Linear Algebra and its Applications • • • (• • • •) • • •–• • •

Contents lists available at ScienceDirect

Linear Algebra and its Applications

www.elsevier.com/locate/laa

On the complexity of the positive semidefinite zero forcing number

Shaun Fallat [a,*,1], Karen Meagher [a,2], Boting Yang [b,3]

[a] *Department of Mathematics and Statistics, University of Regina, Regina, Saskatchewan S4S 0A2, Canada*
[b] *Department of Computer Science, University of Regina, Regina, Saskatchewan S4S 0A2, Canada*

ARTICLE INFO

Article history:
Received 21 July 2014
Accepted 10 March 2015
Available online xxxx
Submitted by F. Zhang

MSC:
05C35
05C50
05C78
05C85
68R10

Keywords:
Positive zero forcing number
Clique cover number
Chordal graphs
Computational complexity

ABSTRACT

The positive semidefinite zero forcing number of a graph is a graph parameter that arises from a non-traditional type of graph colouring and is related to a more conventional version of zero forcing. We establish a relation between the zero forcing and the fast–mixed searching, which implies some NP-completeness results for the zero forcing problem. Relationships between positive semidefinite zero forcing sets and clique coverings are well-understood for chordal graphs. Building upon constructions associated with optimal tree covers and forest covers, we present a linear time algorithm for computing the positive semidefinite zero forcing number of chordal graphs. We also prove that it is NP-complete to determine whether a graph has a positive semidefinite zero forcing set with an additional property.

* Corresponding author.
E-mail addresses: Shaun.Fallat@uregina.ca (S. Fallat), Karen.Meagher@uregina.ca (K. Meagher), Boting.Yang@uregina.ca (B. Yang).

[1] Research supported in part by an NSERC Discovery Research Grant, Application No.: RGPIN-2014-06036.
[2] Research supported in part by an NSERC Discovery Research Grant, Application No.: RGPIN-341214-2013.
[3] Research supported in part by an NSERC Discovery Research Grant, Application No.: RGPIN-2013-261290.

http://dx.doi.org/10.1016/j.laa.2015.03.011
0024-3795/

Please cite this article in press as: S. Fallat et al., On the complexity of the positive semidefinite zero forcing number, Linear Algebra Appl. (2015), http://dx.doi.org/10.1016/j.laa.2015.03.011

1. Introduction

The zero forcing number of a graph is an important graph parameter that was introduced in [2]. In [4], the concept of the *positive semidefinite* zero forcing number, or, simply, *positive* zero forcing number, was developed. First and foremost, the interest in these parameters has been to provide a bound on the maximum nullities (or, equivalently, the minimum rank) of certain symmetric matrices associated with graphs. Recently, this parameter has been considered in other contexts, see, for example, [3,20]. Physicists have also studied this parameter in conjunction with control of quantum systems; in this context it is known as the *graph infection number* [8,19].

The same notion arises in computer science within the context of graph searching. At its most basic level, edge search and node search models represent two significant graph search problems [15,16]. Bienstock and Seymour [5] introduced the mixed search problem that combines the edge search and the node search problems. Dyer et al. [9] introduced the fast search problem. Recently, a fast–mixed search model was introduced in an attempt to combine fast search and mixed search models [21]. For this model, we assume that the simple graph G contains a single *fugitive*, invisible to the searchers, that can move at any speed along a "searcher-free" path and hides on vertices or along edges. In this case, the minimum number of searchers required to capture the fugitive is called the *fast–mixed search number of* G. As we will see, the fast–mixed search number and the zero forcing number of G are indeed equal.

Suppose that G is a simple finite graph with vertex set $V = V(G)$ and edge set $E = E(G)$. We begin by specifying a set of initial vertices of the graph (which we say are coloured black, while all other vertices are white). Then, using a designated colour change rule applied to these vertices, we progressively change the colour of white vertices in the graph to black. Our colouring consists of only two colours (black and white) and the objective is to colour all vertices black by repeated application of the colour change rule to our initial set. In general, we want to determine the smallest set of vertices that must be black initially, to eventually change all of the vertices in the graph to black.

The conventional zero forcing rule results in a partition of the vertices of the graph into sets, such that each such set induces a path in G. Further, each of the initial black vertices is an end point of one of these paths. More recently a refinement of the colour change rule, called the positive zero forcing colour change rule, was introduced. Using this rule, the positive zero forcing number was defined (see, for example, [4,10,11]). When the positive zero forcing colour change rule is applied to a set of initial vertices of a graph, the vertices are then partitioned into sets, so that each such set induces a tree in G.

As mentioned above, one of the original motivations for studying these parameters is that they both provide an upper bound on the maximum nullity of both symmetric and positive semidefinite matrices associated with a graph (see [3,4]). All matrices considered in this paper have real entries. For a given graph $G = (V, E)$, define

$$\mathcal{S}(G) = \{A = [a_{ij}] : A = A^T, \text{ for } i \neq j, a_{ij} \neq 0 \text{ if and only if } \{i, j\} \in E(G)\}$$

Please cite this article in press as: S. Fallat et al., On the complexity of the positive semidefinite zero forcing number, Linear Algebra Appl. (2015), http://dx.doi.org/10.1016/j.laa.2015.03.011

and let $\mathcal{S}_+(G)$ denote the subset of positive semidefinite matrices in $\mathcal{S}(G)$. We use $\text{null}(B)$ to denote the nullity of the matrix B. The *maximum nullity* of G is defined to be $\text{M}(G) = \max\{\text{null}(B) : B \in \mathcal{S}(G)\}$, and, similarly, $\text{M}_+(G) = \max\{\text{null}(B) : B \in \mathcal{S}_+(G)\}$ is called the *maximum positive semidefinite nullity of* G.

The zero forcing number has been studied in the context of graph search; here it is known as the *fast–mixed searching number*; the complexity of computing the zero forcing number is generally better understood. Consequently, our focus will be on the algorithmic aspects of computing the positive zero forcing number for graphs. As with most graph parameters defined in terms of an optimization problem, these parameters are complicated to compute in general. However, some very interesting exceptions arise. An example are the chordal graphs, which are the focus of this paper. For these graphs the positive zero forcing number can be found in linear time. We will also consider a variant of the positive zero forcing problem, called the Min-Forest problem. We will show that this variant is NP-complete even for a special type of chordal (split) graphs, called *echinus* graphs (see Section 5 for details and definitions).

2. Preliminaries

Throughout this paper, we only consider finite, connected graphs with no loops or multiple edges. We use $G = (V, E)$ to denote a graph with vertex set V and edge set E, and we also use $V(G)$ and $E(G)$ to denote the vertex set and edge set of G respectively. We use $\{u, v\}$ to denote an edge with endpoints u and v. For a graph $G = (V, E)$ and $v \in V$, the vertex set $\{u : \{u, v\} \in E\}$ is the *neighbourhood* of v, denoted as $N_G(v)$. For a subset $V' \subseteq V$, the *neighbourhood* of V', denoted as $N_G(V')$, is the vertex set

$$\{x : \{x, y\} \in E, x \in V \setminus V' \text{ and } y \in V'\}.$$

We use $G[V']$ to denote the subgraph induced by V', which consists of all vertices of V' and all of the edges that connect vertices of V' in G. For a vertex $v \in V$, we use $G - v$ to denote the subgraph induced by $V \setminus \{v\}$.

Let G be a graph in which every vertex is initially coloured either black or white. If u is a black vertex of G and u has exactly one white neighbour, say v, then we change the colour of v to black; this rule is called the *colour change rule*. In this case we say "u forces v" and denote this action by $u \to v$. Given an initial colouring of G, in which a set of the vertices is black and all other vertices are white, the *derived set* is the set of all black vertices, including the initial set of black vertices, resulting from repeatedly applying the colour change rule until no more changes are possible. If the derived set is the entire vertex set of the graph, then the set of initial black vertices is called a *zero forcing set*. The *zero forcing number* of a graph G is the size of the smallest zero forcing set of G; it is denoted by $Z(G)$. The procedure of colouring a graph using the colour rule is called a *zero forcing process* or simply a *forcing process*. A zero forcing process

Please cite this article in press as: S. Fallat et al., On the complexity of the positive semidefinite zero forcing number, Linear Algebra Appl. (2015), http://dx.doi.org/10.1016/j.laa.2015.03.011

is called *optimal* if the initial set of black vertices is a zero forcing set of the smallest possible size.

