Simultaneous Dominating Sets for Spanning Tree Factorings

Sebastian S. Johann — **Technische Universität Kaiserslautern**
Sven O. Krumke — **Technische Universität Kaiserslautern**
Manuel Streicher — **Technische Universität Kaiserslautern**

Abstract

A factoring $\mathcal{F}$ is a set of graphs on the same vertex set V. A subset $S \subseteq V$ is a simultaneous dominating set if for every $F \in \mathcal{F}$ each vertex not in S is adjacent to a vertex in S, i.e., the set S is simultaneously a dominating set in every graph in the factoring $\mathcal{F}$. We call $\mathcal{F}$ a spanning tree factoring if it contains all spanning trees of a connected graph G. We consider the problem of finding a minimum size simultaneous dominating set for spanning tree factorings. We show that this problem is NP-complete and present a way to solve it in polynomial time on special graph classes like bipartite or chordal graphs.

1 Introduction

Given a collection of graphs $\mathcal{F}$ on the same vertex set V we look at those subsets of V that dominate all the graphs simultaneously. This was first considered by Brigham and Dutton [1] who termed this a factor dominating set and Sampathkumar [2] 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 in [3] and [4]. Here we use the term "simultaneous domination" rather than "factor domination" or "global domination" since they are slightly different.

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

A subset $S \subseteq V$ is a *simultaneous dominating set* or *SD-set* of $\mathcal{F}$ if S is a dominating set in each graph of $\mathcal{F}$. 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. The *simultaneous dominating set problem* or *SDS-problem* consists of finding an SD-set of minimum size.

In this article we investigate the simultaneous dominating set problem for spanning tree factorings. We show that the SDS-problem is NP-hard, give an alternative definition and present a way of solving the problem by using vertex cover. We also present how to solve the SDS-problem on bipartite or chordal graphs in polynomial time.

2 Complexity and Solution Approach of the SDS-problem

Initially it is not even clear if the problem is contained in NP since a graph can have an exponential number of spanning trees. We will see that it is possible to find another way to verify if a set $S \subseteq V$ is a feasible SD-set in G.

Let $v \in V$ be a vertex that is not an articulation point in G. Since $G - v$ is connected there is a spanning tree T of $G - v$. We obtain a spanning tree of G by adding v and any outgoing edge

of v in G. Since v needs to be dominated in every spanning tree of G the vertex v itself or all the neighbours of v must be in the SD-set.

If v is an articulation point in G, then $G-v$ has more than one connected component. But any spanning tree of G contains v and at least one outgoing edge from v to every connected component of $G-v$. This shows that to dominate v in every spanning tree of G it is enough if the vertex v itself is in the SD-set or it is simultaneously dominated by at least one connected component H of $G-v$. Since v is not an articulation point in $G[V(H)\cup\{v\}]$ *simultaneously dominated by* H means that all neighbours of v in $G[V(H)\cup\{v\}]$ are in the SD-set. This shows the following lemma:

Lemma 1. *Let $G=(V,E)$ be a connected graph and $\mathcal{F}$ a spanning tree factoring. Then a subset $S\subseteq V$ is a simultaneous dominating set if and only if for every $v\in V$ it holds true that:*

- *If v is not an articulation point in G, then $v\in S$ or $N(v)\subseteq S$*
- *If v is an articulation point in G, then $v\in S$ or for at least one connected component of $G-v$ all the vertices that are neighbours of v in G are in S.*

This shows that we can verify if a set S is a feasible SD-set in G in polynomial time. If we take a closer look at the condition for non-articulation points we notice that we have the same condition for every vertex in a vertex cover. Hence we get:

Corollary 2. *If $G=(V,E)$ is a biconnected graph and $\mathcal{F}$ a spanning tree factoring, then a subset $S\subseteq V$ is a simultaneous dominating set if and only if S is a vertex cover.*

And since vertex cover is NP-hard on biconnected graphs [5] we obtain:

Corollary 3. *For a connected graph the SDS-problem is NP-hard.*

Now we present how to solve the SDS-problem on graphs that are not biconnected by decomposing them in biconnected components. To this end we need the following lemma:

Lemma 4. [6] *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 an articulation point in G. We call H a leaf-component.*

The idea of the algorithm is to take v and H as in Lemma 4, use a minimum size vertex cover to find a solution, delete H to obtain a smaller graph and from now on only v matters for the rest of the algorithm. But there may be conditions on some vertices of H since they could have been articulation points in earlier iterations of the algorithm.

For easier notation we define colours that we assign to the vertices during the algorithm:

- colour 1, meaning that the vertex is in the SD-set,
- colour 0, meaning that the vertex is simultaneously dominated by its neighbours and
- colour $\hat{0}$, meaning that the vertex is not simultaneously dominated at the current stage of the algorithm.

We say that colour 1 is better than colour 0 or $\hat{0}$ since if a vertex has colour 1, then there are no further conditions for its neighbours. Colour $\hat{0}$ is only assigned to articulation points and means that it has to be simultaneously dominated in another iteration of the algorithm. Therefore we say that colour 0 is better than colour $\hat{0}$ since if an articulation point has colour 0 it is already simultaneously dominated.

