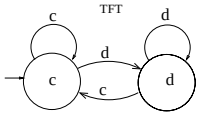

Figure 3. A DFA that implements the strategy TFT for the IPD game. The player's actions (Σ_{out}) appear inside the machine states. The opponent's actions, (Σ_{in}), change the DFA states and appear next to the corresponding transitions. The initial state is marked by an arrow.

In previous work (Carmel and Markovitch, 1996; Carmel and Markovitch, 1998), we presented a model-based strategy against regular opponents in repeated games. The strategy includes an efficient best-response procedure against any regular opponent model. The procedure is based on the work of Papadimitriou and Tsitsiklis, (1987) for finding an optimal policy against a Markov decision process using dynamic programming.

Under the assumption that the opponent's strategy can be modeled as DFA, and according to the *Occam Razor* assumption that smaller models might have a better prediction power, the MB-agent should infer the smallest DFA that is consistent with the sample of the opponent's behavior in the past in order to predict the opponent's actions in the future. Angluin (1987) describes the L^* algorithm that constructs a minimal model by consulting a teacher who can answer queries. Such an approach is not suitable for repeated game against an autonomous egocentric agent. We have developed a learning algorithm (Carmel and Markovitch, 1996; Carmel and Markovitch, 1998), named US-L^*, (unsupervised L^*), that constructs a regular model according to the following three guiding principles:

- Consistency: The new model must be consistent with the given sample.
- Compactness: A smaller model might have a better prediction power.
- Stability: The new model should be similar to the previous model as much as possible.

The algorithm maintains the same observation table, as L^* does, for representing the learned model. When a new counterexample arrives, it tries to extend the model in a minimal fashion. At first, the algorithm constructs a new observation table that covers the data, including the new *counterexample*. Table construction requires a teacher for answering membership queries. In the absence of such a teacher the agent consults its previous model following the stability principle. After that, it arranges the table and constructs a new model consistent with the new table, and therefore with the new counterexample.

3. Exploration Strategies for Model-based Learning

The model-based strategy described in the previous section gives priority to actions with the highest expected utility, according to the current opponent model. Such a behavior, however, does not consider the effect of the player's action on the learning process. It is quite possible that a sub-optimal action will yield long-run benefit by exploring unknown aspects of the opponent's behavior. In the following subsections we will modify the model-based learner to take both exploration and exploitation into consideration.

3.1. Exploration versus Exploitation

Actions taken by the agent, together with the opponent's responses, are used as examples by the model-based learner. The "exploratory value" of an example can be evaluated by predicting the expected value of the revised model as a result of processing the new example. A consistent example will strengthen the belief in the current model. A counterexample will lead to a construction of a new better model by the learner.

In *supervised learning*, it is the teacher's role to provide counterexamples for guiding the learning process. In the absence of an assisting teacher, the agent must search for counterexamples by itself. The *model-based* learner, described in Section 2, may encounter counterexamples by chance during the interaction process as long as it possesses a model that is not equivalent to the opponent's strategy. But such an approach might converge to sub-optimal behavior. The deterministic nature of the model-based strategy decreases the likelihood of observing counterexamples, since following the same behavior repeatedly prevents exploration of unknown aspects of the opponent's strategy.

The left part of Figure 4 shows an example of an opponent's strategy for IPD. The opponent will defect as long as the player does, but will cooperate forever after one cooperation of the player. The best-response against this strategy is "play c then all-d" which yields a utility of $\frac{5\gamma}{1-\gamma}$. If the player uses the model shown in the right part of the figure then the best-response corresponding to this model is "all-d" yielding utility of $\frac{1}{1-\gamma}$ which is suboptimal for any $\gamma \geq \frac{1}{5}$. A model-based player holding this model will repeatedly defect, preventing it from exploring action "c" and observing counterexamples. Hence, the wrong model will never be corrected, and the player will be stuck with sub-optimal strategy.

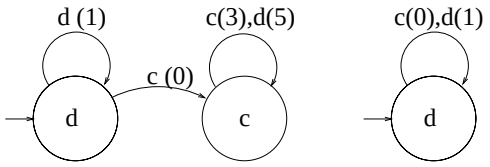

Figure 4. An example of an opponent model in local minimum. (Left): An Opponent's strategy. (Right): An opponent model. The numbers in the parentheses show the outcomes of the PD game according to the players' joint actions. The model dictates playing "all-d" while the actual best-response is "play c then all-d".

The above example demonstrates that it is better sometimes to play sub-optimally in order to explore the opponent's behavior. Action c has low utility according to the current model, but since it has not been

tried already, it is expected to strengthen the belief in the current model if the expectation will be realized, or to provide a counterexample and therefore a better model. Therefore, action c has high expected exploratory value. Action d, on the other hand, has high expected utility, but was tried many times already, and therefore has low expected exploratory value.

3.2. Uninformed Exploration Strategies

In recent years, a variety of exploration strategies have been proposed for reinforcement-learning agents (Thrun, 1992; Sutton, 1990; Moore and Atkeson, 1993; Kaelbling, 1993). In this subsection we show how to adopt these methods for model-based learning.

3.2.1. *Undirected strategies*

Undirected strategies are based on incorporating randomness into the decision procedure of the learning agent. The key idea is to associate a positive probability with any state-action pair. The agent randomly selects an action according to some given distribution and therefore no action is neglected forever.

Semi-uniform exploration (Singh *et al.*, 1998) is a simple undirected strategy that gives priority to the optimal action according to the current model but also gives positive probability to the other alternatives. If there are n alternative actions, an action r will be randomly chosen according to the following probability distribution:

$$Pr(r) = \begin{cases} 1 - \frac{n-1}{n}\alpha & \text{if } r \text{ is optimal} \\ \frac{\alpha}{n} & \text{otherwise.} \end{cases}$$

The *exploration parameter*, $0 \leq \alpha < 1$, determines the weight given to exploration. As α increases, the agent performs more sub-optimal actions and increases the chance of receiving counterexamples.

The distribution used by the semi-uniform exploration associates an equal probability for every non-optimal action without utility consideration. A utility-based exploration strategy makes choices according to a distribution determined by the expected utility of the actions. The better an action is expected to be, the more likely it is to be selected.

Boltzmann exploration (Sutton, 1990) is a utility-based strategy that reduces the tendency for exploration with time. It is based on the assumption that the current model improves as learning progresses. Boltzmann exploration assigns a positive probability to any possible action according to its expected utility and according to a parameter T called a temperature. Assume that the agent has to choose among n actions, $r_1, \ldots, r_n$, with expected utilities $U_1, \ldots, U_n$ according to

its current knowledge. The Boltzmann distribution assigns a positive probability for any possible action:

$$Pr(r_i) = \frac{e^{U_i/T}}{\sum_{j=1}^{n} e^{U_j/T}}.$$

Actions with high utility are associated with higher probability. The temperature T decreases with the stage of the game t. Therefore, as learning progresses the distribution becomes more tight on high utility actions, and the exploration tendency of the agent reduces. A common temperature function is $T = c \cdot \alpha^t$ where c is a positive constant and $\alpha < 1$ is the *exploration parameter*[2]. As alpha increases, the rate of decay of the temperature, and therefore the exploration tendency, decreases. Singh et al. (1998) show conditions on the exploration parameter that guarantee convergence of semi-uniform exploration and Boltzmann exploration.

In the model-based framework, the expected utility of an action r is computed by summing the instant expected reward of executing r and the discounted sum of rewards expected from playing the best response against the current model. Let $\langle \hat{M}_j^t, \hat{q}_j^t \rangle = L_i(h(t))$ be a regular opponent model $\hat{M}_j^t$ at state $\hat{q}_j^t$, inferred by the learning algorithm L_i given history $h(t)$. Let $s_t^* = s^{opt}(\langle \hat{M}_j^t, \hat{q}_j^t \rangle, U_i^{ds})$ be the best-response strategy against the model $\hat{M}_j^t$ at state q_j^t, according to the utility function U_i^{ds}. The *expected utility* of action r is defined to be:

$$\begin{aligned}\hat{U}_i(\langle \hat{M}_j^t, \hat{q}_j^t \rangle, r) \;=\; & u_i\left(r, \hat{M}_j^t(h(t))\right) + \\ & \gamma_i U_i^{ds}\left(s_{t+1}^*, \langle \hat{M}_j^t, \hat{q}_j^{t+1} \rangle\right)\end{aligned}$$

The first element in the sum stands for the immediate expected utility from playing r by the player, and $\hat{M}_j^t(h(t))$ by the opponent. U_i^{ds} is the expected discounted sum of rewards for player i, followed from continuing playing optimally after performing r. $\hat{q}_j^{t+1} = \delta_j^t(\hat{q}_j^t, r)$ is the new state of the opponent model following action r. s_{t+1}^* is the best response against the opponent model in its new state.

