Model-Free Value Iteration Solution for Dynamic Graphical Games

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

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

***Abstract*—The dynamic graphical game is a special class of games where agents interact within a communication graph. This paper introduces an online model-free adaptive learning solution for dynamic graphical games. A reinforcement learning is applied in the form solutions to a set of modified coupled Bellman equations. The technique is implemented in a distributed fashion using the local neighborhood information without having a priori knowledge about the agents' dynamics. This is accomplished by means of adaptive critics, where a multi-layer perceptron neural network is applied to approximate the online solution. To this end, a novel coupled Riccati equation is developed for the graphical game. The validity of the proposed online adaptive learning solution is tested using a graphical example, where follower agents learn to synchronize their behavior to follow a leader.**

I. Introduction

The dynamic graphical game is a special type of games where the synchronization among the agents is achieved using pinning control ideas [1]. Herein, an online model-free adaptive learning approach based on value iteration is proposed to solve the dynamic graphical games. The solution is implemented in real-time using means of neural networks with gradient descent training approach. Undirected graph is considered, where the communications between agents are allowed at both directions. This work brings together reinforcement learning, adaptive critics, optimal control, and cooperative control ideas to solve the graphical games without knowing the dynamics of the agents.

The cooperative control problems are divided in terms of the goals into consensus and synchronization problems [2], [3]. In the consensus problem, the agents interact among each other to reach common goals, while in the synchronization problem, the agents follow a leader or synchronize their dynamics to the dynamics of the leader. The optimal control theory is employed to derive the necessary optimality conditions for the cooperative control problems [4]. The applications involve autonomous and intelligent systems, unmanned aerial vehicles, mobile robots, and smartgrid technologies [2], [3], [5].

The framework of dynamic programming is employed to solve the optimal control problems in [6], [7]. Approximate solutions are developed to overcome the difficulties associated with the dynamic programming problems in [8], [9]. The Approximate Dynamic Programming (ADP) solutions are classified into four main types Heuristic Dynamic Programming, Dual Heuristic Dynamic Programming, Action Dependent Heuristic Dynamic Programming, and Action Dependent Dual Heuristic Dynamic Programming [8], [10]. The ADP solutions are implemented using the Reinforcement Learning (RL) and Adaptive Critics approaches [11]–[13]. The optimal control problems are solved using Reinforcement Learning approaches in [8]. The Reinforcement Learning techniques use two-step learning processes which are used to approximate the optimal solutions in dynamic environments [14]. The learning processes are divided into policy and value iteration techniques [9], [15]–[17]. Means of adaptive critics are used to implement the RL solutions using separate actor-critic neural network structures [9], [18]. The Temporal Difference (TD) techniques use actor-critic neural structures to provide solutions for the ADP problems. The actor structure approximates the optimal decision and the critic approximates the value function [9], [14], [18]. The optimal control problem is solved in real-time using the adaptive critics in [19]. Online Reinforcement Learning techniques are used to solve the cooperative control problems in [1], [6]. The contributions of the paper are two-fold. First, an online model-free value iteration approach is developed to solve the dynamic graphical games. Second, novel coupled Riccati equations are developed for the dynamic graphical games. These equations are equivalent to solving the model-free coupled Bellman equations.

The paper is organized as follows; Section 2 demonstrates the formulation of the synchronization problem on graphs. Section 3 describes the online model-free value iteration solution. Section 4 introduces novel coupled Riccati formulation for the dynamic graphical games. Section 5 shows the adaptive critics implementation of the proposed adaptive learning solution. Section 6 introduces the simulation outcomes.

II. Dynamic Graphical Games

This section introduces the mathematical setup of the synchronization problem for dynamical systems on graphs. The objective of the optimization problem is to search for the optimal strategies for each agent in order to guarantee synchronization among the agents to the leader's dynamics.

A. Graphs

The graph is a communication structure with N nodes and communication weights $a_{ij}, \forall i, j \in N$. The connectivity of the graph is described by the matrix $\tilde{A}$, where $\tilde{A} = [a_{ij}], \forall i, j \in N$ and $i \neq j$. The in-degree matrix D of the graph is given by $D = diag(d_i)$, $d_i = \sum_{j \in N_i} a_{ij}, \forall i, j \in N$, N_i are the agents in the neighborhood of agent *i*. The graph Laplacian L is defined by $L = D - \tilde{A}$.

B. Team of Multi-Agent Systems

Each agent *i* has the following dynamics

$$\delta_{i(\ell+1)} = A\,\delta_{i\ell} + B_i\,\mu_{i\ell}, \tag{1}$$

where ℓ is the time-index, $\delta_i \in R^n$ and $\mu_i \in R^{m_i}$ are the states and the control input signals for agent *i*. A and B_i are the drift and the control input matrices.

The dynamical equation of the leader is given by [20]

$$\delta_{o(\ell+1)} = A\,\delta_{o\ell}, \tag{2}$$

where $\delta_{o\ell} \in R^n$ is a vector of the leader's states.

