Submitted to *INFORMS Journal on Computing*
manuscript (Please, provide the manuscript number!)

Improved computational approaches and heuristics for zero forcing

Boris Brimkov
Department of Mathematics and Statistics, Slippery Rock University, Slippery Rock, PA 16057. boris.brimkov@sru.edu

Derek Mikesell
Department of Computational and Applied Mathematics, Rice University, Houston, TX 77005. derekmikesell@gmail.com

Illya V. Hicks
Department of Computational and Applied Mathematics, Rice University, Houston, TX 77005. ivhicks@rice.edu

Zero forcing is a graph coloring process based on the following color change rule: all vertices of a graph G are initially colored either blue or white; in each timestep, a white vertex turns blue if it is the only white neighbor of some blue vertex. A zero forcing set of G is a set of blue vertices such that all vertices eventually become blue after iteratively applying the color change rule. The zero forcing number $Z(G)$ is the cardinality of a minimum zero forcing set. In this paper, we propose novel exact algorithms for computing $Z(G)$ based on formulating the zero forcing problem as a two-stage Boolean satisfiability problem. We also propose several heuristics for zero forcing based on iteratively adding blue vertices which color a large part of the remaining white vertices. These heuristics are used to speed up the exact algorithms, and can also be of independent interest in approximating $Z(G)$. Computational results on various types of graphs show that in many cases, our algorithms offer a significant improvement on the state-of-the-art algorithms for zero forcing.

Key words: Zero forcing, Boolean satisfiability, heuristic, constraint generation

1. Introduction

Given a simple undirected graph $G = (V, E)$ whose vertices are colored blue or white, the *zero forcing color change rule* dictates that at each timestep, a blue vertex with a single white neighbor causes (or *forces*) that neighbor to become blue. If $S \subset V$ is a set of blue vertices in G, the *closure* of S, denoted $cl(S)$, is the set of blue vertices obtained after the color change rule is applied until no new vertex can be turned blue; it can be shown that $cl(S)$ is uniquely determined by S (AIM Special Work Group 2008). A *zero forcing set* is a set whose closure is all of V; the *zero forcing number* of G, denoted $Z(G)$, is the minimum cardinality of a zero forcing set.

Zero forcing was introduced in (AIM Special Work Group 2008) as a bound on the minimum rank over all symmetric matrices whose entries have the same off-diagonal zero-nonzero pattern as the adjacency matrix of a graph G. This minimum rank problem is a special case of the matrix completion problem which has numerous theoretical and practical applications (such as the million-dollar Netflix challenge (Koren 2009)). The minimum rank problem is extremely hard to solve directly: despite extensive research (Barioli et al. 2010, 2004, DeLoss et al. 2010, Fallat and Hogben 2007, Hsieh 2001, Huang et al. 2010, Nylen 1996), there is still no known algorithm for computing the minimum rank of an arbitrary graph in finite time. While computing $Z(G)$ is also NP-Hard in general (Aazami 2008), it is more tractable than the minimum rank problem. Indeed, the best bounds for the minimum rank problem are based on zero forcing and variants thereof. Thus, improved computational approaches and heuristics for zero forcing lead to better approximations and bounds for the minimum rank problem.

Aside from its original application in the minimum rank problem, zero forcing is also related to other processes that arise from the principle that when all-but-one variables of an equation are known, the remaining unknown variable can be determined. In particular, processes that are equivalent to zero forcing were independently studied in quantum physics (Burgarth and Giovannetti 2007) and theoretical computer science (Yang 2013). In electrical engineering, the phase measurement unit (PMU) placement problem can be formulated as applying the zero forcing color change rule to the closed neighborhood of a set of initially blue vertices; see (Haynes et al. 2002) for the original definition of the PMU placement problem, and (Brueni and Heath 2005) for an equivalent simplified formulation. Another closely related problem to zero forcing is the target set selection problem, where given a set S of blue vertices and a threshold function $\theta : V(G) \to \mathbb{Z}$, all white vertices v that have at least $\theta(v)$ blue neighbors become blue. Zero forcing has also found a variety of uses in physics, logic circuits, coding theory, and in modeling the spread of diseases and information in social networks; see (Barioli et al. 2013, Burgarth and Giovannetti 2007, Burgarth et al. 2015, Trefois and Delvenne 2015) and the bibliographies therein. Several computational methods have been proposed for zero forcing (Agra et al. 2019, Brimkov et al. 2019a, Butler et al. 2014), PMU placement (Aazami 2010, Bozeman et al. 2019, Brimkov et al. 2019b, Mahaei and Hagh 2012), target set selection (Ackerman et al. 2010, Ben-Zwi et al. 2011, Chiang et al. 2013), and related problems; these methods are typically

based on integer programming or combinatorial optimization. Often, techniques and algorithms developed for one of the aforementioned problems can also be adapted and applied to the others.

One of the state-of-the-art approaches for computing the zero forcing number of a graph is a combinatorial algorithm called Wavefront based on dynamic programming, developed by Butler et al. (Butler et al. 2014). Brimkov et al. (Brimkov et al. 2019a) proposed several other competitive approaches based on integer programming, constraint generation, and branch-and-bound. Agra et al. (Agra et al. 2019) presented a compact integer programming formulation for zero forcing, and gave a heuristic to derive upper bounds on the zero forcing number. The performance of these approaches depends on various factors, such as the density of the graph, the presence or absence of certain subgraphs, and the structure of the zero forcing polytope (i.e., the convex hull of the set of all feasible solutions of the problem; see Section 1.1 for the exact definition, and see (Brimkov et al. 2019a) for a discussion about the effects of the polyhedral structure).

In this paper, we present a novel approach for solving the zero forcing problem by formulating it as a Boolean satisfiability problem (SAT) and using SAT solvers to obtain a solution. We also present novel heuristics for quickly obtaining (not necessarily minimum) zero forcing sets; these heuristics can be used to speed up the exact SAT-based models, and can also be of independent interest. We study certain sets of vertices which must be included or excluded from minimum zero forcing sets; we also characterize when certain forts are facets of the zero forcing polytope and show that the number of such facets can be exponential in the order of the graph. These results can be used to reduce the number of clauses in the presented SAT models and to improve integer programming methods for zero forcing. Finally, we compare our methods to existing methods on various standard benchmark graphs; our computational experiments show that in many cases, our methods are faster than the leading algorithms from the literature and can solve the problem on larger graphs.

The paper is organized as follows. In the remainder of this section, we recall some graph theoretic notions and notations. In Section 2, we give several models for zero forcing as a Boolean satisfiability optimization problem. In Section 3, we present several variants of a heuristic for zero forcing. In Section 4, we identify certain sets of vertices which must be included or excluded from some minimum zero forcing set, and characterize when

some of these sets are facets of the zero forcing polytope. In Section 5, we describe the implementation of our proposed approaches and compare them with the leading methods from the literature through computational experiments on various types of graphs. We conclude with some final remarks and open questions in Section 6.

1.1. Preliminaries

A simple graph $G = (V, E)$ consists of a vertex set V and an edge set E of two-element subsets of V. The *order* of G is denoted by $n = |V|$. Two vertices $v, w \in V$ are *adjacent*, or *neighbors*, if $\{v, w\} \in E$. The *neighborhood* of $v \in V$ is the set of all vertices which are adjacent to v, denoted $N(v)$; the *closed neighborhood* of v, denoted $N[v]$, is the set $N(v) \cup \{v\}$. The *degree* of $v \in V$ is defined as $\deg(v) = |N(v)|$. Given $S \subset V$, the *induced subgraph* $G[S]$ is the subgraph of G whose vertex set is S and whose edge set consists of all edges of G which have both endpoints in S. If u is a degree two vertex of G with neighbors v and w, *subdividing* u means deleting u, adding vertices u' and u'', and adding edges $\{v, u'\}$, $\{u', u''\}$, and $\{u'', w\}$. For other graph theoretic terminology and definitions, we refer the reader to (West 2001).

A graph is *empty* if all of its vertices have degree 0, and *cubic* if all of its vertices have degree 3. We will call a graph $G = (V, E)$ *average-cubic* if it has average degree 3, i.e., if $|E| = 3|V|/2$. A *Watts-Strogatz* graph with parameters n, k, and β is a graph obtained from the graph $C(n, k) = (\{0, \ldots, n-1\}, \{\{i, j\} : 0 \leq i, j \leq n-1, |i-j| \leq k/2\})$ by replacing each edge of $C(n, k)$ with probability β by a randomly chosen edge. When n is clear from the context, we will refer to the Watts-Strogatz graphs with parameters n, k, and β as WS(k, β). Watts-Strogatz graphs are a popular random graph model and feature small-world properties such as high clustering and short average path lengths (cf. (Watts and Strogatz 1998)).

A *fort* of a graph $G = (V, E)$ is a non-empty set $F \subset V$ such that no vertex outside F is adjacent to exactly one vertex in F. $\mathcal{B}(G)$ denotes the set of all forts of G; when there is no scope for confusion, dependence on G will be omitted. Given a set of vertices S, a vertex $v \in S$ is on the *border* of S if v has a neighbor outside S.

Given a graph $G = (V, E)$, the *zero forcing polytope* of G, as defined in (Brimkov et al. 2019a), is the convex hull of the subsets of V which intersect all forts of G, i.e., $conv(\{x \in \{0,1\}^n : \sum_{v \in B} x_v \geq 1 \quad \forall B \in \mathcal{B}\})$. The inequality $a^T x \leq b$ is a *valid inequality* for a polytope P if $a^T x \leq b$ for all $x \in P$. The set $\{x \in P : a^T x = b\}$ is a *face* of a polytope P if $a^T x \leq b$

is a valid inequality for P. A *facet* of P is a maximal face of P distinct from P. A valid inequality $a^T x \leq b$ is *facet-defining* for P if $\{x \in P : a^T x = b\}$ is a facet of P.

2. Boolean satisfiability models for zero forcing

In this section, we give several models of the zero forcing problem as a two-stage Boolean satisfiability problem. We explore ways to speed up these models in Sections 3 and 4 and compare them computationally in Section 5. Some of the models in this section are Boolean conversions of integer programming models from (Brimkov et al. 2019a), while others are novel formulations. A comparison of analogous models solved by different paradigms and solvers may be interesting in its own right.

