J Comb Optim (2018) 36:1221–1242
https://doi.org/10.1007/s10878-017-0205-2

Safe sets in graphs: Graph classes and structural parameters

Raquel Águeda[1] · Nathann Cohen[2] · Shinya Fujita[3] · Sylvain Legay[2] · Yannis Manoussakis[2] · Yasuko Matsui[4] · Leandro Montero[2] · Reza Naserasr[5] · Hirotaka Ono[6] · Yota Otachi[7] · Tadashi Sakuma[8] · Zsolt Tuza[9,10] · Renyu Xu[11]

Published online: 17 November 2017

Abstract A *safe set* of a graph $G = (V, E)$ is a non-empty subset S of V such that for every component A of $G[S]$ and every component B of $G[V \setminus S]$, we have $|A| \geq |B|$ whenever there exists an edge of G between A and B. In this paper, we show that a minimum safe set can be found in polynomial time for trees. We then further extend

Partially supported by NKFIH Grant Number 116095 (to ZsT), and MEXT/JSPS KAKENHI Grant Numbers 15K04979 (to SF), 16K05260 (to TS), 24106004 (to HO and YO), 24220003 (to HO), 26400185 (to TS), 26540005 (to HO).

✉ Yota Otachi
otachi@cs.kumamoto-u.ac.jp

Raquel Águeda
Raquel.Agueda@uclm.es

Nathann Cohen
nathann.cohen@gmail.com

Shinya Fujita
fujita@yokohama-cu.ac.jp

Sylvain Legay
sylvain.legay@lri.fr

Yannis Manoussakis
Yannis.Manoussakis@lri.fr

Yasuko Matsui
yasuko@tokai-u.jp

Leandro Montero
lmontero@lri.fr

Reza Naserasr
Reza.Naserasr@liafa.univ-paris-diderot.fr

Hirotaka Ono
ono@i.nagoya-u.ac.jp

the result and present polynomial-time algorithms for graphs of bounded treewidth, and also for interval graphs. We also study the parameterized complexity. We show that the problem is fixed-parameter tractable when parameterized by the solution size. Furthermore, we show that this parameter lies between the tree-depth and the vertex cover number. We then conclude the paper by showing hardness for split graphs and planar graphs.

Keywords Safe set · Graph algorithm · Graph class · Parameterized complexity

1 Introduction

In this paper, we only consider finite and simple graphs. The subgraph of a graph G induced by $S \subseteq V(G)$ is denoted by $G[S]$. A *component* of G is a connected induced subgraph of G with an inclusionwise maximal vertex set. For vertex-disjoint subgraphs A and B of G, if there is an edge between A and B, then A and B are *adjacent*.

In a graph $G = (V, E)$, a non-empty set $S \subseteq V$ of vertices is a *safe set* if, for every component A of $G[S]$ and every component B of $G[V \setminus S]$ adjacent to A, it holds that $|A| \geq |B|$. If a safe set induces a connected subgraph, then it is a *connected safe set*. The *safe number* $\mathtt{s}(G)$ of G is the size of a minimum safe set of G, and the *connected safe number* $\mathtt{cs}(G)$ of G is the size of a minimum connected safe set of G. It is known that $\mathtt{s}(G) \leq \mathtt{cs}(G) \leq 2 \cdot \mathtt{s}(G) - 1$ (Fujita et al. 2016).

The concept of (connected) safe number was introduced by Fujita et al. (2016). Their motivation came from a variant of facility location problems, where the goal is to find a "safe" subset of nodes in a network that can be used for safe evacuation. See

Tadashi Sakuma
sakuma@e.yamagata-u.ac.jp

Zsolt Tuza
tuza@dcs.uni-pannon.hu

Renyu Xu
renyu.xu@lri.fr

[1] Universidad de Castilla-La Mancha, Ciudad Real, Spain

[2] LRI, University Paris-Sud, Orsay, France

[3] Yokohama City University, Yokohama, Japan

[4] Tokai University, Tokyo, Japan

[5] LIAFA, University Paris-Diderot, Paris, France

[6] Nagoya University, Nagoya, Japan

[7] Kumamoto University, Kumamoto, Japan

[8] Yamagata University, Yamagata, Japan

[9] MTA Rényi Institute, Budapest, Hungary

[10] University of Pannonia, Veszprém, Hungary

[11] Shandong University, Jinan, China

Bapat et al. (2016) also for further discussions about the motivation. Fujita et al. (2016) showed that the problems of finding a minimum safe set and a minimum connected safe set are NP-hard in general. They also showed that a minimum connected safe set in a tree can be found in linear time.

The main contribution of this paper is to give polynomial-time algorithms for finding a minimum safe set on trees, graphs of bounded treewidth, and interval graphs. We also show that the problems are fixed-parameter tractable when parameterized by the solution size. These positive results are complemented by a few hardness results.

The rest of the paper is organized as follows. In Sect. 2, we present an $O(n^5)$-time algorithm for finding a minimum safe set on trees. In Sect. 3, we generalize the algorithm to make it work on graphs of bounded treewidth. In Sect. 4, we show that the problem can be solved in $O(n^8)$ time for interval graphs. In Sect. 5, we show the fixed-parameter tractability of the problem when the parameter is the solution size. We also discuss the relationship of safe number to other important and well-studied graph parameters. In Sect. 6, we show that the problems are NP-complete for split graphs and planar graphs.

2 Safe sets in trees

Recall that a tree is a connected graph with no cycles. In this section, we prove the following theorem.

Theorem 2.1 *For an n-vertex tree, a safe set of minimum size can be found in time $O(n^5)$.*

We only show that the size of a minimum safe set can be computed in $O(n^5)$ time. It is straightforward to modify the dynamic program below for computing an actual safe set in the same running time.

In the following, we assume that a tree $T = (V, E)$ has a root and that the children of each vertex are ordered. For a vertex $u \in V$, we denote the set of children of u by $C_T(u)$. By V_u we denote the vertex set that consists of u and its descendants, where a vertex $v \neq u$ is a *descendant* of u if u is on the path from v to the root. We define some subtrees induced by special sets of vertices as follows (see Fig. 1):

- For a vertex $u \in V$, let $T(u) = T[V_u]$.
- For an edge $\{u, v\} \in E$ where v is the parent of u, let $T(u \to v) = T[\{v\} \cup V_u]$.
- For $u \in V$ with children $w_1, \ldots, w_d$, let $T(u, i) = T\left[\{u\} \cup \bigcup_{1 \le j \le i} V_{w_j}\right]$.

Note that $T(u, 1) = T(w_1 \to u)$ if w_1 is the first child of u, $T(u) = T(u, |C_T(u)|)$ if u is not a leaf, and $T = T(\rho)$ if ρ is the root of T.

Fragments For a subtree T' of T and $S \subseteq V(T')$, a *fragment in T' with respect to S* is the vertex set of a component in $T'[S]$ or $T'[V(T') \setminus S]$. We denote the set of fragments in T' with respect to S by $\mathcal{F}(T', S)$.The fragment that contains the root of T' is *active*, and the other fragments are *inactive*. Two fragments in $\mathcal{F}(T', S)$ are *adjacent* if there is an edge of T' between them. A fragment $F \in \mathcal{F}(T', S)$ is *bad* if it is inactive, $F \subseteq S$, and there is another inactive fragment $F' \in \mathcal{F}(T', S)$ adjacent to F with $|F| < |F'|$.

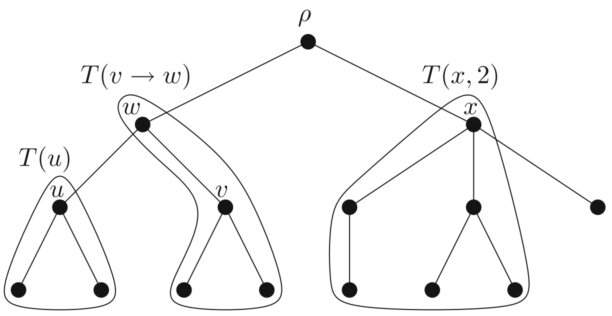

Fig. 1 Subtrees $T(u)$, $T(v \to w)$, and $T(x, 2)$

(T', b, s, a)-feasible sets For $\mathsf{b} \in \{\mathsf{in}, \mathsf{out}\}$, $s \in \{0, \ldots, n\}$, and $a \in \{1, \ldots, n\}$, we say $S \subseteq V(T')$ is *(T', b, s, a)-feasible* if $|S| = s$, the size of the active fragment in $\mathcal{F}(T', S)$ is a, there is no bad fragment in $\mathcal{F}(T', S)$, and $\mathsf{b} = \mathsf{in}$ if and only if the root of T' is in S.

Intuitively, a (T', b, s, a)-feasible set S is "almost safe." If A is the active fragment in $\mathcal{F}(T', S)$, then $S \setminus A$ is a safe set of $T'[V(T') \setminus A]$.

For $S \subseteq V(T')$, we set $\partial_{T'}^{\max}(S)$ and $\partial_{T'}^{\min}(S)$ to be the sizes of maximum and minimum fragments, respectively, adjacent to the active fragment in $\mathcal{F}(T', S)$. If there is no adjacent fragment, then we set $\partial_{T'}^{\max}(S) = -\infty$ and $\partial_{T'}^{\min}(S) = +\infty$.

