DISTRIBUTED GAMES FOR MULTI-AGENT SYSTEMS:
GAMES ON COMMUNICATION GRAPHS

Kyriakos G. Vamvoudakis, Dariusz G. Mikulski, Greg R. Hudas, Frank L. Lewis, Edward Y. Gu

ABSTRACT

Multi-agent systems arise in several domains of engineering and they can be used to solve problems which are difficult for an individual agent to solve. Examples of multi-agent systems include disaster response, online trading, and modeling social structures. Strategies for team decision problems, including optimal control, N-player games (H-infinity control, non zero sum), and so on are normally solved for off-line by solving associated matrix equations such as the coupled Riccati equations or coupled Hamilton-Jacobi equations. However, using that approach, players cannot change their objectives online in real time without calling for a completely new off-line solution for the new strategies. Therefore, in this paper we present an online gaming algorithm based on policy iteration to solve the continuous-time (CT) distributed multi-agent games for cooperative systems with infinite horizon cost with known dynamics. That is, the algorithm learns online in real-time the solution to the game design cooperative coupled HJ equations. This allows for truly dynamical team decisions where objective functions can change in real time and the system dynamics can be time-varying.

1. INTRODUCTION

Multi-agent systems describe a class of systems comprised of autonomous agents that cooperate to meet a system level objective. The most important property of multi-agent systems is their autonomy based only on local information exchanges with neighboring agents. (Schneider et al., 2008) has identified a variety of multi-agent application areas in military domains, of which include: reconnaissance, surveillance and target acquisition (RSTA); ordnance disposal; security, defense and sniper discovery; transport, convoying, and rescue; and sensor and communication networks.

This work was supported by ARO grant W91NF-05-1-0314, AFOSR grant FA9550-09-1-0278, and NSF grant ECCS-0801330. Kyriakos G. Vamvoudakis (corresponding author) and Frank L. Lewis are with Automation and Robotics Research Institute, University of Texas at Arlington, 7300 Jack Newell Blvd. S., Fort Worth, TX 76118 USA, phone/fax: +817-272-5938; e-mail: {kyriakos@arri.uta.edu, lewis@uta.edu}, Dariusz G. Mikulski and Edward Y. Gu are with Oakland University, MI, USA, email: {dgmikuls@oakland.edu, guy@oakland.edu}; Greg Hudus is with the U.S. Army RDECOM-TARDEC, Joint Center for Robotics (JCR), Warren, MI, USA, email: {greg.hudas@us.army.mil}.

In certain cases, game theory (Tijs, 2003) has been successfully used to model the strategic behavior in multi-agent systems, where the outcome for each agent depends not only on their own actions, but also the actions of every other agent. Every agent chooses a control strategy to independently optimize his own performance objective without the knowledge of other agent strategies. Many applications of optimization theory require the solution of coupled Hamilton-Jacobi equations (Başar and Olsder, 1999; Freiling et al., 2002).

In the past, game theory has also been used to address tactical needs and develop strategic direction in various military situations, such as in two-player zero-sum games (Haywood, 1954), ballistic defense (Palmore et al., 2001), and civilian-military peacetime operations (Mockaitis, 2004). Currently, the US military wages asymmetric battles against insurgenecies, where enemy combatants have exhibited quick and deadly adaptations to US strategies (Singer, 2009). Because of these adaptations, heterogeneous teams consisting of humans, ground sensors, and unmanned system must also be able to adapt (preferably in real-time) and learn optimal game strategies, even under changing mission requirements and team objectives.

