AUSTRALASIAN JOURNAL OF COMBINATORICS
Volume **88(3)** (2024), Pages 308–326

# Leaky forcing: a new variation of zero forcing

Joseph S. Alameda*

Shannon Dillman  Franklin Kenter†

*Mathematics Department*
*United States Naval Academy*
*Annapolis, MD, U.S.A.*

**Abstract**

Zero forcing is a one-player game played on a graph. The player chooses some set of vertices to color blue, then iteratively applies a color-change rule, allowing vertices to "force" their neighbors to become blue. The results of the game determine linear-algebraic properties of matrices with a corresponding sparsity pattern.

In this article, we introduce and study a new variation of zero forcing where $\ell \geq 0$ of the vertices may have a "leak" which cannot facilitate any forces. The key is that the locations of the leaks are unknown at the start of the game; hence, to win, the player must implement a strategy that overcomes any configuration of $\ell$ leaks. As such, this variation of zero forcing corresponds to resiliency in solving linear systems.

We compute the $\ell$-leaky forcing numbers for selected families of graphs for various values of $\ell$, including grid graphs and hypercubes. We find examples where additional edges make the graph more "resilient" to these leaks. Finally, we adapt known computational methods to our new leaky forcing variation.

## 1 Introduction

Zero forcing is a game played on a graph, $G$. The game was originally developed to bound the maximum nullity (or equivalently, the minimum rank) of $G$ over the set of matrices with a corresponding zero-nonzero pattern, $\mathcal{S}(G)$ [1]. In the zero forcing game, some vertices start as blue and others white. Blue vertices "force" white vertices to become blue under the *zero forcing color-change rule*: If all but one of a blue vertex's neighbors are blue, the remaining white neighbor becomes blue. A set of vertices is a *zero forcing set* if, when initially colored blue, all vertices in $G$ will

* First author suported by ONR Grant N0001417GTC00161.
† Third author supported, in part, by NSF Grant DMS-1719894 and ONR grant N0001418WX00709. Corresponding author: kenter@usna.edu

ISSN: 2202-3518 

eventually become blue after iteratively applying the color-change rule. The general goal is to find the minimum zero forcing set.

Perhaps surprisingly, zero forcing and its variations have a habit of being rediscovered in different fields and applications. These include power systems and sensor allocation [6, 11, 19], control of quantum systems [7], and controllability of leader-follower systems [15].

Despite all of these applications, most variations of zero forcing arise from the original context of the minimum rank problem: If one places additional restrictions or relaxations on the matrix pattern, those restrictions translate to variations in the color-change rule. For instance, restricting the desired matrices to positive semidefinite and/or skew-symmetric matrices changes the color-change rules, making it easier for vertices to force [3, 4]. On the other hand, the applications mentioned above indicate that there are other potential meaningful variations of zero forcing beyond the minimum rank problem, as we will study here.

The variation we study in this article is based on the following equivalent notion of zero forcing.

**Theorem 1.1** (K.-Lin, [14, Proposition 3:i,iii])**.** *Let $G = (V, E)$ be a graph and $F(\neq \mathbb{F}_2)$ be a field. A set of vertices, $S$, is a zero forcing set of $G$ if and only if for any matrix $\mathbf{A} \in \mathcal{S}_F(G)$, whenever $\mathbf{Ax} = \mathbf{0}$, the partial vector $\mathbf{x}_S$ (i.e., $\mathbf{x}$ restricted to entries corresponding to $S$) is sufficient to uniquely determine the entirety of $\mathbf{x}$.*

Let us put this theorem into an applied context. Suppose you know the pattern of interactions of a system and you want to place sensors on the nodes in order to observe the entire system. Using natural laws (such as Kirchhoff's Laws), once the details of the interactions are known, it is possible to deduce the status of some nodes that do not have a sensor; hence, it is not necessary to place sensors at every node. For example, in a fluid network, you may be able to deduce the flow at every point in the network using only a very limited number of sensors. The theorem considers the context where the sensors must be placed and allocated before the system is realized (e.g., before you know *exactly* what proportions of flow go from one juncture to another, etc.). Indeed, the theorem says the *only* way to guarantee full and accurate observability is to place sensors on nodes corresponding to a zero forcing set. This is exactly the concept behind the application of zero forcing in power grids and quantum systems: One cannot wait until the system is realized to determine what aspects to monitor or measure. Rather, one must be ready to go beforehand, only knowing the rough pattern of interactions.

The variation of zero forcing we introduce here is based upon a variation of the scenario above. In a literal sense, we ask the question: What if there is a leak in the network? If a leak exists in a network at a specific node, then the natural laws could not correctly be applied to further deduce the status of other nodes. In the context of zero forcing, each "force" corresponds to applying a linear equation (or physical law) to deduce further information [1, 14]. Hence, a "leak" corresponds to a vertex that is unable to force. We wish to determine a way to observe the whole system despite the possibility of the presence of a fixed number of leaks. In the context of zero forcing, given a graph $G$ and a number of leaks $\ell$, we wish to find the minimum

number of initially-blue vertices such that *regardless of which specific $\ell$ vertices are unable "force,"* all of the vertices will eventually be blue. We call this minimum quantity the $\ell$-leaky forcing number of $G$, denoted $\mathrm{Z}_{(\ell)}(G)$; it will be more carefully defined in Section 2.

Our variation is not unlike the "fault-tolerant" variant of power domination. Power domination is a variant of zero forcing used in the phasor measurement unit (PMU) placement problem for electrical networks introduced by Haynes, Hedetniemi, Hedetniemi and Henning in [11]. In other work, Pai, Chang and Wang study placement of PMUs (sensors) on grids using a variant of power domination. The goal of their study is to make the entire network observable after iteratively applying natural laws even if some number, $k$, of the PMUs are faulty and unable to provide measurements [16]. The main distinction here is that the faults within leaky forcing may be at a vertex where no sensor is placed (e.g., it is not initially colored blue) whereas Pai, Chang and Wang are only concerned with the case where the faults occur at the locations of the sensors themselves.

This study has faced a few delays in light of the recent pandemic. Some of the contributions of the original preprint [9] of this article have since been superseded by subsequent work. Where applicable, we will refer the reader to these developments. Even so, we contribute the following:

- We provide preliminaries including a definition of leaky forcing and the *$\ell$-leaky forcing number* of a graph, $Z_{(\ell)}(G)$. (Section 2)
- We derive the exact value of $Z_{(\ell)}(G)$ for select families of graphs, including paths, cycles, and wheels. (Section 3)
- We determine exact values and bounds for Cartesian (box) products of graphs $Z_{(\ell)}(G \,\square\, H)$ including the specific families such as grids and hypercubes. (Section 4)
- We construct and implement an integer program algorithm to compute $Z_{(\ell)}(G)$ exactly and run computational experiments. (Section 5)

## 2 Preliminaries, Introduction to Leaks

A *leak* in a graph $G$ is a vertex that is unable to perform a force. An *$\ell$-leaky forcing set* for a graph $G$ is a subset of initial blue vertices $B$ such that if any $\ell$ vertices are chosen to be leaks (chosen only after $B$ has been specified), then iteratively applying the color-change rule will still force every vertex in $G$. The *$\ell$-leaky forcing number* of $G$ is the order of a minimum $\ell$-leaky forcing set and is denoted $\mathrm{Z}_{(\ell)}(G)$. It is worth noting that $Z_{(0)}(G)$ is the zero forcing number of $G$ in the original sense. We will look at this $\ell$-leaky forcing number on various families of graphs in Section 3 and in Section 4.

