2022 American Control Conference (ACC)
Atlanta, USA, June 8-10, 2022

Leader Selection for Strong Structural Controllability in Networks using Zero Forcing Sets

Waseem Abbas, Mudassir Shabbir, Yasin Yazıcıoğlu, and Xenofon Koutsoukos

Abstract— **This paper studies the problem of computing a minimum zero forcing set (ZFS) in graphs. The problem is important because it is directly related to the leader selection problem for the strong structural controllability of networks defined over graphs. Computing a ZFS of minimum size is a well-known NP-hard problem in general. We show that previously known greedy heuristic could give arbitrarily bad solutions for some graphs. We study the problem on trees and present an optimal algorithm to compute a minimum ZFS in linear time in trees. For general graphs, we present a game-theoretic solution by formalizing the minimum ZFS problem as a potential game. Finally, we numerically evaluate our results on random graphs.**

Index Terms— **Strong structural controllability, zero forcing sets, dynamics over graphs.**

I. Introduction

Network controllability is related to controlling a network of agents as desired by injecting external input signals through a subset of agents called *leaders*. Network controllability has been a central theme in network control systems and network science due to many applications in multi-robot systems, sensor networks, social networks, power systems, and other domains. Several aspects of this problem have been studied in the literature, including graph-theoretic characterization of network controllability, methods to quantify network controllability, and energy-based control of networks (e.g., [1], [2], [3]). Another crucial aspect is the computation of the minimum set of leader agents to completely control the network, or the minimum leader selection problem (e.g., [4], [5], [6]).

Network topology-based approaches offer remarkable insights into the minimum leader selection problem. In particular, it has been shown that the notion of *zero forcing sets (ZFS)* in graphs adequately characterizes the leader selection problem [7], [8], [9], [10]. To understand ZFS, consider a graph whose nodes can be colored either white or black. A black node can change the color of its white neighbor to black following some rules. A ZFS is a set of initial black nodes which render the entire graph black by iteratively applying the color-changing rule. We note that ZFS provides conditions on the leader nodes for the network to be strong structurally controllable, which means that controllability is guaranteed regardless of edge weights (as long as they are non-zero).

W. Abbas is with the Systems Engineering Department at the University of Texas at Dallas, Richardson, TX (Email: `waseem.abbas@utdallas.edu`). M Shabbir and X. Koutsoukos are with the Electrical Engineering and Computer Science Department at Vanderbilt University, Nashville, TN (Emails: {`mudassir.shabbir`, `xenofon.koutsoukos`}`@vanderbilt.edu`). Y. Yazıcıoğlu is with the Department of Electrical and Computer Engineering at the University of Minnesota, Minneapolis, MN, USA (Email: `ayasin@umn.edu`).

We are interested in computing the minimum ZFS in undirected graphs, which is a known NP-hard problem [11]. The ZFS problem has been an active research topic in graph theory. However, most of the research in ZFS revolves around finding useful upper and lower bounds on the size of minimum ZFS, also called the zero forcing number, and then refining these bounds for specific graph families (e.g., [12], [13], [14]). There are known algorithms in the literature that compute a minimum ZFS, for instance, wavefront algorithm [15], [16], and Integer Programming formulations [14]; however, they are not suitable for large networks due to their exponential time complexities. As a result, one has to rely on efficient heuristics that return small-sized ZFS without large time-complexity, for instance, greedy heuristic.

This paper studies various aspects of the minimum ZFS problem, including a linear-time algorithm to compute minimum ZFS in trees and other graph families and efficient heuristics for general graphs. The main contributions are:

- We show that a greedy solution, though it typically performs well, could give ZFS whose size is arbitrarily large compared to the minimum ZFS.
- We analyze the effect of removing specific nodes and edges on the minimum ZFS of a graph. We use this analysis to design a linear-time algorithm to compute an optimal ZFS in trees.
- We present heuristics for computing ZFS by formulating the problem as a potential game and then using a learning solution in games.
- Finally, we also numerically evaluate our results, illustrating the usefulness of the game-theoretic approach.

The rest of the paper is organized as follows: Section II presents notation and preliminaries. It also provides examples in which greedy ZFS solution could be arbitrarily bad from the optimal. Section III discusses the minimum ZFS problem in trees. Section IV provides a game-theoretic formulation of the game and also discusses heuristics. Section V contains numerical evaluation and Section VI concludes the paper.