Pinning control ideas are employed to achieve synchronization among agents, where the leader agent is pinned to (communicates with) a small percentage of the agents through pining gains p_i [21]. In order to proceed with the analysis of the synchronization problem on graphs, a local tracking error protocol that employs the pining ideas is used such that [22]

$$x_{i\ell} = \sum_{j \in N_i} a_{ij}\big(\delta_{i\ell} - \delta_{j\ell}\big) + p_i\big(\delta_{i\ell} - \delta_{o\ell}\big), \tag{3}$$

where x_i is the local tracking error for agent *i*.

This local error protocol achieves two objectives; First, it highlights the discrepancies between each agent and its neighbors. Second, it highlights the connectivity of agent *i* to the leader *o*.

In order to solve the optimization problem, it is required to select the optimal strategy $\mu^o_{i\ell}$ for each agent *i* such that $\lim_{\ell \to \infty} \|x_{i\ell}\| = 0, \forall i$. The optimal policy is found by applying Bellman optimality principles [4]. Using (3), the local tracking error dynamics for the agents are given by

$$x_{i(\ell+1)} = Ax_{i\ell} + (d_i + p_i)B_i\mu_{i\ell} - \sum_{j \in N_i} a_{ij}B_j\mu_{j\ell}, \ \forall i. \tag{4}$$

The tracking errors (4) are coupled dynamical systems, where the strategies of agent *i* and those of its neighbors affect the dynamics of each agent *i*. The errors $x_{i\ell}$, $\forall i$ are shown to be bounded if the graph is strongly connected [6].

C. Bellman Equation

In the sequel, Bellman optimality equations are developed for the graphical system. A local measure index is considered to asses the performance of the applied strategies and evaluate the coupling behavior between agent *i* and its neighbors N_i such that

$$J_i = \sum_{\ell=0}^{\infty} C_i(x_{i\ell}, \mu_{i\ell}, \mu_{j\ell}), \ j \in N_i, \tag{5}$$

where $C_{i\ell}$ is a quadratic utility function and it is given by

$$C_{i\ell} = \frac{1}{2}\left(x^T_{i\ell}\,Q_i\,x_{i\ell} + \mu^T_{i\ell}\,R_i\,\mu_{i\ell} + \sum_{j \in N_i} \mu^T_{j\ell}\,R_{ij}\,\mu_{j\ell}\right), \tag{6}$$

where $Q_i \geq 0 \in R^{n \times n}$, $R_i > 0 \in R^{m_i \times m_i}$, and $R_{ij} > 0 \in R^{m_j \times m_j}$ are symmetric time-invariant weighting matrices.

Let the value function $S_i(X_{i\ell})$ be a solution of (5) such that

$$S_i(X_{i\ell}) = \sum_{k=\ell}^{\infty} C_i(X_{ik}, \mu_{ik}, \mu_{jk}), \ j \in N_i, \tag{7}$$

where $X_{i\ell}$ is a vector of the error states of each agent *i* and its neighbors.

Relation (7) is rearranged in order to have Bellman equation for agent *i* such that

$$\begin{aligned} S_i(X_{i\ell}) &= \frac{1}{2}(X^T_{i\ell}\,Q_i\,X_{i\ell} + \mu^T_{i\ell}\,R_i\,\mu_{i\ell} + \sum_{j \in N_i} \mu^T_{j\ell}\,R_{ij}\,\mu_{j\ell}) \\ &+ S_i(X_{i(\ell+1)}). \end{aligned} \tag{8}$$

The goal of the optimization problem is to find the optimal value function S^o_i and the optimal policy μ^o_i to minimize the cost-to-go function (7) or (8). The necessity optimality conditions [1], [6] yield

$$S^o_i(X_{i\ell}) = \operatorname{argmin} \underbrace{S_i(X_{i\ell})}_{\mu_{i\ell}}. \tag{9}$$

Thus, the optimal control policy $\mu^o_{i\ell}$ is given by

$$\mu^o_{i\ell} = -F_i\,\nabla S^o_i(X_{i(\ell+1)}), \tag{10}$$

where $F_i = R_i^{-1}\left([\ldots(d_i + p_i)\cdots - a_{ji}] \otimes B_i^T\right)$, the symbols $\ldots$ denote the respective positions of the weights a_{ji} connected to agent *j* in vector $[\ \ ]$, $\otimes$ is the Kronecker product, and $\nabla S^o_i(X_{i\ell}) = \partial S^o_i(X_{i\ell}) \,/\, \partial X_{i\ell}$.

Using the optimal policies (10) in Bellman equation (8), yields the following Bellman optimality equation

$$\begin{aligned} S^o_i\big(X_{i\ell}\big) &= \frac{1}{2}\Big(x^{\mathrm{T}}_{i\ell}Q_ix_{i\ell} + \mu^{o\mathrm{T}}_{i\ell}R_i\mu^o_{i\ell} + \sum_{j \in N_i} \mu^{o\mathrm{T}}_{j\ell}R_{ij}\mu^o_{j\ell}\Big) \\ &+ S^o_i\big(X_{i(\ell+1)}\big). \end{aligned} \tag{11}$$

It is noted that, the optimal policies (10) depend on the dynamical models of the agents (matrices $B_i, \forall i$) and the connectivity weights ($a_{ji}, \forall j, i$). The next section will propose a solution form that does not use these parameters in a direct fashion.

