Model-Free Adaptive Learning Solutions for Discrete-Time Dynamic Graphical Games*

Mohammed I. Abouheaf, Frank L. Lewis, and Magdi S. Mahmoud

Abstract— This paper introduces novel model-free adaptive learning algorithm to solve the dynamic graphical games in real-time. It allows online model-free tuning of the controller and critic networks. This algorithm solves the dynamic graphical game in a distributed fashion. Novel coupled Bellman equations and Hamiltonian functions are developed for the dynamic graphical games. Nash solution for the dynamic graphical game is given in terms of the solution to a set of coupled Hamilton-Jacobi-Bellman equations developed herein. An online model-free policy iteration algorithm is developed to learn the Nash solution for the dynamic graphical game in real-time. A proof of convergence for this algorithm is given under mild assumptions about the inter-connectivity properties of the graph.

I. INTRODUCTION

The paper introduces an online model-free adaptive learning solution for a class of dynamic games known as dynamic graphical games [1]-[4]. The graphical game results from multi-agent dynamical systems on graphs, where pinning control ideas are used to synchronize the agents’ states to a leader’s state. Herein, a value function solution structure denoted by the “Q-function” is used to solve the dynamic graphical game in a distributed fashion.

The cooperative control problems are classified into synchronization and consensus control problems [5], [6]. All agents synchronize to uncontrollable node dynamics in the consensus control problem. While in the synchronization control problems, each agent reaches the same state [7], [8]. Game theory provides a solution framework for multi-agent control problems [9]. The dynamic game theory brings together the optimal control theory and game theory [10]. Discrete-time canonical forms for the Hamiltonian functions are introduced in [11]. The optimal control theory uses the Hamilton-Jacobi-Bellman (HJB) equation whose solution is the optimal cost-to-go function [12]. Nash Solution for the game is found offline in terms of the solutions to the underlying Hamilton-Jacobi equations [10].

Dynamic programming problems are solved using Approximate Dynamic Programming (ADP) approaches [1], [2], [13]-[17]. ADP algorithms are used in the computational intelligence, adaptive control, operation research, and applied mathematics [16]. Optimal control problems are solved using online and offline neuro-dynamic programming techniques in [2] and [15]. Landelius used the Action Dependent Dual Heuristic Dynamic Programming (ADHDP) approach to solve the linear quadratic control problem and showed that it is equivalent to solving the underlying Riccati equations [18].

Reinforcement Learning (RL) is concerned with learning from interaction in a dynamic environment [19]. RL algorithms are used to learn the optimal control solutions for dynamic systems in real-time [1], [2], [19], [20]. These algorithms involve Value Iteration (VI) or Policy Iteration (PI) techniques [15], [21], [22]. Multi-agent Reinforcement Learning algorithms are used to solve the optimal control problem for discrete-time systems in [23], [24].

The paper is organized as follows. Section 2 reviews the synchronization control problem on graphs. Section 3 formulates the dynamic graphical game in terms of the Q-function based Bellman equations and the Hamiltonian functions. Section 4 finds Nash solution for the graphical game in terms of the solution to a set of coupled HJB equations. Section 5 develops an online adaptive model-free learning algorithm to solve the graphical game. Section 6 uses neural network structures to solve the graphical game in real-time.

II. SYNCHRONIZATION PROBLEM ON GRAPHS

This section reviews the synchronization control problem on communication graphs.

A. Graphs

The directed graph $\hat{G}$ is defined as a nonempty finite set of $N$ agents or vertices $V = \{v_1, ..., v_N\}$ and a set of edges $E \subseteq V \times V$ [25]. The connectivity matrix $E$ is defined such that $E = [e_{ij}]$ with $e_{ij} > 0$ if $(v_i, v_j) \in E$ and $e_{ij} = 0$ otherwise. Define the diagonal in-degree matrix as $D = \text{diag}(d_i)$, with $d_i = \sum_{j \neq i} e_{ij}$ the weighted in-degree of each agent $i$. The graph Laplacian matrix $L$ is defined as $L = D - E$ [25].

B. Synchronization and tracking error dynamics

The dynamics of each agent $i$ is given by

\*This work is supported by the deanship for scientific research (DSR) at KFUPM, RG-1316-1, NSF grant ECCS-1128050, AFOSR grant FA9550-09-1-0278, ARO W911NF-05-1-0314, and China NNSF 61120106011.

M. I. Abouheaf is with Systems Engineering, King Fahd University of Petroleum & Mineral (KFUPM), P.O.Box: 3067, Dhahran–31261, SA (phone: +9660138604744; fax: +9660138602965; e-mail: abouheaf@kfupm.edu.sa).

