21st International Symposium on
Mathematical Theory of Networks and Systems
July 7-11, 2014. Groningen, The Netherlands

Minimal-Agent Control of Evolutionary Games on Tree Networks

James R. Riehl[†] and Ming Cao[†]

Abstract— **We investigate the control of evolutionary games on tree networks, presenting an algorithm that computes a set of agents that, when their strategies are fixed, will drive all other agents in the network to the desired strategy. The nodes in the network represent agents engaged in 2-player evolutionary games and the edges define who plays with whom. After each round, each agent updates its strategy to that of its neighbor who received the highest total payoff. We are interested in finding the minimum number of control agents needed to drive the entire network to a desired strategy. The proposed algorithms compute upper and lower bounds on the solution to this problem for arbitrary trees and payoff matrices. Simulations demonstrate that the control set corresponding to the upper bound is minimal in the majority of cases, and otherwise within a small number of agents from minimal.**

I. Introduction

The study of evolutionary games on networks has until recently focused primarily on understanding the emergence of cooperation and other behaviors for various types of games, reproductive dynamics, and network structures. A comprehensive survey on this topic can be found in [12]. Originally developed by biologists to understand the dynamics of structured populations occurring in nature [9], it is a topic that has become prominent in a wide range of other disciplines. Sociologists and economists for example, use tools from evolutionary game theory to model and predict interactions between humans connected by networks [11][6]. Complexity theorists in physics and mathematics have also shown great interest perhaps due to the wealth of fascinating open problems in this domain [3]. As control engineers, we take the perspective of system designers as we seek to influence the behavior of networks by manipulating the strategies of certain players in the network. There are indeed many situations in which one might want to control the outcome of the game or cause a particular strategy to emerge throughout the fields mentioned above. Although literature is beginning to emerge on evolutionary games from a feedback control perspective [7] [4], to our knowledge there has not yet been an algorithmic method proposed for controlling an evolutionary game on a network. Here we present one approach for computing bounds on the minimal number of control agents needed to achieve a desired uniform strategy on tree networks.

Trees are a class of acyclic connected graphs that represent a wide range of real networks having hierarchical organization such as employees of a business, families, and food chains. It may also be useful to design systems such as robotic networks and distribution grids on tree networks. The highly structured nature of trees makes them well-suited for algorithmic optimization methods. However, control over tree networks still poses interesting challenges, as in [2] for example, where a subclass of tree graphs is shown to be uncontrollable by a single leader.

Control of evolutionary games on networks is a field that is only very recently emerging. Cheng *et al.* presented a framework for studying the control of networked evolutionary games using large-scale logical dynamic network to model transitions between all possible states of the game [4]. They used this framework to derive equivalent conditions for reachability and consensus of strategies on a network given a particular set of control agents, but determining the smallest set of control agents to reach a desired state remains a challenging open problem. Another recent study focused on the idea of pinning control in a prisoner's dilemma on scale-free networks [5]. In this work the authors showed that controlling nodes with higher degree induced a higher frequency of cooperation in the networks than randomly selected pinning nodes, but they did not attempt to derive a minimal control node set to achieve full cooperation nor do they make any claims on the optimality their proposed approach. The main contribution of this work is an algorithmic approach for approximating minimal-agent control sets for arbitrary evolutionary games on tree networks, in combination with a lower bound on the solution to provide a measure of optimality.

[*]The work was supported in part by the European Research Council (ERCStG-307207) and the EU INTERREG program under the auspices of the SMARTBOT project.