$\hat{U}_i$ can be computed efficiently since any game between two automata converges to a finite cycle, and the discounted sum of rewards of the game-path between the opponent model and the best response, can be computed by using the following procedure. Let $g_{(\langle M_i^t, q_i^{t+1} \rangle, \langle \hat{M}_j^t, \hat{q}_j^{t+1} \rangle)}$ be the game-path between the the two automata. This path induces

[2] We use the same symbol, α, to denote the exploration parameter in all the exploration strategies.

an infinite sequence of states, $\hat{P}$, in the opponent's automaton, that converges to a cycle:

$$\hat{P} = (q_0, q_1, \ldots, q_{k-1}, q_k, \ldots, q_{k+n-1}, q_k, \ldots)$$

$\hat{P}$ can be easily found by a simulation of a game played between the two automata. Let r_l be the player's action that changes the model state from q_l to q_{l+1}. The expected discounted sum of rewards for player i is:

$$U_i^{ds}\left(\langle M_i^t, q_i^{t+1}\rangle, \langle \hat{M}_j^t, \hat{q}_j^{t+1}\rangle\right) = \sum_{l=0}^{k-1} \gamma_i^l u_i(r_l, F_j(q_l)) + \\ + \frac{\gamma_i^k}{1-\gamma_i^n} \sum_{l=k}^{k+n-1} \gamma_i^{l-k} u_i(r_l, F_j(q_l)) \quad (3)$$

The first element is the discounted sum of rewards attained while following the path leading to the cycle. The second element is the geometric sum of the infinite run along the cycle.

We can now use Equation 3 to define the probability assigned by Boltzmann exploration to action r, given the current model $\langle \hat{M}_j^t, \hat{q}_j^t\rangle$:

$$Pr(r) = \frac{e^{\hat{U}_i(\langle \hat{M}_j^t, \hat{q}_j^t\rangle, r)/T}}{\sum_{r' \in R_i} e^{\hat{U}_i(\langle \hat{M}_j^t, \hat{q}_j^t\rangle, r')/T}} \quad (4)$$

3.2.2. *Directed strategies*

For undirected exploration strategies, the exploration policy of the agent is determined in advance and does not depend on the actual interactions with which the agent is involved. *Directed strategies* attempt to explore the opponent's behavior more efficiently by using statistics based on the learning experience of the agent. These methods compute an exploration bonus for each possible action and incorporate this bonus into the decision procedure of the agent. The bonus estimates the expected contribution of the action to exploration.

Several statistics were proposed for determining the exploration bonus. (Sato *et al.*, 1991) use action-counts where higher exploration bonuses are given to actions that have been chosen less frequently. Sutton (1990) suggests *Recency-based* exploration, an exploration bonus based on the time that has passed since the action was taken. (Kaelbling, 1993) suggests *Interval estimation*, an exploration bonus based on the upper bound of the confidence interval computed for the expected utility of the possible actions. (Moore and Atkeson, 1993) propose *Prioritized sweeping* exploration strategy, where states found to be

unfamiliar are connected to fictitious state with high utility. Therefore, the agent is encouraged to investigate unfamiliar areas of the environment.

Directed strategies can be incorporated directly into model-based agents by traversing the history of the game and accumulating the above statistics. For example, the history can be used to compute an age parameter for every state-action pair. Recency-based exploration can be implemented by computing an exploration bonus proportional to the age.

Let $\hat{E}_i(\hat{s}_j^t, r)$ be the exploration bonus computed for an action r, according to the current model of the opponent's strategy, $\hat{s}_j^t$, and the history $h(t)$. The value for each action is computed by a linear combination of the expected utility of the given action, $\hat{U}_i$, and the exploration bonus, $\hat{E}_i$:

$$V_i(\hat{s}_j^t, r) = (1-\alpha)\frac{\hat{U}_i(\hat{s}_j^t, r)}{\sum_{r' \in R_i} \hat{U}_i(\hat{s}_j^t, r')} + \alpha \frac{\hat{E}_i(\hat{s}_j^t, r)}{\sum_{r' \in R_i} \hat{E}_i(\hat{s}_j^t, r')} \qquad (5)$$

As for undirected methods, α is the exploration parameter that determines the ratio between exploration and exploitation. Thrun (1992) deals with the question of combining exploration and exploitation. He shows that dynamic combination, e.g., changing the tradeoff parameter α, during learning, might have an advantage over static combination.

To summarize, at any stage t of the game, an exploring MB-agent updates the opponent model $\hat{s}_j^t = L_i(h(t))$. It then find the best response against the model $s_t^* = s^{opt}(\hat{s}_j^t, U_i)$. When using an *undirected exploration method* it computes $Pr(r)$ for every action $r \in R_i$ and randomly selects an action according to this distribution. When using a *directed method* it computes the exploration bonus for every action r based on the history statistics, and then selects the action with the maximal combined utility.

4. Lookahead-based Exploration

The exploration methods described in the previous section were developed to handle the problem of being stuck in local-minima. An exploring agent is willing to perform sub-optimal actions in order to acquire a better model of the opponent yielding a better utility in the long run. It is quite possible, however, that the exploratory action will indeed yield a better model, but will also lead the agent to a poor state where even optimal play yields low utility. Sometimes, knowledge is too expensive

– taking a step into the dark can lead you to the sought treasure, or into a deep chasm. While falling into the chasm, the improved model of the world is not likely to help you much.

For example, the left part of Figure 5 describes a strategy for IPD called *Grim*. A Grim player cooperates as long as its opponent cooperates, but never forgives defection – after one single defection of its opponent it reacts by repeatedly defecting. Assume that an exploring agent holds the model described in the right part of the figure. Defection has lower utility than cooperation according to the model, therefore, a non-exploring agent will always cooperate yielding utility of $\frac{3}{1-\gamma}$. An exploring agent will eventually try d and acquire a perfect model of Grim. However, the exploring action d takes the agent into the "defection sink" without any opportunity to get out, yielding utility of $\frac{5-4\gamma}{1-\gamma}$, which is lower than the utility of cooperation for any $\gamma \geq \frac{1}{2}$. Thus, exploration indeed yielded better knowledge, but the cost of acquiring this knowledge is too high.

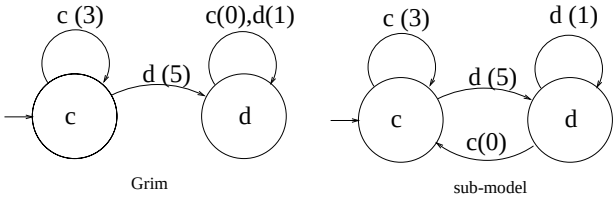

Figure 5. Left: The *Grim* strategy for IPD. Right: the current model held by the agent. An exploratory action against grim (d), will be followed by falling into the "defection sink" without any opportunity to get out.

The above example clearly demonstrates that exploratory behavior requires a better mechanism for predicting the risk involved in taking sub-optimal actions. One of the problems with the exploration mechanisms described in Section 3 is the use of a utility function that assumes a stationary opponent model throughout the expected course of the game. This assumption is not rational for a learning agent who continuously modifies its opponent model. In order to develop a risk-sensitive exploration strategy we need a mechanism that allow the agent to take into consideration the expected revision of its belief while computing the expected utility. Such a mechanism requires a method for representing the agent's uncertainty regarding its opponent's strategy.

In the rest of this section we describe a risk-sensitive exploration methodology. We first describe *mixed strategies* for representing uncertain opponent models, and an *almost best response* procedure against such strategies. We then show a learning mechanism for acquiring uncertain models.

4.1. Mixed strategies

The model-based agent described in Section 2 maintains a regular model of the opponent's behavior. The regular model is incapable of representing the uncertainty of the learning agent regarding the model's prediction. To represent such uncertainty we apply a stronger mechanism for modeling the opponent strategy: The game-theoretic concept of *mixed strategy*.

Definition 2. A *mixed strategy* for repeated game $G^{\#}$ is a pair, $(\overline{S}, \overline{\pi})$ where $\overline{S} = \{s_1, \ldots, s_k\}$ is a finite set of (pure) strategies for $G^{\#}$ called the *set of support* (SOS) and $\overline{\pi} = (\pi_1 \ldots \pi_k)$ is a probability distribution over SOS called the *belief distribution*, where $\pi_l \geq 0$ for $1 \leq l \leq k$, and $\sum_{l=1}^{k} \pi_l = 1$. A *regular mixed strategy*, $(\langle \overline{M}, \overline{q} \rangle, \overline{\pi})$, is a mixed strategy whose set of support includes k automata, $\overline{M} = (M_1, \ldots, M_k)$, in states $\overline{q} = (q_1, \ldots, q_k)$.