Strategies for team decision problems, including optimal control, N-player games (non zero sum, zero-sum), and so on are normally solved for off-line by solving the coupled Hamilton Jacobi equations for nonlinear systems or coupled Riccati equations (Freiling et al., 2002; Gajic and Li, 1988) for linear systems. However, using that approach, players cannot change their objectives online in real time without calling for a completely new off-line solution for the new strategies. Therefore, in this paper methods are given for solving different team decision problems online in real time by observing data along the system trajectories. This provides a truly dynamic framework for team decision-making, since players or teams can change their objectives or optimality criteria on the fly, and the new strategies for all players appropriate to the new situation can be re-computed in real-time. In other words, if we can solve games online in real-time, then the performance objectives can be modified in real-time as team and individual objectives change by modifying the performance functions/costs. This approach also allows for time-varying team dynamics.
Distributed networks have received much attention in the last year because of their flexibility and computational performance. The ability to coordinate agents is very important to many real-world tasks. It is necessary for agents to exchange information with each other. Synchronization behavior among agents is found in flocking of birds, schooling of fish, and other natural systems. Much work has extended consensus and synchronization techniques to manmade systems such as UAV to perform various tasks including surveillance, moving in formation, etc. Consensus has been studied for systems on communication graphs with fixed or varying topologies and communication delays.

In such systems, deriving the exact mathematical models for creation of belief model of neighboring agents is difficult. One solution should be the use of approximate models, but they can drive the system into sub-optimal solutions because of the inherent uncertainty associated with the local data. The use of reinforcement learning techniques will avoid such problems.

Reinforcement learning (RL) is a sub-area of machine learning concerned with how to methodically modify the actions of an agent (player) based on observed responses from its environment (Sutton and Barto, 1998). In game theory, reinforcement learning is considered as a bounded rational interpretation of how equilibrium may arise.

RL methods offer many advantages that have motivated control systems researchers to develop RL algorithms which result in optimal feedback controllers for dynamic systems that are described by difference or ordinary differential equations. These involve a computational intelligence technique known as Policy Iteration (PI) (Bertsekas and Tsitsiklis, 1996; Werbos, 1974, 1992), which refers to a class of two step iteration algorithms: policy evaluation and policy improvement. PI provides effective means of learning solutions to HJ equations online. In control theoretic terms, the PI algorithm amounts to learning the solution to a nonlinear Lyapunov equation, and then updating the policy through minimizing a Hamiltonian function. Online reinforcement learning techniques have been developed for continuous-time systems in (Vamvoudakis and Lewis, 2010; Vrabie et. al., 2009).

This paper proposes an online algorithm for linear continuous-time systems with known dynamics to solve the multi-agent distributed-game problem where each player wants to optimize his own performance index (Başar and Olsder, 1999) that depends on himself and his neighbors. The number of parametric approximator structures that are used is $2N$. Each player maintains a critic approximator neural network (NN) to learn his optimal value and a control actor NN to learn his optimal control policy.

### 2. SYNCHRONIZATION AND NODE ERROR DYNAMICS

#### 2.1 Graphs

Consider a graph $Gr = (V, E)$ with a nonempty finite set of $N$ agents (nodes) $V = \{v_1, \ldots, v_N\}$ and a set of edges or arcs $E \subseteq V \times V$. We assume the graph is simple, e.g. no repeated edges and $(v_i, v_j) \notin E, \forall i$ no self loops. General directed graphs are considered. Denote the connectivity matrix as $E = [e_{ij}]$ with $e_{ij} > 0$ if $(v_j, v_i) \in E$ and $e_{ij} = 0$ otherwise. Note $e_{ii} = 0$.

The set of neighbors of an agent (node) $v_i$ is $N_i = \{v_j : (v_j, v_i) \in E\}$, i.e. the set of agents (nodes) with arcs incoming to $v_i$. Define the in-degree matrix as a diagonal matrix $D = [d_i]$ with $d_i = \sum_{j \in N_i} e_{ij}$ the weighted in-degree of node $i$ (i.e. $i$-th row sum of $E$). Define the graph Laplacian matrix as $L = D - E$, which has all row sums equal to zero.

Note that the graph represents the communication topology of the team. We assume the communication digraph is strongly connected, i.e. there is a directed path from $v_i$ to $v_j$ for all distinct nodes $v_i, v_j \in V$. Then (and only then) $E$ and $L$ are irreducible (Qu, 2009). That is they are not cogredient to a lower triangular matrix, i.e., there is no permutation matrix $U$ such that

$$L = U \begin{bmatrix} * & 0 \\ * & * \end{bmatrix} U^T$$

The results of this paper can easily be extended to graphs having a spanning tree (i.e. not necessarily strongly connected) using the Frobenius form (Abou-Kandil et. al., 2003).

