Towards Control of Evolutionary Games on Networks

James R. Riehl† and Ming Cao†

Abstract—We investigate a problem related to the control of evolutionary games on networks, in which the nodes represent agents engaged in multiple simultaneous 2-player evolutionary games and the edges define who plays with whom. The strategy dynamics are governed by a deterministic evolutionary update rule, while a set of control agents can be fixed to a desired strategy for the duration of the game. We seek here the smallest set of control agents which will result in all agents in the network converging to the desired strategy. We address network types in order of increasing complexity, first providing analytical solutions or bounds for an evolutionary prisoner’s dilemma game on regular networks and trees, and then presenting an algorithm that uses graph partitioning to recursively decompose a larger problem into several smaller problems, the solutions to which can be used to construct an approximate solution on the original network at greatly reduced computation. Finally, we provide simulations to demonstrate that this algorithm finds a near-optimal control set in the majority of cases.

I. INTRODUCTION

Evolutionary games on networks were originally studied by biologists to understand the evolution of cooperation and other behaviors in structured populations occurring in nature [1][2]. Since then the theory has spread to other fields such as sociology and economics, where tools from evolutionary game theory are used to model and predict interactions between humans connected by networks [3][4]. One of the well-established findings in the field is the existence of social dilemmas, where a population of individuals seeking to maximize their own payoff acts against the interests of the population as a whole, which has consequences ranging from costly inefficiencies to traffic congestion [5] and catastrophic network failures [6]. There is thus an urgent need to understand whether manipulation of certain agents in the network can help to resolve such dilemmas. Although we are now starting to see some literature on evolutionary games from a feedback control perspective [7][8], there remain a large number of open research problems including the one addressed here, which is how to efficiently control the outcome a complex networked evolutionary game.

Recent years have seen pioneering work in the control of evolutionary games on networks. Cheng et al. presented a framework for studying the control of networked evolutionary games using large-scale logical dynamic networks to model transitions between all possible strategy states [8]. They used this framework to derive equivalent conditions for reachability and consensus of strategies on a network given a particular set of control nodes, but determining the smallest set of control nodes to reach a desired state remains a challenging open problem. A recent simulation study showed that in a prisoner’s dilemma game on scale-free networks, fixing the strategies of the highest degree nodes was more effective than fixing random nodes [9]. In [10], we presented initial results on bounding the solution to a minimum-agent control problem on tree networks, and this paper is an extension of those results to a broader class of networks.

Our approach begins by constructing analytical solutions to a minimum-agent control problem on ring and torus networks. Adding another level of complexity, we provide bounds for tree networks with uniform branching structure. Finally, we introduce an algorithm that uses graph partitioning to decompose a larger problem into several smaller problems that are easier to compute and whose solutions can be combined into an approximate solution to the original problem, inspired by techniques that have been applied previously to various graph optimization problems [11]. A key advantage of this approach is that the smaller problems are in exactly the same form as the original problem and can thus be further decomposed recursively, allowing for the approximation of complex problems on very large networks. Moreover, the computations are partially decentralized since each of the partitioned components can be solved locally without any knowledge of the larger network.
II. FRAMEWORK

Let $G = (V, E)$ denote an undirected network consisting of a player set $V = \{1, \ldots, n\}$ and an edge set $E \subseteq V \times V$, where each edge represents a 2-player symmetric game between neighbors. The players choose strategies from a finite discrete set $\mathcal{S} := \{A, B, \ldots\}$ and receive payoffs upon completion of the game according to the matrix $M$, which has size $|\mathcal{S}| \times |\mathcal{S}|$.

We assume that at each time step, players use a single strategy against all opponents, and thus the games occur simultaneously. We denote the strategy state by $x(t) = [x_1(t), \ldots, x_n(t)]^T$, where $x_i(t) \in \mathcal{S}$ is the strategy of agent $i$ at time $t$. Total payoffs for each agent are given by

$$y_i(t) = \sum_{j \in N_i} M_{x_i(t), x_j(t)},$$

where $N_i := \{j \in V : \{i, j\} \in E\}$ is the neighbor set of player $i$.

