Exploration and Adaptation in Multiagent Systems: A Model-based Approach

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

Abstract

Agents that operate in a *multi-agent system* can benefit significantly from adapting to other agents while interacting with them. This work presents a general architecture for a model-based learning strategy combined with an exploration strategy. This combination enables adaptive agents to learn models of their rivals and to explore their behavior for exploitation in future encounters. We report experimental results in the *Iterated Prisoner's Dilemma* domain, demonstrating the superiority of the model-based learning agent over non-adaptive agents and over reinforcement-learning agents. The Experimental results also show that exploration can improve the performance of a model-based agent significantly.

1 Introduction

The recent tremendous growth of the Internet has motivated a significant increase of interest in the field of *multi agent systems* (MAS). In such a system, a group of egocentric autonomous agents interact with each other to achieve their private goals. Generally, agents are designed to achieve their masters' goals by trying to maximize some given utility measurement. Designing an "effective" strategy for interaction is a hard problem because its effectiveness depends mostly on the strategies of the other agents involved. However, the agents are autonomous, hence their strategies are private. One way to deal with this problem is to endow agents with the ability to adapt their strategies based on their interaction experience [Weiß and Sen, 1996].

A common technique used by adaptive agents in MAS is *reinforcement learning* [Shoham and Tennenholtz, 1994; Sandholm and Crites, 1995]. The adaptive agent adapts its strategy according to the rewards received while interacting with other agents. A major problem with this approach is its slow convergence. An effective interaction strategy is acquired only after processing a large number of interaction examples. During the long period of adaptation the agent pays the cost of interacting sub-optimally.

In previous work [Carmel and Markovitch, 1996b] we describe a model-based approach that reduces the number of interaction examples needed for adaptation by investing more computational resources in deeper analysis of past interaction experience. This approach splits the learning process into two separate stages. In the first stage, the learning agent infers a model of the other agent based on past interaction. In the second stage the agent utilizes the learned model for designing effective interaction strategy for the future.

A *model-based* adaptive agent might converge to sub-optimal behavior. Acting according to the current best-response strategy may leave unknown aspects of the opponent's strategy unexplored. Finding the balance between *exploitation* of the current model and *exploration* of other alternatives is a well-known dilemma for on-line adaptive agents. In this work we describe methods for combing exploration strategies with model-based learning. The exploring model-based agent sacrifices immediate rewards to explore the opponent behavior; the better model resulted will then yield a better interaction strategy.

The paper is organized as follows: First, we formalize a framework for describing interaction among agents in multi-agent systems. An encounter between agents is described as a *two-player game* and a sequence of encounters as a *repeated game*. We also present a general architecture for a model-based adaptive strategy for repeated games. Second, we describe some exploration methods for on-line learning and show how to combine them with model-based learning. Third, we show experimentally the superiority of a model-based adaptive agent over non-adaptive agent and over reinforcement-learning agent in the *Iterated Prisoner's Dilemma game*. We also report some experimental results comparing different exploration methods for model-based learning in

repeated-games.

2 A general framework for a model-based interacting agent

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.

I \ II	c	d
c	3, 3	5, 0
d	0, 5	1, 1

Figure 1: The payoff matrix for the Prisoner's dilemma game

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, $h(t) = \left\langle (r_1^0, r_2^0), (r_1^1, r_2^1), \ldots, (r_1^{t-1}, r_2^{t-1}) \right\rangle$. $H(G^{\#})$ is the set of all finite histories for $G^{\#}$.

A *strategy* $s_i : H(G^{\#}) \rightarrow R_i$ for player i, $i \in \{1, 2\}$, is a function that takes a history and returns an action. $\mathcal{S}_\rangle$ 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) while playing the game $G^{\#}$, $g_{(s_1,s_2)}(t)$.

The *discounted-sum* function is common utility function for repeated games:

$$U_i^{\gamma}(s_1, s_2) = (1-\gamma) \sum_{t=0}^{\infty} \gamma^t u_i(s_1(g_{(s_1,s_2)}(t)), s_2(g_{(s_1,s_2)}(t))) \quad (1)$$

$0 \leq \gamma < 1$ is a discount factor for future payoffs of player i. It is easy to show that $U^{\gamma}(s_1, s_2)$ converges for any $\gamma < 1$.