DP table We construct a table with values $\mathsf{ps}(T', \mathsf{b}, s, a) \in \{0, \ldots, n\} \cup \{+\infty, -\infty\}$ for storing information of partial solutions, where $\mathsf{b} \in \{\mathsf{in}, \mathsf{out}\}$, $s \in \{0, \ldots, n\}$, and $a \in \{1, \ldots, n\}$, and T' is a subtree of T such that either $T' = T(u)$ for some $u \in V$, $T' = T(u \to v)$ for some $\{u, v\} \in E$, or $T' = T(u, i)$ for some $u \in V$ and $1 \le i \le |C_T(u)|$. The table entries will have the following values:

$$\mathsf{ps}(T', \mathsf{in}, s, a) = \begin{cases} +\infty & \text{if no } (T', \mathsf{in}, s, a)\text{-feasible set exists,} \\ \displaystyle\min_{(T', \mathsf{in}, s, a)\text{-feasible } S} \partial_{T'}^{\max}(S) & \text{otherwise,} \end{cases}$$

$$\mathsf{ps}(T', \mathsf{out}, s, a) = \begin{cases} -\infty & \text{if no } (T', \mathsf{out}, s, a)\text{-feasible set exists,} \\ \displaystyle\max_{(T', \mathsf{out}, s, a)\text{-feasible } S} \partial_{T'}^{\min}(S) & \text{otherwise.} \end{cases}$$

The definition of the table ps implies the following fact.

Lemma 2.2 $\mathsf{s}(T)$ *is the smallest s such that there is $a \in \{1, \ldots, n\}$ with $\mathsf{ps}(T, \mathsf{in}, s, a) \le a$ or $\mathsf{ps}(T, \mathsf{out}, s, a) \ge a$.*

Proof Assume that S is a safe set of T such that $|S| = s$ and the root is contained in S. Let A be the active fragment in $\mathcal{F}(T, S)$. Then, S is $(T, \mathsf{in}, s, |A|)$-feasible. Since A cannot be smaller than any adjacent fragment, we have $\partial_T^{\max}(S) \le |A|$. Hence $\mathsf{ps}(T, \mathsf{in}, s, |A|) \le |A|$ holds. By a similar argument, we can show that if the root is not in S, then $\mathsf{ps}(T, \mathsf{out}, s, |A|) \ge |A|$.

Conversely, assume that $\mathsf{ps}(T, \mathsf{in}, s, a) \le a$ for some $a \in \{1, \dots, n\}$. (The proof for the other case, where $\mathsf{ps}(T, \mathsf{out}, s, a) \ge a$, is similar.) Let S be a (T, in, s, a)-feasible set with $\partial_T^{\max}(S) = \mathsf{ps}(T, \mathsf{in}, s, a)$. Since there is no bad fragment in $\mathcal{F}(T, S)$ and the active fragment (of size a) is not smaller than the adjacent fragments (of size at most $\partial_T^{\max}(S) = \mathsf{ps}(T, \mathsf{in}, s, a) \le a$), all fragments included in S are not smaller than their adjacent fragments. This implies that S is a safe set of size s. □

By Lemma 2.2, after computing all entries $\mathsf{ps}(T', \mathsf{b}, s, a)$, we can compute $\mathsf{s}(T)$ in time $O(n^2)$. There are $O(n^3)$ tuples (T', b, s, a), and thus to prove the theorem, it suffices to show that each entry $\mathsf{ps}(T', \mathsf{b}, s, a)$ can be computed in time $O(n^2)$ assuming that the entries for all subtrees of T' are already computed.

We compute all entries $\mathsf{ps}(T', \mathsf{b}, s, a)$ in a bottom-up manner: We first compute the entries for $T(u)$ for each leaf u. We then repeat the following steps until none of them can be applied. (1) For each u such that the entries for $T(u)$ are already computed, we compute the entries for $T(u \to v)$, where v is the parent of u. (2) For each u such that the entries for $T(u, i-1)$ and $T(w_i \to u)$ are already computed, where w_i is the ith child of u, we compute the entries for $T(u, i)$.

Lemma 2.3 *For a leaf u of T, each table entry $\mathsf{ps}(T(u), \mathsf{b}, s, a)$ can be computed in constant time.*

Proof The set $\{u\}$ is the unique $(T(u), \mathsf{in}, 1, 1)$-feasible set. Since $\mathcal{F}(T(u), \{u\})$ contains no inactive fragment, we set $\mathsf{ps}(T(u), \mathsf{in}, 1, 1) = -\infty$. Similarly, the empty set is the unique $(T(u), \mathsf{out}, 0, 1)$-feasible set. We set $\mathsf{ps}(T(u), \mathsf{out}, 0, 1) = +\infty$. For the other tuples, there are no feasible sets. We set the values accordingly for them. Clearly, each entry can be computed in constant time. □

Lemma 2.4 *For a vertex u and its parent v in T, each table entry $\mathsf{ps}(T(u \to v), \mathsf{b}, s, a)$ can be computed in $O(n)$ time, using the table entries for the subtree $T(u)$.*

Proof We separate the proof into two cases: $a \ge 2$ and $a = 1$. If $a \ge 2$, then we can compute the table entry in constant time. If $a = 1$, we need $O(n)$ time.

Case 1 $a \ge 2$. In this case, for every $(T(u \to v), \mathsf{b}, s, a)$-feasible set S, u and v are in the active fragment of $\mathcal{F}(T(u \to v), S)$ since the root v of $T(u \to v)$ has the unique neighbor u.

Case 1-1: $\mathsf{b} = \mathsf{in}$. Let S be a $(T(u \to v), \mathsf{in}, s, a)$-feasible set that minimizes $\partial_{T(u\to v)}^{\max}(S)$. Observe that $S \setminus \{v\}$ is $(T(u), \mathsf{in}, s-1, a-1)$-feasible and that $\partial_{T(u\to v)}^{\max}(S) = \partial_{T(u)}^{\max}(S \setminus \{v\})$. We claim that $\partial_{T(u)}^{\max}(S\setminus\{v\}) = \mathsf{ps}(T(u), \mathsf{in}, s-1, a-1)$, and thus

$$\mathsf{ps}(T(u \to v), \mathsf{in}, s, a) = \mathsf{ps}(T(u), \mathsf{in}, s-1, a-1).$$

Suppose that some $(T(u), \mathsf{in}, s-1, a-1)$-feasible set Q satisfies $\partial_{T(u)}^{\max}(Q) < \partial_{T(u)}^{\max}(S\setminus\{v\})$. Now $Q \cup \{v\}$ is $(T(u \to v), \mathsf{in}, s, a)$-feasible. However, it holds that

$$\partial_{T(u\to v)}^{\max}(Q \cup \{v\}) = \partial_{T(u)}^{\max}(Q) < \partial_{T(u)}^{\max}(S\setminus\{v\}) = \partial_{T(u\to v)}^{\max}(S).$$

This contradicts the optimality of S.

Case 1-2 $\mathsf{b} = \mathsf{out}$. Let S be a $(T(u \to v), \mathsf{out}, s, a)$-feasible set that maximizes $\partial^{\min}_{T(u\to v)}(S)$. The set S is also $(T(u), \mathsf{out}, s, a-1)$-feasible and satisfies $\partial^{\min}_{T(u\to v)}(S) = \partial^{\min}_{T(u)}(S)$. We claim that $\partial^{\min}_{T(u)}(S) = \mathsf{ps}(T(u), \mathsf{in}, s, a-1)$, and thus

$$\mathsf{ps}(T(u \to v), \mathsf{out}, s, a) = \mathsf{ps}(T(u), \mathsf{out}, s, a-1).$$

Suppose that there is a $(T(u), \mathsf{in}, s, a-1)$-feasible set Q with $\partial^{\min}_{T(u)}(Q) > \partial^{\min}_{T(u)}(S)$. Since Q is also $(T(u \to v), \mathsf{out}, s, a)$-feasible, it holds that

$$\partial^{\min}_{T(u\to v)}(Q) = \partial^{\min}_{T(u)}(Q) > \partial^{\min}_{T(u)}(S) = \partial^{\min}_{T(u\to v)}(S).$$

This contradicts the optimality of S.

Case 2 $a = 1$. For every $(T(u \to v), \mathsf{b}, s, 1)$-feasible set S, the set $\{v\}$ is the active fragment, and the vertex u is in the unique fragment adjacent to the active fragment.

Case 2-1 $\mathsf{b} = \mathsf{in}$. Let S be a $(T(u \to v), \mathsf{in}, s, 1)$-feasible set. Then $S\backslash\{v\}$ is a $(T(u), \mathsf{out}, s-1, a')$-feasible set for some a'. Moreover, since $\mathcal{F}(T(u \to v), S)$ does not contain any bad fragment, $\partial^{\min}_{T(u)}(S\backslash\{v\}) \ge a'$. Thus we can set $\mathsf{ps}(T(u \to v), \mathsf{in}, s, 1)$ as follows:

$$\mathsf{ps}(T(u \to v), \mathsf{in}, s, 1) = \begin{cases} \min\{a' : \mathsf{ps}(T(u), \mathsf{out}, s-1, a') \ge a'\} & \text{if such } a' \text{ exists,} \\ +\infty & \text{otherwise.} \end{cases}$$

Case 2-2: $\mathsf{b} = \mathsf{out}$. Let S be a $(T(u \to v), \mathsf{out}, s, 1)$-feasible set. The set S is a $(T(u), \mathsf{in}, s, a')$-feasible set for some a'. Since $\mathcal{F}(T(u \to v), S)$ does not contain any bad fragment, $\partial^{\max}_{T(u)}(S) \le a'$. Thus we can set $\mathsf{ps}(T(u \to v), \mathsf{out}, s, 1)$ as follows:

$$\mathsf{ps}(T(u \to v), \mathsf{out}, s, 1) = \begin{cases} \max\{a' : \mathsf{ps}(T(u), \mathsf{in}, s, a') \le a'\} & \text{if there is such an } a', \\ -\infty & \text{otherwise.} \end{cases}$$

In both Cases 2-1 and 2-2, we can compute the entry $\mathsf{ps}(T(u \to v), \mathsf{b}, s, 1)$ in $O(n)$ time by looking up at most n table entries for the subtree $T(u)$. □

Lemma 2.5 *For a non-leaf vertex u with the children $w_1, \ldots, w_d$ and an integer i with $2 \le i \le d$, each table entry $\mathsf{ps}(T(u, i), \mathsf{b}, s, a)$ can be computed in $O(n^2)$ time, using the table entries for the subtrees $T(u, i-1)$ and $T(w_i \to u)$.*