Evolutionary games are characterized by the property that better performing strategies are adopted more often. We capture this dynamic with a strategy update rule where agents choose the strategy of a player in their self-inclusive neighborhood who received the greatest total payoff in the previous round. If this payoff is not greater than the payoff of the updating player, then the player keeps its own strategy. This is a standard type of imitation rule that dates back to the earlier studies on the spatial prisoner’s dilemma and cellular automata [12][13][14]. It is also equivalent to the unconditional imitation rule used in [8].

Let $A \in \mathcal{S}$ be the desired strategy for convergence of the network and to which the control agents will be fixed. Also, let $S_i^*(t)$ denote the set of strategies achieving the maximum payoff in the neighborhood of agent $i$ at time $t$. The strategy update rule can be expressed as follows:

$$x_i(t+1) = \begin{cases} A, & i \in \mathcal{L} \\ x_i(t), & x_i(t) \in S_i^*(t) \\ x_i^*(t), & \text{otherwise} \end{cases}$$

for each agent $i = 1, \ldots, n$, where $x_i^*(t)$ denotes a strategy in the set $S_i^*(t)$. When this set contains two or more strategies, in order to preserve deterministic dynamics, we assume that the strategies are chosen by some prescribed order.

III. PROBLEM FORMULATION

In this paper we are interested in how to efficiently drive an evolutionary game on a network to a desired outcome regardless of how the unforced network would behave. Specifically, we seek the smallest set of control agents that will cause all agents in a given network to converge to the desired strategy. We formalize the problem statement as follows:

**Problem 1 (Minimum-Agent Consensus Control):**

Given a graph $G$, payoff matrix $M$ and initial strategy state $x(0)$, what is the minimum number of control agents $|\mathcal{L}^*|$ needed to ensure that all agents, governed by the dynamics (2), will converge to the desired final strategy $A$?

One can compute an exact solution to the MACC problem through brute-force methods, i.e. simulating the game for all possible sets of control sets $\mathcal{L}$ from low cardinality to high until a solution is found, but the complexity of such algorithms are exponential in the size of the network. This is not particularly surprising because when applied to a grid network, the dynamics (2) define a non-linear cellular automaton, which are well-known for the complexity of behaviors that arise from simple sets of rules [15]. Whether or not this problem belongs to a well-known class of difficult problems such as NP-hard remains an open research problem.

IV. SOLUTIONS FOR PD GAMES ON HIGHLY STRUCTURED NETWORKS

We begin our analysis by considering the famous prisoner’s dilemma (PD) game, in which each agent can either cooperate $C$ or defect $D$. A common way to parameterize the PD game is to model it by giving each agent the option of paying a cost $c$ to give a benefit $b$ to the other agent, where $b > c > 0$. This results in the following payoff matrix:

$$M_{PD} = \begin{pmatrix} C & D \\ D & 0 \end{pmatrix} = \begin{pmatrix} C & D \\ D & 0 \end{pmatrix} \frac{1}{b}$$

where we divided by $b$ on the right to obtain a matrix in one parameter, the cost/benefit ratio $r = \frac{c}{b}$, where $r \in (0, 1)$. The dilemma arises out of the fact that the best total outcome occurs when both agents cooperate, but agents can always increase their own payoffs by switching to defect; hence mutual defection is the expected
outcome. We focus here on the prisoner’s dilemma game because it is widely-considered to be the most difficult case in which to achieve sustained cooperation and is thus a challenging problem for control, with the added benefit of having an intuitive interpretation. However, the analysis methods used here can be extended to other types of games, and the algorithm presented in Section V applies to arbitrary networks and payoff matrices.

In what follows we seek solutions or bounds on the solution to the MACC problem for the PD game (3) starting from a network of uniform defectors.

A. Regular networks

We start by analyzing some regular networks, where all nodes have the same degree. The simplest connected regular network is the ring network, where each node has degree two. It is straightforward to show that for any cost-benefit ratio \( r \), a ring of initial defectors in the PD game (3) can only be switched to cooperators by controlling all agents. Adding a dimension results in a grid of uniform degree four that wraps around on either side in the shape of a torus. We define the dimension of the torus by \( m \leq p \) where \( m \geq 3 \). We can now divide the problem into three intervals of the ratio \( r \): (1) \( r < \frac{1}{4} \), (2) \( \frac{1}{4} \leq r < \frac{1}{2} \), and (3) \( r \geq \frac{1}{2} \). In case (1), either one full row or a small block of control agents is sufficient to begin the propagation of strategy \( C \) outward in the network, but the minimum block size is either 4 or 6 depending on the whether the dimensions are even or odd. In case (2), a block of control agents spanning one dimension of the torus and 2 or 3 agents wide in the other dimension will result in all other nodes switching to \( C \). Finally, all agents must be controlled in case (3). The following proposition summarizes these results (proof in [16]).