We now provide some crucial basic results for our study of leaky forcing.

**Lemma 2.1.** *For any graph $G$,*

$$Z_{(0)}(G) \leq Z_{(1)}(G) \leq Z_{(2)}(G) \leq \cdots \leq Z_{(n)}(G).$$

*Proof.* Let $G$ be a graph on $n$ vertices, and let $B$ be a minimum $\ell$-leaky forcing set, $1 \leq \ell \leq n$. Since $B$ is an $\ell$-leaky forcing set, if we choose any $\ell - 1$ vertices of $G$ to be leaks, $B$ will still be able to force $G$ to be blue. Therefore, $B$ is also an $(\ell - 1)$-leaky forcing set. Thus, $\mathrm{Z}_{(\ell-1)}(G) \leq \mathrm{Z}_{(\ell)}(G)$. □

Let us review the concept of "forts" introduced in [10]. A (zero forcing) *fort* is a subset of vertices $F$ such that $F$ is non-empty and that no vertex in $V \setminus F$ is adjacent to exactly one vertex in $F$. This means that a fort can also be thought of as a set $F \subseteq V$ such that $V \setminus F$, as blue, is unable to make any forces. Put another way, we define the *closure* of $S$, $cl(S)$, to be the set of blue vertices after exhaustively applying the color-change rule; in which case, a non-empty set of vertices is a fort of $G$ if and only if it is the complement of the closure of a subset of vertices of $G$.

The power of forts is the following which allows for a constraint generation method to calculate zero forcing numbers based on the following.

**Proposition 2.2.** [5, Model 2 and Theorem 8, reworded] *For a graph $G$, a set of vertices $S$ is a zero forcing set if and only if $S$ intersects every fort.*

We will extend the concept of forts to leaky forcing naturally. Define the closure of $S$ with respect to the leaky set $L$, $cl(S, L)$, to be the set of vertices that are forced blue after exhaustively applying the color-change rule to the initially-blue set $S$ with leaks at the vertices $L$. In this case, an *$\ell$-leaky fort* (or just *fort* if the context is clear) is a non-empty set of vertices $T$ for which $T = V \setminus cl(S, L)$ for some set of vertices $S$ and a set of vertices $L$ of order $\ell$.

**Proposition 2.3.** *For a graph $G$, a set of vertices $S$ is an $\ell$-leaky forcing set if and only if $S$ intersects every $\ell$-leaky fort.*

*Proof.* We proceed in both directions by contrapositive.

Suppose $S$ is not an $\ell$-leaky forcing set. Then, for some set of leaks, $L$, the set $T = V \setminus cl(S, L)$ is non-empty. However, $T$ is a fort by definition and $S$ did not intersect $T$.

Suppose $S$ does not intersect all $\ell$-leaky forts. Let $T$ be such a fort achieved with leak set $L$. Note that $S \subseteq V \setminus T$, so $cl(S, L) \subseteq cl(V \setminus T, L)$. However, by the choice of $T$ and $L$, $cl(V \setminus T, L) = V \setminus T$ is not all of $V$. Hence, $S \subseteq V \setminus T \neq V$ cannot be an $\ell$-leaky forcing set. □

**Lemma 2.4.** *Choose $\ell \geq 0$. Let $G$ be a graph and let $v$ be a vertex of $G$ with degree $\ell$ or less. Then, $\{v\}$ is an $\ell$-leaky fort of $G$. Hence, $v$ must necessarily be in every $\ell$-leaky forcing set of $G$.*

*Proof.* Color all the vertices of $G$ except for $v$. Set the open neighborhood of $v$ as leaks (for which there are necessarily at most $\ell$ neighbors). Since no vertex can force $v$ by the nature of the leaks, it remains white. Hence, $\{v\}$ is a fort. By Proposition 2.3, $v$ must be in every $\ell$-leaky forcing set. □

## 3 Leaky Forcing For Certain Families of Graphs

In this section, in order to get a taste of the topic, we discuss the $\ell$-leaky forcing number for particular graph families. Specifically, we look at paths, trees, cycles, complete graphs, wheels and grids. We will focus on graph products and hypercubes in Section 4.

### 3.1 Paths

**Proposition 3.1.** For the path on $n$ vertices, $P_n$,

$$Z_{(\ell)}(P_n) = \begin{cases} 1 & \text{if } \ell = 0 \\ 2 & \text{if } \ell = 1 \\ n & \text{if } \ell \geq 2 \end{cases}$$

*Proof.* It is a well-known result that $\mathrm{Z}_{(0)}(P_n) = 1$. We will focus on when $\ell \geq 1$. Let $B$ be a minimum 1-leaky forcing set for $P_n$. By Lemma 2.4, both ends of $P_n$ must be in $B$. Therefore, $|B| \geq 2$. If either of the blue vertices is chosen to be a leak, then the other vertex will be able to complete the forcing process. If a vertex in between the two ends of $P_n$ is chosen to be a leak, then both sides will force until the leak is reached. Thus, $|B| \leq 2$.

If $\ell \geq 2$, then again by Lemma 2.4, all vertices of degree $\ell$ or less must be in $B$. Since every vertex in a path is of degree 2 or less, all vertices in $P_n$ must be in $B$. Thus $|B| = n$. □

### 3.2 Trees

**Proposition 3.2.** *For a tree $T \neq K_1$ with $t$ leaves, $Z_{(1)}(T) = t$. Furthermore, the optimum 1-leaky forcing set contains exactly all leaves of $T$.*

*Proof.* By Lemma 2.4, $Z_{(1)}(T) \geq t$ (i.e., all the leaves must be colored blue).

It remains to show that coloring all of the leaves of $T$ necessarily produces a 1-leaky forcing set. We will proceed by strong induction on the number of vertices on $T$. For a base case, note that for $K_2$, all vertices are leaves and the statement holds. For the induction step, consider a tree $T$ with 3 or more vertices; we will assume the result holds for any smaller tree. Let $L$ be the set of all leaves in $T$; color $L$ blue. Necessarily $T$ has at least 2 leaves; at least one of them will not be a leak. Call that vertex $v$ and its sole neighbor $v'$. Now, $v$ forces $v'$. Consider the tree $T - v$ which now has at least $L \cup \{v'\} \setminus \{v\}$ colored blue. Note that $v'$ forces within $T - v$ if and only if it can force in $T$ given that $v$ is blue. Also, the leaves of $T - v$ are $L$ with perhaps $v'$ added; hence by the induction hypothesis, with $L \cup \{v'\} \setminus \{v\}$ blue in $T - v$, we have a 1-leaky forcing set of $T - v$. Since $v$ does not affect whether a vertex can force besides $v'$, this 1-leaky forcing set of $T - v$ will force all of $T - v$, and hence, $T$. □

We emphasize that subsequent work has extended Proposition 3.2 for all $\ell \geq 0$.