If Z is a zero forcing set of a graph G, then we may produce a list of the forces in the order in which they are performed in the zero forcing process. This list can then be divided into paths, known as forcing chains. A *forcing chain* is a sequence of vertices $(v_1, v_2, \ldots, v_k)$ such that $v_i \to v_{i+1}$, for $i = 1, \ldots, k-1$ in the forcing process. In every step of a forcing process, each vertex can force at most one other vertex; conversely every vertex not in the zero forcing set is forced by exactly one vertex. Thus the forcing chains that correspond to a zero forcing set partition the vertices of a graph into disjoint sets, such that each set induces a path. The number of these paths is equal to the size of the zero forcing set and the elements of the zero forcing set are the initial vertices of the forcing chains and hence end points of these paths (see [4, Proposition 2.10] for more details). In the next section we observe that the concept of clearing an edge in the fast–mixed search model is equivalent to the notion of a black vertex forcing a unique white neighbour. Hence fast–mixed searching and the zero forcing colour change rule are equivalent.

The most widely-studied variant of the zero forcing number is called positive semidefinite zero forcing or the positive zero forcing number, and was introduced in [4], see also [10] and [11]. The positive zero forcing number is also based on a colour change rule similar to the zero forcing colour change rule. Let G be a graph and B a set of vertices; we will initially colour the vertices of B black and all other vertices white. Let $W_1, \ldots, W_k$ be the sets of vertices in each of the connected components of G after removing the vertices in B. If u is a vertex in B and w is the only white neighbour of u in the graph induced by the subset of vertices $W_i \cup B$, then u can force the colour of w to black. This is the *positive colour change rule*. The definitions and terminology for the positive zero forcing process, such as, colouring, derived set, positive zero forcing number etc., are similar to those for the zero forcing number, except we use the positive colour change rule.

The size of the smallest positive zero forcing set of a graph G is denoted by $Z_+(G)$. For all graphs G, a zero forcing set is also a positive zero forcing set; thus we have that $Z_+(G) \leq Z(G)$. Moreover, in [4] it was shown that $\mathrm{M}_+(G) \leq Z_+(G)$, for any graph G.

As noted above, applying the zero forcing colour change rule to the vertices of a graph produces a path covering of the vertices in that graph. Suppose G is a graph and that Z is a positive zero forcing set for G. When the colour change rule is applied, two or more vertices can perform forces at the same time, and a vertex can force multiple vertices from different components at the same time. This implies that the positive colour change rule produces a set of vertex disjoint induced rooted trees (rather than paths) in the graph. These trees are referred to as *positive zero forcing trees*. To describe these more precisely, assume that Z is a positive zero forcing set with a chronological list of forces. Then the roots of the positive zero forcing trees for this process are the vertices in Z, and an edge between vertices indicates that one vertex forces the other in the zero forcing process.

Please cite this article in press as: S. Fallat et al., On the complexity of the positive semidefinite zero forcing number, Linear Algebra Appl. (2015), http://dx.doi.org/10.1016/j.laa.2015.03.011

More generally, a *tree covering* of a graph is a family of induced vertex disjoint trees in the graph that cover all vertices of the graph. The minimum number of such trees that cover the vertices of a graph G is the *tree cover number* of G and is denoted by $\mathrm{T}(G)$. Any set of zero forcing trees corresponding to an optimal positive zero forcing set is of size $Z_+(G)$. Hence, for any graph G, we have $\mathrm{T}(G) \leq Z_+(G)$.

A *positive zero forcing tree cover* is a tree cover in which for each tree there is a root, and the set of roots is a zero forcing set where the zero forcing process for this set is described by the edges in the graph. An *optimal* zero forcing tree cover for a graph G, is one for which there are exactly $Z_+(G)$ trees.

Throughout this paper we will use the term *optimal* to refer to the object of the smallest possible size. For example, we will consider both *optimal tree covering* and *optimal positive zero forcing tree covering* (these have the fewest possible number of trees) and also *optimal clique covers* (a clique cover with the fewest cliques).

In this paper we will focus on *chordal graphs*. A graph is chordal if it contains no induced cycles on four or more vertices. Further, we say a vertex v is *simplicial*, if the graph induced by the neighbours of v forms a complete graph (or a clique). If $\{v_1, v_2, \ldots, v_n\}$ is an ordering of the vertices of a graph G, such that the vertex v_i is simplicial in the graph $G - \{v_1, v_2, \ldots, v_{i-1}\}$, then $\{v_1, v_2, \ldots, v_n\}$ is called a *perfect elimination ordering*. Every chordal graph has an ordering of the vertices that is a *perfect elimination ordering*, see also [6,13].

In general it is known for any chordal graph G, that $\mathrm{M}_+(G) = |V(G)| - cc(G)$, where $cc(G)$ denotes the fewest number of cliques needed to cover (or to include) all the edges in G (see [14]). From [4] and [7] we know that for any graph G,

$$|V(G)| - cc(G) \leq \mathrm{M}_+(G) \leq Z_+(G). \tag{1}$$

This number, $cc(G)$, is often referred to as the *clique cover number* of the graph G. Further inspection of the work in [14] actually reveals that, in fact, for any chordal graph, $cc(G)$ is equal to the *ordered set number* of G. In [4], it was proved that for any graph G, the ordered set number of G is equal to $|V(G)| - Z_+(G)$. As a consequence, we have that $\mathrm{M}_+(G) = Z_+(G)$ for any chordal graph G, and, in particular, $Z_+(G) = |V(G)| - cc(G)$. So the value of the positive zero forcing number of chordal graphs may be deduced by determining the clique cover number and vice-versa.

3. The complexity of zero forcing

In the fast–mixed search model [21], a graph G initially contains one fugitive, who hides on vertices or along edges, and no searchers. The fugitive is invisible to searchers, and can move at any rate and at any time from one vertex to another vertex along a searcher-free path between the two vertices. An edge (or a vertex) where the fugitive may hide is said to be *contaminated*, while an edge (or a vertex) where the fugitive cannot hide is said to be *cleared*. A vertex is said to be *occupied* if it has a searcher on it. There

Please cite this article in press as: S. Fallat et al., On the complexity of the positive semidefinite zero forcing number, Linear Algebra Appl. (2015), http://dx.doi.org/10.1016/j.laa.2015.03.011

are two types of actions for searchers in each step of the fast–mixed search model (see also [21]):

1. a searcher can be placed on a contaminated vertex; or,
2. a searcher may slide along a contaminated edge $\{u, v\}$, from u to v, if v is contaminated and all edges incident on u, except $\{u, v\}$, are cleared.

In the fast–mixed search model, a contaminated edge becomes cleared if both endpoints are occupied by searchers or if a searcher slides along it from one endpoint to the other. The graph G is said to be *cleared* if all edges are cleared. The minimum number of searchers required to clear G (i.e., to capture the fugitive) is the *fast–mixed search number* of G, denoted by $\mathrm{fms}(G)$. We first show that the fast–mixed search number of a graph is equal to its zero forcing number.

Theorem 3.1. *For any graph G, $\mathrm{fms}(G) = Z(G)$.*

Proof. We first show that $\mathrm{fms}(G) \leq Z(G)$. Let B be a zero forcing set of G. We now construct a fast–mixed search strategy. Initially we place one searcher on each vertex of B. After these placings, all edges whose two endpoints are occupied are cleared. If u is a black vertex of G and u has exactly one white neighbour, say v, then u must be an occupied vertex, v and $\{u, v\}$ must be contaminated, and all edges incident with u except $\{u, v\}$ are cleared. Thus, we can slide the searcher on vertex u to vertex v along the edge $\{u, v\}$. After this sliding action, the edges between v and the occupied neighbours of v are cleared. Similarly, for each forcing action $x \to y$, we can construct a fast–mixed search action by sliding the searcher on vertex x to vertex y along the edge $\{x, y\}$. Since B is a zero forcing set of G, the zero forcing process can change all white vertices to black vertices. Thus, the corresponding fast–mixed search strategy can clear all vertices and edges of G. Hence, $\mathrm{fms}(G) \leq Z(G)$.

We next show that $\mathrm{fms}(G) \geq Z(G)$. Let S be a fast–mixed search strategy that clears G using $\mathrm{fms}(G)$ searchers. Let B be the set of vertices on which a searcher is placed (not slid to). Since a searcher can be placed only on a contaminated vertex, we know that $|B| = \mathrm{fms}(G)$. We colour all vertices of B black and all other vertices white. For a sliding action in S that slides a searcher on a vertex, say x, to a vertex, say y, along the edge $\{x, y\}$, x is black and y is the only white neighbour of x at the current stage. Thus, x forces y to black. So we can construct a zero forcing process that corresponds to the fast–mixed search strategy S such that all vertices of G are black when G is cleared by S. Therefore $\mathrm{fms}(G) \geq Z(G)$. □

Applying Theorem 3.1 with each of Theorem 6.3, Theorem 6.5 and Corollary 6.6 from [21], we get the following three results.

