Simultaneous Dominating Set for Spanning Tree Factorings

Sebastian S. Johann\textsuperscript{1}, Sven O. Krumke\textsuperscript{1}, and Manuel Streicher\textsuperscript{1}

\textsuperscript{1}Technische Universität Kaiserslautern

October 31, 2018

Abstract

For a connected graph $G$ we call a set $F$ a spanning tree factoring of $G$ if it contains all spanning trees of $G$. A subset $S \subseteq V(G)$ is a simultaneous dominating set if it is a dominating set in every spanning tree $T$ in $F$. We consider the problem of finding a minimum size simultaneous dominating set for spanning tree factorings. We show that the decision version of this problem is $\text{NP}$-complete by pointing out its close relation to the vertex cover problem. We present an exact algorithm to solve this problem and show how to solve it in polynomial time on some graph classes like bipartite or chordal graphs. Moreover, we derive a 2-approximation algorithm for this problem.

1 Introduction

The dominating set problem and many variants have been thoroughly studied in the past. It can readily be seen, that it can be solved on trees in linear time, whereas it is $\text{NP}$-hard on general graphs. In this article we investigate the problem of finding a minimum subset of vertices that is a dominating set in every spanning tree of a given graph.

Most of our notation is standard graph terminology as can be found in e.g. Diestel (2000). We recall some basic notations in the following. All graphs considered in this article are simple and undirected. We denote a graph by $G = (V, E)$, where $V$ is the vertex set and $E$ the edge set. We refer to the vertex set of the graph $G$ by $V(G)$ and to the edge set by $E(G)$. We identify each edge $e \in E$ with endpoints $u, v \in V$ with the tuple $\{u, v\}$. For a vertex $v \in V$ and a subgraph $H$ of $G$ containing $v$ we say that the set $N_H(v) := \{u \in V(H) : \{v, u\} \in E(H)\}$ is the neighborhood from $v$ in $H$, where we omit the subscript $H$ in the notation when the graph is clear from context. For a subset $S \subseteq V$ we call the subgraph with vertex set $S$ and all edges between vertices in $S$ the induced subgraph of $S$ and write $G[S]$. Further we write $G - S$ for the induced subgraph $G[V \setminus S]$. For better readability we abuse notation and also write $G - H$ for the graph $G - V(H)$. Moreover for a subset $E' \subseteq E$ we write $G - E'$ for the graph with vertex set $V$ and edge set $E \setminus E'$. A vertex $v \in V$ is an articulation point if removing it increases
the number of connected components. A biconnected component of a graph is a maximal biconnected subgraph.

For a graph \( G = (V, E) \) a dominating set is a subset of the vertices \( S \subseteq V \) such that every vertex that is not in \( S \) has a neighbour in \( S \). A vertex cover is a subset \( C \subseteq V \) such that for every edge \( e \in E \) at least one of the endpoints of \( e \) is in \( C \) or equivalently for every \( v \in V \) we have \( v \in C \) or \( N(v) \subseteq C \).

A factoring is a finite set \( F \) of (not necessarily edge-disjoint) graphs with common vertex set \( V \). The combined graph \( G(F) \) has vertex set \( V \) and edge set \( \bigcup_{F \in F} E(F) \) (not necessarily the complete graph). For a connected graph \( G \) we call the factoring \( F = \{ T \colon T \text{ is a spanning tree of } G \} \) a spanning tree factoring. The combined graph of a spanning tree factoring of \( G \) is again \( G \). For a connected graph \( G \) we call the set \( F = \{ T \colon T \text{ is a spanning tree of } G \} \) a spanning tree factoring.

A subset \( S \subseteq V \) is a simultaneous dominating set or SD-set of \( F \) if \( S \) is a dominating set in each spanning tree of \( G \). Since we only consider spanning tree factorings of connected graphs the factoring is clear and we say \( S \) is an SD-set in a graph \( G \) instead. For a graph \( G = (V, E) \) and a subset \( S \subseteq V \) we say that a vertex \( v \in V \) is simultaneously dominated by \( S \) if \( v \) is dominated by \( S \) in every spanning tree of \( G \). The simultaneous dominating set problem or SDS-problem consists of finding an SD-set of minimum size.

The SDS-problem was first considered by Brigham and Dutton (1990) who termed this a factor dominating set and Sampathkumar (1989) who named this a global dominating set. The natural question is what is the minimum size of a simultaneous dominating set. For general factorings this question has been studied by Dankelmann et al. (2006) and Caro and Henning (2014). Here we use the term “simultaneous domination” rather than “factor domination” or “global domination”, as our definition of simultaneous domination slightly differs from factor domination and global domination: In factor domination the graphs in the factoring have to be edge-disjoint and in global domination the factoring only consists of a graph and its complement.

In this article we investigate the simultaneous dominating set problem for spanning tree factorings. First we show that the SDS-problem is \( \text{NP} \)-hard and give an alternative characterization. In Section 3 we present a way of solving the problem by splitting it up into smaller problems and in Section 4 we show how to solve these smaller problems with the help of the well known vertex cover problem. In Section 5 we show that the SDS-problem on bipartite, on chordal graphs and on graphs with bounded tree width can be solved in polynomial time. Finally in Section 6 we present a 2-approximation algorithm for the SDS-problem.

## 2 Complexity of the SDS-problem

Let us begin by formally introducing simultaneous dominating sets of spanning tree factorings.

**Definition 1.** For a connected graph \( G = (V, E) \) and a spanning tree factoring \( F \) we call a subset \( S \subseteq V \) a simultaneous dominating set or SD-set of \( F \) if \( S \) is a dominating set in each spanning tree of \( G \). As we only regard spanning tree factorings we also call \( S \) an SD-set of \( G \). We say that a vertex \( v \in V \) is
simultaneously dominated by \( S \) if \( v \) is dominated by \( S \) in every spanning tree of \( G \). Similarly we call a subset \( V' \subseteq V \) simultaneously dominated by \( S \) if all vertices in \( V' \) are simultaneously dominated by \( S \). The simultaneous dominating set problem or SDS-problem consists of finding an SD-set of minimum size.

