Multi-Agent Reinforcement Learning Approach Based on Reduced Value Function Approximations

Mohammed Abouheaf
School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, Canada
mohammed.abouheaf@uottawa.ca

Wail Gueaieb
School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, Canada
wail.gueaieb@uottawa.ca

***Abstract*—This paper introduces novel online adaptive Reinforcement Learning approach based on Policy Iteration for multi-agent systems interacting on graphs. The approach uses reduced value functions to solve the coupled Bellman and Hamilton-Jacobi-Bellman equations for multi-agent systems. This done using only partial knowledge about the agents' dynamics. The convergence of the approach is shown to depend on the properties of the communication graph. The Policy Iteration approach is implemented in real-time using neural networks, where reduced value functions are considered to reduce the computational complexity.**

Keywords—Cooperative Control, Graphs, Reinforcement Learning, Neural Networks Approximations.

I. INTRODUCTION

The large scale optimization problem for multi-agent systems suffers from the amount of the required computational effort, especially in the cases where the solutions of the optimization problems depend on finding matrix inverses of the solving structures. Full size value function structures based on coupled Bellman equations are developed for dynamic graphical systems in [1]-[2]. This motivated us to build reduced value function structure to solve the optimal control problem for multi-agent systems on graphs. This structure is based on the coupled Hamiltonian equations of the dynamic graphical systems. An online policy iteration algorithm is proposed to solve the optimal control problem using the data available to each agent. The introduced policy iteration approach is shown to have fast convergence properties. The implementation of the adaptive learning algorithm is done in real-time using the neural network structures. Unlike the least squares approaches, the developed approach doesn't require calculating the matrix inverses of the solving structures.

The cooperative control problems are divided into synchronization and consensus control problems [3]-[4]. In the synchronization problem, the agents select the policies to synchronize to the leader's dynamics. In the consensus control problem, the agents reach a common value or a common goal [5]-[7]. The Hamilton-Jacobi (HJ) equations are used to solve the optimal control problem by selecting the policies that will minimize the cost-to-go function [9]-[10].

Approximate Dynamic Programming (ADP) is used to solve the Dynamic Programming problems [12]-[15]. The optimal control problems are solved using adaptive learning techniques [13], [15]. The adaptive learning techniques are concerned with learning from interactions in dynamic environments [16], [17]. Value Iteration is used to solve the graphical game online in real-time [1]. Dual Heuristic Programming is used to solve the graphical games in [18]. Q-learning is used to solve the differential dynamic programming problem in [19]. Online Policy Iteration used ADP to solve Riccati equations in [20]. Policy Iteration solution for the adaptive optimal control problem can be obtained by relaxing the Hamilton-Jacobi-Bellman (HJB) equation to the equivalent optimization problem [21].

Actor-critic neural network structures are Temporal Difference methods with separate structures that explicitly represent the policies apart, from the value structures [22]. These structures involve forward-in-time algorithms for computing optimal decisions, that are implemented online in real-time. The actor component applies control policies to their environment, while the critic rewards some decisions and punishes other decisions [16]. Actor-critic neural network implementation for policy iteration is developed in [23].

The paper is organized as follows; Section 2, briefly introduces the graph structure and the consensus control problem. Section 3, shows the optimal control formulation of the problem. Section 4, describes the Policy Iteration algorithm along with its proof of convergence. Section 5, uses actor-critic neural networks to implement the online solution. Section 6, shows the simulation results for a graphical example.

II. COOPERATIVE CONTROL FOR MULTI-AGENT SYSTEMS

In this section, the mathematical setup of the synchronization problem is introduced for discrete-time multi-agent systems interacting on graphs. This setup will highlight the tracking error dynamics between the agents.

A. Communication Graph Structures

The communication graph G is a structure of a number of vertices ν and edges $\varepsilon \subseteq \{q, m \in \nu, m \neq q\}$. Herein, directed

graphs are considered $(q,m)\in\varepsilon$. The adjacency matrix $A=\left[a_{qm}\right]$ of the graph G is a matrix with nonzero elements $a_{qm}\neq 0\Leftrightarrow(m,q)\in\varepsilon$. The set of neighbors of node q can be defined as $N_q=\left\{m\in\nu:a_{qm}\neq 0\right\}=\left\{m\in\nu:(q,m)\in\varepsilon\right\}$.

B. Consensus and Graph Laplacian

The graph Laplacian L is defined for the cooperative control problem. A graph G of order N has adjacency matrix $A=[a_{qm}]$ and in-degree matrix $\Delta=\Delta(A)$ with diagonal in-degree elements $d_q=\sum_{m=1}^{N}a_{qm}$. The graph Laplacian is defined by $L=\Delta(A)-A$. The second eigenvalue $\lambda_2(L)$ of the graph Laplacian, determines how fast the multi-agent systems would converge to the leader's dynamics.

C. The Multi-Agent Systems Control Objectives

Each agent follows the dynamics

$$x_{i(k+1)}=Ax_{ik}+B_iu_{ik}, \tag{1}$$

where $x_{ik}\in R^n$ is the state of agent i, A and B_i are the agent's parameters, and $u_{ik}\in R^{m_i}$ is the control input for the agent.