Corollary 3.2. *Given a graph G and a nonnegative integer k, it is NP-complete to determine whether G has a zero forcing process with k initial black vertices such that all initial*

Please cite this article in press as: S. Fallat et al., On the complexity of the positive semidefinite zero forcing number, Linear Algebra Appl. (2015), http://dx.doi.org/10.1016/j.laa.2015.03.011

black vertices are leaves of G. This problem remains NP-complete for planar graphs with maximum degree 3.

Corollary 3.3. *Given a graph G with ℓ leaves, it is NP-complete to determine whether $Z(G) = \lceil \ell/2 \rceil$. This problem remains NP-complete for graphs with maximum degree 4.*

Corollary 3.4. *Given a graph G and a nonnegative integer k, it is NP-complete to determine whether $Z(G) \leq k$. This problem remains NP-complete even for 2-connected graphs with maximum degree 4.*

We also note here that calculation of the zero forcing number has also been studied under the topic of propagation, see [1]. Further, in [1] it was observed that the solution to the propagation problem (determining ρ), which is equivalent to determining Z, is NP-hard in planar graphs.

At the end of this section, we introduce a searching model, which is an extension of the fast–mixed searching, that corresponds to positive zero forcing. This searching model, called the *parallel fast–mixed searching*, follows the same setting as the fast–mixed searching except that the graph may be split into subgraphs after each placing or sliding action, in such a way that these subgraphs may be cleared in a parallel-like fashion.

Initially, G contains no searchers, and so all vertices of G are contaminated. To begin, let $\mathcal{G} = \{G\}$. After a placing, (e.g., place a searcher on a contaminated vertex u), the subgraph $G - u$ is the graph induced by the current contaminated vertices. If $G - u$ is not connected, let $G_1, \ldots, G_j$ be all of the connected components of $G - u$. We update $\mathcal{G}$ by replacing G by subgraphs $G[V(G_1) \cup \{u\}], \ldots, G[V(G_j) \cup \{u\}]$, where u is occupied in each subgraph $G[V(G_i) \cup \{u\}]$, $1 \leq i \leq j$. Consider each subgraph $H \in \mathcal{G}$ that has not been cleared. After a placing or sliding action, let X be the set of the contaminated vertices in H. If $H[X]$ is not connected, let $X_1, \ldots, X_j$ be the vertex sets of all connected components of $H[X]$. We update $\mathcal{G}$ by replacing H by subgraphs $H[X_1 \cup N_H(X_1)\}], \ldots, H[X_j \cup N_H(X_j)]$, where $N_H(X_i)$ is occupied in each subgraph $H[X_i \cup N_H(X_i)\}]$, $1 \leq i \leq j$. We can continue this searching and branching process until all subgraphs in $\mathcal{G}$ are cleared. It is easy to observe that we can arrange the searching process so that subgraphs in $\mathcal{G}$ can be cleared in a parallel-like way.

The graph G is *cleared* if all subgraphs of $\mathcal{G}$ are cleared. The minimum number of placings required to clear G is called the *parallel fast–mixed search number* of G, and is denoted by $\mathrm{pfms}(G)$.

To illustrate the difference between the parallel fast–mixed searching and the fast–mixed searching, let G_k be a unicyclic graph, with $k \geq 4$, with vertex set $V = \{v_0, v_1, \ldots, v_{k-1}, v_k\}$ and edge set $E = \{\{v_0, v_i\} : i = 1, \ldots, k\} \cup \{\{v_{k-1}, v_k\}\}$ (see Fig. 1).

Initially $\mathcal{G} = \{G_k\}$. After we place a searcher on vertex v_0, the set $\mathcal{G}$ is updated such that it contains $k - 1$ subgraphs, i.e., the edges $\{v_0, v_1\}$, …, $\{v_0, v_{k-2}\}$, and the 3-cycle induced by the vertices $\{v_0, v_{k-1}, v_k\}$, where v_0 is occupied by a searcher in

Please cite this article in press as: S. Fallat et al., On the complexity of the positive semidefinite zero forcing number, Linear Algebra Appl. (2015), http://dx.doi.org/10.1016/j.laa.2015.03.011

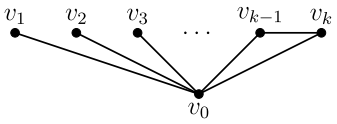

Fig. 1. An example of a graph G with $\text{pfms}(G) < \text{fms}(G)$.

each subgraph. Each edge $\{v_0, v_i\}$ $(1 \leq i \leq k-2)$ can be cleared by a sliding action. For the 3-cycle on vertices $\{v_0, v_{k-1}, v_k\}$, since no sliding action can be performed, we have to place a new searcher on a vertex, say v_{k-1}. Since the graph induced by the contaminated vertices is connected (just an isolated vertex v_k in this case), we do not need to update $\mathcal{G}$. Now we can slide the searcher on vertex v_{k-1} to vertex v_k. After the sliding action, the 3-cycle is cleared, and thus, G_k is cleared. This search strategy contains two placing actions, and it is easy to see that any strategy with only one placing action cannot cleared G_k. Thus $\text{pfms}(G_k) = 2$. On the other hand, we can easily show that $\text{fms}(G_k) = k-2$.

Similar to Theorem 3.1, we can prove the following relation between the parallel fast–mixed searching and the positive zero forcing.

Theorem 3.5. *Let G be a graph, then* $\text{pfms}(G) = Z_+(G)$.

4. A linear time algorithm for positive zero forcing number of chordal graphs

In this section, we give an algorithm for finding optimal positive zero forcing sets and optimal tree covers of chordal graphs. Our algorithm is a modification of the algorithm for computing clique covers as presented in [18].

Algorithm Zplus-Chordal
Input: A connected chordal graph G on n vertices in which all the edges and vertices are uncoloured.

Output: An optimal positive zero forcing tree cover of G and an optimal positive zero forcing set of G.

1. If $n = 1$, then mark the single vertex in G black. Output the black vertex set and stop.
2. Let $(v_1, v_2, \ldots, v_n)$ be a perfect elimination ordering of G (for example, the one given by the lexicographic breadth-first search). Set $i = 1$ and $G_i = G$.
3. For a simplicial vertex v_i in G_i, let C_i be the clique whose vertex set consists of v_i and all its neighbours in G_i.
4. If there is an edge e of G_i incident to v_i that is uncoloured, then
   (a) colour the edge e black, and colour all other uncoloured edges in C_i red;
   (b) colour v_i white;
   (c) go to Step 6.
5. If all edges of G_i incident to v_i are coloured red, then colour v_i black.

Please cite this article in press as: S. Fallat et al., On the complexity of the positive semidefinite zero forcing number, Linear Algebra Appl. (2015), http://dx.doi.org/10.1016/j.laa.2015.03.011

6. Set $G_{i+1} = G_i - v_i$, namely the graph obtained from G_i be deleting the vertex v_i and all edges incident to v_i. If $i < |V(G)| - 1$, then increment i to $i + 1$ and go to Step 3; otherwise, colour the only vertex v_{i+1} in G_{i+1} black and go to Step 7.
7. Remove all red edges from G. Let $\mathcal{T}$ be the set of all connected components of the remaining graph. Output $\mathcal{T}$ and its black vertex set and stop.

Let $V_{\text{black}}(G)$ be the set of all black vertices, and let $V_{\text{white}}(G) = V(G)\backslash V_{\text{black}}(G)$. Then $V_{\text{black}}(G)$ and $V_{\text{white}}(G)$ are the sets of all black vertices and of all white vertices, respectively, after applying Algorithm Zplus-Chordal. The next result says that the set of black vertices generated with this algorithm is a positive zero forcing set for the graph. Algorithm Zplus-Chordal is designed only for connected graphs, but it can be run on each connected component of a disconnected graph. So, without loss of generality, we will assume that our graphs are connected.

Lemma 4.1. *Let G be a connected chordal graph. Then $V_{\text{black}}(G)$ is a positive zero forcing set of G.*

Proof. The set $V_{\text{black}}(G)$ is the set of all vertices that are coloured black when Algorithm Zplus-Chordal is performed on G. We will show that these vertices form a positive zero forcing set for G. Set these vertices to be black in G.

Let $V_{\text{white}}(G) = \{w_1, w_2, \ldots, w_m\}$ where w_i is removed before w_{i+1} in Algorithm Zplus-Chordal for $1 \leq i < m$. At the iteration when w_j is coloured white, let $e_j = \{w_j, b_j\}$ be the edge that is coloured black. In particular, for the last white vertex w_m, the black edge is $e_m = \{w_m, b_m\}$ and b_m is in the set $V_{\text{black}}(G)$. We claim that in a positive zero forcing process in G starting with $V_{\text{black}}(G)$, the vertex b_m can force w_m.

