On Zero Forcing Sets and Network Controllability – Computation and Edge Augmentation

Waseem Abbas, *Member, IEEE,* Mudassir Shabbir, Yasin Yazıcıoğlu, *Member, IEEE,* and Xenofon Koutsoukos, *Fellow, IEEE*

***Abstract*—This paper studies the problem of computing a minimum zero forcing set (ZFS) in undirected graphs and presents new approaches to reducing the size of the minimum ZFS via edge augmentation. The minimum ZFS problem has numerous applications; for instance, it relates to the minimum leader selection problem for the strong structural controllability of networks defined over graphs. Computing a minimum ZFS is an NP-hard problem in general. We show that the greedy heuristic for the ZFS computation, though it typically performs well, could give arbitrarily bad solutions for some graphs. We provide a linear-time algorithm to compute a minimum ZFS in trees and a complete characterization of minimum ZFS in the clique chain graphs. We also present a game-theoretic solution for general graphs by formalizing the minimum ZFS problem as a potential game. In addition, we consider the effect of edge augmentation on the size of the ZFS. Adding edges could improve network robustness; however, it could increase the size of the ZFS. We show that adding a set of carefully selected missing edges to a graph may actually *reduce* the size of the minimum ZFS. Finally, we numerically evaluate our results on random graphs.**

***Index Terms*—Zero forcing sets, dynamics over graphs, strong structural controllability, edge augmentation.**

I. Introduction

Dynamic coloring of vertices in graphs have recently gained significant attention in network science and dynamical systems due to their broad applicability. Such colorings are frequently used to model and analyze various real-world phenomena, including infection propagation, information spread, and control of networked systems. In dynamic coloring, vertices change their colors in discrete time intervals based on pre-defined rules. *Zero forcing* is a popular coloring process with numerous applications, for instance, in the control of quantum systems, logic circuit design, sensor allocation, network controllability, and information spread in social networks [2]–[5]. The main idea of zero forcing in graphs is that each vertex is initially colored black or white. Then, a black vertex with exactly one white neighbor *forces* its only white neighbor to change the color to black. This process continues until no more color changes are possible. A set of initial black vertices that forces the entire vertex set to become black is called a *zero forcing set* (ZFS) (explained in Section III).

Zero forcing in graphs offers remarkable insights into the controllability of multiagent systems, which 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. Several critical issues related to network controllability have been explored in the literature, including capturing the influence of network topology on controllability, fundamental limits, and practical implications of controlling networks [6]–[11]. Another crucial aspect of network controllability is the computation of the minimum set of leader agents to control the network, also referred to as the *minimum leader selection* problem [12]–[14]. The notion of ZFS in graphs adequately characterizes the leader selection problem and provides conditions to select optimal leader agents to control the network [5], [15]–[17].

As a result, the problem of computing a minimum ZFS in graphs is crucial. It is well known that computing a minimum ZFS, is NP-hard [18]. The ZFS problem has been an active research topic in graph theory. However, most of the research in ZFS revolves around finding upper and lower bounds on the size of the minimum ZFS, called the *zero forcing number*, and refining bounds for specific graph families (e.g., [19]–[21]). In the literature, there are algorithms to compute a minimum ZFS, for instance, the wavefront algorithm [22], [23], and Integer Programming formulations [24]–[26]; however, they are not suitable for large graphs due to their exponential time complexities. Thus, one has to rely on efficient heuristics, such as greedy, to obtain small-sized ZFS.

This paper studies the ZFS problem, including its computation and the effects of edge augmentation on ZFS. In particular, we compute minimum ZFS in trees and other graph families, and also provide heuristics to compute small ZFS in general graphs using game-theory ideas. We also examine how to add edges in a graph to *reduce* the size of the minimum ZFS.

- We show that a greedy solution, which typically performs well, could give a ZFS whose size is arbitrarily large compared to the minimum ZFS (Proposition 3.1).
- We provide a linear-time algorithm to compute minimum ZFS in trees (Theorem 4.7). We also compute minimum ZFS in a class of robust graphs called clique chains (Proposition 4.8).

W. Abbas is with the Systems Engineering Department at the University of Texas at Dallas, Richardson, TX, USA (Email: waseem.abbas@utdallas.edu).

M. Shabbir is with the Computer Science Department at Information Technology University, Lahore, Pakistan and with the Computer Science Department at Vanderbilt University, Nashville, TN, USA (Emails: mudassir.shabbir@vanderbilt.edu).

Y. Yazıcıoğlu is with the Department of Mechanical and Industrial Engineering at Northeastern University, Boston, MA, USA (Email: y.yazicioglu@northeastern.edu).

X. Koustoukos is with the Computer Science Department at Vanderbilt University, Nashville, TN, USA (Emails: xenofon.koutsoukos@vanderbilt.edu).

Some preliminary results appeared in [1].

- We present heuristics based on game-theoretic ideas for computing a small-sized ZFS in graphs. For this, we formulate the problem as a potential game and then use a learning solution in games (Lemma 5.1, Theorem 5.2). We also numerically evaluate our results, illustrating the usefulness of the approach.
- We provide conditions in general and for k-connected graphs, in particular, to strategically add missing edges to reduce the size of their minimum ZFS (Theorem 6.2, Proposition 6.3). Edge augmentation to reduce the zero forcing number of graphs is studied here for the first time to the best of our knowledge.

The rest of the paper is organized as follows: Section II presents notation and preliminaries. Section III reviews the connection between graph controllability and minimum ZFS, and discusses greedy solution to the ZFS problem. Section IV discusses the minimum ZFS in trees and clique chains. Section V provides a game-theoretic formulation of the minimum ZFS problem. Section VI studies the edge augmentation in graphs to reduce the size of the minimum ZFS. Finally, Section VII concludes the paper.

II. Preliminaries

A. Notations and System Model

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 $V' = \{\ell_1, \ell_2, \cdots, \ell_m\} \subseteq V = \{v_1, v_2, \cdots, v_n\}$ be the set of leader nodes, 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.

B. Controllable Graphs and Minimum Leaders Problem

An LTI system in (2) is *controllable* if there exists an input driving the system from any initial state $x(t_0)$ to any final state $x(t_f)$, and we say that (M, B) is a *controllable pair*. A pair (M, B) is controllable if and only if the rank of the controllability matrix $\Gamma(M, B)$ is $|V| = n$ (i.e., full rank).

$$\Gamma(M, B) = \begin{bmatrix} B & MB & M^2B & \cdots & M^{n-1}B \end{bmatrix}. \tag{4}$$

Since leader nodes V' define the input matrix B, with a slight abuse of notations, we sometimes use (M, V') *is controllable* to denote that (M, B) is a controllable pair.

Definition 2.1. (Controllable Graph) *Given a graph $G = (V, E)$ and a leader set $V' \subseteq V$, we say that (G, V')* is controllable *if and only if (M, V') is controllable* for all $M \in \mathcal{M}(G)$ *(as defined in* (1)*).*

We note that the controllable graphs notion is akin to the *strong structural controllability* of undirected networks [5], [16]. Also, in Definition 2.1, if we just require the existence of some $M \in \mathcal{M}$ for which (M, V') is controllable, then G is referred to as *weak structurally controllable* with a leader set V' [10]. Clearly, strong structural controllability is a stronger notion and implies weak structural controllability. We are interested in computing and characterizing the minimum set of leader nodes rendering the graph controllable (strong structurally controllable), which is also referred to the *minimum leader selection problem*, as stated below:

$$V'_{\min} = \underset{\{V' \subseteq V | (G, V') \text{ is controllable}\}}{\arg\min} |V'| \tag{5}$$

In the next section, we define ZFS and review the connections between graph controllability and ZFS.