The agents are required to follow a leader with the dynamics

$$x_{0(k+1)}=Ax_{0k}, \tag{2}$$

where $x_0\in R^n$ denotes the leader's state.

Thus, the objective of the synchronization problem is to select local voting protocols such that, the agents will follow the leader $\lim_{k\to\infty}\left\|x_{ik}-x_{0k}\right\|=0,\forall i$. In order to do that, pinning ideas are considered (the leader is exchanging information with few agents at least one) [24]. The tracking error $\varepsilon_{ik}\in R^n$ for agent i is designed to include local and team objectives such that

$$\varepsilon_{ik}=\sum_{j\in N_i}a_{ij}(x_{jk}-x_{ik})+g_i(x_{0k}-x_{ik}), \tag{3}$$

where g_i is the pinning gain of each agent i.

The pinning gain g_i is positive and exists when the agent exchanges information with the leader [25]. The synchronization error is shown to be bounded when the graph is strongly connected [24]. Thus, the local neighborhood tracking error dynamics for each agent i is given by [1]

$$\varepsilon_{i(k+1)}=A\varepsilon_{ik}-(g_i+d_i)B_iu_{ik}+\sum_{j\in N_i}a_{ij}B_ju_{jk}. \tag{4}$$

III. Formulation of the Optimal Control Problem

In this section, Bellman and Hamiltonian optimality equations are structured using forms of reduced value functions [1]-[10]. Finding the optimal policy will require solving the underlying coupled Hamilton-Jacobi-Bellman (HJB) equations. Furthermore, a form of Bellman equations that solves the Dual Heuristic Dynamic Programming is introduced.

A. Performance of the Multi-Agent Systems

Each agent is communicating to its neighbors, thus the associated optimal control problem needs to take into consideration the local and team objectives given by (4). The performance of each agent i is evaluated by the following performance index

$$J_i=\sum_{k=0}^{\infty}U_i(\varepsilon_{ik},u_{ik},u_{-ik}), \tag{5}$$

where $u_{-i}=\left\{u_j\mid j\in N_i\right\}$ are the policies of the neighbors and U_i is the cost function.

This performance index (5) reflects the local, team, and control objectives of each agent i. The cost function U_i is given by

$$U_i(\varepsilon_{ik},u_{ik},u_{-ik})=\tfrac{1}{2}(\varepsilon_{ik}^TQ_{ii}\varepsilon_{ik}+u_{ik}^TR_{ii}u_{ik}+\sum_{j\in N_i}u_{jk}^TR_{ij}u_{jk}), \tag{6}$$

where $Q_{ii}\geq 0\in R^{n\times n}$, $R_{ii}>0\in R^{m_i\times m_i}$, and $R_{ij}>0\in R^{m_j\times m_j}$ are the weighting matrices.

The value function for each agent i is given by

$$V_i(\bar{\varepsilon}_{ik})=\sum_{l=k}^{\infty}U_i(\varepsilon_{il},u_{il},u_{-il}), \tag{7}$$

where $\bar{\varepsilon}_{ik}$ is a local error vector of the states of each agent i ε_{ik} and its neighbors ε_{-ik}.

Herein, a reduced value function form is proposed for each agent i such that

$$V_i(\bar{\varepsilon}_{ik})=\frac{1}{2}\varepsilon_{ik}^T\Pi_i\bar{\varepsilon}_{ik}, \tag{8}$$

where $\Pi_i\in R^{n\times nN_i}$.

Remark 1: This value function enables a reduced solution frame work compared to that proposed in [1], [18]. This will lead to reduction in the computational complexity. Moreover, this will make it smoother, to implement algorithms like Policy Iteration using neural networks. This form utilizes the local information available to each agent i. ■

B. Coupled Bellman Optimality Equations

Equation (7), under the policy π, yields the discrete-time Bellman equation such that

$$V_i^{\pi}(\bar{\varepsilon}_{ik})=\tfrac{1}{2}(\varepsilon_{ik}^TQ_{ii}\varepsilon_{ik}+\pi_{ik}^TR_{ii}\pi_{ik}+\sum_{j\in N_i}\pi_{jk}^TR_{ij}\pi_{jk})+V_i^{\pi}(\bar{\varepsilon}_{i(k+1)}), \tag{9}$$

with initial values $V_i^{\pi}(\mathbf{0})=0$.

Applying Bellman optimality principle yields the optimal value function for each agent *i* such that

$$V_i^o(\bar{\varepsilon}_{ik}) = \min_{u_i}(V_i(\bar{\varepsilon}_{ik})) = \min_{u_i}(\sum_{l=k}^{\infty} U_i(\varepsilon_{il}, u_{il}, u_{-il})), \quad (10)$$

where $V_i^o(\bar{\varepsilon}_{ik})$ is the optimal value function for each agent *i*.

This results in the optimal control policy such that,

$$u_{ik}^o = (g_i + d_i)R_{ii}^{-1}B_i^T \nabla V_i^o(\bar{\varepsilon}_{i(k+1)}) = O_i \nabla V_i^o(\bar{\varepsilon}_{i(k+1)}). \quad (11)$$

where $\nabla V_i^{\pi}(\bar{\varepsilon}_{i(k+1)}) = \dfrac{\partial V_i^{\pi}(\bar{\varepsilon}_{i(k+1)})}{\partial \varepsilon_{i(k+1)}}$ and $O_i = (g_i + d_i)R_{ii}^{-1}B_i^T$.