[†]Faculty of Mathematics and Natural Sciences, ITM, University of Groningen, The Netherlands, {j.r.riehl, m.cao}@rug.nl

ISBN: 978-90-367-6321-9

II. Framework

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote the graph of a network on which an evolutionary game is taking place, meaning that the players are engaged in repeated games with a strategy update rule that is similar to the fitness-based payoffs of evolutionary models. The graph consists of a node set $\mathcal{V} = \{1, \dots, n\}$ and edge set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, where each edge represents a 2-player symmetric game between adjacent nodes, and hence we assume the graph is undirected. Players choose strategies from a binary set $\mathcal{S} = \{A, B\}$ and receive payoffs upon completion of the game according to the matrix M, where

$$M = \begin{array}{c} \\ A \\ B \end{array} \begin{array}{c} \begin{array}{cc} A & B \end{array} \\ \begin{pmatrix} a & b \\ c & d \end{pmatrix} \end{array}. \tag{1}$$

For convenience of notation, we index the matrix M by strategy, *e.g.* $M_{A,B} = b$. Let $x(t) = [x_1(t), \dots, x_n(t)]^T$ denote the vector of strategies for all players, where $x_i(t) \in \mathcal{S}$ is the strategy of player i at time t. The set of neighbors of an agent i is defined by $\mathcal{N}_i := \{j \in \mathcal{V} : (i, j) \in \mathcal{E}\}$. We assume that at each time step, players use a single strategy against all neighbors, and thus the games can be considered to occur simultaneously. Payoffs for each player are summed over all neighbors and can be expressed as the vector $y = [y_1, \dots, y_n]^T$, where

$$y_i(t) = \sum_{j \in \mathcal{N}_i} M_{x_i(t), x_j(t)}. \tag{2}$$

After each round of games, agents only switch strategies if a neighboring agent received a higher payoff in the previous round. This can be thought of as a *win-stay-lose-shift* (WSLS) [10] strategy that operates on total payoff rather than single game outcomes, and is also equivalent to the *unconditional imitation* rule used in [4]. We add to this framework a set of *control agents* whose strategy can be externally controlled. We exclude here the possibility of dynamic control sequences and instead assume that a subset $\mathcal{L} \in \mathcal{V}$ of the agents can be fixed to the desired strategy A. To formalize an expression for the dynamics, let $\mathcal{S}_i^*(t)$ denote the set of strategies achieving the maximum payoff at time t in agent i's self-inclusive neighborhood $\mathcal{N}_i \cup \{i\}$. The WSLS strategy update rule can be expressed as follows:

$$x_i(t+1) = \begin{cases} A, & i \in \mathcal{L} \\ x_i(t), & x_i(t) \in \mathcal{S}_i^*(t) \\ \bar{x}_i(t), & \text{otherwise} \end{cases} \tag{3}$$

for each agent $i = 1, \dots, n$, where $\bar{x}_i(t)$ denotes the alternate strategy of the one used by agent i at time t, which is unique since we consider only two-strategy games. Therefore (3) defines a deterministic strategy update rule.

III. Problem Formulation

An uncontrolled evolutionary game on a network can have many possible outcomes depending on the initial strategies, payoff matrix, network topology, and update rule. There is a large body of literature that considers these outcomes in depth including the survey [12], but for this analysis we are interested in how to drive a networked evolutionary game to a desired state regardless of how the unforced network would behave. Specifically, we seek the smallest set of control agents that will drive all agents in a given network to converge to a desired strategy under the dynamics (3). We formalize the problem statement as follows:

Problem 1: Given a graph $\mathcal{G}$, 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 (3), converge to the desired final strategy A in finite time?

The complexity of this problem is currently an open question, but there is reason to suspect that it is at least as complex as the minimum-agent control problem in networked dynamical systems, which for the case of linear first-order systems on trees, can be determined by solving the NP-hard minimum path cover problem [8]. However, the dynamics of evolutionary games are generally nonlinear, and while a chain of agents can be controlled by a single end in the case of networked linear dynamical systems, this is generally not true of an evolutionary game on a network. Moreover, when applied to a grid network, the dynamics (3) define a nonlinear cellular automaton, which are well-known for the complexity of behaviors that arises from simple sets of rules [1].

IV. Bounding the Solution on Tree Networks

In this section we present algorithms that compute upper and lower bounds on the solution to Problem 1. To compute the upper bound, we take advantage of the tree structure by starting at the lowest level of the tree and working up towards the root, at each level controlling the minimum number of agents needed to maintain the desired strategy in all lower levels, while also preventing the possibility of any agents switching

from strategy A to B. This algorithm also constructs an approximately minimal agent control set $\hat{\mathcal{L}}$. The lower bound is computed by assuming that all agents with the potential to switch to B for some reachable configuration of neighbor states will eventually do so, and this provides a measure of optimality for the control set computed in the upper bound algorithm.