There are several papers in the literature that have used SAT solvers as workhorses for solving combinatorial optimization problems, and have compared integer programming and SAT based solution methods; see, e.g., (Li et al. 2004, Wolfman and Weld 2001, Manquinho et al. 1998, Nieuwenhuis 2015, Zahidi et al. 2012, Aloul 2005) and the bibliographies therein. The computational results of these papers generally show that the two approaches are complementary, with some recent papers finding that SAT based methods tend to outperform IP methods for certain problems (Drummond et al. 2015, Aloul 2005). Empirically, integer programming approaches seem to work better for problems that are amenable to decomposition or that have non-binary variables, whereas SAT based approaches seem to work better for problems that can be represented in conjunctive normal form using few additional clauses and without bottleneck constraints (Li et al. 2004, Nieuwenhuis 2015). While SAT methods have been used to solve a variety of graph optimization problems including graph colorability, vertex cover, hamiltonian path, and independent set (Creignou et al. 2001, Ramani et al. 2006), to our knowledge this is the first time a SAT approach is used to solve a timestep-based dynamic graph coloring problem.

Given a collection of sets $\mathcal{S}$, a *hitting set* for $\mathcal{S}$ is a set which intersects every set in $\mathcal{S}$. Model 1 presents zero forcing as the problem of finding a minimum hitting set of $\mathcal{B}$, the set of all forts of the graph; the Boolean variable s_v is TRUE if vertex v is in the zero forcing set.

Model 1 *Boolean SAT model for zero forcing as a hitting set problem*

$$\min \quad \sum_{v \in V} s_v$$

$$\text{s.t.:} \quad \bigwedge_{B \in \mathcal{B}} \left(\bigvee_{v \in B} s_v \right) \tag{1}$$

PROPOSITION 1. *The optimal value of Model 1 is $Z(G)$, and the optimal solution is a minimum zero forcing set.*

Proof: Let s be a feasible solution of Model 1 and let S be the set of vertices for which s_v is TRUE. Suppose S is not a zero forcing set, i.e. $cl(S) \neq V$. Then, by the definition of a fort, $B = V \setminus cl(S)$ is a fort such that $B \cap S = \emptyset$. This violates Clause (1), contradicting the feasibility of s. Therefore, S must be a zero forcing set. Conversely, it was shown in (Brimkov et al. 2019a) that any zero forcing set S of G must intersect every fort of G, so letting s_v be TRUE for $v \in S$ gives a feasible solution to Model 1. □

Next we give two extensions of Model 1 with a different set of variables that can encode more information. In Models 2 and 3, let $T(B) = \{w \in V : \exists v \in B \ s.t. \ v \in cl(N[w])\}$. $T(B)$ is the set of vertices w for which the closure of $N[w]$ intersects B; in other words, all forcing chains passing through a vertex of $T(B)$ will eventually reach B. Also, let $\bar{N}[v]$ be a set of all-but-one neighbors of v, i.e., $\bar{N}[v] = N[v] \setminus \{u\}$ for some $u \in N(v)$. Since u can always be forced by v, the choice of u does not matter as long as it is consistent; see Section 5 for implementation details for the choice of u. In Models 2 and 3, s_v is a Boolean variable which is TRUE if v is in the zero forcing set, and z_v is a Boolean variable which is TRUE if v and all-but-one neighbors of v are in the zero forcing set.

The constraints of Models 2 and 3 ensure that each zero forcing set contains some vertex together with all-but-one of its neighbors (in order for any forces to be able to occur). The main difference between the two models is that Model 2 counts the z_v variables as part of the objective, while Model 3 has a clause which ensures that if z_v is TRUE, then s_u must be TRUE for all-but-one neighbors u of v; thus, only the s_v variables are counted in the objective. In Model 2, it is not necessarily the case that if v is in the zero forcing set, then s_v is TRUE (since it could be the case that z_v is TRUE). Model 2 is analogous to IP Model 5 from Brimkov et al. (2019a) (where constraint (4) is equivalently encoded as $s_w + z_v \leq 1$).

Model 2 *Boolean SAT model for zero forcing with neighborhood variables (Variant 1)*

$$\min \quad \sum_{v \in V} (\deg(v) z_v + s_v)$$

$$\text{s.t.:} \quad \underbrace{\bigwedge_{B\in\mathcal{B}}\left(\bigvee_{v\in B} s_v \vee \bigvee_{v\in T(B)} z_v\right)}_{(2)} \wedge \underbrace{\left(\bigvee_{v\in V} z_v\right)}_{(3)} \wedge \underbrace{\left(\bigwedge_{\substack{v\in V\\ w\in cl(N[v])}} (\neg s_w \vee \neg z_v)\right)}_{(4)}$$

PROPOSITION 2. *The optimal value of Model 2 is $Z(G)$, and the optimal solution is a minimum zero forcing set.*

Proof: Let s, z be a feasible solution of Model 2. Let S be the set of vertices for which s_v is TRUE, Z be the set of vertices for which z_v is TRUE, and let $X = S \cup \bigcup_{v\in Z} \bar{N}[v]$. If X is not a zero forcing set, then $B = V \setminus cl(X)$ is a fort. Clause (2) assures that each fort B either intersects X or is eventually reached by some forcing chain passing through $T(B)$. However, if $B = V\setminus cl(X)$ is a fort, then neither of these conditions holds for B (because B does not intersect X nor $cl(X)$). Thus, Clause (2) is not satisfied, contradicting the feasibility of s, z. Thus, X is a zero forcing set. Note the choice of $\bar{N}[v]$ for each v in the definition of X does not matter, since any choice uniquely determines whether X is a zero forcing set. Since $X = S\cup\bigcup_{v\in Z} \bar{N}[v]$, it follows that $|X| = \sum_{v\in V}(\deg(v)z_v + s_v)$; thus, since X is a zero forcing set and the objective function is minimized, the optimal value is equal to $Z(G)$.

Now let X be a zero forcing set of G, let $Z = \{v \in X : |N(v) \cap X| = \deg(v) - 1\}$, and let $S = X \setminus \cup_{v\in Z} cl(N[v])$. Let s_v be TRUE for $v \in S$ and z_v be TRUE for $v \in Z$. By construction of Z and S, for each $v \in Z$, if $w \in cl(N[v])$ then $w \notin S$; equivalently, given a pair of vertices v and w with $w \in cl(N[v])$, either $v \notin Z$ or $w \notin S$ (or both). Thus, Clause (4) is satisfied. Clause (2) is satisfied because for each fort B, since X is a zero forcing set, either B contains some vertex in S or a vertex in $cl(N[v])$ for some v in Z. Finally, since there must exist a vertex in X with all-but-one neighbors in X, Z is non-empty and thus Clause (3) is satisfied. □

Model 3 *Boolean SAT model for zero forcing with neighborhood variables (Variant 2)*

$$\min \quad \sum_{v\in V} s_v$$

$$\text{s.t.:} \quad \underbrace{\bigwedge_{B\in\mathcal{B}}\left(\bigvee_{v\in B} s_v\right)}_{(5)} \wedge \underbrace{\left(\bigvee_{v\in V} z_v\right)}_{(6)} \wedge \underbrace{\left(\bigwedge_{\substack{v\in V\\ w\in \bar{N}[v]}} (\neg z_v \vee s_w)\right)}_{(7)} \wedge \underbrace{\left(\bigwedge_{\substack{v\in V\\ w\in cl(N[v])\setminus \bar{N}[v]}} (\neg s_w \vee \neg z_v)\right)}_{(8)}$$

PROPOSITION 3. *The optimal value of Model 3 is $Z(G)$, and the optimal solution is a minimum zero forcing set.*

Proof: Let s, z be a feasible solution to Model 3. Let S be the set of vertices such that s_v is assigned *True* and let Z be the set of vertices such that z_v is assigned *True*. By Clauses (6), (7), and (8) there is at least one vertex v in S such that all-but-one neighbors of v are also in S. If S is not a zero forcing set, then $B = V \setminus cl(S)$ is a fort. Since $B \in \mathcal{B}$, Clause (5) is not satisfied, contradicting the feasibility of s, z. Therefore, S is a zero forcing set. Since the objective function minimizes $|S|$, the optimal value is $Z(G)$.

Now let X be a zero forcing set of G, let $Z \subseteq X$ be the set of vertices with $|N(v) \cap X| = \deg(v) - 1$, and let $S = (X \cup \bigcup_{v \in Z} \bar{N}[v]) \setminus \bigcup_{v \in Z} cl(\bar{N}[v])$. Let s_v be TRUE for $v \in S$ and z_v be TRUE for $v \in Z$. Then by construction of S and Z, Clauses (7) and (8) are satisfied. Moreover, since the process of constructing S removes unnecessary vertices in the closure of the neighborhood variables, it follows that S is a zero forcing set. Since a zero forcing set intersects all forts, Clause (5) is satisfied. Finally, since a zero forcing set has at least one vertex such that all-but-one of its neighbors are also in the zero forcing set, Clause (6) is satisfied. □

To solve Models 1, 2, or 3 directly, one must obtain the entire set $\mathcal{B}$; since a graph can have exponentially many forts (cf. (Boyer et al. 2019)), this approach can be prohibitively expensive. Instead, the models can be solved by constraint generation, i.e., by iteratively generating forts and re-solving with an updated set of constraints. For example, to solve Model 1 with constraint generation, we proceed as follows. In the first iteration, the Reduced Master Problem (RMP) is Model 1 with all the fort constraints (1) omitted. The RMP is solved, and a violated constraint is found using one of the methods outlined below (e.g., Model 4). This violated constraint is added to the RMP and the RMP plus this constraint is solved again. A new violated constraint is found, and the RMP plus this new constraint, plus the first violated constraint, is solved again. This is repeated until no more violated constraints can be found. This process is guaranteed to terminate, because in each iteration, a constraint of type (1) is added; if we add all constraints of type (1), then we will simply have the original Model 1. However, the hope in using constraint generation is that certain constraints will subsume others, and if violated constraints are picked in a clever way, there will be no more violated constraints after relatively few iterations. See Example 1 for an illustration of this concept.

EXAMPLE 1. Let G be a path on five vertices labeled $1,2,3,4,5$ in order along the path. G has five forts: $\{1,3,5\},\{1,2,3,5\},\{1,2,4,5\},\{1,3,4,5\},\{1,2,3,4,5\}$. In the first iteration, the RMP is $\min s_1+s_2+s_3+s_4+s_5$ and its optimal solution is $s=[0,0,0,0,0]$; a violated fort with respect to s is $\{1,3,5\}$. In the next iteration, the optimal solution to the RMP plus the constraint $s_1 \vee s_3 \vee s_5$ is $s=[1,0,0,0,0]$ and there are no more violated forts (since the set of vertices corresponding to this solution intersects all forts of the graph). Thus, instead of solving a Boolean SAT problem with five constraints, we solved a Boolean SAT problem with zero constraints and a Boolean SAT problem with one constraint.