Now we can define the subproblem that we have to solve on v and H as in Lemma 4:

Definition 5 (SDS-subproblem)**.** Let $G = (V, E)$ be a biconnected graph with $v \in V$ and $H := G - v$. Some vertices in H may have colours assigned but v has none. The *SDS-subproblem* for v and H consists of finding a subset $S \subseteq V(H) \cup \{v\}$ with the least amount of 1-assignments and if there are more than one, then the one with the best colour for v, where it holds true that:

- all vertices in $V(H)$ have colour 0 or 1,
- vertices that had assigned colour 1 at the start still have colour 1 at the end and
- vertices that had no colour or colour $\hat{0}$ have colour 1 or all their neighbours have colour 1.

Note that for a solution of the SDS-subproblem, where colour 0 or $\hat{0}$ is assigned to v, the best set that fulfills the last three conditions above and v has assigned colour 1 has exactly one more 1-assignment than the solution of the SDS-subproblem.

Now we can solve the SDS-problem on a connected graph G and a spanning tree factoring $\mathcal{F}$:

Algorithm 1: Computing an SD-set of a spanning tree factoring

Input: A connected graph $G = (V, E)$
Output: A minimum SD-set for the spanning tree factoring of G

```
S = ∅;
if G is biconnected then return S = minimum vertex cover;
while G is not biconnected do
    Find an articulation point v and a leaf-component H as in Lemma 4;
    Solve the SDS-subproblem for v and H as if v has no colour and add the solution to S;
    Save the colour for v, if there is already one take the better one;
    Delete H from G;
Solve the SDS-subproblem for any vertex v and G − v and add the solution to S;
if v has colour 1 or 0 then return S;
else return S ∪ {v};
```

Theorem 6. *The algorithm computes a simultaneous dominating set of minimum size.*

Next we show how we can solve an SDS-subproblem by using vertex cover. Therefore we have to compute three sets, one for each colour assigned to v. For a solution, where v has colour

- 1 in the end, we assign colour 1 to v at the start,
- 0 in the end, we assign colour 0 to v and 1 to all neighbours of v,
- $\hat{0}$ in the end, we remove v and assign colour 1 to all neighbours of v in H with no colour or colour $\hat{0}$.

Then we continue with:

- every vertex with colour 1 gets deleted from $G[V(H) \cup \{v\}]$ and
- all edges between vertices with colour 0 get deleted.

Now we compute a minimum vertex cover in the obtained graph, where colour 1 means in the vertex cover and 0 not in the vertex cover. We choose the solution with the least amount of 1-assignments and if there are more than one, then the one with the best colour for v. Tedious calculations show that this yields a solution of the SDS-subproblem.

Now we show that we can solve the SDS-problem on bipartite graphs in polynomial time. Bipartite graphs are hereditary, that means every induced subgraph is again bipartite. For every SDS-subproblem this means that $G[V(H) \cup \{v\}]$ is bipartite. When we delete vertices the obtained graph is still an induced subgraph and even if we delete edges in the graph the graph stays bipartite. Therefore the preprocessed graphs for the SDS-subproblem are bipartite. With the help of König's theorem and for example the Hopcroft-Karp algorithm [7] we can compute a minimum vertex cover for bipartite graphs and hence solve the SDS-subproblem in polynomial time. Thus the algorithm solves the SDS-problem on bipartite graphs and spanning tree factorings in polynomial time.

Next we consider chordal graphs which are also 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 [8] we can show that the preprocessed graph is perfect. In perfect graphs we can compute a minimum vertex cover in polynomial time [9]. This leads to a polynomial-time algorithm to solve the SDS-subproblem and hence the SDS-problem on chordal graphs and spanning tree factorings.

It also follows from our results that the SDS-problem is solvable in polynomial time on graphs with bounded tree width since in this case we can solve the vertex cover problem in polynomial time.

3 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 SDS-subproblems which we can solve by using vertex cover. The advantage of this algorithm over solving an integer program for the SDS-problem is that in general the SDS-subproblems are solved on smaller graphs. Hence for large graphs a solution is obtained immensely faster when using the presented algorithm, where the integer programs for the SDS-problem or for a minimum vertex cover are solved by using, for example, Gurobi. We also showed that the SDS-problem is solvable in polynomial time on bipartite and on chordal graphs.

For further research we want to consider the SDS-problem on other factorings or even on collection of graphs, where each graph has only a subset of V as its vertex set.

References

[1] R. Brigham, R. Dutton. Factor domination in graphs. Discrete Math. 86 (1990) 127-136

[2] E. Sampathkumar. The global domination number of a graph. J. Math. Phys. Sci. 23 (1989), 377-385

[3] P. Dankelmann, W. Goddard, M. Henning, R. Laskar. Simultaneous graph parameters: Factor domination and factor total domination. Discrete Math. 306 (2006), 2229-2233

[4] Y. Caro, M. Henning. Simultaneous Domination in Graphs. Graphs and Comb. 30 (2014) 1399-1416

[5] B. Mohar. Face Covers and the Genus Problem for Apex Graphs. Journal of Combinatorial Theory, Series B 86 (2001) 102-117

[6] F. Harary. Graph Theory. Addison-Wesley Series in Mathematics, 1969

[7] J. Hopcroft, R. Karp. An $n^{\frac{5}{2}}$ algorithm for maximum matchings in bipartite graphs. SIAM J. Comput, 2(4) (1973) 225-231.

[8] M. Chudnovsky, N. Robertson, P. Seymour, R. Thomas. The strong perfect graph theorem. Annals of Mathematics. Second Series, Princeton University, 2006

[9] M. Grötschel, L. Lovász, A. Schrijver. Geometric Algorithms and Combinatorial Optimization, 1988