II. Notations and Preliminaries

We consider a multiagent network modeled by an undirected graph $G = (V, E)$, where V is the set of nodes representing agents and $E \subseteq V \times V$ is the edge set representing interconnections between agents. The cardinality of the given set V is denoted by $|V|$. The edge between u and v is denoted by an unordered pair (u, v). The *neighborhood* of node u is the set $N_u = \{v \in V | (u, v) \in E\}$. The *degree* of a node u is defined as the size of its neighborhood, i.e., $\deg(u) = |N_u|$. A *path* of length k in a graph G is a sequence of nodes, $P_k := < v_0, v_1, v_2, \ldots, v_k >$, where (v_i, v_{i+1}) is an edge in G for all $0 \leq i \leq k-1$. The *distance* $d(u,v)$ between nodes u and v is the number of edges in the shortest path between them. The *diameter* of G is the maximum distance between any two nodes in the graph. A node v in a graph G with $\deg(v) = 1$ is called a *leaf* node. We define a family of symmetric matrices associated with graph $G = (V, E)$, where $|V| = n$, as following:

$$\mathcal{M}(G) = \{M \in \mathbb{R}^{n\times n} \mid M = M^\top, \text{ and for } i \neq j, \; M_{ij} \neq 0 \Leftrightarrow (i,j) \in E(G)\}. \tag{1}$$

Next, we define a finite dimensional leader-follower system on $G = (V, E)$ as follows:

$$\dot{x}(t) = Mx(t) + Bu(t). \tag{2}$$

Here $x(t) \in \mathbb{R}^n$ is the system state, $u(t) \in \mathbb{R}^m$ is the input, $M \in \mathcal{M}(G)$ (as in (1)), and $B \in \mathbb{R}^{n\times m}$ is the input vector describing which nodes are leaders (i.e., input nodes). For B, let $V_L = \{\ell_1, \ell_2, \cdots, \ell_m\} \subseteq V = \{v_1, v_2, \cdots, v_n\}$, then

$$[B]_{ij} = \begin{cases} 1 & \text{if } v_i = \ell_j, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

For a graph G, matrices in $\mathcal{M}(G)$ capture a broad class of system matrices defined on undirected graphs and encountered in several applications. For example, the adjacency and Laplacian matrices of G also belong to $\mathcal{M}(G)$. We are interested in finding a minimum set of leader nodes that make such systems strong structurally controllable.

A. Strong Structural Controllability and Zero Forcing

Consider a system (2) defined on a graph $G = (V, E)$. We say that the pair (M, B) is a *controllable pair* whenever the rank of the controllability matrix $\Gamma(M, B)$, defined below, is n (i.e., full rank).

$$\Gamma(M, B) = \left[\begin{array}{ccccc} B & MB & M^2B & \cdots & M^{n-1}B \end{array} \right]. \tag{4}$$

It basically means that for a given pair (M, B), there always exists an input that can drive the system from any initial state $x(t_0)$ to any final state $x(t_f)$.

Definition (*Strong Structural Controllability)* A graph $G = (V, E)$ with a given set of leaders $V_L \subseteq V$ (and the corresponding B matrix) is *strong structurally controllable* if and only if (M, B) is a controllable pair for all $M \in \mathcal{M}(G)$.

We are interested in finding the minimum set of leaders rendering the graph strong structurally controllable with a given set of leader nodes V_L. The notion of zero forcing set (ZFS) is instrumental in this regard. We define it below and then relate it to the strong structural controllability of G.

Definition (*Zero forcing Process*) Given a graph $G = (V, E)$ whose nodes are initially colored either *black* or *white*. Consider the following node color changing rule: *If $v \in V$ is colored black and has exactly one white neighbor u, change the color of u to black.* Zero forcing process is the application of the above rule until no further color changes are possible.

If the color of white node u is changed to black due to some black node v, we say that v *infected* u.

Definition (*Derived Set*) Consider a graph $G = (V, E)$ with $V' \subseteq V$ be the set of initial black nodes. Then, the set of black nodes obtained at the end of the zero forcing process is the derived set, denoted by $der(G, V')$, or simply $der(V')$ when the context is clear.

The set of initial black nodes V' is also referred to as the *input set*. For a given input set, the derived set is unique [17].

Definition (*Zero Forcing Set (ZFS)*) Consider a graph $G = (V, E)$ and $V' \subseteq V$. Then, V' is a ZFS if and only if $der(G, V') = V$. We denote a ZFS of G by $\mathcal{Z}(G)$. The size of the minimum zero forcing set is called the *zero forcing number*, and denoted by $\zeta(G)$.

The ZFS and derived set ideas are illustrated in Figure 1.

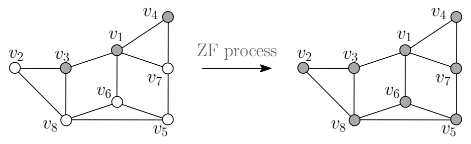


Fig. 1: $V' = \{v_1, v_4, v_7\}$ is the input set. Since $der(V') = V$, the input set is also a ZFS.