Below we give three methods for generating fort constraints violated by the set of vertices S corresponding to a solution of the current iteration of the RMP (or showing that there are no violated forts). The first method generates a minimum fort of G violated by S; it is analogous to the IP Model 3 from Brimkov et al. (2019a). In Model 4, Boolean variable x_v is TRUE if vertex v is in a fort violated by S.

Model 4 *Boolean SAT model for finding minimum violated forts*

$$\begin{aligned} \min \quad & \sum_{v\in V} x_v \\ \text{s.t.:} \quad & \underbrace{\bigwedge_{\substack{v\in V \\ w\in N(v)}} \left(x_w \vee \neg x_v \vee \bigvee_{a\in N(w)\setminus\{v\}} x_a \right)}_{(9)} \wedge \underbrace{\left(\bigvee_{v\in V} x_v \right)}_{(10)} \wedge \underbrace{\left(\bigwedge_{v\in cl(S)} \neg x_v \right)}_{(11)} \end{aligned}$$

PROPOSITION 4. *Let S be the set of vertices corresponding to an optimal solution of a RMP of Models 1, 2, or 3. Then, the optimum of Model 4 is the minimum size of a fort violated by S.*

Proof: Let x be a feasible solution of Model 4 and let B be the set of vertices such that x_v is TRUE. Clause (10) ensures that B is non-empty, while Clause (9) ensures that every vertex w which is a neighbor of $v \in B$ must either be in B or have at least one other neighbor in B. Therefore, B is a fort. Moreover, Clause (11) ensures that $B \cap cl(S) = \emptyset$, i.e., that B is a fort violated by S.

Conversely, let B be a fort of G such that $B \cap cl(S) = \emptyset$, and let x_v be TRUE for $v \in B$ and FALSE otherwise. By definition, B is non-empty and thus Clause (10) is satisfied. By assumption $B \cap cl(S) = \emptyset$ and therefore x_v is FALSE for $v \in V \setminus cl(S)$, so Clause (11) is

satisfied. Clause (9) is satisfied by default for all $v \in V \setminus B$, so consider $v \in B$. By definition of a fort, any vertex $w \in N(v)$ must either be in B or must have a neighbor $a \in N(w) \setminus \{v\}$ in B; thus, Clause (9) is satisfied. □

Instead of finding minimum violated forts with Model 4, we can find *minimal* violated forts with Algorithm 1. To this end, given a set of vertices S corresponding to a solution of the RMP, we first find a maximal set $C \supset S$ which is not zero forcing; then, $V \setminus C$ is a minimal fort that can be added as a constraint. The set C can be found by starting with $cl(S)$ and repeatedly adding a vertex which does not form a zero forcing set. The same greedy approach was used to generate minimal violated forts in (Brimkov et al. 2019a).

Algorithm 1 Greedy algorithm for finding minimal violated forts

Input: Graph $G = (V, E)$, set $S \subset V$ corresponding to solution of RMP
Output: Minimal fort B which violates S

1: $C \leftarrow cl(S)$
2: **for** $v \in V \setminus C$ **do**
3: **if** $cl(C \cup \{v\}) \neq V$ **then**
4: $C \leftarrow C \cup \{v\}$
5: **goto** line 3
6: **end if**
7: **end for**
8: **return** $B \leftarrow V \setminus C$

A third method for generating violated constraints entails finding forts that have a minimum border (but not necessarily minimum size). This is motivated by the following reasoning: since a zero forcing set must intersect every fort of the graph, the zero forcing number is bounded below by the cardinality of any set of pairwise disjoint forts; generating forts with a small border with respect to the current solution could make it more likely that a large set of pairwise disjoint forts is generated. In Model 5, the Boolean variable x_v (respectively, b_v) is TRUE if vertex v is in a fort (respectively, the border of a fort) violated by the solution of an RMP. This model is similar to IP Model 6 from Brimkov et al. (2019a), although the constraints ensuring that the b_v variables correctly indicate whether v is on the border of the fort are encoded in a different way.

Model 5 *Boolean SAT Model for finding violated forts with minimum border*

$$\min \quad \sum_{v \in V} b_v$$

$$\text{s.t.:} \underbrace{\bigwedge_{\substack{v\in V\\ w\in N(v)}} \left(x_w \vee \neg x_v \vee \bigvee_{a\in N(w)\setminus\{v\}} x_a \right)}_{(12)} \wedge \underbrace{\left(\bigvee_{v\in V} x_v \right)}_{(13)} \wedge$$

$$\underbrace{\left(\bigwedge_{\substack{v\in V\\ w\in N(v)}} (\neg x_v \vee b_v \vee x_w) \right)}_{(14)} \wedge \underbrace{\left(\bigwedge_{v\in V} \left(\neg b_v \vee \bigvee_{w\in N(v)} \neg x_w \right) \right)}_{(15)} \wedge \underbrace{\left(\bigwedge_{v\in cl(S)} \neg x_v \right)}_{(16)}$$

PROPOSITION 5. *Let S be the set of vertices corresponding to an optimal solution of a RMP of Models 1, 2, or 3. Then, the optimum of Model 5 is the minimum border size of a fort violated by S.*

Proof: Let b, x be a feasible solution of Model 5, let B be the set of vertices such that x_v is TRUE, and let C be the set of vertices such that b_v is TRUE. Clauses (12), (13), and (16) are equivalent to Clauses (9), (10), and (11) of Model 4; thus, as in Proposition 4, Clauses (12), (13), and (16) imply that B is a fort. It only needs to be shown that C is a set of border vertices. By Clause (14), if $v \in B$ and $\exists w \in N(v)$ such that $w \in V \setminus B$, then b_v is TRUE. Further, by Clause (15), if $v \in C$, there must exist some $w \in N(v)$ such that $w \notin B$. Therefore, by definition of border, C is a set of border vertices.

Conversely, let B be a fort such that $B \cap cl(S) = \emptyset$ and let $C = \{v \in B : \exists u \in N(v) \text{ s.t. } u \in V \setminus B\}$. Let x_v be TRUE for $v \in B$ and b_v be TRUE for $v \in C$. As in Proposition 4, Clauses (12), (13), and (16) are satisfied. For $v \in C$ and $v \notin B$, Clause (14) is automatically satisfied. For $v \notin C$ and $v \in B$, by the definition of border fort, $N(v) \subset B$ and thus Clause (14) is satisfied. Further, by the definition of border fort, for $v \in C$ there exists a vertex $w \in N(v)$ such that $w \in V \setminus B$ and therefore Clause (15) is satisfied. □

There is a trade-off between the time spent on finding violated fort constraints and the number of constraints that have to be found. The minimum forts given by Model 4 are the best possible violated constraints that can be found, in the sense that they subsume the largest number of other fort constraints. However, since finding a minimum size fort of a graph is NP-Complete in general (following from (Shitov 2017), since a maximum size set that is not zero forcing is the complement of a minimum fort), this method is computationally expensive. On the other hand, Algorithm 1 finds minimal forts in polynomial time; however, usually a larger number of minimal forts have to be generated

before there are no more violated constraints. The complexity of finding violated forts with a minimum border is an open problem. The computational experiments in Section 5 identify the combination of models and constraint generation methods that achieves the best value in the trade-off between number of constraints generated and generation time per constraint.

3. Heuristics for zero forcing

In this section, we give a heuristic for the zero forcing problem based on iteratively adding sets of vertices that maximize the closure of the currently blue vertices. We give several versions of this heuristic, based on how the sets of vertices are selected. These heuristics can be used to speed up the exact models presented in Section 2.

In Heuristic 1, $\widehat{Z}$ is the set of vertices currently selected to be in the zero forcing set of a graph $G = (V, E)$; $\widehat{Z}$ is initialized as the empty set, and grows until it becomes a zero forcing set. The set C is the closure of $\widehat{Z}$, and A is the set of vertices that is added to $\widehat{Z}$. In each iteration of Heuristic 1, A is recomputed and added to $\widehat{Z}$ until all vertices in V belong to C (i.e., until $\widehat{Z}$ becomes a zero forcing set). The set A maximizes a function f over certain sets A' of currently white vertices. Finally, a minimal subset of $\widehat{Z}$ is found using a greedy algorithm. The sets A' and the function f can be chosen as follows:

- **Single vertex largest closure**: Given a vertex v and sets of vertices C and Q, define $A'(v, C) = \{v\}$ and $f(Q, v, C) = |Q|$. In this version of the heuristic, a single white vertex v is added to $\widehat{Z}$, so that the closure of the resulting set of blue vertices is maximized.
- **Neighborhood largest closure**: Given a vertex v and sets of vertices C and Q, define $A'(v, C) = (N[v] \cap (V \backslash C)) \setminus \{u\}$ for some $u \in N(v) \cap (V \backslash C)$ and $f(Q, v, C) = |Q|$. In this version of the heuristic, a single white vertex v and all-but-one of its white neighbors are added to $\widehat{Z}$, so that the closure of the resulting set of blue vertices is maximized. Note that it does not matter which all-but-one neighbors of v are chosen, since v always forces its remaining white neighbor.
- **Neighborhood scaled closure**: Given a vertex v and sets of vertices C and Q, define $A'(v, C) = (N[v] \cap (V \backslash C)) \setminus \{u\}$ for some $u \in N(v) \cap (V \backslash C)$ and $f(Q, v, C) = (|cl(Q)| - |C|)/|A'(v, C)|$. In this version of the heuristic, a single white vertex v and all-but-one of its white neighbors are added to $\widehat{Z}$, so that the *ratio* of the closure of the resulting set of blue vertices to the number of vertices added is maximized. Again, it does not matter which all-but-one neighbors of v are chosen.