F. L. Lewis is with University of Texas at Arlington Research Institute, The University of Texas at Arlington, Texas, USA (e-mail: lewis@uta.edu).

M. S. Mahmoud is with Systems Engineering, KFUPM, Dhahran – 31261, SA (e-mail: msmtp@kfupm.edu).
\[ x(k + 1) = Ax(k) + Bu(k) \]  

where \( x(k) \in \mathbb{R}^n \) and \( u(k) \in \mathbb{R}^m \) are the state and control input vectors of each agent \( i \).

A leader \( x_i(k) \in \mathbb{R}^n \) has the dynamics [26] given by

\[ x_i(k + 1) = Ax_i(k). \]  

The objective is to design the control inputs \( u_i(k) \) using the local neighbor information, so that all agents synchronize to the leader’s dynamics, that is \( \lim_{k \to \infty} \| x_i(k) - x_i(k) \| = 0, \forall i \) [3].

Define the local neighborhood tracking error \( e_i(k) \in \mathbb{R}^n \) [27] for each agent \( i \) as

\[ e_i(k) = \sum_{j \in N_i} e_p(x_i(k) - x_j(k)) + g_i(x_i(k) - x_i(k)) \]  

where the pinning gain is \( g_i > 0 \) for at least one agent.

For the simplicity, \( x_i(k) \) is written as \( x_i \), and so on, when the time index \( k \) is clear. The local neighborhood tracking error dynamics for each agent \( i \) is given by

\[ e_{ik(i-1)}(k) = A e_i - (d_i + g_i) B u_i + \sum_{j \in N_i} e_p B \mu_j. \]  

III. DISCRETE-TIME DYNAMIC GRAPHICAL GAMES

The solution for the graphical game is given in terms of the solution to a set of coupled Hamiltonian functions and novel Q-function based Bellman equations [12], [28], [29].

A. Dynamic game performance evaluation

The dynamic interacting games are based on the locally coupled error systems (4). Therefore, define the local performance index for agent \( i \) as

\[ J_i = \sum_{t=0}^{\infty} U_i(e_i, u_i, u_{-i}). \]

where \( U_i \) is the utility function for each agent \( i \), \( Q_s \geq 0 \), \( R_s > 0 \), and \( R_s > 0 \) are symmetric time-invariant weighting matrices. Define the policies \( u_{-i} \), \( u_{i-1} \), and \( u_{i} \) as \( \{ u_i | j \in N_i \} \), \( \{ u_j | j \in N_i, N_{i-1} \} \), and \( \{ u_j | j \in N_i, j \neq i \} \) respectively.

For the multi-player graphical game, it is desired to determine the optimal non-cooperative solutions such that

\[ J_i^* = \min_{u_i} \sum_{t=0}^{\infty} U_i(e_i, u_i, u_{-i}), \forall i \in N \]  

where \( u_{-i} \) are the neighbors’ optimal policies of each agent \( i \).

Given fixed policies \( (\mu_{j}, \mu_{-j}) \) of agent \( i \) and its neighbors, the value function for each agent \( i \) is given by

\[ V_i^*(\varepsilon_i) = \sum_{t=0}^{\infty} U_i(e_i, \mu_i, \mu_{-i}) \]

where \( \varepsilon_i \) is a vector of the state of agent \( i \), \( e_i \), and the states of its neighbors \( e_{-i} \).

B. Hamiltonian function for graphical games

The Hamiltonian function [12] of each agent \( i \) is given by

\[ H_i^*(\varepsilon_i, \lambda_{i+1}, e_i, u_i, u_{-i}) = \lambda_{i+1}^T (\varepsilon_i, \lambda_{i+1}, e_i, u_i, u_{-i}) + U_i(e_i, u_i, u_{-i}) \]  

where \( \lambda_{i} \) is the costate variable of each agent \( i \), and

\[ \varepsilon_i = \sum_{t=0}^{\infty} [I_{i}, \varepsilon_i] \]  

where \( N_{i} \) denotes the number of each agent \( i \) and its neighbors \( j \).

Denote the stationary admissible policy for each agent \( i \) by \( u_i = \pi_i = \pi_i(\varepsilon_i) \). Thus, the Hamiltonian is written such that

\[ H_i^*(\varepsilon_i, \lambda_{i+1}, u_i, \pi_{-i}) = \lambda_{i+1}^T (\varepsilon_i, \lambda_{i+1}, u_i, \pi_{-i}) + U_i(e_i, u_i, \pi_{-i}). \]

The optimal control policy is given [12] such that

\[ u_i = \arg \min_{u_i} (H_i^*(\varepsilon_i, \lambda_{i+1}, u_i, \pi_{-i})). \]

Then,

\[ u_i^* = \lambda_{i+1}^T (\varepsilon_i, \lambda_{i+1}, u_i, \pi_{-i}) \]

where \( M_i = \lambda_{i+1}^T (\varepsilon_i, \lambda_{i+1}, u_i, \pi_{-i}) \).