Initially it is not even clear if the SDS-problem is contained in \( \text{NP} \) since a graph can have an exponential number of spanning trees and thus given a solution we may not simply test dominance in each tree. Nevertheless, there is another way for a graph \( G = (V, E) \) to verify if a set \( S \subseteq V \) is a feasible SD-set:

Let \( v \in V \setminus S \) be a vertex that is not an articulation point in \( G \). We claim that \( v \) is simultaneously dominated by \( S \) if and only if all neighbours of \( v \) are in \( S \):

If all the neighbours of \( v \) are contained in \( S \), then \( v \) is clearly dominated by \( S \) in every spanning tree of \( G \) since there is at least one edge between \( v \) and one of its neighbours, thus \( v \) is simultaneously dominated. Conversely, assume that \( v \) is simultaneously dominated by \( S \). Since \( G - v \) is connected there is a spanning tree of \( G - v \). We obtain a spanning tree of \( G \) by adding \( v \) and any edge incident to \( v \) in \( G \). Thus for any neighbour \( u \) of \( v \) in \( G \) there is at least one spanning tree of \( G \) such that \( u \) is the only neighbour of \( v \). Since \( v \) is dominated in every spanning tree of \( G \) and \( v \not\in S \) we get that all the neighbours of \( v \) must be in \( S \).

Next consider the case that \( v \) is an articulation point and thus \( G - v \) has more than one connected component, say \( H_1, \ldots, H_k \). We show that \( v \) is simultaneously dominated by \( S \) if and only if there is an \( i \in \{1, \ldots, k\} \) such that \( w \in S \) for all \( w \in N(v) \cap V(H_i) \). If for some \( i \in \{1, \ldots, k\} \) we have \( w \in S \) for all \( w \in N(v) \cap V(H_i) \), then \( v \) is clearly simultaneously dominated by \( S \) in every spanning tree of \( G \) since there is at least one edge between \( v \) and one of its neighbours in \( H_i \). Conversely, suppose that \( v \) is dominated by \( S \) in every spanning tree of \( G \). Assume that for every connected component \( H_i \) in \( G - v \) there is at least one neighbour \( u_i \) of \( v \) in \( G[V(H_i) \cup \{v\}] \) that is not in \( S \). Now we can find a spanning tree \( T \) of \( G \) by taking any spanning tree \( T_i \) in \( H_i \) for every \( i \) and adding \( v \) and for every \( i \) the edge between \( v \) and \( u_i \). But now the vertex \( v \) is not dominated in \( T \) since neither \( v \) nor any of its neighbours \( u_i \) is in \( S \), a contradiction.

This shows the following lemma:

**Lemma 2.** Let \( G = (V, E) \) be a connected graph and \( F \) a spanning tree factorization. Then a subset \( S \subseteq V \) is a simultaneous dominating set in \( G \) if and only if for every \( v \in V \) it holds true that \( v \in S \) or:

\[ (i) \ v \text{ is not an articulation point in } G \text{ and } N(v) \subseteq S, \text{ or} \]
\[ (ii) \ v \text{ is an articulation point in } G \text{ with connected components } H_1, \ldots, H_k \text{ of } G - v \text{ and for some } i \in \{1, \ldots, k\} \text{ we have } N(v) \cap V(H_i) \subseteq S. \]

This shows that we can verify for a graph \( G = (V, E) \) if a given set \( S \subseteq V \) is a feasible SD-set in \( G \) in polynomial time by checking the conditions of Lemma 2 for every vertex. Now the vertex cover problem comes into account. As mentioned before for a graph \( G = (V, E) \) a set \( C \subseteq V \) is a vertex cover if and only if for every \( v \in V \) it holds true that \( v \in S \) or \( N(v) \subseteq S \). Lemma 2 requires exactly the same for non-articulation points and hence we get:
Corollary 3. If $G = (V, E)$ is a biconnected graph and $F$ a spanning tree factoring, then a subset $S \subseteq V$ is a simultaneous dominating set in $G$ if and only if $S$ is a vertex cover in $G$.

And since the vertex cover problem on biconnected graphs is NP-complete \cite{Mohar(2001)} we obtain:

Corollary 4. For a connected graph the decision version of the SDS-problem is NP-complete.

3 An Exact Algorithm for the SDS-problem

In the last section we saw that on biconnected graphs the SDS-problem is the same as the vertex cover problem. In this section we use the vertex cover problem to solve the SDS-problem on general graphs. The algorithm heavily depends on the next well known lemma, as it provides us with biconnected subgraphs that can be partially detached from the graph after regarding it.

Lemma 5 \cite{Diestel(2000)}. Let $G$ be a connected but not biconnected graph. Then there is an articulation point $v$ in $G$ and a connected component $H$ in $G - v$ such that $G[V(H) \cup \{v\}]$ is biconnected, i.e., $H$ does not contain a vertex that is an articulation point in $G$. We call $H$ a leaf-component of the graph $G$ and we call $v$ the connection vertex of the leaf-component $H$.

In the following we assign colours to vertices. To get an intuition what these colours represent we now state our interpretation of them:

- colour 1, meaning that the vertex is in the SD-set,
- colour 0, meaning that the vertex is not in the SD-set but it is simultaneously dominated and
- colour \( \hat{0} \), meaning that the vertex is not in the SD-set and it is not simultaneously dominated at the current stage of the algorithm.

We say that colour 1 is better than colours 0 and \( \hat{0} \) and call colour 0 is better than colour \( \hat{0} \). For a set $\text{col} \subseteq \{1, 0, \hat{0}\}$ we denote the best colour of $\text{col}$ by $\text{best}\{\text{col}\}$.

Let us briefly describe the idea of the algorithm: We begin by regarding some leaf-component $H$ with connection vertex $v$ of a graph $G$. We take among all minimum size sets $S \subseteq V(H) \cup \{v\}$ that simultaneously dominate all vertices in $V(H)$, one with the best coverage for $v$, i.e. the best colour for $v$. We then remove $H$ from $G$ and continue with the next leaf-component.