III. Graph Controllability and Zero Forcing

We begin by defining the *zero forcing process* in graphs.

Definition 3.1. *(*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 a black node v, we say v *forced* u, and denote it by $v \mapsto u$.

Definition 3.2. *(*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 $der(G, V')$.*

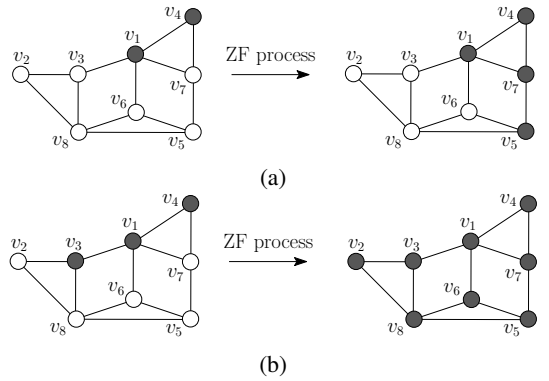
The set of initial black nodes V' is also referred to as the *input set*. For a given input set, the derived set is unique [27].

Definition 3.3. *(*Zero Forcing Set (ZFS)*) Consider a graph $G = (V, E)$ and $V' \subseteq V$. Then, V' is a ZFS if and only if $der(G, V') = V$. The size of the minimum zero forcing set is called the* zero forcing number $\zeta(G)$.

We summarize the zero forcing related notions below:

$V'(G)$	input set / leader nodes
$der(G, V')$	derived set of V';
$Z(G)$	zero forcing set of G;
$\zeta(G)$	zero forcing number of G;

When the context is clear, we drop G from the above notations. Figure 1 illustrates these ideas.



(a)

(b)

Fig. 1: (a) $V' = \{v_1, v_4\}$ and $der(V') = \{v_1, v_4, v_5, v_7\}$. Since $der(V') \subset V$, V' is not a ZFS. (b) $Z = \{v_1, v_4, v_7\}$ is a ZFS as $der(Z) = V$.

ZFS characterizes the leader selection for the graph controllability [5], [15], [16]. A direct consequence of [15, Thm. IV.4, Thm. IV.8, Prop. IV.9] is that a graph $G = (V, E)$ with a leader set $V' \subseteq V$ is controllable (in the sense of Definition 2.1) if and only if V' is a ZFS of G. Thus, among numerous applications, the minimum ZFS problem is significant for network controllability. The following subsection reviews the ZFS computation results and presents graphs for which greedy heuristics can return ZFS of very large sizes.

A. ZFS Computation and Greedy Heuristics

Computing a minimum ZFS and $\zeta(G)$ are NP-hard problems in general [18]. One of the best-known algorithms to compute $\zeta(G)$ (and minimum ZFS) is the *wavefront algorithm* [22], [23]. It is also shown in [23, 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, and 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 [23]–[26]. Though these methods are exact, they are feasible only for small graphs due to their significant 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 [24]. 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.

Algorithm 1: Greedy Heuristic for ZFS

```
1 : given: G
2 : initialization: Z = ∅.
3 : while |der(Z)| < n
4 :     v* = arg max_{v_i ∈ V\Z} der(Z ∪ {v_i})
            (ties are broken arbitrarily.)
5 :     Z = Z ∪ {v*}
6 : end while
    --------- removing redundancies -------
7 : for all v_i ∈ Z
8 :     if |der(Z \ {v_i})| = n
9 :         Z = Z \ {v_i}
10 :    end if
11 : end for
12 : return Z
```

The simple greedy solution (lines 1–6) above could contain redundant nodes, and as a result, the ZFS returned might not be minimal. Therefore, we improve the solution by removing the redundant nodes (lines 7–11). For computation time, we note that for a given set of leader nodes, the derived set can be computed in $O(n+m)$ time, where n and m are the numbers of nodes and edges in G, respectively [23, Proposition 1]. Also, in each (while) iteration, the derived set is computed $O(n)$ times. Finally, the derived set increases by at least one in each (while) iteration. As a result, the time complexity of the greedy heuristic is $O(n^2(n+m))$. The greedy heuristic generally performs well; however, we can construct instances for which the greedy solution performs poorly in the worst case. Proposition 3.1 presents such instances.

Proposition 3.1. *Let $Z_{gr}(G)$ denotes the ZFS returned by the greedy heuristic. Then, there are graphs for which $|Z_{gr}(G)|/\zeta(G)$ can be arbitrarily large.*

Proof. Consider $G = (X \cup Y, E)$, where X and Y are distinct sets of n and $m \leq n$ nodes, respectively. 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, $Z_{gr}(G) = X \cup \{y_1\}$ and $|Z_{gr}(G)| = n+1$. Since we can choose n to be arbitrarily large, $\frac{n+1}{m+1}$ can also be arbitrarily large, which proves the desired claim. ∎

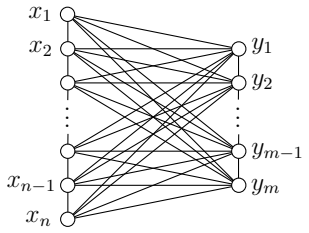


Fig. 2: A graph G with $\frac{|Z_{gr}|}{\zeta} = \frac{n+1}{m+1}$.

IV. Optimal ZFS in Trees and Clique Chains

This section studies the minimum ZFS problem on two important graph families, trees and clique chains. Trees represent the sparse connected graphs, while clique chains belong to a generally dense family of graphs. For both of these extreme families, we provide algorithms to compute ZFS optimally. Here, we use the word "optimal" to mean two things: the ZFS returned is the smallest possible, and the algorithms are asymptotically the most efficient in computational complexity.

A. ZFS Tree Algorithms

A tree is an acyclic-connected graph and 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 4.1. *Let $Z_\ell(T)$ be a ZFS of a tree consisting of leaf nodes. Then, there exist trees whose zero forcing number is almost half of $|Z_\ell(T)|$. 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. There are $2n$ leaf nodes. A minimum ZFS consists of node u and n leaf nodes, as shown in Figure 3. As a result, $\zeta(T) = n+1$ compared to $|Z_\ell(T)| = 2n$.*

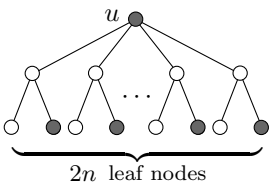


Fig. 3: A minimum ZFS (dark colored nodes) consists of root node u and n leaf nodes.

ZFS of tree graphs is also equivalent to another well-known graph parameter: path cover number [28]. Some previous works report algorithms to compute the path cover number, and hence, the zero forcing number of trees [28], [29]. However, these algorithms are discussed primarily from the existence perspective without the complexity analysis and implementation details. In the following, we present a simple linear-time algorithm to compute a minimum ZFS in trees. Though our results are applied to tree graphs here, they may also help in improving 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.

