Journal of Computer and System Sciences 75 (2009) 435–450

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Almost 2-SAT is fixed-parameter tractable

Igor Razgon *, Barry O'Sullivan

Cork Constraint Computation Centre, Department of Computer Science, University College Cork, Ireland

A R T I C L E   I N F O

Article history:
Received 10 August 2008
Received in revised form 31 March 2009
Available online 8 April 2009

Keywords:
Fixed-parameter algorithms
Satisfiability problems
Separation problems

A B S T R A C T

We consider the following problem. Given a 2-CNF formula, is it possible to remove at most k clauses so that the resulting 2-CNF formula is satisfiable? This problem is known to different research communities in theoretical computer science under the names Almost 2-SAT, All-but-k 2-SAT, 2-CNF deletion, and 2-SAT deletion. The status of the fixed-parameter tractability of this problem is a long-standing open question in the area of parameterized complexity. We resolve this open question by proposing an algorithm that solves this problem in $\mathcal{O}(15^k \times k \times m^3)$ time showing that this problem is fixed-parameter tractable.

1. Introduction

We consider the following problem. Given a 2-CNF formula, is it possible to remove at most k clauses so that the resulting 2-CNF formula is satisfiable? This problem is known to different research communities in theoretical computer science under the names Almost 2-SAT, All-but-k 2-SAT, 2-CNF deletion, and 2-SAT deletion. The status of the fixed-parameter tractability of this problem is a long-standing open question in the area of parameterized complexity. The question was first raised in 1997 by Mahajan and Raman [13] (see [14] for the journal version). The question was also posed by Niedermeier [17], being referred to as one of central challenges for parameterized algorithm design. Finally, in July 2007, this question was included by Fellows in the list of open problems of the Dagstuhl seminar on Parameterized Complexity [6]. In this paper we resolve this open question by proposing an algorithm that solves this problem in $\mathcal{O}(15^k \times k \times m^3)$ time. Thus we show that this problem is fixed-parameter tractable (FPT).

Regarding the name of this problem, we use *Almost 2-SAT* (2-ASAT) to refer to the optimization problem whose output is the smallest subset of clauses that have to be removed from the given 2-CNF formula so that the resulting formula is satisfiable. The *parameterized* 2-ASAT problem gets a parameter k as additional input, and the corresponding decision problem is to determine whether at most k clauses can be removed so that the resulting formula becomes satisfiable. The algorithm proposed in this paper solves the parameterized 2-ASAT problem.

1.1. Overview of the algorithm

We define a variation of the 2-ASAT problem called the *Annotated* 2-ASAT *problem with a single literal* (2-ASLASAT). The input of this problem is a triple (F, L, l), where F is a 2-CNF formula, L is a set of literals such that F is *satisfiable with respect to* L (i.e. $F \wedge \bigwedge_{l' \in L} l'$ is satisfiable), l is a single literal. The task is to find a smallest subset of clauses of F such that after their removal the resulting formula is satisfiable with respect to $(L \cup \{l\})$. The description of the algorithm for the parameterized 2-ASAT problem is divided into two parts. In the first, and most important part, we provide an $\mathcal{O}(5^k \times k \times m^2)$ time algorithm that solves the parameterized 2-ASLASAT problem, where the parameter k is the maximum number of clauses to be removed

* Corresponding author.
E-mail addresses: i.razgon@cs.ucc.ie (I. Razgon), b.osullivan@cs.ucc.ie (B. O'Sullivan).

0022-0000/$ – see front matter
doi:10.1016/j.jcss.2009.04.002

and m is the number of clauses of F. In the second part we show that the parameterized 2-ASAT problem can be solved by $\mathcal{O}(3^k \times m)$ applications of the algorithm solving the parameterized 2-ASLASAT problem. The resulting runtime follows from the product of the two corresponding complexity expressions. The transformation of the 2-ASAT problem into the 2-ASLASAT problem is based on *iterative compression* and can be seen as an adaptation of the method employed in [10] in order to solve the graph bipartization problem. In the following we present an overview of the first part of the algorithm.

We introduce a *polynomially computable lower bound* on the solution size of the 2-ASLASAT problem for input (F, L, l). Then we prove that if a literal l^* is *neutral*, i.e. the lower bound on the solution size for $(F, L \cup \{l^*\}, l)$ is the same as for (F, L, l), then the solution size for $(F, L \cup \{l^*\}, l)$ and (F, L, l) is the same. This theorem allows us to introduce an algorithm that selects a clause C of F and applies the following branching rule. If C includes a neutral literal l^* then the algorithm applies itself recursively to $(F, L \cup \{l^*\}, l)$ *without any branching*. If not, the algorithm produces at most three branches. On one of them it removes C from F and decreases the parameter. On each of the other branches the algorithm adds one of the literals of C to L and applies itself recursively without changing the size of the parameter. The size of the search tree produced by the algorithm is bounded because on each branch either the parameter is decreased or the lower bound on the solution size is increased (because the literals of the selected clause are *not neutral*). Thus on each branch *the gap between the parameter and the lower bound of the solution size is decreased* which ensures that the size of the search tree exponentially depends only on k and not on the size of F.

The lower bound mentioned in the previous paragraph is obtained by representing the 2-ASLASAT as a *separation problem*. In particular, we define the notion of a walk of a 2-CNF formula and show that, given an instance (F, L, l) of the 2-ASLASAT problem, F is unsatisfiable with respect to $L \cup \{l\}$ if and only if there is a walk from $\neg L$ (i.e. from the set of negations of the literals of L) to $\neg l$ or a walk from $\neg l$ to $\neg l$. Thus the 2-ASLASAT problem can be viewed as a problem of finding the smallest set of clauses whose removal breaks all these walks. The considered lower bound on the solution size is the smallest number of clauses separating $\neg L$ from $\neg l$. We show that the size of this *separator* equals the largest number of clause-disjoint *paths* (i.e. walks without repeated clauses) from $\neg L$ to $\neg l$ and that it can be computed in polynomial time by a Ford–Fulkerson-like procedure. For this proof it is essential that F is satisfiable with respect to L.

1.2. Related work

The parameterized 2-ASAT problem was introduced in [13]. In [12], this problem was shown to be a generalization of the parameterized graph bipartization problem, which was also an open problem at that time. The latter problem was resolved in [19]. The additional contribution of [19] was introducing the method of iterative compression that has had a considerable impact on the design of parameterized algorithms. The most recent algorithms based on this method are currently the best known for the undirected Feedback Vertex Set [2] and the first parameterized algorithm for the famous Directed Feedback Vertex Set problem [5]. For earlier results based on iterative compression, we refer the reader to a survey article [11].

The study of parameterized graph separation problems was initiated in [15]. The technique introduced by the author allowed him to design fixed-parameter algorithms for the multiterminal cut problem and for a more general multicut problem. The latter assumed that the number of pairs of terminals to be separated was also a parameter. This result was extended in [9] where fixed-parameter algorithms for multicut problems on several classes of graphs were proposed. The first $\mathcal{O}(c^k \times poly(n))$ algorithm for the multiterminal cut problem was proposed in [4]. A reformulation of the main theorem of [4] is an essential part of the parameterized algorithm for the Directed FVS problem [5] mentioned above. In this paper we apply the proof strategy for this theorem in order to show that adding a *neutral* literal to the set of literals of the input does not increase the solution size. Along with computing the separators, the methods of computing disjoint paths have been investigated. The research led to intractability results [20] and parameterized approximability results [8].

The parameterized MAX-SAT problem (a complementary problem to the one considered in the present paper) where the goal is to satisfy at least k clauses of arbitrary sizes also received a considerable attention from the researchers. The best currently known algorithm for this problem runs in $\mathcal{O}(1.37^k + |F|)$ time, where $|F|$ is the size of the given formula [3].

1.3. Structure of the paper

In Section 2 we introduce the terminology that we use in the rest of the paper. In Section 3 we prove a number of theorems that are necessary for the design and analysis of a parameterized algorithm for the 2-ASLASAT problem. The algorithm and its analysis are presented in Section 4. In Section 5 we present the iterative compression-based transformation from parameterized 2-ASAT problem to the parameterized 2-ASLASAT problem. Section 6 concludes the paper by presenting a number of additional results that follow from the fixed-parameter tractability of the 2-ASAT problem.

2. Terminology

2.1. 2-CNF formulas

A CNF formula F is called a 2-CNF *formula* if each clause of F is of size at most 2. Throughout the paper we make two assumptions regarding the 2-CNF formulas we consider. Firstly, we assume that all the clauses are of size 2. If a formula has a clause (l) of size 1 then this clause is represented as $(l \vee l)$. Secondly, everywhere except in the very last theorem, we assume that all the clauses of any formula are pairwise distinct, i.e. no two clauses have the same set of literals. This assumption allows us to represent the operation of removing clauses from a formula in a set-theoretic manner. In particular, let S be a set of clauses. Then $F \setminus S$ is a 2-CNF formula that is the conjunction of clauses of F that are not contained in S. The result of removing a single clause C is denoted by $F \setminus C$ rather than $F \setminus \{C\}$.

Let F, S, C, L be a 2-CNF formula, a set of clauses, a single clause, and a set of literals, respectively. Then $Var(F)$, $Var(S)$, $Var(C)$, and $Var(L)$ denote the set of variables whose literals appear in F, S, C, and L, respectively. For a single literal l, we denote by $Var(l)$ the variable of l. Also we denote by $Clauses(F)$ the set of clauses of F.