In later iterations of the algorithm we then have vertices, which are basically already simultaneously dominated or even in the SD-set for free. This has to be taken into account when computing a minimum size set in some leaf component at some point of the algorithm. The crucial point why this procedure works is, that any given vertex can be simultaneously dominated by adding only one vertex to the simultaneous dominating set, namely itself. Thus, if all smallest SD-sets for some leaf-component do not simultaneously dominate the connection vertex $v$, then we simply simultaneously dominate $v$ later on, as we can be sure that it never costs us more than it would cost us to simultaneously dominate it with the current leaf-component.
In a first step we introduce a generalized version of the SDS-problem and for the moment assume that we can solve this on biconnected graphs. Section 4 then focuses on solving the generalized version on biconnected graphs.

**Definition 6.** Let $G$ be a connected graph and let $f: V(G) \rightarrow \{1, 0, \hat{0}\}$ be a mapping assigning one of the indicated colours to each vertex in $G$. We call a subset $S \subseteq V(G)$ of $f$-respecting simultaneous dominating set of $G$ if the following conditions hold:

- $f^{-1}(1) \subseteq S$.
- $S$ simultaneously dominates $f^{-1}(\hat{0})$.

If we do not specify the colouring we also use the term colour respecting simultaneous dominating set.

Thus, a colour respecting SD-set $S$ is an SD-set with the property that all vertices with colour 1 are contained in $S$ and all vertices with colour 0 do not have to be simultaneously dominated by $S$. Clearly this is a generalization of an SD-set as if we colour all vertices by $\hat{0}$ a colour respecting SD-set and an SD-set are the same thing.

As mentioned before, in the remainder of this section we assume that we are given an algorithm $crSDS$ that, given a biconnected graph $G$ and a colouring $f$ as in Definition 6 returns an $f$-respecting simultaneous dominating set of minimum size $S$ and the size of this set $#$. Section 4 will then focus on finding such an algorithm.

Algorithm 1 shows a pseudocode version of the complete procedure. Within it we use the black box algorithms $crSDS$ and $GetLeafComponent$. The latter one gets a graph $G$ which is not biconnected as an input and returns a leaf-component $H$ and the corresponding connection vertex $v$ in the fashion of Lemma 5. In each iteration of the while-loop we regard any fixed leaf-component $H$ of the remaining graph $G$ and its connection vertex $v$. We save the current colour of $v$ and compute a colour respecting SD-set in $G[V(H) \cup \{v\}]$ for all possible colours of $v$. We use the simultaneous dominating set, which is the smallest among the three possibilities, where ties are broken by the best coverage of $v$. We then delete $H$ from $G$ and continue with the remaining graph. Before we formally prove the correctness of Algorithm 1 and discuss its running time, we prove two lemmata, which make life easier in the proof of the algorithm.

Before we prove these lemmata and the correctness of the algorithm we introduce some notation that will simplify arguments during the proofs.

**Definition 7.** Let $G$ be a graph and $f: V(G) \rightarrow \{1, 0, \hat{0}\}$ some colouring of the nodes. For any induced subgraph $H$ of $G$ we denote by $f^H$ the colouring $f$ restricted to the nodes of $H$. Further, for any fixed vertex $v \in V(H)$ and $i \in \{1, 0, \hat{0}\}$ we denote by $f^H_{i=v}$ the colouring of $V(H)$ with $f^H_{i=v}(v) = i$ and $f^H_{i=v}(w) = f^H(w)$ for all $w \in V(H) \setminus \{v\}$. Finally we denote for $i \in \{1, 0, \hat{0}\}$ by $S^H_{i=v}$ a minimum $f^H_{i=v}$-respecting simultaneous dominating set in $H$. If $H = G$ we omit the superscript $H$ in the notation.
Algorithm 1: Computing a colour respecting SD-set of minimum size for a spanning tree factoring

**Input:** A connected graph $G = (V, E)$, colouring $f : V(G) \to \{1, 0, \tilde{0}\}$

**Output:** An $f$-respecting SD-set of minimum size for the spanning tree factoring of $G$

1. $S = \emptyset$
2. while $G$ is not biconnected do
3.   $H, v = \text{getLeafComponent}(G)$
4.   $c^* = f(v)$
5.   for $c \in \{1, 0, \tilde{0}\}$ do
6.     $S_c, \#c = \text{crSDS}(H \cup \{v\}, f)$
7.     $f(v) = c$
8.   if $\#1 = \#\tilde{0} = \#0$ then
9.     $f(v) = 1$
10.    $S = S \cup S_1$
11.   else if $\#1 > \#\tilde{0} = \#0$ then
12.     $f(v) = \text{best}\{c^*, 0\}$
13.     $S = S \cup S_{\tilde{0}}$
14.   else
15.     $S = S \cup S_0$
16.   $G = G - H$
17. $S_G, \# = \text{crSDS}(G, f)$
18. return $S \cup S_G$

**Lemma 8.** Let $G$ be a biconnected graph, $v \in V(G)$ a fixed vertex in $G$ and $f : V(G) \to \{1, 0, \tilde{0}\}$ some colouring. Then the following two statements hold true:

1. $|S_{v=0}| \leq |S_{v=\tilde{0}}| \leq |S_{v=1}|$.
2. $|S_{v=1}| - |S_{v=0}| \leq 1$.