Before we introduce the algorithms, we need to define a few quantities and agent sets. First we choose an arbitrary root agent $r = 1$. Conceptually, any tree can be thought of as consisting of many levels or *generations* of parent nodes and corresponding children, and we proceed with this analogy for clarity of presentation of the algorithms. Denoting the number of generations by n_g, let $g : \mathcal{V} \to \{0, \ldots, n_g\}$ be the mapping of agents to their generation in the tree rooted at r, which is equivalent to the distance in edges of the shortest path from each agent to the root. Let $\mathcal{V}_\ell := \{i \in \mathcal{V} : g(i) = \ell\}$ denote the tree generation subsets for each level ℓ, and let $\mathcal{C}_p := \{c \in \mathcal{V}_\ell : (p, c) \in \mathcal{E}\}$ denote the set of children of a given agent p.

The upper bound algorithm is designed to select a set of control agents to ensure that agents playing strategy A will never switch to B but that agents starting with strategy B will eventually be caused to switch to strategy A by their respective parents. However, some agents playing B may switch to A due to influence by lower level agents, which is beneficial but needs to be accounted for in the algorithm design. Hence we categorize the agents by initial strategy into three disjoint subsets of $\mathcal{V}$: those that play strategy A from the start ($\mathcal{A}$), those that start with strategy B and *cannot* be caused to switch to A from below ($\mathcal{B}$), and those than start with strategy B and *can* be caused to switch from below ($\mathcal{U}$). Initially we have $\mathcal{A} := \{i : x_i(0) = A\}$ but note that the size of $\mathcal{A}$ can increase as agents are added to the control set $\hat{\mathcal{L}}$. We will need to construct the set $\mathcal{U}$ from the bottom up, so we initialize it to the empty set and then define $\mathcal{B} := \{i : x_i(0) = B \wedge i \notin \mathcal{U}\}$.

Let d_i^A, d_i^B, and d_i^U denote the number of children of agent i in each respective category. Following are quantities that are used in the algorithms to bound the payoff values:

$$\check{y}_i^{X,Y} := M_{X,Y} + d_i^A M_{X,A} + d_i^B M_{X,B} + d_i^U \min_{Z \in \mathcal{S}} M_{X,Z}$$

$$\hat{y}_i^{X,Y} := M_{X,Y} + d_i^A M_{X,A} + d_i^B M_{X,B} + d_i^U \max_{Z \in \mathcal{S}} M_{X,Z}$$

for $i = 2, \ldots, n$, where $X, Y \in \mathcal{S}$ represent the strategies used by agent i and the parent of agent i respectively. For $i = 1$, the expressions are the same except that the term $M_{X,Y}$ is excluded since the root agent has no parent. We also need to allow for the possibility that the parent strategy may change from B to A at an unknown time. We represent this by $\hat{y}_i^{X,*} := \max_{Z \in \mathcal{S}}(\check{y}_i^{X,Y})$. We also define the following shorthand notation for the maximum payoff of agents playing B in the self-inclusive child set of an agent i: $\hat{y}_{\mathcal{C}_i}^{B,A} := \max(\hat{y}_i^{B,A}, \max_{j \in \mathcal{C}_i} \hat{y}_j^{B,B})$. We are now ready to introduce the upper bound algorithm.