Thus, the coupled Bellman optimality equation is given by

$$V_i^o(\bar{\varepsilon}_{ik}) = \frac{1}{2}(\varepsilon_{ik}^T Q_{ii}\varepsilon_{ik} + \nabla V_i^o(\bar{\varepsilon}_{i(k+1)})^T O_i^T R_{ii} O_i \nabla V_i^o(\bar{\varepsilon}_{i(k+1)}) + \sum_{j\in N_i} \nabla V_j^o(\bar{\varepsilon}_{j(k+1)})^T O_j^T R_{ij} O_j \nabla V_j^o(\bar{\varepsilon}_{j(k+1)})) + V_i^o(\bar{\varepsilon}_{i(k+1)}). \quad (12)$$

C. The Hamiltonian Functions

The Hamiltonian functions will be of great importance to implement the adaptive learning approach. Using (4) and (5), the Hamiltonian function for each agent *i* is given by

$$H_i(\bar{\varepsilon}_{ik}, \lambda_{i(k+1)}, u_{ik}, u_{-ik}) = \lambda_{i(k+1)}^T \bar{\varepsilon}_{i(k+1)} + U_i(\varepsilon_{ik}, u_{ik}, u_{-ik}). \quad (13)$$

where $\lambda_{i(k+1)}$ is the costate variable for each agent *i*.

The optimal policy is found by applying the stationarity condition $(\partial H_i / \partial u_{ik}) = 0$ such that

$$u_{ik}^* = \arg\min_{u_{ik}}(H_i^{\pi}(\bar{\varepsilon}_{ik}, \lambda_{i(k+1)}, u_{ik}, \pi_{-ik})). \quad (14)$$

Then,

$$u_{ik}^* = O_i \lambda_{i(k+1)}. \quad (15)$$

Remark 2: The relation between the coupled Hamiltonian functions and Bellman equations *(i.e. the Hamilton Jacobi (HJ) Theory)* explains the relation between the costate variable and the value function for each agent. Moreover, solving the optimal control problem requires, solving the underlying coupled Hamilton-Jacobi-Bellman (HJB) equations. ■

D. Hamilton-Jacobi-Bellman Equations

The Hamilton-Jacobi (HJ) theory relates the coupled Hamiltonian functions (13) and the coupled Bellman equations (9) [1],[11],[18]. The coupled Hamilton-Jacobi (HJ) equation for each agent *i* is found using the procedure in [1] such that

$$\Delta V_i^{\pi}(\bar{\varepsilon}_{ik}) - \nabla V_i^{\pi}(\bar{\varepsilon}_{i(k+1)})^T \bar{\varepsilon}_{i(k+1)} + H_i^{\pi}(\bar{\varepsilon}_{ik}, \nabla V_i^{\pi}(\bar{\varepsilon}_{i(k+1)}), u_{ik}, \pi_{-ik}) = 0, \quad (16)$$

where $\Delta V_i^{\pi}(\bar{\varepsilon}_{ik}) = V_i^{\pi}(\bar{\varepsilon}_{i(k+1)}) - V_i^{\pi}(\bar{\varepsilon}_{ik})$.

This equation relates the costate variable $\lambda_{i(k+1)}$ to the reduced value function $V_i(\bar{\varepsilon}_{i(k+1)})$ such that

$$\lambda_{i(k+1)} = \nabla V_i^{\pi}(\bar{\varepsilon}_{i(k+1)}). \quad (17)$$

The optimal polices resulting from the coupled Hamiltonian functions and the coupled Bellman optimality equations (11) and (15) are shown to be related [1], [18]. Equations (12), (14), and (16) with the optimal policies (11) result in a set of coupled Hamilton-Jacobi-Bellman (HJB) equations for each agent *i* such that

$$H_i(\bar{\varepsilon}_{ik}, \nabla V_i^o(\bar{\varepsilon}_{i(k+1)}), u_{ik}^o, u_{-ik}^o) = \nabla V_i^o(\bar{\varepsilon}_{i(k+1)})^T \bar{\varepsilon}_{i(k+1)} + U_i(\varepsilon_{ik}, u_{ik}^o, u_{-ik}^o) = 0, \; V_i^o(\mathbf{0}) = 0. \quad (18)$$

E. Bellman Optimality Equations Based on Costate Structures

Dual Heuristic Programming based on Value Iteration for cooperative control problems on graphs was proposed in [18]. The coupled Bellman optimality equations have been derived using the costate equations and the coupled Hamiltonian expressions. Thus, the costate-based Bellman optimality equation based on that frame work would look like

$$\varepsilon_{ik}^T \nabla V_i(\bar{\varepsilon}_{ik}) = \frac{1}{2}(\varepsilon_{ik}^T Q_{ii}\varepsilon_{ik} - u_{ik}^T R_{ii} u_{ik} - \sum_{j\in N_i} u_{jk}^T R_{ij} u_{jk}) + (g_i + d_i)\nabla V_i(\bar{\varepsilon}_{i(k+1)})^T B_i u_{ik} - \sum_{j\in N_i} e_{ij} \nabla V_i(\bar{\varepsilon}_{i(k+1)})^T B_j u_{jk}. \quad (19)$$