C. Bellman equations for graphical games

The value function (7) given stationary admissible policies yields the dynamic graphical game Bellman equations such that

\[ V_i^*(\varepsilon_i) = \frac{1}{2} (e_i^T Q_s e_i + \pi_i^T R_s \pi_i) + \sum_{j \in N_i} \pi_j^T R_s \pi_j + V_{i+1}^*(\varepsilon_{i+1}). \]

Define \( \nabla V_i^*(\varepsilon_{i+1}) \) as \( \nabla V_i^* = \partial V_i^*/\partial \varepsilon_i \). Applying the Bellman optimality principle yields

\[ V_i^*(\varepsilon_i) = \min_{u_i} (U_i(e_i, u_i, \pi_{-i}) + V_{i+1}^*(\varepsilon_{i+1})). \]

Thus, the optimal control policy for each agent \( i \) is given by

\[ \pi_i = u_i = M_i \nabla V_i^*(\varepsilon_{i+1}). \]

Substituting (14) into (13) yields the graphical game Bellman optimality equations

\[ V_i^*(\varepsilon_i) = \min \sum_{i \in N} \pi_i^T R_i \pi_i + \nabla V_i^*(\varepsilon_{i+1}) + V_{i+1}^*(\varepsilon_{i+1}) \]

with initial conditions \( V_i^*(0) = 0 \).

D. Q-Function based Bellman equations

Herein, ADHDP is extended to solve the dynamic graphical game without knowing any of the agents’ dynamics.

The Q-function for each agent \( i \) is defined as follows

\[ Q_i^*(\varepsilon_i, u_i) = U_i(e_i, u_i, \pi_{-i}) + V_{i+1}^*(\varepsilon_{i+1}) \]

Since the policies \( \pi_{-j} \) are stationary admissible, (16) is in fact a best response Bellman equation. Note that \( Q_i^*(\varepsilon_i, u_i) \) is defined such that

\[ Q_i^*(\varepsilon_i, u_i) = V_i^*(\varepsilon_i). \]

Therefore, the best response Q-function based Bellman equation is given by

\[ Q_i^*(\varepsilon_i, u_i) = U_i(e_i, u_i, \pi_{-i}) + Q_{i+1}^*(\varepsilon_{i+1}, \pi_{i+1}) \]

with initial conditions \( Q_i^*(0) = 0 \).

Define \( \Delta Q_i^* \) as \( \Delta Q_i^*(\varepsilon_i, u_i) = Q_i^*(\varepsilon_{i+1}, \pi_{i+1}) - Q_i^*(\varepsilon_i, u_i) \) and \( \nabla Q_i^* \) as \( \nabla Q_i^*(\varepsilon_i, \pi_{-i}) = \partial Q_i^*(\varepsilon_i, \pi_{-i}) / \partial \varepsilon_i \).

The optimal control policy for each agent \( i \) is given such that

\[ \pi_i = u_i^* = \arg \min_{u_i} (Q_i^*(\varepsilon_i, u_i)) \]
or, \[ \pi_a = \bar{u}_a = M^2 \mathcal{Q}_i' (\bar{e}_{i; \mathcal{Q}_i}; \pi_{i; \mathcal{Q}_i}). \] \tag{20}

Thus, the best response Bellman optimality equations based on the \(Q\)-functions \(20\) and optimal policies \(20\) are given by

\[
\begin{align*}
Q_i^* (\bar{e}_i, \bar{u}_i) &= Q_i^* (\bar{e}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}), \\
+ \mathcal{V} Q_i^* (\bar{e}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}, \bar{u}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}) M_j \mathcal{R}_j, \\
+ \sum_{j = k} \mathcal{V} Q_i^* (\bar{e}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}, \bar{u}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}) M_j \mathcal{R}_j M_i \mathcal{V} Q_i^* (\bar{e}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}, \bar{u}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}).
\end{align*}
\tag{21}
\]