III. Model-Free Value Iteration Solution for Graphical Games

Value Iteration (VI) is one form of the Reinforcement Learning's processes, it is used to solve the dynamic graphical games online in [6]. The proposed approach solves Bellman equation (11) in real-time. In the sequel, a model-free value iteration solution is developed for the dynamic graphical games, where the learning process does not know any of the agents' dynamics. In order to do that, first, the classical value iteration solution is introduced for the dynamic graphical games.

A. Value Iteration Solution

An online value iteration algorithm can be developed using Bellman equation (8) and the policy (10), as explained by Algorithm 1.

Algorithm 1: Value Iteration Solution

1) **Initialize:** $S_i^0(X_{i\ell})$ and $\mu_{i\ell}^0, \forall i$.
2) **Evaluate:** $S_i^{(r+1)}(\dots), \forall i$

$$S_i^{(r+1)}(X_{i\ell}) = C_i^r(X_{i\ell}, \mu_{i\ell}, \mu_{j\ell}) + S_i^r(X_{i(\ell+1)}) \ \forall i, \quad (12)$$

where r is the iteration index.
3) **Update the policy:**

$$\mu_{i\ell}^{(r+1)} = -F_i \nabla S_i^{(r+1)}(X_{i(\ell+1)}) \ \forall i. \quad (13)$$

4) **Termination Criteria:** On convergence of $\|S_i^{r+1}(..) - S_i^r(..)\|, \forall i$.

The policy (13) depends partially on the dynamics of the agents (control input signals). In addition, it uses the out neighbors weights a_{ji}, which is a problem in the case of the directed graphs. This makes the distributed solution for the graphical game not possible. Algorithm 1 generalizes the Heuristic Dynamic Programming (HDP) solution for the dynamic graphical games [8]. To handle the aforementioned challenges, a model-free solution which does not consider the dynamics of the agents and does not use the graph's out-neighbors weights is proposed in the following development.

B. Model-Free Value Iteration Solution

In the sequel, a modified version of Bellman equation (8) is introduced for the dynamic graphical games. It generalizes the Action Dependent Heuristic Dynamic Programming (ADHDP) solution for the dynamic graphical games [8], [9]. This version considers the solving structure to be function of $X_{i\ell}$ and $\mu_{i\ell}$ for each agent i. Let the value function $Z_i(X_{i\ell}, \mu_{i\ell})$ be a solution of (8) such that

$$Z_i(X_{i\ell}, \mu_{i\ell}) = \sum_{k=\ell}^{\infty} C_i(X_{ik}, \mu_{ik}, \mu_{jk}), \ j \in N_i. \quad (14)$$

This yields the following Bellman equation

$$Z_i(X_{i\ell}, \mu_{i\ell}) = C_i(X_{i\ell}, \mu_{i\ell}, \mu_{j\ell}) + Z_i(X_{i(\ell+1)}, \mu_{i(\ell+1)}). \quad (15)$$

The general solution form of (14) is given by

$$\begin{aligned} Z_i(X_{i\ell}, u_{i\ell}) &= \frac{1}{2}[X_{i\ell}^T \ \ \mu_{i\ell}^T] \ M_i \ \begin{bmatrix} X_{i\ell} \\ \mu_{i\ell} \end{bmatrix}, \\ M_i &= \begin{bmatrix} M_{iXX} & M_{iX\mu} \\ M_{i\mu X} & M_{i\mu\mu} \end{bmatrix}. \end{aligned} \quad (16)$$

The goal of the optimization problem is to find the optimal value function Z_i^o and the optimal policy μ_i^o to minimize the cost-to-go function (14) or (15). The optimality principles yield

$$\underbrace{\operatorname{argmin} Z_i^o(X_{i\ell}, \mu_{i\ell})}_{\mu_{i\ell}} = \underbrace{\operatorname{argmin} \sum_{k=\ell}^{\infty} C_i\left(X_{ik}, \mu_{ik}, \mu_{jk}\right)}_{\mu_{i\ell}}. \quad (17)$$

Thus, the optimal policy is given by

$$\begin{aligned} \mu_{i\ell}^o &= \underbrace{\operatorname{argmin}}_{\mu_{i\ell}} \left(Z_i(X_{i\ell}, \mu_{i\ell})\right) \\ &= \underbrace{\operatorname{argmin}}_{\mu_{i\ell}} \left(\frac{1}{2}[X_{i\ell}^T \ \ \mu_{i\ell}^T] \ M_i \ \begin{bmatrix} X_{i\ell} \\ \mu_{i\ell} \end{bmatrix} \right). \end{aligned} \quad (18)$$

This yields,

$$\mu_{i\ell}^o = -\left[M_{i\mu_{i\ell}\mu_{i\ell}}^{-1} \cdot M_{i\mu_{i\ell}X_{i\ell}}\right] \cdot X_{i\ell}, \quad (19)$$

where the subscript $_{\mu_{i\ell}\mu_{i\ell}}$ denotes the policy-policy sub-block in matrix M_i and the subscript $_{\mu_{i\ell}X_{i\ell}}$ denotes the policy-states sub-block in matrix M_i.