A set of literals L is called *non-contradictory* if it does not contain a literal and its negation. A literal l *satisfies* a clause $(l_1 \vee l_2)$ if $l = l_1$ or $l = l_2$. Given a 2-CNF formula F, a non-contradictory set of literals L such that $Var(F) = Var(L)$ and each clause of F is satisfied by at least one literal of L, we call L a *satisfying assignment* of F. F is *satisfiable* if it has at least one satisfying assignment. Given a set of literals L, we denote by $\neg L$ the set consisting of negations of all the literals of L. For example, if $L = \{l_1, l_2, \neg l_3\}$ then $\neg L = \{\neg l_1, \neg l_2, l_3\}$.

Let F be a 2-CNF formula and L be a set of literals. F is *satisfiable with respect to* L if $F \wedge \bigwedge_{l' \in L} l'$ is satisfiable. The notion of satisfiability of a 2-CNF formula with respect to the given set of literals will be very frequently used in the paper, hence, in order to save space, we introduce a special notation for this notion. In particular, we say that SWRT(F, L) is true (false) if F is satisfiable (respectively, not satisfiable) with respect to L. If L consists of a single literal l then we write SWRT(F, l) rather than SWRT$(F, \{l\})$.

2.2. Walks and paths

Definition 1 *(Walk of a 2-CNF).* A *walk* of the given 2-CNF formula F is a non-empty sequence $w = (C_1, \ldots, C_q)$ of (not necessarily distinct) clauses of F having the following property. For each C_i one of its literals is specified as the *first literal* of C_i, the other literal is the *second literal*, and for any two consecutive clauses C_i and C_{i+1} the second literal of C_i is the negation of the first literal of C_{i+1}. The walk w is a *path* if all its clauses are pairwise distinct.

Let $w = (C_1, \ldots, C_q)$ be a walk and let l' and l'' be the first literal of C_1 and the second literal of C_q, respectively. Then we say that l' is *the first literal of* w, that l'' is *the last literal of* w, and that w is *a walk from* l' *to* l''. Let L be a set of literals such that $l' \in L$. Then we say that w is a walk *from* L. Let $C = (l_1 \vee l_2)$ be a clause of w. Then l_1 is a first literal of C with respect to w if l_1 is the first literal of some C_i such that $C = C_i$. A second literal of a clause with respect to a walk is defined accordingly. (Generally a literal of a clause may be both a first and a second with respect to the given walk, which is shown in the example below.) We denote by *reverse*(w) a walk $(C_q, \ldots, C_1)$ in which the first and the second literals of each entry are exchanged with respect to w. Given a clause $C'' = (\neg l'' \vee l^*)$, we denote by $w + (\neg l'' \vee l^*)$ the walk obtained by appending C'' to the end of w and setting $\neg l''$ to be the first literal of the last entry of $w + (\neg l'' \vee l^*)$ and l^* to be the second one. More generally, let w' be a walk whose first literal is $\neg l''$. Then $w + w'$ is the walk obtained by concatenating w' to the end of w with the first and second literals of all entries in w and w' preserving their roles in $w + w'$.

Consider an example demonstrating the above notions. Let $w = (l_1 \vee l_2), (\neg l_2 \vee l_3), (\neg l_3 \vee l_4), (\neg l_4 \vee \neg l_3), (l_3 \vee \neg l_2), (l_2 \vee l_5)$ be a walk of some 2-CNF formula presented so that the first literals of all entries appear before the second literals. Then l_1 and l_5 are the first and the last literals of w, respectively, and hence w is a walk from l_1 to l_5. The clause $(\neg l_2 \vee l_3)$ has an interesting property that both its literals are first literals of this clause with respect to w (and therefore the second literals as well). The second item of w witnesses $\neg l_2$ being a first literal of $(\neg l_2 \vee l_3)$ with respect to w (and hence l_3 being a second one), while the second item of w from the end provides the witness for l_3 being a first literal of $(\neg l_2 \vee l_3)$ with respect to w (and hence $\neg l_2$ being a second one). The rest of clauses do not possess this property. For example l_1 is the first literal of $(l_1 \vee l_2)$ with respect to w (as witnessed by the first entry) but not the second one. Next, *reverse*$(w) = (l_5 \vee l_2), (\neg l_2 \vee l_3), (\neg l_3 \vee \neg l_4), (l_4 \vee \neg l_3), (l_3 \vee \neg l_2), (l_2 \vee l_1)$. Let w_1 be the prefix of w containing all the clauses except the last one. Then $w = w_1 + (l_2 \vee l_5)$. Let w_2 be the prefix of w containing the first 4 entries, w_3 be the suffix of w containing the last 2 entries. Then $w = w_2 + w_3$. Finally, observe that w is not a path due to the repeated occurrence of clause $(\neg l_2 \vee l_3)$, while w_2 is a path.

2.3. The 2-ASAT and 2-ASLASAT problems

Definition 2 *(Culprit sets, 2-ASAT and 2-ASLASAT problems).*

- A Culprit Set (CS) of a 2-CNF formula F is a subset S of $Clauses(F)$ such that $F \setminus S$ is satisfiable. We call the problem of finding a *Smallest CS* (SCS) of F the Almost 2-SAT Problem (2-ASAT Problem).
- Let (F, L, l) be a triple where F is a 2-CNF formula, L is a non-contradictory set of literals such that SWRT(F, L) is true and l is a literal such that $Var(l) \notin Var(L)$. A CS of (F, L, l) is a subset S of $Clauses(F)$ such that SWRT$(F \setminus S, L \cup \{l\})$ is true. We call the problem of finding a SCS of (F, L, l) the Annotated Almost 2-SAT problem with single literal (2-ASLASAT Problem).

In this paper we consider the parameterized versions of the 2-ASAT and 2-ASLASAT problems. In particular, the input of the *parameterized* 2-ASAT problem is (F, k), where F is a 2-CNF formula and k is a non-negative integer. The output is a CS of F

of size at most k, if one exists. Otherwise, the output is 'NO'. The input of the *parameterized* 2-ASLASAT problem is (F, L, l, k) where (F, L, l) is as specified in Definition 2. The output is a CS of (F, L, l) of size at most k, if one exists. Otherwise, the output is 'NO'.

3. 2-ASLASAT problem: related theorems

3.1. Basic lemmas

Lemma 1. *Let F be a 2-CNF formula and w be a walk of F. Let l_x and l_y be the first and the last literals of w, respectively. Then $\mathrm{SWRT}(F, \{\neg l_x, \neg l_y\})$ is false. In particular, if $l_x = l_y$ then $\mathrm{SWRT}(F, \neg l_x)$ is false.*

Proof. Since w is a walk of F, $Var(l_x) \in Var(F)$ and $Var(l_y) \in Var(F)$. Consequently for any satisfying assignment P of F both $Var(l_x)$ and $Var(l_y)$ belong to $Var(P)$. Therefore $\mathrm{SWRT}(F, \{\neg l_x, \neg l_y\})$ may be true only if there is a satisfying assignment of F containing both $\neg l_x$ and $\neg l_y$. We going to show that this is impossible by induction on the length of w.

This lemma holds if $|w| = 1$ because in this case $w = (l_x \vee l_y)$. Assume that $|w| > 1$ and the statement is satisfied for all shorter walks. Then $w = w' + (l_t \vee l_y)$, where w' is a walk of w from l_x to $\neg l_t$. By the induction assumption $\mathrm{SWRT}(F, \{\neg l_x, l_t\})$ is false and hence any satisfying assignment of F containing $\neg l_x$ contains $\neg l_t$ and hence contains l_y. As we noted above, this implies that $\mathrm{SWRT}(F, \{\neg l_x, \neg l_y\})$ is false. □

Lemma 2. *Let F be a 2-CNF formula and let L be a set of literals such that $\mathrm{SWRT}(F, L)$ is true. Let $C = (l_1 \vee l_2)$ be a clause of F and let w be a walk of F from $\neg L$ containing C and assume that l_1 is a first literal of C with respect to w. Then l_1 is* not *a second literal of C with respect to any walk from $\neg L$.*

Proof. Let w' be a walk of F from $\neg L$ which contains C so that l_1 is a second literal of C with respect to w'. Then w' has a prefix w'' whose last literal is l_1. Let l' be the first literal of w' (and hence of w''). According to Lemma 1, $\mathrm{SWRT}(F, \{\neg l_1, \neg l'\})$ is false. Therefore, if $l_1 \in \neg L$ then $\mathrm{SWRT}(F, L)$ is false (because $\{\neg l_1, \neg l'\} \subseteq L$) in contradiction to the conditions of the lemma. Thus $l_1 \notin \neg L$ and hence l_1 is not the first literal of w. Consequently, w has a prefix w^* whose last literal is $\neg l_1$. Let l^* be the first literal of w (and hence of w^*). Then $w^* + reverse(w'')$ is a walk from l^* to l', and both belong to $\neg L$. According to Lemma 1, $\mathrm{SWRT}(F, \{\neg l^*, \neg l'\})$ is false and hence $\mathrm{SWRT}(F, L)$ is false in contradiction to the conditions of the lemma. It follows that the walk w' does not exist and the present lemma is correct. □

Lemma 3. *Let F be a 2-CNF formula, let L be a set of literals such that $\mathrm{SWRT}(F, L)$ is true, and let w be a walk from $\neg L$. Then F has a path p with the same first and last literals as w and the set of clauses of p is a subset of the set of clauses of w.*

Proof. The proof is by induction on the length of w. The lemma holds if $|w| = 1$ because w itself is the desired path. Assume that $|w| > 1$ and the lemma holds for all shorter paths from $\neg L$. If all clauses of w are distinct then w is the desired path. Otherwise, let $w = (C_1, \dots, C_q)$ and assume that $C_i = C_j$ where $1 \leqslant i < j \leqslant q$. By Lemma 2, C_i and C_j have the same first (and, of course, the second) literal. If $i = 1$, let w' be the suffix of w starting at C_j. Otherwise, if $C_j = q$, let w' be the prefix of w ending at C_i. If none of the above happens then $w' = (C_1, \dots, C_i, C_{j+1}, C_q)$. In all the cases, w' is a walk of F with the same first and last literals as w such that $|w'| < |w|$ and the set of clauses of w' is a subset of the set of clauses of w. The desired path is extracted from w' by the induction assumption. □

3.2. A non-empty SCS of (F, L, l): necessary and sufficient condition

Theorem 1. *Let (F, L, l) be an instance of the 2-ASLASAT problem. Then $\mathrm{SWRT}(F, L \cup \{l\})$ is false if and only if F has a walk from $\neg l$ to $\neg l$ or a walk from $\neg L$ to $\neg l$.*

Proof. Assume that F has a walk from $\neg l$ to $\neg l$ or from $\neg l'$ to $\neg l$ such that $l' \in L$. Then, according to Lemma 1, $\mathrm{SWRT}(F, l)$ is false or $\mathrm{SWRT}(F, \{l', l\})$ is false, respectively. Clearly in both cases $\mathrm{SWRT}(F, L \cup \{l\})$ is false as $L \cup \{l\}$ is, by definition, a superset of both $\{l\}$ and $\{l', l\}$.

