Reinforcement Learning Algorithms for Uncertain, Dynamic, Zero-Sum Games

Snehasis Mukhopadhyay

Department of Computer and Information Science
Indiana University Purdue University Indianapolis
Indianapolis, IN 46202
Email: smukhopa@cs.iupui.edu

Omkar Tilak

Department of Computer and Information Science
Indiana University Purdue University Indianapolis
Indianapolis, IN 46202
Email: omkar.tilak@gmail.com

Subir Chakrabarti

Department of Economics
Indiana University Purdue University Indianapolis(IUPUI)
425 University Blvd.
Indianapolis, IN 46202
Email: imxl100@iupui.edu

Abstract—Dynamic zero-sum games are a model of multi-agent decision-making that has been well-studied in the mathematical game theory literature. In this paper, we derive a sufficient condition for the existence of a solution to this problem, and then proceed to discuss various reinforcement learning strategies to solve such a dynamic game in the presence of uncertainty where the game matrices at various states as well as the transition probabilities between the states under different agent actions are unknown. A novel algorithm, based on heterogeneous games of learning automata (HEGLA), as well as algorithms based on model-based and model-free reinforcement learning, are presented as possible approaches to learning the solution Markov equilibrium policies when they are assumed to satisfy the sufficient conditions for existence. The HEGLA algorithm involves automata simultaneously playing zero-sum games with some automata and identical pay-off games with some other automata. Simulation studies are reported to complement the theoretical and algorithmic discussions.

Keywords—Dynamic zero-sum games, multi-agent learning, learning automata, shapley recursion, reinforcement learning.

I. INTRODUCTION

In dynamic zero-sum games, there are two agents (or, two groups of agents) who play a different state-dependent zero-sum game at each state and whose actions results in a probabilistic transition of the underlying game (environment) state. When the underlying game matrices at all the finite number of states as well as the action-dependent state transition probability matrices are completely known, well known computational techniques such as Shapley Recursion [1] can be used to find a solution to the dynamic zero-sum game problem. How to solve this problem in the presence of uncertainty when the reward matrices as well as the transition probabilities are unknown, is within the purview of multi-agent reinforcement learning.

Game theory is a branch of mathematics that aims to analyze and synthesize rational group behavior in multi-agent decision-making systems. The extent to which it succeeds in doing so is subject to some debate; however, there is

no alternative theory that does better than game theory in this respect. Computational game theory (CGT) is an area of research that relies on mathematical game theory for analysis, but then proceeds to develop provable, efficient computer algorithms to realize the solutions to the underlying game problems. Starting with the classic works of Lemke Howson [2], Shapley [3] and others, CGT has emerged from treatment of relatively simple game scenarios to more complex games under uncertainty. This now forms much of the formal basis to the recent field of multi-agent reinforcement learning.

[4] is a special journal issue consisting of several papers in the area of learning and computational game theory that relate to computational issues that arise in game learning algorithms. Earlier, in [5], the author described a Q-learning-like algorithm (called minimax-Q) for finding optimal policies and demonstrates its application to a simple two-player dynamic zero-sum game in which the optimal policy may be probabilistic. More recently, in [6], the authors present an approximation algorithm for the NP-complete problem of computing a Nash equilibrium for a specific kind of game formulated as a digraph coloring game. Zero-sum games is also a important class of general mathematical game problems where there are two agents with exactly opposite pay-offs (i.e., if one receives $+x$, the other receives $-x$). Such zero-sum game formulations have been applied to many practical problems including computer security [7], transmission rate adaptation in multimedia networks [8], as well as drug design and treatment planning in the medical domain [9].

In this paper, we consider dynamic zero-sum games defined over an underlying multi-agent Markov Decision Process (MDP) state space. Our interest will be for each agent to learn a Markov equilibrium strategy which is a function of only the current state. Further, this strategies need to be pure (i.e., deterministic) strategies. While it is well-known that all zero-sum games has a Nash equilibrium, such equilibria may be pure (deterministic) or mixed (probabilistic). There is no known necessary and sufficient conditions for the existence of Markov, pure equilibrium strategies. However, without

knowing whether such deterministic solutions exist, it is not meaningful to devise a learn a deterministic strategy, as it is not clear what the agents are attempting to learn in such a case.

The objectives of this paper are three-fold. First, integrating and exploiting the synergy between concepts and methods from Mathematical Game Theory and Multi-agent Reinforcement Learning Algorithms, the paper presents novel analysis and synthesis techniques to compute solutions to complex, zero-sum game decision-making problems. This includes techniques to analytically derive sufficient conditions under which game theoretic equilibria with computable structures can be shown to exist. Then, towards the second objective, assuming solutions exist, multi-agent reinforcement learning algorithms will be developed to learn the equilibria strategy using trial and error, when the underlying game problem is not completely pre-specified. The third objective of the paper is to experimentally demonstrate the practical viability of the developed algorithms by means of extensive simulation studies.