$\hat{\mathcal{L}} := \{1\}$;
$\mathcal{U} := \emptyset$;
$\ell := n_g$;
while $\ell > 0$ **do**
  foreach $p \in \mathcal{V}_{\ell-1}$ **do**
    if $\ell < n_g$ **then**
      foreach $c \in \mathcal{C}_p$ **do**
        if $x_c(0) = B \quad \wedge$
        $\big[\big[x_p(0) = A \wedge \hat{y}_c^{B,A} > \min_{j \in \mathcal{C}_c - \mathcal{B}}(\check{y}_j^{A,B})\big]$
        $\vee \big[x_p(0) = B \wedge \hat{y}_c^{B,*} > \min_{j \in \mathcal{C}_c - \mathcal{B}}(\check{y}_j^{A,B})\big]\big]$
        $\vee \big[x_c(0) = A \wedge \check{y}_c^{A,B} < \max_{j \in \mathcal{C}_c - \mathcal{B}}(\hat{y}_j^{B,A})\big]$
        then
          $\hat{\mathcal{L}} := \hat{\mathcal{L}} \cup \{c\}$;
        else if $x_c(0) = B \quad \wedge$
        $\big[\big[x_p(0) = A \wedge \check{y}_c^{B,A} < \max_{j \in \mathcal{C}_c - \mathcal{B}}(\hat{y}_j^{A,B})\big]$
        $\vee \big[x_p(0) = B \wedge \hat{y}_c^{B,*} < \max_{j \in \mathcal{C}_c - \mathcal{B}}(\hat{y}_j^{A,B})\big]\big]$
        then
          $\mathcal{U} := \mathcal{U} \cup \{c\}$;
        end
      end
    end
    $c^* := \arg\max_{c \in \mathcal{C}_p - \hat{\mathcal{L}}}(\hat{y}_{\mathcal{C}_c}^{B,A})$;
    while $\check{y}_p^{A,A} \leq \hat{y}_{\mathcal{C}_{c^*}}^{B,A}$ **do**
      $\hat{\mathcal{L}} := \hat{\mathcal{L}} \cup c^*$;
      $c^* := \arg\max_{c \in \mathcal{C}_p - \hat{\mathcal{L}}}(\hat{y}_{\mathcal{C}_c}^{B,A})$;
    end
  end
  $\ell := \ell - 1$;
end

Algorithm 1: Computes an an approximately minimal set of control agents $\hat{\mathcal{L}}$ to drive a tree network to uniformity in the desired strategy A.

Algorithm 1 proceeds as follows. For reasons that will be made clear later, we initialize the control set $\hat{\mathcal{L}}$ to

contain the root agent. We also initialize the set $\mathcal{U}$ to the empty set since we will need to construct it as we go. Working upwards in the tree from the bottom, we first check in steps 8-9 for agents playing A that might be caused to switch to B by a neighbor and add control to the appropriate agents in step 10 to prevent that from happening. After updating the set $\mathcal{U}$ in step 13, we then examine each of the one-generation subtrees consisting of a single parent and its children in steps 17-21, adding children to the control set until all children will either switch to A once the next two higher generations are playing A, or are controlled or already playing A. In order to minimize the number of control agents added, children are added to the control set in decreasing order of their maximum neighboring B payoff $y_{\mathcal{C}_c}^{B,A}$. Proceeding to the next level, we repeat the process, such that each agent using strategy B is ensured to have an incentive to switch to the strategy of its parent, until the top subtree for which the root agent is guaranteed to use strategy A. We now prove that the control set $\hat{\mathcal{L}}$ causes all agents in the network to shift from the initial strategy state $x(0)$ to A in finite time.

Theorem 1: Given a tree network $\mathcal{G}$ of agents engaged in an evolutionary game governed by the dynamics (3) with payoff matrix M starting from initial strategy state $x(0)$, Algorithm 1 computes a set of control agents $\hat{\mathcal{L}}$ such that $x_i(t) \to A$ for all $i \in \mathcal{V}$.

Proof: It suffices to show that for all $i \in \mathcal{V}$,

1) $x_i(\tau) = A \implies x_i(t) = A \quad \forall t > \tau$
2) $x_i(0) = B \implies \exists \tau > 0 : x_i(\tau + 1) = A.$