**Theorem 3.3** (A.-Kritschgau-Warnberg-Young 2022 [2, Theorem 4.4])**.** *For any tree $T$ and positive integer $\ell$, the optimal $\ell$-leaky forcing set is to choose all vertices of degree $\ell$ or less.*

### 3.3 Cycles

**Proposition 3.4.** *For the cycle on $n$ vertices, $C_n$,*

$$Z_{(\ell)}(C_n) = \begin{cases} 2 & \text{if } \ell = 0, 1 \\ n & \text{if } \ell \geq 2 \end{cases}$$

*Proof.* It is well-known that the zero forcing number of $C_n$ is 2. This is achieved by coloring two adjacent vertices on the cycle blue. Hence, $Z_{(0)}(C_n) = 2$.

For $\ell = 1$, we know that by Lemma 2.1 $Z_{(1)}(C_n) \geq 2$. Hence, it suffices to show that a zero forcing set will force the graph despite a single leak. We have the following two cases:

*Case 1 - The leak is added to a blue vertex.* If the leak is on one of the blue vertices, then the other may force around the cycle until it reaches the leaky vertex.

*Case 2 - The leak is added to a white vertex.* If the leak is on an initially white vertex, then both blue vertices will force and the forcing repeats until one reaches the vertex with the leak. However, the other may force around the cycle until all vertices are colored blue.

Finally for $\ell \geq 2$ leaks, we know by Lemma 2.4 that since all vertices have degree 2, every vertex must be colored blue to be an $\ell$-leaky forcing set. □

### 3.4 Complete Graphs

**Proposition 3.5.** *For the complete graph on $n$ vertices, $K_n$,*

$$Z_{(\ell)}(K_n) = \begin{cases} n-1 & \text{if } \ell \leq n-2, \\ n & \text{if } \ell = n-1, n. \end{cases}$$

*Proof.* It is well-known that if $G = K_n$, then $\mathrm{Z}_{(0)}(G) = n - 1$. Therefore, by Lemma 2.1, $\mathrm{Z}_{(\ell)}(K_n) \geq n - 1$ for all $\ell \geq 0$. If $\ell \leq n - 2$, let $B$ be a set of $n - 1$ vertices and place $\ell$ leaks anywhere on $K_n$. At least one vertex in $B$ does not have a leak and can force the only remaining white vertex blue. Thus, $B$ is an $\ell$-leaky forcing set and $\mathrm{Z}_{(\ell)}(K_n) = n - 1$. If $\ell \geq n - 1$, then by Lemma 2.4, every vertex in $K_n$ must be in an $\ell$-leaky forcing set. Therefore, $\mathrm{Z}_{(\ell)}(K_n) = n$. □

### 3.5 Wheels

Here, we will compute the leaky forcing numbers for the wheel graph. For $n \geq 3$, we will let $W_n$ denote the wheel graph on $n + 1$ vertices constructed by taking the cycle on $n$ vertices, $C_n$, and adding an additional "central" vertex that is adjacent to all other vertices. We will denote the vertices on the cycle as $v_1, v_2, \ldots, v_n$ with $v_i$ adjacent to $v_j$ whenever $i - j \equiv \pm 1 \mod n$ and the central vertex as $u$. Where applicable, subscripts are taken mod $n$ (e.g., $v_{n+1} = v_1$).

**Proposition 3.6.** *For the wheel graph on $n+1 \geq 4$ vertices, $W_n$,*

$$Z_{(\ell)}(W_n) = \begin{cases} 3 & \text{if } \ell = 0, 1, \\ \lceil 2n/3 \rceil + 1 & \text{if } \ell = 2, \\ n & \text{if } 2 < \ell < n, \\ n+1 & \text{if } \ell = n, n+1. \end{cases}$$

*Proof. Case $\ell = 0$.* It is well-known that $Z(W_n) = Z_{(0)}(W_n) = 3$. This is achieved by coloring the set $B = \{u, v_1, v_2\}$ blue.

*Case $\ell = 1$.* By Lemma 2.1, $Z_{(1)}(W_n) \geq Z_{(0)}(W_n) = 3$. The set $B$ above is sufficient to ensure the entire graph can be blue, similar to the proof of Proposition 3.4. Hence, $Z_{(1)}(W_n) = 3$.

*Case $2 < \ell < n$.* By Lemma 2.4, all vertices on the outer cycle, $v_1, \ldots v_n$, must be colored blue. To show that this is sufficient, note that by the Pigeonhole Principle, one of the vertices on the outer cycle must not have a leak. That vertex can can immediately force the center, the only remaining white vertex.

*Case $\ell = n$ or $n+1$.* By Lemma 2.4, *all* vertices must be colored blue.

It still remains to prove the result for $\ell = 2$.

*Case $\ell = 2$ and $n = 3$.* For $n = 3$, we have $Z_{(2)}(W_3) \geq Z(W_3) = 3 = \lceil 2 \cdot 3/3 + 1 \rceil$. Coloring any three vertices blue will result in at least one vertex without a leak which can force the remaining vertex. Hence, $Z_{(2)}(W_3) = 3$.

*Case $\ell = 2$ and $n = 4$.* Note that any set of three vertices that does not include the center cannot force. However, with a set of three vertices that includes the center, the center cannot force; in which case, leaks placed on the two blue cycle vertices will prevent the other vertices to be forced blue. It follows that four vertices are necessary. The Pigeonhole Principle guarantees that since each vertex has degree 3 or more, 2 leaks cannot prevent the sole remaining vertex from being forced.

*Case $\ell = 2$ and $n \geq 5$.* We focus on the following claims.

*Claim 1.* For any two adjacent cycle vertices $v_i, v_{i+1}$, the set $\{v_i, v_{i+1}\}$ is a fort.

*Proof of Claim 1.* Color all vertices except $v_i$ and $v_{i+1}$. Choose the set of leaks to be the blue cycle vertices adjacent to $v_i$ and $v_{i+1}$, $v_{i-1}$ and $v_{i+2}$. At this point, no vertex can force, and so $\{v_i, v_{i+1}\}$ are white. Hence, $\{v_i, v_{i+1}\}$ forms a fort. △

*Claim 2.* For any 2-leaky forcing set, $S$, if $v_i \in S$ then one of its two neighbors on the cycle is also in $S$.