In this paper, we discuss several approaches of reinforcement learning algorithms that can be used to solve such uncertain dynamic zero-sum game problems, based on trial-and-error outcomes. Some of these methods are based on adaptive Shapley Recursion, and minimax Temporal Difference ([5]). A novel algorithm based on so-called HEterogeneous Games of Learning Automata (HEGLA) is also presented, extending the work reported in ([10]) for a single-agent problem, where each state of the underlying Markov Decision Process (MDP) has two learning automata ([11]) maximizing and minimizing the pay-off respectively. All maximizing automata in all the states play a static identical pay-off game among each other. Similarly, all minimizing automata play a static identical pay-off game with each other. However, the maximizing team play a zero-sum game with the minimizing team. Such incorporation of heterogeneous kinds of games within the same overall team of automata, is both novel and interesting, and forms a basis of the so-called approach of Partially Decentralized Group of Learning Automata (PDGLA).

The PDGLA framework allows a group of learning automata to be subdivided into multiple subgroups. Each subgroup acts as a locally centralized unit, although there is no centralization of the entire group. However, the PDGLA formalism requires the learning algorithms used across all the subgroups to be of the same *type*. Each subgroup can alternatively use a centralized or decentralized approach. But each subgroup participates in one and only one type of game: namely an identical-payoff game. Thus we termed this configuration as HOMogeneous Games of Learning Automata (HOGLA).

However, the PDGLA framework can be made more expressive by allowing different subgroups to participate in different types of games. In this thesis, we focus on two types of games: identical-payoff games and zero-sum games. Thus, one can imagine a configuration where the automata group is divided into two subgroups. While the automata residing in a one subgroup participate in an identical-payoff game. However, the automata in the other subgroup participate in a zero-sum game. This concept can be easily extended to scenarios where there are more than two subgroups present. In such case, automata in a few subgroups will participate in

an identical-payoff game while the automata belonging to the other subgroups will participate in a zero-sum game. We call this framework HEterogeneous Games of Learning Automata (HEGLA) [12].

In addition to the novel HEGLA solution [12], we propose an Adaptive Shapley Recursion (ASR) solution for the control of DZSG problem. We use the dynamic programming based Shapely recursion algorithm [1] to estimate optimal policies for given configuration of DZSG. This optimal policy is then used to estimate the optimal action in a given state. State transition and reward probabilities are estimated based on the frequency with which different transitions are performed and the feedback obtained during these transitions. These estimated parameters of the chain are used as the input to the Shapely algorithm in a recursive manner. In addition, we also discuss a Temporal Difference (TD) style algorithm for the control of DZSGs. We call this algorithm Minimax-TD algorithm. Minimax-TD algorithm estimates the Q-values of every state and action tuple pair using TD formula and these estimates are used for the control operation. The actions for each agent are selected by calculating saddle point in the Q-matrix and the Q-values are then updated based on the reward obtained by performing the selected actions.

II. DYNAMIC ZERO SUM GAMES

A DZSG (defined mathematically in the next section) involves two agents or players. The two agents are called as Maximizing (or Row) player and Minimizing (or Column) player respectively. There are a finite number of states of the game in each of which the two agents play a zero sum game. The state transitions in DZSGs are the result of joint action of both the agents. In turn, the instantaneous rewards also depend on the selected joint actions. The object of each player is to maximize (or, minimize) of overall cumulative, discounted total pay-off.

A dynamic programming solution that optimizes the expected reward for the DZSG when complete information regarding transition and reward probabilities are available was given by Shapley [1]. However, the computational cost of this solution increases dramatically with increasing in number of states and actions. Thus the problem introduced by the presence of many states remains a major practical limitation for this dynamic programming based approach. In this regard, decentralization is a priority and serves to reduce the debilitating effect of large state space. Towards that end, we present a novel, completely decentralized learning algorithm using HEGLA. No knowledge of either transition probabilities or reward values is assumed, other than that these values are normalized to lie in the interval $[0, 1]$. We use L_{R-I} learning algorithm for LA to update its action probabilities so no explicit parameter estimation is needed. Also, the control scheme is implemented in a decentralized fashion.

III. AN EXISTENCE RESULT

A dynamic zero sum game (DZSG) formally is given by $\{S_1, S_2, \dots, S_N; A_1, A_2; R_1, R_2\}$ where $S = \{S_i\}$, $i = 1, 2$ are the discrete set of states of the Markov Chain, A_j , $j = 1$ or 2 , discrete action sets. Player 1 (the row player) is the maximizing agent and 2 (the column player) is the minimizing

agent. $A = A_1 \times A_2$ and $T : S \times A \times S \rightarrow [0, 1]$ is the transition function. $R_i : S \times A \times S \rightarrow R$ is the reward function with $R_1 = -R_2$. The single-period reward function of player 1 is $r_1 : S \times A \rightarrow R$. For player 2, the single-period reward function is $r_2 : S \times A \rightarrow R$ where $r_2 = -r_1$. If the future expected payoff of player 1 is $F_1 : S \rightarrow R$, then the payoff of is given

$$R_1(S_i, a, S_k) = r_1(S_i, a) + \delta F_1(S_k)$$

where $0 < \delta < 1$ is the discount rate used to discount future payoffs. As this is a Zero-sum game it follows that $F_2 = -F_1$.