We prove statement 1 by contradiction. Given $x_i(\tau) = A$ for some $\tau > 0$, suppose there exists a time τ' such that $x_i(\tau') = B$. From (3), this implies $i \notin \hat{\mathcal{L}}$ and that there exists a neighboring agent $j \in \mathcal{N}_i$ such that $x_j(\tau'-1) = B$ and $y_j(\tau'-1) > y_i(\tau'-1)$. Due to the tree structure, we know that $g(j) \in \{g(i)-1, g(i)+1\}$. First, suppose $g(j) = g(i) + 1$, meaning that $g(i) < n_g$. We know that $y_j(\tau' - 1) \leq \hat{y}_j^{B,A}$. If $g(i) > 0$, let p denote the parent of i. If $x_p(0) = A$ then $y_i(\tau'-1) \geq \check{y}_i^{A,A}$. Since $y_j(\tau'-1) \leq \hat{y}_{\mathcal{C}_j}^{B,A}$, our assumption $y_j(\tau'-1) > y_i(\tau'-1)$ implies that $y_i^{A,A} > \hat{y}_{\mathcal{C}_j}^{B,A}$, which would imply from steps 17-21 that $i \in \hat{\mathcal{L}}$, a contradiction. If $x_p(0) = B$ then $y_i(\tau'-1) \geq \check{y}_i^{A,B}$, and our assumption then implies that $\check{y}_i^{A,B} < \max_{j \in \mathcal{C}_c - \mathcal{B}}(\hat{y}_j^{B,A})$, which means that $i \in \hat{\mathcal{L}}$ by steps 8-10, another contradiction. If $g(i) = 0$ then $y_i^{A,A} > \hat{y}_{\mathcal{C}_i}^{B,A}$, again resulting in the $i \in \hat{\mathcal{L}}$ contradiction from steps 17-21. Now suppose that $g(j) = g(i) - 1$, meaning that $g(i) > 0$. In this case, we know that $y_i(\tau' - 1) \geq \min_{k \in \mathcal{C}_j - \mathcal{B}}(\check{y}_k^{A,B})$. Denote by p the parent of agent j. If $x_p(0) = A$ then $y_j(\tau' - 1) \leq \hat{y}_j^{B,A}$. Otherwise if $x_p(0) = B$ then $y_j(\tau' - 1) \leq \hat{y}_j^{B,*}$. Our assumption then implies that $\min_{k \in \mathcal{C}_j - \mathcal{B}}(\check{y}_k^{A,B})$ is strictly less than either $\hat{y}_j^{B,A}$ or $\hat{y}_j^{B,*}$, respectively, and we have that $j \in \hat{\mathcal{L}}$, a contradiction to the earlier fact that $x_j(\tau) = B$. Therefore statement 1 holds.

For statement 2, let us first consider an agent i such that $g(i) = 1$, *i.e.* a child of the root r. For this agent, if $x_i(0) = B$ then $\hat{y}_{C_i}^{B,A} < \check{y}_r^{A,A}$. It follows that $S_i^*(0)$ contains only the strategy A and thus $x_i(1) = A$. Similarly, consider an agent on an arbitrary level $g(i) = \ell$ such that $x_i(0) = B$. Suppose there is a time τ when all agents at generation $\ell - 1$ and higher play strategy A. There then must exist a parent $p \in \mathcal{N}_i$ such that $g(p) = \ell - 1$. Again by steps 17-21, it must be true that $\hat{y}_{C_i}^{B,A} < \check{y}_p^{A,A}$. Therefore $S_i^*(0) = \{A\}$ and $x_i(\tau+1) = A$, meaning that all agents on level ℓ playing strategy B at time τ will switch to A at time $\tau + 1$. Since all agents at level 1 and higher use strategy A, and using the fact proved above that agents who switch to A remain using A, we have proved by induction that statement 2 holds and therefore all agents in the network will play strategy A by time $\tau = n_g$. ■

Upon completion of algorithm, since the root agent may or may not need to be controlled, we can simulate the game for the case where $\hat{\mathcal{L}}$ excludes the root to determine whether it needs to be retained in $\hat{\mathcal{L}}$. This can be done with only a small amount of extra computation, and will reduce the size of $\hat{\mathcal{L}}$ by a maximum of one.