Lemma 4.2. *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 another graph, where*

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

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

Proof. Let X be a ZFS of G. If $w \notin X$, then we claim that X is also a ZFS of H. During the zero forcing process in G, at least one of u, v, is colored black when w is the only white neighbor of that node. So, either u forces w, which in turn forces v, or v forces w and w in turn may force u. Clearly, for the same zero forcing process in H, instead of w, the node v will be able to force u, or the node u will be able to force v. The remaining zero forcing process evolves as in G. Therefore, in this case, X is also a ZFS of H. On the other hand, if $w \in X$, then at most one of u, v can be in the ZFS as anyone of them along with node w can color the third node black during the zero forcing process. If both these nodes are not in X, then during the zero forcing process, w can not color any node black on its own until one of them (i.e., u or v) is forced black by some other node. One of the u, v must be colored black by one of their other neighbors, and then w can color the remaining white neighbor black. Assume without loss of generality that u is colored first, and w then colors v, then $(X \setminus \{w\}) \cup \{v\}$ is a ZFS in H with coloring of w skipped. However, if $u \in X$, then v is the only white neighbor of w that can be colored black in the first step of the zero forcing process. In this case, $(X \setminus \{w\}) \cup \{v\}$ is a ZFS of H and zero forcing process of G can be replicated in H from the second step onwards as u, v are both colored black before the start of the process. Therefore, $\zeta(G) \geq \zeta(H)$. ∎

Remark 4.3. *Given the conditions of Lemma 4.2, in general, it is not true that $\zeta(G) = \zeta(H)$. In other words, 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 as outlined below.*

In the following, we show that in a particular 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 not change.

Lemma 4.4 (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 G by replacing the two length path $< u, w, v >$ with an edge (u, v). Then, $\zeta(G) = \zeta(H)$.

Proof. It follows from Lemma 4.2 that $\zeta(G) \geq \zeta(H)$, so we only need to show that $\zeta(G) \leq \zeta(H)$. We show a slightly stronger statement that the ZFS of H is also a ZFS of G. Let Y be a ZFS of H. If $v \in Y$, then Y is a ZFS of G as well because the coloring process in H can be started at v, which colors its only neighbor u, and the process continues as required. The coloring process in G can be started by v coloring its only neighbor w, which in turn colors its only white neighbor u, and the process continues as it does in H. If $v \notin Y$, then at some point during the zero forcing process of H, u is colored black while v is its only white neighbor. This means that if we follow the zero forcing process of H exactly on G, u would be colored black while w is its only white neighbor. At that point, w would also be colored black. Since u is already black, v is the only white neighbor of w; thus, it can also be colored black. The rest of the coloring process proceeds as it does in H. We conclude that any ZFS of H is a ZFS of G. This completes the proof. ∎

Definition 4.1. *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.

Lemma 4.5. *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.*

Proof. We prove the claim by contradiction. Assume that b_i, b_j are a pair of leaf nodes of S_k that are not in a ZFS X of G. Since a is the only neighbor of these white-colored nodes, a must be colored black during the zero forcing process while both of them are still white. However, there are two white neighbors of a; therefore, a can not color either of them black, and they will remain white at the end of the coloring process. This contradicts the assumption that X is a ZFS. We conclude that at least $k-1$ leaf nodes of S_k are in any ZFS of G. ∎

Based on Lemma 4.5, 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 4.6 (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, $\zeta(G) = \zeta(H) + k - 1$*

Proof. (i) $\zeta(H) \leq \zeta(G) - (k-1)$: From Lemma 4.5, we may assume without the loss of generality that $b_1, b_2, \ldots, b_{k-1}$ are in a ZFS, X, of G. If b_k is not in X, then all nodes in $N(a) \setminus S_k$ must be colored black before b_k. Further, all of these nodes are colored black due to their neighbors that are not in S_k (recall that while a may itself be colored black at this time but it may not force any white nodes in $N(a) \setminus S_k$ because it has another white neighbor in b_k). Therefore, $X \setminus \{b_1, b_2, \ldots, b_{k-1}\}$ is a ZFS of $G \setminus S_k$, which is H. On the other hand, if b_k is in X, then a may color a node black when there is only one white node left in $N(a) \setminus S_k$. Let b_{k+1} be the last node that is colored black in $N(a)$, then $(X \setminus \{b_1, b_2, \ldots b_k\}) \cup \{b_{k+1}\}$ is a ZFS of H of the required size.
(ii) $\zeta(H) \geq \zeta(G) - (k-1)$: Let Y be the ZFS of H. We claim that $Y \cup \{b_1, b_2, \ldots, b_{k-1}\}$ is a ZFS of G. Let us color node a black in the first step of the coloring process of G. Now, using nodes in Y, we can color all other nodes black except b_k. Once all other neighbors of a are colored black, b_k can be colored black in the last step. Thus, $Y \cup \{b_1, b_2, \ldots, b_{k-1}\}$ is a ZFS of G. This concludes the proof. ∎

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

Algorithm: We begin by constructing a Breadth-First Search (BFS) Tree and the corresponding Queue of nodes in the input tree graph starting from any arbitrary root node. Then, we iterate back from the end of the BFS Queue (recall that the last node in this Queue represents a node at the farthest distance from the root). Next, we check if the parent node in the BFS Tree of the last node v has any children other than the node v. If the parent node of v, denoted by $\pi(v)$, has no other descendants, and $\pi(v)$ is not a leaf, we have a $< u, w, v >$ path that meets the requirements of Lemma 4.4. Therefore, we remove the last node from Queue and repeat this step. If $\pi(v)$ is also a leaf node then $\pi(v)$ can force v and we can remove the last node v from the Queue. Otherwise, we add all descendants of $\pi(v)$ except v to our ZFS, and remove the parent $\pi(v)$ and all its descendants from the BFS Tree and the Queue. We repeat this until the Queue is empty or contains a single node. If the Queue has a single node, we add this node to our ZFS; otherwise, we do not add anything. At this point, we return a ZFS. These steps are summarized in Algorithm 2.

Algorithm 2: ZFS for Tree Graph

```
given: G
initialization: Z = ∅.
T = BFS Tree(G), Q = BFS Queue(G).
while |Q| > 1
    v = last node in Q
    π(v) = parent of v, C_π(v) =Children of π(v)
     Z = Z ∪ (C_π(v) \ {v})
     Remove C_π(v) from Q, T
     if C_π(v) ⊃ {v}
          Remove π(v) from Q, T
     end if
end while
return Z
```

Theorem 4.7. *Algorithm 2 computes a minimum ZFS in Tree graphs in (optimal) linear time.*

Proof. The correctness of the algorithm follows directly from Lemma 4.6. We construct a BFS Tree and Queue in time $O(|V|)$. We iterate over the nodes in the Queue and in each

iteration spend constant time. Since the Queue contains $|V|$ nodes, the total run time of the algorithm is $O(|V|)$. ∎

B. Optimal ZFS in Clique Chains

Next, we consider the minimum ZFS problem in a class of graphs called clique chains, which are widely studied and are significant due to their robustness properties. For instance, for a given number of nodes n and diameter D, graphs with the maximum robustness (measured by the algebraic connectivity or the Kirchhoff index of the graph) are necessarily clique chains [30], [31]. Similarly, these graphs also have other extremal properties [30]–[32]. We define clique chains below.