Assume now that $\mathrm{SWRT}(F, L \cup \{l\})$ is false. Let I be a set of literals including l and all literals l' such that F has a walk from $\neg l$ to l'. Let S be the set of all clauses of F satisfied by I.

Assume that I is non-contradictory and does not intersect with $\neg L$. Let P be a satisfying assignment of F which does not intersect with $\neg L$ (such an assignment exists according to definition of the 2-ASLASAT problem). Let P' be the subset of P such that $Var(P') = Var(F) \setminus Var(I)$. Observe that $P' \cup I$ is non-contradictory. Indeed, P' is non-contradictory as being a subset of a satisfying assignment P of F, I is non-contradictory by assumption, and due to the disjointness of $Var(I)$ and $Var(P')$, there is no literal $l' \in I$ and $\neg l' \in P'$. Next, note that every clause C of F is satisfied by $P' \cup I$. Indeed, if $C \in S$ then C is satisfied by I, by definition of I. Otherwise, assume first that $Var(C) \cap Var(I) \neq \emptyset$. Then $C = (\neg l' \vee l'')$, where $l' \in I$. Then either $l' = l$ or F has a walk w from $\neg l$ to l'. Consequently, either $(\neg l' \vee l'')$ or $w + (\neg l' \vee l'')$ is a walk from $\neg l$ to l'' witnessing that $l'' \in I$ and hence $C \in S$, giving a contradiction.

It remains to consider the case where $Var(C) \cap Var(I) = \emptyset$, i.e. $Var(C) \subseteq Var(P')$. If P' contains contradictions of both literals of C then $P \setminus P'$ contains at least one literal of C implying that P contains a literal and its negation in contradiction to the definition of P. Consequently, C is satisfied by P'. Taking into account that $Var(P' \cup I) = Var(F)$, $P' \cup I$ is a satisfying assignment of F. Observe that $P' \cup I$ does not intersect with $\neg(L \cup l)$. Indeed, both I and P' do not intersect with $\neg L$, the former by assumption the latter by definition. Next, $l \in I$ and $P' \cup I$ is non-contradictory, hence $\neg l \notin P' \cup I$. Thus $P' \cup I$ witnesses that $\textsc{swrt}(F, L \cup \{l\})$ is true in contradiction to our assumption. Thus our assumption regarding I made in the previous paragraph is incorrect.

It follows from that either I contains a literal and its negation or I intersects with $\neg L$. In the former case if $\neg l \in I$ then by the definition of I there is a walk from $\neg l$ to $\neg l$. Otherwise I contains l' and $\neg l'$ such that $Var(l') \neq Var(l)$. Let w_1 be the walk from $\neg l$ to l' and let w_2 be the walk from $\neg l$ to $\neg l'$ (both walks exist according to the definition of I). Clearly $w_1 + reverse(w_2)$ is a walk from $\neg l$ to $\neg l$. In the latter case, F has a walk w from $\neg l$ to $\neg l'$ such that $l' \in L$. Clearly $reverse(w)$ is a walk from $\neg L$ to $\neg l$. Thus we have shown that if $\textsc{swrt}(F, L \cup \{l\})$ is false then F has a walk from $\neg l$ to $\neg l$ or a walk from $\neg L$ to $\neg l$, which completes the proof of the theorem. □

3.3. Smallest separators

Definition 3. A set SC of clauses of a 2-CNF formula F is a separator with respect to a set of literals L and literal l_y if $F \setminus SC$ does not contain a path from L to l_y.

We denote by $SepSize(F, L, l_y)$ the size of a smallest separator of F with respect to L and l_y, and by $\mathbf{OptSep}(F, L, l_y)$ the set of all smallest separators of F with respect to L and l_y. Thus for any $S \in \mathbf{OptSep}(F, L, l_y)$, $|S| = SepSize(F, L, l_y)$.

Given Definition 3, we derive an easy corollary from Lemma 1.

Corollary 1. *Let (F, L, l) be an instance of the 2-ASLASAT problem. Then the size of an SCS of this instance is greater than or equal to $SepSize(F, \neg L, \neg l)$.*

Proof. Assume by contradiction that S is a CS of (F, L, l) such that $|S| < SepSize(F, \neg L, \neg l)$. Then $F \setminus S$ has at least one path p from a literal $\neg l'$ ($l' \in L$) to $\neg l$. According to Lemma 1, $F \setminus S$ is not satisfiable with respect to $\{l', l\}$ and hence it is not satisfiable with respect to $L \cup \{l\}$ which is a superset of $\{l', l\}$. That is, S is *not* a CS of (F, L, l), giving a contradiction. □

Let $D = (V, A)$ be the *implication graph* on F, which is a digraph whose set $V(D)$ of nodes corresponds to the set of literals of the variables of F and (l_1, l_2) is an arc in its set $A(D)$ of arcs if and only if $(\neg l_1 \vee l_2) \in Clauses(F)$. We say that arc (l_1, l_2) *represents* the clause $(\neg l_1 \vee l_2)$. Note that each arc represents exactly one clause, while a clause including two distinct literals is represented by two different arcs. In particular, if $\neg l_1 \neq l_2$, the other arc which represents $(\neg l_1 \vee l_2)$ is $(\neg l_2, \neg l_1)$. In the context of D we denote by L and $\neg L$ the set of nodes corresponding to the literals of L and $\neg L$, respectively. We adopt the definition of a walk and a path of a digraph given in [1]. Taking into account that all the walks of D considered in this paper are non-empty we represent them as the sequences of arcs instead of alternative sequences of arcs and nodes. In other words, if $w = (x_1, e_1, \ldots, x_q, e_q, x_{q+1})$ is a walk of D, we represent it as $(e_1, \ldots, e_q)$. The *arc separator* of D with respect to a set of literals L and a literal l is a set of arcs such that the graph resulting from their removal has no path from L to l. Similar to the case with 2-CNF formulas, we denote by $ArcSepSize(D, L, l)$ the size of the smallest arc separator of D with respect to L and l.

Theorem 2. *Let F be a 2-CNF formula, let L be a set of literals such that $\textsc{swrt}(F, \neg L)$ is true. Let l_y be a literal such that $Var(l_y) \notin Var(L)$. Then the following statements hold.*

1. *The largest number of clause-disjoint paths from L to l_y in F equals $SepSize(F, L, l_y)$.*
2. *$SepSize(F, L, l_y) = ArcSepSize(D, \neg L, l_y)$.*

Remark. Note that generally (if there is no requirement that $\textsc{swrt}(F, \neg L)$ is true) $SepSize(F, L, l_y)$ may differ from $ArcSepSize(D, \neg L, l_y)$. The reason is that a separator of D may correspond to a smaller separator of F due to the fact that some arcs may represent the same clause. As we will see in the proof, the requirement that $\textsc{swrt}(F, \neg L)$ is true rules out this possibility.

Proof of Theorem 2. We can safely assume that $Var(L) \subseteq Var(F)$ because literals whose variables do not belong to $Var(F)$ cannot be starting points of paths in F. Also since $l_y \notin \neg L$ any walk from $\neg L$ to l_y in D is non-empty. We use this fact implicitly in the proof without referring to it.

We start from establishing a correspondence between walks of F and walks of D. Let $w = (C_1, \ldots, C_q)$ be a walk from l' to l'' in F. Let $w(D) = (a_1, \ldots, a_q)$ be the sequence of arcs of D constructed as follows. For each $C_i = (l_1 \vee l_2)$, $a_i = (\neg l_1, l_2)$; we assume that l_1 is the first literal of C_i. Then $\neg l'$ is the tail of a_1 and l'' is the head of a_q. Also, by definition of w, for any two arcs a_i and a_{i+1}, the head of a_i is the same as the tail of a_{i+1}. It follows that $w(D)$ is a walk from $\neg l'$ to l'' in

D such that each a_i represents C_i. Conversely, let $p = (a_1, \dots, a_q)$ be a walk from $\neg l'$ to l'' in D. Let $p(F)$ be the sequence $(C_1, \dots, C_q)$ of clauses defined as follows. For each $a_i = (\neg l_1, l_2)$, $C_i = (l_1 \vee l_2)$, l_1 and l_2 are specified as the first and the second literals of C_i, respectively. Then l' is the first literal of C_1, l'' is the last literal of C_q and for each consecutive pair C_i and C_{i+1} the second literal of C_i is the negation of the first literal of C_{i+1}. In other words, $p(F)$ is a walk from l' to l'' in F where each C_i is represented by a_i.