Applying the optimal policies (19) to Bellman equation (15) yields the following Bellman optimality expression

$$\begin{aligned} S_i^o\left(X_{i\ell}, \mu_{i\ell}\right) &= \frac{1}{2}\Big(x_{i\ell}^{\mathrm{T}} Q_i x_{i\ell} + \mu_{i\ell}^{o\mathrm{T}} R_i \mu_{i\ell}^o + \sum_{j\in N_i} \mu_{j\ell}^{o\mathrm{T}} R_{ij} \mu_{j\ell}^o\Big) \\ &+ S_i^o\left(X_{i(\ell+1)}, \mu_{i(\ell+1)}^o\right). \end{aligned} \quad (20)$$

Solving Bellman optimality equations (20) and (11) is equivalent to solving the dynamic graphical game using the optimal strategies (19) and (10) respectively. Thus, a model-free value iteration approach that solves Bellman optimality equation (20) using the optimal strategy (19) is proposed as follows

Algorithm 2: Model-Free Value Iteration Solution

1) **Initialize:** $S_i^0(X_{i\ell}, \mu_{i\ell}^0)$ and $\mu_{i\ell}^0, \forall i$.
2) **Evaluate:** $S_i^{(r+1)}(\dots), \forall i$

$$\begin{aligned} S_i^{(r+1)}(X_{i\ell}, \mu_{i\ell}) &= C_i^r(X_{i\ell}, \mu_{i\ell}, \mu_{j\ell}) \\ &+ S_i^r(X_{i(\ell+1)}, \mu_{i(\ell+1)}) \ \forall i, \end{aligned} \quad (21)$$

where r is the iteration index.
3) **Update the policy:**

$$\begin{aligned} \mu_{i(\ell+1)}^{r+1} &= -M_{i(\mu_i\mu_i)}^{-1(r+1)} M_{i(\mu_i x_i)}^{(r+1)} x_{i(\ell+1)}^{r+1} \\ &- \sum_{j\in N_i} M_{i(\mu_i\mu_i)}^{-1(r+1)} M_{i(\mu_i x_j)}^{(r+1)} x_{j(\ell+1)}^{r+1}. \end{aligned} \quad (22)$$

4) **Termination Criteria:** On convergence of $\|S_i^{r+1}(..) - S_i^r(..)\|, \forall i$.

The proposed algorithm does not use any priori knowledge about the agents dynamics'. At the same time, it does not take into account the out-neighbors weights a_{ji} which appeared to be one of the main concerns in *Algorithm 1*.

IV. Coupled Riccati Equation

Herein, novel coupled Riccati equation is developed for the dynamic graphical games. This equation is equivalent to solving the coupled Bellman equation (20), where the solution for each agent i is quadratic in the states $X_{i\ell}$ and the strategy $\mu_{i\ell}$.

Theorem 1: Let the solution for the dynamic graphical game for each agent i be given by $S_i(X_{i\ell}, \mu_{i\ell}) = \frac{1}{2}[X_{i\ell}^T \ \ \mu_{i\ell}^T] \ \Theta_i \ \begin{bmatrix} X_{i\ell} \\ \mu_{i\ell} \end{bmatrix}$ (θ_i is the Riccati

solution gain) and the policies of the agents are given by (19). Then, each agent i has the following coupled Riccati equation

$$O_i^{\mathrm{T}}\ \Theta_i^{(r+1)}\ O_i = \left[\begin{array}{cc} \bar{Q}_i + \Pi^{\mathrm{T}}\bar{R}_{ij}\Pi + A_c^{\mathrm{T}}L_i^{\mathrm{T}}\Pi_i^{\mathrm{T}}\Theta_i\Pi_i L_i A_c & 0 \\ 0 & R_i \end{array}\right]^{(r)} \quad (23)$$

Note: The undefined notations are listed later on in the proof.

Proof: The overall vector of the tracking error dynamics (4) is given by

$$X_{(\ell+1)} = \bar{A}X_\ell + ((L+P)\otimes I_n)\ \bar{B}\,\mu_\ell, \quad (24)$$

where $\bar{A} = (I_N \otimes A) \in R^{nN\times nN}$, $\mu_\ell = \left[\mu_{1\ell}^{\mathrm{T}} \dots \mu_{N\ell}^{\mathrm{T}}\right]^{\mathrm{T}} \in R^{m_i N\times 1}$, $P = diag(p_i), \forall i$, and $\bar{B} = diag(B_i), \forall i \in R^{nN\times m_i N}$.