Heuristic 1 Closure-based heuristic for zero forcing

Input: Graph $G = (V, E)$
Output: Zero forcing set of G

1: $\widehat{Z} \leftarrow \emptyset$
2: $C \leftarrow \emptyset$
3: **while** $C \neq V$ **do**
4: $\quad C^* \leftarrow \emptyset$
5: $\quad A \leftarrow \emptyset$
6: $\quad$ **for** $v \in V \backslash C$ **do**
7: $\quad\quad S \leftarrow C \cup A'(v, C)$
8: $\quad\quad$ **if** $f(cl(S), v, C) > f(C^*, v, C)$ **then**
9: $\quad\quad\quad A \leftarrow A'(v, C)$
10: $\quad\quad\quad C^* \leftarrow cl(S)$
11: $\quad\quad$ **end if**
12: $\quad$ **end for**
13: $\quad C \leftarrow C^*$
14: $\quad \widehat{Z} \leftarrow \widehat{Z} \cup A$
15: $\quad$ **for** $v \in \widehat{Z}$ **do**
16: $\quad\quad$ **if** $cl(\widehat{Z} \setminus \{v\}) = V$ **then**
17: $\quad\quad\quad \widehat{Z} \leftarrow \widehat{Z} \setminus \{v\}$
18: $\quad\quad$ **end if**
19: $\quad$ **end for**
20: **end while**
21: **return** $\widehat{Z}$

Heuristic 1 always returns a zero forcing set, because $\widehat{Z}$ is grown in each iteration until the closure of $\widehat{Z}$ equals the whole vertex set of the graph. Since the closure C of the currently blue vertices increases by at least one vertex in each iteration, there are at most n iterations of the loop on line 3 for a graph G of order n. If set intersections are computed in a naive way, the overall complexity of Heuristic 1 is $O(n^3)$. Computational results for Heuristic 1 on various types of graphs are reported in Section 5. Heuristic 1 can also be modified to use backtracking in order to potentially choose better sets of vertices by which to grow the closure. However, preliminary computational experiments showed that backtracking did not offer consistent benefits in performance.

4. Facets and mandatory vertices

In this section, we study certain sets of vertices which must be included or excluded from minimum zero forcing sets. We also characterize when certain forts are facets of the zero forcing polytope, and show that the number of these facets can be exponential. We use this information to reduce the number of clauses in the Boolean SAT models from

Section 2. Studying the polyhedral structure of the zero forcing polytope could also be used to improve integer programming methods for zero forcing, and may be of independent theoretical interest. Similar investigations for other problems have been carried out, e.g., in (Balas 1975, Balas and Saltzman 1989, Fischetti 1991, Grötschel and Wakabayashi 1990). Information about the polyhedral structure of an instance could also potentially contribute to deciding whether the instance can be more efficiently solved by a SAT or IP based approach (e.g., akin to the work of Li et al. (2004) and Wolfman and Weld (2001)).

A *junction vertex* of a graph G is a vertex with degree at least 3. P is a *pendant path of a junction vertex* v if $G[P]$ is a path component of $G-v$ and one of the endpoints of P is adjacent to v in G. The neighbor of v in P will be called the *base* of P and the other endpoint of P will be called the *terminal*. Similarly, C is a *pendant cycle of a junction vertex* v if $G[C]$ is a path component of $G-v$ and both endpoints of C are adjacent to v in G. One of the neighbors of v in C will be called the *base* of C, and the other will be called the *terminal*. Finally, J is a *join path of junction vertices* u, v if $G[J]$ is a path component of $G-\{u,v\}$ with one endpoint adjacent to u and the other adjacent to v in G. The two endpoints of $G[J]$ will be called *terminals*. See Figure 1 for an illustration. With a slight abuse of notation, by saying "pendant path P", we may refer to either the set of vertices of the path or the path itself (and similarly for a pendant cycle and a join path).

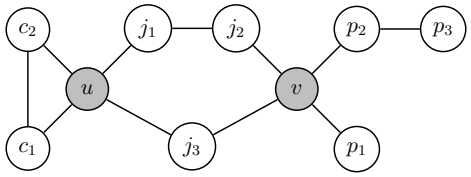

Figure 1 **A graph with junction vertices u and v, pendant paths $\{p_1\}$ and $\{p_2, p_3\}$, pendant cycle $\{c_1, c_2\}$, and join paths $\{j_1, j_2\}$ and $\{j_3\}$.**

Definition 1. Let $G=(V,E)$ be a graph and let $v, v' \in V$. $\mathcal{P}(v)$ denotes the set of pendant paths of a junction vertex v, $\mathcal{C}(v)$ denotes the set of pendant cycles of a junction vertex v, and $\mathcal{J}(v,v')$ denotes the set of join paths of junction vertices v and v'. Moreover,

1) Let $\mathcal{F}(\mathcal{P}(v))$ be the set of minimal forts of G such that $\forall F \in \mathcal{F}(\mathcal{P}(v))$, $v \notin F$, $F \subseteq S_i \cup S_j$ for some $S_i, S_j \in \mathcal{P}(v)$, and $\forall S_k \neq S_i, S_j, F \not\subseteq S_k$.

2) Let $\mathcal{F}(\mathcal{C}(v))$ be the set of minimal forts of G such that $\forall F \in \mathcal{F}(\mathcal{C}(v))$, $v \notin F$, $F \subseteq S_i$ for some $S_i \in \mathcal{C}(v)$, and $\forall S_k \neq S_i, F \not\subseteq S_k$.

3) Let $\mathcal{F}(\mathcal{J}(v,v'))$ be the set of minimal forts of G such that $\forall F \in \mathcal{F}(\mathcal{J}(v,v'))$, $v,v' \notin F$, $F \subseteq S_i \cup S_j$ for some $S_i, S_j \in \mathcal{J}(v,v')$, and $\forall S_k \neq S_i, S_j, F \not\subseteq S_k$.

We will now characterize the facet defining forts contained in the sets in Definition 1. Recall that the *Padovan Sequence* is the integer sequence with initial values $\rho(-1) = \rho(0) = \rho(1) = 1$ and recurrence relation $\rho(n) = \rho(n-2) + \rho(n-3)$. The proof of Theorem 1 is in the Online Supplement.

THEOREM 1. *Let $G = (V,E)$ be a graph and v, v' be junction vertices of G. Let $T_P = \{S \in \mathcal{P}(v) : |S| = 1\}$ and $T_J = \{S \in \mathcal{J}(v,v') : |S| = 1\}$. Then,*

1) The number of facet defining forts in $\mathcal{F}(\mathcal{P}(v))$ is

$$\begin{cases} \frac{1}{2} \sum\limits_{\substack{S_i, S_j \in \mathcal{P}(v) \\ S_i \neq S_j}} \rho(|S_i|-2)\rho(|S_j|-2) & \text{if} \quad |T_P| = 0 \\ \sum\limits_{S_i \in \mathcal{P}(v)} \rho(|S_i|-2) - 1 & \text{if} \quad |T_P| = 1 \\ 1 & \text{if} \quad |T_P| = 2 \\ 0 & \text{if} \quad |T_P| \geq 3. \end{cases} \tag{17}$$

2) All forts in $\mathcal{F}(\mathcal{C}(v))$ are facet defining.

3) The number of facet defining forts in $\mathcal{F}(\mathcal{J}(v,v'))$ is given by replacing "$\mathcal{P}(v)$" with "$\mathcal{J}(v,v')$" and "T_P" with "T_J" in (17).

The following example shows a family of graphs whose zero forcing polytopes have an exponential number of facets corresponding to pendant paths, pendant cycles, and join cycles.

EXAMPLE 2. Let G be a graph with two junction vertices u and v, two pendant paths of u, one of size 2 and the other of size $k := \frac{n-6}{3}$, one pendant cycle of v of size k, and two join paths of u and v, one of size 2 and the other of size k. Then, by Theorem 1, the zero forcing polytope of G has $\rho(k-2)$ facet defining forts in each of $\mathcal{F}(\mathcal{P}(u))$, $\mathcal{F}(\mathcal{C}(v))$, and $\mathcal{F}(\mathcal{J}(u,v))$. Since the ratio of successive terms in the Padovan sequence approaches the *plastic number* which has a value approximately 1.324718, G has $\Omega(1.3^{n/3})$ facets corresponding to forts in $\mathcal{F}(\mathcal{P}(u))$, $\mathcal{F}(\mathcal{C}(v))$, and $\mathcal{F}(\mathcal{J}(u,v))$.

We now identify certain sets of vertices which must be present or not present in some minimum zero forcing set. We also show that in a sense, these sets of vertices are the best

possible that can be found for the corresponding structures. Given two junction vertices u, v, let $T_J(u,v)$ denote the set of join paths of u and v of size 1, i.e., $T_J(u,v) = \{S \in \mathcal{J}(u,v) : |S| = 1\}$. Further, let $p(v)$, $c(v)$, and $j(u,v)$ respectively denote the set of terminals in $\mathcal{P}(v)$, $\mathcal{C}(v)$, and $\mathcal{J}(u,v)$. Let $\overline{p}(v)$ denote a set of all-but-one elements of $p(v)$ and $\overline{T}_J(u,v)$ denote a set of all-but-one elements of $T_J(u,v)$.

THEOREM 2. *Let $G = (V,E)$ be a graph. There exists a minimum zero forcing set Z of G such that for all junction vertices u, v of G,*

1) If $|\mathcal{P}(v)| \geq 2$ then $\{v\} \cup \mathcal{P}(v) \backslash \overline{p}(v) \subset V \backslash Z$ and $\overline{p}(v) \subset Z$

2) If $|\mathcal{P}(v)| = 1$ then $\{v\} \cup \mathcal{P}(v) \backslash p(v) \subset V \backslash Z$

3) $c(v) \subset Z$ and $\mathcal{C}(v) \backslash c(v) \subset V \backslash Z$

4) If $|T_J(u,v)| \geq 2$ and $\mathcal{J}(u,v) = T_J(u,v)$ then $\overline{T}_J(u,v) \subset Z$

5) If $|T_J(u,v)| \geq 1$ and $\mathcal{J}(u,v) \neq T_J(u,v)$ then $T_J(u,v) \subset Z$

6) $\{v \in S : S \in \mathcal{J}(u,v) : |S| \geq 2\} \backslash j(u,v) \subset V \backslash Z$.

Moreover, the left-hand-side sets in each inclusion cannot consistently be increased by adding more vertices from the corresponding collections of pendant paths, pendant cycles, and join paths.