**Proposition 1:** For an \( m \times p \) (\( 3 \leq m \leq p \)) torus network with payoff matrix \( M_{PD} \), and governed by the dynamics (2) with \( x_i(0) = D \) for all agents \( i \), the minimum number of control agents \( |L^*| \) is given in Table I.


| \( r \) | \( m \) | \( p \) | \( |L^*| \) |
|-------|-------|-------|-------|
| \( r < \frac{1}{4} \) | – | – | \( 4 \leq |L^*| \leq m \) |
| \( \frac{1}{4} \leq r < \frac{1}{2} \) | even | even | \( 2m \) |
| \( \frac{1}{4} \leq r < \frac{1}{2} \) | even | odd | \( \min(3m, 2p) \) |
| \( \frac{1}{4} \leq r < \frac{1}{2} \) | odd | odd | \( 3m \) |
| \( r \geq \frac{1}{2} \) | – | – | \( n \) |

**B. Tree networks**

Trees are connected graphs without cycles, whose branching structure makes them well-suited to algorithms that use a hierarchical approach. In [10], we presented such a pair of algorithms for bounding the solution to the MACC problem for arbitrary tree networks. Here we seek to derive analytical bounds to the solution on tree networks with uniform branching structure, that is, an \( L \)-level rooted-tree in which each node has exactly \( k \) branches except for the final level. In other words, the degree of the root node is \( k \), the degree of nodes on level 2 to \( L \) is \( k + 1 \), and the degree of nodes on level \( L \) is 1. The total number of nodes is then \( n = \sum_{i=0}^{L-1} k^i \). By dividing the tree into its subtree components and computing the minimum number of control agents needed to switch each subtree from \( D \) to \( C \) from the bottom up, we obtain the following closed-form bounds on the solution to Problem 1 for a network engaged in the PD game (proof in [16]).

**Proposition 2:** For an \( L \)-level tree with uniform \( k \)-branching structure, payoff matrix \( M_{PD} \), and governed by the dynamics (2) with \( x_i(0) = D \) for all agents \( i \), the minimum number of control agents is bounded as follows:

\[
k^{L-2} \min(k, \rho_1) \leq |L^*| \leq 1 + \min(k, 1 + \rho_1(L - 2) + \rho_0) + \sum_{i=1}^{L-1} k^{i-1} \min(k, \rho_1(L - i)),
\]

where \( \rho_0 := [1 + r k] \) and \( \rho_1 := [1 + r (k + 1)] \).

V. APPROXIMATION FOR ARBITRARY NETWORKS

We now turn our attention to networks of arbitrary structure, and show how to use graph partitioning to decompose a larger problem into several smaller and computationally feasible problems, the solutions to which we can combine into an approximate solution to the original problem. The proposed meta-algorithm can make use of any algorithm that solves or approximates the general problem, including recursive algorithms for itself since the decomposed problems are in exactly the same form as the original one. This approach has the additional advantage of being partially decentralized since the solutions for each of the partitioned subgraphs can be computed locally without knowing anything about the structure of the rest of the network.
A. Partitioning the network

A \( k \)-partition of a graph \( G = (V, E) \) is a set of vertex subsets \( \bar{V} := \{\bar{V}_1, \ldots, \bar{V}_k\} \) that is disjoint (\( \bar{V}_i \cap \bar{V}_j = \emptyset \) for all \( i \neq j \)), and includes all vertices (\( \bigcup_{i=1}^{k} \bar{V}_i = V \)).

In particular, we seek a graph partition that will facilitate an accurate approximation to the problem while minimizing computation. The resulting bound will be tightest when the sub-problems are maximally independent, which is achieved when the number of edges connecting nodes of different components is minimized. However, computation is minimized when the size of the largest component is minimized, which occurs when the partitioned subgraphs have equal size. We can reformulate this objective as a constraint by requiring that each component of the partition contains a minimum number of \( q \in [0, \left\lceil \frac{n}{k} \right\rceil] \) nodes. This is called the \( q \)-bounded graph partition problem and is known to be NP-hard. However, there are good heuristic approximations such as in [17], which is a spectral graph partition that uses a \( k \)-means algorithm to cluster the eigenvectors associated with the adjacency matrix into \( k \) groups. The result is a partition that tries to minimize the number of edges cut by the partition while keeping the size of the partitions close to the same.