*Proof of Claim 2.* We proceed by contrapositive. Without loss of generality, let $v_3$ be an initially blue vertex with $v_2$ and $v_4$ white. Suppose the two leaks are placed at $v_1$ and $v_5$ (these vertices are distinct as $n \geq 5$). Then, the only potential neighbors to force $v_2$ are $u$ or $v_3$; the same applies for $v_4$. However, both $u$ and $v_3$ have the same two white neighbors $v_2$ and $v_4$ that cannot be forced otherwise. Hence, the initial configuration with $v_3$ blue and $v_2$ and $v_4$ white cannot be a 2-leaky forcing set. △

Together with the two claims, it follows that among any three consecutive vertices on the cycle, two of them must be colored blue in a 2-leaky forcing set. (If only the first or last are blue, or none at all, Claim 1 is violated; if only the middle vertex among the three is blue, Claim 2 is violated.) Further, no forcing can occur unless

either the center vertex $u$ is blue or both the center vertex $u$ is white and all of the cycle vertices are blue. Hence, $Z_{(2)}(W_n) \geq \lceil 2n/3 \rceil + 1$ (e.g., there is at most one white vertex at every third vertex on the cycle). We now construct a 2-leaky forcing set that achieves this bound. Let $B$ be the set $\{u\} \cup \{v_i \mid i \equiv 1 \text{ or } 2 \mod 3\}$. Any white vertex, $v_a$ is adjacent to exactly three blue neighbors, two on the outer cycle, $v_{a-1}$ and $v_{a+1}$, and the center vertex, $u$. If either of the $v_{a-1}$ or $v_{a+1}$ are without a leak, that one will force $v_a$ immediately. Hence, the only way a vertex will not be immediately forced in this manner is if both $v_{a-1}$ and $v_{a+1}$ have a leak; in which case, all other vertices on the outer cycle become immediately colored blue (as their neighbors on the outer cycle cannot both have leaks). After which, $u$ can force $v_a$. This completes the proof for $n \geq 5$. □

## 4 Cartesian Products

As found in [17], the Cartesian (box) product of two finite graphs $G$ and $H$ is denoted $G \,\square\, H$. Its vertex set consists of $V(G) \times V(H)$, i..e, the ordered pairs $(x, y)$, where $x \in V(G)$ and $y \in V(H)$. Two vertices $(x, y)$ and $(x', y')$ are adjacent in the Cartesian product if and only if either both $x = x'$ in $G$ and $y$ is adjacent to $y'$ in $H$ or both $y = y'$ in $H$ and $x$ is adjacent to $x'$ in $G$. For example, the Cartesian product $P_3 \,\square\, P_5$ is the $3 \times 5$ grid of vertices, illustrated in Figure 1.

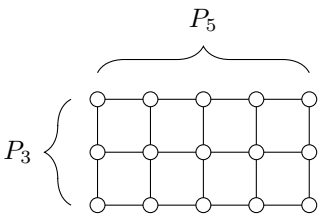


Figure 1: The Cartesian product $P_3 \,\square\, P_5$.

The $d$-dimensional hypercube, $Q_d$, is the graph

$$Q_d := \underbrace{K_2 \,\square\, \ldots \,\square\, K_2}_{d \text{ times}}.$$

In particular, $Q_1 = K_2$ and $Q_2 = C_4$.

It was shown in the original paper on zero forcing [1] that

$$Z(G \,\square\, H) \leq \min\{|V(G)|Z(H), |V(H)|Z(G)\}.$$

We demonstrate an analogous result for leaky forcing.

**Lemma 4.1.** *For graphs $G$ and $H$ and a number of leaks $\ell \geq 0$,*

$$Z_{(\ell)}(G \,\square\, H) \leq \min\{|V(G)|Z_{(\ell)}(H), |V(H)|Z_{(\ell)}(G)\}.$$

*Proof.* Without loss of generality, let us look at $G \,\square\, H$ as $|V(H)|$ copies of the graph $G$ and assume $|V(H)|Z_{(\ell)}(G) \leq |V(G)|Z_{(\ell)}(H)$. Choose a minimum $\ell$-leaky forcing set of $G$, $S$.

Color all vertices $S \times V(H)$ blue; that is, color all corresponding vertices to $S$ in all $|V(H)|$ copies of $G$ blue.

Arbitrarily choose $\ell$ leaks of $G \,\square\, H$: $(g_1, h_1), \ldots, (g_\ell, h_\ell)$. Let $L_G = \{g_1, \ldots, g_\ell\}$. We will show that $S \times V(H)$ forces the entirety of $G \times H$ with leaks $L_G \times V(H)$. Note that if $S = V(G)$, this holds trivially, so we will assume that $S \neq V(G)$

Let $S^{(0)} = S$ be the set of initially blue vertices of $G$. Let $S^{(i)}$ be the set of vertices colored blue after $i$ simultaneous applications of the color-change rule within $G$ when forcing with the set of leaks $L_G$. Since $S^{(0)} = S$ is an $\ell$-leaky forcing set of $G$, each $S^{(i)}$ is also an $\ell$-leaky forcing set of $G$.

Suppose $s \in S^{(0)}$ forces $s' \in S^{(1)} \setminus S^{(0)}$ during the first iteration of the color-change rule in $G$ (such an $s$ exists by assumption that $S \neq V(G)$). Then, for any $h \in V(H)$, $(s, h)$ has exactly one white neighbor, $(s', h)$. Hence, $(s, h)$ forces $(s', h)$ during the first iteration of the color-change rule in $G \,\square\, H$. Therefore, after one application of the color-change rule in $G \,\square\, H$, all vertices $S^{(1)} \times V(H)$ will be blue. Inductively, since $S^{(i)}$ is an $\ell$-leaky forcing set of $G$, whenever $S^{(i)} \times V(H)$ is colored blue, $S^{(i+1)} \times V(H)$ will become blue. It follows that all vertices of $G \times H$ will become blue. Hence, $S \times V(H)$ is an $\ell$-leaky forcing set of $G \,\square\, H$ with leaks $L_G \times V(H)$.

Since the set of $\ell$ leaks was chosen arbitrarily, this completes the proof. □

### 4.1 Leaky Forcing on Hypercubes

In this subsection, we will compute the $\ell$-leaky forcing number of hypercubes for some values of $\ell$ as well as provide upper bounds in other cases. We will view the hypercube $Q_d$ to have a vertex set of all length $d$ binary strings whereby two vertices are adjacent if and only if their strings differ in exactly one location (bit). For an example, see $Q_4$ in Figure 2.

Let us recall the zero forcing number of the hypercube.