Proof For the sake of simplicity, let $T_1 = T(u, i-1)$ and $T_2 = T(w_i \to u)$. Let S be a $(T(u, i), \mathsf{b}, s, a)$-feasible set and A be the active fragment in $\mathcal{F}(T(u, i), S)$. For $j \in \{1, 2\}$, let $S_j = S \cap V(T_j)$ and $A_j = A \cap V(T_j)$. Observe that S_j is a $(T_j, \mathsf{b}, |S_j|, |A_j|)$-feasible set. If $\mathsf{b} = \mathsf{in}$, then $S_1 \cap S_2 = \{u\}$; otherwise $S_1 \cap S_2 = \emptyset$. Thus $|S_1| + |S_2| = |S| + 1$ if $\mathsf{b} = \mathsf{in}$, and $|S_1| + |S_2| = |S|$ otherwise. Similarly, since $A_1 \cap A_2 = \{u\}$, it holds that $|A_1| + |A_2| = |A| + 1$. Therefore, we can set the table entries as follows:

$$\mathsf{ps}(T(u,i),\mathsf{in},s,a) = \min_{\substack{s_1+s_2=s+1\\ a_1+a_2=a+1}} \max\{\mathsf{ps}(T_1,\mathsf{in},s_1,a_1),\mathsf{ps}(T_2,\mathsf{in},s_2,a_2)\},$$

$$\mathsf{ps}(T(u,i),\mathsf{out},s,a) = \max_{\substack{s_1+s_2=s\\ a_1+a_2=a+1}} \min\{\mathsf{ps}(T_1,\mathsf{out},s_1,a_1),\mathsf{ps}(T_2,\mathsf{out},s_2,a_2)\}.$$

In both cases, we can compute the entry $\mathsf{ps}(T(u,i),\mathsf{b},s,a)$ in $O(n^2)$ time since there are $O(n)$ possibilities for each (s_1,s_2) and (a_1,a_2). □

3 Safe sets in graphs of bounded treewidth

In this section, we show that for any fixed constant k, a minimum safe set and a minimum connected safe set of a graph of treewidth at most k can be found in $O(n^{5k+8})$ time.

Basically, the algorithm in this section is a generalization of the one in the previous section. The most crucial difference is that here we may have many active fragments, and each active fragment may have many vertices adjacent to the "outside." This makes the algorithm much more complicated and slow.

A *tree decomposition* of a graph $G = (V, E)$ is a pair $(\{X_p : p \in I\}, T)$ such that each X_p, called a *bag*, is a subset of V, and T is a tree with $V(T) = I$ such that

- for each $v \in V$, there is $p \in I$ with $v \in X_p$;
- for each $\{u, v\} \in E$, there is $p \in I$ with $u, v \in X_p$;
- for $p, q, r \in I$, if q is on the p–r path in T, then $X_p \cap X_r \subseteq X_q$.

The *width* of a tree decomposition is the size of a largest bag minus 1. The *treewidth* of a graph, denoted by $\mathsf{tw}(G)$, is the minimum width over all tree decompositions of G.

A tree decomposition $(\{X_p : p \in I\}, T)$ is *nice* if

- T is a rooted tree in which every node has at most two children;
- if a node p has two children q, r, then $X_p = X_q = X_r$ (such a node p is a *join node*);
- if a node p has only one child q, then either
  - $X_p = X_q \cup \{v\}$ for some $v \notin X_q$ (p is a *introduce node*), or
  - $X_p = X_q \setminus \{v\}$ for some $v \in X_q$ (q is a *forget node*);
- if a node p is a leaf, then $X_p = \{v\}$ for some $v \in V$ (p is a *leaf node*).

Theorem 3.1 *Let k be a fixed constant. For an n-vertex graph of treewidth at most k, a (connected) safe set of minimum size can be found in time* $O(n^{5k+8})$.

Proof We only show that $\mathsf{s}(G)$ and $\mathsf{cs}(G)$ can be computed in the claimed running time. It is straightforward to modify the dynamic program below for computing an actual set in the same running time.

Let $G = (V, E)$ be a graph of treewidth at most k. We compute a nice tree decomposition $(\{X_p : p \in I\}, T)$ with at most $4n$ nodes. It can be done in $O(n)$

time (Bodlaender 1996; Kloks 1994). For each $p \in I$, let $V_p = X_p \cup \bigcup_q X_q$, where q runs through all descendants of p in T.

Fragments For a node p and a vertex set $S \subseteq V_p$, a *fragment* is a component in $G[S]$ or $G[V_p \setminus S]$. We denote the set of fragments with respect to p and S by $\mathcal{F}(p, S)$. A fragment $F \in \mathcal{F}(p, S)$ is *active* if $F \cap X_p \neq \emptyset$, and it is *inactive* otherwise. Two fragments in $\mathcal{F}(p, S)$ are *adjacent* if there is an edge of $G[V_p]$ between them. A fragment F is *bad* if it is inactive, $F \subseteq S$, and there is another inactive fragment F' adjacent to F with $|F| < |F'|$.

DP table For storing information of partial solutions, we construct a table with values $\mathsf{ps}(p, s, \mathcal{A}, \beta, \gamma, \phi, \psi) \in \{\mathsf{t}, \mathsf{f}\}$ with indices $p \in I$, $s \in \{0, \dots, n\}$, a partition $\mathcal{A}$ of X_p, $\beta \colon \mathcal{A} \to \{1, \dots, n\}$, $\gamma \colon \mathcal{A} \to \{1, \dots, n\} \cup \{\pm\infty\}$, $\phi \colon \mathcal{A} \to \{\mathsf{t}, \mathsf{f}\}$, and $\psi \colon \binom{\mathcal{A}}{2} \to \{\mathsf{t}, \mathsf{f}\}$. We set

$$\mathsf{ps}(p, s, \mathcal{A}, \beta, \gamma, \phi, \psi) = \mathsf{t}$$

if and only if there exists a set $S \subseteq V_p$ of size s with the following conditions:

- there is no bad fragment in $\mathcal{F}(p, S)$,
- for each active fragment F in $\mathcal{F}(p, S)$,
  - there is a unique element $A_F \in \mathcal{A}$ such that $A_F = F \cap X_p$,
  - $\beta(A_F) = |F|$,
  - $\phi(A_F) = \mathsf{t}$ if and only if $F \subseteq S$,
  - if $F \subseteq S$, then $\gamma(A_F)$ is the size of a *maximum* inactive fragment adjacent to F (if no such fragment exists, we set $\gamma(A_i) = -\infty$),
  - if $F \not\subseteq S$, then $\gamma(A_F)$ is the size of a *minimum* inactive fragment adjacent to F (if no such fragment exists, we set $\gamma(A_i) = +\infty$),
- for two active fragments F, F' in $\mathcal{F}(p, S)$, $\psi(\{A_F, A_{F'}\}) = \mathsf{t}$ if and only if F and F' are adjacent, where $A_F = F \cap X_p$ and $A_{F'} = F' \cap X_p$.[1]

Let ρ be the root of T. The definition of the table ps implies the following fact.

Observation 3.2 $\mathsf{s}(G)$ is the smallest s with $\mathsf{ps}(\rho, s, \mathcal{A}, \beta, \gamma, \phi, \psi) = \mathsf{t}$ for some $\mathcal{A}$, β, γ, ϕ, and ψ such that $\beta(A) \geq \gamma(A)$ for each $A \in \mathcal{A}$ with $\phi(A) = \mathsf{t}$, $\beta(A) \leq \gamma(A)$ for each $A \in \mathcal{A}$ with $\phi(A) = \mathsf{f}$, and $\beta(A) \geq \beta(A')$ for any $A, A' \in \mathcal{A}$ with $\phi(A) = \mathsf{t}$ and $\psi(A, A') = \mathsf{t}$.

For computing $\mathsf{cs}(G)$, we need to compute additional information for each tuple $(p, s, \mathcal{A}, \beta, \gamma, \phi, \psi)$. For $A \in \mathcal{A}$, let $\beta'(A)$ be the size of the fragment in $\mathcal{F}(\rho, S)$ that is a superset of the fragment $F_A \supseteq A$ in $\mathcal{F}(p, S)$. If $A \subseteq S$, then $\beta'(A) = \beta(A)$; otherwise $\beta'(A) = |C_A \setminus X_p| + \sum_{A' \in \mathcal{A},\ A' \subseteq C_A} \beta(A')$, where C_A is the component in $G[(V \setminus V_p) \cup (X_p \setminus S)]$ that includes A. We can compute $\beta'(A)$ for all $A \in \mathcal{A}$ in time $O(n)$ by running a breadth-first search from $X_p \setminus S$.

[1] In the following, we (ab)use simpler notation $\psi(A_F, A_{F'})$ instead of $\psi(\{A_F, A_{F'}\})$.

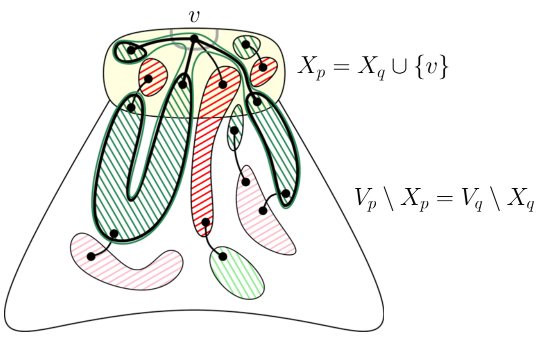

Fig. 2 Introducing a vertex v. Fragments are colored in such a way that the fragments in S are colored with one color and the fragments not in S are colored with the other color (Color figure online)

Observation 3.3 $\mathsf{cs}(G)$ is the smallest s with $\mathsf{ps}(p, s, \mathcal{A}, \beta, \gamma, \phi, \psi) = \mathsf{t}$ for some p, $\mathcal{A}$, β, γ, ϕ, and ψ such that $\beta(A) \ge \gamma(A)$ for each $A \in \mathcal{A}$ with $\phi(A) = \mathsf{t}$, $\beta(A) \le \gamma(A)$ for each $A \in \mathcal{A}$ with $\phi(A) = \mathsf{f}$, and $\beta(A) \ge \beta'(A')$ for any $A, A' \in \mathcal{A}$ with $\phi(A) = \mathsf{t}$ and $\psi(A, A') = \mathsf{t}$.