Definition 4.2. *For a given a set of positive integers $\{n_1, n_2, \cdots, n_{D+1}\}$, a clique chain, denoted by $\mathcal{C}(n_1, \cdots, n_{D+1})$ is a graph obtained from a path graph with $D+1$ nodes as following: Replace the i^{th} node in the path graph with a clique of n_i nodes,[1] i.e. K_{n_i}. Then, make nodes in distinct cliques adjacent if and only if the corresponding nodes in the path graph are adjacent.*

Note that the diameter of $\mathcal{C}(n_1, \cdots, n_{D+1})$ is D. Figure 4 shows an example of a clique chain $\mathcal{C}(2, 3, 3, 2)$. We present the zero forcing number and optimal ZFS in clique chains.

Proposition 4.8. *Let $\mathcal{C}(n_1, \cdots, n_{D+1})$ be a clique chain with $n = \sum_{i=1}^{D+1} n_i$ nodes and D diameter. Let X be a set consisting of one node (arbitrarily chosen) from each clique K_{n_i}, where $i \in \{2, \cdots, D+1\}$. Then,*

$$\zeta(\mathcal{C}(n_1, \cdots, n_{D+1})) = n - D,$$

Moreover, an optimal ZFS consists of all nodes in $\mathcal{C}(n_1, \cdots, n_{D+1})$ excluding the nodes in X.

Proof. (Necessity) Assume $\zeta < n - D$, then one of the following must be true: (1) there is some K_{n_i} with at least two white nodes, or (2) each K_{n_i} has at least one white node. In (1), since all nodes in a clique K_{n_i} have the same neighborhood, the two white nodes in the same clique can not be forced by any black node. In case (2), consider a white node from each K_{n_i}. Note that such white nodes induce a path, say P, of length D. Also, observe that each of the remaining nodes in the clique chain is adjacent to at least two nodes in P. Thus, no black node can force any of the white nodes in P. Hence, the number of nodes in a ZFS must be at least $n - D$.

(Sufficiency) In the given solution set, all nodes in K_{n_1} are black (included in ZFS) and there is exactly one white node in K_{n_i}, $\forall i \in \{2, \cdots, D+1\}$. Thus, the only white node in K_{n_2} becomes black (gets infected) by some node in K_{n_1}, which means all nodes in K_{n_2} become black. Subsequently, the only white node in each of the K_{n_i} becomes black due to some node in $K_{n_{i-1}}$ (whose all nodes have already become black). Thus, all nodes in the clique chain become black, and the given set containing $n - D$ nodes is indeed a ZFS. ∎

Note that Proposition 4.8 also provides a way to construct an optimal ZFS in clique chains, as Figure 4 illustrates.

[1]A clique is a subset of nodes in a graph such that every two nodes in a clique are pairwise-adjacent.

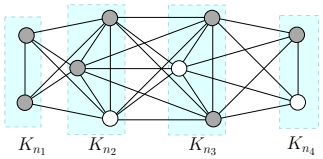


Fig. 4: A clique chain $\mathcal{C}(n_1, n_2, n_3, n_4)$, where $n_1 = n_4 = 2$, and $n_2 = n_3 = 3$. Here, $n = 10$ and diameter $D = 3$. Nodes in an optimal ZFS are colored gray.

V. ZFS Heuristics using Potential Games

We present a game-theoretic approach for finding an optimal ZFS in a distributed manner on 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 5.1:

$$\phi(a) = \frac{1}{n}\left(|der(a)| - \sum_{i=1}^{n} a_i\right), \tag{6}$$

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 5.1. *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* (6)*, 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 (6). 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 $v_i \notin der(a) \cup \{v_j\}$, 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

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

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 $der(a)$ 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. Accordingly, using (6) and $|der(a')| = n$, we obtain

$$\phi(a') - \phi(a) = \frac{1}{n}\left(n - |der(a)| - \sum_{i=1}^{n} a'_i + \sum_{i=1}^{n} a_i\right). \tag{8}$$

Note that (7) and (8) together imply $\phi(a') - \phi(a) = 1/n > 0$, 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 (6) 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 5.1, an optimal ZFS can be obtained by searching for a maximizer of $\phi(a)$ in (6). Such a maximization can be achieved in a distributed manner by using a game-theoretic formulation (e.g., [33]). 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 [34] 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) $I = \{1, 2, \ldots, n\}$, (2) an action space $A = A_1 \times A_2 \times \ldots \times A_n$, where each A_i is the action set of player i, and (3) a set of utility functions $U = U_1, U_2, \ldots, U_n$, 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 solving cooperative multi-agent 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 $a_{-i} \in A_{-i}$,

$$U_i(a'_i, a_{-i}) - U_i(a_i, a_{-i}) = \phi(a'_i, a_{-i}) - \phi(a_i, a_{-i}). \quad (9)$$

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) or Metropolis learning (e.g., [34]–[36]) can be used to have the agents spend most of their time at the global maximizers of $\phi(a)$. More specifically, these algorithms induce an irreducible and aperiodic Markov chain over the action space A such that the limiting distribution, μ_ϵ, satisfies

$$\lim_{\epsilon \to 0^+} \mu_\epsilon(a) > 0 \iff \phi(a) \geq \phi(a'), \forall a' \in A, \quad (10)$$

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

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 (6). While there are also other methods to design such utility functions (e.g., wonderful life utility [37]), one choice is to set all the utilities equal to the global objective, i.e.,

$$U_i(a) = \frac{1}{n}\left(|der(a)| - \sum_{i=1}^n a_i\right), \ \forall i \in I. \quad (11)$$

One can easily verify that the resulting game, Γ_{ZFS}, is a potential game with the potential function $\phi(a)$ in (6), i.e., the utilities in (11) satisfy (9). Accordingly, an optimal ZFS can be found by employing a noisy best-response algorithm such as LLL in a repetitive play of the resulting game, Γ_{ZFS}.

Theorem 5.2. *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*

$$\lim_{\epsilon \to 0^+} \mu_\epsilon(a) > 0 \iff a \text{ corresponds to an optimal ZFS}, \quad (12)$$

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

Proof. Since Γ_{ZFS} is a potential game with the potential function $\phi(a)$ in (6), LLL is known to induce a Markov chain over the action space $A = \{0,1\}^n$ whose limiting distribution, μ_ϵ, satisfies (10) [34]. Due to Lemma 5.1, the maximizers of $\phi(a)$ in (6) are the optimal ZF sets. Consequently, we conclude that $\lim_{\epsilon \to 0^+} \mu_\epsilon(a) > 0$ if and only if a corresponds to an optimal ZFS. ∎

In light of Theorem 5.2, 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 the redundant black nodes in a. In our next result, we show that this heuristic returns an optimal ZFS with an arbitrarily high probability for any connected graph G when the algorithm parameters, $\bar{k}$ and ϵ, are selected properly. Such a performance guarantee is the main advantage of this heuristic

when compared with the greedy heuristic in Algorithm 1, which may produce arbitrarily poor results for some graphs as we have shown in Proposition 3.1.