Next, we show that a set of t arc-disjoint paths from $\neg L$ to l_y in D corresponds to a set of t clause-disjoint paths from L to l_y in F. In particular, let $\mathbf{P} = \{p_1, \dots, p_t\}$ be a set of arc-disjoint paths from $\neg L$ to l_y in D. Then $\{p_1(F), \dots, p_t(F)\}$ is a set of walks from L to l_y in F. Assume that these walks are not clause-disjoint or contain repeated occurrences of clauses. It follows that there are two distinct arcs participating in the paths of $\mathbf{P}$ that represent the same clause. In particular, there is a clause $C = (l_1 \vee l_2)$ that belongs to $p_i(F)$ and $p_j(F)$ (i and j can be equal) that is represented by arc $(\neg l_1, l_2)$ in p_i and by arc $(\neg l_2, l_1)$ in p_j. By construction of $p_i(F)$ and $p_j(F)$, l_1 is the first literal of C with respect to $p_i(F)$ and the second literal of C with respect to $p_j(F)$ in contradiction to Lemma 2. This contradiction shows that $\{p_1(F), \dots, p_t(F)\}$ are indeed clause-disjoint paths.

Let S be a smallest arc separator of D with respect to $\neg L$ and l_y. For each $a \in S$, let p_a be a path in D from $\neg L$ to l_y which includes a. Let $C(a)$ be a clause of $p_a(F)$ which is represented by a. Denote the set of all $C(a)$ by $S(F)$. Then we can show that $S(F)$ is a separator with respect to L and l_y in F. In particular, let p^* be a path from L to l_y in $F \setminus S(F)$. Then $p^*(D)$ necessarily includes an arc $a \in S$. Let C^* be a clause of p^* represented by a. Since $C^* \neq C(a)$, the arc a represents two different clauses in contradiction to the definition of D. Consequently, taking into account that $|S(F)| \leqslant |S|$, $ArcSepSize(D, \neg L, l_y) \geqslant SepSize(F, L, l_y)$.

Since we assumed that $l_y \notin \neg L$, it follows from the arc version of Menger's Theorem for directed graphs [1] that D has $ArcSepSize(D, \neg L, l_y)$ arc-disjoint paths from $\neg L$ to l_y. According to the proven above, F has $ArcSepSize(D, \neg L, l_y)$ clause-disjoint paths from L to l_y. Since any separator with respect to L and l_y intersects each of these paths, $ArcSepSize(D, \neg L, l_y) \leqslant SepSize(F, L, l_y)$. Taking into account the previous paragraph $ArcSepSize(D, \neg L, l_y) = SepSize(F, L, l_y)$ and $SepSize(F, L, l_y)$ is the largest possible number of clause-disjoint paths from L to l_y in F. □

3.4. Neutral literals

Definition 4. Let (F, L, l) be an instance of the 2-ASLASAT problem. A literal l^* is a *neutral literal* of (F, L, l) if $(F, L \cup \{l^*\}, l)$ is a valid instance of the 2-ASLASAT problem and $SepSize(F, \neg L, \neg l) = SepSize(F, \neg(L \cup \{l^*\}), \neg l)$.

The following theorem has a crucial role in the design of the algorithm provided in the next section.

Theorem 3. *Let (F, L, l) be an instance of the 2-ASLASAT problem and let l^* be a neutral literal of (F, L, l). Then there is a CS of $(F, L \cup \{l^*\}, l)$ of size smaller than or equal to the size of an SCS of (F, L, l).*

Before we prove Theorem 3, we extend our terminology.

Definition 5. Let (F, L, l) be an instance of the 2-ASLASAT problem. A clause $C = (l_1 \vee l_2)$ of F is *reachable* from $\neg L$ if there is a walk w from $\neg L$ including C. Assume that l_1 is a first literal of C with respect to w. Then l_1 is called *the main* literal of C with respect to (F, L, l).

Given Definition 5, Lemma 2 immediately implies the following corollary.

Corollary 2. *Let (F, L, l) be an instance of the 2-ASLASAT problem and let $C = (l_1 \vee l_2)$ be a clause reachable from $\neg L$. Assume that l_1 is the main literal of C with respect to (F, L, l). Then l_1 is not a second literal of C with respect to any walk w' starting from $\neg L$ and including C.*

Now we are ready to prove Theorem 3.

Proof of Theorem 3. Let $SP \in \mathbf{OptSep}(F, \neg(L \cup \{l^*\}), \neg l)$. Since $\neg L$ is a subset of $\neg(L \cup \{l^*\})$, SP is a separator with respect to $\neg L$ and $\neg l$ in F. Moreover, since l^* is a neutral literal of (F, L, l), $SP \in \mathbf{OptSet}(F, \neg L, \neg l)$.

In the 2-CNF $F \setminus SP$, let R be the set of clauses reachable from $\neg L$ and let NR be the rest of the clauses of $F \setminus SP$. Observe that the sets R, NR and SP are a partition of the set of clauses of F.

Let X be a SCS of (F, L, l). Denote $X \cap R$, $X \cap SP$, $X \cap NR$ by XR, XSP, and XNR, respectively. Observe that the sets XR, XSP, and XNR are a partition of X.

Let Y be the subset of $SP \setminus XSP$ including all clauses $C = (l_1 \vee l_2)$ (we assume that l_1 is the main literal of C with respect to (F, L, l)) such that there is a walk w from l_1 to $\neg l$ with C being the first clause of w and all clauses of w following C (if any) belong to $NR \setminus XNR$. We call this walk w *a witness walk* of C. By definition, $SP \setminus XSP = SP \setminus X$ and $NR \setminus XNR = NR \setminus X$, hence the clauses of w do not intersect with X.

Claim 1. $|Y| \leqslant |XR|$.

Proof. By definition of the 2-ASLASAT problem, SWRT(F, L) is true. Therefore, according to Theorem 2, there is a set $\mathbf{P}$ of $|SP|$ clause-disjoint paths from $\neg L$ to $\neg l$. Clearly each $C \in SP$ participates in exactly one path of $\mathbf{P}$ and each $p \in \mathbf{P}$ includes exactly one clause of SP. In other words, we can make a one-to-one correspondence between paths of $\mathbf{P}$ and the clauses of SP they include. Let $\mathbf{PY}$ be the subset of $\mathbf{P}$ consisting of the paths corresponding to the clauses of Y. We are going to show that for each $p \in \mathbf{PY}$ the clause of SP corresponding to p is preceded in p by a clause of XR.

Assume by contradiction that this is not true for some $p \in \mathbf{PY}$ and let $C = (l_1 \vee l_2)$ be the clause of SP corresponding to p with l_1 being the main literal of C with respect to (F, L, l). By our assumption, C is the only clause of SP participating in p, hence all the clauses of p *preceding* C belong to R. Consequently, the only possibility of those preceding clauses intersecting with X is by intersecting with XR. Since this possibility is ruled out according to our assumption, we conclude that no clause of p preceding C belongs to X.

Next, according to Corollary 2, l_1 is the first literal of C with respect to p, hence the suffix of p starting at C can be replaced by the walk that is a witness of C[1] and as a result of this replacement, a walk w' from $\neg L$ to $\neg l$ is obtained. Taking into account that the witness walk of C does not intersect with X, we get that w' does not intersect with X. By Theorem 1, SWRT$(F \setminus X, L \cup \{l\})$ is false in contradiction to being X a CS of (F, L, l). This contradiction shows that our initial assumption fails and C is preceded in p by a clause of XR.

In other words, each path of $\mathbf{PY}$ intersects with a clause of XR. Since the paths of $\mathbf{PY}$ are clause-disjoint, $|XR| \geqslant |\mathbf{PY}| = |Y|$, as required. □

Consider the set $X^* = Y \cup XSP \cup XNR$. Observe that $|X^*| = |Y| + |XSP| + |XNR| \leqslant |XR| + |XSP| + |XNR| = |X|$, the first equality follows from the mutual disjointness of Y, XSP and XNR by definition, the inequality follows from Claim 1, the last equality was justified in the paragraph where the sets XP, XSP, XNR, and X have been defined. We are going to show that X^* is a CS of $(F, L \cup \{l^*\}, l)$ which will complete the proof of the present theorem.

Claim 2. *$F \setminus X^*$ has no walk from $\neg(L \cap \{l^*\})$ to $\neg l$.*

Proof. Assume by contradiction that w is a walk from $\neg(L \cap \{l^*\})$ to $\neg l$ in $F \setminus X^*$. Taking into account that SWRT$(F \setminus X^*, L \cup \{l^*\})$ is true (because we know that SWRT$(F, L \cup \{l^*\})$ is true), and applying Lemma 3, we get that $F \setminus X^*$ has a path p from $\neg(L \cap \{l^*\})$ to $\neg l$. As p is a path in F, it includes at least one clause of SP (recall that SP is a separator with respect to $\neg(L \cap \{l^*\})$ and $\neg l$ in F). Let $C = (l_1 \vee l_2)$ be the last clause of SP as we traverse p from $\neg(L \cap \{l^*\})$ to $\neg l$ and assume without loss of generality that l_1 is the main literal of C with respect to $(F \setminus X^*, L \cup \{l^*\}, l)$ (and hence with respect to $(F, L \cup \{l^*\}, l)$). Let p^* be the suffix of p starting at C.

According to Corollary 2, l_1 is the first literal of p^*. In the next paragraph we will show that no clause of R follows C in p^*. Combining this statement with the observation that the clauses of $F \setminus X^*$ can be partitioned into R, $SP \setminus XSP$ and $NR \setminus XNR$ (the rest of the clauses belong to X^*) we conclude that p^* is a walk witnessing that $C \in Y$. But this is a contradiction because by definition $Y \subseteq X^*$. This contradiction will complete the proof of the present claim.