An agent uses a mixed strategy $(\overline{S}, \overline{\pi})$, by randomly drawing a strategy s_l from the set $\overline{S}$, according to the belief distribution $\overline{\pi}$, and performing action $s_l(h(t))$.

Mixed strategies can be used as opponent models in two different ways. In the first one, we assume that the opponent applies a mixed strategy, i.e., it randomly chooses one of the strategies from the set of support at any stage of the game. Gilboa (1988) shows how to play against n regular opponents by solving the best response problem against the product automaton of the opponents' DFA. A similar method can be used to find the best response against a regular mixed strategy. First, construct the product automaton of the automata belonging to the set of support, with an output function which is probabilistic according to the given distribution. Second, find the best response automaton against the probabilistic product automaton. The problem of finding the best response against a *probabilistic action automaton* (PAA) is equivalent to the best response problem against a deterministic automaton (Freund *et al.*, 1995).

The above interpretation still assumes that the agent does not modify its opponent model throughout the game-path while computing the utility, and is therefore not suitable for the belief revision framework. The second interpretation considers the opponent's strategy to be one of the models belonging to the set of support, and the belief distribution to reflect the subjective beliefs of the player on the alternative opponent models. This interpretation is suitable for the learning framework studied in this work and can be viewed as a form of *Bayesian adaptive control*. By choosing a specific action, the agent forces the opponent to

respond. The opponent's response enables a revision of the belief distribution over the set of the opponent models. Following the opponent's response, r_j^t, the beliefs on all models that do not expect to perform r_j^t should be reduced to zero. The beliefs on all models that expect to output r_j^t should be increased.

Let $(\overline{S}, \overline{\pi})$ be the current mixed strategy, and let $Pr(r_j) = \sum_{l=1}^{k}[\pi_l | s_l(h(t)) = r_j]$ be the probability of performing action r_j by the opponent at the current stage of the game. The posterior belief, $\overline{\pi}^{r_j}$, can be computed directly by Bayes law for every $1 \leq l \leq k$:

$$\pi_l^{r_j} = \begin{cases} 0 & s_l(h(t)) \neq r_j \\ \dfrac{\pi_l}{Pr(r_j)} & \text{otherwise} \end{cases} \tag{6}$$

Two useful propositions are implicit in the Bayesian belief-revision process:

- The confidence of the agent in the opponent's models is reflected by its prior belief distribution. The more the agent is confident in one of its models, that is, the belief distribution is more tight on one of the models in SOS, the more the posterior distribution will resemble the prior distribution for any given response. This is clear in the limit. An agent with absolute prior confidence who associates a probability of 1.0 with one of the models, e.g. S_1, can learn nothing from new evidence. From equation 6, if the prior probabilities π_l, are all zero for any $S_l \neq S_1$, the corresponding post-prior probabilities will also be zero.

- The more "surprising" the opponent's response is, the bigger the impact upon the post-prior belief. In terms of Equation 6, a surprising response would be associated with low probability $Pr(r_j)$. The smaller this probability is, the more the post-prior belief diverges from the prior belief.

We can now define the utility of the expected game-path between a learning strategy and a mixed model, while considering the modified beliefs throughout the path:

$$\hat{U}_i^{ds}(s_i, (\overline{S}, \overline{\pi})) = \sum_{r_j \in R_j} Pr(r_j)[u_i(s_i(h(t)), r_j) + \gamma_i \hat{U}_i^{ds}(s_i, (\overline{S}, \overline{\pi}^{r_j}))]$$

Ben-Porath (1990) shows that under this interpretation, the best response problem against a regular mixed strategy is NP-complete. He also shows that the problem becomes polynomial in the size of the product automaton when fixing the size of the set of support. However,

the best response procedure described by Ben-Porath is impractical since the product automaton is extremely large even for a small set of support. In the next subsection we describe an alternative approach that is more suitable for a computational bounded agent.

4.2. An almost-best response strategy against a regular mixed model

In this subsection we develop an algorithm that returns an almost-best response automaton against a regular mixed model. The level of approximation can be determined in advance at any stage of the game, according to the computational resources available for the agent. The strategy returned by the algorithm enables the agent to efficiently balance between exploration and exploitation throughout the game-path expected to be played.

Given an opponent's strategy s_j, and an approximation parameter, ϵ, an ϵ-best response strategy, with respect to s_j, guarantees the player a utility which is not far more than ϵ from the maximal possible utility against s_j (Kalai, 1990).

Definition 3. A strategy $s_\epsilon^{opt}(s_j, U_i)$ will be called *ϵ-best response* for player i with respect to strategy s_j and utility function U_i, iff $\forall s \in \mathcal{S}_i$, $[U_i(s_\epsilon^{opt}(s_j, U_i), s_j) \geq U_i(s, s_j) - \epsilon]$.

Let u_{max} be the maximal value of the stage game G, and $U_{max} = \frac{u_{max}}{1-\gamma}$ be the maximal utility in the repeated game $G^{\#}$. The algorithm receives a regular mixed model, $(\langle \overline{M}, \overline{q} \rangle, \overline{\pi})$, and a depth parameter d that is determined by the approximation parameter ϵ, $d \geq \frac{\ln(\frac{\epsilon}{U_{max}})}{\ln(\gamma)}$. The set of support includes k automata, $M_l = (Q_l, R_i, q_l^0, \delta_l, R_j, F_l), l = 1, \ldots k$, in states $\overline{q} = (q_1 \ldots q_k)$, and a belief distribution $\overline{\pi} = (\pi_1 \ldots, \pi_k)$, $\pi_l > 0$ [3]. The algorithm returns a pair of an ϵ-best response automaton and the expected utility of the game against the given mixed model.

The algorithm classifies the states of the mixed strategy to *certain states* where all the models predict the same output, and *uncertain states* where at least two models disagree on the opponent's current action. For *uncertain states*, the algorithm splits the models of SOS into disjoint sets according to their expected output, $\overline{M}^{r_j} = \{M_l \in \overline{M} | F_l(q_l) = r_j\}$. Any opponent's action reduces the number of models in the corresponding set of support since the beliefs in all models that are not consistent with this action are reduced to zero. For each pair (r_i, r_j)

[3] We ignore models with a belief of zero.

of a player's action and an opponent's action, we construct a posterior mixed strategy, $(\langle \overline{M}^{r_j}, \overline{q}^{r_i} \rangle, \overline{\pi}^{r_j})$. The posterior belief, $\overline{\pi}^{r_j}$, over $\overline{M}^{r_j}$, can be computed using Equation 6. The player's action, r_i, determines the new state of the posterior mixed model, $q_l^{r_i} = \delta_l(q_l, r_i), 1 \leq l \leq k$.

The algorithm calls itself recursively with a reduced depth limit. The recursion stops when the search reaches the depth limit, or, when the set of support includes only one model, for which the problem is reduced to the best response problem against a single automaton.

Certain states do not reveal information to the player, since all the models predict the same output. Therefore, the set of support and the belief distribution cannot be modified. For such states, the models' current states are modified according to the player's action and the best response is found recursively with a reduced depth limit.

Finally, when the recursion terminates at depth d, the algorithm returns a constant policy – a one-state automaton, A^0, that outputs one of the player's actions for any possible history.

Denote the automaton returned by the recursive call for the pair (r_i, r_j) by A^{r_i,r_j}, and the expected payoff of the game to be played by U^{r_i,r_j}. Let r_i^* be the action that maximizes the expected utility at the given state of the mixed model:

$$r_i^* = arg\max_{r_i} \sum_{r_j \in R_j} Pr(r_j)[u_i(r_i, r_j) + \gamma U^{r_i,r_j}]$$

The ϵ-best response automaton begins with r_i^* at the initial state and then plays $A^{r_i^*,r_j}$, according to the opponent's action r_j.

Note that the automaton described above provides a plan of d steps for the player for exploring and exploiting the mixed model. The plan optimizes the player's behavior by considering the expected utility, and also the expected revealed information throughout the alternative game-paths. This plan is in contrast to the infinite plan returned by the best response procedure against a *single automaton*, which does not direct the agent's behavior when the opponent does not play as predicted. Figure 6 describes a pseudo-code of the algorithm.