**Lemma 4.2** (AIM Group, [1, Theorem 3.1])**.** *For $d \geq 1$,*

$$Z_{(0)}(Q_d) = 2^{d-1}.$$

**Proposition 4.3.** *For $d \geq 2$,*

$$Z_{(1)}(Q_d) = 2^{d-1}.$$

*Proof.* By Lemma 4.1, since $Q_d = Q_{d-2} \,\square\, C_4$, $Z_{(1)}(Q_d) \leq 2^{d-2} Z_{(1)}(C_4)$. By Proposition 3.4, $Z_{(1)}(C_4) = 2$, so altogether, $Z_{(1)}(Q_d) \leq 2^{d-1}$. For equality, we apply Lemmas 2.1 and 4.2: $Z_{(0)}(Q_d) = 2^{d-1} \leq Z_{(1)}(Q_d)$. □

Arguably, the easiest proof for the next proposition computing $Z_{(2)}(Q_3)$ is "the computer said so." (Simply try all $2^8$ possible subsets of vertices each with all 28 possible leak placements.) However, we present a more elegant proof using a fort intersection argument more in line with Proposition 2.3 and how our algorithm works in Section 5 based on the work in [5].

**Proposition 4.4.** *The 2-leaky forcing number of the 3-dimensional hypercube is given by*

$$Z_{(2)}(Q_3) = 6.$$

*Proof.* We will consider $Q_3$ a graph whose vertices are binary strings of length 3 whereby two vertices are adjacent if they differ in exactly one bit.

Color all vertices except 000 and 001 blue. We will show that this is a 2-leaky forcing set. If there is no leak on the vertex 100 or no leak on 010, then 000 will be colored blue which can then force 001. However, if both 100 and 010 have leaks, then 001 will become blue, after which, 001 forces 000.

To see that this is optimal, note any set of vertices consisting of all but a pair of vertices distance 2 apart, $u_1, u_2$, will not be 2-leaky forcing set (as the leaks could be on the vertices representing the complement strings of $u_1$ and $u_2$, preventing $u_2$ and $u_1$ from being forced respectively); hence, $\{u_1, u_2\}$ is a fort. By Proposition 2.3, at least one vertex of each such pair must be initially colored blue to be 2-leaky forcing set. The set system comprised of all such pairs of vertices can be viewed as a simple graph, $D$, on $V(Q_3)$ whose edge set consists of pairs vertices that differ in exactly two bits (e.g., they are distance 2 in $Q_3$). Any 2-leaky forcing set of $Q_3$ must be a transversal of $D$ (e.g., the set intersects every edge at least once). However, $D$ is simply a graph consisting of two disjoint copies of $K_4$. Any transversal of $K_4$ requires 3 vertices, so $D$ requires 6. It follows that any 2-leaky forcing of $Q_3$ requires 6 vertices. □

**Lemma 4.5.** *The following $\ell$-leaky forcing numbers hold.*

$$\begin{aligned} Z_{(2)}(Q_4) &= 8. \\ Z_{(3)}(Q_4) &= 10. \\ Z_{(3)}(Q_5) &= 16. \end{aligned}$$

□

Unfortunately, we do not have an elegant or easily verifiable proof of the above results, and this time, we omit a formal proof and resort to "the computer said so," using the computational techniques in Section 5.

For $Z_{(2)}(Q_4)$ and $Z_{(3)}(Q_5)$, one can check (hopefully by computer) that coloring any all of the vertices with a leading 0 in their binary representation (e.g. a "sub-cube") form a 2-leaky or 3-leaky forcing set respectively. These would be optimal as $Z_{(0)}(Q_4) = 8 \le Z_{(2)}(Q_4)$ and $Z_{(0)}(Q_5) = 16 \le Z_{(3)}(Q_5)$ by Lemmas 2.1 and 4.2.

For $Z_{(3)}(Q_4)$, an optimal 3-leaky forcing set for $Q_4$ can be found in Figure 2. We resort to using the integer programming techniques in Section 5.

MATLAB code used to perform these calculations can be found in [8].

**Proposition 4.6.** *For $d \ge 4$,*

$$Z_{(2)}(Q_d) = 2^{d-1}.$$

*Proof.* Since $Q_d = Q_{d-4} \,\square\, Q_4$, we can apply Lemmas 4.1 and 4.5 to yield: $Z_{(2)}(Q_d) \le 2^{d-4} Z_{(2)}(Q_4) = 2^{d-4} \cdot 8 = 2^{d-1}$. Equality is achieved, as $Z_{(0)}(Q_d) = 2^{d-1} \le Z_{(2)}(Q_d)$ by Lemmas 2.1 and 4.2. □

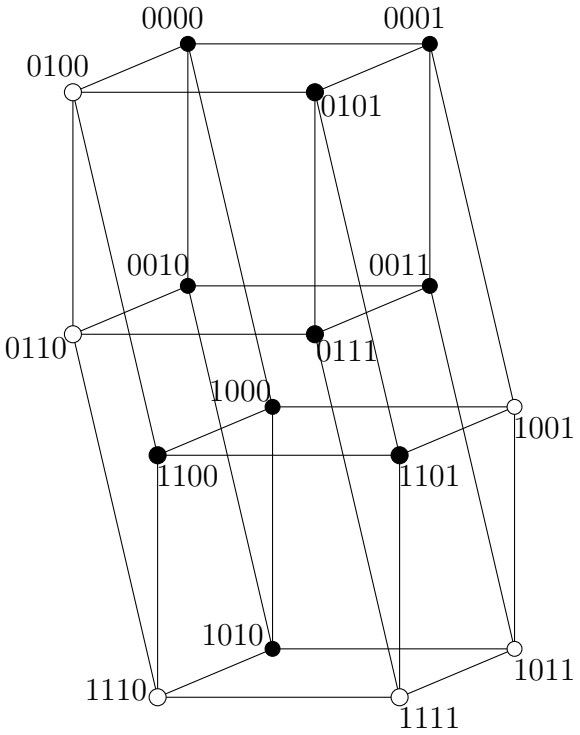


Figure 2: An optimal 3-leaky forcing set of $Q_4$.

**Proposition 4.7.** *For $d \geq 5$,*

$$Z_{(3)}(Q_d) = 2^{d-1}.$$

*Proof.* By Lemmas 4.1 and 4.5, we have $Q_d = Q_{d-5} \,\square\, Q_5$, $Z_{(3)}(Q_d) \leq 2^{d-5} Z_{(3)}(Q_5) = 2^{d-5} \cdot 16 = 2^{d-1}$. As before, equality follows by applying Lemmas 2.1 and 4.2. □

A summary of the known values of $Z_{(\ell)}(Q_d)$ can be found in Table 1. In the preprint version of this article, we commented that it appeared as though $Z_{(d-2)}(Q_d) = 2^{d-1}$; Herrman recently proved this equality.

**Theorem 4.8** (Herrman [12, Theorem 1.1])**.** *For $d \geq 3$,*

$$Z_{(d-2)}(Q_d) = 2^{d-1}.$$

The nontrivial values of $Z_{(d-1)}(Q_d)$ do not appear to be powers of 2; we hesitate to conjecture any pattern from only two values. Hence, we ask the following.