Assume by contradiction that C is followed in p^* by a clause $C' = (l'_1 \vee l'_2)$ of R (we assume, without loss of generality, that l'_1 is the main literal of C' with respect to $(F \setminus X^*, L \cup \{l^*\}, l)$). Let p' be a suffix of p^* starting at C'. It follows from Corollary 2 that the first literal of p' is l'_1. By definition of R and taking into account that $R \cap X^* = \emptyset$, $F \setminus X^*$ has a walk w_1 from $\neg L$ whose last clause is C' and all clauses of which belong to R. By Corollary 2, the last literal of w_1 is l'_2. Therefore we can replace C' by w_1 in p'. As a result we get a walk w_2 from $\neg L$ to $\neg l$ in $F \setminus X^*$. By Lemma 3, there is a path p_2 from $\neg L$ to $\neg l$ whose set of clauses is a subset of the set of clauses of w_2. As p_2 is also a path of F, it includes a clause of SP. However, w_1 does not include any clause of SP by definition. Therefore, p' includes a clause of SP. Consequently, p^* includes a clause of SP following C in contradiction to the selection of C. This contradiction shows that clause C' does not exist, which completes the proof of the present claim as noted in the previous paragraph. □

Claim 3. *$F \setminus X^*$ has no walk from $\neg l$ to $\neg l$.*

Proof. Assume by contradiction that $F \setminus X^*$ has a walk w from $\neg l$ to $\neg l$. By definition of X and Theorem 1, w contains at least one clause of X. Since XSP and XNR are subsets of X^*, w contains a clause $C' = (l'_1 \vee l'_2)$ of XR. Assume w.l.o.g. that l'_1 is the main literal of C' with respect to (F, L, l). If l'_1 is a first literal of C' with respect to w then let w^* be a suffix of w whose first clause is C' and first literal is l'_1. Otherwise, let w^* be a suffix of *reverse*(w) having the same properties. In any case, w^* is a walk from l'_1 to $\neg l$ in $F \setminus X^*$ whose first clause is C'. Arguing as in the last paragraph of proof of Claim 2, we see that $F \setminus X^*$ has a walk w_1 from $\neg L$ to l'_2 whose last clause is C'. Therefore we can replace C' by w_1 in w^* and get a walk w_2 from $\neg L$ to $\neg l$ in $F \setminus X^*$ in contradiction to Claim 2. This contradiction shows that our initial assumption regarding the existence of w is incorrect and hence completes the proof of the present claim. □

[1] This replacement is valid because the replacing walk and the walk being replaced have the same first literal.

It follows from the combination of Theorem 1, Claim 2, and Claim 3 that X^* is a CS of $(F, L \cup \{l^*\}, l)$, which completes the proof of the present theorem. □

4. Algorithm for the parameterized 2-ASLASAT problem

4.1. The algorithm

We present our algorithm for the parameterized 2-ASLASAT, FINDCS below. We formally analyze the algorithm in the following sections.

FINDCS(F, L, l, k)
Input: An instance (F, L, l, k) of the parameterized 2-ASLASAT problem.
Output: A CS of (F, L, l) of size at most k if one exists. Otherwise 'NO' is returned.

1. **if** SWRT$(F, L \cup \{l\})$ is true **then** return $\emptyset$
2. **if** $k = 0$ **then** Return 'NO'
3. **if** $k \geqslant |Clauses(F)|$ **then** return $Clauses(F)$
4. **if** $SepSize(F, \neg L, \neg l) > k$ **then** return 'NO'[2]
5. **if** F has a walk from $\neg L$ to $\neg l$ **then**
   Let $C = (l_1 \vee l_2)$ be a clause such that $l_1 \in \neg L$ and $Var(l_2) \notin Var(L)$
6. **else** Let $C = (l_1 \vee l_2)$ be a clause which belongs to a walk of F from $\neg l$ to $\neg l$ and SWRT$(F, \{l_1, l_2\})$ is true[3]
7. **if** Both l_1 and l_2 belong to $\neg(L \cup \{l\})$ **then**
   7.1 $S \leftarrow$ FINDCS$(F \setminus C, L, l, k-1)$
   7.2 **if** S is not 'NO' **then** Return $S \cup \{C\}$
   7.3 Return 'NO'
8. **if** Both l_1 and l_2 do not belong to $\neg(L \cup \{l\})$ **then**
   8.1 $S_1 \leftarrow$ FINDCS$(F, L \cup \{l_1\}, l, k)$
   8.2 **if** S_1 is not 'NO' **then** Return S_1
   8.3 $S_2 \leftarrow$ FINDCS$(F, L \cup \{l_2\}, l, k)$
   8.4 **if** S_2 is not 'NO' **then** Return S_2
   8.5 $S_3 \leftarrow$ FINDCS$(F \setminus C, L, l, k-1)$
   8.6 **if** S_3 is not 'NO' **then** Return $S_3 \cup \{C\}$
   8.7 Return 'NO'
   (In the rest of the algorithm we consider the cases where exactly one literal of C belongs to $\neg(L \cup \{l\})$. Without loss of generality, we assume that this literal is l_1.)
9. **if** l_2 is not neutral in (F, L, l)
   9.1 $S_2 \leftarrow$ FINDCS$(F, L \cup \{l_2\}, l, k)$
   9.2 **if** S_2 is not 'NO' **then** Return S_2
   9.3 $S_3 \leftarrow$ FINDCS$(F \setminus C, L, l, k-1)$
   9.4 **if** S_3 is not 'NO' **then** Return $S_3 \cup \{C\}$
   9.5 Return 'NO'
10. Return FINDCS$(F, L \cup \{l_2\}, l, k)$

4.2. Additional terminology and auxiliary lemmas

In order to analyze the above algorithm, we extend our terminology. Let us call a quadruple (F, L, l, k) a *valid input* if (F, L, l, k) is a valid instance of the parameterized 2-ASLASAT problem (as specified in Section 2.3).

Now we introduce the notion of the *search tree* $ST(F, L, l, k)$ produced by FINDCS(F, L, l, k). The root of the tree corresponds to (F, L, l, k). If FINDCS(F, L, l, k) does not apply itself recursively then (F, L, l, k) is the only node of the tree. Otherwise the children of (F, L, l, k) correspond to the inputs of the calls applied *within* the call FINDCS(F, L, l, k). For example, if FINDCS(F, L, l, k) performs Step 9 then the children of (F, L, l, k) are $(F, L \cup \{l_2\}, l, k)$ and $(F \setminus C, L, l, k-1)$. For each child (F', L', l', k') of (F, L, l, k), the subtree of $ST(F, L, l, k)$ rooted by (F', L', l', k') is $ST(F', L', l', k')$. It is clear from the description of FINDCS that the third item of a valid input is not changed for its children hence in the rest of this section when we denote a child or descendant of (F, L, l, k) we will leave the third item unchanged, e.g. (F_1, L_1, l, k_1).

Lemma 4. *Let (F, L, l, k) be a valid input. Then* FINDCS(F, L, l, k) *succeeds to select a clause in Steps* 5 *and* 6.

Proof. Assume that F has a walk from $\neg L$ to $\neg l$ and let w be the shortest possible such walk. Let l_1 be the first literal of w and let $C = (l_1 \vee l_2)$ be the first clause of F. By definition $l_1 \in \neg L$. We claim that $Var(l_2) \notin Var(L)$. Indeed, assume that this is not true. If $l_2 \in \neg L$ then SWRT$(F, \{\neg l_1, \neg l_2\})$ is false and hence SWRT(F, L) is false as L is a superset of $\{\neg l_1, \neg l_2\}$. But this contradicts the definition of the 2-ASLASAT problem. Assume now that $l_2 \in L$. By definition of the 2-ASLASAT problem, $Var(l) \notin Var(L)$, hence C is not the last clause of w. Consequently the first literal of the second clause of w belongs to $\neg L$. Thus if we remove the first clause from w we obtain a shorter walk from $\neg L$ to $\neg l$ in contradiction to the definition of w. It follows that our claim is true and the required clause C can be selected if the condition of Step 5 is satisfied.

[2] The correctness of this step follows from Corollary 1.
[3] We will prove that in Steps 5 and 6 F has at least one clause with the required property.

Consider now the case where the condition of Step 5 is not satisfied. Note that $\text{SWRT}(F, L \cup \{l\})$ is false because otherwise the algorithm would have finished at Step 1. Consequently by Theorem 1, F has a walk from $\neg l$ to $\neg l$. We claim that any such walk w contains a clause $C = (l_1 \vee l_2)$ such that $\text{SWRT}(F, \{l_1, l_2\})$ is true. Let P be a satisfying assignment of F (which exists by definition of the 2-ASLASAT problem). Let F' be the 2-CNF formula created by the clauses of w and let P' be the subset of P such that $Var(P') = Var(F')$. By Lemma 1, $\text{SWRT}(F', l)$ is false and hence, taking into account that $Var(l) \in Var(F')$, $\neg l \in P'$. Consequently $l \in \neg P'$. Therefore $\neg P'$ is not a satisfying assignment of F', i.e. $\neg P'$ does not satisfy at least one clause of F'. Taking into account that $Var(\neg P') = Var(F')$, it contains negations of both literals of at least one clause C of F'. Therefore P' (and hence P) contains both literals of C. Clearly, C is the required clause. □

The soundness of Steps 5 and 6 of FindCS is assumed in the rest of the paper without explicitly referring to Lemma 4.