By Observations 3.2 and 3.3, provided that all entries $\mathsf{ps}(p, s, \mathcal{A}, \beta, \gamma, \phi, \psi)$ are computed in advance, we can compute $\mathsf{s}(G)$ and $\mathsf{cs}(G)$ by spending time $O(1)$ and $O(n)$, respectively, for each tuple. We compute all entries $\mathsf{ps}(p, s, \mathcal{A}, \beta, \gamma, \phi, \psi)$ by a bottom-up dynamic program.

There are at most $4n \cdot (n+1) \cdot (k+1)^{k+1} \cdot n^{k+1} \cdot (n+2)^{k+1} \cdot 2^{k+1} \cdot 2^{(k+1)k/2} = O(n^{2k+4})$ tuples $(p, s, \mathcal{A}, \beta, \gamma, \phi, \psi)$. Now, to prove the theorem, it suffices to show that each entry $\mathsf{ps}(p, s, \mathcal{A}, \beta, \gamma, \phi, \psi)$ can be computed in time $O(n^{3k+4})$ assuming that the entries for the children of p are already computed.

Leaf nodes For a leaf node p with $X_p = \{v\}$, $\mathsf{ps}(p, s, \mathcal{A}, \beta, \gamma, \phi, \psi) = \mathsf{t}$ if and only if the following conditions are satisfied: $\mathcal{A} = \{\{v\}\}$; $s = 1$ if $\phi(\{v\}) = \mathsf{t}$, $s = 0$ otherwise; $\beta(\{v\}) = 1$; $\gamma(\{v\}) = -\infty$ if $\phi(\{v\}) = \mathsf{t}$, $\gamma(\{v\}) = +\infty$ otherwise; and ψ is the empty function. The conditions can be checked in $O(1)$ time.

Introduce nodes Let p be an introduce node with the child q and $X_p = X_q \cup \{v\}$. Observe from the definition of tree decompositions that v has no neighbor in $V_p \setminus X_p$. Let $S \subseteq V_p$ and $S' = S \cap V_q$. If $v \in S$, then $|S'| = |S| - 1$; otherwise, $|S'| = |S|$. Now $\mathcal{F}(p, S)$ and $\mathcal{F}(q, S')$ have the same set of the inactive fragments, and thus $\mathcal{F}(p, S)$ has a bad fragment if and only if so does $\mathcal{F}(q, S')$.Let $\mathcal{J}(v)$ be the set of active fragments in $\mathcal{F}(q, S')$ that contain a neighbor of v and are contained in S' if $v \in S$ and in $V_q \setminus S'$ otherwise. By introducing v, we merge the fragments in $\mathcal{J}(v)$ with v into a single active fragment in $\mathcal{F}(p, S)$. (See Fig. 2.) Thus, if we set $F_v = \{v\} \cup \bigcup_{F \in \mathcal{J}(v)} F$, then $\mathcal{F}(p, S) = \{F_v\} \cup (\mathcal{F}(q, S') \setminus \mathcal{J}(v))$ and $|F_v| = 1 + |\bigcup_{F \in \mathcal{J}(v)} F|$. The fragment F_v is adjacent to an inactive fragment F if and only at least one fragment in $\mathcal{J}(v)$ is adjacent to F. Other active fragments have the same adjacent inactive fragments in $\mathcal{F}(p, S)$ and $\mathcal{F}(q, S')$. The fragment F_v is adjacent to another active fragment F in $\mathcal{F}(p, S)$ if and only if either there is an edge between v and F or F is adjacent to an active fragment F' in $\mathcal{F}(q, S')$ that is a subset of F_v.

The next proposition follows.

Proposition 3.4 *For an introduce node p with the child q and $X_p = X_q \cup \{v\}$, the entry $\mathsf{ps}(p, s, \mathcal{A}, \beta, \gamma, \phi, \psi) = \mathsf{t}$ if and only if $\mathsf{ps}(q, s', \mathcal{A}', \beta', \gamma', \phi', \psi') = \mathsf{t}$ for some s', $\mathcal{A}'$, β', γ', ϕ', and ψ' with the following conditions satisfied, where A_v is the element of $\mathcal{A}$ that contains v and $\mathcal{A}'_v = \{A \in \mathcal{A}' : A \subseteq A_v\}$:*

- *$s = s' + 1$ if $\phi(A_v) = \mathsf{t}$, $s = s'$ otherwise;*
- *$\mathcal{A} \backslash \{A_v\} = \mathcal{A}' \backslash \mathcal{A}'_v$;*
- *for each $A \in \mathcal{A}' \backslash \mathcal{A}'_v$, it holds that $\beta(A) = \beta'(A)$, $\gamma(A) = \gamma'(A)$, and $\phi(A) = \phi'(A)$;*
- *$\beta(A_v) = 1 + \sum_{A \in \mathcal{A}'_v} \beta'(A)$;*
- *$\gamma(A_v) = \max_{A \in \mathcal{A}'_v} \gamma'(A)$ if $\phi(A_v) = \mathsf{t}$, $\gamma(A_v) = \min_{A \in \mathcal{A}'_v} \gamma'(A)$ otherwise;*
- *$\phi(A_v) = \phi'(A)$ for each $A \in \mathcal{A}'_v$;*
- *$\psi(A, A') = \psi'(A, A')$ for $A, A' \in \mathcal{A}' \backslash \mathcal{A}'_v$, $\psi(A_v, A') = (\exists u \in A', \{u, v\} \in E) \vee \bigvee_{A \in \mathcal{A}'_v} \psi'(A, A')$ for $A' \in \mathcal{A}' \backslash \mathcal{A}'_v$;*

There are at most $|A_v|^{|A_v|} = O(1)$ candidates for $\mathcal{A}'$, at most $n^{|A_v|} = O(n^{k+1})$ for β', at most $(n+2)^{|A_v|} = O(n^{k+1})$ for γ', and at most $2^{|\mathcal{A}'_v| \cdot |\mathcal{A}' \backslash \mathcal{A}'_v|} = O(1)$ for γ'. Hence there are only $O(n^{2k+2})$ candidates of tuples $(q, s', \mathcal{A}', \beta', \gamma', \phi', \psi')$ satisfying the conditions in Proposition 3.4. We can check in $O(1)$ time whether a candidate satisfies the conditions, and thus we can compute the entry $\mathsf{ps}(p, s, \mathcal{A}, \beta, \gamma, \phi, \psi)$ in $O(n^{2k+2})$ time.

Forget nodes Let p be a forget node with the child q and $X_p = X_q \backslash \{v\}$. First note that $V_p = V_q$, and thus for any $S \subseteq V_p$, the sets of fragments $\mathcal{F}(p, S)$ and $\mathcal{F}(q, S)$ are the same. For a set $S \subseteq V_p$, let F_v be the fragment containing v. Since $v \in X_q$, F_v is active in $\mathcal{F}(q, S)$. On the other hand, F_v is inactive in $\mathcal{F}(p, S)$ if and only if $F_v \cap X_q = \{v\}$, that is, F_v becomes inactive by forgetting v.

If F_v is active also in $\mathcal{F}(p, S)$, then the sets of active fragments in $\mathcal{F}(p, S)$ and $\mathcal{F}(q, S)$ are the same. In such a case, we do not have to update any information about the fragments.

If $F_v \cap X_q = \{v\}$, then F_v is inactive in $\mathcal{F}(p, S)$, see Fig. 3. For an active fragment F in $\mathcal{F}(p, S)$, F_v may be a largest or smallest inactive fragment adjacent to it. In such a case, we may have to update the γ value of $A_F = F \cap X_p$. Let F be an inactive fragment in $\mathcal{F}(p, S)$ that is adjacent to F_v. If $F_v \subseteq S$ and $|F_v| < |F|$, then F_v is bad. If $F_v \nsubseteq S$ and $|F_v| > |F|$, then F is bad.

From the observations above, we can conclude the following rule for updating the table for forget nodes.

Proposition 3.5 *For a forget node p with the child q and $X_p = X_q \backslash \{v\}$, the table entry $\mathsf{ps}(p, s, \mathcal{A}, \beta, \gamma, \phi, \psi) = \mathsf{t}$ if and only if either (1) $\mathsf{ps}(q, s, \mathcal{A} \backslash \{A\} \cup \{A \cup \{v\}\}, \beta, \gamma, \phi, \psi) = \mathsf{t}$ for some $A \in \mathcal{A}$, or (2) $\mathsf{ps}(q, s, \mathcal{A} \cup \{\{v\}\}, \beta', \gamma', \phi', \psi') = \mathsf{t}$ for some β', γ', ϕ', and ψ' with the following conditions satisfied:*

- *$\beta'(\{v\}) \geq \gamma'(\{v\})$ if $\phi'(\{v\}) = \mathsf{t}$; $\beta'(\{v\}) \leq \gamma'(\{v\})$ if $\phi'(\{v\}) = \mathsf{f}$;*
- *$\beta = \beta'|_{\mathcal{A}}$, $\phi = \phi'|_{\mathcal{A}}$, $\psi = \psi'|_{\binom{\mathcal{A}}{2}}$*
- *if $\psi'(A, \{v\}) = \mathsf{f}$ for $A \in \mathcal{A}$, then $\gamma(A) = \gamma'(A)$;*
- *if $\psi'(A, \{v\}) = \mathsf{t}$ for $A \in \mathcal{A}$, then $\gamma(A) = \max\{\gamma'(A), \beta'(\{v\})\}$ if $\phi(A) = \mathsf{t}$ and $\gamma(A) = \min\{\gamma'(A), \beta'(\{v\})\}$ if $\phi(A) = \mathsf{f}$.*

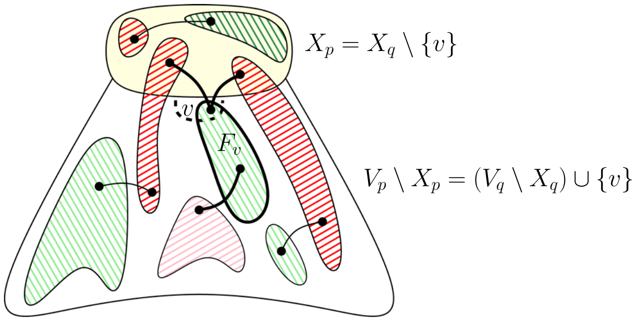
$X_p = X_q \setminus \{v\}$
v
F_v
$V_p \setminus X_p = (V_q \setminus X_q) \cup \{v\}$

Fig. 3 Forgetting a vertex v

For the first case in Proposition 3.5, there are only $|\mathcal{A}| = O(1)$ candidates of tuples. For the second case, there are options of function values only for the arguments involving $\{v\}$. Thus there are only $O(n^2)$ candidates of tuples. The conditions of each candidate can be checked in $O(1)$ time, and thus we can compute the entry $\mathsf{ps}(p, s, \alpha, \beta, \gamma, \phi, \psi)$ in $O(n^2)$ time.