**Proof.** We begin by showing that every $f_{v=0}$-respecting SD-set $S$ is also $f_{v=\tilde{0}}$-respecting. Clearly we have $f_{v=0}^{-1}(1) = f_{v=\tilde{0}}^{-1}(1) \subseteq S$. Further $f_{v=0}^{-1}(\tilde{0}) \subseteq f_{v=\tilde{0}}^{-1}(\tilde{0})$. Thus, $S$ simultaneously covers $f_{v=0}^{-1}(\tilde{0})$ for and is thereby also $f_{v=0}$-respecting. With similar arguments we get that any $f_{v=1}$-respecting SD-set is also $f_{v=\tilde{0}}$-respecting. These two small observations directly imply $\square$

To see that $\square$ also holds, let $S_0$ be a minimum $f_{v=0}$-respecting simultaneous dominating set of $G$. Then $S_0 \cup \{v\}$ is $f_{v=1}$-respecting, as

$$f_{v=1}^{-1}(1) = f_{v=0}^{-1}(1) \cup \{v\} \subseteq S_0 \cup \{v\}$$

and $f_{v=1}^{-1}(\tilde{0}) \subseteq f_{v=0}^{-1}(\tilde{0})$. This already implies that the minimum $f_{v=1}$-respecting SD-set has at most one element more than $S_0$. $\square$

**Lemma 9.** Let $G$ be a graph with some colouring $f : V(G) \to \{1, 0, \tilde{0}\}$ and $H$ be some leaf component of $G$ with connection vertex $v \in V(G)$ and let the graph $H' = G[V(H) \cup \{v\}]$. Then the following three statements hold true:
• If $|S_{v=0}^{H'}| = |S_{v=1}^{H'}| = |S_{v=1}^{H}|$ then $S_{v=1}^{H'} \cup S_{v=1}^{G-H}$ is a minimum $f$-respecting simultaneous dominating set in $G$.

• If $|S_{v=0}^{H'}| < |S_{v=1}^{H'}| = |S_{v=1}^{H}|$ then $S_{v=0}^{H'} \cup S_{v=0}^{G-H}$ is a minimum $f$-respecting simultaneous dominating set in $G$.

• If $|S_{v=0}^{H'}| = |S_{v=1}^{H'}| < |S_{v=1}^{H'}|$ then $S_{v=0}^{H'} \cup S_{v=best(f,v),0}^{G-H}$ is a minimum $f$-respecting simultaneous dominating set in $G$.

Proof. It is easy to see that all claimed sets are $f$-respecting simultaneous dominating sets in $G$, we now focus on their minimality. To this end let $S$ be a minimum $f$-respecting simultaneous dominating set in $G$. We begin with the case that $|S_{v=0}^{H'}| = |S_{v=1}^{H'}| = |S_{v=1}^{H}|$. If $v \in S$ the first statement holds as it is clear that $S$ must also be minimum restricted to $H'$ or $G - H$. So assume $v \notin S$ and regard $S \cap V(H')$. This set simultaneously dominates all vertices in $H'$ with respect to $f$ except possibly $v$. As $S_{v=0}^{H'}$ is minimum among these sets we have $|S \cap V(H')| \geq |S_{v=0}^{H'}| = |S_{v=1}^{H'}|$ and we can replace $S \cap V(H')$ by $S_{v=1}^{H'}$ without making it larger. We now have a minimum $f$-respecting simultaneous dominating set containing $v$ and get that $S_{v=1}^{H'} \cup S_{v=1}^{G-H}$ is also simultaneously dominating with respect to $f$.

Next assume that $|S_{v=0}^{H'}| < |S_{v=1}^{H'}| = |S_{v=1}^{H}|$. If $v \notin S$ is not simultaneously dominated by $S \cap V(H')$ we are done, so assume $v$ is simultaneously dominated by $S \cap V(H')$ and hence $|S \cap V(H')| > |S_{v=0}^{H'}|$. Further, if $v \in S$ by Lemma 8 we have $|S \cap V(G - H)| \geq |S_{v=1}^{G-H}| \geq |S_{v=0}^{f(v)}|$ and hence,

$$|S| = |S \cap V(H')| + |S \cap V(G - H)| - 1 > |S_{v=0}^{H'}| + |S_{v=0}^{G-H} - 1|.$$ 

If $v \notin S$ Lemma 8 implies $|S \cap V(G - H)| \geq |S_{v=0}^{G-H} - 1|$ and we get

$$|S| = |S \cap V(H')| + |S \cap V(G - H)| > |S_{v=0}^{H'}| + |S_{v=0}^{G-H} - 1|.$$ 

Both cases then imply $|S| \geq |S_{v=1}^{H'}| + |S_{v=0}^{G-H} - 1|$. 

Finally let us assume $|S_{v=0}^{H'}| = |S_{v=1}^{H'}| < |S_{v=1}^{H}|$. First note that this implies $v \notin S_{v=0}^{H'}$. Let the set $S' = (S \setminus V(H)) \cup S_{v=0}^{H'}$. Then $|S'| \leq |S|$ and $S'$ is still simultaneously dominating with respect to $f$. Furthermore it holds true that $|S' \setminus V(H)| \geq S_{v=best(f,v),0}^{G-H}$ and we get

$$|S| \geq |S'| = |S' \setminus V(H)| + |S' \cap V(H)| \geq |S_{v=best(f,v),0}^{G-H}| + |S_{v=0}^{H'}|,$$

which implies the desired result.

\(\Box\)

Theorem 10. For a connected graph $G$ and a colouring $f : V(G) \rightarrow \{1, 0, \hat{0}\}$, Algorithm 1 correctly computes a minimum $f$-respecting simultaneous dominating set $S$ of $G$. It can be implemented to run in polynomial time if CrSDDS can be implemented to run in polynomial time.

Proof. In each iteration the algorithm simultaneously dominates the current leaf-component as cheap as possible and only simultaneously dominates the connection vertex or even adds it to the set if this does not increase the size of the set simultaneously dominating the leaf-component. This intuitively makes
sense, as we can simultaneously dominate any node at any point in time by simply adding it to the set.

The proof of correctness can be regarded as a direct consequence of Lemma 9. Nevertheless we give a formal proof here for the sake of completeness. To this end note, that Algorithm 1 can be regarded as a recursive algorithm, where in each step one leaf-component is cut off the graph. We do induction on the number of biconnected components of $G$. If $G$ is biconnected the claim trivially holds. So let $H$ be a leaf-component of $G$ with connection vertex $v$ and set $H' = G[(H) \cup \{v\}]$. In the algorithm we now compute $S_v^{H\cup\{v\}}$ for $i \in \{1,0,\hat{0}\}$. By Lemma 8 the three case distinction made in the algorithm (concerning the sizes of these sets) are the only cases that may occur. The algorithm now handles the cases as follows:

- If $|S_v^{H'}| = |S_v^{H'}| = |S_v^{H'}|$ it adds $S_v^{H'}$ to the current set and colour $v$ in colour 1. Thus, by induction the algorithm returns $S_v^{H'} \cup S_v^{G-H}$, which is a minimum $f$-respecting simultaneous dominating set by Lemma 9.