ZFS characterizes the leader selection for strong structural controllability of G. It is shown in [7] that a graph $G = (V, E)$ with a given leader set $V_L \subseteq V$ is strong structurally controllable if and only if V_L is a ZFS of G [7, Thm. IV.8, Prop. IV.9]. Thus, solving the minimum leader selection for strong structural controllability is equivalent to finding the minimum zero forcing set. The next subsection reviews the ZFS computation results and shows graphs for which greedy heuristics can return ZFS of very large sizes.

B. ZFS Computation and Greedy Heuristics

Computing $\zeta(G)$ is an NP-hard problem in general [11]. One of the best known algorithms to compute $\zeta(G)$ (and minimum ZFS) is the *wavefront algorithm* proposed in [15] and analyzed in [16]. It is also shown in [16, Theorem 5] that in the worst case, the wavefront algorithm is the same as enumerating all possible subsets of vertices. Other competitive approaches based on integer programming, satisfiability (SAT)-based models, branch-and-bound techniques have also been presented, whose performances rely on various graph characteristics such as the existence of certain subgraphs, density and other structural constraints [16], [14], [18]. Though these methods are exact, they are practically feasible only for small graphs due to their large time complexities. Thus, there is a need to design more practical heuristics that return small ZFS.

We can utilize a simple *greedy* approach to iteratively select a ZFS [14]. The main idea is that in each iteration, change the color of a white node to black to maximize the size of the derived set. Continue this process until a ZFS is obtained. As a final step, remove redundant nodes in a ZFS to achieve a minimal ZFS.

Greedy Heuristics for ZFS

```
1 : given: $G$
2 : initialization: $Z = \emptyset$.
3 : while $|der(Z)| < n$
4 :     $v^* = \arg\max_{v_i \in V \setminus Z} der(Z \cup \{v_i\})$
5 :     $Z = Z \cup \{v^*\}$
6 : end while
    --------- removing redundancies -------
7 : for all $v_i \in Z$
8 :     if $|der(Z \setminus \{v_i\})| = n$
9 :         $Z = Z \setminus \{v_i\}$
10 :    end if
11 : end for
12 : return $Z$
```

The solution returned by the simple greedy approach (lines 1–6) above could contain redundant nodes, and as a result, the ZFS returned might not be minimal. Therefore, we can improve the solution by removing the redundant nodes (lines 7–11). Moreover, the greedy heuristic performs well generally; however, there are instances where the difference between greedy and optimal solutions can be quite large.

Proposition 2.1: There exists graphs for which $|\mathcal{Z}_g(G)|/\zeta(G)$ can be arbitrarily large. Here, $\mathcal{Z}_g(G)$ is a greedy ZFS solution.

Proof: Consider $G = (X \cup Y, E)$, where X and Y are distinct sets of n and m nodes, respectively. Without the loss of generality, assume $n \geq m$. Nodes in X induce a path $< x_1, x_2, \cdots, x_n >$, and similarly, nodes in Y induce a path $< y_1, y_2, \cdots, y_m >$. Moreover, each node in X is adjacent to all the nodes in Y. Figure 2 illustrates the graph. It is easy to verify that $Y \cup \{x_1\}$ is a ZFS; thus, $\zeta(G) \leq m+1$. Similarly, $X \cup \{y_1\}$ is a ZFS obtained by the greedy heuristic. To see this, consider all nodes to be white initially. Making any node black would increase the size of the derived set by one. So, select x_1 to be black. In the next iteration, changing any white node to black will again increase the size of the derived set by one. So, include x_2 in the solution. This trend continues for the first n iterations, thus, making all nodes in X black. In the $(n+1)^{th}$ iteration, changing the color of y_1 to black will change the colors of all the remaining nodes to black due to the zero-forcing process. Thus, $\mathcal{Z}_g(G) = X \cup \{y_1\}$ and $|\mathcal{Z}_g(G)| = n+1$. Since we can choose n to be arbitrarily large, $\frac{n+1}{m+1}$ can also be arbitrarily large, which proves the desired claim. ■

III. Optimal ZFS in Trees

This section studies the minimum ZFS problem on trees, which represent the sparsest connected graphs. We present a linear time algorithm to compute a minimum ZFS.[1]

[1]Proofs of all the results in this section are available in [19].

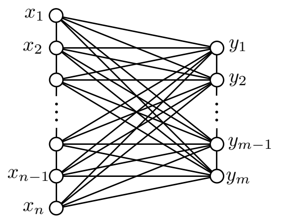


Fig. 2: A graph G with arbitrarily large $\mathcal{Z}_g(G)/\zeta(G)$.