Join nodes Let p be a join node with the children q and r with $X_p = X_q = X_r$. From the definition of tree decompositions, we can see that $V_q \cap V_r = X_p$ and there is no edge between $V_q \setminus X_p$ and $V_r \setminus X_p$. Let $S \subseteq V_p$, $S' = S \cap V_q$, and $S'' = S \cap V_r$.

Observe that an inactive fragment in $\mathcal{F}(p, S)$ is an inactive fragment in exactly one of $\mathcal{F}(q, S')$ and $\mathcal{F}(r, S'')$, and there is no other inactive fragment in $\mathcal{F}(q, S')$ or $\mathcal{F}(r, S'')$. On the other hand, an active fragment in $\mathcal{F}(p, S)$ may be split into several active fragments in $\mathcal{F}(q, S')$ and $\mathcal{F}(r, S'')$, see Fig. 4. Let $H = (\mathcal{F}', \mathcal{F}''; \mathcal{E})$ be the bipartite graph such that $\mathcal{F}'$ is the set of active fragments in $\mathcal{F}(q, S')$, $\mathcal{F}''$ is the set of active fragments in $\mathcal{F}(r, S'')$, and $\{F', F''\} \in \mathcal{E}$ for $F' \in \mathcal{F}'$ and $F'' \in \mathcal{F}''$ if and only if $F' \cap F'' \neq \emptyset$. Then, each active fragment F in $\mathcal{F}(p, S)$ corresponds to a unique component C of H. That is, $F = \bigcup_{F' \in C} F'$.

By considering the sizes and adjacency among the merged active fragments and the unmodified inactive fragments, we can update a table entry for a join node as follows.

Proposition 3.6 *For a join node p with the children q and r, the table entry $\mathsf{ps}(p, s, \mathcal{A}, \beta, \gamma, \phi, \psi) = \mathsf{t}$ if and only if $\mathsf{ps}(q, s', \mathcal{A}', \beta', \gamma', \phi', \psi') = \mathsf{t}$ and $\mathsf{ps}(r, s'', \mathcal{A}'', \beta'', \gamma'', \phi'', \psi'') = \mathsf{t}$ for some s', s'', $\mathcal{A}'$, $\mathcal{A}''$, β', β'', γ'', ϕ', ϕ'', ψ', and ψ'' with the following conditions satisfied, where $\mathcal{A}'_A = \{A' \in \mathcal{A}' : A' \subseteq A\}$ and $\mathcal{A}''_A = \{A'' \in \mathcal{A}'' : A'' \subseteq A\}$:*

- *$s = s' + s'' - \sum_{A \in \mathcal{A},\ \phi(A) = \mathsf{t}} |A|$;*
- *$\mathcal{A}'$ and $\mathcal{A}''$ are refinements of $\mathcal{A}$;*
- *for any $A \in \mathcal{A}$, any $A' \in \mathcal{A}'_A$, and any $A'' \in \mathcal{A}''_A$, there is a family $\{A_1, \ldots, A_t\} \subseteq \mathcal{A}'_A \cup \mathcal{A}''_A$ such that $A_1 = A'$, $A_t = A''$, and $A_i \cap A_{i+1} \neq \emptyset$ for $1 \leq i < t$;*
- *for $A \in \mathcal{A}$, $\beta(A) = \sum_{A' \in \mathcal{A}'_A} \beta'(A') + \sum_{A'' \in \mathcal{A}''_A} \beta''(A'') - |A|$;*
- *$\phi(A) = \phi'(A')$ for all $A' \in \mathcal{A}'_A$ and $\phi(A) = \phi''(A'')$ for all $A'' \in \mathcal{A}''_A$;*
- *for $A \in \mathcal{A}$, if $\phi(A) = \mathsf{t}$, then $\gamma(A) = \max\left(\{\gamma'(A') : A' \in \mathcal{A}'_A\} \cup \{\gamma''(A'') : A'' \in \mathcal{A}''_A\}\right)$, otherwise $\gamma(A) = \min\left(\{\gamma'(A') : A' \in \mathcal{A}'_A\} \cup \{\gamma''(A'') : A'' \in \mathcal{A}''_A\}\right)$;*

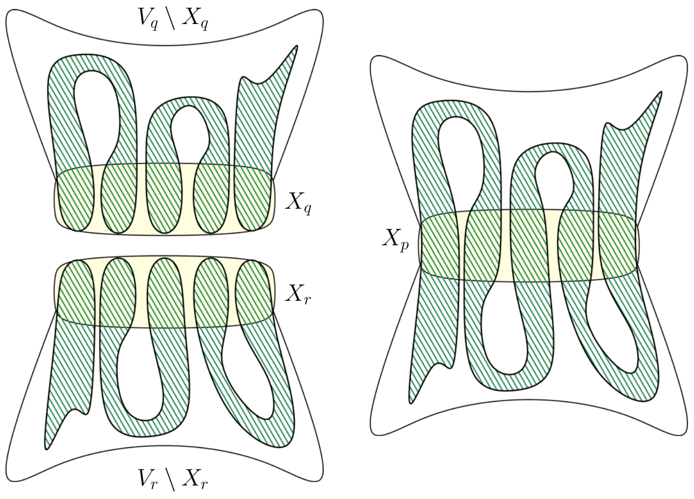

Fig. 4 Joining at bags X_q and X_r. (An active fragment in $\mathcal{F}(p, S)$ is depicted.)

– *for all* $A, B \in \mathcal{A}$,

$$\psi(A, B) = \left(\bigvee_{A' \in \mathcal{A}'_A,\ B' \in \mathcal{A}'_B} \psi'(A', B') \right) \vee \left(\bigvee_{A'' \in \mathcal{A}''_A,\ B'' \in \mathcal{A}''_B} \psi''(A'', B'') \right).$$

Under the conditions of Proposition 3.6. there are $O(n)$ possible options for (s', s''), $O(n^{k+1})$ for (β', β''), $O(n^{2k+2})$ for (γ', γ''), and $O(1)$ for the remaining parameters. Thus we can compute the entry $\mathsf{ps}(p, s, \alpha, \beta, \gamma, \phi, \psi)$ in $O(n^{3k+4})$ time because the validity of each tuple can be checked in $O(1)$ time. □

Note that our algorithm for graphs of treewidth at most k runs in $n^{O(k)}$ time. Such an algorithm is called an XP algorithm, and an FPT algorithm with running time $f(k) \cdot n^c$ is more preferable, where f is an arbitrary computable function and c is a fixed constant. It would be interesting if one can show that such an algorithm exists (or does not exist under some complexity assumption).

3.1 Weighted graphs

For a vertex-weighted graph $G = (V, E)$ with a weight function $w\colon V \to \mathbb{Z}^+$, a set $S \subseteq V$ is a *weighted safe set* of weight $\sum_{s \in S} w(s)$ if for each component C of $G[S]$ and each component D of $G[V \setminus S]$ with an edge between C and D, it holds that $w(C) \geq w(D)$. Bapat et al. (2016) showed that finding a minimum (connected) weighted safe set is weakly NP-hard even for stars. Here we note that the problem is not strongly NP-hard unless P = NP. Let $W = \sum_{v \in V} w(v)$. Our dynamic program above works for the weighted version if we extend the ranges of parameters s, β, and γ by including $\{1, \ldots, W\}$. The running time becomes polynomial in W.

Theorem 3.7 *For a vertex weighted graph of bounded treewidth, a weighted (connected) safe set of the minimum weight can be found in pseudo-polynomial time.*

4 Safe sets in interval graphs

In this section, we present a polynomial-time algorithm for finding a minimum safe set and a minimum connected safe set in an interval graph.

A graph is an *interval graph* if it can be represented as the intersection graph of intervals on a line. Given a graph, one can determine in linear time whether the graph is an interval graph, and if so, find a corresponding interval representation in the same running time (Booth and Lueker 1976).

Theorem 4.1 *For an n-vertex interval graph, a minimum safe set and a minimum connected safe set can be found in time $O(n^8)$.*

Proof Let $G = (V, E)$ be a given interval graph. As we can deal with each component of G separately, we assume that G is connected. The algorithm uses the dynamic programming technique on an interval representation of G. We assume that its vertices (i.e. intervals) $v_1, \dots, v_n$ are ordered increasingly according to their left ends, and write $X_i = \{v_1, \dots, v_i\}$.

At each step i of the algorithm, we want to store all subsets $S \subseteq X_i$ which can potentially be completed (with vertices from $V \setminus X_i$) into a safe set. The number of such sets can be exponential: we thus define a notion of *signature*, and store the signatures of the sets instead of storing the sets themselves. The cost of this storage is bounded by the number of possible signatures, which is polynomial in n.