Remark 3: This expression was implemented using Value Iteration. Herein, a simplified approach based on reduced value functions is developed. A simplified Policy Iteration structure will be developed based on the Hamilton-Jacobi-Bellman (HJB) equation. ■

IV. Policy Iteration Solution

In the sequel, an online Policy Iteration algorithm is developed for multi-agent systems based on the reduced value function structures. This solution doesn't require the full knowledge of the agent's dynamics and solves simultaneously 'in real-time' the coupled Bellman optimality equations and the coupled Hamilton-Jacobi-Bellman (HJB) equations.

A. Policy Iteration Algorithm

Algorithm 1. Real-Time Policy Iteration Algorithm

1. Initialize the values $\Theta_i^0(\bar{\varepsilon}_{ik}), \forall i$ and the policies $u_{ik}^0, \forall i$ with admissible values.

2. Evaluate $\Theta_i^l(.), \forall i$ such that

$$\nabla \Theta_i^l(\bar{\varepsilon}_{i(k+1)}^{(u_{ik}^l, u_{-ik}^l)})^T \bar{\varepsilon}_{i(k+1)}^{(u_{ik}^l, u_{-ik}^l)} = -U_i(\varepsilon_{ik}, u_{ik}^l, u_{-ik}^l), \forall i. \quad (20)$$

3. Evaluate the policy $u_{ik}^{l+1}, \forall i$ such that

$$u_{ik}^{l+1} = (g_i + d_i)R_{ii}^{-1}B_i^T \nabla \Theta_i(\bar{\varepsilon}_{i(k+1)})^l, \forall i, \quad (21)$$

4. End when $\left\|\Theta_i^{l+1}-\Theta_i^{l}\right\|, \forall i$ is converging. ■

Definition 1: The control actions $\bar{u}_i = \{u_{ik}\}_{k=0}^{\infty}, \forall i \in N$ are said to be admissible if they stabilize (4) and guarantee that $V_i(\bar{\varepsilon}_{ik}), \forall i$ are finite. ■

Remark 4: The following theorem, shows the convergence of *Algorithm 1* when all agents update their policies simultaneously using (21). The convergence proof uses the reduced value function structures. Consequently, the chosen reduced value function structures will impact the structure of the optimal policy for each agent *i*. The graph connectivity properties will also impact the convergence of the proposed Policy Iteration algorithm. Assumptions about the connectivity of the graph will be made to maintain convergence. ■

Theorem 1. Let all agents use (12) or (20) and update their policies according to (21) simultaneously. Then

1) The policies $u_{ik}^{l}, \forall i, l > 0$ *are admissible, if the value* $\bar{\sigma}(R_{jj}^{-1}R_{ij})$ *is chosen small for each agent i.*

2) Algorithm 1 generates a sequence for each agent i $\Theta_{ik}^{*} \le . \le \Theta_{ik}^{l+1} \le \Theta_{ik}^{l}$*, such that* $\Omega_i^{*}, \forall i$ *are the solutions of (12) or (20).*

Proof: 1) The Hamilton-Jacobi (HJ) equation relates the costate variable to the reduced value function. The Hamilton-Jacobi-Bellman (HJB) and Bellman optimality equations yield

$$\nabla\Theta_i^{l}(\bar{\varepsilon}_{i(k+1)}^{(u_{ik}^{l},u_{-ik}^{l})})^{T}\,\bar{\varepsilon}_{i(k+1)}^{(u_{ik}^{l},u_{-ik}^{l})} < 0, \forall i, l, \tag{22}$$

and

$$\Theta_i^{l}(\bar{\varepsilon}_{i(k+1)}^{(u_{ik}^{l},u_{-ik}^{l})}) - \Theta_i^{l}(\bar{\varepsilon}_{ik}^{(u_{ik}^{l},u_{-ik}^{l})}) < 0, \forall i, l. \tag{23}$$

Thus, Θ_i^{l} works as Lyapunov function. The initial policies $u_{ik}^{0}, \forall i$ are admissible, then the values $\Theta_i^{l}, \forall i$ satisfy (7) or equivalently (20) such that

$$\Theta_i^{l}(\bar{\varepsilon}_{ik}^{(u_{ik}^{l},u_{-ik}^{l})}) > \Theta_i^{l}(\bar{\varepsilon}_{ik}^{(u_{ik}^{l+1},u_{-ik}^{l})}). \tag{24}$$

In order to fulfill the following inequality

$$\Theta_i^{l}(\bar{\varepsilon}_{ik}^{(u_{ik}^{l+1},u_{-ik}^{l})}) > \Theta_i^{l}(\bar{\varepsilon}_{ik}^{(u_{ik}^{l+1},u_{-ik}^{l+1})}), \tag{25}$$

an assumption can be made such that

$$\sum_{j\in N_i}\frac{1}{2}(u_{jk}^{l}-u_{jk}^{l+1})^{T}R_{ij}(u_{jk}^{l}-u_{jk}^{l+1}) - u_{jk}^{(l+1)T}R_{ij}(u_{jk}^{l+1}-u_{jk}^{l}) > 0. \tag{26}$$