Algorithm 3: Log-Linear Learning (LLL) 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 (11):

$$\Pr[a_i = a_i'] \sim \exp\left(\frac{U_i(a_i', a_{-i})}{\epsilon}\right), \forall a_i' \in \{0, 1\}.$$

6 : **end for**
7 : $Z = \{v_i \in V \mid a_i = 1\}$,
`-- adding leaders if` Z `is not a ZFS --`
8 : $Z = Z \cup (V \setminus der(Z))$,
`--------- 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** Z

Corollary 5.3. *For any connected $G = (V, E)$, Algorithm 3 always returns a ZFS. Furthermore, let $\Pr[Z \text{ is optimal}; \bar{k}, \epsilon]$ be the probability that the set of nodes $Z \subseteq V$ returned by Algorithm 3 is an optimal ZFS for a specific choice of the algorithm parameters $\bar{k}$ (number of iterations) and ϵ (noise). Then, $\Pr[Z \text{ is optimal}; \bar{k}, \epsilon]$ approaches 1 as ϵ becomes smaller and $\bar{k}$ gets larger, i.e.,*

$$\lim_{\epsilon \to 0^+, \bar{k} \to \infty} \Pr[Z \text{ is optimal}; \bar{k}, \epsilon] = 1. \tag{13}$$

Proof. We will first prove that Algorithm 3 always returns a ZFS. To this end, consider any $Z \subseteq V$ that may be obtained at line 7 of Algorithm 3. In line 8, adding all the nodes that remain white under the zero-forcing process when staring with Z as the initial set of black nodes, i.e., $V \setminus der(Z)$, to Z clearly results in a ZFS. In the final recursive part of the algorithm (lines 9-13), a node v_i is removed from Z only if the the $Z \setminus v_i$ is also a ZFS. Hence, the final Z returned by Algorithm 3 is guaranteed to be a ZFS. Next, we prove that this output, Z, also satisfies (13).

Algorithm 3 starts with an arbitrary $a \in A = \{0, 1\}^n$, which is then updated by following log-linear learning (LLL) for $\bar{k}$ iterations (lines 3-6). In light of Theorem 5.2, this part induces induces a Markov chain over the action space $A = \{0, 1\}^n$ whose limiting distribution, μ_ϵ, satisfies (12). Note that, as $\bar{k}$ goes to infinity, the probability that these iterations result in a specific $a \in A$ is equal to $\mu_\epsilon(a)$. Accordingly, for the resulting Z in line 7 of Algorithm 3, we obtain

$$\lim_{\epsilon \to 0^+, \bar{k} \to \infty} \Pr[Z \text{ is optimal}; \bar{k}, \epsilon] = 1. \tag{14}$$

When Z computed in line 7 is an optimal ZFS, the remainder of Algorithm 3 does not make any further modifications to Z. Hence, we conclude that the set of nodes Z returned by Algorithm 3 satisfies (13). ∎

Remark 5.4. *Variants of Algorithm 3 can be obtained by replacing LLL (lines 3-6) with any other noisy best-response type algorithm (e.g., Metropolis learning) that induces the same limiting behavior in* (10)*. While the resulting algorithms may have some differences in their transient behavior (e.g., [36]), they would all yield the same performance guarantee in Corollary 5.3.*

Our potential game-based solution (Algorithm 3) is similar to [38] in that both methods are based on inducing a Markov chain whose limiting distribution accumulates over the set of states corresponding to optimal ZFS as the noise/temperature parameter diminishes. However, the two methods use different cost/objective and transition probability functions. Furthermore, our game theoretic approach is a distributed method that relies on agents/nodes randomly updating their own variables (their initial color in the zero forcing process).

C. 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 3.1 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 3.1), 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 5 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 5, after about 250 iterations the potential function equals 0.78 most of the time, which is the maximum possible value of (6) for this example.

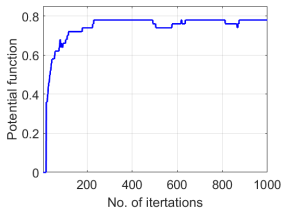
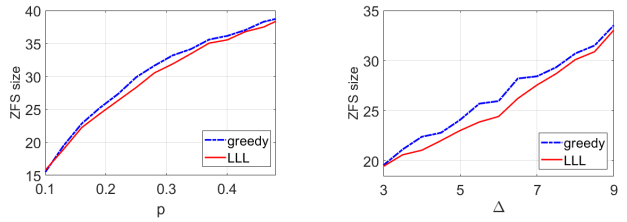


Fig. 5: An example of potential function as a function of number of iterations in LLL.

Next, we consider Erdös-Rényi (ER) random graphs with $n = 50$ nodes. Figure 6(a) 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 6(b), 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. However, as illustrated in the plots, LLL based solution can outperform greedy when ϵ is chosen properly and the algorithm runs for a sufficient number of iterations.



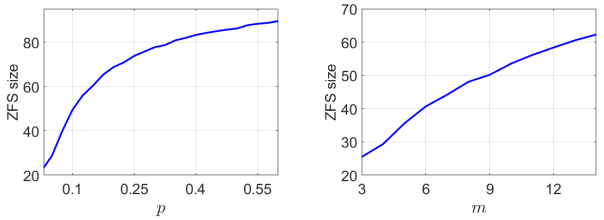
(a) ER graphs (b) Δ-disk graphs

Fig. 6: Comparison of the greedy and LLL based heuristics for ZFS in (a) ER graphs and (b) Δ-disk graphs.

VI. Edge Augmentation to Reduce ZFS

In this section, we study a design problem: *how to add edges in a graph to reduce the size of the zero forcing number of the graph.* To the best of our knowledge, edge augmentation to *reduce* the zero forcing number of a graph has not been studied. Several factors motivate this issue.

First, it is plausible that adding edges in a graph increases the zero forcing number of the resulting graph (as in a ZF process, a colored node can force another node if and only if there is a unique white node in its neighborhood). Additionally, it is typically observed that the zero forcing number of dense graphs tend to be higher. For instance, Figure 7 plots the ZF number as a function of p in Erdös-Rényi (ER) random graphs $G_{n,p}$, and as a function of m in Barabási-Albert (BA) graphs, where m is the number of edges added to the graph with each new node addition. The considered graphs consist of 100 nodes; each point on the plots averages 25 randomly generated instances. Thus, it is compelling to identify missing edges in a graph whose addition does not increase but reduces the zero forcing number of the graph (given that such edges exist).



(a) ER (b) BA

Fig. 7: $\zeta(G)$ in ER and BA random graphs of 100 nodes.

Second, adding edges improves the network's robustness to faults and failures [31]. However, edge augmentation might deteriorate the network controllability since adding edges might increase the zero forcing number of the resulting graph, which describes the minimum number of input nodes for network controllability. Consequently, network controllability and robustness properties could be conflicting at times [32], [39]. Thus, identifying missing edges whose addition not only avoids increasing the zero forcing number but reduces it is significant in co-optimizing the network robustness and controllability [40]–[42]. We begin edge augmentation to reduce the zero forcing number by some useful notions below.