#### 2.2 Synchronization and Node Error Dynamics

Consider the $N$ dynamic agents distributed on communication graph $Gr$ with dynamics

$$\dot{x}_i = Ax_i + B_i u_i$$

where $x_i(t) \in \mathbb{R}^n$ is the state of node $i$, $u_i(t) \in \mathbb{R}^m_i$ its control input, and $A(t) \in \mathbb{R}^{m \times n}, B_i \in \mathbb{R}^{m \times m}$. The control objective is to drive the system to the equilibrium point.

The dynamics of each agent (node) can describe the motion of a robot, unmanned autonomous vehicle, or missile that together with his neighbor agents satisfy a performance objective. For example, consider a swarm of unmanned autonomous ground vehicles, transmitting vital
intelligence, like the enemy position, strengths, and assets back to the military personnel, and also another swarm of robotic air vehicles providing aerial surveillance images to augment the intelligence collected by the ground vehicles.

Cooperative team objectives may be prescribed in terms of the local neighborhood tracking error \( \delta_i \in \mathbb{R}^n \) (Khoo et al., 2009) as

\[
\delta_i = \sum_{j \in N_i} e_{ij}(x_i - x_j) + g_i(x_i - x_0) \tag{3}
\]

The pinning gain \( g_i \geq 0 \) is nonzero for a small number of nodes \( i \) that are coupled directly to the leader or control node \( x_0 \), and \( g_i > 0 \) for at least one \( i \). We refer to the nodes \( i \) for which \( g_i \neq 0 \) as the pinned or controlled nodes. Note that \( \delta_i \) represents the information available to node \( i \) for state feedback purposes as dictated by the graph structure.

The state of the control or target node is \( x_0(t) \in \mathbb{R}^n \) which satisfies the dynamics

\[
\dot{x}_0 = Ax_0 \tag{4}
\]

Note that this is in fact a command generator (Lewis, 1992) and we seek to design a cooperative control command generator tracker.

The synchronization control design problem is a distributed multiagent game problem that seeks to design local control protocols for all the nodes in \( G_r \) to synchronize to the state of the control node, i.e. one requires \( x_i(t) \rightarrow x_0(t), \forall i \).

To find the dynamics of the local neighborhood tracking error, write

\[
\dot{\delta}_i = \sum_{j \in N_i} e_{ij}(\dot{x}_i - \dot{x}_j) + g_i(\dot{x}_i - \dot{x}_0) 
\]

which can also be written as

\[
\dot{\delta}_i = A\delta_i + (d_i + g_i)B_iu_i - \sum_{j \in N_i} e_{ij}B_ju_j \tag{5}
\]

This is a dynamical system with multiple control inputs, from node \( i \) and all of its neighbors, networked over a communication graph topology.

### 3. COOPERATIVE OPTIMAL CONTROL: MULTI-AGENT GAMES ON GRAPHS

#### 3.1 Local Cost Functions for Team Behaviors in Multi-Agent Games

We wish to achieve synchronization while simultaneously optimizing some performance specifications on the agents. To capture this, we intend to use the machinery of multi-player games ( Başar and Olsder, 1999). Therefore, define the local performance objectives

\[
J_i(\delta_i(0),u_i,u_{-i}) = \int_0^\infty (\delta_i^TQ_i\delta_i + u_i^TR_iu_i + \sum_{j \in N_i} u_j^TR_ju_j) dt 
\]

\[
= \int_0^\infty L_i(\delta_i(t),u_i(t),u_{-i}(t)) dt \tag{6}
\]

where \( u_{-i}(t) \) is the vector of the control inputs \( \{u_j : j \in N_i\} \) of the neighbors of node \( i \). All weighting matrices are constant and symmetric with \( Q_i \geq 0, R_i > 0, R_{ij} \geq 0 \).

The dynamics (2) are not coupled but the performance objective of synchronizing the activities of the team by making (3) small has resulted in coupled error performance dynamics (6).

Note that the performance function includes only the information from node \( i \) and its neighbors.