**Question 4.9.** What are the explicit values of $Z_{(d-1)}(Q_d)$?

### 4.2 Leaky Forcing on Grids

A *grid*, is the graph $P_n \,\square\, P_m$ for $m, n \geq 2$. In this section, we focus on computing the 1-leaky forcing number of a grid, $Z_{(1)}(P_n \,\square\, P_m)$.

In the preprint version of this article, we proved the following.

| | | $d$ | | | |
|---|---|---|---|---|---|
| | | 3 | 4 | 5 | 6 |
| $\ell$ | 0 | 4 | 8 | 16 | 32 |
| | 1 | 4 | 8 | 16 | 32 |
| | 2 | 6 | 8 | 16 | 32 |
| | 3 | 8 | 10 | 16 | 32 |
| | 4 | ⋮ | 16 | ? | *32* |

Table 1: Summary of values of $Z_{(\ell)}(Q_d)$. The value $Z_{(d-2)}(Q_d) = 2^{d-1}$ for $d \geq 6$ was determined by Herrman [12].

**Theorem 4.10** (See [9])**.** *For the grid $P_n \,\square\, P_m$ with $m \geq n$,*

$$Z_{(1)}(P_n \,\square\, P_m) \leq 2m - n.$$

The preprint version of this article also showed for select cases of $m, n$, $Z_{(1)}(P_n \,\square\, P_m) \leq \min(2n, m)$. These results have since been superseded by the following result for all grids.

**Theorem 4.11** (A.-Kritschgau-Warnberg-Young [2, Theorem 4.10])**.** *For the grid $P_n \,\square\, P_m$, with $m \geq n$,*

$$Z_{(1)}(P_n \,\square\, P_m) \leq \min(2n, m).$$

We have omitted full proofs of these statements. However, to showcase the complicated nature of these constructions, illustrations of 1-leaky forcing sets used to prove Theorem 4.10 are displayed in Figure 3. For a full proof of Theorem 4.10, the reader is referred to the preprint version of this article [9]. The proof of Theorem 4.11 provides an improved construction. The reader is referred to [2] for details.

Note that the both of these theorems only provide an upper bound. We remark that computational evidence using the techniques provided in Section 5, it appears that $Z_{(1)}(P_n \,\square\, P_m) = \min(2n, m)$, but so far this remains elusive. To our understanding this problem remains open.

However, for $n = m$ have the following.

**Theorem 4.12.** *For $n \geq 1$, the grid $P_n \,\square\, P_n$,*

$$Z_{(1)}(P_n \,\square\, P_n) = n.$$

To prove this we use the following result.

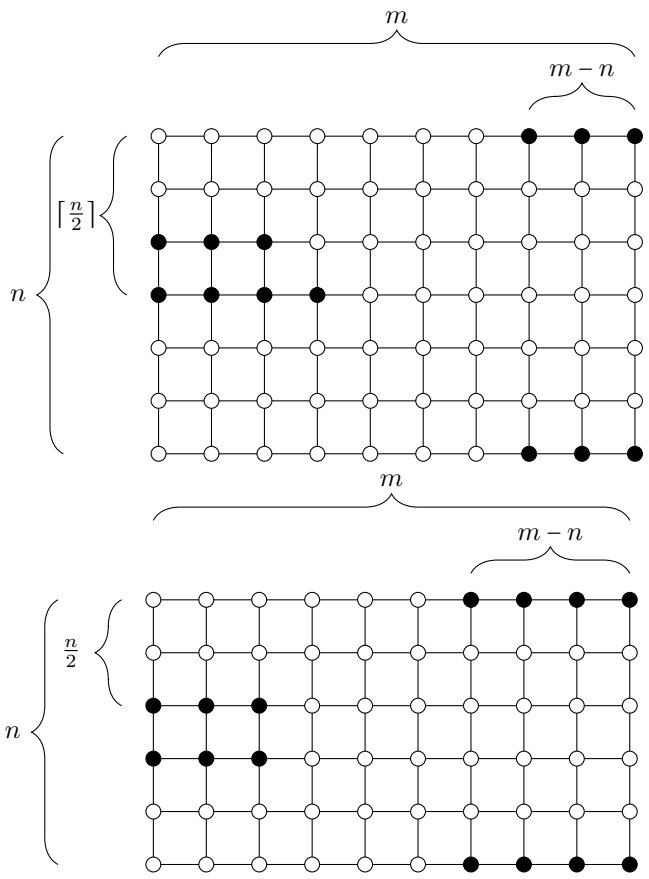


Figure 3: Patterns for $n$ odd (top) and $n$ even (bottom) for the proof of Theorem 4.10 given in [9].

**Proposition 4.13** (AIM Group [1, Corollary 3.7])**.** *For $1 \le n \le m$, the grid $P_n \,\square\, P_m$ has*

$$Z_{(0)}(P_n \,\square\, P_m) = n.$$

*Proof of Theorem 4.12.* By Theorem 4.10, $Z_{(1)}(P_n \,\square\, P_n) \le 2n - n = n$. A lower bound is achieved by Lemmas 2.1 and Proposition 4.13:

$$Z_{(1)}(P_n \,\square\, P_n) \ge Z_{(0)}(P_n \,\square\, P_n) = n.$$

□

## 5 Integer Program Methods For Leaky Forcing

In this section, we develop a constraint generation method to calculate the $\ell$-leaky forcing number similar to [5, Model 2 and Theorem 8], using Proposition 2.3. The full integer program can be found in Model 5.1 (overleaf) and the algorithm using constraint generation is found in Algorithm 5.3. Finally, we provide some computational results, including those related to the results in Section 4.

**Model 5.1.** A fort-covering integer programming model given a set of forts $\mathcal{F} \subset 2^{V(G)}$. Later, we will refer to this model (as a function) as $IP(G, \mathcal{F})$ whose output is a candidate for an $\ell$-leaky forcing set. Since the inputs include a set of forts, the number of leaks, $\ell$, is implicitly included in the choice of forts. Ideally, $\mathcal{F}$ is the set of all $\ell$-leaky forts for some $\ell$. Indeed, when $\mathcal{F}$ is the set of *all* $\ell$-leaky forts, Theorem 5.2 guarantees the model returns a minimum $\ell$-leaky set. However, since generating all $\ell$-leaky forts is impractical, we leverage this model using an incomplete set of forts in Algorithm 5.3.

| | | |
|---|---|---|
| **Inputs** | | |
| $G$ | | a graph |
| $\mathcal{F}$ | | a set of forts |
| **Other sets** | | |
| $V$ | | the set of vertices of $G$ |
| **Parameters** | | |
| $f_{k,v}$ | $k \in \mathcal{F}, v \in V$ | indicator variable whether vertex $v$ is in fort $k$ |
| **Decision Variables** | | |
| $x_v$ | $v \in V$ | indicator variable whether vertex $v$ in the candidate $\ell$-leaky forcing set $S$ |