Figure 7 demonstrates how the ϵ-best response strategy can avoid falling into the defection sink when playing the IPD against Grim. The top part of Figure 7 shows two models held by the exploring agent, after t mutual cooperations in the IPD game. M_0 is TFT which predicts the opponent to defect after defection but assumes a possible withdrawal by cooperation. M_1 is the Grim strategy. The bottom part of the figure shows the computation of the ϵ-best response. The player's actions are marked by solid lines and the opponent's actions are marked by dashed lines. To save space, duplicated subtrees are drawn only once. Action c has no exploration benefit since all the models predict cooperation of

Procedure $\epsilon\text{BR}\big((\langle\overline{M},\overline{q}\rangle,\overline{\pi}),d\big)$
  $k \leftarrow |\overline{M}|$
  if $(d=0)$ or $(k=0)$ **then return** $\langle A^0, 0\rangle$
  if $(k=1)$ **then return** $\text{PureBR}(\langle M, q\rangle)$
  else
    if uncertain$(\overline{q})$
      for each $r_j \in R_j$
        $Pr(r_j) \leftarrow \sum_{l=1}^{k}[\pi_l | F_l(q_l) = r_j]$
        $\overline{M}^{r_j} \leftarrow \{M_l \in \overline{M} | F_l(q_l) = r_j\}$
        for $l = 1 \ldots, k$
          if $F_l(q_l) = r_j$ **then** $\pi_l^{r_j} \leftarrow \frac{\pi_l}{Pr(r_j)}$
          else $\pi_l^{r_j} \leftarrow 0$
        for each $r_i \in R_i$
          $q_l^{r_i} \leftarrow \delta_l(q_l, r_i), 1 \le l \le k$
          $\langle A^{r_i,r_j}, U^{r_i,r_j}\rangle \leftarrow \epsilon\text{BR}\Big((\langle\overline{M}^{r_j}, \overline{q}^{r_i}\rangle, \overline{\pi}^{r_j}), d-1\Big)$
        $U \leftarrow \max_{r_i} \sum_{r_j} Pr(r_j)[u_i(r_i, r_j) + \gamma U^{r_i,r_j}]$
        $r_i^* \leftarrow arg\max_{r_i} \sum_{r_j} Pr(r_j)[u_i(r_i, r_j) + \gamma U^{r_i,r_j}]$
        $A \leftarrow$ (a DFA that begins with r_i^* and plays $A^{r_i^*,r_j}$ according to r_j)
    else /* $\overline{q}$ is certain, i.e., all models predict the same action r_j^* */
      for each $r_i \in R_i$
        $q_l^{r_i} \leftarrow \delta_l(q_l, r_i), 1 \le l \le k$
        $\langle A^{r_i,r_j^*}, U^{r_i,r_j^*}\rangle \leftarrow \epsilon\text{BR}\big((\langle\overline{M}, \overline{q}^{r_i}\rangle, \overline{\pi}), d-1\big)$
      $U \leftarrow \max_{r_i}[u_i(r_i, r_j^*) + \gamma U^{r_i,r_j^*}]$
      $r_i^* \leftarrow arg\max_{r_i}[u_i(r_i, r_j^*) + \gamma U^{r_i,r_j^*}]$
      $A \leftarrow$ (a DFA that begins with r_i^* and plays $A^{r_i^*,r_j^*}$ according to r_j^*)
    return (A, U)

Figure 6. The ϵBR algorithm that returns an ϵ-best response automaton against a regular mixed model.

the opponent after c. The utility of cooperation is $U(c) = 3 + 5\gamma + \frac{2\gamma^3}{1-\gamma}$. The utility of defection is $U(d) = 5 + \frac{2\gamma^2}{1-\gamma}$. Hence, c is prefered over d for any $\gamma > 0.5$. Note that the exploration strategy returned by the algorithm is "c then all-d". When the algorithm searches deeper, the returned strategy will postpone defection for later stages of the game-path.

The following theorem proves that algorithm ϵ-BR() returns an ϵ-best response strategy against the given mixed strategy, with respect to $\hat{U}^{ds}$.

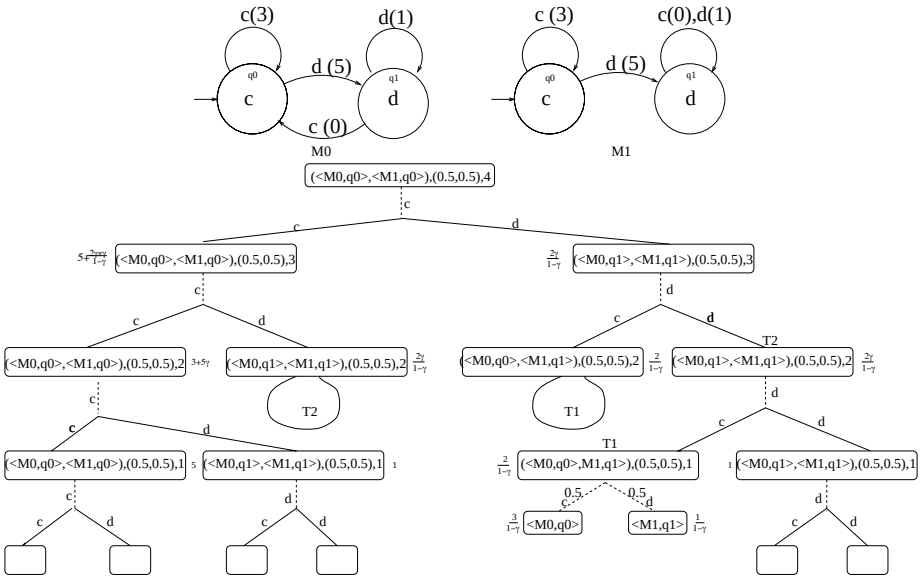

Figure 7. Top: The 2 models belong to the mixed model after t mutual cooperations in the IPD game. Bottom: The search tree spanned by ϵ-BR. The player's actions are marked by solid lines. The opponent's actions are marked by dashed lines. To save space, duplicated subtrees are drawn only once. Action c is preferred over d for any $\gamma > 0.5$.

Theorem 1. Let ϵ be a positive number. Let $\epsilon' = \frac{\epsilon}{|U_{max}|}$, and let $d \geq \frac{ln(\epsilon')}{ln(\gamma)}$. Let $(\langle \overline{M}, \overline{q} \rangle, \overline{\pi})$ be a regular mixed strategy. The algorithm $\epsilon\text{BR}((\langle \overline{M}, \overline{q} \rangle, \overline{\pi}), d)$, returns an ϵ-best response automaton, against $(\langle \overline{M}, \overline{q} \rangle, \overline{\pi})$, with respect to $\hat{U}_i^{ds}$. The computation time is polynomial in $\frac{1}{\epsilon'}$, the number of models in SOS, and the size of the maximal automaton belonging to the set of support.

Proof. The proof is by induction on the depth d. For $d = 0$ the proof is trivial since $\epsilon \geq U_{max}$. Hence, any strategy is ϵ-best response. For the induction step, if the number of models is $k = 1$, then the algorithm returns the best response against a single DFA which is also an ϵ-best response. For $k > 1$, assume correctness for $d-1$. Let s be the strategy returned by the algorithm, and s' be an arbitrary strategy. We denote the expected utility of s returned by a search to depth d, by U_d, and the expected utility of s' by U'_d. We also denote the corresponding approximation parameter ϵ according to the given depth d, by ϵ_d. Note that from the relation between ϵ and d, it follows that $\gamma\epsilon_{d-1} = \epsilon_d$. If the current state is certain, i.e., all models predict the opponent to perform

r_j^*. The utility returned by the algorithm is:

$$U_d = \max_{r_i}[u_i(r_i, r_j^*) + \gamma_i U_{d-1}] \overset{induction}{\geq} \max_{r_i}[u_i(r_i, r_j^*) + \gamma_i(U'_{d-1} - \epsilon_{d-1})] =$$
$$= \max_{r_i}[u_i(r_i, r_j^*) + \gamma_i U'_{d-1}] - \epsilon_d \geq U'_d - \epsilon_d$$

A Similar computation can show the claim for uncertain states.

For computing the complexity of the algorithm, let us look at the tree it traverses. At each inner node of the tree, we update the belief distributions; each such computation depends on N, the size of SOS. For leaf nodes, each computation is polynomial in the size of the maximal automaton, M_{max}. Note that the maximal number of computations is bounded by the number of leaves in the search tree, $(|R_i||R_j|)^d = O(\text{poly}(\frac{1}{\epsilon'}))$, where poly() is a polynomial function. Therefore, the total complexity is $O(\text{poly}(\frac{1}{\epsilon'}) * (M_{max} + N))$. □

4.3. Learning a mixed model

Describing the opponent model by a mixed strategy requires a learning procedure for acquiring alternative models for the set of support, SOS, and a computational procedure for determining the belief distribution over SOS. In this subsection we describe a lookahead learning procedure which expands the tree of all possible future paths of the game.

4.3.1. *Lookahead-based learning of mixed models*

Ideally, the set of support should include all the models consistent with the given history, but such a set of strategies is infinite.