Interpreting now the control input as policies or strategies \( u_i(x_i), u_j(x_j) \), e.g. as feedback maps from \( \mathbb{R}^n \) to \( \mathbb{R}^m \), the value of the objective function for node \( i \) corresponding to those policies is

\[
V_i(\delta_i(t),\delta_{-i}(t)) = \int_0^\infty (\delta_i^TQ_i\delta_i + u_i^TR_iu_i + \sum_{j \in N_i} u_j^TR_ju_j) dt \tag{7}
\]

where \( \delta_{-i} \) is the vector of the states \( \{\delta_j : j \in N_i\} \) of the neighbors of node \( i \). Denote \( \delta_i(t) = [\delta_i(t) \ \delta_{-i}(t)] \).

#### 3.2 Local Hamiltonian Functions and Nash Equilibrium

When \( V_i(\delta_i(t)) \) is finite, using Leibniz’ formula, a differential equivalent to this is given in terms of the Hamiltonian function as

\[
\dot{H}_i(\delta_i(t),\delta_{-i}(t)) = \sum_{j \in N_i} e_{ij}B_ju_j + g_i(x_i - x_0) 
\]

\[
\dot{\delta}_i = A\delta_i + (d_i + g_i)B_iu_i - \sum_{j \in N_i} e_{ij}B_ju_j 
\]

\[
\dot{\delta}_{-i} = A\delta_{-i} + (d_{-i} + g_{-i})B_{-i}u_{-i} - \sum_{j \in N_{-i}} e_{ij}B_ju_j 
\]
\[ H_i(\delta_i, u_i, u_{-i}) = \frac{\partial V_i^T}{\partial \delta_i} \left( A \delta_i + (d_i + g_i) B_i u_i + \sum_{j \in N_i} e_{ij} B_j u_j \right) \]

\[ + \delta_i^T Q_i \delta_i + u_i^T R_i u_i + \sum_{j \in N_i} u_j^T R_j u_j = 0 \]  

(8)

with boundary condition \( V_i(0) = 0 \). (The gradient is disabused here as a column vector.) That is, solution of equation (8) serves as an alternative to evaluating the current feedback policies. We refer to this equation, somewhat incorrectly, as a nonlinear Lyapunov equation.

The control objective of agent \( i \) is to determine

\[ V_i^*(\delta_i(t)) = \min_{u_i} \int_0^\infty \left( \delta_i^T Q_i \delta_i + u_i^T R_i u_i + \sum_{j \in N_i} u_j^T R_j u_j \right) dt \]  

(9)

i.e. to find the optimal policy and value for node \( i \). In view of the fact that each node has this same objective with regards to its own policy and value, this is an \( N \)-player game (Başar and Olsder, 1999).

**Definition 1.** An \( N \)-tuple of strategies \( \{\mu_i^*, \mu_2^*, ..., \mu_N^*\} \) is said to constitute a Nash equilibrium solution for an \( N \)-player distributed finite game in extensive form, if the following \( N \) inequalities are satisfied for all \( \mu_i^*, i \in N \)

\[ J_i^* \triangleq J_i(\mu_i^*, \mu_{-i}^*) \leq J_i(\mu_i, \mu_{-i}), i \in N \]  

(10)

where \( \mu_{-i}^* \) is the vector of all the neighborhood nodes (Abou-Kandil et al., 2003).

The \( N \)-tuple of quantities \( \{J_1^*, J_2^*, ..., J_N^*\} \) is known as a Nash equilibrium outcome of the \( N \)-player distributed game.