While it is well known that a Zero sum game has a value and an equilibrium when mixed strategies are allowed, zero sum games do not necessarily have an equilibrium in pure strategies as is illustrated by the following game of matching pennies

	H	T
H	1	-1
T	-1	1

Therefore, in order to guarantee the existence of an equilibrium in pure strategies in DZSM, a necessary condition is that the single-period game associated with each state S_i has a pure strategy equilibrium point. In fact, the equilibrium points must be a strict equilibrium. Recall that a *strict equilibrium* is an equilibrium in which there is a unique best reply to the strategies of the other players. In other words a deviation from a strict equilibrium gives a strictly lower payoff to the deviating player. Denote by $G(S_i)$ the matrix game for which the rows are given by the discrete choices in A_1 and the columns are given by the discrete choices in A_2 and the entry in the m -th row and the n -th column is $r_1(S_i, A_{1,m}, A_{2,n})$.

Lemma 1: If for each state S_i , the matrix game $G(S_i)$ has a strict equilibrium, then there is a $\bar{\delta}$ such that for all $\delta \leq \bar{\delta} < 1$, there is a Markov-perfect equilibrium of the DZSM.

Proof: Let $f^* : \{S_1, \dots, S_N\} \rightarrow G(S_\ell)$ denote a selection of the strict equilibrium points of the matrix games $G(S_\ell)$. As $f^*(S_\ell)$ is a strict equilibrium $G(S_\ell)$, therefore for any S_ℓ and $a_1 \in A_1$

$$r_1(S_\ell, (a_1, f_2^*(S_\ell))) - r_1(S_\ell, f^*(S_\ell)) < 0$$

Therefore, for each S_ℓ and $f^*(S_\ell)$ there is a $\bar{\delta}_{1,S_\ell} > 0$ such that for $\delta \leq \bar{\delta}_{1,S_\ell}$

$$\begin{aligned} & \max_{a_1 \in A_1} \{[r_1(S_\ell, (a_1, f_2^*(S_\ell)))] \\ & + \delta \sum_{k=1}^N r_1(S_k, f^*(S_k))q(S_k|S_\ell, (a_1, f_2^*(S_\ell)))\} \\ & - [r_1(S_\ell, f^*(S_\ell)) + \delta \sum_{k=1}^N r_1(S_k, (f^*(S_k))q(S_k|S_\ell, f^*(S_\ell)))] \\ & = \max_{a_1 \in A_1} \{[r_1(S_\ell, (a_1, f_2^*(S_\ell))) - r_1(S_\ell, f^*(S_\ell))] \\ & + \delta [\sum_{k=1}^N r_1(S_k, f^*(S_k))q(S_k|S_\ell, (a_1, f_2^*(S_\ell))) \\ & - \sum_{k=1}^N r_1(S_k, (f^*(S_k))q(S_k|S_\ell, f^*(S_\ell)))]\} \end{aligned}$$

$$\leq 0.$$

That is, for $\delta \leq \bar{\delta}_{1,S_\ell}$

$$\begin{aligned} & r_1(S_\ell, f^*(S_\ell)) + \delta [\sum_{k=1}^N r_1(S_k, f^*(S_k))q(S_k|S_\ell, f_2^*(S_\ell))] \\ & \geq r_1(S_\ell, (a_1, f_2^*(S_\ell))) \\ & + \delta [\sum_{k=1}^N r_1(S_k, f^*(S_k))q(S_k|S_\ell, (a_1, f_2^*(S_\ell)))] \end{aligned} \quad (1)$$

for all $a_1 \in A_1$.

Claim 1: The Stationary Markov strategy f_1^* of player 1 is an optimal stationary strategy of player 1 when player 2 uses the stationary Markov strategy f_2^* for $\delta \leq \bar{\delta}$.