E. Coupled Hamilton-Jacobi-Bellman (HJB) equations

The Hamilton-Jacobi (HJ) theory \[11\] is used to relate the Hamiltonian functions \(9\) and the \(Q\)-function based Bellman equations \(18\) such that

\[
\Delta Q_i^* (\bar{e}_i, \bar{u}_i) - \mathcal{V} Q_i^* (\bar{e}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}, \bar{u}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}) \bar{e}_i + H_i (\bar{e}_i, \mathcal{V} Q_i^* (\bar{e}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}, \bar{u}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}), \bar{u}_a, \pi_{a; \mathcal{Q}_i}) = 0
\]

\[
\text{where,} \quad H_i (\bar{e}_i, \mathcal{V} Q_i^* (\bar{e}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}, \bar{u}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}), \bar{u}_a, \pi_{a; \mathcal{Q}_i}) = \mathcal{V} Q_i^* (\bar{e}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}, \bar{u}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}).
\tag{22}
\]

The next result relates the Hamiltonians \(9\) along the optimal trajectories and the Bellman optimality equations \(21\).

Theorem 1 (Discrete-time coupled HJB equation)

a. Let \(0 < Q_i^* (\bar{e}_i, \bar{u}_i) \in C^2, \forall i \) satisfy the DTHJB equation

\[
H_i (\bar{e}_i, \mathcal{V} Q_i^* (\bar{e}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}, \bar{u}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}), \bar{u}_a, \pi_{a; \mathcal{Q}_i}) = \mathcal{V} Q_i^* (\bar{e}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}, \bar{u}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}) + U_i (\epsilon_a, \bar{u}_a, \pi_{a; \mathcal{Q}_i}) = 0
\]

with initial conditions \(\mathcal{Q}_i^* (0) = 0\), \(i = 1, \ldots, n\).

Then, \(Q_i^*\) satisfies the Bellman optimality equation \(21\).

b. Let \((A, B) \forall i \) be reachable. Let \(0 < Q_i^* (\bar{e}_i, \bar{u}_i) \in C^2, \forall i \) satisfy \(21\). Then \(Q_i^* (\bar{e}_i, \bar{u}_i)\) satisfies \(24\).

Proof:

a. If \(Q_i^* (\bar{e}_i, \bar{u}_i)\) satisfies \(24\) and \(u^*_a\) is given by \(25\), then

\[
H_i (\bar{e}_i, \mathcal{V} Q_i^* (\bar{e}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}, \bar{u}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}), u^*_a, \pi_{a; \mathcal{Q}_i}) = 0
\]

Then \(22\) yields \(\Delta Q_i^* (\bar{e}_i, \bar{u}_i) = \mathcal{V} Q_i^* (\bar{e}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}, \bar{u}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}) \bar{e}_i\).

Therefore, \(Q_i^* (\bar{e}_i, \bar{u}_i)\) satisfies \(21\).

b. Completing the squares on the Hamiltonian \(9\) for arbitrary smooth function \(Q_i^* (\bar{e}_i, \bar{u}_i)\) yields

\[
H_i (\bar{e}_i, \mathcal{V} Q_i^* (\bar{e}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}, \bar{u}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}), \bar{u}_a, \pi_{a; \mathcal{Q}_i}) = \mathcal{V} Q_i^* (\bar{e}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}, \bar{u}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}) M_j \mathcal{R}_j, \\
+ H_i (\bar{e}_i, \mathcal{V} Q_i^* (\bar{e}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}, \bar{u}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}), \bar{u}_a, \pi_{a; \mathcal{Q}_i}) = \frac{1}{2} \sum_{\pi_{a; \mathcal{Q}_i}} \mathcal{V} Q_i^* (\bar{e}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}, \bar{u}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}) M_j \mathcal{R}_j M_i \mathcal{V} Q_i^* (\bar{e}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}, \bar{u}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}) \\
+ \sum_{\pi_{a; \mathcal{Q}_i}} \mathcal{V} Q_i^* (\bar{e}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}, \bar{u}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}) M_j \mathcal{R}_j M_i \mathcal{V} Q_i^* (\bar{e}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}, \bar{u}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}})
\]

\[
\text{where,} \quad \mathcal{V} Q_i^* (\bar{e}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}, \bar{u}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}) = \frac{1}{2} \sum_{\pi_{a; \mathcal{Q}_i}} \mathcal{V} Q_i^* (\bar{e}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}, \bar{u}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}) M_j \mathcal{R}_j M_i \mathcal{V} Q_i^* (\bar{e}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}, \bar{u}_{i; \mathcal{Q}_i; \pi_{i; \mathcal{Q}_i}}).
\tag{26}
\]

IV. NASH EQUILIBRIUM FOR THE GRAPHICAL GAMES

In this section, it is shown that the solution to a set of coupled Bellman optimality equations \(21\) represents a Nash equilibrium solution for the dynamic graphical game.