Lemma 5. *Let (F, L, l, k) be a valid input and assume that* FindCS(F, L, l, k) *applies itself recursively. Then all the children of (F, L, l, k) in the search tree are valid inputs for* FindCS.

Proof. Let (F_1, L_1, l, k_1) be a child of (F, L, l, k). Observe that $k_1 \leqslant k - 1$. Observe also that $k > 0$ because FindCS(F, L, l, k) would not apply itself recursively if $k = 0$. It follows that $k_1 \geqslant 0$.

It remains to be proven that (F_1, L_1, l) is a valid instance of the 2-ASLASAT problem. If $k_1 = k - 1$ then $(F_1, L_1, l) = (F \setminus C, L, l)$ where C is the clause selected in Steps 5 and 6. In this case the validity of instance $(F \setminus C, L, l)$ immediately follows from the validity of (F, L, l). Consider the remaining case where $(F_1, L_1, l, k_1) = (F, L \cup \{l^*\}, l, k)$ where l^* is a literal of the clause $C = (l_1 \vee l_2)$ selected in Steps 5 and 6. In particular, we are going to show that

- $L \cup \{l^*\}$ is non-contradictory;
- $Var(l) \notin Var(L \cup \{l^*\})$; and
- $\text{SWRT}(F, L \cup \{l^*\})$ is true.

That $L \cup \{l^*\}$ is non-contradictory follows from the description of the algorithm because it is explicitly stated that the literal being joined to L does not belong to $\neg(L \cup \{l\})$. This also implies that the second condition may be violated only if $l^* = l$. In this case assume that C is selected in Step 5. Then, without loss of generality, $l_1 \in \neg L$ and $l_2 = l$. Let P be a satisfying assignment of F which does not intersect with $\neg L$ (which exists since $\text{SWRT}(F, L)$ is true). Then $l_2 \in P$, i.e. $\text{SWRT}(F, L \cup \{l\})$ is true, which is impossible since in this case the algorithm would stop at Step 1. The assumption that C is selected in Step 6 also leads to a contradiction because on the one hand $\text{SWRT}(F, l)$ is false by Lemma 1 due to existence of a walk from $\neg l$ to $\neg l$, on the other hand $\text{SWRT}(F, l)$ is true by the selection criterion. It follows that $Var(l) \notin Var(L \cup \{l^*\})$.

Let us prove the last item. Assume first that C is selected on Step 5 and assume, without loss of generality, that $l_1 \in \neg L$. Then, by the first item, $l^* = l_2$. Moreover, as noted in the previous paragraph $l_2 \in P$ where P is a satisfying assignment of F which does intersect with $\neg L$, i.e. $\text{SWRT}(F, L \cup \{l_2\})$ is true in the considered case. Assume that C is selected in Step 6 and let w be the walk from $\neg l$ to $\neg l$ in F to which C belongs. Observe that F has a walk w' from l^* to $\neg l$: if l^* is a first literal of C with respect to w then let w' be a suffix of w whose first literal is l^*, otherwise let be the suffix of $reverse(w)$ whose first literal is l^*. Assume that $\text{SWRT}(F, L \cup \{l^*\})$ is false. Since $L \cup \{l^*\}$ is non-contradictory by the first item, $Var(l^*) \notin Var(L)$. It follows that (F, L, l^*) is a valid instance of the 2-ASLASAT problem. In this case, by Theorem 1, F has either a walk from $\neg L$ to $\neg l^*$ or a walk from $\neg l^*$ to $\neg l^*$. The latter is ruled out by Lemma 1 because $\text{SWRT}(F, l^*)$ is true by selection of C. Let w'' be a walk from $\neg L$ to $\neg l^*$ in F. Then $w'' + w'$ is a walk of F from $\neg L$ to $\neg l$ in contradiction to our assumption that C is selected in Step 6. Thus $\text{SWRT}(F, L \cup \{l^*\})$ is true. The proof of the present lemma is now complete. □

Now we introduce two measures of the input of the FindCS procedure. Let $\alpha(F, L, l, k) = |Var(F) \setminus Var(L)| + k$ and $\beta(F, L, l, k) = max(0, 2k - SepSize(F, \neg L, \neg l))$.

Lemma 6. *Let (F, L, l, k) be a valid input and let (F_1, L_1, l, k_1) be a child of (F, L, l, k). Then $\alpha(F, L, l, k) > \alpha(F_1, L_1, l, k_1)$.*

Proof. If $k_1 = k - 1$ then the statement is clear because the first item in the definition of the α-measure does not increase and the second decreases. So, assume that $(F_1, L_1, l, k_1) = (F, L \cup \{l^*\}, l, k)$. In this case it is sufficient to prove that $Var(l^*) \notin Var(L)$. Due to the validity of $(F, L \cup \{l^*\}, l, k)$ by Lemma 5, $l^* \notin \neg L$, so it remains to prove that $l^* \notin L$. Assume that $l^* \in L$. Then the clause C is selected in Step 6. Indeed, if C is selected in Step 5 then one of its literals belongs to $\neg L$ and hence cannot belong to L, due to the validity of (F, L, l, k) (and hence being L non-contradictory), while the variable of the other literal does not belong to $Var(L)$ at all. Let w be the walk from $\neg l$ to $\neg l$ in F to which C belongs. Due to the validity of $(F, L \cup \{l^*\}, l, k)$ by Lemma 5, $l^* \neq \neg l$. Therefore either w or $reverse(w)$ has a suffix which is a walk from $\neg l^*$ to $\neg l$, i.e. a walk from $\neg L$ to $\neg l$. But this contradicts the selection of C in Step 6. So, $l^* \notin L$ and the proof of the lemma is complete. □

For the next lemma we extend our terminology. We call a node (F', L', l, k') of $ST(F, L, l, k)$ a *trivial* node if it is a leaf or its only child is of the form $(F', L' \cup \{l^*\}, l, k')$ for some literal l^*.

Lemma 7. *Let (F, L, l, k) be a valid input and let (F_1, L_1, l, k_1) be a child of (F, L, l, k). Then $\beta(F, L, l, k) \geqslant \beta(F_1, L_1, l, k_1)$. Moreover if (F, L, l, k) is a non-trivial node then $\beta(F, L, l, k) > \beta(F_1, L_1, l, k_1)$.*

Proof. Note that $\beta(F, L, l, k) > 0$ because if $\beta(F, L, l, k) = 0$ then FINDCS(F, L, l, k) does not apply itself recursively, i.e. does not have children. It follows that $\beta(F, L, l, k) = 2k - SepSize(F, \neg L, \neg l) > 0$. Consequently, to show that $\beta(F, L, l, k) > \beta(F_1, L_1, l, k_1)$ or that $\beta(F, L, l, k) \geqslant \beta(F_1, L_1, l, k_1)$ it is sufficient to show that $2k - SepSize(F, \neg L, \neg l) > 2k_1 - SepSize(F_1, \neg L_1, \neg l)$ or $2k - SepSize(F, \neg L, \neg l) \geqslant 2k_1 - SepSize(F_1, \neg L_1, \neg l)$, respectively.

Assume first that $(F_1, L_1, l, k_1) = (F \setminus C, L, l, k - 1)$. Observe that $SepSize(F \setminus C, \neg L, \neg l) \geqslant SepSize(F, \neg L, \neg l) - 1$. Indeed assume the opposite and let S be a separator with respect to $\neg L$ and $\neg l$ in $F \setminus C$ whose size is at most $SepSize(F, \neg L, \neg l) - 2$. Then $S \cup \{C\}$ is a separator with respect to $\neg L$ and $\neg l$ in F of size at most $SepSize(F, \neg L, \neg l) - 1$ in contradiction to the definition of $SepSize(F, \neg L, \neg l)$. Thus $2(k-1) - SepSize(F \setminus C, \neg L, \neg l) = 2k - SepSize(F \setminus C, \neg L, \neg l) - 2 \leqslant 2k - SepSize(F, \neg L, \neg l) - 1 < 2k - SepSize(F, \neg L, \neg l)$.

Assume now that $(F_1, L_1, l, k_1) = (F, L \cup \{l^*\}, l, k)$ for some literal l^*. Clearly, $SepSize(F, \neg L, \neg l) \leqslant SepSize(F, \neg(L \cup \{l^*\}), \neg l)$ due to $\neg L$ being a subset of $\neg(L \cup \{l^*\})$. It follows that $2k - SepSize(F, \neg L, \neg l) \geqslant 2k - SepSize(F, \neg(L \cup \{l^*\}), \neg l)$. It remains to show that $\geqslant$ can be replaced by $>$ in the case where (F, L, l, k) is a non-trivial node. It is sufficient to show that in this case $SepSize(F, \neg L, \neg l) < SepSize(F, \neg(L \cup \{l^*\}), \neg l)$. If (F, L, l, k) is a non-trivial node then the recursive call FINDCS$(F, L \cup \{l^*\}, l, k)$ is applied in Steps 8.1, 8.3, or 9.1. In the last case, it is explicitly said that l^* is not a neutral literal in (F, L, l). Consequently, $SepSize(F, \neg L, \neg l) < SepSize(F, \neg(L \cup \{l^*\}), \neg l)$ by definition.