**Objective and Constraints**

$$\min \sum_{v \in V} x_v \tag{1a}$$

$$\text{s.t.} \sum_{v \in V} f_{k,v} x_v \geq 1 \quad \forall k \in \mathcal{F}, \tag{1b}$$

$$x_v \in \{0, 1\} \quad \forall v \in V. \tag{1c}$$

| | | |
|---|---|---|
| **Outputs** | | |
| $S$ | | candidate $\ell$-leaky forcing set comprised of the set of vertices $v \in V$ for which $x_v = 1$ |

**Theorem 5.2.** *When $\mathcal{F}$ is the set of all $\ell$ leaky forts of $G$, Model 5.1 returns an optimal $\ell$-leaky forcing set $S$.*

*Proof.* Constraint 1b requires that $S$ have at least one element in every fort of $\mathcal{F}$. Hence by Lemma 2.3, $S$ is an $\ell$-leaky forcing set. The objective 1a requires further requires that this be optimal. □

Algorithm 5.3 provides a process to compute an optimal $\ell$-leaky forcing set without necessarily enumerating all $\ell$-leaky forcing sets. An overview of the algorithm is as follows. First, an initial set of forts is initialized; necessarily $V(G)$ is a fort, but also as a consequence of Lemma 2.4, each individual vertex with degree at most $\ell$ forms a fort. As a note, one could augment this initialization using other forts (perhaps by computing $V(G) \setminus cl(S, L)$ for random choices of $S, L \subseteq V$ or by leveraging the specific structure of $G$, etc.). Then, Model 5.1 is run using the current set of forts. If the result is an $\ell$-leaky forcing set, we are done; if not, there are additional forts to add and Model 5.1 can be rerun with those additional forts. This iterates until an optimal $\ell$-leaky forcing set is found. We note that this algorithm still computes optimal zero forcing sets (in the original sense) when $\ell = 0$. The proof termination and accuracy of Algorithm 5.3 is given in Theorem 5.4.

**Algorithm 5.3** Finding an optimal $\ell$-leaky forcing set for a graph $G$
**Input:** A graph $G$ and a nonnegative integer $\ell$ (e.g., the number of leaks)
**Output:** An optimal $\ell$-leaky forcing set of $G$, $S$

(Initialize a set of forts, $\mathcal{F}$: The set of all vertices is a fort; also, by Lemma 2.4, each vertex with degree at most $\ell$ is a fort.)
$\mathcal{F} \leftarrow \{V(G)\}$
**for** each $r \in V(G)$ with degree at most $\ell$ **do**
    Add $\{r\}$ to $\mathcal{F}$.

(Run Model 5.1 to generate a candidate $\ell$-leaky forcing set $S$)
$S \leftarrow IP(G, \mathcal{F})$

(While $S$ is not an $\ell$-leaky forcing set, generate new forts by computing the closure of $S$ for all possible sets of leaks, and rerun Model 5.1.)
**while** $S$ is not an $\ell$-leaky forcing set on $G$ **do**
    **for** all sets of possible sets of leaks $L$ **do**
        **if** $V(G) \setminus cl(S, L) \neq \varnothing$ **then**
            Add $V(G) \setminus cl(S, L)$ to $\mathcal{F}$
    $S \leftarrow IP(G, \mathcal{F})$

(When $S$ is finally an $\ell$-leaky forcing set, return $S$)
**return** $S$

**Theorem 5.4.** *Algorithm 5.3 terminates and returns an optimal $\ell$-leaky forcing set of $G$.*

*Proof.* To see that Algorithm 5.3 terminates, at each iteration of the loop, either the set $S$ calculated from $IP(G, \mathcal{F})$ will be an $\ell$-leaky forcing set or it will not. In the first case, we are done. For the latter case, there will be an additional fort to be added to $\mathcal{F}$ that was not previously included. In the worst case, $\mathcal{F}$ will eventually include all, finitely many forts. At which point, by Proposition 2.3, $IP(G, \mathcal{F})$ will necessarily return an $\ell$-leaky forcing set and the loop will terminate.

Let $S$ be the returned set of Algorithm 5.3. To see that $|S| = Z_{(\ell)}(G)$, note that the result of each iteration of $IP(G, \mathcal{F})$ calculates a minimum set of vertices that intersect some set of forts. Hence, it follows that for every iteration, $|S| \leq Z_{(\ell)}(G)$, as adding more constraints (i.e., more forts) cannot decrease the objective value which is at most $Z_{(\ell)}(G)$ (achieved whenever $\mathcal{F}$ is the set of *all* forts). On the other hand, $S$ is necessarily a leaky forcing set, so $|S| \geq Z_{(\ell)}(G)$. □

### 5.1 Computational Results

We first tested the runtimes of our algorithm and integer program on various families of graphs. The MATLAB code containing the functions and scripts of these computations can be found in [8].

First, we computed $Z_{(1)}(P_m \,\square\, P_m)$ for values of $m = 3, 4, 5$, and 6. Our results are detailed in Table 2. Each of these four graphs has the same structure, so the dramatic increase in runtime shows that our integer program is sensitive to the number of vertices in the graph.

| $m$ | $\lvert V\rvert$ | $Z_{(1)}(P_m \,\square\, P_m)$ | Time (seconds) |
|---|---|---|---|
| 3 | 9 | 3 | 0.7851 |
| 4 | 16 | 4 | 7.9365 |
| 5 | 25 | 5 | 101.3456 |
| 6 | 36 | 6 | 1134.9772 |

Table 2: Runtimes for finding $Z_{(1)}(P_m \,\square\, P_m)$.

We also tested the runtimes of our algorithm and integer program on nine cubic graphs from the Wolfram database [18] with one leak, the results of which are found in Table 3. Cubic graphs are those in which all vertices have a degree of 3. Since these nine graphs have very similar structures, we can more easily compare their runtimes. Just as with the $P_m \,\square\, P_m$ graphs, we can see that the number of vertices does affect the runtime. However, within a fixed number of vertices, smaller zero forcing numbers have smaller runtimes. Since our method begins with the smallest potential $\ell$-leaky forcing set and iteratively adds more vertices to this set as it runs, this increase in time makes sense as $Z_{(\ell)}(G)$ increases.

Comparable methods were timed on cubic graphs with no leaks from [5]. Their results for cubic graphs on 20 vertices yielded an average runtime of 0.40 seconds, and for cubic graphs on 30 vertices yielded an average runtime of 2.05 seconds. Although our runtimes are not nearly as fast, our algorithm took into account the potential for a leak to be on any vertex in the graph, significantly increasing the complexity of our process.