A. ZFS Tree Algorithms

A tree always contains leaf nodes. Since a leaf node has only one neighbor, it can immediately force its only white neighbor if the leaf node is included in a ZFS. This observation gives an easy scheme to select a ZFS in trees: a ZFS consists of all leaf nodes in a tree. The set of all leaf nodes is indeed a ZFS because leaf nodes can force their only neighbors, the predecessors of the leaf nodes, which in turn can force their predecessors until all nodes in the tree are colored black. This is an efficient scheme since all leaf nodes in a tree can be computed in linear time. However, if we run this algorithm on a path graph, we will select both end nodes of a path as a ZFS, while only one end node suffices. Therefore, the ZFS returned is not optimal. Finally, we note that the ZFS returned by this scheme can be significantly worse than the optimal solution.

Remark 3.1: Let $\mathcal{Z}_\ell(T)$ be a ZFS of a tree consisting of leaf nodes. Then, there exist trees whose zero forcing number is almost the half of $|\mathcal{Z}_\ell(T)|$. For instance, consider the tree in Figure 3. A root node u is adjacent to n nodes, each of which is adjacent to a pair of leaf nodes. Thus, there are $2n$ leaf nodes. An optimal ZFS consists of node u and n leaf nodes, as shown in Figure 3. As a result, $\zeta(T) = n+1$ compared to $|\mathcal{Z}_\ell(T)| = 2n$.

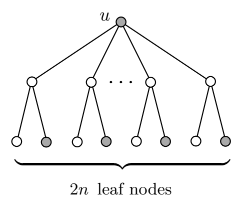


Fig. 3: An optimal ZFS (dark colored nodes) consists of the root node u and n leaf nodes.

We develop some tools to design an efficient algorithm that returns an optimal ZFS for any arbitrary tree in linear time. While these tools are applied to a particular family of graphs here, they are general and may help improve the performance of algorithms for general graphs by reducing the input graph size. We first show that a path of length two can be contracted to an edge without increasing the zero forcing number of the graph.

Proposition 3.2: Let $G = (V, E)$ be a graph, and let u, v be two non-adjacent nodes with a path of length two $< u, w, v >$, and $\deg(w) = 2$. Let $H = (V', E')$ be an other graph, where

$$V' = V \setminus \{w\}, \quad E' = E \setminus \{(u,w),(w,v)\} \cup \{(u,v)\},$$

i.e., H is constructed from replacing the two length path $< u, w, v >$ with an edge (u, v). Then, $\zeta(G) \geq \zeta(H)$.

Remark 3.3: Given the conditions of proposition 3.2, in general, it is not true that $\zeta(G) = \zeta(H)$, that is, the degree two vertices can not be collapsed in general without affecting the zero forcing number. To claim that, we need slightly more strict conditions.

In the following, we show that in a special case, when one of the nodes on a path of length two is a leaf, the path can be contracted to an edge, and the zero forcing number will remain unchanged.

Lemma 3.4 (Collapsing Lemma): Let $< u, w, v >$ be a path in a graph $G = (V, E)$, with $\deg(w) = 2, \deg(v) = 1$. Let $H = (V', E')$ be an other graph, where

$$V' = V \setminus \{w\}, \quad E' = E \setminus \{(u.w),(w,v)\} \cup \{(u,v)\},$$

i.e., H is constructed from replacing the two length path $< u, w, v >$ with an edge (u, v). Then, $\zeta(G) = \zeta(H)$.

A proof of the above Lemma is available in [19].

Definition A pendant $S_k = (V_s, E_s)$ in a graph $G = (V, E)$ is an induced star graph, where $V_s = \{a, b_1, \cdots, b_k\}$ and $E_s = \{(a, b_i)| \ 1 \leq i \leq k\}$ with the added condition that all b_i's are leaf nodes in G.

We observe that most of the nodes of any arbitrary pendant of a graph must be included in a ZFS, as we state in the following proposition.

Proposition 3.5: Let S_k be a pendant in graph G with nodes $a, b_1, b_2, \ldots, b_k, k > 1$, where b_i are the leaf nodes. At least $k - 1$ of the leaf nodes of S_k must be in a ZFS of G.

Based on the proposition 3.5, we outline a scheme to reduce the size of a graph by removing a pendant from a graph while computing its effect on the zero forcing number.

Lemma 3.6 (Pruning Lemma): Let S_k be a pendant in graph G with nodes $a, b_1, b_2, \ldots, b_k, k > 1$ where b_i are the leaf nodes. Let H be constructed from G by removing the nodes $a, b_1, b_2, \ldots, b_k$ of G. Then, $\zeta(G) = \zeta(H) + k - 1$.