Let $R_1^* : \{S_1, \dots, S_N\} \rightarrow \mathbb{R}$ denote the future expected payoff of player 1 when player 1 use the stationary Mark strategy f_1^* and player 2 uses the stationary Markov strategy f_2^* . The expected payoff of player 1 starting from period 1 is

$$r_1(S_\ell, f^*(S_\ell)) + \delta \sum_{z=1}^N R_1^*(S_z)q(S_z|S_\ell, f^*(S_\ell)) \quad (2)$$

But this is also equal to

$$\begin{aligned} & r_1(S_\ell, f^*(S_\ell)) + \delta \sum_{k=1}^N r_1(S_k, f^*(S_k))q(S_k|S_\ell, f^*(S_\ell)) \\ & + \delta^2 \sum_{k=1}^N \sum_{z=1}^N [R_1^*(S_z)q(S_z|S_k, f^*(S_k))]q(S_k|S_\ell, f^*(S_\ell)) \end{aligned} \quad (3)$$

(4)

In general the expected payoff of player 1

$$\begin{aligned} & r_1(S_\ell, f^*(S_\ell)) + \delta \sum_{k=1}^N r_1(S_k, f^*(S_k))q(S_k|S_\ell, f^*(S_\ell)) + \\ & \delta^2 \sum_{k=1}^N \sum_{z=1}^N [r_1(S_z, f^*(S_z)) \times \\ & q(S_z|S_k, f^*(S_k))]q(S_k|S_\ell, f^*(S_\ell)) + \dots + \\ & \delta^{T-1} \underbrace{\sum_{k=1}^N \sum_{z=1}^N \dots \sum_{w=1}^N [r_1(S_w, f^*(S_w)) \times} \\ & \underbrace{q(S_w|S_{T-1}, f^*(S_{T-1})) \dots q(S_k|S_\ell, f^*(S_\ell))}_{T-1}] + \\ & \delta^T \underbrace{\sum_{k=1}^N \dots \sum_{p=1}^N [R_1^*(S_p)q(S_p|S_T, f^*(S_T)) \times} \\ & \underbrace{q(S_w|S_{T-1}, f^*(S_{T-1})) \dots q(S_k|S_\ell, f^*(S_\ell))}_{T-1}]. \end{aligned} \quad (5)$$

If player 1 uses some other stationary Markov strategy $\hat{f}$, then the expected payoff of player 1 would be given by an expression like (4) with the single-period payoffs being

$r_1(S_\ell, \hat{f}_\ell(S_\ell))$ for $\ell = 1, \dots, \ell$. As f^* is a selection from the set of strict Nash equilibria of the game G_ℓ , we have

$$r_1(S_w, f^*(S_w)) - r_1(S_w, \hat{f}(S_w)) \leq 0$$

for all $w = 1, \dots, N$. Therefore, from (1) and from (4) we observe that after any period T , the difference between the expected payoffs from f_1^* and $\hat{f}_1$ for player 1 is at most

$$\delta^T \sum_{p=1}^N [R_1[\hat{f}_1](S_p)q(S_p|S_T, \hat{f}(S_T)) - \quad (6)$$

$$R_1^*(S_p)q(S_p|S_T, f^*(S_T))]. \quad (7)$$

As $\delta < 1$, this implies that player 1 cannot increase his payoff in the infinite horizon game by using any other stationary Markov strategy. This proves the claim that f_1^* is an optimal stationary Markov strategy of player 1.

We now observe that there is a $\bar{\delta}_2$ for player 2 for which f_2^* is an optimal stationary Markov strategy of player δ_2 using the same arguments as used to establish the optimality of f_1^* for player 1. But this shows that for $\delta \leq \min\{\bar{\delta}_1, \bar{\delta}_2\}$, f^* is a stationary Markov equilibrium of the DZSM. ■

IV. WHEELER-NARENDRA CONTROL ALGORITHM

The Wheeler-Narendra control algorithm [10] for a single-agent MDP uses the framework of identical payoff game of learning agents to solve the control problem. In the configuration proposed in [10], one learning automaton is associated with each state of the MDP. Each LA uses simple L_{R-I} learning scheme [11] to update its action probabilities.

When the Markov chain transitions to a state, the automaton acting within that state receives information about the cumulative reward generated by the chain so far and the current global time from the central controller. From these, the automaton calculates the average reward value which is used as the payoff β for the learning process. The L_{R-I} algorithm is used to update the action probabilities of the automaton in the following manner:

$$p(k+1) = p(k) + \lambda\beta(e_\alpha - p_i(k)), i = l, 2, \dots, N$$

where $0 < \lambda < 1$ is a parameter. α is the action selected by this automaton during previous time when Markov chain was in the current state and e_α is a unit vector of appropriate dimension with α -th component unity.

We use a similar technique for the control of DZSGs. Two LAs are associated with each state of the DZSG (one for the ROW player or the MAX player and other for the COLUMN player or the MIN player) which learn using L_{R-I} algorithm. Also, the environment response for the the proposed HEGLA-based control scheme is calculated as the average reward generated so far by the Markov chain.