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

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 proven to be extremely 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^{γ} depends on the discount parameter γ [Axelrod, 1984]:

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

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 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 for 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.

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 acquires 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.

The efficiency of this adaptive strategy depends mainly 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. In previous work [1996b] we show that when the players are restricted to *regular strategies*, i.e. strategies that can be

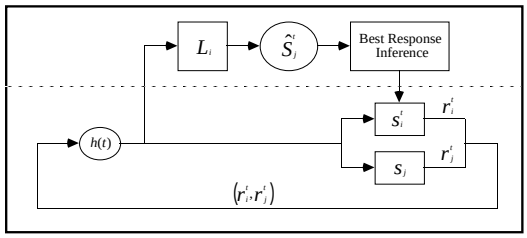

Figure 2: An architecture for a model-based learning agent in repeated games.

represented by deterministic finite automata (DFA) [Rubinstein, 1986], the best-response problem can be solved efficiently. We also describe a learning algorithm for inferring a regular model based on the history of the game.

3 Exploring the opponent's strategy

One of the weaknesses of the *model-based* approach is that it might converge to sub-optimal behavior. The best-response strategy ignores the possibility that the current model is not identical to the opponent's strategy and that other actions might have a better utility (according to the actual opponent's strategy). The left part of Figure 3 shows an example of an opponent's strategy for IPD. 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" which yields sub-optimal utility. Furthermore, playing "all-d" prevents the player from observing counterexamples, therefore, the wrong model will never be corrected and the player will never discover the best-response for the actual opponent strategy – "play c and than all-d".

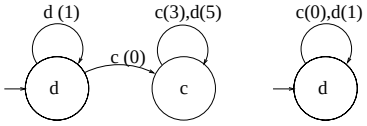

Figure 3: An example of an opponent model in local minimum. (Left): An Opponent's strategy. (Right): An opponent model. The model dictates the play "all-d" while the actual best-response is "play c and then all-d".

The example above demonstrates that it is better sometimes to play sub-optimally in order to explore the opponent's behavior. This phenomenon is known as the dilemma between exploration and exploitation. The learning player has to choose between the wish to exploit the current model maximally, and the desire to explore other alternatives to improve its future play.

In recent years a variety of exploration strategies have been proposed for reinforcement-learning agents. *indirect strategies* are based on incorporating randomness into the decision procedure of the learning agent. The key idea is to associate a positive probability for any state-action pair. The agent randomly selects an action according to the given distribution and therefore no action is neglected forever. A common exploration strategy in the reinforcement learning paradigm is to play according to a distribution that takes into account the expected utility of the actions and the learning stage. We will use similar method for the MB-agent with the discounted-sum utility function. 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 following the optimal policy from then on.

Let $\hat{s}_j^t = L_i(h(t))$ be an opponent model inferred by the learning algorithm L_i based on history $h(t)$. Let $s_i^t = s_i^{opt}(\hat{s}_j^t, U_i^\gamma)$ be the best-response for player i. The *expected utility* for action r is defined by:

$$U_i'(\hat{s}_j^t, r) = u_i\left(r, \hat{s}_j^t(h(t))\right) + \gamma U_i^\gamma\left(s_i^t, \hat{s}_j^t\right)$$

U_i' can be computed efficiently since any game between two automata converges to a finite cycle, and the discounted sum of rewards can be computed by analyzing the expected game-cycle of the two automata.

The Boltzmann distribution [Sutton, 1990] assigns a probability for any possible action according to its expected utility and according to a decreasing parameter T called a temperature.

$$Pr(r) = \frac{e^{U_i'(\hat{s}_j^t, r)/T}}{\sum_{r' \in R_i} e^{U_i'(\hat{s}_j^t, r')/T}} \quad (2)$$

T depends on the stage of the game and determines the rate of convergence. We use a common temperature function $T = \alpha^t$ where $\alpha < 1$ is the exploration parameter.