We will then prove that all possible signatures of sets at step i can be deduced from the set of signatures at step $i-1$. The cardinality of a minimum safe set (and a minimum connected safe set) can finally be deduced from the set of signatures stored during the last step. We can easily modify the algorithm so that it also outputs a minimum set.

We define the *signature of S at step i* as the 8-tuple that consists of the following items (see Fig. 5):

1. The size of S.
2. The vertex v_r^S of S with the most neighbors in $V \setminus X_i$.
3. The vertex $v_r^{\bar{S}}$ of $\bar{S} := X_i \setminus S$ with the most neighbors in $V \setminus X_i$.
4. The size of S_r (the rightmost component of S).
5. The size of $\bar{S}_r$ (the rightmost component of $\bar{S}$).
6. The largest size of a component of $\bar{S} \setminus \bar{S}_r$ adjacent with S_r.
7. The smallest size of a component of $S \setminus S_r$ adjacent with $\bar{S}_r$.
8. A Boolean value indicating whether S is connected.

Assuming that we know the signature of a set S at step i, we show how to obtain the signature at step $i+1$ of (a) $S' = S$ and (b) $S' = S \cup \{v_{i+1}\}$. With this procedure, all signatures of step $i+1$ can be obtained from all signatures at step i.

1. The size of S' (at step i: $|S|$).
   (a) $|S|$.

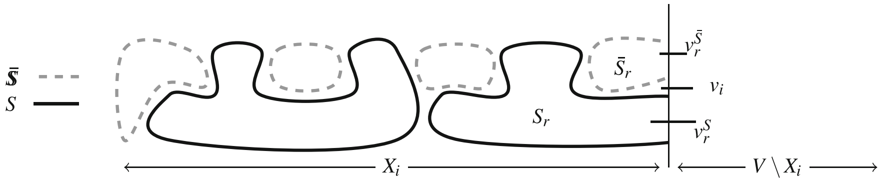

Fig. 5 The dynamic programming technique applied to an interval representation

   (b) $|S| + 1$.
2. The vertex of S' with the most neighbors in $V \backslash X_{i+1}$ (at step i: v_r^S).
   (a) v_r^S.
   (b) The one of v_r^S and v_{i+1} which has the most neighbors in $V \backslash X_{i+1}$.
3. The vertex of $\bar{S}' := X_{i+1} \backslash S'$ with the most neighbors in $V \backslash X_{i+1}$ (at step i: $v_r^{\bar{S}}$).
   (a) The one of $v_r^{\bar{S}}$ and v_{i+1} which has the most neighbors in $V \backslash X_{i+1}$.
   (b) $v_r^{\bar{S}}$.
4. The size of the rightmost component S'_r of S' (at step i: $|S_r|$).
   (a) $|S_r|$.
   (b) $|S_r| + 1$ if v_{i+1} and v_r^S are adjacent, and 1 otherwise (new component). In the latter case, we discard the signature if $|S_r|$ is strictly smaller than the largest size of a component of $\bar{S} \backslash \bar{S}_r$ adjacent with S_r at step i.
5. The size of the rightmost component $\bar{S}'_r$ of $\bar{S}'$ (at step i: $|\bar{S}_r|$).
   (a) 1 if v_{i+1} and $v_r^{\bar{S}}$ are not adjacent (new component), and $|\bar{S}_r| + 1$ otherwise. In the latter case, we discard the signature if $|\bar{S}_r|$ is strictly larger than the smallest size of a component of $S \backslash S_r$ adjacent with $\bar{S}_r$ at step i.
   (b) $|\bar{S}_r|$.
6. The largest size of a component of $\bar{S}' \backslash \bar{S}'_r$ adjacent with S'_r (at step i: c).
   (a) c if no new component of $\bar{S}'$ was created (see 5.), and $\max\{c, |\bar{S}'_r|\}$ otherwise.
   (b) c if no new component of S' was created (see 4.), and $-\infty$ otherwise.
7. The smallest size of a component of $S' \backslash S'_r$ adjacent with $\bar{S}'_r$ (at step i: c).
   (a) c if no new component of $\bar{S}'$ was created (see 5.), and $+\infty$ otherwise.
   (b) c if no new component of S' was created (see 4.), and $\min\{c, |S'_r|\}$ otherwise.
8. A Boolean variable indicating whether S' is connected (at step i: b).
   (a) b
   (b) t if $|S| = 0$, b if v_{i+1} and v_r^S are adjacent, and f otherwise.

When all signatures at step n have been computed, we use the additional information that S and $\bar{S}$ cannot be further extended to discard the remaining signatures corresponding to non-safe sets. That is, we discard a signature if $|S_r| < |\bar{S}_r|$, or $|S_r|$ is strictly smaller than the largest size of a component of $\bar{S} \backslash \bar{S}_r$ adjacent to it, or $|\bar{S}_r| + 1$ is strictly larger than the smallest size of a component of $S \backslash S_r$ adjacent to it.

The minimum sizes of a safe set and a connected safe set can be obtained from the remaining signatures. For each step i, there are $O(n^7)$ signatures. From a signature for step i, we can compute the corresponding signature for step $i + 1$ in $O(1)$ time. Therefore, the total running time is $O(n^8)$. □

5 Fixed-parameter tractability

In this section, we show that the problems of finding a safe set and a connected safe set of size at most s is fixed-parameter tractable when the solution size s is the parameter. For the standard concepts in parameterized complexity, see the recent textbook (Cygan et al. 2015).

We first show that graphs with small safe sets have small treewidth. We then show that for any fixed constants s the property of having a (connected) safe set of size at most s can be expressed in the monadic second-order logic on graphs. Then we use the well-known theorems by Bodlaender (1996) and Courcelle (1992) to obtain an FPT algorithm that depends only linearly on the input size.

Lemma 5.1 *Let $G = (V, E)$ be a connected graph. If $\mathsf{tw}(G) \geq s^2 - 1$, then $\mathsf{s}(G) \geq s$.*

Proof It is known that every graph G has a path of $\mathsf{tw}(G) + 1$ vertices as a subgraph (Bodlaender 1993). Thus $\mathsf{tw}(G) \geq s^2 - 1$ implies that G has a path of s^2 vertices as a subgraph.

Let P be a path of s^2 vertices in G, and let $S \subseteq V$ be an arbitrary set of size less than s. By the pigeon-hole principle, there is a subpath Q of P such that $|Q| \geq s$ and $S \cap V(Q) = \emptyset$. Hence there is a component B of $G[V \setminus S]$ with $V(Q) \subseteq B$. Since G is connected there is a component A of $G[S]$ adjacent to B. Now we have $|A| \leq |S| < s \leq |Q| \leq |B|$, which implies that S is not a safe set. □

The syntax of the *monadic second-order logic of graphs* (MS_2) includes (i) the logical connectives $\vee$, $\wedge$, $\neg$, $\Leftrightarrow$, $\Rightarrow$, (ii) variables for vertices, edges, vertex sets, and edge sets, (iii) the quantifiers $\forall$ and $\exists$ applicable to these variables, and (iv) the following binary relations:

- $v \in U$ for a vertex variable v and a vertex set variable U;
- $e \in F$ for an edge variable e and an edge set variable F;
- $\mathsf{inc}(e, v)$ for an edge variable e and a vertex variable v, where the interpretation is that e is incident with v;
- equality of variables.

Lemma 5.2 *For a fixed constant s, the property of having a safe set of size at most s can be expressed in MS_2.*

Proof For two sets U and W of vertices, we can express the adjacency between them as:

$$\mathsf{adjacent}(U, W) := \exists e \in E, u \in U, w \in W \ (\mathsf{inc}(e, u) \wedge \mathsf{inc}(e, w)).$$

For a set U of vertices, the following formula implies that the subgraph induced by U is connected:

$$\mathsf{connected}(U) := \forall X \subseteq U \ (X \neq \emptyset \wedge X \neq U) \implies \mathsf{adjacent}(X, U \setminus X).$$

By testing maximality also, the following formula implies that W is the vertex set of a component of the subgraph induced by U.

$$\begin{aligned}\mathsf{component}(U, W) := (W \subseteq U) \wedge \mathsf{connected}(W)\\ \wedge \neg(\exists u \in U, \mathsf{connected}(W \cup \{u\})).\end{aligned}$$

To express the formula that a vertex set is a safe set, we need to compare the sizes of two sets. In general, MS_2 is not capable of measuring the size of a set. However, for a fixed constant s, we can express the relation $|S| = s$ (and thus $|S| \le s$ and $|S| \ge s$ also) for any set S in MS_2. For example, $|S| = 2$ is equivalent to the formula $\exists v_1, v_2 \in S\ (v_1 \ne v_2 \wedge \neg(\exists v_3 \in S\ (v_3 \ne v_1 \wedge v_3 \ne v_2)))$. This obviously extends to $|S| = s$ for any fixed s. Using such a formula, we can express the relation $|Q| \ge |W|$ for a set Q with $|Q| \le s$ and for any set W as follows: $\bigvee_{s' \le s}(|Q| \ge s' \wedge |W| \le s' - 1)$. Now a set S is a safe set of size at most s if and only if it satisfies the following formula:

$$\begin{aligned}s\text{-}\mathsf{safe}(S) := (|S| \le s) \wedge (\forall Q \subseteq S, \forall W \subseteq V \setminus S, (\mathsf{component}(S, Q) \wedge\\ \mathsf{component}(V \setminus S, W) \wedge \mathsf{adjacent}(Q, W) \implies |Q| \ge |W|)).\end{aligned}$$

This implies the lemma as $\mathsf{s}(G) \le s$ if and only if G models "$\exists S, s\text{-}\mathsf{safe}(S)$." □

Corollary 5.3 *For a fixed constant s, the property of having a connected safe set of size at most s can be expressed in MS_2.*

Theorem 5.4 *The problems of finding a safe set and a connected safe set of size at most s are fixed-parameter tractable when the solution size s is the parameter. Furthermore, the running time depends only linearly on the input size.*

Proof Let G be a given graph. Since we can handle the components separately, we assume that G is connected. We first check whether $\mathsf{tw}(G) < (s+1)^2 - 1$ in $O(n)$ time by Bodlaender's algorithm (Bodlaender 1996). If not, Lemma 5.1 implies that $\mathsf{s}(G) \ge s + 1$. Otherwise, Bodlaender's algorithm gives us a tree decomposition of G with width less than $(s+1)^2 - 1$. Courcelle's theorem (Courcelle 1992) says that it can be checked in linear time whether a graph satisfies a fixed MS_2 formula if the graph is given with a tree decomposition of constant width (see also Arnborg et al. 1991). Therefore, Lemma 5.2 and Corollary 5.3 imply the theorem. □

5.1 Relationships to other structural graph parameters

As we showed in Lemma 5.1, the treewidth of a graph is bounded by a constant if it has constant safe number. Here we further discuss the relationship to other well-studied graph parameters: tree-depth and vertex cover number. As bounding these parameters is more restricted than bounding treewidth, more problems can be solved efficiently when the problems are parameterized by tree-depth or vertex cover number (see Fellows et al. 2008; Gutin et al. 2015). In the following, we show that the safe number *lies between* these two parameters. This implies that parameterizing a problem by safe number may give a finer understanding of the parameterized complexity of the problem.