Proof: 1) Each pair of pendant paths $S_i \cup S_j$ in $\mathcal{P}(v)$ forms a fort. Thus, every zero forcing set must intersect each pair of pendant paths in $\mathcal{P}(v)$, and hence all or all-but-one pendant paths must intersect every zero forcing set. If the terminal of $S \in \mathcal{P}(v)$ is in a zero forcing set Z, then no other vertices of S need to be in Z, since the terminal of S is sufficient to force S. Moreover, since $|\mathcal{P}(v)| \geq 2$, Z contains at least one terminal, which will force v. Thus, there exists a zero forcing set which contains $\overline{p}(v)$ and does not contain v or any other vertices of $\mathcal{P}(v)$. If G is a star with center v, then every minimum zero forcing set Z of G satisfies $\{v\} \cup \mathcal{P}(v) \backslash \overline{p}(v) = V \backslash Z$ and $\overline{p}(v) = Z$, and hence no vertex can be added to either inclusion.

2) The entire pendant path S in $\mathcal{P}(v)$, along with v, can be forced by its terminal. Thus, there exists a zero forcing set Z which does not contain v or any non-terminal vertex of S. If G is the graph in Figure 2, top left, then a minimum zero forcing set Z is $\{p(u_1), p(u_2)\}$ (or symmetrically $\{p(v_1), p(v_2)\}$), and neither $p(v_1)$ nor $p(v_2)$ can be added to $V \backslash Z$.

3) A pendant cycle S is a fort, so at least one vertex of each $S \in \mathcal{C}(v)$ must be in every zero forcing set. Since v may be forced from outside S, if the terminal of each $S \in \mathcal{C}(v)$

is in a zero forcing set Z, then the entire pendant cycle S will get forced once v becomes blue. Moreover, no other vertices of S need to be in Z, since v and the terminal of S are sufficient to force S. If G is the graph in Figure 2, top right, then $Z = \{v\} \cup c(v)$ is a minimum zero forcing set which satisfies $\mathcal{C}(v) \backslash c(v) = V \backslash Z$.

4) and 5) In both cases $|\mathcal{J}(u,v)| \geq 2$, so each pair of join paths $S_i \cup S_j$ in $T_J(u,v)$ forms a fort. Thus, every zero forcing set must intersect each pair of join paths in $T_J(u,v)$, and hence all or all-but-one join paths must intersect every zero forcing set. If $\mathcal{J}(u,v) = T_J(u,v)$, then there exists a zero forcing set Z which contains $\overline{T}_J(u,v)$. If $\mathcal{J}(u,v) \neq T_J(u,v)$, then there exists a zero forcing set which contains $T_J(u,v)$ (and either intersects all or all-but-one join paths in $\mathcal{J}(u,v) \backslash T_J(u,v)$). In 4), if G is the graph in Figure 2, bottom left, every minimum zero forcing set of G is symmetric to the one shown with shaded vertices (where any vertex within the dashed triangle can be selected). In every minimum zero forcing set Z of G, there is a pair of junction vertices u, v such that all join paths between them cannot be in Z. Thus, the inclusion $\overline{T}_J(u,v) \subset Z$ cannot be increased with another vertex in $T_J(u,v)$. In 5), consider the graph G obtained by subdividing each vertex in the dashed triangle of the graph in Figure 2, bottom left. There is a minimum zero forcing set that includes all join paths of size 1, along with a degree 4 vertex and one of its degree 2 neighbors in a join path of size 2. Moreover, every minimum zero forcing set of G contains a pair of junction vertices u, v such that some join path of u, v does not intersect Z. Thus, the inclusion $T_J(u,v) \subset Z$ cannot in general be increased with another vertex from $\mathcal{J}(u,v)$.

6) Any $S \in \mathcal{J}(u,v)$ with $|S| \geq 2$ can be forced by u and a terminal of S adjacent to u or by v and a terminal of S adjacent to v. Thus, there exists a zero forcing set Z which does not contain any of the internal vertices of join paths of size at least 2. While one of the terminals of each such join path can also be excluded, it is not always possible to exclude an arbitrary terminal. For example, if G is the graph in Figure 2, bottom right, then every minimum zero forcing set excludes the two vertices in the dashed oval and includes exactly one of the white vertices outside the dashed oval. Thus, the inclusion $\{v \in S : S \in \mathcal{J}(u,v) : |S| \geq 2\} \backslash j(u,v) \subset V \backslash Z$ cannot in general be increased with an arbitrary element of $j(u,v)$. $\square$

While Theorem 1 showed that the presence of pendant paths, pendant cycles, and join paths leads to a large number of facet defining inequalities, Theorem 2 indicates that the existence of such subgraphs can actually improve our ability to find a minimum zero forcing

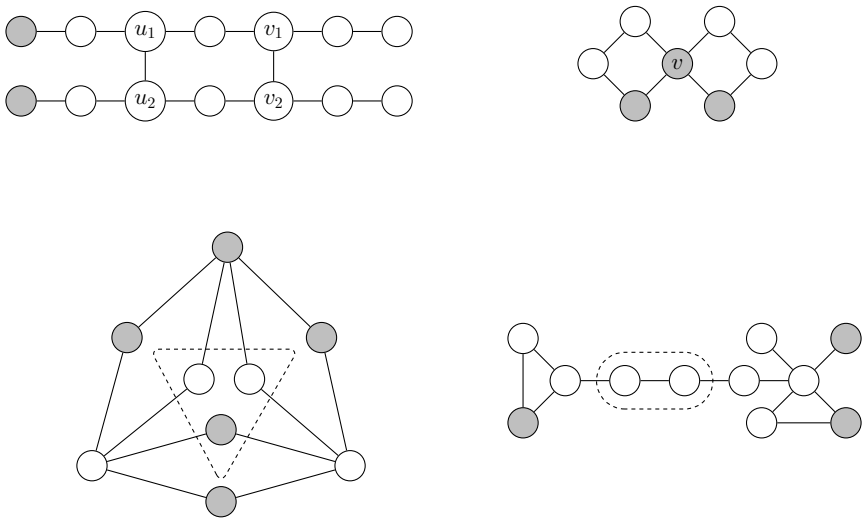

Figure 2 Graphs showing tightness of inclusions in Theorem 2.

set, because including or excluding certain subsets of these vertices can significantly reduce the number of clauses to be satisfied in the models presented in Section 2.

5. Computational results

This section presents implementation details and computational results for the algorithms and heuristics discussed in Sections 2 and 3. Our computational results were obtained on an AMD Ryzen 7 2700X 3.7GHz 8-Core processor, using 16 GB of RAM, and Ubuntu version 18.04.2 LTS. The Boolean satisfiability models were solved using Glucose version 4.1 (Audemard and Simon 2009), utilizing the core solver. Integer programs were solved with an Intel Xeon E3-1270 v5 3.60GHz processor, 16 GB of RAM, and Red Hat Linux version 4.8.5-11 using Gurobi version 7.5.2 (Gurobi Optimization, LLC 2017) set to use a single thread. The Wavefront algorithm was implemented in C++ and compiled with g++ version 4.8.5 while the methods and heuristics supporting the SAT models were written in C++ and compiled with g++ version 7.3.0. Although the Boolean satisfiablity models and the integer programming models were solved on different machines, the single core performance of the two processors differs by only 1% (PassMark Software 2019), and the available RAM was identical; thus, the runtimes obtained by the two machines should be comparable.

The different models and heuristics were tested on several standard benchmark graphs, including DIMACS10 graphs (Bader et al. 2014) (adjnoun, celegansneural, chesapeake, dolphins, football, jazz, karate, lesmis, and polbooks), IEEE graphs (ICSEG 2018) (14-Bus, 24-Bus, 30-Bus, 39-Bus, 57-Bus, 118-Bus, 300-Bus, and 96-RTS), random cubic

graphs, random average-cubic graphs, WS$(5, 0.3)$ graphs, and WS$(10, 0.3)$ graphs. The random cubic and Watts-Strogatz graphs instances were identical to the ones in (Brimkov et al. 2019a) to allow fair comparison. Random cubic graphs, WS$(5, 0.3)$ graphs, and WS$(10, 0.3)$ graphs include five instances with $10i$, $1 \leq i \leq 10$ vertices; random average-cubic graphs include five instances with $20i$, $1 \leq i \leq 6$ vertices. All instances are available at `https://github.com/derekmikesell/improved_zero_forcing`. The algorithms were set to timeout after two hours for each instance.

5.1. Implementation Details

To compute the zero forcing number of a graph $G = (V, E)$, we first obtain a set of vertices which can be included or excluded from a minimum zero forcing set, then seed a Boolean satisfiability model with a heuristic solution based on those vertices, and finally solve the model to optimality. More precisely, we first construct sets S_{inc} and S_{exc} respectively consisting of all vertices included in Z and $V \backslash Z$ in Theorem 2. Next, we take the best heuristic solution from the three versions of Heuristic 1 (heuristic runtimes are small compared to the runtimes of the exact models), while incorporating the sets S_{inc} and S_{exc} by replacing line 1 with "$Z \leftarrow S_{inc}$", line 2 with "$C \leftarrow cl(Z)$", and line 6 with "**for** $v \in V \setminus (C \cup S_{exc})$ **do**". We then build Model 1, 2, or 3 and add the clauses $\left(\bigwedge_{v \in S_{inc}} s_v\right)$ and $\left(\bigwedge_{v \in S_{exc}} \neg s_v\right)$ to account for the vertices included or excluded from a minimum zero forcing set. These clauses propagate through and reduce the encoding of the model, which decreases overall computation time. The size of the zero forcing set produced by the heuristic is used as an upper bound on the objective function; decreasing the size of the encoding of the objective function also provides significant improvements in runtime. In Models 2 and 3, we avoid potential conflicts of $\bar{N}[v] := N[v] \setminus \{u\}$ with vertices in S_{exc} by choosing $u \in N(v) \cap S_{exc}$ if $N(v) \cap S_{exc} \neq \emptyset$, and $u \in N(v) \setminus S_{inc}$ otherwise. We also construct sets Z_{inc} and Z_{exc} analogous to S_{inc} and S_{exc} to account for included and excluded vertices that have the neighborhood variable property satisfied by S_{inc} and S_{exc}. The corresponding clauses $\left(\bigwedge_{v \in Z_{inc}} z_v\right)$ and $\left(\bigwedge_{v \in Z_{exc}} \neg z_v\right)$ are added to Models 2 and 3, and the variables for vertices in S_{inc}, S_{exc}, Z_{inc}, and Z_{exc} are removed from the objective function, which significantly reduces the number of variables and clauses necessary in the encoding.