Direct strategies try to explore the environment more efficiently by using statistics based on the learning experience of the agent. These heuristic methods compute an exploration bonus for each possible action and incorporate this bonus into the decision procedure of the agent. Let $E_i'(\hat{s}_j^t, r)$ be the exploration bonus computed for an action r according to the current model $\hat{s}_j^t$ and the history $h(t)$. The value for each action is computed according to the following formula:

$$V_i(\hat{s}_j^t, r) = (1-\alpha)\frac{U_i'(\hat{s}_j^t, r)}{\sum_{r' \in R_i} U_i'(\hat{s}_j^t, r')} + \alpha\frac{E_i'(\hat{s}_j^t, r)}{\sum_{r' \in R_i} E_i'(\hat{s}_j^t, r')} \quad (3)$$

As for indirect methods, α is the exploration parameter that determines the ratio between exploration and exploitation.

Statistics might include action-counts [Sato *et al.*, 1991] where higher exploration bonuses are given to actions that have been chosen less frequently. Sutton [1990] suggests an exploration bonus based on the time that

have passed since the action was taken. Kaelbling [1993] suggests an exploration bonus based on the upper bound of the confidence interval computed for the expected utility of the possible actions.

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

4 Experimentation: On-line learning in repeated-games

We conducted a set of experiments to test the capabilities of a model-based learner in repeated games. The stage-game used in these experiments is the PD-game. The first experiment tests the basic capabilities of an on-line learner against random opponents, the second compares indirect and direct exploration methods, and the third tests the MB-agent against non-random automata.

For each experiment, a random opponent automaton was generated by choosing a random transition function and a random output function. The MB-agent begins with a random DFA as a model of its opponent's strategy and plays according to the *best response* strategy combined with an exploration strategy. The opponent model is modified whenever it fails to predict the opponent's actual play by using the learning algorithm IT-US-L^* with $(h_i(t), r_j^t)$ as a counterexample [Carmel and Markovitch, 1996b]. We use the average cumulative reward achieved by the player to measure the quality of its learning strategy:

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

Figure 4 shows the average cumulative reward attained during 400 stages of the IPD game, averaged over 100 trials against 100 different random automata of size 20. The lowest curve shows the results of the non-adaptive player TFT. TFT achieves only 2.25 points, which is the average payoff of the PD game, and this result is not improved during the game.

The three upper curves stand for MB-strategies combined with Boltzmann strategies with different exploration parameters, $T = 5.0 * \alpha^t$. All the MB-strategies achieve results that are far superior to those achieved by TFT. The exploring adaptive agents are far more successful than the non-exploring agent (with $\alpha = 0$). The graphs highlight an interesting phenomenon. At early stages of the game, the exploring agents pay for the sub-optimal 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.

The second curve shows the results of a reinforcement-learning (RL) agent. RL is based on the idea that the tendency to produce an action should be reinforced if it produces a favorable results, and weakened if it produces unfavorable results. 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 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 cause the agent to use 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 Tit-for-tat (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 and a Boltzmann exploration strategy. It is quite clear that the performance of the Q-agent is much inferior to that of the MB-agent in the given domain. Increasing the width of the window did not help. The Q-agent managed to achieve results comparable to those achieved by the MB-agents only when the length of the game was increased up to 40,000.

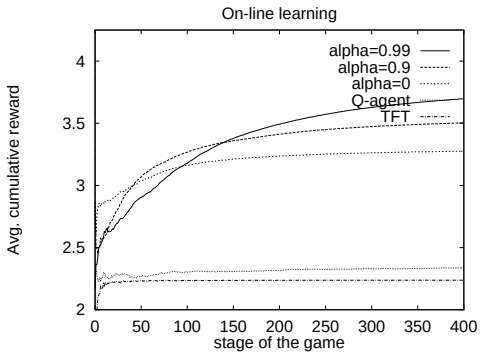

Figure 4: On-line learning: The average reward attained by different playing strategies against 100 random automata of size 20.

The results for the MB-agents 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. For each γ we tried several values of exploration parameter α and recorded the one for which the agent achieved the best performance. The left part of Figure 5 shows the best α for each γ. As expected, as γ increases, it is better to invest more on exploration by using larger α.