For restricting the set of support we concentrate on a subset of models consistent with the given history that differ in predicting the expected opponent's action at the next stage of the game. We use the learning algorithm of the model-based strategy, L_i, for constructing these models. By concatenating all the possible pairs of actions, (r_i, r_j), to the given history, and by applying the learning algorithm L_i, to the new expanded histories, we acquire models that are consistent with the given history and predict the opponent's next action to be r_j at stage $t+1$ of the game. The player action at stage $t+1$, r_i, does not affect the learning algorithm since the opponent's action at stage $t+1$ is only affected by the previous player's actions. Hence, we can acquire at most $|R_j|$ different models for the set of support.

Definition 4. Given a history $h(t)$, and a learning algorithm L_i. *The set of support*, $SOS^1(L_i, h(t))$, is defined to be the set of models

returned by the algorithm L_i, applied to the history $h(t)$ concatenated with one more joint action (r_i, r_j):

$$SOS^1(L_i, h(t)) = \{M | M = L_i\left(h(t) \circ (r_i, r_j)\right), r_i \in R_i, r_j \in R_j\}$$

After acquiring the different models for the set of support, the agent should determine its belief distribution over the set. Since all models are consistent with the given history, each one can serve as an opponent model. However, there are models that seem to be more reasonable than others. The agent can choose any way for computing its subjective beliefs. The only restriction is that its beliefs should be non-negative and should sum to one.

A reasonable heuristic for belief assignment can be implemented as follows: According to the Occam razor principle, the belief probability for a given model should decrease in its size, i.e. the smaller the model, the larger is its associated belief. The beliefs should also be in direct relation to the coverage of the models: the smaller the number of edges of the model that have been tried by the agent during the given history, the smaller the agent's belief in the given model.

$$\pi_j \propto \frac{cover(h(t), M_j)}{|M_j|}. \tag{7}$$

Figure 8 shows an example for a mixed strategy acquired after a history of t mutual defections in the IPD game. M_0 is a one state model that predicts consecutive defections by the opponent. M_1 is a $t+1$ states model that predicts consecutive cooperations by the opponent after the t defections. Note that the belief in M_1 is much smaller than the belief in M_0. For large t, it is not reasonable to predict cooperation in the future after t consecutive defections.

Modeling the opponent's strategy by a mixed strategy enables the agent to quantify its uncertainty about the opponent's behavior and to make better decisions under uncertainty. The strategies acquired for the set of support were inferred by predicting different responses of the opponent to the given history. For testing the influence of the agent's behavior on the opponent's behavior, we should search some stages deeper, testing the effects of the agent's actions on the opponent's expected responses.

When we search d stages forward, we shall look at all possible expanded histories of length $t + d$, $h(t) \circ (r_i^t, r_j^t)^d$. By concatenating all the possible d pairs of actions, (r_i, r_j), to the given history, and by applying the learning algorithm to the expanded histories, we acquire models that are consistent with the history $h(t)$, and predict differently the opponent responses for the player sequences of actions. Note that

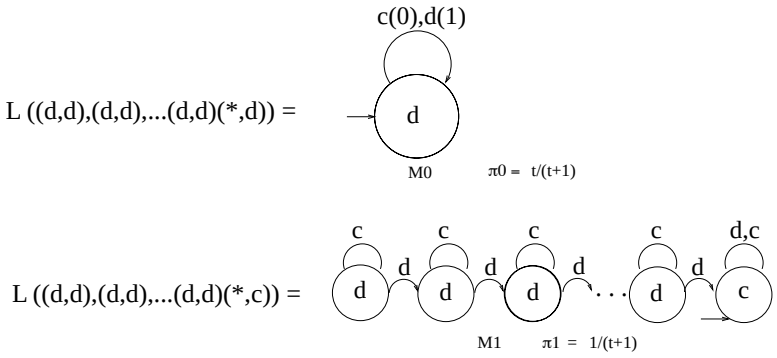

Figure 8. A mixed strategy acquired after t mutual defection in the IPD game. M_0 predicts future defection by the opponent. M_1 predicts cooperation. The beliefs are computed in relation to the model's size and the model's coverage according the game history.

the player action at stage $t + d$ does not affect the learning algorithm since the opponent's action at stage $t+d$ is only affected by the previous player's actions. Hence, we can acquire at most $|R_i|^{d-1}|R_j|^d$ different models for the set of support.

Definition 5. Given a history $h(t)$, and a learning algorithm L_i, *the set of support*, $SOS^d(L_i, h(t))$, is defined to be the set of the models returned by the algorithm L_i, applied to the history $h(t)$, concatenated with d more joint actions, $(r_i, r_j)^d$.

$$SOS^d(L_i, h(t)) = \left\{ M | M = L_i \left(h(t) \circ (r_i, r_j)^d \right), r_i \in R_i, r_j \in R_j \right\}$$

To summarize, for exploring the opponent's strategy using a mixed model, the agent first searches d stages forward for collecting different opponent models to the set of support. It then infers a belief distribution over this set. Following that, it finds the ϵ-best response against the mixed model and performs a sequence of actions dictated by this strategy. By doing so, the agent rationally balances between exploration and exploitation, and reduces the risk involved in performing sub-optimal actions. This may carry additional computational cost, but the investment may be profitable when the risk involved in exploration is high. Subsection 4.3.3 analyzes the complexity of the algorithm.

4.3.2. *An example*

To illustrate the process of learning a mixed model devising best-response against it using ϵBR, we will show how our algorithm behaves when facing Grim. Playing against Grim strategy is a great challenge for any exploration method since there is no possibility to regret after

performing an exploration action. Model-based strategies that invoke exploration by incorporating randomness into their decision procedure, will fail against Grim when the exploration action d will be eventually chosen.

Assume that the current history consists of several mutual cooperations with Grim. The key to detecting Grim is to observe that the opponent does not cooperate after the player defects then cooperates. Since the next opponent action is a response to the player's previous action, a lookahead depth of at least 3 is needed.

A 3-lookahead strategy will produce 32 extensions of the current history. One of the extensions is $((d,c),(c,d),(*,d))$ which will lead to inferring Grim. The other 31 extensions will yield several alternative models which will be included in the set of support. Most of those models, however, will be larger than Grim. Therefore, after updating the belief distribution according to the models' size and coverage, most of the members in the set of support will have very low associated belief, leaving only four models. The set of four models with their associated beliefs is shown in Figure 9.

The ϵBR procedure will detect that playing the exploratory action d will indeed improve the knowledge about the opponent since response of the opponent to d will decide between All-c and the three other models. It will also, however, detect the low expected utility of playing d due to the significant belief in Grim and will therefore prefer to play c.

The ability of the algorithm to avoid falling into the defection sink of Grim depends on the history of mutual cooperations being long enough to eliminate alternative models, leading to significant belief in Grim. A shorter history may be sufficient when the punishment of the defection increases.

4.3.3. *Complexity analysis*

After each interaction the lookahead-based exploration strategy first extends the history with all possible paths of length d_l, applies the learning algorithm to each of the extensions, and accumulates the resulting models into the set of support. It then computes the belief distribution over SOS and calls the ϵBR procedure to depth d_ϵ.

The complexity of the learning process depends on the lookahead horizon d_l, on the complexity of the learning algorithm $Cost(L)$, the history length $|h|$ and on the "branching factor" of the lookahead tree $B = |R_i||R_j|$. In each leaf of the expanded tree, we apply the learning algorithm L to the extended history. The size of the resulting SOS is bounded by the number of leaves, which is $|R_i|^{d_l-1}|R_j|^{d_l} \leq B^{d_l}$. For each of the models in SOS we compute the belief using Equation 7

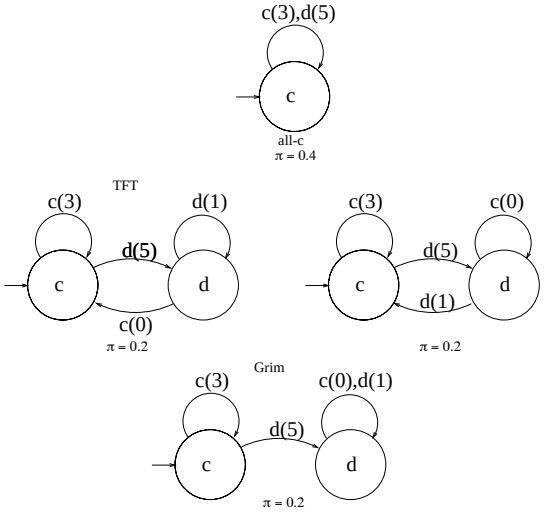

Figure 9. Left: The models acquired by the 3-lookahead exploration strategy after 10 mutual cooperations with Grim while playing the IPD game. All other possible models are neglectable since the beliefs in them are lower than a given threshold (0.1). The ϵBR algorithm will decide to cooperate for any search depth greater than 5 and any γ greater than 0.5.

whose complexity determined by $|h|$. Therefore, the cost of the learning process, C_l, is bounded by

$$B^{d_l}(Cost(L) + |h|) \tag{8}$$

In the previous section we showed that the cost of ϵBR, C_b, is $B^{d_\epsilon}(M_{max} + |SOS|))$ where M_{max} is the size of the largest automaton in SOS. Since $|SOS| \leq B^{d_l}$ and $M_{max} \leq |h|$ we get the following bound on the cost of the lookahead-based exploration, C_e:

$$C_e = C_l + C_b \leq B^{d_l}(Cost(L) + |h|) + B^{d_\epsilon}(|h| + B^{d_l}) \tag{9}$$

It is obvious that to keep this computation feasible, d_l and d_ϵ should be small. With small depth values, the dominant cost is the call to the learning algorithm L which computes the models at the leaves of the lookahead tree. The complexity of L usually depends on the length of the history. For example, the complexity of the US-L^* algorithm used for the experiments described in the following section is $O(|h|^3)$ (Carmel and Markovitch, 1998). One way to reduce this cost is by using windowing to decrease $|h|$. Another way is to make the whole exploration process incremental. Instead of learning the whole set of support after each interaction, we can maintain the set between interactions, eliminating inconsistent models. Then, we will need to call L only at leaves whose associated path contradicts all existing models.