Definition 1 \[10\]: The \(N\)-player game with \(N\)-tuple of optimal control policies \(u^*_1, u^*_2, \ldots, u^*_n\) is said to form a Nash equilibrium solution if for all \(i \in N\)

\[
J_i^* (u^*_i, u^*_{-i}) \leq J_i (u_i, u^*_{-i}).
\tag{34}
\]

Theorem 2 (Nash equilibrium solution):

Let \(0 < Q_i^* (\bar{e}_i, \bar{u}_i) \in \mathbb{C}^2\) satisfy \(24\) or \(21\). Let all agents use the control policies \(25\). Let the graph have a spanning tree with non-zero pinning gain into at least one root agent. Then:

a. All agents synchronize to the leader’s dynamics \(2\).
b. The optimal performance index is \(J_i^* = Q_i^* (\bar{e}_i, \bar{u}_i)\).
c. The tuple \( (u^*, u^+) \) represents the Nash equilibrium solution.

**Proof:**

a. The value function \( Q^*(\bar{e}_i, u^*_i) \) satisfies (21) such that

\[
Q^*(\bar{e}_i, u^*_i) - Q^*(\bar{e}_i, u^-_i) = -U'_i(e_i, u^*_i, u^-_i) < 0.
\]

Thus \( Q^*(\bar{e}_i, u^*_i) \) serves as a Lyapunov function, and the error systems (4) are asymptotically stable. Then according to [3], all agents synchronize to the leader’s dynamics (2).

b. The performance index at time index \( l \) is given by

\[
J'_l = Q^*(\bar{e}_i, u^*_i) + \sum_{i \in I} \left( U_i(e_i, u^*_i, \pi^-_i) - U'_i(e_i, u^*_i, u^-_i) \right).
\]

The result in part (a) yields \( Q^*(\bar{e}_i, \pi^-_i) = 0 \). Thus, rearranging (36) yields

\[
J'_l = Q^*(\bar{e}_i, u^*_i) + \sum_{i \in I} \left( U_i(e_i, u^*_i, \pi^-_i) - U'_i(e_i, u^*_i, u^-_i) \right) + \hat{Q}(Q^*(\bar{e}_i, \pi^-_i), \pi^-_i).
\]

Using the optimal control policies \((u^*_i, \pi^-_i)\), the optimal performance index (40) is given by the unique value \( Q^*(\bar{e}_i, u^*_i) \) such that

\[
J'_l = Q^*(\bar{e}_i, u^*_i).
\]

Given that

\[
\sum_{i \in I} \left( U_i(e_i, u^*_i, u^-_i) - U'_i(e_i, u^*_i, u^-_i) \right) > 0.
\]

Then (37), (41), and (42) yield

\[
J'_l(u^*_i, u^-_i) \leq J'_l(u^*_i, u^-_i).
\]

So that Nash equilibrium exists according to Definition 1.

V. MODEL-FREE ADAPTIVE LEARNING SOLUTION

An online model-free policy iteration algorithm is developed to solve the graphical game in real-time. This is a cooperative version of the ADHDP single agent’s methods introduced in [1], [2]. The Q-function is given such that

\[
Q^* = \frac{1}{2} \left[ \bar{e}_i^T u^+_i \right] \tilde{H}_i \left[ \bar{e}_i^T u^+_i \right]^T,
\]

where \( \tilde{H}_i \) is the solution matrix for each agent \( i \).

The optimal control policy is given such that arg \( \min(Q^*) = 0 \).

Then,

\[
u^*_i = -H_i^{-1}(e_i, \pi^-_i, e_{-i} - \sum_{j \neq i} H_j^{-1}(e_j, \pi^-_j, e_{-j}).
\]

Algorithm 1. (Model-free policy iteration algorithm)

**Step 1:** Start with arbitrary initial admissible policies \( u^*_i \).

**Step 2:** (Value Evaluation). Solve for \( \hat{Q}(\bar{e}_i, u^*_i) \), \( \forall i \).

\[
\hat{Q}(\bar{e}_i, u^*_i) = U_i(e_i, u^*_i, u^-_i) + \hat{Q}(\hat{e}_i, e_{-i}^*, u^-_{-i}).
\]

**Step 3:** (Policy Improvement).

\[
u^*_i = -H_i^{-1}(e_i, \pi^-_i, e_{-i} - \sum_{j \neq i} H_j^{-1}(e_j, \pi^-_j, e_{-j}), \forall i.
\]

**Step 4:** On convergence of \( \|\hat{H}_i - H_i\| \). End.

The following theorem provides the convergence proof for Algorithm 1.