Let H_m be the connected component in the subgraph of G induced by the vertices $V(G)\backslash V_{\text{black}}(G)$ that contains the last white vertex, w_m. We will show that the only vertex in H_m that is adjacent to b_m is w_m. The vertex b_m is adjacent to w_m and assume that it is also adjacent to another vertex, say u_1 in H_m (this vertex must be part of $V_{\text{white}}(G)$). Let $\{b_m, u_1, u_2, \ldots, u_k, w_m, b_m\}$ be a cycle of minimal length with $u_1, u_2, \ldots, u_k \in H_m$. Such a cycle exists since H_m is connected.

If this cycle has length three, then u_1 and w_m are adjacent. But at the iteration when u_1 is marked white, the edge $\{w_m, b_m\}$ will be coloured red. This is a contradiction, as the edge $\{w_m, b_m\}$ is black.

Assume this cycle has length more than three and let u_ℓ be the vertex in the cycle that was coloured white first. At the iteration where u_ℓ is coloured white, it is a simplicial vertex. This implies that the neighbours of u_ℓ in the cycle are adjacent. But this is a contradiction with the choice of $\{b_m, u_1, u_2, \ldots, u_k, w_m, b\}$ being a cycle of minimal length with $u_1, u_2, \ldots, u_k \in H_m$.

By the positive zero forcing rule, we know that b_m can force w_m to be black. We now change the colour of vertex w_m to black, add it to $V_{\text{black}}(G)$ and delete it from $V_{\text{white}}(G)$.

Please cite this article in press as: S. Fallat et al., On the complexity of the positive semidefinite zero forcing number, Linear Algebra Appl. (2015), http://dx.doi.org/10.1016/j.laa.2015.03.011

Similarly, at the iteration when w_{m-1} is coloured white, let $e_{m-1} = \{w_{m-1}, b_{m-1}\}$ be the edge that is coloured black. Using the above argument, we can show that b_{m-1} can force w_{m-1}. Continuing this process, all the white vertices are forced to be black. Therefore, $V_{\text{black}}(G)$ is a positive zero forcing set for G. □

For a graph G, the set $V_{\text{white}}(G) = \{w_1, w_2, \ldots, w_m\}$ is the set of all white vertices produced from applying Algorithm Zplus-Chordal to G (or on each of its connected components). Let C_i be the clique whose vertex set consists of w_i and all its neighbours at the point when w_i is coloured white. Define $\mathcal{C}(G) = \{C_1, C_2, \ldots, C_m\}$.

Every edge of G is coloured in Algorithm Zplus-Chordal and at the iteration when it is coloured, it must belong to some clique C_i. Thus we have the following lemma.

Lemma 4.2. *Let G be a chordal graph. Then $\mathcal{C}(G)$ is a clique cover of G.*

From Lemmas 4.1 and 4.2, we can prove the correctness of Algorithm Zplus-Chordal as follows.

Theorem 4.3. *Let G be a chordal graph. Then $V_{\text{black}}(G)$ is an optimal positive zero forcing set of G.*

Proof. Without loss of generality, we suppose that G is a connected chordal graph. From Lemma 4.1 we know that $V_{\text{black}}(G)$ is a positive zero forcing set of G, so we only need to show that it is the smallest possible.

From Lemma 4.2 we know that $\mathcal{C}(G)$ is a clique cover of G. Thus, $cc(G) \leq |\mathcal{C}(G)|$. Using this with (1) we have that

$$|V(G)| - |\mathcal{C}(G)| \leq |V(G)| - cc(G) \leq Z_+(G) \leq |V_{\text{black}}(G)|.$$

The pair $V_{\text{black}}(G)$ and $V_{\text{white}}(G)$ forms a partition of $V(G)$ with $|V_{\text{white}}(G)| = |\mathcal{C}(G)|$, so

$$|V(G)| - |\mathcal{C}(G)| = |V_{\text{black}}(G)|.$$

Therefore, $|V(G)| - cc(G) = Z_+(G) = |V_{\text{black}}(G)|$, and $V_{\text{black}}(G)$ is an optimal positive zero forcing set for G. □

As a byproduct from the proof of Theorem 4.3, we have the following result for chordal graphs.

Corollary 4.4. *Let G be a chordal graph. Then*

1. *$\mathcal{C}(G)$ is an optimal clique cover for G, and*
2. *$|V(G)| - cc(G) = Z_+(G)$.*

Please cite this article in press as: S. Fallat et al., On the complexity of the positive semidefinite zero forcing number, Linear Algebra Appl. (2015), http://dx.doi.org/10.1016/j.laa.2015.03.011

Note that Corollary 4.4 (1) is proved in [18] by using primal and dual linear programming. Corollary 4.4 (2) can be deduced from the work in [7], where the concept of orthogonal removal is used along with an inductive proof technique.

We can easily modify Algorithm Zplus-Chordal so that it can also output the coloured graph G. In this graph, the number of black edges is equal to the number of white vertices. From the proof of Lemma 4.1, we know that every black edge can be used to force a white vertex to black. Define $T_{\text{black}}(G)$ to be the subgraph of G formed by taking all the edges (and their endpoints) that are coloured black by Algorithm Zplus-Chordal. The next result gives some of the interesting properties of $T_{\text{black}}(G)$.

Theorem 4.5. *Let G be a chordal graph and let $T_{\text{black}}(G)$ be the subgraph formed by all the edges that are coloured black in Algorithm Zplus-Chordal. Then*

1. *the graph $T_{\text{black}}(G)$ is a forest;*
2. *all of the white vertices of G are contained among the vertices of $T_{\text{black}}(G)$;*
3. *each component of $T_{\text{black}}(G)$ contains exactly one black vertex; and*
4. *each component of $T_{\text{black}}(G)$ is an induced subgraph of G.*

Proof. First, we will simply denote $T_{\text{black}}(G)$ by T_{black}. Suppose there is a cycle C in T_{black}. Let v_i be the vertex in C with $i < j$ for all other vertices $v_j \in C$. Assume that v_i is adjacent to v_j and v_k in C. Since $\{v_i, v_j\}$ is a black edge, this is the only edge incident with v_i that is black and v_i must be white. But this implies that $\{v_i, v_k\}$ is not a black edge. Thus T_{black} contains no cycles, so T_{black} is a forest.

Whenever a vertex of G is coloured white in Step 4 of Algorithm Zplus-Chordal, an edge incident to it is marked black in the same step. Thus T_{black} contains all the vertices that are coloured white by Algorithm Zplus-Chordal.

The vertices of G, ordered by the perfect elimination ordering, are $\{v_1, v_2, \ldots, v_{|V(G)|}\}$. For a connected component T in T_{black} assume that the vertices in T are $V(T) = \{v_{i_1}, v_{i_2}, \ldots, v_{i_t}\}$, where $i_1 < i_2 < \cdots < i_t$. We will show that v_{i_t} is the only black vertex in $V(T)$. This is true if $t = 1$, so assume $t > 1$.

Assume that the vertex v_{i_t} is white. Then at Step 4 in the algorithm it is coloured white and there is an edge $e = \{v_{i_t}, v_{i_j}\}$ that is coloured black. The edge e must be in T (as it is black) so $i_j < i_t$. But $v_{i_j} \in V(G_{i_t})\backslash\{v_{i_t}\}$ and since the vertices are being removed in order, this implies that $i_j > i_t$. This is a contradiction, so v_{i_t} must be black. Next we will show that v_{i_t} is the only vertex in T that is coloured black.

Assume that v_{i_j} is the vertex with the smallest subscript among $\{i_1, i_2, \ldots, i_{t-1}\}$ that is coloured black in the algorithm. Since the vertex v_{i_j} is coloured black at Step 5 in the algorithm, all the edges in G incident to v_{i_j} in G_{i_j} are coloured red. This means that v_{i_j} is not adjacent, in T, to any vertex with a larger subscript.

Since v_{i_j} is in T, (and T is non-trivial) there is a black edge incident to v_{i_j} that is in T, say $\{v_k, v_{i_j}\}$. This edge was coloured black at Step 4 in the algorithm, which implies that $k < i_j$. Also at this step, v_k is coloured white. In Step 6 v_k, and all its incident edges are

Please cite this article in press as: S. Fallat et al., On the complexity of the positive semidefinite zero forcing number, Linear Algebra Appl. (2015), http://dx.doi.org/10.1016/j.laa.2015.03.011

removed. This implies that there can be no black edges incident to v_k that are incident to a vertex with a subscript larger than i_j. So, in T, the vertex v_k is not adjacent to any vertices with a subscript larger than i_j. Similarly, any vertex adjacent to v_k, in T, will not be adjacent to any vertex with subscript larger than i_j. In fact, any vertex with index less than i_j will not be adjacent to a vertex with subscript larger than i_j. This implies that T is not connected, so v_{i_t} is the unique black vertex in T.