Tree-depth The *tree-depth* (Nešetřil and de Mendez 2012) [also known as *elimination tree height* (Pothen 1988) and *vertex ranking number* (Bodlaender et al. 1998)] of a connected graph G is the minimum depth of a rooted tree T such that T^* contains G as a subgraph, where T^* is the supergraph of T with the additional edges connecting all comparable pairs in T. We can easily see that the tree-depth of a graph is at least its treewidth. It is known that a graph has constant tree-depth if and only if it has a constant upper bound on the length of paths in it (Nešetřil and de Mendez 2012). Hence the proof of Lemma 5.1 implies the following relation.

Lemma 5.5 *The tree-depth of a connected graph is bounded by a constant if it has constant safe number.*

The converse of the statement above is not true in general. The *complete k-ary tree of depth d* is the rooted tree such that each non-leaf vertex has k children and the distance between the root and each leaf is d. The complete k-ary tree of depth 2 has tree-depth 2 and safe number k.

Vertex cover number A set $C \subseteq V(G)$ is a *vertex cover* of a graph G if each edge in G has at least one end in C. A *connected vertex cover* is a vertex cover that induces a connected subgraph. The *vertex cover number* of a graph is the size of a minimum vertex cover in the graph. We can see that C is a vertex cover if and only if each component of $G \setminus C$ has size 1. Thus a vertex cover is a safe set, and the following relation follows.

Lemma 5.6 *The safe number of a graph is at most its vertex cover number.*

Again the converse is not true. Consider the graph obtained from the star graph $K_{1,k}$ by subdividing each edge. It has a (connected) safe set of size 2, while its vertex cover number is k.

Note that Lemma 5.6 and Theorem 5.4 together imply that the problem of finding a (connected) safe set is fixed-parameter tractable when parameterized by vertex cover number.

6 Hardness results

A graph is *chordal* if it has no induced cycle of length 4 or more. Trees and interval graphs form the most well-known subclasses of the class of chordal graphs. A natural question would be deciding the complexity of the problems on chordal graphs. Another question is about planar graphs. As the original motivation of the problem was from a facility location problem, it would be natural to study the problem on planar graphs. In this section, we show that both cases are hard, even for some of their subclasses. We also discuss the approximation hardness of the problem.

The problem of deciding whether $\mathsf{s}(G) \leq k$ (or $\mathsf{cs}(G) \leq k$) is clearly in NP. Thus in the next subsections we only show NP-hardness.

6.1 Split graphs

A *clique* in a graph is a set of pairwise adjacent vertices. An *independent set* in a graph is a set of pairwise nonadjacent vertices. A graph $G = (V, E)$ is a *split graph* if V can be partitioned into a clique C and an independent set I. We denote such a split graph by $(C, I; E)$. In this subsection, we show that the problem of deciding whether a split graph has a safe set (or a connected safe set) of size at most k is NP-complete. This also implies the NP-completeness for chordal graphs because a split graph cannot have an induced cycle of length 4 or more.

Lemma 6.1 *Every connected split graph $G = (C, I; E)$ with two or more vertices has a minimum safe set S such that $S \subseteq C$.*

Proof First observe that $\mathsf{s}(G) \le |C|$ since C itself is a safe set. If $\mathsf{s}(G) = |C|$, then C is the desired safe set. In the following, we assume that $\mathsf{s}(G) < |C|$.

Let S be a minimum safe set of G that minimizes $|S \cap I|$. Suppose to the contrary that S includes a vertex $u \in I$. Let $v \in C \setminus S$ and $S' := S \setminus \{u\} \cup \{v\}$.

Let X and X' be the unique components of $G[V(G) \setminus S]$ and $G[V(G) \setminus S']$ that contains vertices of C, respectively. (Such components exists since $|S'| = |S| < |C|$.) All other components of $G[V(G) \setminus S]$ and $G[V(G) \setminus S']$ are of size 1. Observe that X' is obtained from X by first removing v with its neighbors in $X \cap I$ that have v as the unique neighbor in $X \cap C$, and then adding u if u has a neighbor in $(X \cap C) \setminus \{v\}$. Hence, $X' \subseteq X \cup \{u\} \setminus \{v\}$ holds.

Observe that S is adjacent to X: it is trivial if S includes a vertex in C; otherwise, X contains C and is a dominating set of G. Since S is a safe set, $|S| \ge |X|$ holds. Thus we have that $|S'| = |S| \ge |X| \ge |X'|$. This implies that S' is a minimum safe set of G. This contradicts the choice of S as $|S' \cap I| = |S \cap I| - 1$. □

Corollary 6.2 *For every split graph G, $\mathsf{s}(G) = \mathsf{cs}(G)$.*

Now we reduce Clique to our problem. In the problem Clique, we are given a graph H and an integer q and asked whether H contains a clique of size at least q. Clique is one of Karp's 21 NP-complete problems (Karp 1972). Note that Clique is NP-complete even if $2q < |V(H)|$ since we can add a long path to H without changing the maximum clique size.

Theorem 6.3 *The problem of deciding $\mathsf{s}(G) \le k$ is NP-complete even for split graphs.*

Proof Let (H, q) be an instance of Clique with $2q < |V(H)|$. We construct an instance (G, k) of our problem as follows.

Let $n = |V(H)|$ and $m = |E(H)|$. We set $k = n + m - q - \binom{q}{2}$. Let $C = V(H) \cup W$, where W is a set of $k - q$ new vertices. If $k - q$ is not positive, then $2q < n$ implies that $m < \binom{q}{2}$ and thus (H, q) is a no-instance. Thus we assume $k - q$ is positive. Let $I = E(H) \cup P$, where P is a set of $k(k - q)$ new vertices. We set $V(G) = C \cup I$. That is, we use the edges of H as vertices of G. In G, each $\{u, v\} \in E(H)$ is adjacent to u and v. Each $w \in W$ is adjacent to k vertices in P, and no two vertices in W share neighbors in P. Finally we make C a clique. Note that I is an independent set, and thus G is a split graph. See Fig. 6.

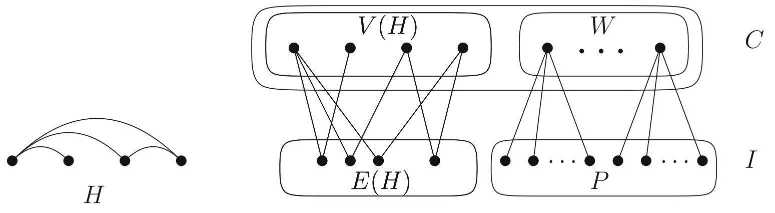

Fig. 6 The reduction for split graphs

Now we show that H has a clique of size at least q if and only if G has a safe set of size at most k.

To show the only-if part, assume that H has a clique Q of size at least q. Let $Q' \subseteq Q$ be a clique of size exactly q. Let $S = Q' \cup W$. The set S has size k. We show that S is a safe set. Let X be the unique component of $G[V(G)\setminus S]$ that contains vertices of C. It suffices to show that $|X| \le k$. It holds that

$$X \cap I = I \setminus \{v \in I : N_G(v) \subseteq S\} = E(H) \setminus E(H[Q']).$$

Since Q' is a clique of size q, we have $|X \cap I| = m - \binom{q}{2}$. As $|X \cap C| = |V(H) \setminus Q'| = n - q$, it holds that $|X| = |X \cap C| + |X \cap I| = n + m - q - \binom{q}{2} = k$.

To show the if part, assume that G has a safe set S of size at most k. By Lemma 6.1, we can assume that $S \subseteq C$. Let S' be a set of size exactly k satisfying that $S \subseteq S' \subseteq C$. (Recall that $|C| = |V(H)| + |W| = n + k - q \ge k$.) Obviously S' is a safe set. If some $w \in W$ does not belong to S', then some component of $G[V(G)\setminus S']$ contains $N_G[w]$ and thus has size at least $k + 1$. This contradicts that S' is a safe set. Hence $W \subseteq S$ holds. Let $Q = S' - W$. Since $|W| = k - q$, we have $|Q| = q$. Let X be the unique component of $G[V(G)\setminus S']$ that contains vertices of C. Since S' is a safe set, $|X| \le k$. As before, we have $|X| = |X \cap C| + |X \cap I| = n - q + m - |E(H[Q])|$, and thus $|E(H[Q])| \ge \binom{q}{2}$. This implies that Q is a clique (of size q). □

By Lemma 6.1, $\mathsf{s}(G) = \mathsf{cs}(G)$ holds for every split graph G. Thus we have NP-completeness also for the connected safe set problem.