The optimal policy for each agent i is given by

$$\mu_{i\ell} = -\Theta_{i(\mu_i\mu_i)}^{-1}\Theta_{i(\mu_i x_i)}x_{i\ell} - \sum_{j\in N_i}\Theta_{i(\mu_i\mu_i)}^{-1}\Theta_{i(\mu_i x_j)}x_{j\ell}.$$

Using this local policy, the global policy is given by

$$\mu_\ell = -\left[\begin{array}{cccc} \Theta_{1(\mu_1\mu_1)}^{-1} & 0 & .. & 0 \\ 0 & \Theta_{2(\mu_2\mu_2)}^{-1} & .. & 0 \\ 0 & 0 & .. & \Theta_{N(\mu_N\mu_N)}^{-1} \end{array}\right] \times \left[\begin{array}{ccc} \Theta_{1(\mu_1 x_r),r\in 1,N_1} & .. & .. \\ .. & \Theta_{2(\mu_2 x_r),r\in 2,N_2} & .. \\ .. & .. & \Theta_{N(\mu_N x_r),r\in N,N_N} \end{array}\right] \times \left[\begin{array}{c} x_{1\ell} \\ x_{2\ell} \\ \vdots \\ x_{N\ell} \end{array}\right].$$

Let $A^c = \left(\bar{A} - ((L+P)\otimes I_n)\,\bar{B}\Pi\right)$ and $\mu_\ell = -\Pi X_\ell$, where $\Pi = \left[\begin{array}{cccc} \Theta_{1(\mu_1\mu_1)}^{-1} & 0 & .. & 0 \\ 0 & \Theta_{2(\mu_2\mu_2)}^{-1} & .. & 0 \\ 0 & 0 & .. & \Theta_{N(\mu_N\mu_N)}^{-1} \end{array}\right] \times \left[\begin{array}{ccc} \Theta_{1(\mu_1 x_r),r\in 1,N_1} & .. & .. \\ .. & \Theta_{2(\mu_2 x_r),r\in 2,N_2} & .. \\ .. & .. & \Theta_{N(\mu_N x_r),r\in N,N_N} \end{array}\right]$.

Then the vector (24) is rearranged using the global policy μ_ℓ such that

$$X_{(\ell+1)} = \left(\bar{A} - ((L+P)\otimes I_n)\,\bar{B}\Pi\right)X_\ell = A^c\,X_\ell. \quad (25)$$

Bellman equation for agent i is given by

$$S_i\big(X_{i\ell},\mu_{i\ell}\big) = \frac{1}{2}\Big(x_{i\ell}^{\mathrm{T}}Q_i x_{i\ell} + \mu_{i\ell}^{\mathrm{T}}R_i\mu_{i\ell} + \sum_{j\in N_i}\mu_{j\ell}^{\mathrm{T}}R_{ij}\mu_{j\ell}\Big) + S_i\big(X_{i(\ell+1)},\mu_{i(\ell+1)}\big). \quad (26)$$

where the value function $S_i\big(X_{i\ell},\mu_{i\ell}\big)$ can be mapped into the global states' vector X_ℓ so that

$$\begin{aligned} S_i(X_{i\ell},u_{i\ell}) &= \frac{1}{2}[X_{i\ell}^T\ \ \mu_{i\ell}^T]\ \Theta_i\ \left[\begin{array}{c} X_{i\ell} \\ \mu_{i\ell} \end{array}\right] \\ &= \frac{1}{2}[X_\ell^T\ \ \mu_{i\ell}^T]\,O_i^{\mathrm{T}}\ \Theta_i\ O_i\left[\begin{array}{c} X_\ell \\ \mu_{i\ell} \end{array}\right], \end{aligned} \quad (27)$$

where the mapping O_i is given by $O_i = \left[\begin{array}{ccc} I_i & 0 & \dots\ 0 \\ 0 & I_j, j\in N_i & \dots\ 0 \\ 0 & 0 & \dots\ I_{mi} \end{array}\right] \in R^{(Nn+m_i)\times(Nn+m_i)}$.

In a similar fashion, $S_i\big(X_{i(\ell+1)},\mu_{i(\ell+1)}\big)$ can be written so that

$$S_i\big(X_{i(\ell+1)},\mu_{i(\ell+1)}\big) = X_\ell^{\mathrm{T}}A_c^{\mathrm{T}}L_i^{\mathrm{T}}\Pi_i^{\mathrm{T}}\Theta_i\Pi_i L_i A_c X_\ell, \quad (28)$$

where $X_{i\ell} = L_i X_\ell$, $L_i = diag(I_{ni}, I_{nj}, j\in N_i) \in R^{(nN_i)\times(nN)}$, $\bar{R}_{ij} = diag(R_{ij}, j\in N_i, 0) \in R^{Nm_i\times Nm_i}$, $\bar{Q}_i = diag(0,Q_i,0), \in R^{(nN)\times(nN)}$, and $\Pi_i = \left[\begin{array}{cccc} I_i & 0 & .. & 0 \\ 0 & I_j, j\in N_i & .. & 0 \\ -\Theta_{i(\mu_i\mu_i)}^{-1}\Theta_{i(\mu_i x_i)} & -\Theta_{i(\mu_i\mu_i)}^{-1}\Theta_{i(\mu_i x_j)}, j\in N_i & .. & .. \end{array}\right] \in R^{(nN_i+m_i)\times(nN_i)}$.