Finally, we will show that any connected component T in T_{black} is an induced subgraph of G. If this is not the case, then there are two vertices v_{i_p} and v_{i_q} (we will assume that $i_p < i_q$), in $V(T)$ such that $\{v_{i_p}, v_{i_q}\}$ is an edge in G but not in T. Further assume that the distance in T between v_{i_p} and v_{i_q} is minimal over all pairs of vertices in T that are adjacent in G, but not T.

Since both vertices are in T, there is a path in T between the vertices. We claim that v_{i_p} is the vertex with the least index among all the vertices on this path. If not, there would be a vertex v_a in the path that has a black edge to two vertices in the path with a larger index. But this can't happen, since in the step when one of these edges is coloured black, the other is coloured red.

At the i_p-th iteration of the algorithm, v_{i_p} is the simplicial vertex in G_{i_p}. At this step, the edge $\{v_{i_p}, v_{i_{p+1}}\}$ (this is the first edge in the path from v_{i_p} to v_{i_q} in T) is coloured black. Since v_{i_p} is simplicial, $v_{i_{p+1}}$ and v_{i_q} are adjacent and the edge between these vertices is coloured red. This means that $v_{i_{p+1}}$ and v_{i_q} are adjacent in G, but not in T and the distance between these two vertices in T is strictly less than the distance between v_{i_p} and v_{q_i}. This is a contradiction. □

From Theorem 4.5, we have the following result.

Corollary 4.6. *Let G be a chordal graph. Then the output $\mathcal{T}$ of Algorithm Zplus-Chordal is an optimal positive zero forcing tree cover of G.*

A chordal graph is called *non-trivial* if it has at least two distinct maximal cliques; thus a trivial chordal graph is just a complete graph. Further, a simplicial vertex is called *leaf-simplicial* if none of its neighbours are simplicial. A tree with only one vertex and no edges is called a *trivial* tree. In a positive zero forcing tree cover for a graph, any tree that is trivial consists of precisely one black vertex.

Lemma 4.7. *Let G be a non-trivial chordal graph.*

1. *For any optimal tree cover $\mathcal{T}(G)$, each simplicial vertex of G is either a trivial tree in $\mathcal{T}(G)$ or a leaf of a non-trivial tree in $\mathcal{T}(G)$.*
2. *There is an optimal tree cover $\mathcal{T}(G)$ of G such that each leaf-simplicial vertex is a leaf of a non-trivial tree in $\mathcal{T}(G)$.*

Proof. Let v be a simplicial vertex of G. Let $\mathcal{T}(G)$ be an optimal tree cover of G. Let T_v be the tree in $\mathcal{T}(G)$ that contains v; this means that T_v is an induced tree in G. The

Please cite this article in press as: S. Fallat et al., On the complexity of the positive semidefinite zero forcing number, Linear Algebra Appl. (2015), http://dx.doi.org/10.1016/j.laa.2015.03.011

neighbours of v form a clique in G, so at most one of them can be in T_v. This implies that the degree of v in T_v is less than 2; thus either T_v only contains the vertex v or v is a leaf in T_v.

Next assume that v is a leaf-simplicial vertex and suppose that T_v is a trivial tree. We will show that $\mathcal{T}(G)$ can be transformed into a new optimal tree covering of G in which v is a leaf of a non-trivial tree. Let C be in the clique in G that contains v and all its neighbours.

If no edge of C is also an edge of a tree in $\mathcal{T}(G)$, then for each $u \in C$ there is a unique tree $T_u \in \mathcal{T}(G)$ that contains u. But then T_v and any T_u can be merged by adding the edge $\{u, v\}$ to T_u. This is a contradiction, as it implies that $\mathcal{T}(G)$ is not an optimal tree covering of G. So at least one edge from C is in a tree from $\mathcal{T}(G)$.

Assume that there is an edge $\{u, w\}$ in C that is also an edge of the tree $T \in \mathcal{T}(G)$. Since v is simplicial, and T is an induced subgraph, no other vertices of C can be in T. The tree T can be split into two subtrees by deleting the edge $\{u, w\}$; call these trees T_u and T_w. The subgraph T' induced by $V(T_u) \cup \{v\}$ is an induced tree in G. Replacing T and $\{v\}$ in $\mathcal{T}(G)$, with T' and T_w produces another optimal tree cover of G, in which v is a leaf in a non-trivial tree. Since w is not simplicial (as v was leaf simplicial) T_w will not be a trivial tree containing only a simplicial vertex. By using the above operation, we can transform all trivial trees in T that contain a leaf-simplicial vertex so that these vertices are leaves in trees for another optimal tree cover. □

We are now in a position to verify that the positive zero forcing number can be computed in linear time (in terms of the number of edges and vertices of G), whenever G is chordal.

Theorem 4.8. *Let G be a chordal graph with n vertices and m edges. Then Algorithm Zplus-Chordal can be implemented to find an optimal positive zero forcing set of G and an optimal positive zero forcing tree cover of G in $O(n + m)$ time.*

Proof. The lexicographic breadth-first search algorithm is a linear time algorithm that finds a lexicographic ordering of the vertices of G [17]. The reverse of a lexicographic ordering of a chordal graph is always a perfect elimination ordering. Thus, Step 2 requires $O(n + m)$ time.

In the loop between Step 3 and Step 6 in the algorithm, each edge and each vertex is coloured exactly once and deleted from the graph once. In Step 4 every edge is checked at most once to find uncoloured edges. The total running time of the loop is $O(n + m)$.

Finally, it takes linear time to remove all red edges from G in Step 7. Therefore, the running time of the algorithm is $O(n + m)$. □

5. Min-Forest problem

In this section, we consider the structure of the trees in the zero forcing tree covers of the graph. So for a given graph G and a positive integer ℓ, among all the positive

Please cite this article in press as: S. Fallat et al., On the complexity of the positive semidefinite zero forcing number, Linear Algebra Appl. (2015), http://dx.doi.org/10.1016/j.laa.2015.03.011

zero forcing tree covers of G with size ℓ, we want to minimize the number of positive zero forcing trees that are non-trivial trees. We call this the *Min-Forest Problem*. If we consider the corresponding parallel fast–mixed searching model, each non-trivial positive zero forcing tree corresponds to an induced tree cleared by a "mobile" searcher and each trivial tree corresponds to an "immobile" searcher (perhaps a trap or a surveillance camera). Typically, the goal is to minimize the number of mobile searchers among parallel fast–mixed search strategies with a given number of searchers. The decision version of the Min-Forest problem is as follows.

Min-Forest
Instance: A graph G and positive integers k and ℓ.
Question: Does G have a positive zero forcing tree cover of size ℓ in which there are at most k positive zero forcing trees that are non-trivial?

A *split graph* is a graph in which the vertices can be partitioned into two sets C and I, where C induces a clique and I induces an independent set in the graph. It is not difficult to show that a graph is split if and only if it is chordal and its complement is also chordal.

For a graph G a set $U \subseteq V(G)$ is a *vertex cover* of G if every edge of G is incident to at least one vertex in U.

Theorem 5.1. *The Min-Forest problem is NP-complete. The problem remains NP-complete for split graphs whose simplicial vertices all have degree* 2.

Proof. To verify that the Min-Forest problem is NP-complete we will construct a reduction from the vertex cover problem for cubic graphs, which, from [12], is known to be NP-complete.

Let H be a cubic graph with vertex set $\{v_1, \ldots, v_n\}$ and edge set $\{e_1, \ldots, e_m\}$. We construct a connected chordal graph G using H. First set

$$V(G) := \{v'_1, \ldots, v'_n,\ e'_1, \ldots, e'_m,\ x_1, x_2, y\}$$

where v'_i corresponds to the vertex v_i from H and e'_i corresponds to the edge e_i in H. We construct a clique in G with the vertices in the set $\{v'_1, \ldots, v'_n, x_1, x_2\}$. For each e'_i corresponding to the edge $e_i = \{v_{i_1}, v_{i_2}\}$ in H, we connect the vertex e'_i to vertices v'_{i_1} and v'_{i_2} in G. We finish the construction of G by connecting y to both vertices x_1 and x_2. It is easy to see that the graph G is a connected chordal graph and can be constructed in polynomial time.