Employing the stationarity condition \( \partial H_i / \partial u_i = 0 \) (Lewis and Syrmos, 1995), one obtains the control policies

\[ u_i = -\frac{1}{2} (d_i + g_i) R_i^{-1} B_i^T \frac{\partial V_i}{\partial \delta_i} = -h_i(\delta_i) \]  

(11)

Substituting into (8) yields the distributed game Hamilton-Jacobi (HJ) equation,

\[ \frac{\partial V_i^T}{\partial \delta_i} A_i^c + \delta_i^T Q_i \delta_i + \frac{1}{2} (d_i + g_i) B_i^T R_i^{-1} B_i^T \frac{\partial V_i}{\partial \delta_i} + \frac{1}{2} \sum_{j \in N_i} (d_j + g_j) B_j R_j^{-1} B_j^T \frac{\partial V_i}{\partial \delta_j} = 0 \]  

(12)

where the closed-loop matrix is

\[ A_i^c = A_i^c - \frac{1}{2} (d_i + g_i) B_i R_i^{-1} B_i^T \frac{\partial V_i}{\partial \delta_i} + \frac{1}{2} \sum_{j \in N_i} e_{ij} (d_j + g_j) B_j R_j^{-1} B_j^T \frac{\partial V_j}{\partial \delta_j} \]  

(13)

There is one coupled HJ equation corresponding to each agent (node), so solution of this multi-agent game problem is blocked by requiring a solution to \( N \) coupled partial differential equations. In the next section we show how to solve this multi-agent cooperative game online in a distributed fashion at each node, requiring only measurements from neighbor agents (nodes), by using techniques from reinforcement learning.

### 3.3 Team and Individual Performance Objectives: Team Interest vs. Self Interest

To understand teamwork in military scenarios, (Chandler and Pachter, 2009) differentiate between three different degrees of team interaction: coordination, cooperation, and collaboration. The spectrum between coordination, cooperation, and collaboration is a continuum that depends on the closeness of the objective functions of individual players. Coordination is considered the strongest degree since members share a single team objective and strive to optimize a single payoff function. There is no conflict of interest (i.e. members having different objective functions). Cooperation is slightly weaker than coordination, as members each have private objective functions to optimize in addition to team objectives. Conflict of interest is possible, but generally not dominant. Collaboration is even weaker than cooperation since members try to maximize their own individual objectives to be consistent with team tasks while avoiding tasking conflicts. Conflicts are often resolved with negotiation protocols - or a member may simply opt out of a team or join another team since team participation is not mandatory.

Different degrees of team interaction are appropriate for different military situations – and these examples are only offered to illustrate this point. Military vehicle formations and convoys represent scenarios where all team members are obligated to participate and are bound to all assignments, tasks, or agreements. The behavior of
an Army medic and a wounded Soldier, however, can be modeled by cooperative solutions, since both have different private objective functions, but share a team objective to move to a safe zone. A Predator drone used for intelligence, surveillance, and reconnaissance (ISR) is more likely to exhibit collaborative behaviors since it may simultaneously belong to different teams and assist in a team’s tasks when there are no conflicts of interest (such as being low on fuel).

The objective functions of each agent can be written as a team average term plus a conflict of interest term

\[
J_i = \frac{1}{N} \sum_{j=1}^{N} J_j + \frac{1}{N} \sum_{j=1}^{N} (J_i - J_j) \equiv J_{\text{team}} + J_i^{\text{conflict}}
\]

(14)

where \(J_{\text{team}}\) is the overall performance objective of the networked team and \(J_i^{\text{conflict}}\) is the conflict of interest or competitive objective. \(J_{\text{team}}\) measures how much the agents are vested in common goals, and \(J_i^{\text{conflict}}\) expresses to what extent their objectives differ.

The objective functions can be chosen by the individual agents, or they may be assigned to yield some desired team behavior. It should be noted that the decomposition of (14) is closely related to the altruistic and competitive contributions of a team, which are used to determine marginal contribution of team in a fully-cooperative and cohesive game (Arney and Peterson, 2008). These altruistic and competitive contributions are calculated using payoff (utility) functions, whose results could be incorporated in to the \(J_{\text{team}}\) and \(J_i^{\text{conflict}}\) terms, respectively.

4. DISTRIBUTED SOLUTION OF THE N-AGENT GAME USING REINFORCEMENT LEARNING

4.1 Reinforcement Learning Solution for Multi-Agent Distributed Games

To solve the multi-agent game in a distributed fashion at each node by a practically implementable means, the value functions (7) must be parameterized. In the standard linear quadratic \(N\)-player game ( Başar and Olsder, 1999), there is only one system dynamics, having \(N\) control inputs. That is, the problem is in fact centralized. Then, the system of coupled HJ equations (12) becomes a set of coupled (generalized or game) Riccati equations.