Next, using the above results, we present an optimal algorithm to compute the minimum ZFS of a tree.

Theorem 3.7: There is a linear time algorithm to compute an optimal ZFS of any tree graph.

A proof of the the above theorem is available in [19].

IV. ZFS Heuristics using Potential Games

In this section, we present a game-theoretic approach for finding an optimal ZFS in a distributed manner in arbitrary graphs. Given a graph $G = (V, E)$ with n nodes, $V = \{v_1, \ldots, v_n\}$, let $a \in \{0, 1\}^n$ be an indicator of the node colors. Accordingly, $a_i = 1$ if v_i is black, and $a_i = 0$ if v_i is white. Next, we define a function, $\phi(a)$, whose maximization is equivalent to finding an optimal ZFS as we will show in Lemma 4.1:

$$\phi(a) = \frac{1}{n}\left(|der(a)| - \sum_{i=1}^{n} a_i\right), \tag{5}$$

which is equal to $1/n$ times the size of the derived set, $der(a)$, minus the number of black nodes when the colors are assigned as per a. We use $der(a)$ to denote the derived set of black nodes indicated by a, and the scaling term $1/n$ is used for keeping $\phi(a)$ finite regardless of the network size.

Lemma 4.1: Let $G = (V, E)$ be a connected graph and let $a \in \{0, 1\}^n$ represent the node colors, i.e., $a_i = 1$ if v_i is black. A vector $a \in \{0, 1\}^n$ is a maximizer of ϕ in (5), i.e., $\phi(a) \geq \phi(a'), \forall a' \in \{0, 1\}^n$, if and only if a indicates an optimal ZFS.

Proof: $(\Rightarrow:)$ Let $a \in \{0, 1\}^n$ be a maximizer of ϕ in (5). We will first show that a necessarily indicates a ZFS, i.e., $der(a) = V$. For the sake of contradiction, suppose that this is not true and $der(a) \subset V$. Then, pick any $v_j \notin der(a)$ and define a new vector $a' \in \{0, 1\}^n$ as follows: $a'_i = 1$ if $a_i = 1$ or $v_i \notin der(a) \setminus \{v_j\}$, and $a'_i = 0$ otherwise. In other words, the black nodes under a' comprise of all the black nodes under a and all the nodes other than v_j that are not included in the derived set $der(a)$. Accordingly, the resulting increase in the number of black nodes is

$$\sum_{i=1}^{n} a'_i - \sum_{i=1}^{n} a_i = n - |der(a)| + 1. \tag{6}$$

Since every black node under a is also black under a', we have $der(a) \subseteq der(a')$. Furthermore, since every node other than v_j that are not included in the derived set $der(a)$ are also selected as black, then either $der(a') = V$ or $der(a') = V \setminus \{v_j\}$. However, $der(a') = V \setminus \{v_j\}$ is not possible since it implies that the zero forcing process ends with a single white node v_j, which is guaranteed to be the only white neighbor of a black node in the end since G is connected. Such a node v_j must become infected under the zero forcing process. Hence, $der(a') = V$. Accordingly,

$$\phi(a') - \phi(a) = \frac{1}{n}\left(n - |der(a)| + \sum_{i=1}^{n} a'_i - \sum_{i=1}^{n} a_i\right). \tag{7}$$

Note that (6) and (7) together imply $\phi(a') > \phi(a)$, which contradicts with a being a maximizer of $\phi(a)$. Hence, a must indicate a ZFS.

Next, we will show that a must indicate an optimal ZFS, i.e., a ZFS with the fewest possible number of nodes. For the sake of contradiction, suppose that a does not correspond to an optimal ZFS. Then, there exists $a' \in \{0, 1\}^n$ which corresponds to a ZFS and has fewer black nodes compared to a. Accordingly, $|der(a)| = |der(a')|$ and $\sum_{i=1}^{n} a'_i < \sum_{i=1}^{n} a_i$, which imply $\phi(a') > \phi(a)$. Hence, once again we obtain a contradiction with a being a maximizer of ϕ. Consequently, any maximizer of ϕ is an optimal ZFS.

$(\Leftarrow:)$ If any two vectors, a and a', both indicate optimal ZF sets, then $\phi(a) = \phi(a')$ for ϕ in (5) since $|der(a)| = |der(a')| = n$ and, by definition, both a and a' have the minimum number of leaders among the ZF sets (hence $\sum_{i=1}^{n} a_i' = \sum_{i=1}^{n} a_i$). Since every optimal ZFS have equal ϕ and we have already shown that any maximizer of $\phi(a)$ is necessarily an optimal ZFS, we conclude that every optimal ZFS is a maximizer of ϕ. ■

Based on Lemma 4.1, an optimal ZFS can be obtained by searching for a maximizer of $\phi(a)$ in (5). Such a maximization can be achieved in a distributed manner by using a game-theoretic formulation (e.g., [20]). More specifically, the problem of finding an optimal ZFS can be formulated as a potential game with the potential function $\phi(a)$ and a learning algorithm such as log-linear learning [21] can be used to find an optimal a. Before presenting such a game-theoretic approach, we first provide some preliminaries.