Definition 6.1. *[43] Let Z be a ZFS of graph $G = (V, E)$, we define the following notions (and illustrate them in Figure 8):*

- *A* chronological list of forces *is a list of forces recorded in the order in which they are performed to construct the derived set.*
- *A* forcing chain *(for a given chronological list of forces) is a sequence of nodes $[v_1, v_2, \cdots, v_k]$ such that $v_i \mapsto v_{i+1}$, for $i = 1, 2, \cdots, k-1$.*
- *A* maximal forcing chain *is a forcing chain that is not a proper subsequence of another zero forcing chain.*
- *A* terminal set of Z, *denoted by $R(Z)$, is the set of* last *vertices of the maximal zero forcing chains of a chronological list of forces. A* terminal node*, is a node in a terminal set $R(Z)$.*

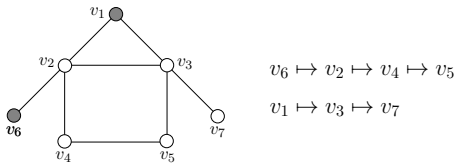


Fig. 8: A ZFS is $Z = \{v_1, v_6\}$. Two maximal forcing chains are $[v_6, v_2, v_4, v_5]$ and $[v_1, v_3, v_7]$. The terminal set of Z is $R(Z) = \{v_5, v_7\}$, which is also a ZFS.

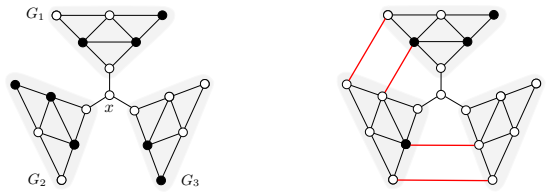
An important observation regarding $R(Z)$ is as follows.

Theorem 6.1. *[43] If Z is a ZFS of G, then so is any terminal set $R(Z)$. Moreover, $|Z| = |R(Z)|$.*

Next, we illustrate below that it is possible to add edges to a graph G to obtain a graph G' such that $\zeta(G') < \zeta(G)$.

Examples: Consider G in Figure 9(a), where black nodes constitute a ZFS of G, and $\zeta(G) = 8$. By adding four edges to G, we obtain G', as shown in Figure 9(b) with $\zeta(G') = 4$. Thus, adding four edges reduced the size of ZFS by four.

We can also generalize this example as follows: Let $G_1, \cdots, G_k$ be a set of graphs, and each G_i has a minimum ZFS Z_i, and ZF number $\zeta(G_i) = \zeta_i$. Without loss of generality, assume $\zeta_1 \geq \zeta_2 \geq \cdots \geq \zeta_k$. Let G be a graph obtained by adjoining a new vertex x to each G_i through some vertex in $R(Z_i)$, for all i. Here, $R(Z_i)$ is a reversal of Z_i as defined above. An example is illustrated in Figure 9(a). The ZF number of G is $\zeta(G) = \left(\sum_{i=1}^{k} \zeta_i\right) - 1$. Now, there exists at least $\left(\sum_{i=2}^{k} \zeta_i\right) - 2$ edges (missing in G) such that each of those edges, if added to G, will reduce the size of its ZFS by one. In other words, by adding $\left(\sum_{i=2}^{k} \zeta_i\right) - 2$ edges to G, we can obtain G' such that $\zeta(G')$ is at most $\zeta_1 + 1$. In Figure 9(a), $\zeta(G) = 8$ (as for each G_i, $\zeta_i = 3$). By adding four missing edges, we get G' in Figure 9(b) with $\zeta(G') = 4$.



(a) G (b) G'

Fig. 9: (a) Colored nodes indicate a ZFS of G with $\zeta(G) = 8$. (b) Adding four (red) edges gives G' with $\zeta(G') = 4$.

Next, we provide a condition to identify missing edges in a graph whose addition reduces the zero forcing number of the resulting graph. Figure 10 illustrates the result.

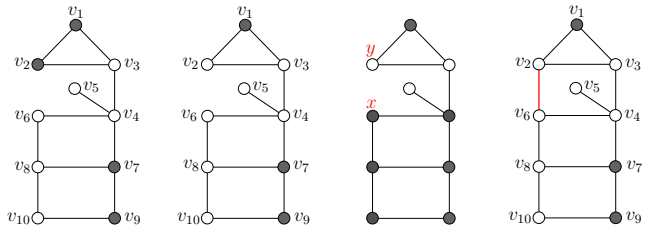
Theorem 6.2. *Let G be a graph with a minimal ZFS Z. If there exists a $Z' \subset Z$, with a derived set H containing a terminal vertex with all neighbors colored black, then there is a nonempty set of edges E' such that the graph $G' = (V, E \cup E')$ has a ZFS strictly smaller than Z.*

Proof. Let G, H, Z, Z' be as in the statement of the theorem. Let $x \in H$ be a terminal vertex for a zero forcing process with input nodes Z', and x has no white neighbors. As Z' is a strict subset of Z, there must be at least one input node $y \in Z \setminus Z'$. Consider the graph $G' = (V, E \cup E')$, where $E' = \{(x, y)\}$. It is enough to show that $Z \setminus \{y\}$ is a ZFS of G'.

We start the zero forcing process with the nodes in Z'. At some point in this process, all nodes in H including x will be colored black. The edge (x, y) does not affect this zero forcing process on G' because node y is not in H and the node x is a terminal node of zero forcing process, i.e., it is not used to force any other node in the graph G'. Due to the minimality of ZFS Z, we know that $y \notin H$ and should be colored white at this point. Since x was a terminal node with black neighbors in the corresponding zero forcing process in G, y is the only white neighbor of black colored node x in G' due to the addition of edge (x, y). According to the rules of zero forcing process, x can color the only white neighbor y now. At this point, the black colored nodes, Z'', in G' include $H \cup Z \setminus Z'$ which include all nodes in Z. The set Z'' is a ZFS of G because $Z'' \supset Z$. Note that for two sets $A \subseteq B$, the derived set of B can not be smaller than $der(A)$ because one can always start the zero forcing with the smaller input set A and get the same set of black nodes that are in the derived set $der(A)$. Furthermore, Z'' is also a ZFS of G' because the added edge (x, y) is now between two already black nodes, and the addition of edges among input nodes does not affect the zero forcing process. Thus, $Z \setminus \{y\}$ is a ZFS of G'. ∎

Theorem 6.2 implies that we can reduce the size of a ZFS whenever we can find a terminal node with all black neighbors in the derived set of some subset of the ZFS (as shown in Figure 10). In particular, we can repeatedly apply this result to add multiple edges and reduce the size of ZFS.

Next, we consider the family of k-connected graphs and get the following result. Figure 11 illustrates the result.



(a) G (b) (c) (d) G'

Fig. 10: (a) $Z(G) = \{v_1, v_2, v_7, v_9\}$. (b) Consider $Z' = \{v_1, v_7, v_9\}$. (c) The colored nodes constitue the derived set H of Z'. Here, $x = v_6$ is a terminal node with all black neighbors. (d) G' is obtained by adding an edge between x and $y = v_2$, and has a smaller ZFS than G.