V. SHAPLEY RECURSION

Shapely recursion [1] uses Dynamic Programming in a recursive manner. During each iteration, an estimated value matrix for every state of the Markov chain is calculated. The minimax value corresponding to this value matrix is calculated which is then used in the next iteration. It is proven that the

ordering in this value matrix becomes same as the ordering in the actual game matrix in an asymptotic manner. Author proves that this algorithm converges to the optimal strategy asymptotically. Assume that the DZSG consists of N states. We use $\|A\|$ to denote the cardinality of the set A . We use the operator *minimax* to denote minimax value obtained by the maximizing player. It is easy to prove that for two matrices X and Y ,

$$|\minmax[X] - \minmax[Y]| \leq i, j \max |x_{ij} - y_{ij}|$$

Let $\vec{\alpha}$ be the N -dimensional, numeric vector where. For each state s of the Markov chain, the algorithm calculates a value matrix $A^s(\vec{\alpha})$ as

$$R(s, i, j) + \sum_{l=1}^N T(s, i, j, l) \alpha^l$$

where $i = 1, 2, \dots, \|A_1\|$. At the start of the recursion, values in $\vec{\alpha}(0)$ are set arbitrarily. As mentioned earlier, $\vec{\alpha}(t)$ is calculated recursively using a dynamic programming technique as follows:

$$\vec{\alpha}^s(t) = \minmax[A^s(\vec{\alpha}(t-1))]$$

It is shown that the limit of $\vec{\alpha}(t)$ as $t \rightarrow \infty$ exists and is independent of $\vec{\alpha}(0)$, and its components are the optimal values of the stochastic game.

Shapley recursion is a dynamic programming technique which computes optimal strategy given all the parameters of the game. Thus, Shapely recursion needs complete information about the state of the chain with all the transition and reward probabilities known at the start of the algorithm. However, for many systems, these values are not known in advance. Thus there is a need to learn these values dynamically as the algorithm evolves. Also, the computational cost of Shapley recursion increases dramatically with increase in the number of states of the DZSG. Thus decentralization is a desirable characteristic of the algorithm used to control DZSGs. Towards this end, we propose our novel HEGLA-based algorithm which operates in a completely decentralized manner.

VI. HEGLA BASED ALGORITHM FOR DZSG CONTROL

As described earlier, in HEGLA, a subset of LAs participate in identical-payoff game while another subset of LAs participate in zero-sum game. We assign one LA with each agent/player present in each state of the DZSG. Thus one LA is associated for maximizing (Row) player and one LA is associated with the minimizing (Column) player in every state of the DZSG.

The LAs acting in a state s_i are not aware of the one-step reward values $R(s_i, A)$ resulting from their selected action tuple $\in A$. The LAs in a state s_i receive information about the effect of their selected actions only when the Markov chain returns to that state. At each state s_i , we maintain two matrices: $REW^{s_i}(A, T)$ stores the cumulative reward obtained for action tuple $\in A$ upto time T and $TIME^{s_i}(A, T)$ stores time passed since the action tuple $\in A$ was last tried.

From these, the algorithm computes Δ_T as the elapsed global time since the current state was last visited and Δ_R

as the corresponding change in the global reward. Then we compute

$$REW^{s_i}(A, T+1) = REW^{s_i}(A, T) + \Delta_R$$

and

$$TIME^{s_i}(A, T+1) = TIME^{s_i}(A, T) + \Delta_T$$

Based on this information, the reward for the maximizing player is calculated as: $\beta^{Max} = \frac{REW^{s_i}(A, T+1)}{TIME^{s_i}(A, T+1)}$. The reward for minimizing player is given by: $\beta^{Min} = 1 - \beta^{Max}$.

Both maximizing player and minimizing player LAs update their action probabilities as follows:

$$p^{Max/Min}(k+1) = p^{Max/Min}(k) + \lambda \beta^{Max/Min}(e_{\alpha_j} - p^{Max/Min}(k))$$

where α_j is the action chosen by Max (or Min) player and e_{α_j} is a unit vector of appropriate dimension with α_j -th component set to unity and all other components are set to zero.

We now prove that the configurations of LAs proposed in this heterogeneous game algorithm will converge to the optimal policy for a given DZSG. Let M_i^k indicate the LA associated with each agent i (Max-player and Min-player) in the state k . Let A_i^k be the corresponding action set. Then $\Gamma(k) = (N, A^k, J(\alpha^k))$ denotes a finite, zero-sum game among M_1^k and M_2^k in which the play $A^k = A_1^k \times A_2^k$ results in the payoff $J(A^k)$, where $J(A^k)$ is given by Shapley as

$$J(A^k) = R^k(A^k) + \sum_l T(k, A^k, l) \Pi^l$$

where Π^l represents the limiting reward values. We first state and prove some necessary lemmas.

Lemma 2. $\Gamma(k)$ has a unique equilibrium.

Proof: Follows directly from Theorems 1 and 2 of Shapley.