The models are solved by iteratively decrementing the objective function by the addition of a single clause; violated forts are found with Model 4, Algorithm 1, or Model 5, and added to the outer models (1, 2, or 3) which are repeatedly solved. This continues until the outer

model has a satisfiable assignment whose closure is all of V, or until the model verifies that there is no solution of size one smaller than the best heuristic value. We also initially add all forts of the form $S_{uv} \cup S_{vw} \cup S_{wu}$, where $S_{uv} \in \mathcal{J}(u,v)$, $S_{vw} \in \mathcal{J}(v,w)$ and $S_{wu} \in \mathcal{J}(w,u)$ such that $(S_{uv} \cup S_{vw} \cup S_{wu}) \cap S_{inc} = \emptyset$, for all $\{u,v,w\} \subset V$ such that $\mathcal{J}(u,v) \neq \emptyset$, $\mathcal{J}(v,w) \neq \emptyset$, and $\mathcal{J}(w,v) \neq \emptyset$. In the case of a timeout, the last satisfiable assignment to Model 1, 2, or 3 gives a lower bound on $Z(G)$. This assignment is expanded via the heuristic methods into a zero forcing set, and the smallest heuristic solution gives an upper bound on $Z(G)$.

5.2. Results and discussion

Tables 1 and 2 compare the exact SAT models with the Wavefront algorithm given in (Butler et al. 2014) and the best result among the integer programming (IP) models given in (Brimkov et al. 2019a). The IP models include the Infection model (I), Maximum Closure model (M), Maximum Closure with addition of facet inducing constraints (C), Extended Cover model (E), Fort Cover model (F), and Fort Cover model with addition of facet inducing constraints (D). The Fort Cover IP model is analogous to Model 1 from Section 2. For the SAT models, constraint generation via Model 4 generally resulted in fastest runtimes; hence, computational results obtained by generating forts with Algorithm 1 and Model 5 are omitted. All computational results are available at `https://github.com/derekmikesell/improved_zero_forcing`.

For DIMACS graphs, the SAT models were faster than Wavefront and the IP models in every instance that was solved to optimality. In some instances, the runtime of the SAT models was more than 700 times faster than the fastest previously available method. For instances that were not solved to optimality, the bounds produced by the SAT models were generally comparable with the bounds produced by the IP models, although the latter usually produced better lower bounds.

For IEEE graphs, the SAT models were again faster than Wavefront and the IP models in every instance that was solved to optimality. In some instances, the runtime of the SAT models was more than 70 times faster than the fastest previously available method. For the instance that was not solved to optimality, again the lower bound produced by the IP models was better, while the upper bounds were comparable. For both DIMACS and IEEE graphs, Models 1 and 3 had roughly similar runtimes, while Model 2 was somewhat slower. The non-unit coefficients in the objective function of Model 2 result in a larger encoding than Model 3, explaining the slower runtimes.

For cubic and Watts-Strogatz graphs, the Wavefront Algorithm was generally the fastest, except on small instances where the SAT models were fastest. This can possibly be explained by the fact that making a few new vertices blue often makes the closure of a set of vertices grow significantly; this means the Wavefront algorithm will store fewer closures and require fewer iterations. For these families of random graphs, Models 2 and 3 were faster than Model 1 and were able to solve larger graphs to optimality. In most instances, Models 2 and 3 also outperformed all of the IP models. Due to the denser structure of cubic and Watts-Strogatz graphs, the neighborhood variables in Models 2 and 3 provide a significant improvement over the encoding of Model 1, outweighing the increased encoding size.

In average-cubic graphs, the Wavefront algorithm performed very poorly. The SAT models were the fastest on the smaller and medium-sized instances, while the IP models performed better on the largest instances. It is worth noting that the Infection model is the only IP model not based on the hitting set problem and not solved through constraint generation. The SAT models outperformed all IP models other than the Infection model. The reason the hitting set based models performed worse than the Infection model on these graphs is that they are likely to have a large number of short induced paths, which means they have a larger number of forts that must be covered by a zero forcing set, and hence require more constraints to be generated before an optimal solution is found.

Table 3 gives information about different characteristics of the instances, including number of edges, pendent paths, pendent cycles, and join paths, and number of mandatory vertices added to the Models according to Theorem 2. Cubic graphs clearly have $\frac{3n}{2}$ edges and do not have any pendant paths, pendant cycles, or join paths, and hence no mandatory vertices were added. By construction, Watts-Strogatz graphs with parameters n, k and β have $\frac{nk}{2}$ edges. None of the Watts-Strogatz instances in our test set had any pendant paths or pendant cycles; the WS(10,0.3) graphs did not have any join paths, while the WS(5,0.3) graphs had between 0 and 6 join paths. However, most of these were too small to be usable by Theorem 2; only in one instance (one of the graphs on 40 vertices) a join path led to the addition of a single mandatory vertex. Thus, these instances (with the exception of the one on 40 vertices) are not included in Table 3. Table 3 also gives the runtimes of Models 1, 2, and 3 without any mandatory vertices added. Comparing these to the runtimes in Tables 1 and 2, we see that in most small instances, the addition of mandatory vertices did

G	$\|V\|$	$Z(G)$	Wavefront	IP Models	Model 1	Model 2	Model 3
karate	34	13	329.10	0.16 (M)	**0.02**	0.03	0.02
chesapeake	39	14	10.91	35.43 (M)	7.51	3.51	**1.94**
dolphins	62	14	2405.56	246.24 (D)	**108.59**	985.38	199.30
lesmis	77	40	T	2708.95 (D)	5.07	12.20	**3.74**
polbooks	105	N/A	T	{14/27} (D)	{19/29}	**{18/27}**	**{18/27}**
adjnoun	112	N/A	T	**{9/30}** (M)	{9/32}	{9/32}	{9/32}
football	115	N/A	T	**{9/38}** (M)	{0/40}	{0/40}	{0/40}
jazz	198	N/A	T	**{27/96}** (M)	{8/102}	{12/103}	{10/102}
celegansneural	297	N/A	T	**{28/100}** (M)	{14/90}	**{15/89}**	{13/89}
IEEE 14	14	4	0.01	0.01 (M)	**0.001**	0.002	**0.001**
IEEE 24	24	6	0.07	0.01 (M)	**0.004**	0.007	**0.004**
IEEE 30	30	7	0.82	0.03 (M)	**0.010**	0.021	0.012
IEEE 39	39	7	6.69	0.04 (M)	0.026	0.061	**0.019**
IEEE 57	57	9	18.76	1.97 (M)	0.409	0.789	**0.282**
RTS 96	73	15	T	0.78 (M)	0.931	6.563	**0.641**
IEEE 118	118	26	T	734.96 (I)	28.675	{23/26}	**10.367**
IEEE 300	300	N/A	T	**{73/75}** (I)	{62/77}	{50/78}	{61/77}

Table 1 Comparison of runtimes (in seconds) for different zero forcing algorithms. Runtimes are reported for the Wavefront algorithm from (Butler et al. 2014), the fastest IP solution from (Brimkov et al. 2019a), and Models 1, 2, and 3 from the present paper using the minimum violated fort model for constraint generation. A "N/A" in the $Z(G)$ column indicates that the exact zero forcing number of the graph is not known. 'T' indicates that none of the instances were solved within 2 hours or within the available amount of memory (16 GB). In instances that could not be solved by the IP and SAT methods, $\{\ell/u\}$ denotes the lower bound ℓ and upper bound u on $Z(G)$ at the point of timeout. Bold text indicates the best performance or best bound for each graph. The letters in the "IP Models" column indicate which IP model produced the reported result.

not have a measurable effect on the runtime, while in larger instances it had a considerable effect. In particular, in some cases adding mandatory vertices determined whether or not the problem could be solved at all within the time limit. Even in instances that were not solved by either trial within the time limit, adding mandatory vertices yielded better bounds.

Tables 4 and 5 compare the different variants of Heuristic 1 as standalone methods (without adding mandatory vertices using Theorem 2, or any other preprocessing). In all instances, the runtimes of the heuristics were very low compared to the runtimes of the exact models. The two Neighborhood variants were overall the fastest, because they usually add several vertices in each iteration and hence require fewer iterations to find a zero forcing set.

The Single Node Largest Closure variant generally gave the best bounds for DIMACS and IEEE graphs, while the Neighborhood Scaled Closure variant generally gave the best bounds for random cubic graphs and Watts-Strogatz graphs. In all instances for which the exact value of $Z(G)$ is known, at least one variant of the heuristic produced a zero

	$\|V\|$	$Z(G)$	Wavefront	IP Models	Model 1	Model 2	Model 3
Cubic graphs	10	3.8	0.01	0.01 (M)	0.001	0.000	**0.000**
	20	5.2	0.02	0.04 (M)	0.020	0.011	**0.009**
	30	6.6	**0.13**	0.42 (M)	0.308	0.142	0.146
	40	8.8	**1.95**	6.10 (E)	9.051	6.722	2.624
	50	9.2	**6.69**	30.90 (E)	42.471	24.371	17.527
	60	11.4	**156.83**	3138.83 (E)	{10.4/11.4}	[1] 1363.052	[1] 3610.884
	70	12.0	**370.50**	[3] 2852.27 (E)	{10.6/12.4}	{10.8/12.4}	{10.2/12.2}
	80	12.8*	[4] **1615.85**	[2] 5341.15 (E)	{10.4/13.4}	{10.8/13.8}	{10.4/13.4}
	90	13.0*	[2] **3101.29**	{9.4/13.6} (F)	{8.6/14.2}	{9.6/14.4}	{9.4/14.2}
	100	N/A	T	**{9.6/15.4}** (F)	{8.6/16.2}	{9.6/15.8}	{9.6/16}
WS(5,0.3)	10	4.4	0.01	0.01 (M)	0.001	0.000	**0.000**
	20	6.2	**0.01**	0.06 (M)	0.027	0.018	**0.013**
	30	7.0	**0.05**	0.89 (M)	0.495	0.227	0.199
	40	9.4	**0.91**	38.52 (E)	14.049	12.212	3.308
	50	10.8	**7.20**	687.45 (E)	829.446	405.851	301.819
	60	11.6	**47.55**	[3] 594.42 (E)	[2] 1421.286	[3] 1933.398	[3] 2668.168
	70	14.0	**474.31**	{10.2/14.0} (E)	{11/14.4}	{11/14.4}	{10.8/14.4}
	80	14.8	**1441.92**	{9.8/14.8} (E)	{10.6/16}	{10.6/16.4}	{10.6/16.4}
	90	16.0*	[1] **4537.77**	{8.6/16.8} (F)	{10.2/17.2}	{10.4/17.4}	{10.2/17.6}
	100	N/A	T	{7.2/19.0} (M)	{9.6/19.8}	**{10/19.2}**	{10/19.4}
WS(10,0.3)	20	12.0	**0.01**	0.97 (M)	0.364	0.038	0.026
	30	15.4	**0.08**	435.93 (F)	72.022	5.134	2.339
	40	18.0	**0.64**	{15.2/18.0} (F)	{16/18}	1247.623	268.791
	50	21.8	**6.44**	{13.0/21.8} (F)	{13/22}	{18/22}	{16.8/22}
	60	24.6	**45.00**	{10.0/24.6} (F)	{10.8/24.8}	{17.2/24.6}	{16.4/24.8}
	70	27.4	**287.17**	{9.6/27.8} (C)	{5.6/27.6}	{14/27.6}	{14/27.6}
	80	31.2	**2753.69**	{8.4/31.6} (C)	{1.2/32.2}	{9.2/32}	{9.2/32}
	90	N/A	T	{8.2/35.6} (C)	{1/35.2}	**{7.8/35}**	{7.8/34.8}
	100	N/A	T	{8.0/39.8} (C)	{1/38.6}	**{7.4/38.6}**	{7.4/39}
Avg-cubic	20	5.6	0.10	0.02 (M)	0.003	0.007	**0.003**
	40	10.0	796.89	0.63 (M)	0.065	0.399	**0.063**
	60	14.4	T	9.24 (I)	1.077	57.806	**0.867**
	80	19.4	T	15.45 (I)	11.708	[4] 28.320	**7.150**
	100	23.4	T	**37.23** (I)	345.232	{22.2/24.8}	1215.674
	120	30.5	T	**109.13** (I)	[2] 382.478	{27.2/31}	[2] 3613.361