Proposition 6.3. *Let G be a k-connected graph with a minimum vertex cut C and a minimal ZFS Z. Also, let $G' = G \setminus C$ be the disconnected graph after removing vertices in C from G. If there exists a connected component X in G' with more than k vertices in $(X \cap Z) \subset Z$, and $X \cap Z$ is also a ZFS of the induced graph on $X \cup C$, then there exist edges whose addition to G will strictly decrease the size of ZFS of the resulting graph.*

Proof. Let G, G', C, X, Z be as in the proposition statement. As $(X \cap Z)$ is a strict subset of Z, there is at least one y that is in Z but not in $(X \cap Z)$. As $(X \cap Z)$ is a ZFS of the graph on $X \cup C$, y can not be one of the cut vertices as they can already be colored black by the nodes in $(X \cap Z)$. Therefore, there is at least one $y \in Z \setminus (X \cup C)$ due to the minimality of ZFS Z. Also, there are exactly $|X \cap Z|$ terminal nodes of the zero forcing process on $X \cup C$. As $|(X \cap Z)| > |C|$, at least one of those terminal nodes, say x, is in the connected component X. Note that all neighbors of x are in $X \cup C$ as C is a vertex cut of G. We now show that $Z \setminus \{y\}$ is ZFS of $G' = G \cup \{(x, y)\}$, the graph G with added edge (x, y). We start the zero forcing process with nodes in $(X \cap Z)$. At some point in this process, x will be a terminal node with no white neighbors - recall that all neighbors of x are in $X \cup C$ that lies in the derived set of $(X \cap Z)$. It follows that y is the only white neighbor of x and can be colored black. Therefore all nodes in Z are colored black. As Z is a ZFS of G, $Z \cup \{x\}$ is a ZFS of G. Further, $Z \cup \{x\}$ is also a ZFS of G' because the edge (x, y) between two input nodes does not affect the coloring process. Thus, adding edge (x, y) to G reduces the size of the ZFS by at least one. This completes the proof. ∎

We note that repeated application of Proposition 6.3 may potentially increase the density of a graph while reducing the size of the input nodes. As a result we get the following:

Corollary 6.4. *Let G be a graph with a vertex cut C of size k (in particular but not necessarily when G is k-connected) and a ZFS Z. Further, let $G' = G \setminus C$ be the disconnected graph after removing vertices in C from G. If there exists a connected component X in G' with $k + t$, $t > 0$ vertices of $(X \cap Z)$, and $X \cap Z$ is also a ZFS of the induced graph on $X \cup C$; then there exist $\binom{m}{2}$ edges whose addition to G will*

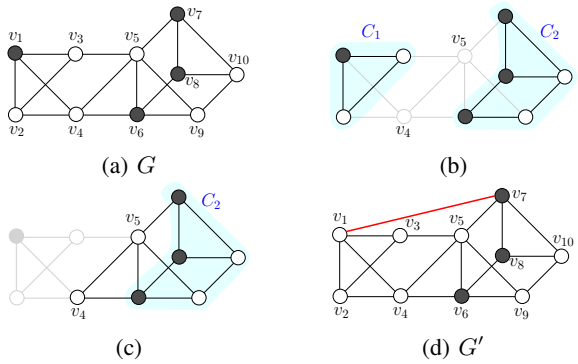


Fig. 11: (a) G is 2-connected. A minimum ZFS is $\{v_1, v_6, v_7, v_8\}$. (b) A cut set of G is $\{v_4, v_5\}$. The two resulting components are C_1 and C_2. (c) $\{v_6, v_7, v_8\}$ is a ZFS of the graph induced by vertices in C_2 and the cut set. (d) Adding edge (red) between v_1 and v_7 gives G', which has a minimum ZFS of three nodes $\{v_6, v_7, v_8\}$.

reduce the size of ZFS of the resulting graph by at least m where $m = \min(t, |Z \setminus X|)$.

Proof. As in the proof of Proposition 6.3, if we start the zero forcing process with nodes in $(X \cap Z)$, at some point in this process, there are at least t terminal nodes $x_1, x_2, \ldots, x_t$ with no white neighbors - recall that all neighbors of x_i are in $X \cup C$ that lies in the derived set of $(X \cap Z)$. Similarly, there are $|Z \setminus X|$ nodes $y_1, y_2, \ldots, y_{|Z \setminus X|}$ outside X. Therefore, there are at least m pairs of nodes x_i, y_j such that adding edges $\{(x_i, y_j) : 1 \leq i \leq m, 1 \leq j \leq i\}$ to G will reduce the size of ZFS of the resulting graph by at least m as $Z \setminus \{y_j : 1 \leq j \leq m\}$ is a ZFS of the resulting graph. ∎

Theorem 6.2, Proposition 6.3, and Corollary 6.4 can be used to design an algorithm to iteratively add edges to a graph to reduce its ZFS. Intuitively, these results can be used whenever heterogeneity appears in a network, i.e., there is a non-uniform distribution of edges, which is often the case in practical networks. Typically, a denser part of the network requires more input nodes for control. However, the potential of these input nodes is somewhat underutilized, and strategically adding extra edges within the network provides a way to utilize these input nodes to control the rest of the network. To illustrate this point, consider a graph consisting of two cliques on 100 nodes each and connected by a single edge. The size of the minimum ZFS of the graph is 197. However, we can add ε edges to the graph, where $98 \leq \varepsilon \leq \binom{98}{2}$, such that the resulting graph has a ZFS of size 99, which is a significant improvement compared to the ZFS size of the original graph. While this is an extreme example, we note that our results can exploit even a small heterogeneity (non-uniformity) in the network density.

VII. Conclusion

We studied various aspects of the minimum ZFS problem in undirected graphs. We provided a linear time algorithm to solve the problem in trees optimally and also characterized the minimum ZFS in clique chains. Further, we formulated the problem as a potential game and utilized log-linear learning (LLL) to solve the game. Adding edges could improve the graph's robustness; however, it could increase the size of the minimum ZFS. We characterized missing edges in a graph whose addition would reduce the size of ZFS. In the future, we will extend these methods to combine graphs by edge augmentations while exploring the trade-off between the number of augmented edges and the size of the ZFS of the combined graph. Another interesting direction to consider is the extension of results to adversarial settings, i.e., how the zero forcing process and the zero forcing number change due to failures/attacks preventing nodes from forcing other nodes.

References

[1] W. Abbas, M. Shabbir, Y. Yazıcıoğlu, and X. Koutsoukos, "Leader selection for strong structural controllability in networks using zero forcing sets," in *American Control Conference (ACC)*, 2022.

[2] D. Burgarth, D. D'Alessandro, L. Hogben, S. Severini, and M. Young, "Zero forcing, linear and quantum controllability for systems evolving on networks," *IEEE Transactions on Automatic Control*, vol. 58, no. 9, pp. 2349–2354, 2013.

[3] D. Burgarth, V. Giovannetti, L. Hogben, S. Severini, and M. Young, "Logic circuits from zero forcing," *Natural Computing*, vol. 14, no. 3, pp. 485–490, 2015.

[4] F. H. Kenter and J. C.-H. Lin, "On the error of a priori sampling: zero forcing sets and propagation time," *Linear Algebra and its Applications*, vol. 576, pp. 124–141, 2019.

[5] 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.

[6] 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.

[7] L. Xiang, F. Chen, W. Ren, and G. Chen, "Advances in network controllability," *IEEE Circuits and Systems Magazine*, vol. 19, no. 2, pp. 8–32, 2019.

[8] G. Baggio, F. Pasqualetti, and S. Zampieri, "Energy-aware controllability of complex networks," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, pp. 465–489, 2022.

[9] Z. Kong, L. Cao, L. Wang, and G. Guo, "Controllability of heterogeneous networked systems with nonidentical inner-coupling matrices," *IEEE Transactions on Control of Network Systems*, vol. 9, no. 2, pp. 867–878, 2021.