B. Solving the decomposed problem

Once the graph has been partitioned, the goal is to solve the \( k \) sub-problems on each of the subgraphs and use the results to construct an approximate solution to the original problem. Since the subgraph interdependencies are completely defined by the edges connecting them, we include the agents at either end of these edge in a new set \( V_{out} := \bigcup_{i=1}^{k} V_{out i} \), where the in-agents and out-agents are defined as

\[
\bar{V}_j^{in} := \{i \in \bar{V}_j : A_i \subseteq \bar{V}_j\} \\
\bar{V}_j^{out} := \bar{V}_j - \bar{V}_j^{in},
\]

for each \( j = 1, \ldots, k \). Algorithm 1 begins by solving the MACC problem on each of the subgraphs, assuming that all agents in \( V_{out} \) are controlled. We then relax control on the out-agents as much as possible by solving the MACC problem on each pair of connected subgraphs, this time finding the minimum number of out-agents needed to achieve convergence assuming that the control sets computed for the subgraph remain controlled. This guarantees that we will obtain a set \( \hat{\mathcal{L}} \) of control agents that achieves the objective, because in the worst case the assumption made for the subgraphs cannot be relaxed and all out-agents must remain controlled.

\[
\text{foreach } \bar{V}_j \in \bar{V} \text{ do} \\
\quad x_i := \begin{cases} x_i(0), & i \in \bar{V}_j^{in} \\ A_i, & \text{otherwise} \end{cases} \quad \text{for each } i \in \bar{V}_j \\
\quad \hat{L}_j := L(G, M, \bar{V}_j^{in}, x); \\
\quad \hat{L}^{in} = \bigcup_{j=1}^{k} \hat{L}_j; \\
\quad \hat{x}_i := \begin{cases} A_i, & i \in \hat{L}^{in} \\ x_i(0), & \text{otherwise} \end{cases} \quad \text{for each } i \in \bar{V} \\
\quad \hat{L} := L(G, M, \bar{V}^{out}, \hat{x}); \\
\]

Algorithm 1: Finding a near-minimal set of control agents for an evolutionary game on an arbitrary network.

Algorithm 1 results in a set of control agents that drive all agents in the given network to the desired final strategy starting from any initial state and is thus an upper bound on the solution to the MACC problem. Let \( \hat{L} := L(G, M, \bar{V}, x(0)) \) denote the smallest subset of \( \bar{V} \subseteq \bar{V} \) needed to drive all agents in the network to strategy \( A \) given payoff matrix \( M \) and initial state \( x(0) \). This can be computed by any suitable algorithm, including an exhaustive search if necessary. Now let \( \hat{V} \subseteq \bar{V} \) denote the set of agents not using strategy \( A \). The following Lemma simply states that as long as all agents not using strategy \( A \) are available to be controlled, i.e. contained in \( \bar{V} \), then a solution exists to the problem \( P(G, M, \bar{V}, x(0)) \) (proof in [16]).

Lemma 1: If \( \hat{V} \subseteq \bar{V} \), then the problem \( L(G, M, \bar{V}, x(0)) \) admits a solution.

Theorem 1: Given a network game on graph \( G \) with payoff matrix \( M \), governed by the dynamics (2), Algorithm 1 computes a control set \( \hat{L} \) that will drive all agents in the network from the initial state \( x(0) \) to the desired strategy \( A \) and is thus an upper bound to the solution to Problem 1.

\[
\text{Proof: } \text{For each subgraph } \bar{V}_j, \text{ since } x_i = A \text{ for all agents not in } \bar{V}_j^{in}, L(G, M, \bar{V}_j^{in}, x) \text{ admits a solution by Lemma 1. When the agents } \bar{V}_j^{out} \text{ are included in the control set, the problems on each of the subgraphs can be solved independently. This is because the agents in}
\]

2880
each \( V_{j}^{\text{out}} \) interact only with agents in \( V_{j}^{\text{in}} \) and other agents in \( V_{j}^{\text{out}} \), which are controlled. Hence the union of the control sets for each subgraph \( \mathcal{L}_{j} \) and \( V_{j}^{\text{out}} \) will drive all agents to strategy \( A \), and thus \( L(G, M, V_{j}^{\text{out}}, x) \) admits a solution corresponding to \( \mathcal{L} = \mathcal{L}_{j}^{\text{in}} \cup V_{j}^{\text{out}} \). The last step is effectively a relaxation of the control on the out-agents \( V_{j}^{\text{out}} \), but if no smaller solution is found, the original solution containing all out-agents will be returned.