For the first two cases note that Step 8 is applied only if the clause C is selected in Step 6. That is, F has no walk from $\neg L$ to $\neg l$. In particular, F has no path from $\neg L$ to $\neg l$, i.e. $SepSize(\neg L, \neg l) = 0$. Let w be the walk from $\neg l$ to $\neg l$ in F to which C belongs. Note that by Lemma 5, $(F, L \cup \{l^*\}, l, k)$ is a valid input, in particular $Var(l^*) \neq Var(l)$. Therefore either w or $reverse(w)$ has a suffix which is a walk from $\neg l^*$ to $\neg l$, i.e. a walk from $\neg(L \cup \{l^*\})$ to $\neg l$. Applying Lemma 3 together with Lemma 5, we see that F has a path from $\neg(L \cup \{l^*\})$ to $\neg l$, i.e. $SepSize(F, \neg(L \cup \{l^*\}, \neg l) > 0$. □

Lemma 8. *Let (F, L, l, k) be a valid input. Then the following statements are true regarding $ST(F, L, l, k)$.*

- *The height of $ST(F, L, l, k)$ is at most $\alpha(F, L, l, k)$.*[4]
- *Each non-root node (F', L', l, k') of $ST(F, L, l, k)$ is a valid input, the subtree rooted by (F', L', l, k') is $ST(F', L', l, k')$ and $\alpha(F', L', l, k') < \alpha(F, L, l, k)$.*
- *For each node (F', L', l, k') of $ST(F, L, l, k)$, $\beta(F', L', l, k') \leqslant \beta(F, L, l, k) - t$ where t is the number of non-trivial nodes besides (F', L', l, k') in the path from (F, L, l, k) to (F', L', l, k') of $ST(F, L, l, k)$.*

Proof. This lemma is clearly true if (F, L, l, k) has no children. Consequently, it is true if $\alpha(F, L, l, k) = 0$. Now, apply induction on $\alpha(F, L, l, k)$ and assume that $\alpha(F, L, l, k) > 0$. By the induction assumption, Lemma 5, and Lemma 6, the present lemma is true for any child of (F, L, l, k). Consequently, for any child (F^*, L^*, l, k^*) of (F, L, l, k), the height of $ST(F^*, L^*, l, k^*)$ is at most $\alpha(F^*, L^*, l, k^*)$. Hence the first statement follows by Lemma 6. Furthermore, any non-root node (F', L', l, k') of $ST(F, L, l, k)$ belongs to $ST(F^*, L^*, l, k^*)$ of some child (F^*, L^*, l, k^*) of (F, L, l, k) and the subtree rooted by (F', L', l, k') in $ST(F, L, l, k)$ is the subtree rooted by (F', L', l, k') in $ST(F^*, L^*, l, k^*)$. Consequently, (F', L', l, k') is a valid input, the subtree rooted by it is $ST(F', L', l, k')$, and $\alpha(F', L', l, k') \leqslant \alpha(F^*, L^*, l, k^*) < \alpha(F, L, l, k)$, the last inequality follows from Lemma 6. Finally, $\beta(F', L', l, k') \leqslant \beta(F^*, L^*, l, k^*) - t^*$ where t^* is the number of non-trivial nodes besides (F', L', l, k') in the path from (F^*, L^*, l, k^*) to (F', L', l, k') in $ST(F^*, L^*, l, k^*)$, and hence in $ST(F, L, l, k)$.[5] If (F, L, l, k) is a trivial node then $t = t^*$ and the last statement of the present lemma is true by Lemma 7. Otherwise $t = t^* + 1$ and by another application of Lemma 7 we get that $\beta(F', L', l, k') \leqslant \beta(F, L, l, k) - t^* - 1 = \beta(F, L, l, k) - t$. □

4.3. Correctness proof

Theorem 4. *Let (F, L, l, k) be a valid input. Then* FINDCS(F, L, l, k) *correctly solves the parameterized* 2-ASLASAT *problem. That is, if* FINDCS(F, L, l, k) *returns a set, this set is a CS of (F, L, l) of size at most k. If* FINDCS(F, L, l, k) *returns 'NO' then (F, L, l) has no CS of size at most k.*

Proof. Let us prove first the correctness of FINDCS(F, L, l, k) for the cases when the procedure does not apply itself recursively. It is only possible when the procedure returns an answer in Steps 1–4. If the answer is returned in Step 1 then the validity is clear because nothing has to be removed from F to make it satisfiable with respect to L and l. If the answer is returned in Step 2 then SWRT$(F, L \cup \{l\})$ is false (since the condition of Step 1 is not satisfied) and consequently the size of a CS of (F, L, l) is at least 1. On the other hand, $k = 0$ and hence the answer 'NO' is valid in the considered case. For the answer returned in Step 3 observe that $Clauses(F)$ is clearly a CS of (F, L, l) (since SWRT$(\emptyset, L \cup \{l\})$ is true) and the size of

[4] Besides providing the upper bound on the height of $ST(F, L, l, k)$, this statement claims that $ST(F, L, l, k)$ is finite and hence we may safely refer to a path between two nodes.

[5] Note that this inequality applies to the case where $(F', L', l, k') = (F^*, L^*, l, k^*)$.

Clauses(F) does not exceed k by the condition of Step 3. Therefore, the answer returned on this step is valid. Finally if the answer is returned in Step 4 then the condition of Step 4 is satisfied. According to Corollary 1, this condition implies that any CS of (F, L, l) has the size greater than k, which justifies the answer 'NO' in the considered step.

Now we prove the correctness of FINDCS(F, L, l, k) by induction on $\alpha(F, L, l, k)$. Assume first that $\alpha(F, L, l, k) = 0$. Then it follows that $k = 0$ and, consequently, FINDCS(F, L, l, k) does not apply itself recursively (the output is returned in Step 1 or Step 2). Therefore, the correctness of FINDCS(F, L, l, k) follows from the previous paragraph. Assume now that $\alpha(F, L, l, k) > 0$ and that the theorem holds for any valid input (F', L', l, k') such that $\alpha(F', L', l, k') < \alpha(F, L, l, k)$. Due to the previous paragraph we may assume that FINDCS(F, L, l, k) applies itself recursively, i.e. the node (F, L, l, k) has children in $ST(F, L, l, k)$.

Claim 4. *Let (F_1, l_1, l, k_1) be a child of (F, L, l, k). Then* FINDCS(F_1, L_1, l, k_1) *is correct.*

Proof. By Lemma 5, (F_1, L_1, l, k_1) is a valid input. By Lemma 6, $\alpha(F_1, L_1, l, k_1) < \alpha(F, L, l, k)$. The claim follows by the induction assumption. □

Assume that FINDCS(F, L, l, k) returns a set S. By the description of the algorithm, either S is returned by FINDCS$(F, L \cup \{l^*\}, l, k)$ for a child $(F, L \cup \{l^*\}, l, k)$ of (F, L, l, k) or $S = S_1 \cup \{C\}$ and S_1 is returned by FINDCS$(F \setminus C, L, l, k-1)$ for a child $(F \setminus C, L, l, k-1)$ of (F, L, l, k). In the former case, the validity of output follows from Claim 4 and from the easy observation that a CS of $(F, L \cup \{l^*\}, l, k)$ is a CS of (F, L, l, k) because L is a subset of $L \cup \{l^*\}$. In the latter case, it follows from Claim 4 that $|S_1| \leqslant k-1$ and that S_1 is a CS of $(F \setminus C, L, l)$, i.e. SWRT$((F \setminus C) \setminus S_1, L \cup \{l\})$ is true. But $(F \setminus C) \setminus S_1 = F \setminus (S_1 \cup \{C\}) = F \setminus S$. Consequently, S is a CS of (F, L, l) of size at most k, hence the output is valid in the considered case.

Consider now the case where FINDCS(F, L, l, k) returns 'NO' and assume by contradiction that there is a CS S of (F, L, l) of size at most k. Assume first that 'NO' is returned in Step 7.3. It follows that $C \notin S$ because otherwise $S \setminus C$ is a CS of $(F \setminus C, L, l)$ of size at most $k-1$ and hence, by Claim 4, the recursive call of Step 7.2 would not return 'NO'. However, this means that no satisfying assignment of $F \setminus S$ disjoint with $\neg(L \cup \{l\})$ (which exists by definition) can satisfy clause C, a contradiction. Assume now that 'NO' is returned in Step 10. By Claim 4, $(F, L \cup \{l_2\}, l)$ has no CS of size at most k. Therefore, according to Theorem 3, the size of a SCS of (F, L, l) is at least $k+1$ which contradicts the existence of S. Finally, assume that 'NO' is returned in Step 8.7 or in Step 9.5. Assume first that the clause C selected in Steps 5 and 6 does not belong to S. Let P be a satisfying assignment of $(F \setminus S)$ which does not intersect with $\neg(L \cup \{l\})$. Then at least one literal l^* of C is contained in P. This literal does not belong to $\neg(L \cup \{l\})$ and hence FINDCS$(F, L \cup \{l^*\}, l, k)$ has been applied and returned 'NO'. However, P witnesses that S is a CS of $(F, L \cup \{l^*\}, l, k)$ of size at most k, that is FINDCS$(F, L \cup \{l^*\}, l, k)$ returned an incorrect answer in contradiction to Claim 4. Finally assume that $C \in S$. Then $S \setminus C$ is a CS of $(F \setminus C, L, l)$ of size at most $k-1$ and hence answer 'NO' returned by FINDCS$(F \setminus C, L, l)$ contradicts Claim 4. Thus the answer 'NO' returned by FINDCS(F, L, l, k) is valid. □

4.4. Evaluation of the runtime

Theorem 5. *Let (F, L, l, k) be a valid input. Then the number of leaves of $ST(F, L, l, k)$ is at most $\sqrt{5}^t$, where $t = \beta(F, L, l, k)$.*

Proof. Since $\beta(F, L, l, k) \geqslant 0$ by definition, $\sqrt{5}^t \geqslant 1$. Hence if FINDCS(F, L, l, k) does not apply itself recursively, i.e. $ST(F, L, l, k)$ has only one node, the theorem clearly holds. We prove the theorem by induction on $\alpha(F, L, l, k)$. If $\alpha(F, L, l, k) = 0$ then as we have shown in the proof of Theorem 4, FINDCS(F, L, l, k) does not apply itself recursively and hence the theorem holds as shown above. Assume that $\alpha(F, L, l, k) > 0$ and that the theorem holds for any valid input (F', L', l, k') such that $\alpha(F', L', l, k') < \alpha(F, L, l, k)$. Clearly we may assume that (F, L, l, k) applies itself recursively, i.e. $ST(F, L, l, k)$ has more than 1 node.