The complexity of the uninformed exploration methods, described in Section 3.2, is much lower since they ignore the effect of their actions on the learning process. For example, the complexity of Boltzmann exploration is bounded by $Cost(L) + |R_i||h|$. The higher cost of lookahead-based exploration is outweighed by the benefit of obtaining more informative examples, which hopefully leads to lower number of examples required for adaptation.

5. Experimentation: Exploration strategies in repeated-games

We conducted a set of experiments to test the capabilities of the different exploration methods for repeated games.

5.1. Experimentation methodology

The experiments described in this section compare various exploration strategies and test their characteristics. Each experiment consists of 100 repeated IPD games of length 400 between the tested agent and randomly-generated *deterministic* opponents. The method of randomly generating many strategies simulates a MAS where an agent can benefit from adapting to many different opponents. 100 opponent automata were generated by choosing a random transition function and a random output function. The randomly-generated machines were minimized by taking out all unreachable states and by using the DFA-minimization algorithm (Hopcroft and Ullman, 1979).

The agent's performance was measured using the following dependent variables:

1. *Relative utility:* Since the actual opponent strategy $\langle M_j, q_j^t \rangle$ is known to the experimenter, the expected utility of a model $\langle \hat{M}_j^t, \hat{q}_j^t \rangle$ can be computed by the infinite discounted sum of rewards of the game between the opponent automaton and the best response against the given model:

$$U_{exp}(\langle \hat{M}_j^t, \hat{q}_j^t \rangle) = U_i^{ds}\left(s^{opt}(\langle \hat{M}_j^t, \hat{q}_j^t \rangle), \langle M_j, q_j^t \rangle\right)$$

   The computation of U_{exp} can be done efficiently according to Equation 3. Instead of using absolute utilities, we use *relative utility*, which is the ratio between the expected utility of the current model, $\langle \hat{M}_j^t, \hat{q}_j^t \rangle$, and the expected utility of the best possible model –

the actual opponent DFA:

$$U_r(\langle \hat{M}_j^t, \hat{q}_j^t \rangle) = \frac{U_{exp}(\langle \hat{M}_j^t, \hat{q}_j^t \rangle)}{U_{exp}(\langle M_j, q_j^t \rangle)}$$

2. *Average cumulative reward:* The cumulative reward of the game divided by the number of stage games:

$$\frac{\sum_{k=0}^{t-1} u_i(r_i^k, r_j^k)}{t}$$

3. *Model size:* The number of model states.

We have experimented with the following agents:

1. *TFT:* The stationary strategy Tit-for-tat.

2. *Q-strategy:* Q-learning (Watkins and Dayan, 1992) is a well-known RL algorithm that works by estimating the values of all state-action pairs. For repeated games, an entire history is needed for representing a game state. In such a framework, any state is visited only once and no generalization can be made. A possible alternative is to use a fixed window of previous moves for representing a state. A too wide window can make the agent using a too sparse table and to disable convergence of the learning process in practical time. A too narrow window can cause perceptual aliasing, i.e., different states can appear identical and therefore can be represented by the same state.

   Q-learning was tried in repeated games against TFT by Sandholm and Crites (1995). The Q-agent succeeded in learning an optimal strategy against TFT using a window of width one, but it needed about 100,000 iterations for convergence. Similar results were obtained by us. In our experiments the Q-agent used a window of width 2, Boltzmann exploration using a temperature function $T = 5 * \alpha^t$, and a learning parameter $\beta_t = 0.1 * 0.999^t$.

3. *MB-strategy:* The non-exploring model-based strategy described in Section 2. The MB-agent begins with a random DFA as a model of its opponent's strategy and modifies the model whenever it fails to predict the opponent's actual play. It computes the best response strategy against the current model and acts according to it.

4. *Boltzmann exploration:* An undirected MB-agent with incorporated Boltzmann exploration using a temperature function $T = 5 * \alpha^t$.

5. *Recency-based exploration:* A directed MB-agent using *recency-based* exploration (Sutton, 1990). For every action r, it counts the number of stages that have passed since the last time that r was taken, $\rho(\hat{s}_j^t, r)$, where $\hat{s}_j^t$ is the current opponent model. It then computes an exploration bonus $E'(\hat{s}_j^t, r) = \sqrt{\rho(\hat{s}_j^t, r)}$ and measures the utility of action r according to Equation 5.

6. *Combined exploration:* A combined strategy that uses a combination of the directed and undirected strategies: it uses *recency-based* exploration strategy but decreases the exploration parameter with time, $\alpha_t = 0.1 * 0.99^t$.

7. *Lookahead-based exploration:* The ϵ-BR algorithm with a depth-limit of 2 combined with 2-lookahead learning algorithm.

In addition, we have used as independent variables the exploration parameter, α, the discount parameter, γ, and the opponent automaton size D. Unless otherwise specified, the default values used for this parameters are: $\gamma = 0.9$, and $D = 20$.

5.2. The effect of the exploration parameter on the agent's performance

In the first experiment we tested the effect of the exploration parameter on the performance of an MB-agent using Boltzmann exploration. We used exploration parameter (α) values of 0, 0.9 and 0.99. For comparison we also tested the fixed TFT strategy and the Q-learning strategy.

Table I. The average cumulative reward, relative utility and model-size, attained by the agents after 400 stages of the PD game. The results are averaged over the 100 trials with 100 randomly-generated opponents. The standard deviation is given in parentheses.

	Av. Cumulative Reward		Relative Utility	Model size
	Agent	Opponent		
TFT	2.24 (0.05)	2.24 (0.05)	–	–
Q-agent	2.34 (0.28)	2.18 (0.3)	–	–
MB-Agent ($\alpha = 0$)	3.28 (0.33)	0.72 (0.06)	0.86 (0.95)	12.6 (18.87)
MB-Agent($\alpha = 0.9$)	3.50 (0.36)	1.11 (0.1)	0.91 (0.53)	84.87 (123.1)
MB-agent($\alpha = 0.99$)	3.70 (0.32)	1.01 (0.1)	0.95 (0.35)	85.07 (171.3)

The average cumulative reward, relative utility and model-size, attained by the agents after 400 stages of the PD game are shown in

Table I. The adaptive players achieved significantly better results than the non-adaptive TFT player. TFT gained only 2.24 points which is the average payoff of the PD game (its randomly-generated opponents achieved similar results). The second row shows the results of the Q-agent. The Q-agent fails to learn a reasonable strategy using the given resources. Increasing the width of the window and changing the learning parameters did not help. The Q-agent managed to achieve results comparable to those achieved by the MB-agent in 400 stages only when the length of the game was increased up to 40,000. Therefore, while the Q-agent can achieve results similar to those of the MB-agent, it requires significantly more resources.

The results also show the benefit of exploration. The exploring model-based agents are far more successful than the non-exploring agent (with $\alpha = 0$) and the benefit is increased when increasing the exploration tendency (with larger α). The non-exploring agent achieves 3.28 points while the exploring agent (with $\alpha = 0.99$) achieves 3.7 points. This is close to the optimal performance of 4.2 which was measured by playing the best response strategies against the randomly-generated opponents.

The results also indicate that the non-exploring agent infers much smaller models than those inferred by the exploring agent. The reason for that is that the non-exploring agent is trapped in local minima with sub-models that are smaller but less utile.