Next we introduce an algorithm to compute a lower bound on the solution to Problem 1, after first defining a new set of bounds on the agent payoffs based purely on their degree d_i and placement within the tree:

$$\check{y}_i := M_{B,A} + (d_i - 1)(\min_{X \in \mathcal{S}} M_{B,X})$$
$$\hat{y}_i := M_{A,B} + (d_i - 1)(\max_{X \in \mathcal{S}} M_{A,X}),$$

where $\check{y}_i$ defines the minimum payoff needed to switch agent i from B to A, $\hat{y}_i$ defines the maximum payoff agent i can attain while playing A and neighboring an agent playing B. The following algorithm uses these quantities to compute a lower bound $|\check{\mathcal{L}}| \leq |\mathcal{L}^*|$. Let $\mathcal{P} := \{i \in \mathcal{V} : \mathcal{C}_i \neq \emptyset\}$ denote the set of all agents with neighbors on lower levels.

Theorem 2: The size of the control set $|\check{\mathcal{L}}|$ computed in algorithm 2 is a lower bound on $|\mathcal{L}^*|$.

$\check{\mathcal{L}} := \emptyset$;
foreach $p \in \mathcal{P}$ **do**
  $c^* := \max_{c \in \mathcal{C}_p \cap \mathcal{B}}(\check{y}_c)$;
  while $\hat{y}_p \leq \hat{y}_{c^*}$ **do**
    $\check{\mathcal{L}} := \check{\mathcal{L}} \cup c^*$;
    $c^* := \max_{c \in \mathcal{C}_p \cap \mathcal{B} - \check{L}}(\check{y}_c)$;
  end
end

Algorithm 2: Computes a lower bound on the solution to Problem 1.

Proof: Let $\check{\mathcal{L}}_p$ denote the agents added to $\check{\mathcal{L}}$ for each agent $p \in \mathcal{P}$ in Algorithm 2, *i.e.* $\check{\mathcal{L}} = \bigcup_{p \in \mathcal{P}} \check{\mathcal{L}}_p$. Suppose that the network can be switched to all A with a smaller number of agents than $\check{\mathcal{L}}$. It follows that there exists p such that the number children of p included in the control set is smaller than $\check{\mathcal{L}}_p$. This implies that there exists $j \in C_p$ such that $x_j(0) = B$ and $\check{y}_j \geq \hat{y}_p$, which means that $x_j(t) = B$ for all t, a contradiction. Therefore $|\check{\mathcal{L}}|$ is a lower bound on $|\mathcal{L}^*|$. ■

A. Example: Evolutionary Snowdrift on a Tree

In this section we provide an example showing how to select a set of control agents to achieve full cooperation in the snowdrift game on a small tree. In the snowdrift game, each player has the option of helping the opponent (strategy A), for example by clearing snow from the street, or not (strategy B) and letting the opponent do all the work. The sum of the payoffs are the same whether one or both players cooperate, but the worst outcome occurs when both players do nothing. Consider the network shown in Fig. 1 engaged in an evolutionary snowdrift game governed by the dynamics (3) who are initially all doing nothing. The payoff matrix is:

$$M_{SD} = \begin{matrix} \\ A \\ B \end{matrix} \begin{matrix} \begin{matrix} A & B \end{matrix} \\ \begin{pmatrix} 3 & 1 \\ 5 & 0 \end{pmatrix} \end{matrix}. \tag{4}$$