- If $|S_v^{H'}| < |S_v^{H'}| = |S_v^{H'}|$ it adds $S_v^{H'}$ to the current set and leave the colour as it was. Thus, by induction the algorithm returns $S_v^{H'} \cup S_v^{G-H}$, which is a minimum $f$-respecting simultaneous dominating set by Lemma 9.

- If $|S_v^{H'}| = |S_v^{H'}| < |S_v^{H'}|$ it adds $S_v^{H'}$ to the current set and sets the colour of $v$ to best{$f(v),0$}. Thus, by induction the algorithm returns $S_v^{H'} \cup S_v^{G-H}$, which is a minimum $f$-respecting simultaneous dominating set by Lemma 9.

As we can see in all considered cases the algorithm correctly computes a minimum $f$-respecting simultaneous dominating set.

Considering the running time of Algorithm 1, note that we can find all biconnected components in linear time, cf. Hopcroft and Tarjan (1973). With a small adjustment of the usual lowpoint algorithm by Hopcroft and Tarjan (1973) we can get the components in order such that each time we regard the next component it is a leaf-component of the remaining graph. Doing this as a preprocessing step, each call to GetLeafComponent takes constant time and the deletion of $H$ is done implicitly. In each iteration, besides the three calls to crSDS we only do steps that can be realized in polynomial time, thus if crSDS can be implemented to run in polynomial time so can Algorithm 1.

4 Finding a minimum colour respecting simultaneous dominating set

In the previous section we saw how we can find a minimum SD-set provided we can find a minimum colour respecting SD-set on biconnected graphs. In this section we focus on finding such an SD-set.

Recall that a colour respecting simultaneous dominating set $S$ in a biconnected graph $G$ is in some sense a simultaneous dominating set in $G$ with the additional constraint that all vertices with colour 1 are contained in $S$ and the exception that all vertices with colour 0 do not actually have to be simultaneously
dominated, cf. Definition 6. Algorithm 2 describes how to solve this problem using an algorithm (MinVertexCover) for solving the minimum vertex cover problem as a black box.

**Algorithm 2:** crSDS($G, f$): Finding a minimum colour respecting simultaneous dominating set on biconnected graphs

| Input: A biconnected graph $G = (V, E)$ and a colouring $f: V(G) \rightarrow \{1, 0, \hat{0}\}$ |
| Output: A minimum $f$-respecting simultaneous dominating set $S$ |

1. $G = G - f^{-1}(1)$  
2. $G = G - E(G[f^{-1}(0)])$  
3. $S = \text{MinVertexCover}(G)$  
4. return $S \cup f^{-1}(1), |S \cup f^{-1}(1)|$

**Theorem 11.** Given a biconnected graph $G$ and a colouring $f: V(G) \rightarrow \{1, 0, \hat{0}\}$ Algorithm 2 returns a minimum $f$-respecting simultaneous dominating set of $G$.

It can be implemented to run in polynomial time if MinVertexCover can be implemented to run in polynomial time.

**Proof.** We begin by proving that the set returned by the algorithm, say $S^*$ is an $f$-respecting simultaneous dominating set. It is obvious that $f^{-1}(1) \subseteq S^*$, thus as $G$ is biconnected by Definition 6 we only need to prove that for all vertices $v$ with $f(v) = \hat{0}$, we have $v \in S^*$ or $N_G(v) \subseteq S^*$. So let $v \in V$ with $f(v) = 0$. After having deleted all vertices with colour 1 we do not delete edges incident to $v$. Thus, the vertex cover computed either contains $v$ itself or all neighbours of $v$ which do not have colour 1. As all deleted vertices are contained in $S^*$ the required condition follows and we conclude that $S^*$ is indeed an $f$-respecting simultaneous dominating set.

Let $G' = (G - f^{-1}(1)) - E(G[f^{-1}(0)])$. To see that the algorithm actually returns a minimum $f$-respecting simultaneous dominating set we show that for any $f$-respecting simultaneous dominating set $S$ in $G$ it holds true that $S'=f^{-1}(1)$ is a vertex cover in $G'$. The correctness then follows immediately. So let $S$ be any $f$-respecting simultaneous dominating set in $G$ and let $e = (u, v) \in E(G')$. Then at least one endpoint of $e$, say $v$, has colour $\hat{0}$ and neither $u$ nor $v$ have colour 1. Otherwise $e$ would not exist in $G'$. By Definition 6 this means either $v$ or all vertices in $N_G(v)$ are contained in $S$. But we have $u \notin N'_G(v)$, so we have $u \in S \setminus f^{-1}(1)$ or $v \in S \setminus f^{-1}(1)$. As $e$ was any edge in $E(G')$ we know that $S \setminus f^{-1}(1)$ is a vertex cover in $G'$.

It is easy to see that all steps of the algorithm, except possibly the call to MinVertexCover can be implemented to run in polynomial time.

We now know that we can solve the SDS-problem on a graph $G$ in polynomial time if we can find a minimum vertex cover on graphs strongly related to the biconnected components of $G$. In the next section we use this fact to find polynomial time algorithms for the SDS-problem on some graph classes.
5 The SDS-problem on Special Graph Classes

In this section we show that we can solve the SDS-problem in polynomial time on special graph classes. From Theorem 10 and Theorem 11 we get the following theorem:

**Theorem 12.** Let $\mathcal{H}$ be a class of graphs on which vertex cover is solvable in polynomial time. Further let $\mathcal{G}$ be a class of graphs such that for all $G \in \mathcal{G}$ and subsets $U, W \subseteq V(G)$ with $U \cap W = \emptyset$ we have $(G - U) - E(G[W]) \in \mathcal{H}$. Then we can solve the SDS-problem in polynomial time on graphs from $\mathcal{G}$.