A. Game Theory Basics

A finite strategic game $\Gamma = (I, A, U)$ has three components: (1) a set of players (agents) $I = \{1, 2, \dots, n\}$, (2) an action space $A = A_1 \times A_2 \times ... \times A_n$, where each A_i is the action set of player i, and (3) a set of utility functions $U = U_1, U_2, \dots, U_n$, where each $U_i : A \to \mathbb{R}$ is a mapping from the action space to real numbers. For any action profile $a \in A$, we use a_{-i} to denote the actions of players other than i. Using this notation, an action profile a can also be represented as $a = (a_i, a_{-i})$.

A class of games that is widely utilized in cooperative control problems is the *potential games*. A game is called a potential game if there exists a potential function, $\phi : A \to \mathbb{R}$, such that the change of a player's utility resulting from its unilateral deviation from an action profile equals the resulting change in ϕ. More precisely, for each player i, for every a_i, $a_i' \in A_i$, and for all $a_{-i} \in A_{-i}$,

$$U_i\left(a_i', a_{-i}\right) - U_i\left(a_i, a_{-i}\right) = \phi\left(a_i', a_{-i}\right) - \phi\left(a_i, a_{-i}\right). \quad (8)$$

In game-theoretic learning, the agents start with arbitrary initial actions and follow a learning algorithm to update their actions based on past observations in a repetitive play of the game. For potential games, noisy best-response type algorithms such as *log-linear learning* (LLL) [21] can be used to have the agents spend most of their time at the global maximizers of $\phi(a)$. More specifically, LLL induces an irreducible and aperiodic Markov chain over the action space A such that the limiting distribution, μ_ϵ, satisfies

$$\lim_{\epsilon \to 0^+} \mu_\epsilon(a) > 0 \iff \phi(a) \geq \phi(a'), \forall a' \in A, \quad (9)$$

where $\epsilon > 0$ is the noise parameter of LLL.

B. ZFS Game

We formulate the problem of finding an optimal ZFS as a game, $\Gamma_{ZFS} = (I, A, U)$, where the set of players is the set of nodes, i.e., $I = V$, and the action space is $A = \{0, 1\}^n$. Accordingly, the action of each agent v_i is a binary variable indicating its initial color in the zero forcing process, i.e., black ($a_i = 1$) or white ($a_i = 0$). Finally, we need to define the utility functions $U_i(a)$ such that $\Gamma_{ZFS} = (I, A, U)$ is a potential game whose potential function is $\phi(a)$ in (5). While there are also other methods to design such utility functions (e.g., wonderful life utility [22]), one choice is to set all the utilities equal to the global objective, i.e.,

$$U_i(a) = \frac{1}{n}\left(|der(a)| - \sum_{i=1}^{n} a_i\right), \ \forall i \in I. \quad (10)$$

One can easily verify that the resulting game, Γ_{ZFS}, is a potential game with the potential function $\phi(a)$ in (5), i.e., the utilities in (10) satisfy (8). Accordingly, an optimal ZFS can be found by employing LLL in a repetitive play of the resulting game, Γ_{ZFS}.

Theorem 4.2: Let Γ_{ZFS} be the ZFS game on a connected $G = (V, E)$. Then, log-linear learning (LLL) induces a Markov chain over the action space $A = \{0, 1\}^n$ whose limiting distribution, μ_ϵ, satisfies

$$\lim_{\epsilon \to 0^+} \mu_\epsilon(a) > 0 \iff a \text{ corresponds to an optimal ZFS}, \quad (11)$$

where $\epsilon > 0$ is the noise parameter of LLL.

A proof of Theorem 4.2 is available in [19]. In light of Theorem 4.2, when a is updated via LLL with a sufficiently small noise parameter, ϵ, it indicates an optimal ZFS with a very high probability as the number of iterations goes to infinity. However, since there is only a finite amount of time to search for an optimal ZFS in real-life problems, we propose an LLL-based heuristic that has three steps: 1) following LLL to update a for a finite number of iterations, 2) if the resulting a does not indicate a ZFS, then switching all the nodes in $V \setminus der(a)$ to black (a becomes a ZFS), and 3) removing redundant black nodes in a.