Observe that the graph G has exactly $n+m+3$ vertices, the clique cover number for G is $m+2$, and consequently $Z_+(G) = n+1$.

Let k be a positive integer. We will show that G has a positive zero forcing tree cover of size $n+1$ in which there are at most k non-trivial trees if and only if there is a vertex cover of H of size at most k.

Please cite this article in press as: S. Fallat et al., On the complexity of the positive semidefinite zero forcing number, Linear Algebra Appl. (2015), http://dx.doi.org/10.1016/j.laa.2015.03.011

Initially, suppose that $U \subseteq V(H)$ is a vertex cover of H of size k and let U' be the set of vertices in G that correspond to vertices of U. We will show that the vertices in U' are the only vertices in a positive zero forcing set for G of size $n+1$ that are the roots for non-trivial positive zero forcing trees.

Define the set $S = \{e'_1, \ldots, e'_m, y\}$. Every vertex in S is a simplicial vertex in G, so we can assume that these are the first $m+1$ vertices in a perfect elimination ordering of the vertices. Any vertex of S has exactly two neighbours in the clique induced by $\{v'_1, \ldots, v'_n, x_1, x_2\}$. Since U is a vertex cover in H, every vertex e'_i in G is adjacent to at least one vertex in U'. Finally, since y is adjacent to x_1, each vertex $u \in S$ has at least one neighbour in $U' \cup \{x_1\}$; denote one of these neighbours by u'.

Run Algorithm Zplus-Chordal on G, assuming that the vertices in S occur first in the ordering. For each $u \in S$, at Step 4 in the algorithm, colour the edge $\{u, u'\}$ black and colour the vertex u white. At Step 6 the vertex u is removed and after $m+1$ iterations all vertices of S are removed. At this point, x_1 is a simplicial vertex. Assume that it is next in a perfect elimination ordering and at Step 4 colour the edge $\{x_1, u'\}$ black, where u' is an arbitrary vertex in U' and in Step 6 the vertex x_1 is removed.

At this point all the remaining edges have been coloured and all the vertices in the set $\{v'_1, \ldots, v'_n, x_2\}$ are coloured black and form an (optimal) positive zero forcing set for G. Let $V_{\text{black}}(G)$ be the graph formed by taking all edges (and their endpoints) that were coloured black in Algorithm Zplus-Chordal. The number of non-trivial components in T_{black} is no more than $|U'| = k$. Thus, G has a zero forcing tree cover of size $n+1$ in which there are at most k non-trivial trees.

Conversely, suppose that $\mathcal{T}$ is a positive zero forcing tree cover of G with size $n+1$ that contains k non-trivial trees. Assume that these non-trivial trees are $\{T_1, T_2, \ldots, T_k\}$ and denote the root of tree T_i by r_i. From $\mathcal{T}$, we will find a vertex cover of H with size at most k.

Define the following subset of vertices of G

$$V' = \{v'_1, \ldots, v'_n, x_1, x_2\}.$$

By definition, the vertices of V' induce a clique in G. Since each non-trivial tree T_j in $\mathcal{T}$ is an induced tree of G, it can contain at most two vertices from V'. Then these vertices will be adjacent in T_i. In particular, assume that T_i contains two vertices a and b from V'. Since T_i has only one black vertex a and b are not both black. Since they are adjacent in a zero forcing tree, one must force the other to black.

Assume that T_1 and T_2 each contain two vertices from V'. We can assume that a_1 and b_1 are in T_1 and V' where a_1 forces b_1; similarly that a_2 and b_2 are in T_2 and V' and a_2 forces b_2. We can further assume in the zero forcing process that a_2 does not force b_2, before a_1 forces b_1. But this will lead to a contradiction, since when a_1 forces b_1, it is adjacent to b_2. At this step both b_1 and b_2 are white and in the same component, so this zero forcing process contradicts the colour change rule. From this we can conclude that at most one tree in $\mathcal{T}$ contains two vertices from V'.

Please cite this article in press as: S. Fallat et al., On the complexity of the positive semidefinite zero forcing number, Linear Algebra Appl. (2015), http://dx.doi.org/10.1016/j.laa.2015.03.011

Since $|V'| = n + 2$, among the $n + 1$ positive zero forcing trees in $\mathcal{T}$, exactly one of them contains two vertices of V' and all others contain only one vertex of V'. Without loss of generality, suppose that T_1 is the unique tree in $\mathcal{T}$ that contains two vertices of V'.

Since every tree $\mathcal{T}$ contains at least one vertex from V', no vertex in $V(G) \setminus V' = \{y, e'_1, e'_2, \dots e'_m\}$ forms a trivial tree in $\mathcal{T}$. Every vertex from $V(G) \setminus V'$ is in one of the trees $\{T_1, T_2, \dots, T_k\}$. Since any tree in the set $\{T_2, \dots, T_k\}$ contains exactly one element from V', any vertex from $V(G) \setminus V'$ that is in the tree, must be a leaf in the tree.

We have three cases for T_i, and in each case we can construct a vertex cover of H that has exactly k vertices.

1. T_1 contains two vertices from $\{v'_1, \dots, v'_n\}$.
   In this case one tree in $\{T_2, \dots, T_k\}$, say T_k consists of only one edge that connects y to either x_1 or x_2. So the set of all vertices in the trees $\{T_1, T_2, \dots, T_{k-1}\}$ contains a subset of k vertices $U' \subseteq \{v'_1, \dots, v'_n\}$. Each vertex of $\{e'_1, \dots, e'_m\}$ is adjacent to one vertex of U'. Thus, the set of vertices in H, corresponding to vertices in U' is a vertex cover of H of size k.
2. T_1 contains one vertex of $\{v'_1, \dots, v'_n\}$ and x_1.
   Here one tree in $\{T_2, \dots, T_k\}$, say T_k, consists of only the edge $\{y, x_2\}$. Thus each tree in $\{T_1, \dots, T_{k-1}\}$ contains one vertex from $\{v'_1, \dots, v'_n\}$. Let $U' \subseteq \{v'_1, \dots, v'_n\}$ be the set of these $k - 1$ vertices. Each vertex of $\{e'_1, \dots, e'_m\}$ is adjacent in some T_i to a vertex of U'. Thus the vertex set in H corresponding to U', is a vertex cover of H of size $k - 1$.
3. T_1 contains both x_1 and x_2.
   In this case, each tree in $\{T_2, \dots, T_k\}$ contains exactly one vertex of $\{v'_1, \dots, v'_n\}$. Let $U' \subseteq \{v'_1, \dots, v'_n\}$ be the set of these $k - 1$ vertices. Each vertex of $\{e'_1, \dots, e'_m\}$ is adjacent in some T_i to one vertex of U'. So we have that the set of vertices in H corresponding to U' is a vertex cover of H of size $k - 1$.

In all the above cases, H has a vertex cover of size at most k, so the result holds. □

An *echinus graph* is a split graph with vertex set $\{C, I\}$, where C induces a clique and I is an independent set, such that every vertex of I has two neighbours in C and every vertex of C has three neighbours in I. It is easy to see that echinus graphs are special chordal graphs. We consider such graphs here, since it came as a surprise that the Min-Forest problem would be so complicated even when considering such a specialized class of graphs. In some sense the echinus graph is a first basic example where the number of trivial positive zero forcing trees is maximal. From the proof of Theorem 5.1, we verify that the Min-Forest problem is NP-complete for echinus graphs.

Corollary 5.2. *The Min-Forest problem remains NP-complete even for echinus graphs.*

Please cite this article in press as: S. Fallat et al., On the complexity of the positive semidefinite zero forcing number, Linear Algebra Appl. (2015), http://dx.doi.org/10.1016/j.laa.2015.03.011

Proof. In the proof of Theorem 5.1, we can modify the construction of G by adding two more vertices y' and y'' and connecting them to vertices x_1 and x_2, respectively. It is easy to see that the new graph G' is an echinus graph. Similarly, we can show that G' has a positive zero forcing tree cover of size $n+1$ in which there are at most k non-trivial trees, if and only if there is a vertex cover of H with size at most k. □

If a graph G has a positive zero forcing tree cover of size ℓ in which there are at most k non-trivial trees, then for any $n \geq \ell' > \ell$, G has a positive zero forcing tree cover of size ℓ', with at most k non-trivial trees. Note that the smallest possible value of ℓ is the positive zero forcing number. So next we consider the case when ℓ equals the positive zero forcing number. The following theorem presents a family of chordal graphs for which there exists an optimal positive zero forcing tree cover that contains only one non-trivial tree.