Table 2 Comparison of runtimes (in seconds) for different zero forcing algorithms, averaged over five instances. Runtimes are reported for the Wavefront algorithm from (Butler et al. 2014), the fastest IP solution from (Brimkov et al. 2019a), and Models 1, 2, and 3 from the present paper using the minimum violated fort model for constraint generation. A "*" (resp., "N/A") in the $Z(G)$ column indicates that some (resp., all) of the zero forcing numbers of those five instances are not known, and the reported value is the average of the ones that are known. A number $[x]$ indicates that only x of the five instances of the specified size were solved; 'T' indicates that none of the instances were solved within 2 hours or within the available amount of memory (16 GB). In instances that could not be solved by the IP and SAT methods, $\{\ell/u\}$ denotes the lower bound ℓ and upper bound u on $Z(G)$ at the point of timeout, averaged over all instances. Bold text indicates the best performance or best bound for each set of instances. The letters in the "IP Models" column indicate which IP model produced the reported result.

G	$\lvert E\rvert$	$\mathcal{P}$	$\mathcal{C}$	$\mathcal{J}$	Mand.	Model 1	Model 2	Model 3
IEEE 14	20	1	0	5	1	0.001	0.005	0.001
IEEE 24	34	1	0	7	1	0.003	0.008	0.003
IEEE 30	41	3	1	8	5	0.014	0.042	0.015
IEEE 39	46	8	0	9	12	0.025	0.086	0.03
IEEE 57	78	1	0	16	1	0.439	2.282	0.506
RTS 96	108	2	0	23	2	1.269	37.19	1.328
IEEE 118	179	6	0	38	10	31.25	{22/27}	24.291
IEEE 300	409	49	1	54	92	{55/79}	{36/77}	{52/79}
karate	78	1	0	6	2	0.038	0.297	0.025
chesapeake	170	0	0	0	0	7.514	3.510	1.942
dolphins	159	7	0	5	11	3662.361	{14/17}	{14/16}
lesmis	254	7	1	8	19	42.495	6107.473	26.292
polbooks	441	0	0	1	0	{19/29}	{18/27}	{18/27}
adjnoun	425	9	0	12	11	{10/32}	{10/32}	{9/32}
football	613	0	0	0	0	{0/40}	{0/40}	{0/40}
jazz	2742	5	0	3	5	{8/102}	{14/102}	{13/103}
celegansneural	2192	3	0	4	17	{13/89}	{13/89}	{13/89}
WS(5,.3), $n=40$	100	0	0	1	1	47.063	35.749	7.480
Avg-cubic, $n=20$	30	4	0	3	7	0.004	0.015	0.004
Avg-cubic, $n=40$	60	11	0	5	17	0.719	66.874	0.575
Avg-cubic, $n=60$	90	15	0	7	28	56.541	[2] 2617.368	54.806
Avg-cubic, $n=80$	120	21	0	9	40	[1] 791.007	{15/21}	[1] 2444.069
Avg-cubic, $n=100$	150	25	0	12	48	{21/25}	{16/25}	{21/25}
Avg-cubic, $n=120$	180	30	0	13	60	{24/31}	{17/32}	{23/32}

Table 3 Characteristics of graph instances and the effects of not adding mandatory vertices on different models. The $|E|$ column gives the number of edges in the graph (from which density can be determined as $|E|/\binom{|V|}{2}$); the $\mathcal{P}$, $\mathcal{C}$, and $\mathcal{J}$ columns respectively give the number of pendant paths, pendant cycles, and join paths in the graph. The "Mand." column gives the number of mandatory vertices added. The Model 1, 2, and 3 columns give the runtime of each model without adding any mandatory vertices; this can be compared to the runtimes in Tables 1 and 2. For average-cubic graphs, the runtimes are averaged over five instances; a number $[x]$ indicates that only x of the five instances of the specified size were solved. For instances that could not be solved in two hours, $\{\ell/u\}$ denotes the obtained lower bound ℓ and upper bound u on $Z(G)$ at the point of timeout, averaged over all instances. Instances not included in this table did not have any mandatory vertices added.

forcing set whose size differs by at most 3 from $Z(G)$ (and for most instances, differs by at most 1 from $Z(G)$). This indicates that the difficulty in solving the zero forcing problem does not lie in arriving to a near-optimal solution. Furthermore, these heuristics may be valuable in the application of computing the minimum rank of a graph: since zero forcing is an approximation of minimum rank (or, more precisely, n minus the minimum rank), the heuristics give a very quick and often accurate approximation of the minimum rank.

6. Concluding remarks

In this paper, we introduced new methods for computing the zero forcing numbers of graphs based on Boolean satisfiability. Much effort has been put into developing closed formulas,

			Single Node Largest Closure		Neighborhood Largest Closure		Neighborhood Scaled Closure	
G	$\|V\|$	$Z(G)$	Z_h	time	Z_h	time	Z_h	time
karate	34	13	**13**	0.003	14	0.001	14	0.002
chesapeake	39	14	**15**	0.007	**15**	0.002	**15**	0.002
dolphins	62	14	**17**	0.015	19	0.003	**17**	0.004
lesmis	77	40	**41**	0.040	**41**	0.008	**41**	0.026
polbooks	105	{19/27}	**29**	0.057	37	0.010	34	0.020
adjnoun	112	{9/30}	**32**	0.072	36	0.009	43	0.017
football	115	{9/38}	44	0.129	41	0.017	**40**	0.017
jazz	198	{27/96}	**103**	1.468	116	0.109	**103**	0.451
celegansneural	297	{28/89}	**89**	2.241	130	0.178	101	1.292
IEEE 14	14	4	**4**	0.000	5	0.000	5	0.000
IEEE 24	24	6	7	0.001	**6**	0.000	**6**	0.000
IEEE 30	30	7	**7**	0.000	9	0.000	**7**	0.000
IEEE 39	39	7	9	0.003	**8**	0.001	**8**	0.001
IEEE 57	57	9	**10**	0.007	**10**	0.002	11	0.003
RTS 96	73	15	17	0.019	17	0.008	**16**	0.006
IEEE 118	118	26	**27**	0.048	32	0.017	28	0.029
IEEE 300	300	{73/75}	**79**	1.102	88	0.381	82	0.698

Table 4 Comparison of runtimes (in seconds) for different variants of Heuristic 1. Z_h denotes size of the zero forcing set found by each heuristic. The smallest set found for each graph is indicated in bold. For some graphs the exact value of $Z(G)$ is not known, and instead the best known lower and upper bounds on $Z(G)$ are reported.

characterizations, and bounds for the zero forcing numbers of graphs with special structure; however, there are still relatively few algorithms for computing the zero forcing numbers of arbitrary graphs, and these algorithms cannot handle very large graphs. Our computational experiments revealed that the methods proposed in this paper outperform the state-of-the-art algorithms for zero forcing on many standard benchmark graphs; in some cases, the runtime of our methods was several orders of magnitude faster than the previously best methods. One open problem related to the constraint generation framework proposed in Section 2 is the computational complexity of finding a fort with a minimum border. It would also be interesting to develop a SAT model using a timestep-based paradigm rather than a hitting set paradigm. The challenge in attempting this would be to creatively encode the integer variables as binary in order to manage the encoding size.

Next, we presented novel heuristics that can be used to speed up the exact SAT-based models, or as standalone approximations for $Z(G)$. Our computational experiments showed the heuristics were very fast, and produced zero forcing sets of near-minimum (and often minimum) size for all test instances. This leads to efficient, high quality approximations for the minimum rank problem, and is also valuable in other applications of zero forcing