Corollary 6.4 *The problem of deciding* $\mathsf{cs}(G) \le k$ *is NP-complete even for split graphs.*

6.2 Planar bipartite graphs

In Connected Vertex Cover, we are given a graph H and an integer $k < |V(H)|$, and our goal is to decide whether H has a connected vertex cover of size at most k. It is known that Connected Vertex Cover is NP-complete for connected planar graphs of maximum degree 4 (Garey and Johnson 1977). We reduce this problem to ours for connected planar bipartite graphs of maximum degree 7.

We first show that the problem is NP-hard even if k is relatively small.

Lemma 6.5 Connected Vertex Cover *is NP-complete for connected planar graphs G of maximum degree at most 6, even if* $4k - 2 < |V(G)|$.

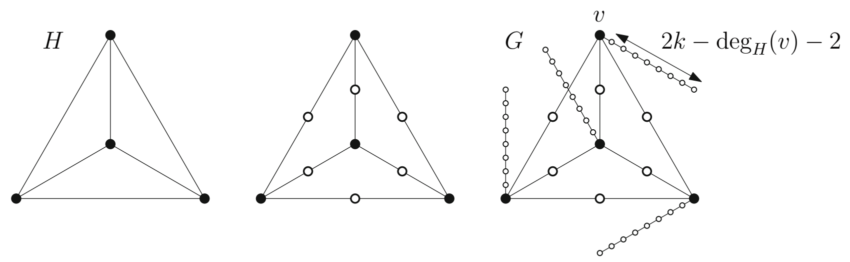

Fig. 7 The reduction for planar bipartite graphs

Proof In the reduction by Garey and Johnson (1977), the obtained graph H has a vertex v with a degree-1 neighbor. We can see that there is a minimum connected vertex cover of H that contains v.

Let d be the integer such that $(5^{d-1} - 1)/12 \le |V(H)| < (5^d - 1)/12$, and T be the complete 5-ary tree of depth d, which is with $(5^{d+1} - 1)/4$ vertices. Observe that the unique minimum connected vertex cover of T includes all inner vertices and is of size $(5^d - 1)/4$. Note that $|V(T)|$ is bounded by a polynomial in $|V(H)|$. To see this, observe that $|V(H)| \ge 2$ implies $d \ge 3$. Thus we have

$$\frac{|V(T)|}{3 \cdot |V(H)|} \le \frac{5^{d+1} - 1}{5^{d-1} - 1} = 5^2 + \frac{5^2 - 1}{5^{d-1} - 1} \le 5^2 + 1.$$

This implies that $|V(T)| \le 78 \cdot |V(H)|$.

Now take the disjoint union of H and T, and then add an edge between v and the root of T. We call the obtained graph G. This graph G has maximum degree at most 6. From the arguments above, we can see that H has a connected vertex cover of size at most k if and only if G has a connected vertex cover of size at most $k' := k + (5^d - 1)/4$. Now we can see that $4k' - 2 < |V(G)|$ as follows:

$$\begin{aligned} 4k' - 2 &< 4k + 5^d < 4|V(H)| + 5^d = |V(H)| + 3|V(H)| + 5^d \\ &< |V(H)| + (5^d - 1)/4 + 5^d = |V(H)| + (5^{d+1} - 1)/4 = |V(G)|. \end{aligned}$$

This implies the lemma. □

Let H be a connected planar graph of maximum degree at most 6, and let k be an integer such that $4k - 2 < |V(H)|$. We may also assume that $k \ge 5$ because otherwise the problem can be solved in polynomial time by an exhaustive search. First subdivide each edge in $E(H)$ once. Then, for each $v \in V(H)$, join v to one endpoint of a path P_v of $2k - \deg_H(v) - 2$ new vertices. We call the obtained graph G. See Fig. 7. Note that G is a connected planar bipartite graph of maximum degree at most 7. For each vertex $v \in V(H)$, let G_v denote the subgraph of G induced by $N_G[v] \cup V(P_v)$. Note that $|V(G_v)| = 2k - 1$ for every $v \in V(H)$.

Lemma 6.6 *If $k \ge 5$, then the following are equivalent:*

(i) *H has a connected vertex cover of size k;*
(ii) *G has a connected safe set of size $2k-1$;*
(iii) *G has a safe set of size $2k-1$.*

Proof To show that (i) implies (ii), assume that C is a connected vertex cover of H such that $|C| = k$. Let F be the set of edges in a spanning tree of $H[C]$, and let $S_F \subseteq V(G)$ be the set of corresponding subdivision vertices. We claim that $S := C \cup S_F$ is a desired connected safe set of G. Since S is connected and $|S| = 2k-1$, it suffices to show that $G - S$ has no component of size at least $2k$. Since C is a vertex cover, no two vertices in $V(H)$ are in the same component of $G[V(G)\setminus S]$. This implies that each component of $G[V(G)\setminus S]$ is contained in G_v for some $v \in V(H)$ and thus has size at most $2k-1$.

Now we show that (ii) implies (i). Let S be a connected safe set of G with $|S| = 2k-1$. Let S' be the set of vertices $v \in V(H)$ such that S intersects $\{v\} \cup V(P_v)$. Since S is connected in G, S' is connected in H. Furthermore, $|S'| \le k$ holds since the connectedness of $G[S]$ implies that $|S| \ge |S'| + |S'| - 1$. Now it suffices to show that S' is a vertex cover of H. Suppose to the contrary that $H[V(H)\setminus S']$ has an edge $\{u, v\}$. That is, $S \cap (\{u\} \cup P_u \cup \{v\} \cup P_v) = \emptyset$. Let $w \in V(G)$ be the subdivision vertex corresponding to the edge $\{u, v\}$. Observe that $w \notin S$ also holds since otherwise $\{w\}$ is a component of $G[S]$ adjacent to the component of $G[V(G)\setminus S]$ containing P_u. Thus $G[V(G)\setminus S]$ has a component that contains $\{u\} \cup P_u \cup \{v\} \cup P_v \cup \{w\}$. This component has size at least

$$(2k - \deg_H(u) - 2) + (2k - \deg_H(v) - 2) + 3 \ge 4k - 13 > k.$$

This contradicts that S is a safe set. Thus S' is a vertex cover of H.

By definition, (ii) implies (iii). To show that (iii) implies (ii), let S be a safe set of G such that $|S| = 2k-1$. A vertex of G can belong to at most two subgraphs G_v and $G_{v'}$ for some $v, v' \in V(H)$. Since $4k-2 < |V(H)|$, there is a vertex $v \in V(H)$ such that S does not intersect G_v. Hence, $G[V(G)\setminus S]$ has a connected component D of size at least $|V(G_v)| = 2k-1$. If S is not connected, then each component of $G[S]$ has size less than $2k-1$. Since G is connected, some component of $G[S]$ is adjacent to D. This contradicts that S is a safe set. □

Theorem 6.7 *The problems of deciding* $\mathsf{s}(G) \le k$ *and of deciding* $\mathsf{cs}(G) \le k$ *are NP-complete even for planar bipartite graphs of maximum degree at most 7.*

6.3 Approximation hardness

Fujita et al. (2016) presented a reduction that transforms an instance (G, k) of Independent Set to an instance $(G', |V(G)|-k+1)$ of Safe Set and Connected Safe Set. This can be seen as a reduction from an instance (G, k') of Vertex Cover to an instance $(G', k'+1)$ of Safe Set and Connected Safe Set. It is known that approximating the vertex cover number within a factor of 1.3606 is NP-hard (Dinur and Safra 2005). Since the reduction above is gap-preserving, we have the following approximation hardness for our problems.

Corollary 6.8 *It is NP-hard to approximate the safe number and the connected safe number within a factor of 1.3606.*

References

Arnborg S, Lagergren J, Seese D (1991) Easy problems for tree-decomposable graphs. J Algorithms 12:308–340

Bapat RB, Fujita S, Legay S, Manoussakis Y, Matsui Y, Sakuma T, Tuza Z (2016) Safe sets, network majority on weighted trees. Networks (To appear).

Bodlaender HL (1993) On linear time minor tests with depth-first search. J Algorithms 14:1–23

Bodlaender HL (1996) A linear-time algorithm for finding tree-decompositions of small treewidth. SIAM J Comput 25:1305–1317

Bodlaender HL, Deogun JS, Jansen K, Kloks T, Kratsch D, Müller Haiko, Tuza Z (1998) Rankings of graphs. SIAM J Discrete Math. 11:168–181

Booth KS, Lueker GS (1976) Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. J Comput Syst Sci 13:335–379

Courcelle B (1992) The monadic second-order logic of graphs III: tree-decompositions, minor and complexity issues. Theor Inform Appl 26:257–286

Cygan M, Fomin FV, Kowalik L, Lokshtanov D, Marx D, Pilipczuk M, Pilipczuk M, Saurabh S (2015) Parameterized algorithms. Springer, Berlin

Dinur I, Safra S (2005) On the hardness of approximating minimum vertex cover. Ann Math 162:439–485

Fellows MR, Lokshtanov D, Misra N, Rosamond FA, Saurabh S (2008). Graph layout problems parameterized by vertex cover. In: ISAAC 2008, volume 5369 of Lecture notes in computer science, pp 294–305

Fujita S, MacGillivray G, Sakuma T (2016) Safe set problem on graphs. Discrete Appl Math 215:106–111

Garey MR, Johnson DS (1977) The rectilinear Steiner tree problem is NP-complete. SIAM J Appl Math 32:826–834

Gutin G, Jones M, Wahlström M (2015) Structural parameterizations of the mixed chinese postman problem. In: ESA 2015, volume 9294 of Lecture notes in computer science, pp 668–679

Karp RM (1972) Reducibility among combinatorial problems. In: Proceedings of a symposium on the complexity of computer computations. The IBM research symposia series, pp 85–103

Kloks T (1994) Treewidth, computations and approximations, volume 842 of Lecture notes in computer science. Springer

Nešetřil J, de Mendez PO (2012) Sparsity: graphs, structures, and algorithms, volume 28 of Algorithms and combinatorics. Springer, Berlin

Pothen A (1988) The complexity of optimal elimination trees. Technical Report CS-88-13, Pennsylvania State University