An important property of this algorithm is that the subproblems \( L(G, M, V_{j}^{\text{in}}, x) \) and \( L(G, M, V_{j}^{\text{out}}, x) \) may themselves be approximated by Algorithm 1 after applying another level of graph partition, hence making possible the bounding of \( |\mathcal{L}^{\ast}| \) on very large networks using a recursive implementation.

C. Example

Here we demonstrate Algorithm 1 for the PD game (3) with \( r = 0.2 \) on a network that was generated by randomly placing 24 agents uniformly in the unit square and connecting every pair of agents that fall within a distance 0.3 of each other. We seek the minimum number of control agents needed to catalyze uniform cooperation in the given network, shown in Fig. 2.

We use a spectral graph partitioning algorithm [17] to divide the network into three subgraphs as shown in Fig. 2(b). Next, we temporarily assume the out-agents are controlled as in Fig. 2(c) and then use a brute force search to find the minimum number of agents needed to control each of the partitioned components Fig. 2(d). Relaxing control on the out-agents allows us to remove one of them as we see in Fig. 2(e). The result is a set of 13 control agents that will catalyze uniform cooperation in the network.

D. Simulations

To test the tightness of the upper bound \( |\hat{\mathcal{L}}| \) computed by Algorithm 1, we generated 500 random geometric networks with up to 15 nodes – small enough to compute the true value of \( |\mathcal{L}^{\ast}| \) using a brute-force algorithm for comparison. In addition to the PD game (3) with \( r = .2 \), we also include results for snowdrift (SD) and stag-hunt (SH) games, using the payoff matrices

\[
M_{SD} = \begin{pmatrix} C & D \\ D & 3 \end{pmatrix}, \quad M_{SH} = \begin{pmatrix} C & D \\ D & -1 \end{pmatrix}.
\]

Fig. 1 shows a histogram of \( |\hat{\mathcal{L}}| - |\mathcal{L}^{\ast}| \) for each of the three games. A total of 92% of the upper bounds \( |\hat{\mathcal{L}}| \) were within one node of \( |\mathcal{L}^{\ast}| \), with a slightly greater accuracy for the snowdrift game than for the other two games.

We also examined the performance on scale-free networks to compare this approach to the results of [9] mentioned in the introduction. We generated 500 random scale-free networks by preferential attachment with 50 agents and a minimum degree of 3. Using the same prisoner’s dilemma matrix (3), we compared the results of Algorithm 1 using an 8-partition against highest degree and random agent selection. For each of the latter two methods, agents are added to the control set until uniform cooperation is achieved in the network. The results were that an average of \( |\hat{\mathcal{L}}| = 7.4 \) agents were computed by Algorithm 1, compared to 23.8 for degree-based control, and 31.8 for random agent control.

VI. CONCLUSIONS AND FUTURE WORK

We have investigated a new problem related to the control of evolutionary games on networks. Specifically, given a network and a payoff matrix, we seek the minimum number of agents that, when fixed to a desired strategy, will catalyze uniform convergence of the network to that desired strategy. For a parameterized prisoner’s dilemma game, we have provided solutions and bounds for two-dimensional torus graphs and tree networks with uniform branching structure. We also presented a method for approximating the solution on arbitrary networks using graph partitioning. This algorithm has the advantage of being decentralized such that local solutions can be computed without knowledge of the larger network. Moreover, by applying the approach recursively, one can obtain an approximate solution for
very large networks at greatly reduced computation. Simulations showed that the results for one level of decomposition were very often at or near the optimal values for three different types of games.

In addition to extending the analytical results on structured networks to arbitrary payoff matrices, we are working on developing a lower bound algorithm as a measure of optimality for Algorithm 1. We also plan to allow dynamic control sequences and to apply similar methods to more complex evolutionary dynamics, including stochastic models and asynchronous play.

REFERENCES