**Bipartite Graphs** Bipartite graphs are hereditary, this means every induced subgraph is again bipartite. Even if we delete edges in the graph it remains bipartite. With the help of König’s theorem [Schrijver (2003)] and for example the Hopcroft-Karp algorithm [Hopcroft and Karp (1971)] we can compute a minimum vertex cover for bipartite graphs. By Theorem 12 the algorithm solves the SDS-problem on spanning tree factorings of bipartite graphs in polynomial time.

**Graphs with bounded Tree Width** For a graph $G = (V, E)$ a tree decomposition $D$ of $G$ is given by $(S, T)$, where $S := \{X_i \subseteq V : i \in I\}$ is a set of subsets of $V$, and $T$ is a tree with the elements of $I$ as nodes. The following three properties must hold:

1. $\bigcup_{i \in I} X_i = V$,
2. for every edge $\{u, v\} \in E$ there is at least one $X_i \in S$ containing $u$ and $v$,
3. for every $v \in V$ the node set $\{i \in S : v \in X_i\}$ induces a subtree of $T$.

The width of a tree decomposition $D$ is given by $\text{width}(D) := \max_{i \in I} |X_i| - 1$. The treewidth of a graph $G$ is

$$\text{tw}(G) := \min\{\text{width}(D) : D \text{ is a tree decomposition for } G\}.$$ 

Many algorithmic problems that are NP-complete for arbitrary graphs can be solved in polynomial time by dynamic programming for graphs with bounded treewidth. For deeper insight into the concept of tree decompositions and treewidth we refer to Bodlaender (1998).

For fixed $k$ regard the class $\mathcal{G}$ of graphs with treewidth at most $k$. Then we can find a tree decomposition of graphs in $\mathcal{G}$ in linear time, cf. Bodlaender (1998). Arnborg and Proskurowski (1989) showed that a vertex cover of minimum size can be computed for a graph with bounded treewidth and given tree decomposition in linear time. As deleting vertices or edges does not increase the treewidth by Theorem 12 we can compute an SD-set of minimum size in polynomial time for graphs from $\mathcal{G}$.

**Chordal Graphs** A bit different are chordal graphs. They are hereditary but if we delete edges in a chordal graph, then it may be not chordal anymore. However, with the help of the strong perfect graph theorem from Chudnovsky et al. (2006) we can show that the graph after the edge deletion of Algorithm 2 is perfect. Grötschel et al. (1988) showed that in perfect graphs we can compute a minimum vertex cover in polynomial time. This leads to a polynomial-time
algorithm for solving the SDS-problem on spanning tree factorings of chordal graphs by Theorem 12.

**Corollary 13.** In bipartite graphs, chordal graphs and graphs with bounded tree width we can compute a minimum simultaneously dominating set in polynomial time.

It remains to show that for chordal graphs the graph obtained after the preprocessing steps is perfect. To this end we need the strong perfect graph theorem. For a graph $G$ an **odd hole** of $G$ is an induced subgraph of $G$ which is a cycle of odd length at least 5 and an **odd antihole** of $G$ is an induced subgraph of $G$ whose complement is a hole in $G$.

**Theorem 14** (Strong perfect graph theorem, Chudnovsky et al. (2006)). A graph $G$ is perfect if and only if $G$ has no odd hole or odd antihole.

**Lemma 15.** Let $G$ be a chordal graph and $I \subseteq V(G)$. The graph $G'$ is obtained by deleting all edges between the vertices of $I$ in $G$, i.e. $G' = G - E(G[I])$. Then $G'$ is perfect.

**Proof.** Assume $G'$ has an odd hole $C_{2k+1}$. Then at most $k$ vertices of $C_{2k+1}$ can be in $I$ since $I$ is an independent set in $G'$. Hence there are two consecutive vertices on $C_{2k+1}$ which are not in $I$. Since these two vertices do not have the same neighbour in $C_{2k+1}$ and only edges between vertices of $I$ are deleted there exits a cycle in $G$ of length at least four that is contained in $G$ but has no chord. But in this case $G$ is not chordal which contradicts the assumptions and hence $G'$ cannot have an odd hole.

Now let us assume that the graph $G'$ has an odd antihole $\bar{C}_{2k+1}$ with the vertices $u_1, \ldots, u_{2k+1}$. Observe that the subgraph of $G$ induced by $\{u_1, \ldots, u_{2k+1}\}$ has exactly one additional edge in comparison to the subgraph $\bar{C}_{2k+1}$ of $G$. Otherwise if there is no additional edge in $G[\{u_1, \ldots, u_{2k+1}\}]$, then it follows that $G'[\{u_1, \ldots, u_{2k+1}\}] = \bar{C}_{2k+1}$ since no edge is deleted but this contradicts the assumption that $G$ is chordal and hence perfect. If there are two or more additional edges then there are at least three vertices in $I$ and since all the edges between the vertices in $I$ are deleted we cannot have an odd antihole in $G'$.

So assume that the additional edge is between $u_2$ and $u_3$ in $G[\{u_1, \ldots, u_{2k+1}\}]$ and so these two vertices are the only vertices of $V(\bar{C}_{2k+1})$ in $I$. Then the cycle $C = (u_2, u_3, u_1, u_4, u_2)$ is contained in $G$ and has length four but no chord. Again this contradicts the assumptions and hence $G'$ has no odd antihole.

This lemma shows that for a chordal graph the graph obtained after the edge deletion of Algorithm 2 is perfect. The set $I$ corresponds to the vertices with colour 0 at the start and hence they are an independent set in $G'$, i.e. after the deletion. Together with the earlier explanations Lemma 14 shows Corollary 13.

### 6 A 2-Approximation Algorithm for the SDS-problem

In this section we present a 2-approximation algorithm for the SDS-problem. First we formulate an integer program for the SDS-problem. Then we use the
solution of its LP relaxation to obtain an integral solution of at most twice the optimal value of the LP and thus also at most twice the optimal value of the IP.

But first we need a generalization of Lemma 5 which can be proven by an easy induction on the number of biconnected components.