Equations (26), (27), and (28) yield

$$\begin{aligned} &\frac{1}{2}[X_\ell^T\ \ \mu_{i\ell}^T]\,O_i^{\mathrm{T}}\ \Theta_i\ O_i\left[\begin{array}{c} X_\ell \\ \mu_{i\ell} \end{array}\right] = \\ &\frac{1}{2}\left(X_\ell^{\mathrm{T}}\bar{Q}_i X_\ell + \mu_{i\ell}^{\mathrm{T}}R_i\mu_{i\ell} + \right. \\ &\left. X_\ell^{\mathrm{T}}\Pi^{\mathrm{T}}\bar{R}_{ij}\Pi X_\ell + X_\ell^{\mathrm{T}}A_c^{\mathrm{T}}L_i^{\mathrm{T}}\Pi_i^{\mathrm{T}}\Theta_i\Pi_i L_i A_c X_\ell\right). \end{aligned} \quad (29)$$

This equation can be finally re-arranged in the form (23).

To the best of our knowledge, this coupled Riccati equation is the first Riccati development for the dynamic graphical games with a value function solution that depends on the states $X_{i\ell}$ and the policy $\mu_{i\ell}$.

V. Adaptive Critics Implementation

The model-free value iteration *Algorithm 2* learns the optimal solution to the Bellman optimality equations (20). The adaptive critics are used to implement the graphical game's online solution using means of neural networks. The actor and critic neural networks are used to approximate the optimal strategies and value functions respectively in a dynamic learning environment. The actor neural structure approximates the optimal control policy μ_i^o (19). The critic component approximates the optimal value function S_i^o (20).

The value function $S_i(X_{i\ell},\mu_{i\ell})$ for each agent i is approximated by a critic neural network structure such that

$$\hat{S}_{i\ell}(.|\omega_{ic}) = \frac{1}{2}[X_{i\ell}^T\ \ \mu_{i\ell}^T]\ \omega_{ic}^{\mathrm{T}}\ \left[\begin{array}{c} X_{i\ell} \\ \mu_{i\ell} \end{array}\right], \quad (30)$$

where ω_{ic} are the critic weights for agent i.

The optimal strategy $\mu_i^r(\dots)$ for each agent i is approximated by

$$\hat{\mu}_{i\ell}(.|\omega_{ia}) = \omega_{ia}^{\mathrm{T}}\,X_{i\ell}, \quad (31)$$

where ω_{ia} are the actor weights for agent i.

The gradient descent approach is used to tune and update the actor-critic weights. The target value for the critic function approximation (30) is given by

$$\begin{aligned}\bar{S}_i\big(X_{i\ell},\hat{\mu}_{i\ell}\big) &= \frac{1}{2}\Big(x_{i\ell}^{\mathrm{T}}Q_i x_{i\ell}+\hat{\mu}_{i\ell}^{\mathrm{T}}R_i\hat{\mu}_{i\ell}+\sum_{j\in N_i}\hat{\mu}_{j\ell}^{\mathrm{T}}R_{ij}\hat{\mu}_{j\ell}\Big)\\ &+ \hat{S}_i\big(X_{i(\ell+1)},\hat{\mu}_{i(\ell+1)}\big). \qquad (32)\end{aligned}$$

Similarly, the target value for the optimal strategy (31) is given by

$$\begin{aligned}\bar{\mu}_{i\ell} &= -\Big[\omega_{ic(\mu_i\mu_i)}^{T}\Big]^{-1}\omega_{ic(\mu_i x_i)}^{T}x_{i\ell}\\ &- \sum_{j\in N_i}\Big[\omega_{ic(\mu_i\mu_i)}^{T}\Big]^{-1}\omega_{ic(\mu_i x_j)}^{T}x_{j\ell}. \qquad (33)\end{aligned}$$

Thus, using the gradient descent tuning approaches, the actor-critic weights' update equations are given by

$$\omega_{ia}^{(r+1)\mathrm{T}}=\omega_{ia}^{r\mathrm{T}}-\rho_{ia}\,\big(\bar{\mu}_{i\ell}^{r}-\omega_{ia}^{r\mathrm{T}}X_{i\ell}^{r}\big)\,X_{i\ell}^{r\mathrm{T}}, \qquad (34)$$

$$\omega_{ic}^{(r+1)\mathrm{T}}=\omega_{ic}^{r\mathrm{T}}-\rho_{ic}\left(\bar{S}_{i\ell}^{r}-\frac{1}{2}\left(X_{i\ell}^{r\mathrm{T}}\,\omega_{ic}^{r\mathrm{T}}\,X_{i\ell}^{r}\right)\right)X_{i\ell}^{r}\,X_{i\ell}^{r\mathrm{T}},$$

where ρ_{ia} and ρ_{ic} are the actor and critic learning rates respectively.

The following algorithm summarizes the actor-critic implementations of the solution proposed by Algorithm 2.