Theorem 5.3. *Let G be a connected non-trivial chordal graph. Further assume that for every maximal clique C in G, there are two vertices $x_C, y_C \in C$ such that any other maximal clique C' in G with $V(C') \cap V(C) \neq \emptyset$ must contain exactly one of x_C and y_C. Then there is an optimal positive zero forcing tree cover of G in which only one tree is non-trivial.*

Proof. Let $\mathcal{C}$ be the set of all maximal cliques in G. Since G is connected and non-trivial, each maximal clique in $\mathcal{C}$ must contain at least two vertices.

Assume that for every $C \in \mathcal{C}$, there is a pair of vertices x_C, y_C such that each $C' \in \mathcal{C}$ with $V(C') \cap V(C) \neq \emptyset$ contains exactly one of x_C and y_C. We call the vertices x_C and y_C the *critical vertices for* C. If C contains a critical vertex that lies in C, but not in any other clique that intersects with C, then we call this vertex a *representative* of C. Observe that since G is connected, if G contains at least two maximal cliques, then at most one critical vertex in a clique can be a representative for that clique. Fix a maximal clique $C \in \mathcal{C}$ and assume that x_C is not a representative of C. Let $\{C_1, \ldots, C_i\}$ be the set of all of the maximal cliques, other than C, in $\mathcal{C}$ that contain x_C. Define $D = C \cap C_1 \cap \cdots \cap C_i$; clearly $x_C \in D$. In fact, any vertex in D, along with y_C, forms a pair of critical vertices for C. To see this, suppose $z \in D$, and consider the set $\{y_C, z\}$ of vertices from C. Suppose C' is any maximal clique such that $C \cap C' \neq \emptyset$. Then C' must contain exactly one of x_C of y_C. If C' contains y_C, we are done, otherwise assume C' contains x_C. Then $C' \in \{C_1, \ldots, C_i\}$. Hence $z \in D \subseteq C'$. Thus the set $\{y_C, z\}$ forms a pair of critical vertices for C. Hence it follows that if two cliques intersect nontrivially, then we may assume that they share a critical vertex. Moreover, this shared critical vertex is not a representative of either clique.

Next we verify that the subgraph induced by the edges incident with two identified critical vertices of a given clique forms a tree in G. To see this, for each clique C in G colour the edge between the two critical vertices in C green. Let T_G be the graph induced on all of the green edges. Since G is a connected chordal graph and each maximal clique

Please cite this article in press as: S. Fallat et al., On the complexity of the positive semidefinite zero forcing number, Linear Algebra Appl. (2015), http://dx.doi.org/10.1016/j.laa.2015.03.011

of G contains exactly one green edge, we know that T_G does not contain a cycle. If C_i and C_j are two cliques that intersect in at least one vertex in G, then they share a critical vertex. Thus the green edges in C_i and in C_j share a vertex. Since G is connected, T_G is also connected, and hence T_G is a tree in G.

Next we will show that $T = T_G$, along with the empty trees on the vertices in $V(G) \setminus V(T)$, is an optimal positive zero forcing tree cover of G. That is, the vertices in $V(G) \setminus V(T)$ plus one more vertex will form an optimal positive zero forcing set for G.

Let b be a leaf of T. Set b and all vertices in $V(G) \setminus V(T)$ to be the initial set of black vertices in G. Since b is a leaf of T, it is contained in exactly one maximal clique in $\mathcal{C}$ (that is, b is a representative of the clique C_0). Let $\{a, b\}$ be the critical vertices in C_0. All vertices in $V(C_0) \setminus \{a\}$ are initially black and a is the unique neighbour of b in T after all black vertices are removed. Thus b forces a to black. Continuing, in this fashion, suppose $C_0, C_1, \ldots, C_i \in \mathcal{C}$ are all maximal cliques that contain a. Assume that the critical vertices for C_j are $\{a, a_j\}$, where $j \in \{1, \ldots, i\}$. Since the graph obtained from G by removing all of the current black vertices must be a subgraph of T, it follows that in each clique C_j with $1 \leq j \leq i$, all the vertices in $V(C_j) \setminus \{a_j\}$ are black. Hence a_j is a unique neighbour of a in some component containing a_j after all black vertices are removed. Thus a forces a_j black. Further, the positive zero forcing process continues until all white vertices of T are forced black. Hence, the tree T along with the vertices in $V(G) \setminus V(T)$ is a positive zero forcing tree cover of G.

Next we will show that this positive zero forcing tree is optimal. To do this we show that $cc(G) = |E(T)|$. Since no clique in G contains two edges from T, we know that $cc(G) \geq |E(T)|$. On the other hand, each edge of G is contained in a maximal clique in $\mathcal{C}$ so we also have that $cc(G) \leq |\mathcal{C}| = |E(T)|$.

Now it follows from Corollary 4.4 that

$$Z_+(G) = |V(G)| - cc(G) = |V(G)| - |E(T)| = |(V(G) \setminus V(T))| + 1.$$

Therefore, $\{T\} \cup (V(G) \setminus V(T))$ is an optimal zero forcing tree cover of G. □

Lemma 5.4. *Let G be a connected non-trivial chordal graph. Assume that there is an optimal positive zero forcing tree cover of G in which only one tree T is non-trivial. Then every maximal clique in G meets T in exactly two vertices.*

Proof. Assume that G has an optimal positive zero forcing tree cover $\mathcal{T} = \{T, v_1, \ldots, v_{m-1}\}$ in which T is the only non-trivial tree. Let $\mathcal{C}$ be the set of all maximal cliques in G. We will show that for every clique $C \in \mathcal{C}$ that $|V(C) \cap V(T)| = 2$. We have three cases to consider.

1. There is a $C \in \mathcal{C}$ such that $|V(C) \cap V(T)| = 0$.
   The clique C must contain at least two vertices, assume that these are v_1 and v_2. Then we can remove the two isolated vertices v_1 and v_2 from $\mathcal{T}$ and add the edge

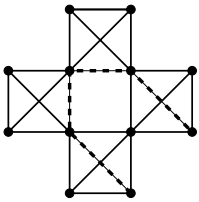

Fig. 2. An example of a graph G with an induced tree T with $|V(T)| - 1 = cc(G)$.

$\{v_1, v_2\}$ to it. This produces a new positive zero forcing tree cover that contains $m - 1$. This contradicts the optimality of the original tree cover $\mathcal{T}$.

2. There is a $C \in \mathcal{C}$ such that $|V(C) \cap V(T)| = 1$.
   Let $V(C) \cap V(T) = \{u\}$ and assume that v_1 is also in C. Thus we can remove the isolated vertex v_1 from $\mathcal{T}$ and add the edge $\{u, v_1\}$ to T. This produces a positive zero forcing tree cover of G of size $m - 1$, which is a contradiction.
3. There is a $C \in \mathcal{C}$ such that $|V(C) \cap V(T)| \geq 3$.
   Since C is a clique $V(C) \cap V(T)$ is also clique with at least 3 vertices, but this is impossible since T is a tree. □

This result can be generalized to a family of graphs that are not chordal.

Lemma 5.5. *Let G be a graph and T an induced tree in G. If $|V(T)| - 1 = cc(G)$, then G has an optimal positive zero forcing set with only one non-trivial positive zero forcing tree.*

Proof. Colour all the vertices in $V(G)\backslash V(T)$ black and colour exactly one vertex in T black. This set of black vertices forms a positive zero forcing set for which there is a positive zero forcing process where T is the only non-trivial tree. The size of this positive zero forcing set is $|V(G)| - |V(T)| + 1 = |V(G)| - cc(G)$. Thus, by (1), this set is an optimal positive zero forcing set. □

Example 5.6. To illustrate Lemma 5.5, consider the graph in Fig. 2.

Observe that the clique cover number of G is 4. Consider the induced tree T (based on the dashed edges in Fig. 2 containing 5 vertices). Hence $|V(T)| - 1 = cc(G)$. Following the algorithm in Lemma 5.5, if all remaining vertices plus one are initially coloured black, then T is the only non-trivial tree associated with this positive zero forcing tree cover.

6. Cycles of cliques

Let G be a graph and assume that $\{C_1, C_2, \ldots, C_k\}$ is a set of maximal cliques in G that covers all the edges in G. We say that G is a *cycle of cliques* if $V(C_i) \cap V(C_j) \neq \emptyset$ whenever $j = i + 1$ or $(i, j) = (k, 1)$ and $V(C_i) \cap V(C_j) = \emptyset$ otherwise. If $k = 1$ then G is a clique; we will not consider a graph that is a clique to be a cycle of cliques.