In the multi-agent scenario in this paper, one has \(N\) dynamical systems (2) and it is not at all clear what parametric form the value needs to take in the Hamiltonian (8). It does need to be in terms only of local variables to admit a local solution procedure.

Reinforcement learning (RL) techniques have been used to solve the optimal control problem online using adaptive learning techniques to determine the optimal value function. RL has been well developed for discrete-time systems.

An iterative offline solution technique is given by the following policy iteration algorithm:

Algorithm 1. Policy Iteration Solution for multi-agent distributed games

1. Start with stabilizing initial policies \(u_1^0(x), \ldots, u_N^0(x)\).

2. Given the \(N\)-tuple of policies \(\mu^k_1(x), \ldots, \mu^k_N(x)\), solve for the \(N\)-tuple of costs \(V^k_1(x(t)), V^k_2(x(t))\ldots V^k_N(x(t))\) using:

\[
0 = \delta_t^T Q \delta_t + \mu_i^T R_i \mu_i + \sum_{j \in N} u_i^T R_i u_j + \left( \frac{\partial V^k}{\partial \delta_i} \right)^T
\]

\[
\left( A \delta_t + (d_i + g_i) R_i u_i - \sum_{j \in N} e_{ij} B u_j \right), \quad V^k_i(0) = 0 \quad i \in N
\]

(15)

3. Update the \(N\)-tuple of control policies using:

\[
u_i^{k+1} = \arg \min_{u_i} [H_i(x, \nabla V_i, u_i, u_{-i})] \quad i \in N
\]

(16)

which explicitly is

\[
u_i^{k+1}(x) = -\frac{1}{2} (d_i + g_i) R_i^{-1} B_i^T \frac{\partial V^k_i}{\partial \delta_i} \quad i \in N
\]

(17)

4.2 Online Solution of Multi-Agent Cooperative Games using Neural Networks

In this subsection the main algorithm of the paper is presented. This algorithm provides update laws to tune the critic and the actor NNs simultaneously in real time.

The online solution of games in real time is tricky and cannot be implemented by simply throwing the standard NN structures and adaptation approaches. The first issue is that approximation is required of both the value functions and their gradients. This requires approximation in Sobolev space. Second, one requires learning of the values associated with the current control policies. Therefore, we first carefully develop proper approximator structures which lead to solution of the problem.
This paper uses nonlinear approximator structures for Value Function Approximation (VFA) (Bertsekas and Tsitsiklis, 1996; Vamvoudakis and Lewis, 2010; Werbos, 1974, 1992).

Assume there are neural network (NN) weights \( W_i \) such that the value function \( V_j(\overline{\delta}_j) \) is approximated as
\[
V_j(\overline{\delta}_j) = W^T_i \phi_j(\overline{\delta}_j) + e_j(\overline{\delta}_j)
\]
(18)
with \( \phi_j(\overline{\delta}_j) : \mathbb{R}^n \rightarrow \mathbb{R}^b \) the NN activation function vectors, \( h \) the number of neurons in the hidden layer, and \( e_j(\overline{\delta}_j) \) the NN approximation error.

Using the NN VFA considering fixed policies \( u_i \) the Hamiltonians become
\[
H_j(\delta_j, W_i, u_i, u_{-j}) = \delta_j^T Q_j \delta_j + u_i^T R_i u_i + \sum_{j \in N_i} u_j^T R_j u_j
+ W^T_i \frac{\partial \varphi_j}{\partial \delta_j} (A \delta_j + (d_i + g_i) B_i u_i - \sum_{j \in N_i} e_j B_j u_j) = e_{Hj}
\]
(19)