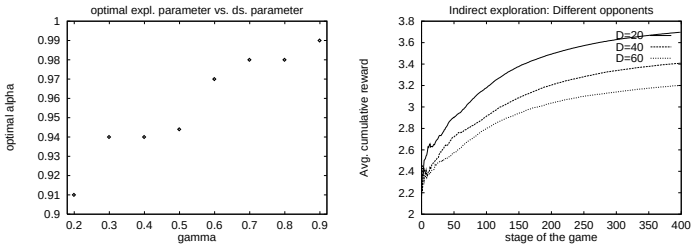

Figure 5: Left: The best exploration parameter for various discount parameters. Right: On-line learning of random automata with various sizes.

The right part of Figure 5 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 $\alpha = 0.99$ against 100 random automata of sizes 20, 40, and 60. The results show that increasing the size of the automata deteriorates the rate of adaptation. Learning effective models for complex automata demands more examples and hence the learning process converges slower.

In the second experiment we compared indirect and direct exploration strategies. The indirect-MB agent uses Boltzmann distribution with temperature function $T = 5 * 0.99^t$. The direct-MB agent uses *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, $\rho(\hat{s}_j^t, r)$, at the current state of the opponent model $\hat{s}_j^t$. 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 3, using $\alpha = 0.1$. For automata of size 20, the direct and indirect methods showed comparable performance. However, when the size of the automata was increased to 40, the direct strategies outperformed the indirect one. The results are shown in Figure 6.

The exploration behavior of the indirect 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 direct strategy, on the other hand, utilizes the content of the history to modify its exploration behavior. This is the reason why it outperforms the indirect method when increasing the complexity of the opponent. However, the

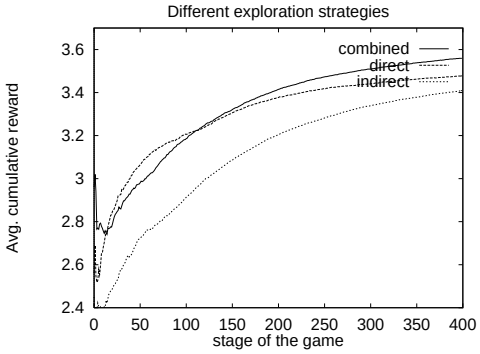

Figure 6: Different exploration strategies while playing IPD against 100 random automata of size 40.

direct method does not reduce its exploration tendency with time. Therefore, even after learning a perfect model of the opponent, it will still pick non-optimal actions.

We have designed a *combined* exploration strategy that uses a combination of the direct and indirect strategies: it uses *recency-based* exploration strategy but decreases the exploration parameter with time, $\alpha_t = 0.1 * 0.99^t$. Figure 6 shows that the combined strategy enjoys the advantages of both methods and outperforms them both.

In the third experiment we tested the MB-strategy against non-random opponents that were hand-crafted specifically for the IPD game. We repeated the famous tournament organized by Axelrod [1984] for IPD. In the original tournament, fifteen attendees (strategies) competed in a round-robin tournament, where every interaction was based on 200 repetitions of the PD game. The participated strategies were sent as computer programs by different researchers around the world. Most programs were designed to basically cooperate. They differ mainly in the way that they deal with defection of their rivals. For this experiment we allowed only deterministic regular strategies to participate. Figure 7 shows the models and the best-responses learned by the MB-agent for five different opponents. The MB-agent learned its opponents on-line during the 200 iterations using Boltzmann exploration with the temperature function $T = 5 * 0.99^t$. The best-response strategies were computed according to the discounted sum utility function with $\gamma = 0.9$.