Algorithm 3: Online Actor-Critic Implementation

1) **Initialize:** ω_{ia}^{0} and $\omega_{ic}^{0},\forall i$.
2) **Loop:** *(r iterations)*
   a) Initialize the states $X_{i0}^{0},\forall i$.
      - Evaluate $X_{i(\ell+1)}^{r}$, $\hat{\mu}_{i(\ell+1)}$, $\bar{S}_{i\ell}$, and $\bar{\mu}_{i\ell}$.
   b) Update the critic weights using (26).
   c) Update the actor weights (25).
3) **Termination Criteria:** On convergence of $\|S_i^{r+1}(..)-S_i^{r}(..)\|,\forall i$.

VI. Graphical Game Simulation Example

A graphical example is considered to validate the effectiveness of the proposed solution. The purpose of the simulation is to assess the convergence of the actor and critic weights, the synchronization of the agents' dynamics to that of the leader, and the stability of the local tracking error. The graph shown in Figure 1 depicts the agents' interactions. The agents drift dynamics and the control input matrices (which are used to generate the measurements) are given by [1]:

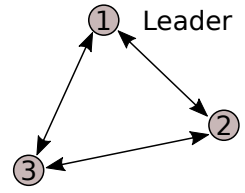

Fig. 1. Graphical game example.

$$A=\begin{bmatrix}0.995 & 0.09983\\ -0.09983 & 0.995\end{bmatrix},\ B_1=\begin{bmatrix}0.2047\\ 0.08984\end{bmatrix},$$
$$B_2=\begin{bmatrix}0.2147\\ 0.2895\end{bmatrix},\ B_3=\begin{bmatrix}0.2097\\ 0.1897\end{bmatrix}.$$

The weight matrices, connectivity weights, pinning gains, and the learning parameters are given as follows:

$$R_i=R_{ij}=1,\forall i,j\ ,\ Q_i=10,\forall i\ ,\ a_{ij}=0.25, j\neq i$$
$$\rho_{ia}=\rho_{ic}=0.01\ ,\ p_i=\begin{cases}1 & \text{, if } i=1\\ 0 & \text{, otherwise}\end{cases}. \qquad (35)$$

Figure 2 shows the convergence properties during the learning and the tuning processes for the actor and critic weights. It is shown that the actor-critic weights converge after 500 iterations. Figure 3 shows the synchronization properties for the proposed online solution. As can be seen, the tracking error dynamics decay exponentially. This means that the graphical game is led by the proposed controller to have an asymptotic stability. This is re-emphasized also by examining to the control input signals. Figure 4 illustrates the agents dynamics (a second order system is considered) and the phase-plane plots (δ_{i1} versus δ_{i2} for each agent i). This demonstrates how agents 2 and 3 managed to learn to follow their leader.

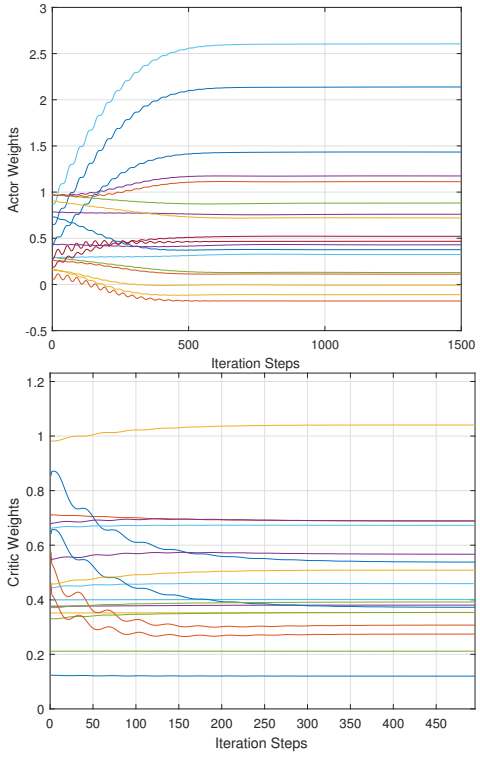

Fig. 2. Actor and critic weights.

VII. Conclusion

This paper introduced a novel adaptive learning solution for dynamic games on graphs. It is based on finding an

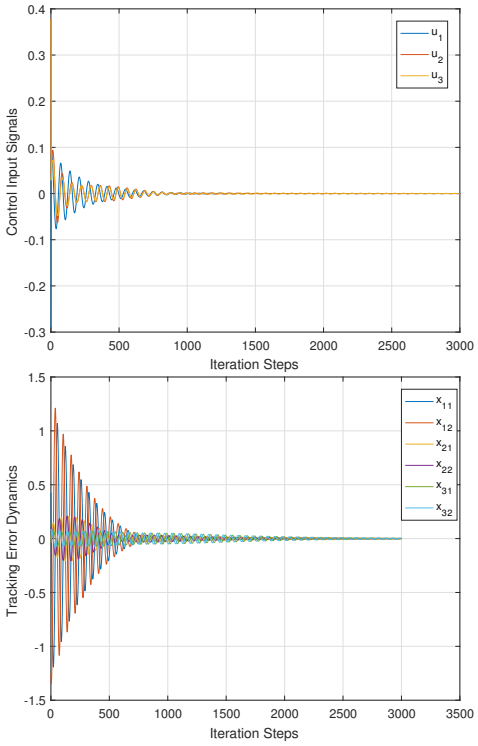

Fig. 3. Control input signals and tracking error dynamics.