Please cite this article in press as: S. Fallat et al., On the complexity of the positive semidefinite zero forcing number, Linear Algebra Appl. (2015), http://dx.doi.org/10.1016/j.laa.2015.03.011

Lemma 6.1. *If G is a cycle of cliques $\{C_1, C_2, \ldots, C_k\}$ with $k \geq 3$, then there is a zero forcing set of size $|V(G)| - (k-2)$ and exactly one non-trivial forcing tree.*

Proof. To prove this we simply construct a zero forcing set that has this property. Colour exactly one vertex in $V(C_i) \cap V(C_{i+1})$ white for $i = 1, \ldots, k-2$ and colour all other vertices in G black. This set of black vertices forms a zero forcing set of size $|V(G)| - (k-2)$.

Start with any vertex in $V(C_k) \cap V(C_1)$. Since all the vertices in $V(C_k)$ are black, this vertex can force the only white vertex in $V(C_1) \cap V(C_2)$. In turn, this new black vertex can force the one white vertex in $V(C_2) \cap V(C_3)$, which in turn forces the one white vertex in $C_3 \cap C_4$. Continuing like this we see that the claim holds. □

If G is a cycle of cliques, then the cliques $\{C_1, C_2, \ldots, C_k\}$ form a clique cover of G. So we have that

$$|V(G)| - |\operatorname{cc}(G)| = |V(G)| - k \leq Z_+(G) \leq Z(G) \leq |V(G)| - k + 2.$$

Observe that the positive zero forcing sets in the previous lemma may not always be optimal positive zero forcing sets. But, in some cases it is possible to find an optimal zero forcing set for a cycle of cliques that has exactly one non-trivial forcing tree and is also an optimal positive zero forcing set.

Lemma 6.2. *Assume that G is a graph that is a cycle of cliques $\{C_1, C_2, \ldots, C_k\}$ with $k \geq 3$. Further assume that there is a vertex $x \in V(C_1)$ that is in no other clique and a vertex $y \in V(C_k)$ but is not in any other clique. Then there is an optimal positive zero forcing set of size $|V(G)| - k$ with exactly one non-trivial forcing tree.*

Proof. Colour exactly one vertex in $V(C_i) \cap V(C_{i+1})$ white for each of $i = 1, \ldots, k-1$, also colour the vertex $y \in V(C_k)$ white and then colour all other vertices in G black.

The vertex $x \in V(C_1)$ can force the one white vertex in $V(C_1) \cap V(C_2)$. In turn, this vertex can force the one white vertex in $V(C_2) \cap V(C_3)$, which in turn forces the one white vertex in $V(C_3) \cap V(C_4)$. Continue like this until the one white vertex in $V(C_{k-1}) \cap V(C_k)$ is forced to black. This vertex can then force y to be black.

Thus this set is a zero forcing set of size $|V(G)| - k$ that has only one non-trivial zero forcing tree. Since the clique cover number for this graph is k, from (1) we have that this is an optimal zero forcing set and an optimal positive zero forcing set. □

This also gives a family for which the positive zero forcing number and the zero forcing number agree.

7. Further work

The complexity of computing any type of graph parameter is an interesting task. For zero forcing parameters it is known that the problem of finding $Z(G)$ for a graph G is

Please cite this article in press as: S. Fallat et al., On the complexity of the positive semidefinite zero forcing number, Linear Algebra Appl. (2015), http://dx.doi.org/10.1016/j.laa.2015.03.011

NP-complete. We also suspect that the same is true for computing $Z_+(G)$ for a general graph G. In fact, we resolve part of this conjecture, by assuming an additional property on the nature of the zero forcing tree cover that results. However, for chordal graphs G, we have verified that determining the exact value of $Z_+(G)$ can be accomplished via a linear time algorithm; the best possible situation. We also believe that it would be interesting to consider the complexity of determining $Z_+(G)$ when G is a partial 2-tree, since for 2-trees it is known that the positive zero forcing number is equal to the tree cover number.

A solution of the Min-Forest problem describes an inner structure between maximal cliques of G. In Section 5, we highlight a couple of instances where we restrict the number of non-trivial trees in a positive zero forcing tree cover of a given size and then draw conclusions concerning the NP-completeness of computing the positive zero forcing number can be made. We are interested in exploring this notion further.

Acknowledgment

We would like to thank the anonymous referee for a careful reading of an earlier version of our manuscript and for the numerous suggestions which improved the presentation of our work.

References

[1] A. Aazami, Hardness results and approximation algorithms for some problems on graphs, Doctoral dissertation, University of Waterloo, 2008.
[2] AIM Minimum Rank-Special Graphs Work Group, Zero forcing sets and the minimum rank of graphs, Linear Algebra Appl. 428 (7) (2008) 1628–1648.
[3] F. Barioli, W. Barrett, S. Fallat, H.T. Hall, L. Hogben, B. Shader, P. van den Driessche, H. van der Holst, Parameters related to tree-width, zero forcing, and maximum nullity of a graph, J. Graph Theory 72 (2013) 146–177.
[4] F. Barioli, W. Barrett, S. Fallat, H.T. Hall, L. Hogben, B. Shader, P. van den Driessche, H. van der Holst, Zero forcing parameters and minimum rank problems, Linear Algebra Appl. 433 (2) (2010) 401–411.
[5] D. Bienstock, P. Seymour, Monotonicity in graph searching, J. Algorithms 12 (1991) 239–245.
[6] J.R.S. Blair, B. Peyton, An Introduction to Chordal Graphs and Clique Trees, The IMA Volumes in Mathematics and Its Applications, vol. 56, Springer-Verlag, New York, 1993, pp. 1–31.
[7] M. Booth, P. Hackney, B. Harris, C.R. Johnson, M. Lay, L.H. Mitchell, S.K. Narayan, A. Pascoe, K. Steinmetz, B.D. Sutton, W. Wang, On the minimum rank among positive semidefinite matrices with a given graph, SIAM J. Matrix Anal. Appl. 30 (2008) 731–740.
[8] D. Burgarth, V. Giovannetti, Full control by locally induced relaxation, Phys. Rev. Lett. 99 (10) (2007) 100–501.
[9] D. Dyer, B. Yang, O. Yaşar, On the fast searching problem, in: Proceedings of the 4th International Conference on Algorithmic Aspects in Information and Management (AAIM'08), in: Lecture Notes in Computer Science, vol. 5034, Springer-Verlag, New York, 2008, pp. 143–154.
[10] J. Ekstrand, C. Erickson, H.T. Hall, D. Hay, L. Hogben, R. Johnson, N. Kingsley, S. Osborne, T. Peters, J. Roat, et al., Positive semidefinite zero forcing, Linear Algebra Appl. 439 (2013) 1862–1874.
[11] J. Ekstrand, C. Erickson, D. Hay, L. Hogben, J. Roat, Note on positive semidefinite maximum nullity and positive semidefinite zero forcing number of partial 2-trees, Electron. J. Linear Algebra 23 (2012) 79–97.

Please cite this article in press as: S. Fallat et al., On the complexity of the positive semidefinite zero forcing number, Linear Algebra Appl. (2015), http://dx.doi.org/10.1016/j.laa.2015.03.011

[12] M. Garey, D. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, San Francisco, 1979.
[13] M.C. Golumbic, Algorithmic Graph Theory and Perfect Graphs, Academic Press, New York, 1980.
[14] P. Hackney, B. Harris, M. Lay, L.H. Mitchell, S.K. Narayan, A. Pascoe, Linearly independent vertices and minimum semidefinite rank, Linear Algebra Appl. 431 (2009) 1105–1115.
[15] L. Kirousis, C. Papadimitriou, Searching and pebbling, Theoret. Comput. Sci. 47 (1986) 205–218.
[16] N. Megiddo, S. Hakimi, M. Garey, D. Johnson, C. Papadimitriou, The complexity of searching a graph, J. ACM 35 (1988) 18–44.
[17] D.J. Rose, R.E. Tarjan, G.S. Lueker, Algorithmic aspects of vertex elimination on graphs, SIAM J. Comput. 5 (2) (1976) 266–283.
[18] E.R. Scheinerman, A.N. Trenk, On the fractional intersection number of a graph, Graphs Combin. 15 (1999) 341–351.
[19] S. Severini, Nondiscriminatory propagation on trees, J. Phys. A 41 (2008) 482002, Fast Track Communication.
[20] L.-H. Huang, G.J. Chang, H.-G. Yeh, On minimum rank and zero forcing sets of a graph, Linear Algebra Appl. 432 (2010) 2961–2973.
[21] B. Yang, Fast–mixed searching and related problems on graphs, Theoret. Comput. Sci. 507 (7) (2013) 100–113.

Please cite this article in press as: S. Fallat et al., On the complexity of the positive semidefinite zero forcing number, Linear Algebra Appl. (2015), http://dx.doi.org/10.1016/j.laa.2015.03.011