Applying the norm on this inequality yields,

$$\frac{1}{2}\underline{\sigma}(R_{ij})\left\|\Delta u_{jk}^{l}\right\| > (g_j + d_j)\bar{\sigma}(R_{jj}^{-1}R_{ij})\left\|\nabla_j\Theta_j^{l}(\bar{\varepsilon}_{j(k+1)}^{u_{jk}^{l},u_{-jk}^{l}})\right\|\left\|B_{jk}\right\|, \tag{27}$$

where $\underline{\sigma}(M)$ and $\bar{\sigma}(M)$ are the minimum and maximum singular values of a matrix M and $\Delta u_{jk}^{l} = (u_{jk}^{l} - u_{jk}^{l+1})$.

Under the assumption of having large values for R_{ii} compared to R_{ij}, then the inequalities (24), (25), and (26) yield

$$\Theta_i^{l}(\bar{\varepsilon}_{ik}^{(u_{ik}^{l},u_{-ik}^{l})}) > \Theta_i^{l}(\bar{\varepsilon}_{ik}^{(u_{ik}^{l+1},u_{-ik}^{l})}) > \Theta_i^{l}(\bar{\varepsilon}_{ik}^{(u_{ik}^{l+1},u_{-ik}^{l+1})}). \tag{28}$$

Thus, the policies $u_{ik}^{l+1}, \forall i, l$ are stabilizing and admissible.

2) Since, the following inequality holds

$$\Theta_i^{l}(\bar{\varepsilon}_{i(k+1)}^{(u_{ik}^{l+1},u_{-ik}^{l+1})}) - \Theta_i^{l}(\bar{\varepsilon}_{ik}^{(u_{ik}^{l+1},u_{-ik}^{l+1})}) \le \Theta_i^{l+1}(\bar{\varepsilon}_{i(k+1)}^{(u_{ik}^{l+1},u_{-ik}^{l+1})}) - \Theta_i^{l+1}(\bar{\varepsilon}_{ik}^{(u_{ik}^{l+1},u_{-ik}^{l+1})}). \tag{29}$$

Applying infinite summation yields

$$\sum_{k=K}^{\infty}(\Theta_i^{l}(\bar{\varepsilon}_{i(k+1)}^{(u_{ik}^{l+1},u_{-ik}^{l+1})}) - \Theta_i^{l}(\bar{\varepsilon}_{ik}^{(u_{ik}^{l+1},u_{-ik}^{l+1})})) <$$
$$\sum_{k=K}^{\infty}(\Theta_i^{l+1}(\bar{\varepsilon}_{i(k+1)}^{(u_{ik}^{l+1},u_{-ik}^{l+1})}) - \Theta_i^{l+1}(\bar{\varepsilon}_{ik}^{(u_{ik}^{l+1},u_{-ik}^{l+1})})).$$

This inequality yields

$$\Theta_i^{l}(\bar{\varepsilon}_{i(\infty)}^{(u_{i\infty}^{l+1},u_{-i\infty}^{l+1})}) - \Theta_i^{l}(\bar{\varepsilon}_{iK}^{(u_{iK}^{l+1},u_{-iK}^{l+1})}) < \Theta_i^{l+1}(\bar{\varepsilon}_{i(\infty)}^{(u_{i\infty}^{l+1},u_{-i\infty}^{l+1})}) - \Theta_i^{l+1}(\bar{\varepsilon}_{iK}^{(u_{iK}^{l+1},u_{-iK}^{l+1})}).$$

Stability results in part *(1)* ensures that, $\Theta_i^{l}(\bar{\varepsilon}_{i(\infty)}^{(u_{i\infty}^{l+1},u_{-i\infty}^{l+1})}) \to 0$ and $\Theta_i^{l+1}(\bar{\varepsilon}_{i(\infty)}^{(u_{i\infty}^{l+1},u_{-i\infty}^{l+1})}) \to 0$ such that

$$\Theta_i^{l+1}(\bar{\varepsilon}_{iK}^{(u_{iK}^{l+1},u_{-iK}^{l+1})}) < \Theta_i^{l}(\bar{\varepsilon}_{iK}^{(u_{iK}^{l+1},u_{-iK}^{l+1})}). \tag{30}$$

This yields,

$$\Theta_i^{l+1} < \Theta_i^{l} < .. < \Theta_i^{0},\ \forall i, l. \tag{31}$$

This sequence is converging to a lower bound Θ_i^{*}, which is the solution to (12) or (18) such that

$$0 < .. < \Theta_i^{*} .. < \Theta_i^{l+1} < \Theta_i^{l} < .. < \Theta_i^{0},\ \forall i, l. \tag{32}$$ ■

Remark 5: The Policy Iteration implementation will be done using neural networks rather than using the traditional least square methods. This is a new prospective to use the neural networks with reduced value functions to implement the online Policy Iteration algorithms. ■

V. ACTOR-CRITIC NEURAL NETWORKS IMPLEMENTATION

In this section, an online implementation for Policy Iteration *Algorithm 1* in real-time is introduced. Two neural network structures namely critic and actor are used to approximate the reduced value functions (20) and the optimal policies (21) respectively. This is done simultaneously, using the local information available to each agent *i*. It is worth to note that, this implementation presents an easier alternative to the least square approach in terms of the stability and applicability.