			Single Node Largest Closure		Neighborhood Largest Closure		Neighborhood Scaled Closure	
G	$\|V\|$	$Z(G)$	Z_h	time	Z_h	time	Z_h	time
Cubic	10	3.8	**3.8**	0.000	**3.8**	0.000	**3.8**	0.000
	20	5.2	5.4	0.000	**5.2**	0.000	**5.2**	0.000
	30	6.6	7.4	0.001	7.2	0.000	**7.0**	0.000
	40	8.8	9.8	0.003	9.6	0.001	**9.2**	0.001
	50	9.2	10.4	0.005	**9.4**	0.001	9.6	0.001
	60	11.4	12.4	0.008	12.0	0.003	**11.8**	0.003
	70	12.0	**12.6**	0.008	13.2	0.004	12.8	0.004
	80	12.8*	15.4	0.014	15.0	0.005	**14.2**	0.005
	90	13.0*	16.6	0.022	15.6	0.007	**14.4**	0.007
	100	N/A	16.8	0.028	17.2	0.010	**16.4**	0.011
WS(5,.3)	10	4.4	4.8	0.000	**4.6**	0.000	**4.6**	0.000
	20	6.2	**6.2**	0.000	6.4	0.000	6.4	0.000
	30	7.0	8.0	0.001	**7.4**	0.000	7.6	0.000
	40	9.4	10.2	0.002	10.6	0.000	**9.6**	0.000
	50	10.8	**11.6**	0.005	12.2	0.002	**11.6**	0.002
	60	11.6	13.6	0.008	14.0	0.002	**13.2**	0.002
	70	14.0	15.0	0.016	16.2	0.003	**14.8**	0.004
	80	14.8	**16.8**	0.015	18.2	0.005	17.2	0.006
	90	16.0*	19.0	0.025	19.0	0.007	**17.6**	0.010
	100	N/A	21.0	0.036	**20.4**	0.010	**20.4**	0.014
WS(10,.3)	20	12.0	**12.2**	0.000	**12.2**	0.000	**12.2**	0.000
	30	15.4	16.2	0.002	16.8	0.000	**16.0**	0.000
	40	18.0	18.4	0.004	18.4	0.001	**18.2**	0.001
	50	21.8	**22.0**	0.009	24.0	0.002	23.6	0.002
	60	24.6	**24.8**	0.015	25.6	0.003	25.8	0.003
	70	27.4	28.8	0.025	**28.4**	0.005	**28.4**	0.004
	80	31.2	**32.6**	0.036	33.8	0.008	33.2	0.007
	90	N/A	**35.6**	0.052	38.4	0.011	36.8	0.012
	100	N/A	40.4	0.076	40.0	0.013	**39.2**	0.014

Table 5 Comparison of runtimes (in seconds) for different variants of Heuristic 1, averaged over five instances. $Z(G)$ denotes the zero forcing number, Z_h denotes the size of the zero forcing set found by each heuristic; both are averaged over five instances. The smallest set found for each graph is indicated in bold. A "*" (resp., "N/A") in the $Z(G)$ column indicates that some (resp., all) of the zero forcing numbers of those five instances are not known, and the reported value is the average of the ones that are known; in such cases, the averages may not be comparable with the sets found by the heuristic.

where an exact solution is not necessary. It would be a problem of interest to derive certain guarantees on the quality of heuristic solutions, and to design heuristics for quickly finding forts of near-minimum size.

Finally, we characterized and enumerated facets of the zero forcing polytope corresponding to forts contained in pendant paths, pendant cycles, and join paths, and we showed that the number of these facets can be exponential in the order of the graph. It would be a

problem of interest to characterize and count families of facets of other related polytopes, e.g., those corresponding to the power domination or target set selection problems.

Acknowledgments

We thank three anonymous reviewers for their helpful advice and detailed comments. This work was supported by the National Science Foundation, grant number CMMI 1634550.

References

Aazami A (2008) *Hardness results and approximation algorithms for some problems on graphs*. Ph.D. thesis, University of Waterloo.

Aazami A (2010) Domination in graphs with bounded propagation: algorithms, formulations and hardness results. *Journal of Combinatorial Optimization* 19(4):429–456.

Ackerman E, Ben-Zwi O, Wolfovitz G (2010) Combinatorial model and bounds for target set selection. *Theoretical Computer Science* 411(44-46):4017–4022.

Agra A, Cerdeira JO, Requejo C (2019) A computational comparison of compact milp formulations for the zero forcing number. *Discrete Applied Mathematics* 269:169–183.

AIM Special Work Group (2008) Zero forcing sets and the minimum rank of graphs. *Linear Algebra and its Applications* 428(7):1628–1648.

Aloul FA (2005) On solving optimization problems using boolean satisfiability. *International Conference on Modeling, Simulation, and Applied Optimization*.

Audemard G, Simon L (2009) Glucose: a solver that predicts learnt clauses quality. *SAT Competition* 7–8, URL `https://www.labri.fr/perso/lsimon/glucose/`.

Bader DA, Kappes A, Meyerhenke H, Sanders P, Schulz C, Wagner D (2014) Benchmarking for graph clustering and partitioning. *Encyclopedia of Social Network Analysis and Mining*, 73–82 (Springer).

Balas E (1975) Facets of the knapsack polytope. *Mathematical programming* 8(1):146–164.

Balas E, Saltzman MJ (1989) Facets of the three-index assignment polytope. *Discrete Applied Mathematics* 23(3):201–229.

Barioli F, Barrett W, Fallat SM, Hall HT, Hogben L, Shader B, Van Den Driessche P, Van Der Holst H (2010) Zero forcing parameters and minimum rank problems. *Linear Algebra and its Applications* 433(2):401–411.

Barioli F, Barrett W, Fallat SM, Hall HT, Hogben L, Shader B, van den Driessche P, Van Der Holst H (2013) Parameters related to tree-width, zero forcing, and maximum nullity of a graph. *Journal of Graph Theory* 72(2):146–177.

Barioli F, Fallat S, Hogben L (2004) Computation of minimal rank and path cover number for certain graphs. *Linear Algebra and its Applications* 392:289–303.

Ben-Zwi O, Hermelin D, Lokshtanov D, Newman I (2011) Treewidth governs the complexity of target set selection. *Discrete Optimization* 8(1):87–96.

Boyer K, Brimkov B, English S, Ferrero D, Keller A, Kirsch R, Phillips M, Reinhart C (2019) The zero forcing polynomial of a graph. *Discrete Applied Mathematics* 258:35–48.

Bozeman C, Brimkov B, Erickson C, Ferrero D, Flagg M, Hogben L (2019) Restricted power domination and zero forcing problems. *Journal of Combinatorial Optimization* 37(3):935–956.

Brimkov B, Fast CC, Hicks IV (2019a) Computational approaches for zero forcing and related problems. *European Journal of Operational Research* 273(3):889 – 903.

Brimkov B, Mikesell D, Smith L (2019b) Connected power domination in graphs. *Journal of Combinatorial Optimization* 38(1):292–315.

Brueni DJ, Heath LS (2005) The PMU placement problem. *SIAM Journal on Discrete Mathematics* 19(3):744–761.

Burgarth D, Giovannetti V (2007) Full control by locally induced relaxation. *Physical Review Letters* 99(10):100501.

Burgarth D, Giovannetti V, Hogben L, Severini S, Young M (2015) Logic circuits from zero forcing. *Natural Computing* 14(3):485–490.

Butler S, DeLoss L, Grout J, Hall H, LaGrange J, McKay T, Smith J, Tims G (2014) Minimum rank library. Sage programs for calculating bounds on the minimum rank of a graph, and for computing zero forcing parameters. Available at `https://github.com/jasongrout/minimum_rank`.

Chiang CY, Huang LH, Li BJ, Wu J, Yeh HG (2013) Some results on the target set selection problem. *Journal of Combinatorial Optimization* 25(4):702–715.

Creignou N, Khanna S, Sudan M (2001) *Complexity classifications of boolean constraint satisfaction problems*, volume 7 (SIAM).

DeLoss L, Grout J, Hogben L, McKay T, Smith J, Tims G (2010) Techniques for determining the minimum rank of a small graph. *Linear Algebra and its Applications* 432(11):2995–3001.

Drummond J, Perrault A, Bacchus F (2015) SAT is an effective and complete method for solving stable matching problems with couples. *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Fallat SM, Hogben L (2007) The minimum rank of symmetric matrices described by a graph: a survey. *Linear Algebra and its Applications* 426(2-3):558–582.

Fischetti M (1991) Facets of the asymmetric traveling salesman polytope. *Mathematics of Operations Research* 16(1):42–56.

Grötschel M, Wakabayashi Y (1990) Facets of the clique partitioning polytope. *Mathematical Programming* 47(1-3):367–387.

Gurobi Optimization, LLC (2017) Gurobi optimizer reference manual. URL `http://www.gurobi.com`.

Haynes TW, Hedetniemi SM, Hedetniemi ST, Henning MA (2002) Domination in graphs applied to electric power networks. *SIAM Journal on Discrete Mathematics* 15(4):519–529.

Hsieh LY (2001) *On minimum rank matrices having a prescribed graph* (University of Wisconsin–Madison).

Huang LH, Chang GJ, Yeh HG (2010) On minimum rank and zero forcing sets of a graph. *Linear Algebra and its Applications* 432(11):2961–2973.

ICSEG (2018) Illinois center for a smarter electric grid. URL `http://icseg.iti.illinois.edu/power-cases/`.

Koren Y (2009) The BellKor solution to the Netflix grand prize. *Netflix prize documentation* 81(2009):1–10.

Li R, Zhou D, Du D (2004) Satisfiability and integer programming as complementary tools. *Asia and South Pacific Design Automation Conference*, 880–883.

Mahaei SM, Hagh MT (2012) Minimizing the number of PMUs and their optimal placement in power systems. *Electric Power Systems Research* 83(1):66–72.

Manquinho VM, Silva JPM, Oliveira AL, Sakallah KA (1998) Satisfiability-based algorithms for 0-1 integer programming. *Proceedings of the IEEE/ACM International Workshop on Logic Synthesis.*

Nieuwenhuis R (2015) SAT-based techniques for integer linear constraints. *Global Conference on Artificial Intelligence*, 1–13.

Nylen PM (1996) Minimum-rank matrices with prescribed graph. *Linear Algebra and its Applications* 248:303–316.

PassMark Software (2019) Cpu benchmarks. URL `https://www.cpubenchmark.net/`.

Ramani A, Markov IL, Sakallah KA, Aloul FA (2006) Breaking instance-independent symmetries in exact graph coloring. *Journal of Artificial Intelligence Research* 26:289–322.

Shitov Y (2017) On the complexity of failed zero forcing. *Theoretical Computer Science* 660:102–104.

Trefois M, Delvenne JC (2015) Zero forcing number, constrained matchings and strong structural controllability. *Linear Algebra and its Applications* 484:199–218.

Watts DJ, Strogatz SH (1998) Collective dynamics of small-world networks. *Nature* 393(6684):440.

West DB (2001) *Introduction to Graph Theory*, volume 2 (Prentice Hall Upper Saddle River).

Wolfman SA, Weld DS (2001) Combining linear programming and satisfiability solving for resource planning. *The Knowledge Engineering Review* 16(1):85–99.

Yang B (2013) Fast–mixed searching and related problems on graphs. *Theoretical Computer Science* 507:100–113.

Zahidi S, Aloul F, Sagahyroon A, El-Hajj W (2012) Using SAT & ILP techniques to solve enhanced ILP formulations of the clustering problem in MANETS. *2012 8th International Wireless Communications and Mobile Computing Conference*, 1085–1090.