Here, the equilibrium could be in mixed strategies. Since we are going to compute a stationary Markov perfect equilibrium in pure strategies we need sufficient conditions that guarantee the existence of a stationary Markov equilibrium in pure strategies. If we make the assumption that each $\Gamma(k)$ has a unique pure strategy equilibrium that is a strict equilibrium then, by lemma 1 there is a stationary Markov perfect equilibrium in pure strategies if δ is small. The unique strict equilibrium is a saddle point of $\Gamma(k)$. The algorithm we present here works for the DZSGs where the saddle point exists in pure strategies.

Lemma 3. Saddle point of $\Gamma(k)$ represents optimal policy for the control of the zero-sum Markov chain.

Proof: We represent the controlled Markov chain using a game Γ , which is shown to have a unique equilibrium. We further assume that the equilibrium exists in pure strategies. If Γ were an automata game, then players using L_{R-I} learning scheme would be ϵ -optimal (based on [13], Theorem on Page 4). Thus each automata team acting within the state will converge to the saddle point of the game matrix D^k . However, a payoff in Γ is obtained asymptotically by using a fixed policy. Thus at each time step n , the Γ is approximated by $\Gamma(n)$. Since each decision maker uses L_{R-I} updating procedure, Γ becomes a limiting game $\Gamma = \lim_{n \rightarrow \infty} \Gamma(n)$.

The elements of $\Gamma(n)$, $d^k(A^k, n) = E[\beta(n) | \alpha(n) = A^k]$, depend on n . Thus the Zero-Sum Markov chain updating is not the same as the updating in the zero-sum automata game described in Lakshmivarahan-Narendra [13]. However, since we assume that the Markov chain is ergodic, it follows that for a sufficiently large n , the ordering among the $d^k(A^k, n)$ will be identical to that among the $J(A^k)$ in Γ . Therefore, it is sufficient to analyze the automata game Γ .

Based on these two lemmas, the main result can be stated as follows.

Theorem 1. Let an automaton M_i^k using L_{R-I} learning scheme, having A_i^k actions, be associated with each agent of an N -state DZSG. If the chain is ergodic, then for any $\epsilon > 0$, there exists an $0 < \lambda^* < 1$ such that for any $\lambda < \lambda^*$ in L_{R-I} learning scheme, $\lim_{n \rightarrow \infty} \Gamma(n) > J(A^k) - \epsilon$.

Proof: The proof follows immediately from lemma 2 and lemma 3.

VII. ADAPTIVE SHAPLEY RECURSION

The idea behind indirect adaptive methods is to estimate the unknown model parameters based on the data obtained during the execution of the model. For the DZSGs, the task is to estimate unknown transition and reward probabilities based on the history of transitions and immediate costs observed as the Markov chain goes through various transitions. The Adaptive Shapley Recursion (ASR) algorithm for *learning* the optimal strategies for a DZSG proceeds as follows. To initialize, the estimated model parameters of the DZSG are set to arbitrary values. At each time step t during the execution of the algorithm, we use Shapley recursion algorithm to determine optimal action values based on the latest estimated parameter values of the DZSG. Using *certainty equivalence principle*, we assume this to be optimal action of the DZSG and execute that action. The information obtained during the execution (reward values, action taken during transition etc.) of this action is further used to update the estimates of the model parameter and this process repeats.

The action based on certainty equivalence principle appears to be best based on information obtained upto time t and hence it is pursued. However, since the estimated model at time t may not be the same as actual model parameter values we are trying to estimate, we must occasionally also pursue actions that are different from the one given by the certainty equivalence principle. A simple way of doing it is by using randomized policies where actions are selected according to a probability distribution. We use ϵ -greedy action selection technique. Thus with probability ϵ , we select the optimal action as suggested by Shapley recursion and we select some other random action with probability $1 - \epsilon$. The objective of this strategy is to achieve a balance between exploration and optimization.

Following description explain the proposed ASR algorithm for learning optimal control policy for a DZSG.

- 1) For each state of the DZSG Markov chain, we estimate the state transition probabilities and reward values based on the past experience of state-transitions and rewards observed by the agents acting in that state.

- 2) On the basis of the rewards obtained for every transition, transition rewards estimates $\hat{R}_k(s_i, s_j, a_{lm})$ can be directly determined.
- 3) On the basis of the model parameters estimated at time step k (namely $\hat{P}_k(s_i, s_j, a_{lm})$ and $\hat{R}_k(s_i, s_j, a_{lm})$ time, we then execute Shapley recursion to obtain the optimal policy for given configuration of model parameters.
- 4) Once the Shapley recursion algorithm has converged, we use ϵ -greedy technique to choose an action tuple $a_{l'm'}^{s_i}$ in current state s_i .
- 5) Based on the selected action tuple, the DZSG estimate parameters (namely $\hat{P}$ and $\hat{R}$) are updated as explained in step 1 and the process repeats.