The value function Θ_{ik} is approximated by

$$\hat{\Theta}_{ik}(.\,|\,\mho_i) = \frac{1}{2}\varepsilon_{ik}^{T}\mho_i^{T}\bar{\varepsilon}_{ik}, \tag{33}$$

where $\mho_i$ are the critic weights approximations for agent *i*.

The policy for agent *i* is approximated by

$$\hat{u}_i = \Phi_i\,\bar{\varepsilon}_{ik}, \tag{34}$$

where Φ_i are the policy approximations weights for agent *i*.

A. Critic Neural Networks Approximations

The value function $\hat{\Theta}_{ik}(.|\bar{\mho}_i)$ is restructured such that

$$\hat{\Theta}_{ik}(.|\bar{\mho}_i) = \bar{\mho}_i^T Z_{ik}, \quad (35)$$

where Z_{ik} and $\bar{\mho}_i$ are the appropriate vector transformations (matrices to vectors transformations). $N_{i,j}$ is a number indicating the count of agent *i* and its neighbors $(Z_{ik} \in R^{nN_{i,j}})$.

The Hamilton-Jacobi-Bellman (HJB) equation (20) along with the approximations (34) and (35) can be represented by

$$-\bar{\mho}_i^T Z_{i(k+1)} = \frac{1}{2}(\varepsilon_{ik}^T Q_{ii} \varepsilon_{ik} + \hat{u}_{ik}^T R_{ii} \hat{u}_{ik} + \sum_{j \in N_i} \hat{u}_{jk}^T R_{ij} \hat{u}_{jk}). \quad (36)$$

The target values of the value function $-\bar{\mho}_i^T Z_{i(k+1)}$ are given by

$$\delta_i^{critic} = \frac{1}{2}(\varepsilon_{ik}^T Q_{ii} \varepsilon_{ik} + \hat{u}_{ik}^T R_{ii} \hat{u}_{ik} + \sum_{j \in N_i} \hat{u}_{jk}^T R_{ij} \hat{u}_{jk}). \quad (37)$$

The error in the critic approximation is given by

$$E_i^{critic} = \delta_i^{critic} + \bar{\mho}_i^T Z_{i(k+1)}. \quad (38)$$

The positive sign in (38), follows directly the structure of the Hamilton-Jacobi-Bellman (HJB) equation. Using gradient descent to tune the critic weights yields

$$\bar{\mho}_i^{(l+1)T} = \bar{\mho}_i^{lT} - \xi_{ci}(\Im \delta_i^{critic} - \bar{\mho}_i^{lT} \Im Z_{i(k+1)}))\Im Z_{i(k+1)}^T, \quad (39)$$

where $\Im \delta_i^{critic}$ stacks $(n^2 N_{i,j} - 1)$ instances of δ_i^{critic} in a vector, $\Im Z_{i(k+1)}$ is a square matrix that stacks $(n^2 N_{i,j} - 1)$ samples of $Z_{i(k+1)}$, and $0 < \xi_{ci} < 1$ is the critic learning rate.

B. Actor Neural Networks Approximations

The target value of the policy is given by

$$\delta_i^{actor} = (g_i + d_i) R_{ii}^{-1} B_i^T \bar{\mho}_i^T \bar{\varepsilon}_{ik}, \forall i. \quad (40)$$

The approximation error for the actor neural network is given by

$$E_i^{actor} = \delta_i^{actor} - \Phi_i \bar{\varepsilon}_{ik}. \quad (41)$$

Using gradient descent yields

$$\Phi_i^{(l+1)} = \Phi_i^l - \xi_{ai}(\delta_i^{actor} - \Phi_i^l \bar{\varepsilon}_{ik}) \bar{\varepsilon}_{ik}^T, \quad (42)$$

where $0 < \xi_{ai} < 1$ is the actor learning rate.

The following algorithm is developed to tune the actor-critic neural network weights in real-time using data measured along the system trajectories.

Algorithm 2. Actor-Critic Online Implementation of Policy Iteration Algorithm.

1. Initialize the weights $\bar{\mho}_i^0, \Phi_i^0 \ \forall i$.

2. Loop 1 (*l* iterations) {

2.1 Initialize the states $\bar{\varepsilon}_i^0, \forall i$.

Loop 2 (ρ iterations) {

- Update the value function weights $\bar{\mho}_i^\rho = \bar{\mho}_i^l, \forall i$.
- Evaluate $\hat{u}_i^\rho, \forall i$ using (40).
- Measure the states $\bar{\varepsilon}_{i(k+1)}^\rho, \forall i$.
- Evaluate the value $\hat{\Theta}_{ik}(.|\bar{\mho}_i), \forall i$.

End Loop 2 when $\rho = (n^2 N_{i,j} - 1)$ }.