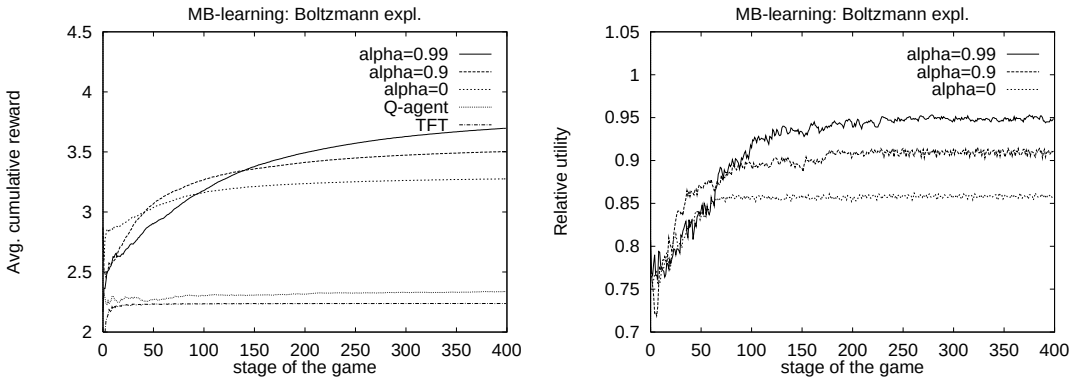

Figure 10. Left: The average cumulative reward attained by different playing strategies against 100 randomly-generated automata of size 20. Right: The average relative utility of the models acquired by the model-based agents during the game.

Figure 10 shows the cumulative reward and the relative utility of the models attained during the 400 stages of the game. The graphs highlight an interesting phenomenon. At early stages of the game, the exploring

agents pay for the suboptimal decisions induced by their "curiosity". The cost for exploration increases with the exploration parameter α. However, the better models generated by the exploring agents pay off in later stages of the game, and the exploring strategies outperform the non-exploring strategy dominantly.

5.3. The effect of the discount parameter on the agent's performance

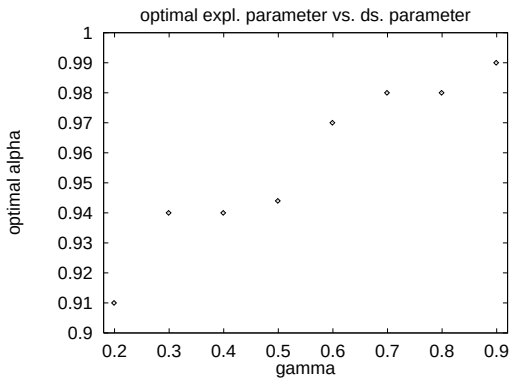

Figure 11. The best exploration parameter for Boltzmann strategy with various discount parameters .

The previous results address a question of how much exploration is needed during interaction. The answer depends on the "greed" of the learning agent. The more weight the agent gives to future payoffs, the more resources it should spend on exploration. This weight is expressed by the discount parameter γ. We repeated the last experiment for various values of γ, used by an MB-agent with Boltzmann exploration. For each γ we tried several values of exploration parameter α and recorded the one for which the agent achieved the best performance. Figure 11 shows the best α for each γ. As expected, as γ increases, it is better to invest more on exploration by using larger α.

5.4. The effect of the opponent's size on the agent's performance

Figure 12 shows the effect of increasing the complexity of the opponent strategy on the learning process. The three curves show the average results attained by an MB-agent with Boltzmann exploration with $\alpha = 0.1 * 0.99^t$ against 100 randomly-generated automata of sizes 20, 40, and 60. The results show that increasing the size of the automata

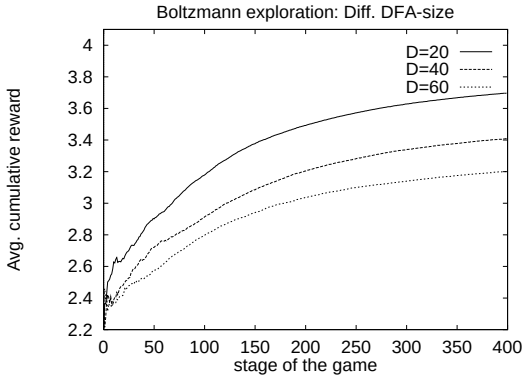

Figure 12. Model-based learning of randomly-generated automata with various sizes.

slows down the rate of adaptation. Learning effective models for complex automata demands more examples and hence the learning process converges slower.

5.5. The effect of the exploration strategy on the agent's performance

In this subsection we compare the four main exploration methods described in this paper: undirected Boltzmann, directed recency-based, combined and lookahead-based exploration.

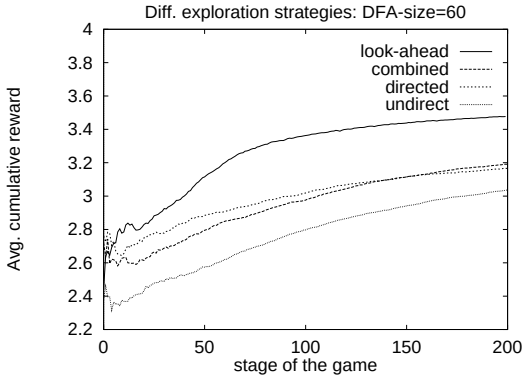

Figure 13. The average cumulative reward of different exploration strategies while playing the IPD against 100 randomly-generated automata of size 60.

The results for opponent automata of size 60 are shown in Figure 13. When the experiment was performed with automata of size 20, the directed and undirected methods showed comparable performance.

However, when the size of the automata was increased to 60, the directed strategies outperformed the undirected one. The exploration behavior of the undirected strategy is sensitive only to the length of the history and not to its content. Therefore it does not modify its exploration behavior when playing against different opponents. The directed strategy, on the other hand, utilizes the content of the history to modify its exploration behavior. This is the reason why it outperforms the undirected method when increasing the complexity of the opponent. The directed method, however, does not reduce its exploration tendency with time. The combined strategy enjoys the advantages of both methods and outperforms them both. The 2-lookahead-based exploration strategy outperforms significantly the other exploration methods. This advantage carries the cost of higher computational costs. Due to this higher costs we performed IPD games of length 200 instead of 400.

6. Conclusions

This work studies exploration methods for model-based learning in multi-agent systems. Exploration is essential for a learning agent while adapting to the other agents, for exploring better alternatives and for avoiding being stuck in sub-optimal behavior. Several exploration strategies, undirected and directed, have been developed for the reinforcement learning paradigm. Section 3 describes ways to incorporate these methods into model-based learning.

There are two issues that should be considered when dealing with exploration. The first one is the balance between exploration and exploitation. The second one is the risk involved in exploration. The undirected and directed exploration methods do not offer a way for a rational agent to consider both factors when interacting with the others. The lookahead-based exploration strategy presented in this work deals with these two issues. The opponent's strategy is represented by a mixed model – a distribution over a set of strategies. Every action is evaluated according to its long run contribution to the expected utility and to the knowledge regarding the opponent's strategy. Risky actions are more likely to be detected by considering their expected outcome according to the alternative models of the opponent's behavior.

Assuming that starting from a certain stage of the game, the set of support contains the target opponent's strategy. Will our algorithm converge to a probability distribution giving the target strategy a probability 1? If the history of the game provides a "good" sample of the opponent's behavior, then, as the history becomes longer, the alternative models proposed by the look-ahead mechanism would become

larger, leading to a decrease in their associated belief and to an increase of the belief in the target strategy. However, the main goal of our algorithm is to obtain high expected *utility*. Obviously, acquiring a good model of the opponent can improve our strategy leading to high utility. Nevertheless, the process of acquiring such a model may be too costly. It is therefore quite possible that the algorithm would purposely avoids actions leading to a better model of the opponent if such actions are judged to be too risky according to the current knowledge.

Model-based learning of interaction strategies in repeated-games has received a lot of attention in the game-theory literature, especially in the context of the expected outcome of a repeated game played by learning players (Jordan, 1991a; Jordan, 1991b; Fudenberg and Levine, 1993; Nachbar, 1997). A simple form of model-based learning in game theory is *fictitious play* (Luce and Raiffa, 1957). In such a game, the agents are myopic, i.e., they only consider the current stage of the game. Furthermore, no exploration is done by the players. The agents update their own beliefs on the opponent's choices for the next interaction, and then choose the optimal response to these beliefs. Each agent models the other agents with a distribution over their set of pure strategies. It can be shown that for two-player zero-sum games, fictitious play converges to a Nash-equilibrium[4].

Kalai and Lehrer (1993) describe a repeated game among Bayesian players who consider the future of the game. Each starts with subjective beliefs on the individual strategies used by each of its opponents. It then uses these beliefs to compute its own optimal strategy. In addition, the players update their subjective beliefs by a Bayesian updating rule throughout the game. They show that under some constraints, a game between Bayesian players will converge to a Nash equilibrium. Their work does not deal with exploration, but mentions that in a rational choice model, the optimal determination of when and how to experiment is difficult. Kalai and Lehrer deal with the general case with no assumptions about the players' strategies. However, the best response and the modeling problems are intractable in the general case. In our framework, by assuming *regular* opponents, we were able to offer algorithms for solving these problems.