[10] Y.-Y. Liu, J.-J. Slotine, and A.-L. Barabási, "Controllability of complex networks," *Nature*, vol. 473, no. 7346, pp. 167–173, 2011.

[11] Z. Yuan, C. Zhao, Z. Di, W.-X. Wang, and Y.-C. Lai, "Exact controllability of complex networks," *Nature Communications*, vol. 4, no. 1, pp. 1–9, 2013.

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

[13] 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.

[14] A. Clark and R. Poovendran, "A submodular optimization framework for leader selection in linear multi-agent systems," in *50th IEEE Conference on Decision and Control and European Control Conference*, 2011, pp. 3614–3621.

[15] 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.

[16] 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.

[17] Y. Yazıcıoğlu, M. Shabbir, W. Abbas, and X. Koutsoukos, "Strong structural controllability of networks: Comparison of bounds using distances and zero forcing," vol. 146. Elsevier, 2022, p. 110562.

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

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

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

[21] T. Kalinowski, N. Kamcev, and B. Sudakov, "The zero forcing number of graphs," *SIAM Journal on Discrete Mathematics*, vol. 33, no. 1, pp. 95–115, 2019.

[22] 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

[23] 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.

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

[25] 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.

[26] D. D. Row, "Zero forcing number: Results for computation and comparison with other graph parameters," Ph.D. dissertation, Iowa State University, 2011.

[27] 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.

[28] S. M. Fallat and L. Hogben, "The minimum rank of symmetric matrices described by a graph: a survey," *Linear Algebra and its Applications*, vol. 426, no. 2-3, pp. 558–582, 2007.

[29] C. Johnson and C. Saiago, "Estimation of the maximum multiplicity of an eigenvalue in terms of the vertex degrees of the graph of a matrix," *The Electronic Journal of Linear Algebra*, vol. 9, pp. 27–31, 2002.

[30] H. Wang, R. Kooij, and P. Van Mieghem, "Graphs with given diameter maximizing the algebraic connectivity," *Linear Algebra and its Applications*, vol. 433, no. 11-12, pp. 1889–1908, 2010.

[31] W. Ellens, F. Spieksma, P. Van Mieghem, A. Jamakovic, and R. Kooij, "Effective graph resistance," *Linear Algebra and its Applications*, vol. 435, no. 10, pp. 2491–2506, 2011.

[32] W. Abbas, M. Shabbir, A. Y. Yazıcıoğlu, and A. Akber, "Tradeoff between controllability and robustness in diffusively coupled networks," *IEEE Transactions on Control of Network Systems*, vol. 7, no. 4, pp. 1891–1902, 2020.

[33] 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.

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

[35] J. R. Marden and J. S. Shamma, "Revisiting log-linear learning: Asynchrony, completeness and payoff-based implementation," *Games and Economic Behavior*, vol. 75, no. 2, pp. 788–808, 2012.

[36] H. Jaleel and J. S. Shamma, "Path to stochastic stability: Comparative analysis of stochastic learning dynamics in games," *IEEE Transactions on Automatic Control*, vol. 66, no. 11, pp. 5253–5268, 2020.

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

[38] K. Yashashwi, S. Moothedath, and P. Chaporkar, "Minimizing inputs for strong structural controllability," in *2019 American Control Conference (ACC)*, 2019, pp. 2048–2053.

[39] F. Pasqualetti, S. Zhao, C. Favaretto, and S. Zampieri, "Fragility limits performance in complex networks," *Scientific Reports*, vol. 10, no. 1, pp. 1–9, 2020.

[40] W. Abbas, M. Shabbir, Y. Yazıcıoğlu, and X. Koutsoukos, "Edge augmentation with controllability constraints in directed laplacian networks," *IEEE Control Systems Letters*, vol. 6, pp. 1106–1111, 2021.

[41] S. S. Mousavi, M. Haeri, and M. Mesbahi, "Robust strong structural controllability of networks with respect to edge additions and deletions," in *American Control Conference (ACC)*, 2017, pp. 5007–5012.

[42] X. Chen, S. Pequito, G. J. Pappas, and V. M. Preciado, "Minimal edge addition for network controllability," *IEEE Transactions on Control of Network Systems*, vol. 6, no. 1, pp. 312–323, 2018.

[43] F. Barioli, W. Barrett, S. M. Fallat, H. T. Hall, L. Hogben, B. Shader, P. Van Den Driessche, and H. Van Der Holst, "Zero forcing parameters and minimum rank problems," *Linear Algebra and its Applications*, vol. 433, no. 2, pp. 401–411, 2010.

Waseem Abbas is an Assistant Professor in the System Engineering Department at the University of Texas at Dallas, TX, USA. Previously, he was a Research Assistant Professor at Vanderbilt University, Nashville, TN, USA. He received Ph.D. (2013) and M.Sc. (2010) degrees, both in Electrical and Computer Engineering, from Georgia Institute of Technology, Atlanta, GA, and was a Fulbright scholar from 2009 to 2013. His research interests are in the areas of control of networked systems, resilience, and robustness in networks, distributed optimization, and graph-theoretic methods in complex networks.

Mudassir Shabbir is an Associate Professor and the Chair of the Department of Computer Science at the Information Technology University, Lahore, Pakistan and a Research Assistant Professor at Vanderbilt University, Nashville TN. He received his Ph.D. from Division of Computer Science, Rutgers University, NJ, USA in 2014. Previously, Mudassir has worked at Lahore University of Management Sciences, Pakistan, Los Alamos National Labs, NM, Bloomberg L.P. New York, NY and at Rutgers University. He was Rutgers Honors Fellow for 2011-12. His main area of research is Algorithmic and Discrete Geometry and has developed new methods for the characterization and computation of succinct representations of large data sets with applications in non-parametric statistical analysis. He also works in Graph Machine Learning and Resilient Network Systems.

Yasin Yazıcıoğlu is an Assistant Professor in the Department of Mechanical and Industrial Engineering at Northeastern University, Boston, MA, USA. Previously, he was a Research Assistant Professor at the University of Minnesota. He received the Ph.D. degree in Electrical and Computer Engineering from the Georgia Institute of Technology in 2014. He was a Postdoctoral Research Associate at the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology from 2014-2017. His research is primarily focused on distributed decision making, control, and learning with applications to robotics, cyber-physical systems, and networks.

Xenofon Koutsoukos is the Thomas R. Walters Professor and Chair of the Department of Computer Science at the School of Engineering at Vanderbilt University. He is also a Senior Research Scientist with the Institute for Software Integrated Systems (ISIS) and holds a secondary appointment in the Department of Electrical and Computer Engineering. He received his PhD from the University of Notre Dame in 2000. He was a Member of Research Staff with the Xerox Palo Alto Research Center (PARC) (2000–2002). He has coauthored more than 350 journal and conference papers and he is co-inventor of four US patents. His research work is in the area of cyber-physical systems with emphasis on learning-enabled systems, security and resilience, diagnosis and fault tolerance, distributed algorithms, formal methods, and adaptive resource management. Prof. Koutsoukos was the recipient of the NSF Career Award in 2004, the Excellence in Teaching Award in 2009 from the Vanderbilt University School of Engineering, and the 2011 NASA Aeronautics Research Mission Directorate (ARMD) Associate Administrator (AA) Award in Technology and Innovation. He is a Fellow of the IEEE for his contributions to the design of resilient cyber-physical systems.