2.2 Update the critic weights using (39).

2.3 Update the actor weights using (42).

2.4 End Loop 1 when $\left\| \bar{\mho}_i^{l+1} - \bar{\mho}_i^l \right\|$ converges}. ■

VI. Simulation Results

An example, composed of six agents is considered. The agents have the following parameters

$$A=\begin{bmatrix} 1 & 1 \\ -0.9 & 0 \end{bmatrix}, B_1=\begin{bmatrix} 0.2 \\ 0.02 \end{bmatrix}, B_2=\begin{bmatrix} 0.3 \\ 0.15 \end{bmatrix}, B_3=\begin{bmatrix} 0.25 \\ 0.18 \end{bmatrix}, B_4=\begin{bmatrix} 0.2 \\ 0.1 \end{bmatrix}, B_5=\begin{bmatrix} 0.2 \\ 0.19 \end{bmatrix}, B_6=\begin{bmatrix} 0.21 \\ 0.29 \end{bmatrix}.$$

The connectivity weights are $a_{12} = 0.8, a_{14} = 0.7, a_{23} = 0.6,$

$a_{25} = 0.7, a_{31} = 0.8, a_{36} = 0.7, a_{41} = 0.7, a_{52} = 0.7, a_{63} = 0.7$.

The leader is agent number 5. The weighting matrices are chosen such that $Q = 10,\ R = 1$. The critic and actor learning rates are chosen such that $\xi_{ai} = \xi_{ci} = 0.1, \forall i$.

The simulation results show and discuss how fast the agents can converge to the leader's dynamics using the proposed online adaptive learning algorithm. Figure 1 shows the update of the critic weights for agent No.1. It is shown that, the critic weights converge after few iterations. The critic weights are not updated at each iteration, as imposed by the policy iteration structure. The critic weights of the agents are updated every 7 or 11 iterations, depending on the connectivity structure available to each agent in the communication graph example.

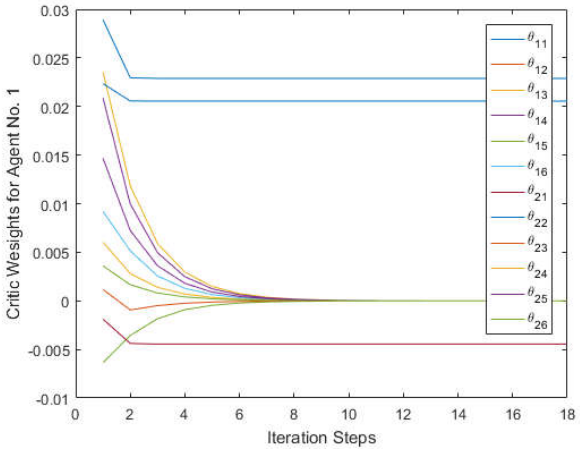

Figure 1 Critic Weights Update of Agent No.1.

Figures 2 and 3 show the asymptotic stability properties of the developed learning structure, Figure 2 shows that, the tracking error dynamics are vanishing and the agents

synchronize to the dynamics of the leader (agent number No. 5). Figure 3 shows the dynamics of the agents. The simulation results show the validity of the proposed adaptive learning algorithm.

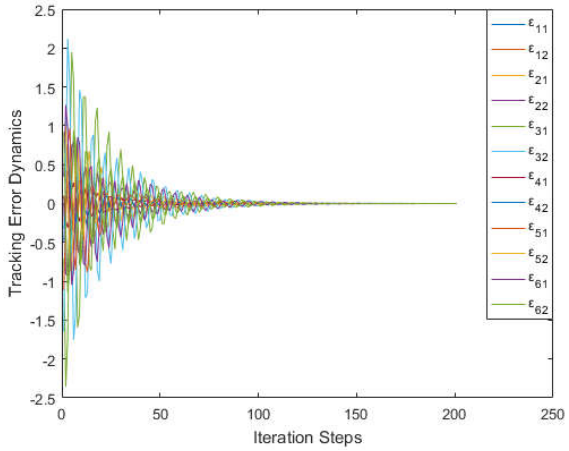

Figure 2 Tracking Error Dynamics.

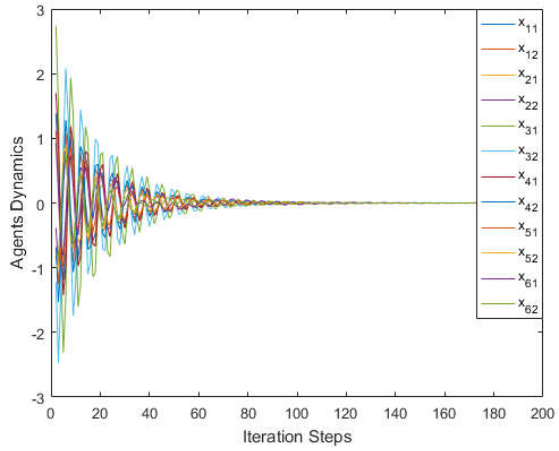

Figure 3 Agents' Dynamics.

VII. Conclusion

This paper introduces novel online adaptive learning approach based on reduced value function structures for multi-agent systems interacting on graphs. The online Policy Iteration algorithm is implemented in real-time using local information available to each agent without knowing all the agent's dynamics. The implementation is done using means of actor-critic neural networks. The convergence of the algorithm is shown to depend on the graph connectivity. Thus assumptions were made to guarantee convergence. The proposed technique overcomes the difficulties arising from the mathematical manipulations of the extensive large dimensions of the solving structures.