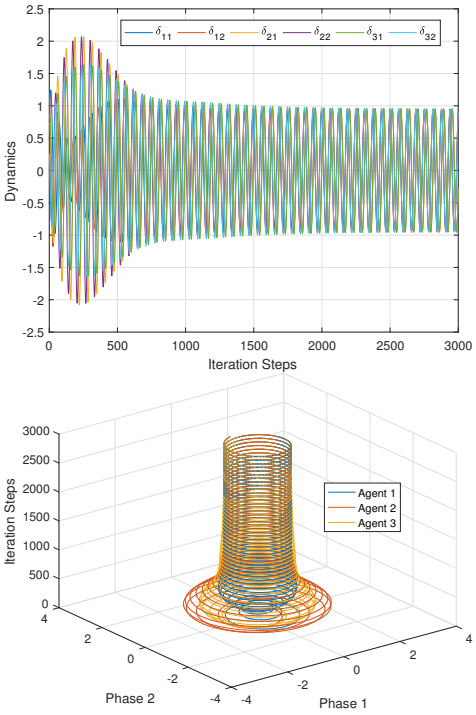

Fig. 4. The dynamics and phase plane plot.

approximate online solution for a set of modified-coupled Bellman optimality equations without the need to know the agents dynamics. The developed solution is implemented by means of adaptive critics, where the optimal strategies are approximated by an actor neural network, while the optimal value functions are approximated by a critic network. Both networks are in the form of a multi-layer perceptron with a linear output layer and no hidden layer. The algorithm was tested on a graphical simulation example where two follower agents learn how to follow a leader with no prior knowledge of the system dynamics. The actor-critic weights are shown to converge smoothly, leading to an asymptotically stable system where the tracking error dynamics exponentially vanish.

REFERENCES

[1] 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.

[2] R. Olfati-Saber, J. Fax, and R. Murray, "Consensus and cooperation in networked multi-agent systems," *Proc. IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[3] 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.

[4] F. Lewis, D. Vrabie, and V. Syrmos, *Optimal Control*, ser. Third Edition. New York: John Wiley, 2012.

[5] M. Abouheaf and M. Mahmoud, "Chapter 5 - online adaptive learning control schemes for microgrids," in *Microgrid*, M. S. Mahmoud, Ed. Butterworth-Heinemann, 2017, pp. 137 – 171.

[6] M. Abouheaf and F. Lewis, "Multi-agent differential graphical games: Nash online adaptive learning solutions," *conf. Dec. and Con. CDC13*, pp. 5803–5809, 2013.

[7] P. Werbos, *Beyond Regression: New Tools for Prediction and Analysis in the Behavior Sciences*. Ph.D. Thesis, 1974.

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

[9] A. Barto, *Reinforcement learning: An introduction*. Massachusetts: MIT Press, 1998.

[10] P. Werbos, "A menu of designs for reinforcement learning over time," *Neural networks for control*, pp. 67–95, 1990.

[11] D. Prokhorov and D. Wunsch, "Adaptive critic designs," *IEEE Transactions on Neural Networks*, vol. 8, no. 5, pp. 997–1007, 1997.

[12] J. Si, A. Barto, W. Powell, and I. Wunsch, *Handbook of Learning and Approximate Dynamic Programming*. IEEE Press, 2004.

[13] D. Bertsekas and J. Tsitsiklis, *Neuro-dynamic programming*. Massachusetts: Athena Scientific, 1996.

[14] R. S. Sutton and A. G. Barto, *Reinforcement learning–an introduction*. Cambridge: MIT Press, 1998.

[15] M. Abouheaf, F. Lewis, and M. Mahmoud, "Model-free adaptive learning solutions for discrete-time dynamic graphical games," *53rd IEEE Conf. on Dec. and Con.*, pp. 3578–3583, 2014.

[16] M. Abouheaf and M. Mahmoud, "Online policy iteration solution for dynamic graphical games," *13th Int. Multi-Conf. on Sys., Sig. Dev.*, pp. 787–797, 2016.

[17] M. Abouheaf and M. Mahmoud, "Policy iteration and coupled riccati solutions for dynamic graphical games," *Int. J. of Digital Sig. and Smrt Sys.*, vol. 1, no. 2, pp. 143–162, 2017.

[18] B. Widrow, N. Gupta, and S. Maitra, "Punish/reward: Learning with a critic in adaptive threshold systems," *IEEE Tran. Sys. Man Cyb.*, vol. 3, no. 5, pp. 455–465, 1973.

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

[20] F. Lewis, *Applied Optimal Control and Estimation: Digital Design and Implementation*. New Jersey: Prentice-Hall, 1992.

[21] 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.

[22] 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, 2014.