The MB-agent succeeded very well against the strategies described above. The adaptive strategy begins without any knowledge about the game and only few iterations are needed to converge to the *best response*: Cooperate with TFT and exploit players like TF2T and Pavlov. The best-response for Nydegger which is "all-c" was found after about 100 iterations. The model given in the figure was learned after 200 iterations. It encapsulates many features of Nydegger. For example it cooper-

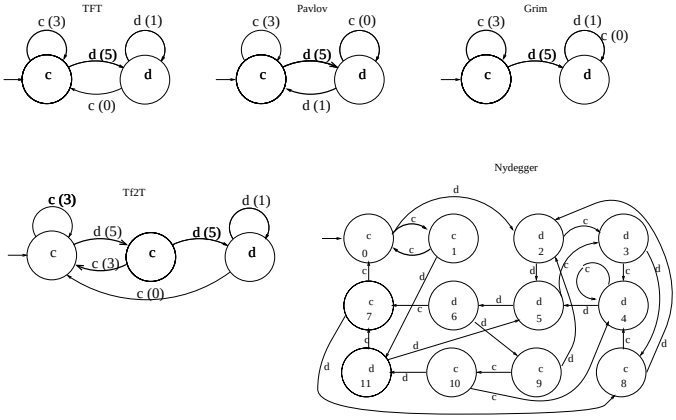

Figure 7: The models learned by a MB-agent after 200 iterations of the PD-game. The best-response cycles are highlighted.

ates after 3 mutual defections (see the paths 2-5-6-9, 4-5-6-9, 11-5-6-9). The MB-agent also succeeds in finding the right model for Grim. However, the exploring agent falls into the "defection sink" and is not being able to get back to the cooperation cycle. This problem is common to all exploring strategies that eventually will explore the "wrong" action and will fall into the sink. Grim is a typical example for strategies that are problematic for on-line adaptive players.

5 Conclusions

This work presents a model-based approach for learning interaction strategies. We present an architecture for a model-based learner that uses the *best-response* strategy combined with an exploration strategy to explore its opponent's behavior for exploitation in future encounters.

We conducted a set of experiments where an adaptive model-based agent played against randomly generated opponents. In these experiments, the MB-agent performed significantly better than non-adaptive agents and Q-agents. This result supports our claim that it is more effective to learn a model and use it for designing a best-response strategy than trying to directly learn such a strategy. We compared direct and indirect exploration methods and showed that a combination of these two methods can have an advantage. We also showed that the MB adaptive learner, without any background knowledge, outperforms hand-crafted opponents in the IPD domain.

The above experimental results demonstrate average-case performance of our learning strategy. In the worst-case, Fortnow and Whang [1994] show that for any learning algorithm there is an adversary DFA 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 automata available for the opponent [Mor *et al.*, 1996]. Such methods are based on long exploration sequences during the course of the game. The high cost of such exploration sequences diminishes in infinite games for utility functions that measure only asymptotic performance. However, these methods may fail for the discounted-sum utility which takes into account also immediate rewards. The exploration methods described in this work take into account the cost of exploration and are therefore suitable for such utility functions as well.

One of the basic assumptions of our framework is that the opponent is stationary. When this assumption is removed, we may want to look at windows of the input rather than the complete history, and to increase the exploratory rate of the algorithms. An interesting class of non-stationary opponents, is the class of adaptive opponents. For example, we might assume that the opponent is a MB-adaptive agent. This can be further extended to a general framework of recursive modeling, where a player holds a model of the opponent which is also a modeling player. We have developed a similar framework for two-player zero-sum games [Carmel and Markovitch, 1996a].

References

[Axelrod, 1984] R. Axelrod. *The Evolution of Cooperation*. Basic Books, New York, 1984.

[Carmel and Markovitch, 1996a] David Carmel and Shaul Markovitch. Incorporating opponent models into adversary search. In *Proceedings of thirteenth National Conference on Artificial Intelligence (AAAI 96)*, pages 120 – 125, Portland, Oregon, August 1996.

[Carmel and Markovitch, 1996b] 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.

[Fortnow and Whang, 1994] 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.

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

[Mor *et al.*, 1996] Yishay Mor, Claudia V. Goldman, and Jeffery 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.

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

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

[Sato *et al.*, 1991] 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.

[Shoham and Tennenholtz, 1994] Y. Shoham and M. Tennenholtz. Co-Learning and the evolution of social activity. Technical Report STAN-CS-TR-94-1511, Stanford University, Department of Computer Science, 1994.

[Sutton, 1990] 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.

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

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