References

[1] M. Abouheaf and M. Mahmoud, "Online policy iteration solution for dynamic graphical games," 2016 13th International Mult. Conf. on Systems, Signals & Devices (SSD), Leipzig, pp. 787-797, 2016.

[2] M. Abouheaf, F. Lewis, S. Haesaert, R. Babsuka, and K. Vamvoudakis, "Multi-agent discrete-time graphical games: interactive Nash equilibrium and value iteration solution," Amer Con Conf, pp. 4189-4195, 2013.

[3] R. Beard and V. Stepanyan, "Synchronization of information in distributed multiple vehicle coordination control," Proc. of the IEEE Conf on Dec and Con, Maui, HI, pp., 2029-2034, 2003.

[4] J. Tsitsiklis. Problems in Decentralized Decision Making and Computation. Ph.D. dissertation. Dept. Elect. Eng. and Comput. Sci., MIT, Cambridge, MA, 1984.

[5] Z. Li, Z. Duan, G. Chen, and L. Huang, "Consensus of multi-agent systems and synchronization of complex networks: A unified viewpoint," IEEE Trans Circ & Syst, vol. 57, no. 1, pp. 213–224, 2010.

[6] X. Li, X. Wang, and G. Chen, "Pinning a complex dynamical network to its equilibrium," IEEE Trans. Circuits Syst., vol. 51, no. 10, pp. 2074–2087, 2004.

[7] W. Ren, K. Moore, and Y. Chen, "High-order and model reference consensus algorithms in cooperative control of multivehicle systems," J. Dynam. Syst., Meas., Control, vol. 129, no. 5, pp. 678–688, 2007.

[8] M. Abouheaf, F. Lewis, K. Vamvoudakis, S Haesaert, and R Babuska, "Multi-agent discrete-time graphical games and reinforcement learning solutions," Automatica, vol. 50, no. 12, pp. 3038-3053, 2014.

[9] R. Bellman. Dynamic Programming. Princeton, 1957.

[10] F. Lewis, D. Vrabie, V. Syrmos. Optimal Control. 3rd edition, John Wiley: New York, 2012.

[11] S. Lall and M. West, "Discrete variational Hamiltonian mechanics," Jrnl of Phys A: Math and Gen, vol. 39, no.19 pp. 5509-5519, 2006.

[12] P. Werbos, "Neural networks for control and system identification," IEEE Proc. CDC, 1989, pp. 260-265, 1989.

[13] P. Werbos. Approximate dynamic programming for real-time control and neural modeling. Handbook of Intelligent Control. ed. D.A. White and D.A. Sofge, New York: Van Nostrand Reinhold, 1992.

[14] A. Barto, R. Sutton, and C. Anderson, "Neuron like elements that can solve difficult learning control problems," IEEE Trans. Syst., Man, Cybern., vol. 13, pp. 835–846, 1983.

[15] D. Bertsekas and J. Tsitsiklis. Neuro-Dynamic Programming. Athena Scientific, MA, 1996.

[16] R. Sutton and A. Barto. Reinforcement Learning – An Introduction. MIT Press: Cambridge, Massachusetts, 1998.

[17] S. Sen and G. Weiss. Learning in multiagent systems, in Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. MIT Press: Cambridge. pp. 259–298, 1999.

[18] M. Abouheaf and F. Lewis, "Approximate dynamic programming solutions of multi-agent graphical games using actor-critic network structures," International Joint Conference on Neural Networks (IJCNN), Dallas, USA, pp. 1-8, 2013.

[19] J. Morimoto, G. Zeglin, and C. Atkeson, "Minmax Differential Dynamic Programming: Application to a Biped Walking Robot," IEEE Int Conf on Intel Rob and Sys, Las Vegas, USA, 2003.

[20] Y. Jiang and Z. Jiang, "Computational adaptive optimal control for continuous-time linear systems with completely unknown dynamics," Automatica, vol. 48, no. 10, pp. 2699-2704, 2012.

[21] Y. Jiang and Z.P. Jiang, "Global adaptive dynamic programming for continuous-Time nonlinear systems," ArXiv:1401.0020 [math.DS], http://arxiv.org/abs/1401.0020, 2013.

[22] B. Widrow, N. Gupta, and S. Maitra, "Punish/reward: Learning with a critic in adaptive threshold systems," IEEE Trans. Syst. Man Cybern, vol. 3, no. 5, pp. 455–465, 1973.

[23] M. Abouheaf, F. Lewis, M. Mahmoud, and D. Mikulski, "Discrete-time dynamic graphical games: model-free reinforcement learning solution," Cont.Thr. & Techn., vol.13, no.1, pp. 55-60, 2015.

[24] S. Khoo, L. Xie, and Z. Man, "Robust Finite-Time Consensus Tracking Algorithm for Multi-robot Systems," IEEE Trans. on Mechat., vol. 14, pp. 219-228, 2009.

[25] X. Li, X. Wang, and G. Chen, "Pinning a complex dynamical network to its equilibrium," IEEE Trans. Circ Syst. I, vol. 51, no. 10, pp. 2074–2087, 2004.