**Lemma 16.** Let $G$ be a connected but not biconnected graph. Then there is an articulation point $v$ in $G$ such that at least all but one connected components in $G - v$ are leaf-components. We call a sequence $(v_1, \ldots, v_k)$ an AP-sequence if $G$ has $k$ articulation points, $v_1$ is such an articulation point and $v_i$ is such an articulation point in the graph obtained from $G$ when all leaf-components of $v_j$ are removed from $G$ for $j = 1, \ldots, i - 1$.

The following IP describes the SDS-problem for a graph $G = (V, E)$. Let $\text{AP}$ be the set of articulation points in $G$ and $N\text{AP} := V \setminus \text{AP}$. In the solution the variable $x_v$ states if the vertex $v$ is in the SD-set or not. The variable $y_v, H$ is only used if $v$ is an articulation point and states if $v$ is simultaneously dominated by the biconnected component $H$.

\[
\begin{align*}
(IP) \quad & \min x_v \\
& x_u + x_v \geq 1 \quad \forall v \in \text{NAP} \text{ and } u \in N_G(v) \quad (1) \\
& x_u \geq y_v, H \quad \forall v \in \text{AP}, \forall \text{ bicon. comp. } H \quad (2) \\
& \sum_{H: H \text{ bicon. comp.} v \in V(H)} y_{v, H} + x_v \geq 1 \quad \forall v \in \text{AP} \quad (3) \\
& x_v, y_v, H \in \{0, 1\} \quad (4)
\end{align*}
\]

**Lemma 17.** Let $G = (V, E)$ be a graph. Then the set $S = \{v: x_v = 1\}$ is an SD-set of minimum size of $G$ if and only if there is a $y$ such that $(x, y)$ is an optimal solution for (1)-(4).

**Proof.** The lemma follows if we show that the set $S = \{v: x_v = 1\}$ is an SD-set of $G$ if and only if there is a $y$ such that $(x, y)$ is a feasible solution of (1)-(4).

First let $(x, y)$ be a feasible solution for (1)-(4) and set $S = \{v: x_v = 1\}$. Note that by (3) the entries in $x_v$ and $y_{v, H}$ are only 0 or 1. By (1) we have for every non-articulation point that either itself or all its neighbours are in $S$ which implies (i) of Lemma 2. Condition (4) makes sure that $x_v$ is in $S$ or for at least one biconnected component $H$ containing $v$ that $y_{v, H}$ has value 1 and hence together with (2) all neighbours of $v$ in $H$ are in $S$. This implies (ii) of Lemma 2 and hence it follows that $S$ is a feasible SD-set.

Now suppose that $S$ is a feasible SD-set. Set $x_v = 1$ if $v \in S$ and $x_v = 0$ otherwise. We have for every articulation point $v$ that $v$ itself is in $S$ or it is simultaneously dominated, i.e. there is a biconnected component $H$ such that all neighbours of $v$ in $H$ are in $S$. We set $y_{v, H} = 1$ if and only if the latter case is true. This immediately shows that (2) and (3) are fulfilled. Condition (1) is also satisfied since this is implied by (i) of Lemma 2. This shows that $(x, y)$ is a feasible solution of (1)-(4).
Now consider the LP relaxation:

(LP) \[ \begin{align*}
\min & \quad x_v \\
\text{s.t.} & \quad x_u + x_v \geq 1 & \forall v \in \text{NAP} \text{ and } u \in N_G(v) \quad (5) \\
& \quad x_u \geq y_{v,H} & \forall v \in \text{AP}, \forall \text{bicon. comp. } H \text{ containing } v, \forall u \in N_H(v) \quad (6) \\
& \quad \sum_{H \text{ bicon. comp.} \atop v \in V(H)} y_{v,H} + x_v \geq 1 & \forall v \in \text{AP} \quad (7) \\
& \quad x_v, y_{v,H} \geq 0 
\end{align*} \]

Let \((x, y)\) be an optimal solution for the LP. We construct a new solution \((x', y')\) that will be integral in the end and at most doubles the objective function value of \((x, y)\).

The idea is to round at least one variable in (5) up so (1) is fulfilled. It remains to make sure that (6) and (7) for the articulation points in \(G\) are satisfied. To do so we use an AP-sequence in \(G\). Every biconnected component of \(G\) belongs to exactly one leaf-component of a vertex in the AP-sequence. For every vertex \(v_i\) in the AP-sequence, with leaf-components \(H_1, \ldots, H_l\), we check if (6) and (7) are already fulfilled by integral values. We show that if this is not the case, then we can find in every leaf-component \(H_p\) a vertex \(u_p\) such that \(0 < x_{u_p} < 1\) and \(x_{v_i} + \sum_{p=1}^l x_{u_p} \leq \frac{1}{2}\). We satisfy (6) and (7) by setting \(x_{v_i} = 1\) and to maintain the approximation quality of two we set \(x_{u_p} = 0\) for \(p = 1, \ldots, l\). We will see that the rounding down of the \(x\)-values has no negative effect for the rest of the algorithm.

First Rounding Step: We start with setting \(x'_v := 1\) if \(x_v \geq \frac{1}{2}\) and otherwise \(x'_v := x_v\). Moreover we set \(y'_{v,H} := \min \{x'_u : u \in N_H(v)\}\) and update it every time we change a value of \(x'\).

By (5) we have that either \(x_u\) or \(x_v\) is larger or equal than \(\frac{1}{2}\) and thus either \(x'_u\) or \(x'_v\) is set to 1 and we do not decrease the value afterwards. Thus (1) of the IP is fulfilled even if the other entry is set to 0 later on. Now let \((v_1, \ldots, v_k)\) be an AP-sequence. We consider one after another of the articulation points in the AP-sequence and make sure that (6) and (7) are fulfilled. We only decrease \(x'\) values in leaf-components of the actual articulation point. Let \(v_i\) be the current vertex, which means that the leaf-components of \(v_j\) are already deleted for \(j = 1, \ldots, i - 1\). Let \(H_1, \ldots, H_l\) be the leaf-components of \(v_i\) and \(\tilde{H}\) the biconnected component of the remaining connected component that contains \(v_i\), if there is one. We have not decreased the \(x'\) values of the current leaf-components yet since they were not leaf-components in earlier iterations. Thus we have together with (7):

\[ x'_{v_i} + \sum_{p=1}^l y'_{v_i,H_p} \geq \frac{1}{2} \quad \text{or} \quad y'_{v_i,\tilde{H}} \geq \frac{1}{2} \]

In the latter case we get by (6) and the first rounding step that all the \(x'_u = 1\) for \(u \in N_{\tilde{H}}(v_i)\) and hence we updated \(y'_{v_i,\tilde{H}} = 1\) at some point before and
thus (3) is fulfilled for $v_i$. So assume we are in the first case and no summand equals 1 because then (3) is satisfied with an integer value and we are finished.