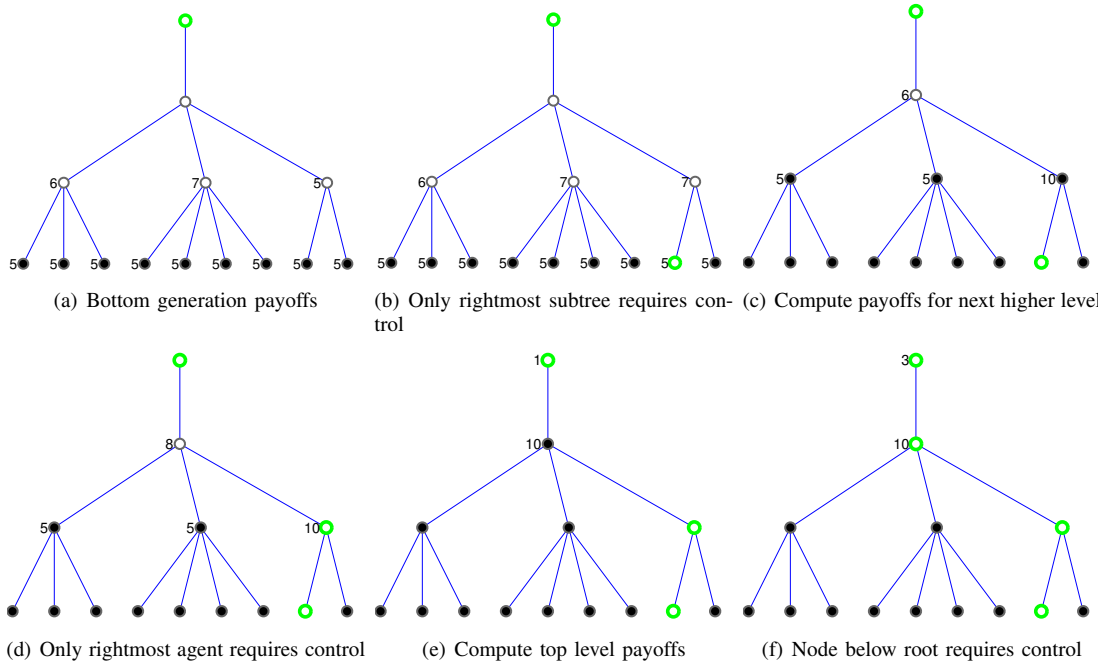
We seek the minimal set of control agents needed to shift the whole network to strategy A. The network has 14 agents spanning 3 generations from the perspective of the chosen root. Although steps 6 to 15 in Algorithm 1 are necessary in the general to ensure that no agents playing strategy A switch to strategy B, they have no effect in this example, and thus we focus on the second part of the algorithm, steps 17-21, which we can think of as computing the minimum number of control agents needed in each level assuming that all agents in higher levels play strategy A. Initially $\hat{\mathcal{L}}$ includes only the root agent. Starting at the bottom level of the tree, the agents are switched to strategy B, payoffs are computed (a), and then control is added to the child agents until all children have lower payoffs than their respective parents (b). In this example, only the rightmost subtree in the bottom generation requires additional control. The process is repeated in the next level (c), where we see that controlling the rightmost agent will give the parent a higher expected payoff than any of the remaining children (d). One generation higher (e), we must also add the single child of the root agent to $\hat{\mathcal{L}}$. The result of Algorithm 1 is the four-agent control set shown in (f). Finally, we simulate the game after removing the root agent from the control set and find that the three agents excluding the root result in all agents switching from B to A. This is indeed a minimal control set, which can be verified by trying all possible control sets since it is a relatively small tree. In the next section, we run several simulations to analyze the performance of the algorithms on some common evolutionary games.

B. Simulations

To measure the tightness of the bounds computed by the proposed algorithms, we generated 500 random trees with up to 15 agents starting with random initial strategies, small enough that we can compare the results of the proposed algorithm to the optimal solutions for a large number of trials. Minimal control sets $\mathcal{L}^*$ are found by exhaustive search, that is, by trying all possible one-agent control sets, then all two-agent sets, and so forth until a solution is obtained. We tested the algorithm on three different well-known evolutionary games: the prisoner's dilemma (PD), snowdrift (SD), and stag hunt (SH), for which we used the following payoff matrices in addition to (4):

$$M_{PD} = \begin{matrix} \\ A \\ B \end{matrix} \begin{matrix} \begin{matrix} A & B \end{matrix} \\ \begin{pmatrix} 4 & -1 \\ 5 & 0 \end{pmatrix} \end{matrix}, \quad M_{SH} = \begin{matrix} \\ A \\ B \end{matrix} \begin{matrix} \begin{matrix} A & B \end{matrix} \\ \begin{pmatrix} 2 & 0 \\ 1 & 1 \end{pmatrix} \end{matrix}.$$

Table I shows the simulation results. Columns 2 and 3 show the mean difference between the size of the true minimal-agent control set and the upper and lower bounds, respectively. The final column lists the percentage of runs for which the approximate set $\hat{\mathcal{L}}$ returned by Algorithm 1 was optimal. We see that the algorithm yielded minimal control sets in the majority of PD and SD games, but performed slightly worse in the SH game, possibly due to the fact that strategy A is a symmetric Nash equilibrium of M_{SH} and thus the algorithm may conservatively add control when none is needed.