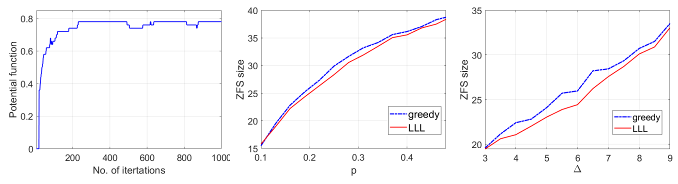
Log-Linear Learning (LLL) Based Heuristic for ZFS

```
1 : given: $G = (V, E)$, #iterations $\bar{k}$ (large), noise $\epsilon > 0$ (small)
2 : initialization: arbitrary $a \in \{0, 1\}^{|V|}$
    ----- running LLL for $\bar{k}$ iterations ----
3 : for $k = 1$ to $\bar{k}$
4 :   Pick a random node $v_i$.
5 :   Randomize $a_i$ based on the utility function in (10):
        $\Pr[a_i = a_i'] \sim \exp\left(\frac{U_i(a_i', a_{-i})}{\epsilon}\right), \forall a_i' \in \{0, 1\}.$
6 : end for
7 : $Z = \{v_i \in V \mid a_i = 1\}$,
    -- adding leaders if $Z$ is not a ZFS --
8 :  $Z = Z \cup (V \setminus dset(Z))$,
    --------- removing redundancies -------
9 : for all $v_i \in Z$
10 :     if $|der(Z \setminus \{v_i\})| = n$
11 :         $Z = Z \setminus \{v_i\}$
12 :     end if
13 : end for
14 : return $a$
```

V. Numerical Evaluation

In this section, we compare the LLL-based ZFS solution with the greedy solution. First, we compute the ZFS of graphs discussed in Proposition 2.1 using LLL. These are the graphs for which the greedy heuristic performed poorly. In our experiments, the LLL solution returned a ZFS, whose size is at most one more than the minimum ZFS. For instance, consider $G = (X \cup Y, E)$ (as in Proposition 2.1), where $|X| = 40$ and $|Y| = 10$, the greedy heuristic returned ZFS with 41 nodes, whereas LLL returned ZFS with 11 nodes, which is optimal. Figure 4(a) illustrates the potential function as a function of the number of iterations in LLL for the above example. The value of ϵ used is 0.004. As shown in Figure 4(a), after about 250 iterations the potential function equals 0.78 most of the time, which is the maximum possible value of (5) for this example. In particular, $\phi(a) = 0.78$ only when a corresponds to an optimal ZFS, i.e., $\sum_{i=1}^{n} a_i = 11$ and $|der(a)| = n = 50$.

Next, we consider Erdös-Rényi (ER) random graphs with $n = 50$ nodes. Figure 4(b) plots the size of ZFS returned by greedy and LLL solutions as functions of p, where p is the probability of having an edge between any two nodes in the graph. Each point on the plots is an average of 25 randomly generated instances. In LLL solution, $\epsilon = 0.005$ and the 2000 iterations are performed in each instance. We observe that LLL produces ZFS of smaller size compared to the greedy solution. Similarly, in Figure 4(c), the same results are plotted for the Δ-disk proximity graphs with $n = 50$ nodes. In such a graph, nodes u and v are adjacent whenever the Euclidean distance between them is at most Δ. In our simulation, nodes are randomly placed in a planar region of area 15×15 [unit length]^2. Again, each point on the plots is an average of 25 randomly generated instances. For LLL, we used $\epsilon = 0.008$ and 2000 iterations in each instance. Again, the LLL solution outperforms the greedy solution. We note that greedy heuristics typically provide ZFS of small sizes, which are not too far from the optimal solution. However, as illustrated in the plots, LLL based solution can perform even better than the greedy when ϵ is chosen properly and the algorithm runs for a sufficient number of iterations.



(a) (b) ER (c) Δ-disk

Fig. 4: Comparison of the greedy and LLL based heuristics for ZFS in (a) ER graphs and (b) Δ-disk graphs. (c) An example of potential function as a function of number of iterations in LLL.

VI. Conclusion

We studied the minimum ZFS problem in undirected graphs to solve the minimum leader selection problem for strong structural controllability in undirected networks. The minimum ZFS problem is computationally challenging, and a greedy heuristic could give arbitrarily bad solutions in some cases. We provided a linear time algorithm to optimally solve the problem in trees. Further, we formulated the problem as a potential game and utilized log-linear learning to solve the game. The numerical evaluation showed that the LLL based method performed better than the greedy heuristic. There are some interesting directions to further pursue, for instance, if two graphs G_1 and G_2 are combined through some operation (e.g., Cartesian product, join), how can we obtain a ZFS of the resulting graph in terms of ZF sets of G_1 and G_2.