VIII. MINIMAX-TD

Temporal Difference (TD) learning combines the dynamic programming methodology with Monte Carlo paradigm. Like Monte Carlo methods, TD methods learn directly from online experience without a model of the dynamics of the environment. Like dynamic programming, temporal difference method updates estimates of the current state based on the previously learned estimates which are calculated during previous states of the system evolution. We propose a Temporal Difference (TD) algorithm to control the DZSG. It is called as Minimax-TD algorithm. Minimax-TD uses action tuples for the indices of the Q-matrix. The action tuple selection is done by calculating the *minimax* value in the Q-matrix. The Minimax-TD algorithm proceeds as follows:

- 1) Initialize $Q(s, \alpha_1^s, \alpha_2^s)$ arbitrarily $\forall s \in N$ and $\forall \alpha \in A_1 \times A_2$.
- 2) Arbitrarily select the starting state $curr \in N$ and call it the current state.
- 3) Repeat forever
 - a) Select (α_1^t, α_2^t) by performing $\minmax(Q(curr))$. Then select α_1^{curr} and α_2^{curr} by using ϵ -greedy technique.
 - b) Based on action tuple $(\alpha_1^{curr}, \alpha_2^{curr})$, transition to state $next \in N$ and observe the reward r .
 - c) Select $(\alpha_1^{t'}, \alpha_2^{t'})$ by performing $\minmax(Q(next))$. Then select α_1^{next} and α_2^{next} by using ϵ -greedy technique.
 - d) Calculate
$$Q(curr, \alpha_1^{curr}, \alpha_2^{curr}) = Q(curr, \alpha_1^{curr}, \alpha_2^{curr}) + \delta[r + Q(next, \alpha_1^{next}, \alpha_2^{next}) - Q(c, \alpha_1^{curr}, \alpha_2^{curr})]$$
 - e) Set $curr = next$ and goto step 1.

IX. SIMULATION RESULTS

In this section, we will present simulation results for different algorithms discussed in this paper, i.e., HEGLA Based Algorithm, Adaptive Shapley Recursion Algorithm, and Minimax-TD algorithm. We simulated these algorithms on a 2-state DZSG. Suppose the states of the DZSG are numbered 1 and 2 and the actions for each agent are also numbered 1 and 2. The transition probabilities of the DZSG are summarized below:

$$\begin{aligned}
t^1 &= \{[1, \{1, 1\}] = 0.5; [2, \{1, 1\}] = 0.5; [1, \{1, 2\}] = 0.5; [2, \{1, 2\}] = 0.5; [1, \{2, 1\}] = 0.5; [2, \{2, 1\}] = 0.5; [1, \{2, 2\}] = 0.5; [2, \{2, 2\}] = 0.5\} \\
t^2 &= \{[1, \{1, 1\}] = 0.5; [2, \{1, 1\}] = 0.5; [1, \{1, 2\}] = 0.5; [2, \{1, 2\}] = 0.5; [1, \{2, 1\}] = 0.5; [2, \{2, 1\}] = 0.5; [1, \{2, 2\}] = 0.5; [2, \{2, 2\}] = 0.5\}
\end{aligned}$$

Here t^i represent transition function for the i -th state of the Markov chain. The entry $t^i = [j, \{k, l\}]$ gives the transition probability for the transition from state i to state j when the action tuple $\{k, l\}$ is selected (i.e. when one LA selects action k and the other selects action l).

The reward probabilities are as follows:

$$\begin{aligned}
r^1 &= \{[\{1, 1\}] = 0.1; [\{1, 2\}] = 0.2; [\{2, 1\}] = 0.3; [\{2, 2\}] = 0.4; \} \\
r^2 &= \{[\{1, 1\}] = 0.5; [\{1, 2\}] = 0.6; [\{2, 1\}] = 0.7; [\{2, 2\}] = 0.8; \}
\end{aligned}$$

Here r^i represent reward function for the i -th state of the Markov chain. The entry $r^i = [\{k, l\}]$ gives the reward probability for the state i when the action tuple $\{k, l\}$ is selected (i.e. when one LA selects action k and the other selects action l).

The action tuple $\{2, 1, 2, 1\}$ is the optimal action tuple for this particular DZSG. There are four agents in the system (two states and two agents per state means $2 \times 2 = 4$ agents). First two actions of the optimal action tuple give the optimal actions for the first two agents in state 1 and the last two correspond to the two agents in state 2.

For the HEGLA-based DZSG control algorithm, the convergence value was set as 0.95. Whenever action probability value of any action belonging to a LA reaches 0.95, that automaton is termed as converged. If a system has multiple learning automata (as is the case with the heterogeneous game algorithm presented in this paper), when all the individual automaton in the system converge, the whole system is regarded as a converged system and the algorithm execution stops. In the Figures (1) and (2), we show the evolution of action probabilities of the Maximum (Row) and Minimum (Column) automaton that reside in the first state of the DZSG. As indicated in the figure, the Row player converges to action 2 and Column player converges to action 1. These are the optimal policies for these two player. This demonstrates the stochastic and temporal behavior of the LA based DZSG control algorithm.