(a) Bottom generation payoffs (b) Only rightmost subtree requires control (c) Compute payoffs for next higher level

(d) Only rightmost agent requires control (e) Compute top level payoffs (f) Node below root requires control

Fig. 1. Step-by-step application of algorithm 1 to a snowdrift game on a small tree network in which all agents initially play strategy B. Circles filled with black indicate strategy B while white indicates strategy A. Starting from the bottom level, control is added to the agents (indicated by green circles) until all children have lower payoffs than their parents, assuming that all agents in higher levels play strategy A.

TABLE I
SIMULATION RESULTS.

Game	mean($\lvert\hat{\mathcal{L}}\rvert - \lvert\mathcal{L}^*\rvert$)	mean($\lvert\mathcal{L}^*\rvert - \lvert\check{\mathcal{L}}\rvert$)	% Opt
PD	0.34	1.31	71.8
SD	0.26	1.22	76.2
SH	0.87	0.32	38.0

V. CONCLUSIONS AND FUTURE WORK

In this paper we introduced algorithms to compute upper and lower bounds on the minimum number of control agents needed to drive an evolutionary game on a tree network to a uniform desired strategy. The upper bound algorithm also produces an approximately minimal set of control agents. An interesting topic for future research is the use of dynamic control sequences, which in some cases may make it possible to control a network with an even fewer agents. There may also be situations where a non-uniform final strategy state is desired. The broader research goal is to extend these results to more complex networks, and we are currently developing a method for computing bounds on the solution to the minimal-agent control problem for arbitrary networks that uses graph partitioning to decompose a larger problem into several smaller problems.

REFERENCES

[1] F. Bagnoli, R. Rechtman, and S. El Yacoubi. Control of cellular automata. *Physical Review E*, 86(6):066201, 2012.
[2] Ming Cao, Shuo Zhang, and M Kanat Camlibel. A class of uncontrollable diffusively coupled multiagent systems with multichain topologies. *Automatic Control, IEEE Transactions on*, 58(2):465–469, 2013.
[3] C. Castellano, S. Fortunato, and V. Loreto. Statistical physics of social dynamics. *Reviews of Modern Physics*, 81(2):591, 2009.
[4] D. Cheng, F. He, H. Qi, and T. Xu. Modeling, analysis and control of networked evolutionary games. *To appear in IEEE Transactions on Automatic Control*.
[5] W. Du, H. Zhou, Z. Liu, and X. Cao. The effect of pinning control on evolutionary prisoner's dilemma game. *Modern Physics Letters B*, 24(25):2581–2589, 2010.
[6] J. H. Fowler and N. A. Christakis. Cooperative behavior cascades in human social networks. *Proceedings of the National Academy of Sciences*, 107(12):5334–5338, 2010.
[7] M. J. Fox and J. S. Shamma. Population games, stable games, and passivity. *Games*, 4(4):561–583, 2013.
[8] N. Monshizadeh. *Model reduction and control of complex systems*. PhD thesis, University of Groningen, 2013.
[9] M. Nakamaru, H. Matsuda, and Y. Iwasa. The evolution of cooperation in a lattice-structured population. *Journal of theoretical Biology*, 184(1):65–81, 1997.
[10] Martin Nowak, Karl Sigmund, et al. A strategy of win-stay, lose-shift that outperforms tit-for-tat in the prisoner's dilemma game. *Nature*, 364(6432):56–58, 1993.
[11] F. C. Santos, M. D. Santos, and J. M. Pacheco. Social diversity promotes the emergence of cooperation in public goods games. *Nature*, 454(7201):213–216, 2008.
[12] G. Szabó and G. Fáth. Evolutionary games on graphs. *Physics Reports*, 446(4):97–9216, 2007.