**Theorem 3.** Let all agents perform Algorithm 1 simultaneously. Assume all initial policies \( u^*_i, \forall i \) are admissible. Suppose that \( \sigma(R^*_i R) \) is small. Then

a. \( u^*_i, \forall i > 0 \) are stabilizing and hence admissible policies.

b. Algorithm 1 generates monotonically decreasing sequence of value functions \( \hat{Q}_i, \forall i \) and this sequence converges to the best response value functions \( \hat{Q}_i, \forall i \) that satisfy (24).

**Proof:**

a. Equations (18) or (46) yield,

\[
\hat{Q}(\hat{e}_i, u^*_i, u^-_i) < 0, \forall i, l.
\]

Thus, the value functions \( \hat{Q}_i, \forall i, l \) are Lyapunov functions. For a similar reason, \( \hat{Q}_i, \forall i, l \) are Lyapunov functions.

Using \( u^*_i, \forall i \) are admissible, then there exist value functions \( \hat{Q}_i, \forall i \) such that

\[
\hat{Q}_i(e_i, u^*_i, u^-_i) = \hat{Q}_i(e_i, u^*_i, u^-_i) + \Delta \hat{U}_i(u^*_i, u^-_i)
\]

where

\[
\Delta \hat{U}_i(u^*_i, u^-_i) = \sum_{i \in I} \left( u_i^*- u_i^- \right) R_i(u_i^*, u_i^-).
\]

The policies \( u^*_i, \forall i, l \) are given by (47). Therefore, \( \Delta \hat{U}_i(u^*_i, u^-_i) > 0 \) and (49) yields

\[
\hat{Q}(\hat{e}_i, u^*_i, u^-_i) > \hat{Q}(\hat{e}_i, u^*_i, u^-_i).
\]

Similarly,

\[
\hat{Q}(\hat{e}_i, u^*_i, u^-_i) = \hat{Q}(\hat{e}_i, u^*_i, u^-_i) + \Delta \hat{U}_i(u^*_i, u^-_i).
\]

The assumption \( \Delta \hat{U}_i(u^*_i, u^-_i) > 0 \) guarantees that

\[
\hat{Q}(\hat{e}_i, u^*_i, u^-_i) > \hat{Q}(\hat{e}_i, u^*_i, u^-_i).
\]

or,

\[
\sum_{i \neq j} \left( u_i^*- u_i^- \right) R_j(u_i^*- u_j^-) > 0.
\]

Using the norm properties on this inequality yields

\[
\sum_{i \neq j} \left( u_i^*- u_i^- \right) R_j(u_i^*- u_j^-) > \sigma(R^*_i R) (g + d) \tilde{\sigma}(R^*_i R) \left( g + d \right).
\]

where \( \tilde{\sigma}(R^*_i R) \) is the maximum singular value of a matrix.

Under the assumption (54). The inequalities (50) and (52) yield

\[
\hat{Q}(\hat{e}_i, u^*_i, u^-_i) > \hat{Q}(\hat{e}_i, u^*_i, u^-_i) > \hat{Q}(\hat{e}_i, u^*_i, u^-_i).
\]
The assumption (54) can be guaranteed for any choice of $R_{ij}$ of agent $i$ by selecting $R_{ij}$ large enough. Therefore, $u_{ii}^{n+1}, \forall i, l$ are stabilizing and hence admissible policies.