Lastly, we computationally investigated the 1-leaky forcing number of random graphs using the traditional Erdős-Reyni graph model, $G(n,p)$ with $p = 1/4$ (i.e., there are $n$ vertices and an edge between a pair of vertices exists with probability $p = 1/4$ independent of all other pairs). Our results and runtimes are reported in Table 4. We encounter the similar growth in runtime around 20 vertices as before. In addition, there is a gap between $Z(G)$ and $Z_{(1)}(G)$, suggesting that Lemma 2.1 is often not sharp (e.g., an additional leak will often cause the forcing number to increase). It is difficult to compare these computations for small graphs to previous asymptotic results on zero forcing for random graphs. In their work, Kalinowski, Kamcev and Sudakov show that for certain ranges of $p$, $Z(G(n,p))$ is almost always all of the vertices; notably, $Z(G(n,1/2)) = n - (2+\sqrt{2}+o(1))\log_2 n$ asymptotically almost surely [13]. It is unclear if these same constants would hold for the case of 1-leaky forcing.

| Graph name | $\lvert V\rvert$ | $Z_{(1)}(G)$ | Time (seconds) |
|---|---|---|---|
| Cubic_20_1 | 20 | 6 | 101.2786 |
| Cubic_20_2 | 20 | 6 | 75.9405 |
| Cubic_20_3 | 20 | 7 | 223.4605 |
| Cubic_22_1 | 22 | 7 | 392.8827 |
| Cubic_22_2 | 22 | 7 | 301.0180 |
| Cubic_24_1 | 24 | 8 | 1058.7689 |
| Cubic_24_2 | 24 | 6 | 481.3399 |
| Cubic_24_3 | 24 | 8 | 927.1612 |
| Cubic_24_4 | 24 | 8 | 968.2742 |

Table 3: Values and runtimes for $Z_{(1)}(G)$ for various cubic graphs.

## 6 Conclusion

We introduced zero forcing and $\ell$-leaky forcing and provided the $\ell$-leaky forcing numbers of various families of graphs, such as paths, cycles, complete graphs, and wheels. Additionally, we explored $\ell$-leaky forcing on hypercubes and grids. We used

| $n$ | Trials | Mean $Z(G(n,1/4))$ | Mean $Z_{(1)}(G(n,1/4))$ | Mean $Z_{(1)}$ Time (sec.) |
|---|---|---|---|---|
| 10 | 100 | 3.80 | 5.38 | 0.1924 |
| 11 | 100 | 4.05 | 5.49 | 0.3452 |
| 12 | 10 | 4.5 | 5.6 | 0.388 |
| 13 | 10 | 4.6 | 6.1 | 1.164 |
| 14 | 10 | 4.7 | 6.4 | 2.084 |
| 15 | 10 | 5.0 | 5.9 | 2.972 |
| 16 | 10 | 5.2 | 7.0 | 6.350 |
| 17 | 10 | 5.6 | 7.4 | 14.191 |
| 18 | 10 | 6.2 | 7.9 | 13.891 |
| 19 | 10 | 6.7 | 8.1 | 60.938 |
| 20 | 10 | 7.1 | 8.7 | 150.145 |

Table 4: Values and runtimes for random graphs, $Z_{(1)}(G(n,1/4))$.

the idea of forts in graphs as the foundation for our algorithm and integer program which find the $\ell$-leaky forcing number for any finite, connected graph.

For future work, there are a couple of appetizing questions for specific graphs. First, computing $Z_{(d-1)}(Q_d)$ remains open (Question 4.9). Second, for $m \geq n$, proving that $Z_{(1)}(P_n \,\square\, P_m) = \min(m, 2n)$ (e.g., equality) remains elusive. Also, given the exponential growth of runtimes for our algorithm, improving the computational approach would be helpful for future study of leaky forcing.

Finally, we are ever grateful to the reviewers and referees who have made substantial comments toward improving this manuscript. Thank you.

## References

[1] "AIM Special Work Group", Zero forcing sets and the minimum rank of graphs, *Linear Algebra Appl.* 428 (7) (2008), 1628–1648.

[2] J. S. Alameda, J. Kritschgau, N. Warnberg and M. Young, On leaky forcing and resilience, *Discrete Appl. Math.* 306 (2022), 32–45.

[3] M. Allison, E. Bodine, L. M. DeAlba, J. Debnath, L. DeLoss, C. Garnett, J. Grout, L. Hogben, B. Im, H. Kim et al., Minimum rank of skew-symmetric matrices described by a graph, *Linear Algebra Appl.* 432 (2010).

[4] 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 Appl.* 433 (2) (2010), 401–411.

[5] B. Brimkov, C. C. Fast and I. V. Hicks, Computational approaches for zero forcing and related problems, *Eur. J. Oper. Res.* 273 (3) (2019), 889–903.

[6] D. J. Brueni and L. S. Heath, The PMU placement problem, *SIAM J. Discrete Math.* 19 (3) (2005), 744–761.

[7] D. Burgarth, D. D'Alessandro, L. Hogben, S. Severini and M. Young, Zero forcing, linear and quantum controllability for systems evolving on networks, *IEEE Trans. Automat. Contr.* 58 (9) (2013), 2349–2354.

[8] S. Dillman and F. Kenter, MATLAB code for Leaky Forcing, `https://github.com/fkenter/leakyforcing`.

[9] S. Dillman and F. Kenter, Leaky forcing: a new variation of zero forcing, *arXiv preprint arXiv:1910.00168* (2019).

[10] C. C. Fast and I. V. Hicks, Effects of vertex degrees on the zero-forcing number and propagation time of a graph, *Discrete Appl. Math.* 250 (2018), 215–226.

[11] T. W. Haynes, S. M. Hedetniemi, S. T. Hedetniemi and M. A. Henning, Domination in graphs applied to electric power networks, *SIAM J. Discrete Math.* 15 (4) (2002), 519–529.

[12] R. Herrman, The $(d-2)$-leaky forcing number of $q_d$ and $\ell$-leaky forcing number of $GP(n,1)$, *Discrete Optim.* 46 (2022), 100744.

[13] T. Kalinowski, N. Kamcev and B. Sudakov, The zero forcing number of graphs, *SIAM J. Discrete Math.* 33 (1) (2019), 95–115.

[14] F. H. Kenter and J. C.-H. Lin, On the error of a priori sampling: zero forcing sets and propagation time, *Linear Algebra Appl.* 576 (2019), 124–141.

[15] N. Monshizadeh, S. Zhang and M. K. Camlibel, Zero forcing sets and controllability of dynamical systems defined on graphs, *IEEE Trans. Automat. Contr.* 59 (9) (2014), 2562–2567.

[16] K.-J. Pai, J.-M. Chang and Y.-L. Wang, Restricted power domination and fault-tolerant power domination on grids, *Discrete Appl. Math.* 158 (10) (2010), 1079–1089.

[17] V. G. Vizing, The Cartesian product of graphs, *Vyčisl. Sistemy* No. 9 (1963), 30–43.

[18] Wolfram Research, GraphData, Wolfram language function, (updated 2023), `https://reference.wolfram.com/language/ref/GraphData.html`, 2007.

[19] M. Zhao, L. Kang and G. J. Chang, Power domination in graphs, *Discrete Math.* 306 (15) (2006), 1812–1816.

(Received 13 Jan 2023; revised 19 Aug 2023, 8 Jan 2024)