The effect of the approximation error in (12) is
\[
e_{Hj} = W^T_i \frac{\partial \varphi_j}{\partial \delta_j} A \delta_j + \delta_j^T Q_j \delta_j
- \frac{1}{2} (d_i + g_i)^T W^T_i \frac{\partial \varphi_j}{\partial \delta_j} B_i R_i^{-1} B_i^T \frac{\partial \varphi_j}{\partial \delta_j} W_i
+ \frac{1}{2} W^T_i \frac{\partial \varphi_j}{\partial \delta_j} \sum_{j \in N_i} e_j (d_j + g_j) B_j R_j^{-1} B_j^T \frac{\partial \varphi_j}{\partial \delta_j} W_j
+ \frac{1}{2} \sum_{j \in N_i} (d_j + g_j)^T W^T_j \frac{\partial \varphi_j}{\partial \delta_j} B_j R_j^{-1} B_j^T \frac{\partial \varphi_j}{\partial \delta_j} W_j, V_j(0) = 0
\]
(20)

**Assumption 1.** For each admissible control policy the nonlinear Lyapunov equations have smooth solutions \( V_j(\overline{\delta}_j) \geq 0 \).

Assuming current NN weight estimates \( \hat{W}_i \), the outputs of the critic NN are given by
\[
\hat{V}_j(\overline{\delta}_j) = \hat{W}^T_i \phi_j(\overline{\delta}_j)
\]
(21)
the approximate Lyapunov-like equations are:
\[
\hat{H}_j(\delta_j, \hat{W}_i, u_i, u_{-j}) = \delta_j^T Q_j \delta_j + u_i^T R_i u_i + \sum_{j \in N_i} u_j^T R_j u_j
+ \hat{W}^T_i \frac{\partial \varphi_j}{\partial \delta_j} (A \delta_j + (d_i + g_i) B_i u_i - \sum_{j \in N_i} e_j B_j u_j) = e_{Hj}
\]
(22)

It is desired to select \( \hat{W}_i \) to minimize the square residual error.
\[
E_i = \frac{1}{2} e_{Hi}^T e_{Hi}
\]
(23)

Then \( \hat{W}_i \rightarrow W_i \) and \( e_{Hi} \rightarrow e_{Hi} \).

Define the control policy in the form of an action neural network which computes the control input in the structured form
\[
\dot{u}_{i+n} = -\frac{1}{2} (d_i + g_i) R_i^{-1} B_i^T \frac{\partial \varphi_j}{\partial \delta_j} \hat{W}_{i+n}
\]
(24)
where \( \hat{W}_{i+n} \) denote the current estimated values of the ideal NN weights \( W_i \). Define the actor NN estimation errors as \( \tilde{W}_i = W_i - \hat{W}_i \) and \( \tilde{W}_{i+n} = W_i - \hat{W}_{i+n} \).

**Theorem 1.** Online Cooperative Games. Let the dynamics be given by (5), and consider the cooperative game formulation in (9). Let the critic NNs be given by (21), the control input be given by actor (agent \( i \)) NN (24). Let tuning for the \( i^{th} \) critic NN be provided by
\[
\dot{W}_i = -a_i \frac{\partial E_i}{\partial W_i}
\]
(25)

\[
= -a_i \frac{\sigma_i}{(1 + \sigma_i)} [\sigma_i^T \hat{W}_i + \sigma_i^T Q_i \delta_i + \frac{1}{2} \hat{W}_{i+n}^T \tilde{D}_i \hat{W}_{i+n} + \frac{1}{4} \sum_{j \in N_i} (d_j + g_j)^T \hat{W}_{j+n}^T \frac{\partial \varphi_j}{\partial \delta_j} B_j R_j^{-1} B_j^T \frac{\partial \varphi_j}{\partial \delta_j} \hat{W}_{j+n}]
\]

where \( \sigma_{i+n} = \frac{\partial \varphi_j}{\partial \delta_j} (A \delta_j + (d_j + g_j) B_j \hat{u}_{j+n} - \sum_{j \in N_i} e_j B_j \hat{u}_{j+n} ) \)
and assume \( \tilde{\sigma}_{i+n} = \sigma_{i+n} / (\sigma_{i+n}^T \sigma_{i+n} + 1) \) is persistently exciting.