Fig. 1. Evolution of Action Probabilities for the Maximum (Row) Automaton in A 2-state DZSG

For the Minimax-TD algorithm, the Figure (3) demonstrates the behavior of the Q values of the for the agents in

Fig. 2. Evolution of Action Probabilities for the Minimum (Column) Automaton In A 2-state DZSG

the first state of the DZSG. The figure depicts the variation in Q-values for a 4000 iteration size window during the execution of the algorithm. As it is evident from the figure, the Q-value for action $\{2, 1\}$ represent the *minimax* value in the matrix formed by the four action tuples. Thus the algorithm will select the $\{2, 1\}$ action tuple for the control of the DZSG.

Fig. 3. Evolution of the Q-values for the Minimax-TD algorithms In A 2-state DZSG

Figure (4) shows the behavior of the Shapley recursion algorithm for the first 20 iterations. As demonstrated in the figure, all the matrix values always increase monotonically over the execution of the algorithm. However, at this scale and resolution, we can clearly see how the individual *A* matrix (value matrix) entries are ordered with respect to each other. It is clear that the *minimax* value in the *A* matrix resides in the entry $\{2, 1\}$.

X. CONCLUSIONS

The simulations presented in the previous section show that the HEGLA approach is applicable to DZSG where optimal control policy exists in the pure strategies. If the optimal policies exist in mixed strategies, other learning automata algorithms (e.g. L_{R-P} algorithm or some modification of L_{R-I} algorithm) might be useful. However, this particular issue needs more investigation. The TD-minimax algorithm uses the *minimax* operator which finds optimal policies in pure strategies. However, it will be straightforward to extend this technique to include cases where optimal policy exists in mixed strategies. We can use linear programming technique used in [14] for learning mixed strategies under TD-minimax formalism. The use of linear programming in learning algorithm can lead to large computational complexity and thus slow convergence speed. However, it might be possible to use

Fig. 4. The value matrix (*A* matrix) entries for the Shapley recursion shown at a higher resolution.

approximate algorithms to find linear solutions which might be sufficient to obtain optimal convergence.

REFERENCES

- [1] L. S. Shapley. Stochastic games. *Proc Natl Acad Sci*, 39:1095–1100, 1953.
- [2] C. E. Lemke and J. T. Howson, Jr. Equilibrium points of bimatrix games. *Journal of the Society for Industrial and Applied Mathematics*, 12(2), pp.413–423, 1964.
- [3] L. S. Shapley. Some topics in two-person games. *Advances in game theory*, 52(1–29), pp.1–2, 1964.
- [4] A. Greenwald, A. and M. L. Littman. Introduction to the special issue on learning and computational game theory. *Machine Learning*, 67(1), pp.3–6, 2007.
- [5] M. L. Littman. Value-function reinforcement learning in markov games. *Journal of Cognitive Systems Research*, 2:55 – 66, 2001.
- [6] R. Carosi, M. Flammini and G. Monaco. Computing Approximate Pure Nash Equilibria in Digraph k-Coloring Games. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pp. 911–919. International Foundation for Autonomous Agents and Multiagent Systems, 2017.
- [7] M. Tambe, A. X. Jiang, B. An, and M. Jain, M. Computational game theory for security: Progress and challenges. In *AAAI spring symposium on applied computational game theory*, 2014.
- [8] J. W. Huang, H. Mansour, and V. Krishnamurthy, V. A dynamical games approach to transmission-rate adaptation in multimedia WLAN. *IEEE Transactions on Signal Processing*, 58(7), pp.3635–3646, 2010.
- [9] T. Sandholm. Steering evolution strategically: Computational game theory and opponent exploitation for treatment planning, drug design, and synthetic biology. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.
- [10] R. M. Wheeler and K. S. Narendra. Decentralized Learning in Finite Markov Chains. *IEEE Trans. on Automatic Control*, vol. 31, pages 519 – 526, 1986.
- [11] K. S. Narendra and M. A. L. Thathachar. *Learning Automata: An Introduction*. Prentice-Hall, 1989.
- [12] O. Tilak and S. Mukhopadhyay. Partially decentralized reinforcement learning in finite, multi-agent markov chains. *AI Communications*, 24(4), 293–309, 2011.
- [13] S. Lakshmivarahan and K. S. Narendra. Learning algorithms for two-person zero-sum stochastic games with incomplete information: A unified approach. *Control and Optimization*, 20:541–552, 1982.
- [14] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. *Eleventh International Conference on Machine Learning*, pages 157–163, 1994.
- [15] O. Tilak, R. Martin, and S. Mukhopadhyay. A decentralized indirect method for learning automata games. *IEEE Transactions on Systems, Man., and Cybernetics B*, 41(5), 1213–1223, 2011.