We know there is a vertex $u_p \in N_{H_p}(v_i)$ with $x'_{u_p} = y'_{v_i,H_p}$. These vertices $u_p$ for $p = 1, \ldots, l$ have not been touched before nor will they be used again later on since they are in the leaf-components and hence are deleted after this iteration.

**Second Rounding Step** For every $v_i$ in the AP-sequence check if $v_i$ is already simultaneously dominated, if not set $x'_{v_i} := 1$ and $x_{u_p} := 0$ for $p = 1, \ldots, l$.

By doing this we at most double the value of the objective function because $x_{v_i} + \sum_{p=1}^{l} x_{u_p} \geq \frac{1}{2}$. We delete the leaf-components and continue with the next articulation point in the AP-sequence.

**Last Rounding Step** After we handled the last articulation point in the AP-sequence we set all the entries in $x'$ and $y'$ to 0 that do not equal 1.

**Theorem 18.** The described algorithm is a 2-approximation algorithm for the SDS-problem and runs in polynomial time.

**Proof.** First we show the correctness. Clearly, we made sure that the value of the objective function is at most doubled. We now show that $(x', y')$ is a feasible solution for the IP. All entries in $x'$ and $y'$ are integral. In (4) $x_u$ or $x_v$ was larger or equal to $\frac{1}{2}$ and hence $x'_u$ or $x'_v$ was set to 1. We do not decrease it later on, so (1) is satisfied. Moreover we made sure that for every articulation point $v$ at least one of the variables $x'_v$ or $y'_v$ for a biconnected component $H$ containing $v$ equals 1 and hence (3) is fulfilled. Condition (2) is also satisfied since we set $y'_{v,H}$ only to 1 if all the corresponding $x_u$ equal 1 otherwise we set it to 0.

This shows that $(x', y')$ is a feasible solution of the IP that has at most twice the value of the objective function of an optimal solution of the LP and hence of the IP.

We need polynomial time to set up and solve the LP. All the rounding steps need also polynomial time, so we get overall a polynomial running time.

7 Conclusion

We considered the simultaneous dominating set problem for spanning tree factorings of connected graphs. First we showed that this problem is NP-hard. Then we presented an algorithm that solves the SDS-problem by decomposing it in smaller problems which we can solve by using some preprocessing and vertex cover computations. We also showed that the SDS-problem is solvable in polynomial time on bipartite graphs, on chordal graphs and on graphs with bounded treewidth. Finally, we presented a 2-approximation algorithm that uses LP rounding.

An interesting open question is if there is a class of graphs on which the SDS-problem is NP-hard whereas the vertex cover problem can be solved in polynomial time. One candidate for such a class are perfect graphs as perfect graphs do not fulfill the requirements of Theorem 12.
References

S. Arnborg and A. Proskurowski. Linear time algorithms for np-hard problems restricted to partial k-trees. *Discrete Applied Mathematics*, 23(1):11 – 24, 1989. ISSN 0166-218X. doi: https://doi.org/10.1016/0166-218X(89)90031-0.

H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theor. Comput. Sci.*, 209(1-2):1–45, 1998. doi: 10.1016/S0304-3975(97)00228-4.

R. C. Brigham and R. D. Dutton. Factor domination in graphs. *Discrete Mathematics*, 86(1):127 – 136, 1990. ISSN 0012-365X. doi: https://doi.org/10.1016/0012-365X(90)90355-L.

Y. Caro and M. A. Henning. Simultaneous domination in graphs. *Graphs and Combinatorics*, 30(6):1399–1416, Nov 2014. ISSN 1435-5914. doi: 10.1007/s00373-013-1353-5.

M. Chudnovsky, N. Robertson, P. Seymour, and R. Thomas. The strong perfect graph theorem. *ANNALS OF MATHEMATICS*, 164:51–229, 2006.

P. Dankelmann, M. A. Henning, W. Goddard, and R. Laskar. Simultaneous graph parameters: Factor domination and factor total domination. *Discrete Mathematics*, 306(18):2229 – 2233, 2006. ISSN 0012-365X. doi: https://doi.org/10.1016/j.disc.2006.04.017.

R. Diestel. *Graph Theory*. Graduate Texts in Mathematics. Springer, 2000. ISBN 9780387989761.

M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer, 1988. ISBN 3-540-13624-X, 0-387-13624-X (U.S.).

J. Hopcroft and R. M. Karp. A n5/2 algorithm for maximum matchings in bipartite. In *Proceedings of the 12th Annual Symposium on Switching and Automata Theory (Swat 1971)*, SWAT ’71, pages 122–125, Washington, DC, USA, 1971. IEEE Computer Society. doi: 10.1109/SWAT.1971.1.

J. Hopcroft and R. Tarjan. Algorithm 447: Efficient algorithms for graph manipulation. *Commun. ACM*, 16(6):372–378, June 1973. ISSN 0001-0782. doi: 10.1145/362248.362272.

B. Mohar. Face covers and the genus problem for apex graphs. *Journal of Combinatorial Theory, Series B*, 82(1):102 – 117, 2001. ISSN 0095-8956. doi: https://doi.org/10.1006/jctb.2000.2026.

E. Sampathkumar. The global domination number of a graph. *J. Math. Phys. Sci.*, 23 (5):377–385, 1989.

A. Schrijver. *Combinatorial Optimization - Polyhedra and Efficiency*. Springer, 2003.