Let the \( i^{th} \) actor NN be tuned as
\[
\dot{\hat{W}}_{i+n} = -\alpha_{i+n} ((F_{i+n} \hat{W}_{i+n} - F_{i+n} \hat{W}_i) - \frac{1}{4} \tilde{D}_{i+n} \hat{\sigma}_{i+n}^T m_i) \hat{W}_{i+n}
\]
(26)

where
\[
\tilde{D}_{i+n}(x) = \frac{\partial \varphi_j}{\partial \delta_j} B_j R_j^{-1} B_j^T \frac{\partial \varphi_j}{\partial \delta_j}, m_i = (\sigma_{i+n}^T \sigma_{i+n} + 1)
\]
and \( F_i > 0, \ldots F_{i+N} > 0 \) are tuning parameters. Assume that \( Q_d > 0 \) then the closed-loop system state, the critic NN errors \( \tilde{W}_i \), and the actor NN errors \( \tilde{W}_{i+N} \) are uniformly ultimately bounded.

**Proof:** Omitted due to space.

**Remark 1.** Theorem 1 provides the base of tuning the actor/critic networks of the \( N \) agents at the same time (meaning that the team learns online in real time), which is very important in military applications.

**Remark 2.** Persistence of excitation is needed for proper identification of the value functions by the critic NNs, and that nonstandard tuning algorithms are required for the actor NNs to guarantee stability. It is important to notice that the actor tuning law of every agent needs information of the critic of all his neighbors.

**Remark 3.** NN usage suggests starting with random, nonzero control NN weights in (24) in order to converge to the coupled Riccati solution. However, it should be noted that convergence is more sensitive to the persistence of excitation in the control inputs than to the NN weight initialization. If the proper persistence of excitation is not selected, the control weights may not converge to the correct values.

5. SIMULATIONS

This section will show the effectiveness of the ONLINE distributed synchronous policy iteration of Theorem 1. In these simulations, exponentially decreasing noise is added to the control inputs to ensure persistence of excitation until convergence is obtained.

Consider the three node communication graph shown in Figure 1. This graph simulates a military situation where a deployed unit (Node 2) can receive orders from a commander (Node 1), but does not have a transmitter strong enough to acknowledge the order directly. Thus, it must use a closer intermediate router (Node 3) to pass messages back to the commander.

The edge weights and the pinning gains in (5) were taking equal to 1.

For the graph structure shown above consider the following node dynamics

\[
\begin{align*}
\dot{x}_1 &= \begin{bmatrix} -1 & -2 \\ 1 & -4 \end{bmatrix} x_1 + \begin{bmatrix} 2 \\ -1 \end{bmatrix} u_1 \\
\dot{x}_2 &= \begin{bmatrix} -1 & -2 \\ 1 & -4 \end{bmatrix} x_2 + \begin{bmatrix} 1 \\ -3 \end{bmatrix} u_2 \\
\dot{x}_3 &= \begin{bmatrix} -1 & -2 \\ 1 & -4 \end{bmatrix} x_3 + \begin{bmatrix} 2 \\ 0 \end{bmatrix} u_3
\end{align*}
\]

Select \( Q_d, R_d \) and \( R_y \) in the online algorithm (Theorem 1) as identity matrices of appropriate dimensions. Also each agent maintains two neural networks, a critic to compute his cost and an actor to update his action. Figure 2 shows the convergence of local neighborhood \( \delta_1, \delta_2, \delta_3 \) to zero. Finally Figure 3 shows the convergence of the critic neural network weights for all the agents.

![Figure 1. Communication Graph.](image)

![Figure 2. Local neighborhood error for every agent.](image)

![Figure 3. Critic neural networks convergence for every agent.](image)
CONCLUSIONS

In this paper, an online gaming algorithm based on policy iteration to solve distributed multi agent games was presented. This method finds real-time approximations of the optimal value and the Nash equilibrium while guaranteeing closed-loop stability. This algorithm is implemented as an actor critic structure which involves simultaneous tuning of all critic and actor neural networks. Simulation results verify the theoretical results.

REFERENCES


Başar, T. and Olsder, G. J., 1999: Dynamic Noncooperative Game Theory, 2nd ed. SIAM.