b. Equation (48) yields
\[
\widetilde{Q}(\mathbf{E}_{ik}^{a}, \mathbf{W}_{lk}^{a}, u_{ik}^{0}) - \widetilde{Q}(\mathbf{E}_{ik}^{a}, \mathbf{W}_{lk}^{a}, u_{ik}^{n}) < 0.
\]
Equation (17) or (46) yields,
\[
\widetilde{Q}_{iR}^{(a)}(\mathbf{E}_{ik}^{a}, \mathbf{W}_{lk}^{a}, u_{ik}^{0}) - \widetilde{Q}_{iR}^{(a)}(\mathbf{E}_{ik}^{a}, \mathbf{W}_{lk}^{a}, u_{ik}^{n}) + U_{i}(e_{a}, u_{a}^{l}, u_{i}^{l-1}) = 0.
\]
Equations (56), (57), and the assumption (54) yield
\[
\sum_{a} \left( \widetilde{Q}(\mathbf{E}_{ik}^{a}, \mathbf{W}_{lk}^{a}, u_{ik}^{0}) - \widetilde{Q}(\mathbf{E}_{ik}^{a}, \mathbf{W}_{lk}^{a}, u_{ik}^{n}) \right) = \sum_{a} \left( \widetilde{Q}_{iR}^{(a)}(\mathbf{E}_{ik}^{a}, \mathbf{W}_{lk}^{a}, u_{ik}^{0}) - \widetilde{Q}_{iR}^{(a)}(\mathbf{E}_{ik}^{a}, \mathbf{W}_{lk}^{a}, u_{ik}^{n}) \right) + U_{i}(e_{a}, u_{a}^{l}, u_{i}^{l-1}) = 0.
\]
This reduces to
\[
\sum_{a} \left( \widetilde{Q}(\mathbf{E}_{ik}^{a}, \mathbf{W}_{lk}^{a}, u_{ik}^{0}) - \widetilde{Q}(\mathbf{E}_{ik}^{a}, \mathbf{W}_{lk}^{a}, u_{ik}^{n}) \right) = 0
\]
Part (a) implies that $\sum_{a} \left( \widetilde{Q}(\mathbf{E}_{ik}^{a}, \mathbf{W}_{lk}^{a}, u_{ik}^{0}) \right) \rightarrow 0$ and $\sum_{a} \left( \widetilde{Q}(\mathbf{E}_{ik}^{a}, \mathbf{W}_{lk}^{a}, u_{ik}^{0}) \right) \rightarrow 0$ such that
\[
\sum_{a} \left( \widetilde{Q}(\mathbf{E}_{ik}^{a}, \mathbf{W}_{lk}^{a}, u_{ik}^{0}) \right) = \sum_{a} \left( \widetilde{Q}(\mathbf{E}_{ik}^{a}, \mathbf{W}_{lk}^{a}, u_{ik}^{n}) \right).
\]
Therefore, by induction (58) yields
\[
0 < \cdots < \widetilde{Q}_{i}^{(a)} < \widetilde{Q}_{j}^{(a)} < \cdots < \widetilde{Q}^{(a)}_{i}, \forall i, l.
\]
The stabilizing policies (47) form a Nash equilibrium tuple for the graphical game. The decreasing sequence (59) is bounded by $\{0, \widetilde{Q}(\mathbf{E}_{ik}^{a}, \mathbf{W}_{lk}^{a}, u_{ik}^{0})\}$. Then the best response value functions $\widetilde{Q}^{(a)}_{i}, \forall i$ exist and satisfy (24) according to Theorem 2 such that
\[
0 < \cdots < \widetilde{Q}_{i}^{(a)} < \widetilde{Q}^{(a)}_{i} < \cdots < \widetilde{Q}_{i}, \forall i, l.
\]

VI. NETWORK SOLUTIONS FOR THE GRAPHICAL GAME

This section uses neural network structures to solve the dynamic graphical game in real-time.

The Q-function for each agent $i$, $Q(\mathbf{E}_{ik}, u_{i})$ (44) is approximated by the network structure $\hat{Q}(\mathbf{E}_{ik}, \mathbf{W}_{ik}^{a})$, such that
\[
\hat{Q}_{i}(\mathbf{E}_{ik}, \mathbf{W}_{ik}^{a}) = \frac{1}{2} \mathbf{L}_{a}^{T} \hat{\mathbf{W}}_{ik}^{a} \mathbf{L}_{a}
\]
where $\hat{\mathbf{W}}_{ik}^{a} \in R^{(nN_{i} + m_{i}) \times (nN_{i} + m_{i})}$, $\forall i$. $\mathbf{L}_{a} = [e_{a}^{1} \cdots e_{a}^{l} \hat{u}_{i}^{a}]$ is a vector of the states $\mathbf{E}_{ik}$ and the control action approximation of agent $i$, $\hat{u}_{i}$. The policy $\hat{u}_{i}$ is given by (47) with $\mathbf{W}_{ik} = \hat{\mathbf{W}}_{ik}^{a}$ such that
\[
\hat{u}_{i} = -\mathbf{W}_{ik}^{aT} \mathbf{E}_{ik} + \sum_{a} \mathbf{W}_{ik}^{aT} \mathbf{E}_{ik}.
\]
The Q-function (61) is written as
\[
\hat{Q}_{i}(\mathbf{E}_{ik}, \mathbf{W}_{ik}^{a}) = \mathbf{W}_{ik}^{aT} \mathbf{L}_{a}
\]
where $\mathbf{E}_{ik} \in R^{(nN_{i} + m_{i}) \times (nN_{i} + m_{i}) + 2d}$ denotes the Kronecker product quadratic polynomial basis vector and $\hat{\mathbf{W}}_{ik} = \nu(\mathbf{W}_{ik})$ with $\nu(\cdot)$ a vector valued matrix function that acts on the symmetric matrices and returns a column vector.