Gilboa and Samet (1989) also deal with bounded regular players. They describe a model-based learning strategy for repeated games that learns the best response against any regular strategy. Their procedure enumerates the set of all automata and chooses the current opponent model to be the first automaton in the sequence that is consistent with the current history. Exploration is achieved by designing a sequence

[4] We say that two strategies are found in a Nash-equilibrium if each one of them is the best response against the other.

of actions that distinguish between the current model and the next consistent automaton in the enumeration. The risk involved in exploration is bypassed by assuming that the opponents' strategies are limited to strongly connected automata, where there are no "sinks" and there is always opportunities to regret. For such automata, the learning algorithm is guaranteed to converge to the best response in the limit. This learning procedure is based on exhaustive search in the space of automata, and therefore, is impractical for computational bounded agents.

Fortnow and Whang (1994) show that for any learning algorithm there is an adversary regular strategy for which the learning process will converge to the *best response* in at least exponential number of stages. One way of dealing with this complexity problem is by limiting the space of strategies available for the opponent (Freund *et al.*, 1993; Ron and Rubinfeled, 1995). Mor et al. (1996) follow this paradigm and show that for a limited class of regular strategies, the best response automaton can be learned efficiently. In these methods, exploration is achieved by *random walk*, that is, incorporating randomness into the decision procedure of the agent. Such learning methods embark on long exploration sequences during the course of the game. The high cost of the exploration sequences diminishes in infinite games for the limit-of-the-mean utility function, which measures only asymptotic performance. However, these methods may fail for the discounted-sum utility function which takes into account also immediate rewards.

The main role of the opponent model is to predict its behavior in the future. Choosing a proper class of strategies for modeling is essential for the success of the model-based strategy. If the model class is too restricted it will probably fail in prediction. On the other hand, a too general class will make the best response problem and the learning problem intractable. Often, there are many ways to model a given behavior. Therefore, general background information on the opponent, or its type, is important for discriminating among different possible model classes. This paper concentrates on deterministic finite automata for modeling the agents' strategies. The question of how the model-based framework can be extended for more powerful agents remains open for future research. An interesting class of such agents is the class of adaptive agents. The framework described in this paper assumes a stationary opponent. Extending it to adaptive opponents is difficult and deserves further investigation.

References

D. Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75:87–106, 1987.

Robert Axelrod. *The Evolution of Cooperation*. Basic Books, New York, 1984.

E. Ben-Porath. The complexity of computing a best response automaton in repeated games with mixed strategies. *Games and Economic Behavior*, 2:1–12, 1990.

Donal A. Berry and Bert Fristedt. *Bandit Problems, sequential allocation and experiments*. Chapman and Hall, 1985.

David Carmel and Shaul Markovitch. Learning models of intelligent agents. In *Proceedings of thirteenth National Conference on Artificial Intelligence (AAAI 96)*, pages 62 –67, Portland, Oregon, August 1996.

David Carmel and Shaul Markovitch. Exploration and adaptation in multi-agent systems: A model-based approach. In *Proceedings of the fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97)*, pages 606 – 611, Nagoya, Japan, August 1997.

David Carmel and Shaul Markovitch. Model-based learning of interaction strategies in multi-agent systems. *Journal of experimental and theoretical Artificial Intelligence (JETAI)*, 10:309–332, June 1998.

D. A. Cohn. Neural network exploration using optimal experimental design. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems 6*, pages 679–686. Morgan Caufmann, 1994.

Peter Dayan and Terrence J. Sejnowski. Exploration bonuses and dual control. *Machine Learning*, 25 (1):5–22, 1996.

V. Fedorov. *Theory of optimal experiments*. New-york: Academic Press, 1972.

L. Fortnow and D. Whang. Optimality and domination in repeated games with bounded players. In *Proceedings of the 25th Annual ACM Symposium on Theory and Computing*, pages 741–749, 1994.

Y. Freund, M. Kearns, D. Ron, R. Rubinfeld, R. E. Schapire, and Linda Sellie. Efficient learning of typical finite automata from random walks. In *Proceedings of the 25th Annual ACM Symposium on Theory and Computing*, pages 315–324, 1993.

Yoav Freund, Michael Kearns, Yishay Mansour, Dana Ron, Ronitt Rubinfeled, and Robert E. Schapire. Efficient algorithms for learning to play repeated games against computationally bounded adversaries. In *Proceedings of the Annual Symposium on the Foundations of Computer Science*, pages 332–341, 1995.

D. Fudenberg and D. Levine. Steady state learning and nash equilibrium. *Econometrica*, 61:547 – 574, 1993.

I. Gilboa and D. Samet. Bounded versus unbounded rationality: The tyranny of the weak. *Games and Economic Behavior*, 1:213–221, 1989.

I. Gilboa. The complexity of computing best response Automata in repeated games. *Journal of economic theory*, 45:342 –352, 1988.

J. C. Gittins. *Multi-armed Bandit allocation indices*. New York: Wiley and Sons, 1989.

J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, Mass., 1979.

J. S. Jordan. Bayesian learning in normal form games. *Games and Economic Behavior*, 3:60 –81, 1991.

J. S. Jordan. The exponential convergence of bayesian learning in normal form games. *Games and Economic Behavior*, 4:202 –217, 1991.

L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237 – 285, 1996.

Leslie P. Kaelbling. *Learning in embedded Systems*. MIT Press, Cambridge, Mass, 1993.

Ehud Kalai and Ehud Lehrer. Rational learning leads to Nash equilibrium. *Econometrica*, 61(5):1019–1045, September 1993.

Ehud Kalai. Bounded rationality and strategic complexity in repeated games. In T. Ichiishi, A. Neyman, and Y. Tauman, editors, *Game Theory and Applications*, pages 131–157. Academic Press, San Diego, 1990.

Grigoris I. Karakoulas. Probabilistic exploration in planning while learning. In *Proceedings of the 11th Conference on uncertainty in Artificial Intelligence (UAI 95)*, pages 352 – 361, July 1995.

Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. *Proceedings of the eleventh International Conference on Machine Learning*, pages 157–163, July 1994.

R. D. Luce and H. Raiffa. *Games and Decisions, Introduction and critical survey.* John Wiley & Sons., 1957.

A. W. Moore and C. G. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, 13 (1), 1993.

Yishay Mor, Claudia V. Goldman, and Jeffrey S. Rosenschein. Learn your opponent's strategy (in polynomial time). In G. Weiß and S. Sen, editors, *Adaptation and Learning in Multi-agent Systems, Lecture Notes in AI.* Springer-Verlag, 1996.

J. H. Nachbar. Optimization and rational learning in games. *Econometrica 65 (2)*, 1997.

Christos H. Papadimitriou and J. N. Tsitsiklis. The complexity of Markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.

Christos H. Papadimitriou. On players with a bounded number of states. *Games and Economic Behavior*, 4:122–131, 1992.

Dana Ron and Ronitt Rubinfeled. Exactly learning automata with small cover time. In *Proceedings of the Seventh Annual ACM Conference on Computational Learning Theory*, pages 427–436, 1995.

A. Rubinstein. Finite automata play the repeated Prisoner's Dilemma. *Journal of Economic Theory*, 39:83–96, 1986.

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

Mitsuo Sato, Kenichi Abe, and Hiroshi Takeda. Learning control of finite Markov chains with an explicit trade-off between estimation and control. In *IEEE Transactions on Systems, Man and Cybernetics*, volume 18 (5), September 1991.

S. Sen, M. Sekaran, and J. Hale. Learning to coordinate without sharing information. In *Proceeding of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, pages 426 – 431, Seattle, Washington, 1994.

S. Singh, T. Jaakkola, M. L. Littman, and C. Szpezvari. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine learning Journal (to appear)*, 1998.

Richard S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the 7th international conference on Machine Learning*, pages 216–224, San Mateo CA,, 1990. Morgan Kaufman.

Sebastian B. Thrun. The role of exploration in learning control. In David A. White and Donald Sopfge, editors, *Handbook for Intelligent Control.* Multiscience Press Inc., 1992.

C. J. C. H. Watkins and P. Dayan. Technical notes: Q-learning. *Machine Learning*, 8:279–292, 1992.

G. Weiß and S. Sen. *Adaptation and Learning in Multi-agent Systems, Lecture Notes in AI 1042.* Springer-Verlag, 1996.

Address for correspondence:
Dr. Shaul Markovitch
Computer Science Department
Technion, Haifa 32000, Israel