References

[1] A. Chapman and M. Mesbahi, "On strong structural controllability of networked systems: A constrained matching approach." in *American Control Conference (ACC)*, 2013, pp. 6126–6131.

[2] F. Pasqualetti, S. Zampieri, and F. Bullo, "Controllability metrics, limitations and algorithms for complex networks," *IEEE Transactions on Control of Network Systems*, vol. 1, no. 1, pp. 40–52, 2014.

[3] T. H. Summers, F. L. Cortesi, and J. Lygeros, "On submodularity and controllability in complex dynamical networks," *IEEE Transactions on Control of Network Systems*, vol. 3, no. 1, pp. 91–101, 2015.

[4] A. Y. Yazicioğlu and M. Egerstedt, "Leader selection and network assembly for controllability of leader-follower networks," in *2013 American Control Conference*. IEEE, 2013, pp. 3802–3807.

[5] J. Ruths and D. Ruths, "Control profiles of complex networks," *Science*, vol. 343, no. 6177, pp. 1373–1376, 2014.

[6] F. Lin, M. Fardad, and M. R. Jovanović, "Algorithms for leader selection in stochastically forced consensus networks," *IEEE Transactions on Automatic Control*, vol. 59, no. 7, pp. 1789–1802, 2014.

[7] N. Monshizadeh, S. Zhang, and M. K. Camlibel, "Zero forcing sets and controllability of dynamical systems defined on graphs," *IEEE Transactions on Automatic Control*, vol. 59, pp. 2562–2567, 2014.

[8] M. Trefois and J.-C. Delvenne, "Zero forcing number, constrained matchings and strong structural controllability," *Linear Algebra and its Applications*, vol. 484, pp. 199–218, 2015.

[9] S. S. Mousavi, M. Haeri, and M. Mesbahi, "On the structural and strong structural controllability of undirected networks," *IEEE Transactions on Automatic Control*, vol. 63, no. 7, pp. 2234–2241, 2018.

[10] Y. Yazıcıoğlu, M. Shabbir, W. Abbas, and X. Koutsoukos, "Strong structural controllability of diffusively coupled networks: Comparison of bounds based on distances and zero forcing," in *IEEE Conference on Decision and Control (CDC)*, 2020, pp. 566–571.

[11] A. Aazami, "Hardness results and approximation algorithms for some problems on graphs," Ph.D. dissertation, University of Waterloo, 2008.

[12] R. Davila, T. Kalinowski, and S. Stephen, "A lower bound on the zero forcing number," *Discrete Applied Mathematics*, vol. 250, pp. 363–367, 2018.

[13] M. Gentner and D. Rautenbach, "Some bounds on the zero forcing number of a graph," *Discrete Applied Mathematics*, vol. 236, pp. 203–213, 2018.

[14] B. Brimkov, D. Mikesell, and I. V. Hicks, "Improved computational approaches and heuristics for zero forcing," *INFORMS Journal on Computing*, 2021.

[15] S. Butler, L. DeLoss, J. Grout, H. Hall, J. LaGrange, T. McKay, J. Smith, and G. Tims, "Minimum rank library. Sage programs for calculating bounds on the minimum rank of a graph, and for computing zero forcing parameters," 2014, accessed 04-Sep-2021. [Online]. Available: https://github.com/jasongrout/minimum_rank

[16] B. Brimkov, C. C. Fast, and I. V. Hicks, "Computational approaches for zero forcing and related problems," *European Journal of Operational Research*, vol. 273, no. 3, pp. 889–903, 2019.

[17] AIM Minimum Rank Special Graphs Work Group, "Zero forcing sets and the minimum rank of graphs," *Linear Algebra and its Applications*, vol. 428, no. 7, pp. 1628–1648, 2008.

[18] A. Agra, J. O. Cerdeira, and C. Requejo, "A computational comparison of compact MILP formulations for the zero forcing number," *Discrete Applied Mathematics*, vol. 269, pp. 169–183, 2019.

[19] W. Abbas, M. Shabbir, A. Y. Yazıcıoğlu, and X. Koutsoukos, "Leader selection for strong structural controllability in networks using zero forcing sets," *arxiv*, 2022.

[20] J. R. Marden, G. Arslan, and J. S. Shamma, "Cooperative control and potential games," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 39, no. 6, pp. 1393–1407, 2009.

[21] L. E. Blume, "The statistical mechanics of strategic interaction," *Games and Economic Behavior*, vol. 5, no. 3, pp. 387–424, 1993.

[22] K. Tumer and D. H. Wolpert, *Collectives and the design of complex systems*. Springer Science & Business Media, 2004.