Using (63) in (46) yields
\[
\hat{W}_{ik}^{aT}(\mathbf{E}_{ik} + \mathbf{E}_{ik}^{a}) = \frac{1}{2} \left( e_{a}^{T} \hat{Q}_{i} \mathbf{E}_{ik} + \hat{u}_{i}^{aT} \mathbf{E}_{ik} + \sum_{a} \hat{u}_{i}^{aT} \mathbf{W}_{ik}^{aT} \mathbf{E}_{ik} \right).
\]
Let $\hat{\mathbf{W}}_{ik}^{aT}$ be the target value of the network structure $\hat{Q}_{i}(\mathbf{E}_{ik}, \hat{u}_{i}) = (\hat{Q}_{i}(\mathbf{E}_{ik}, \mathbf{W}_{ik}^{a}), \hat{Q}_{i}(\mathbf{E}_{ik}, \mathbf{W}_{ik}^{a}))$ such that
\[
\hat{\mathbf{W}}_{ik}^{aT} = \frac{1}{2} \left( e_{a}^{T} \hat{Q}_{i} \mathbf{E}_{ik} + \hat{u}_{i}^{aT} \mathbf{E}_{ik} + \sum_{a} \hat{u}_{i}^{aT} \mathbf{W}_{ik}^{aT} \mathbf{E}_{ik} \right).
\]
The square sum of the approximation error is given by
\[
err_{i}^{2} = \frac{1}{2} \left( e_{a}^{T} \hat{Q}_{i} \mathbf{E}_{ik} + \hat{u}_{i}^{aT} \mathbf{E}_{ik} + \sum_{a} \hat{u}_{i}^{aT} \mathbf{W}_{ik}^{aT} \mathbf{E}_{ik} \right)
\]
where $\hat{\mathbf{W}}_{ik} = (\mathbf{W}_{ik}^{aT}, \mathbf{W}_{ik}^{aT})$, $\forall i$. $\mathbf{W}_{ik}^{aT} \in R^{(nN_{i} + m_{i}) \times (nN_{i} + m_{i})}$ is a row vector of the target values (65) and $\mathbf{W}_{ik}^{aT} \in R^{(nN_{i} + m_{i}) \times (nN_{i} + m_{i})}$ is a square matrix of $(nN_{i} + m_{i})(nN_{i} + m_{i} + 1) / 2$ samples of $(\mathbf{E}_{ik} + \mathbf{E}_{ik}^{a})$.

The update rules for the neural network weights using gradient descent are given by
\[
\mathbf{W}_{ik}^{aT} = \mathbf{W}_{ik}^{aT} - \mu_{a} \left( \nu(\mathbf{W}_{ik}^{aT}) - \mathbf{W}_{ik}^{aT} \right)\mathbf{E}_{ik} \mathbf{E}_{ik}^{aT}
\]
where $0 < \mu_{a} < 1$ is the network-learning rate.

The following algorithm uses the data measured along the system trajectories to tune the network weights in real-time.

Algorithm 2. (Online tuning of the neural network).

1: Initialize the critic weights $\mathbf{W}_{ik}$.
2: Do Loop (l iterations) {
2.1: Start with initial state $\mathbf{E}_{ik}, \forall i$ on the system trajectory.
2.2: Update the neural network weights using (68).
2.3: On convergence of $\left| \mathbf{Q}_{i}^{(a)} \right|$ End Loop}.

Consider the directed graph example shown in Figure 1.

Figure 1. Graphical game example.

The data of the graph example are given as follows:

Agents’ dynamics: $A = \begin{bmatrix} 0.995 & 0.09983 \\ -0.09983 & 0.995 \end{bmatrix}$
Graph parameters: $g_i = 1$, $e_{i_1} = 0.8$, $e_{i_2} = 0.7$, $e_{i_3} = 0.6$, $e_{i_4} = 0.8$.

Figure 2 shows the neighborhood tracking error dynamics. This figure shows that Algorithm 2 yields stability and synchronization to the leader’s state.

**Figure 2. Tracking error dynamics.**

**VII. CONCLUSIONS**

Novel Q-function based Bellman equations and Hamiltonian functions are developed to solve a class of dynamic games known as dynamic graphical games. The Nash solution for the dynamic graphical game is given in terms of the solution to a set of coupled HJB equations developed herein. An online model-free policy iteration algorithm is developed to solve the dynamic graphical game in real-time. Convergence proof for the policy iteration algorithm is provided under mild assumptions about the inter-connectivity properties of the graph.

**REFERENCES**