Claim 5. *For any non-root node (F', L', l, k') of $ST(F, L, l, k)$, the subtree of $ST(F, L, l, k)$ rooted by (F', L', l, k') has at most $\sqrt{5}^{t'}$ leaves, where $t' = \beta(F', L', l, k')$.*

Proof. According to Lemma 8, (F', L', l, k') is a valid input, $\alpha(F', L', l, k') < \alpha(F, L, l, k)$, and the subtree of $ST(F, L, l, k)$ rooted by (F', L', l, k') is $ST(F', L', l, k')$. Therefore the claim follows by the induction assumption. □

If (F, L, l, k) has only one child (F_1, L_1, l, k_1) then clearly the number of leaves of $ST(F, L, l, k)$ equals the number of leaves of the subtree rooted by (F_1, L_1, l, k_1) which, by Claim 5, is at most $\sqrt{5}^{t_1}$, where $t_1 = \beta(F_1, L_1, l, k_1)$. According to Lemma 7, $t_1 \leqslant t$ so the present theorem holds for the considered case. If (F, L, l, k) has 2 children (F_1, L_1, l, k_1) and (F_2, L_2, l, k_2) then the number of leaves of $ST(F, L, l, k)$ is the sum of the numbers of leaves of subtrees rooted by (F_1, L_1, l, k_1) and (F_2, L_2, l, k_2) which, by Claim 5, is at most $\sqrt{5}^{t_1} + \sqrt{5}^{t_2}$, where $t_i = \beta(F_i, L_i, l, k_i)$ for $i = 1, 2$. Taking into account that (F, L, l, k) is a non-trivial node and applying Lemma 7, we get that $t_1 < t$ and $t_2 < t$. Hence the number of leaves of $ST(F, L, l, k)$ is at most $(2/\sqrt{5}) \times (\sqrt{5}^t) < \sqrt{5}^t$, so the theorem holds for the considered case as well.

For the case where (F, L, l, k) has 3 children, denote them by (F_i, L_i, l, k_i), $i = 1, 2, 3$. Assume, without loss of generality, that $(F_1, L_1, l, k_1) = (F, L \cup \{l_1\}, l, k)$, $(F_2, L_2, l, k_2) = (F, L \cup \{l_2\}, l, k)$, $(F_3, L_3, l, k_3) = (F \setminus C, l, k-1)$, where $C = (l_1 \vee l_2)$ is the clause selected in Steps 5 and 6. Let $t_i = \beta(F_i, L_i, l, k_i)$ for $i = 1, 2, 3$.

Claim 6. *$t \geqslant 2$ and $t_3 \leqslant t - 2$.*

Proof. Note that $k > 0$ because otherwise FindCS(F, L, l, k) does not apply itself recursively. Observe also that $SepSize(F, \neg L, \neg l) = 0$ because clause C can be selected only in Step 6, which means that F has no walk from $\neg L$ to $\neg l$ and, in particular, F has no path from $\neg L$ to $\neg l$. Therefore $2k - Sepsize(F, \neg L, \neg l) = 2k \geqslant 2$ and hence $t = \beta(F, L, l, k) = 2k \geqslant 2$. If $t_3 = 0$ the second statement of the claim is clear. Otherwise $t_3 = 2(k-1) - SepSize(F \setminus (l_1 \vee l_2), \neg L, \neg l) = 2(k-1) - 0 = 2k - 2 = t - 2$. □

Assume that some $ST(F_i, L_i, l, k_i)$ for $i = 1, 2$ has only one leaf. Assume, without loss of generality, that this is $ST(F_1, L_1, l, k_1)$. Then the number of leaves of $ST(F, L, l, k)$ is the sum of the numbers of leaves of the subtrees rooted by (F_2, L_2, l, k_2) and (F_3, L_3, l, k_3) plus one. By Claims 5 and 6, and Lemma 7, this is at most $\sqrt{5}^{t-1} + \sqrt{5}^{t-2} + 1$. Then $\sqrt{5}^{t} - \sqrt{5}^{t-1} - \sqrt{5}^{t-2} - 1 \geqslant \sqrt{5}^{2} - \sqrt{5}^{2-1} - \sqrt{5}^{2-2} - 1 = 5 - \sqrt{5} - 2 > 0$, the first inequality follows from Claim 6. That is, the present theorem holds for the considered case.

It remains to assume that both $ST(F_1, L_1, l, k_1)$ and $ST(F_2, L_2, l, k_2)$ have at least two leaves. Then for $i = 1, 2$, $ST(F_i, L_i, l, k_i)$ has a node having at least two children. Let (FF_i, LL_i, l, kk_i) be such a node of $ST(F_i, L_i, l, k_i)$ which lies at *the smallest distance* from (F, L, l, k) in $ST(F, L, l, k)$.

Claim 7. *The number of leaves of the subtree rooted by (FF_i, LL_i, l, kk_i) is at most $(2/5) \times \sqrt{5}^{t}$.*

Proof. Assume that (FF_i, LL_i, l, kk_i) has 2 children and denote them by $(FF_1^*, LL_1^*, l, kk_1^*)$ and $(FF_2^*, LL_2^*, l, kk_2^*)$. Then the number of leaves of the subtree rooted by (FF_i, LL_i, l, kk_i) equals the sum of numbers of leaves of the subtrees rooted by $(FF_1^*, LL_1^*, l, kk_1^*)$ and $(FF_2^*, LL_2^*, l, kk_2^*)$. By Claim 5, this sum does not exceed $2 \times \sqrt{5}^{t^*}$ where t^* is the maximum of $\beta(FF_j^*, LL_j^*, l, kk_j^*)$ for $j = 1, 2$. Note that the path from (F, L, l, k) to any $(FF_j^*, LL_j^*, l, kk_j^*)$ includes at least 2 non-trivial nodes besides $(FF_j^*, LL_j^*, l, kk_j^*)$, namely (F, L, l, k) and (FF_i, LL_i, l, kk_i). Consequently, $t^* \leqslant t - 2$ by Lemma 8 and the present claim follows for the considered case.

Assume that (FF_i, LL_i, l, kk_i) has 3 children. Then let $tt_i = \beta(FF_i, LL_i, l, kk_i)$ and note that according to Claim 5, the number of leaves of the subtree rooted by (FF_i, LL_i, l, kk_i) is at most $\sqrt{5}^{tt_i}$. Taking into account that (FF_i, LL_i, l, kk_i) is a valid input by Lemma 8 and arguing analogously to the second sentence of the proof of Claim 6, we see that $SepSize(FF_i, \neg LL_i, \neg l) = 0$. On the other hand, using the argumentation in the last paragraph of the proof of Lemma 7, we can see that $SepSize(F_i, \neg L_i, \neg l) > 0$. This means that $(F_i, L_i, l, k_i) \neq (FF_i, LL_i, l, kk_i)$. Moreover, the path from (F_i, L_i, l, k_i) to (FF_i, LL_i, l, kk_i) includes a pair of consecutive nodes (F', L', l, k') and (F'', L'', l, k''), the former being the parent of the latter, such that $SepSize(F', \neg L', \neg l) > SepSize(F'', \neg L'', \neg l)$. This only can happen if $k'' = k' - 1$ (for otherwise $(F'', L'', l, k'') = (F', L' \cup \{l'\}, l, k')$ for some literal l' and clearly adding a literal to L' does not decrease the size of the separator). Consequently, (F', L', l, k') is a non-trivial node. Therefore, the path from (F, L, l, k) to (FF_i, LL_i, l, kk_i) includes at least 2 non-trivial nodes besides (FF_i, LL_i, l, kk_i): (F, L, l, k) and (F', L', l, k'). That is $tt_i \leqslant t - 2$ by Lemma 8 and the present claims follow for this case as well which completes its proof. □

It remains to notice that the number of leaves of $ST(F, L, l, k)$ is the sum of the numbers of leaves of subtrees rooted by (FF_1, LL_1, l, kk_1), (FF_2, LL_2, l, kk_2), and (F_3, L_3, l, k_3) which, according to Claims 5, 6 and 7, is at most $5 \times \sqrt{5}^{t-2} = \sqrt{5}^{t}$. □

Theorem 6. *Let (F, L, l, k) be an instance of the parameterized 2-ASLASAT problem. Then the problem can be solved in time $\mathcal{O}(5^k \times k(n+k) \times (m + |L|))$, where $n = |Var(F)|$, $m = |Clauses(F)|$.*

Proof. According to assumptions of the theorem, (F, L, l, k) is a valid input. Assume that (F, L, l, k) is represented by its implication graph $D = (V, A)$ which is almost identical to the implication graph of F with the only difference that $V(D)$ corresponds to $Var(F) \cup Var(L) \cup Var(l)$, that is if for any literal l' such that $Var(l') \in (Var(L) \cup \{Var(l)\}) \setminus Var(F)$, D has isolated nodes corresponding to l' and $\neg l'$. We also assume that the nodes corresponding to L, $\neg L$, l, $\neg l$ are specifically marked. This representation of (F, L, l, k) can be obtained in polynomial time from any other reasonable representation. It follows from Theorem 4 that FindCS(F, L, l, k) correctly solves the parameterized 2-ASLASAT problem with respect to the given input.

Let us evaluate the complexity of FindCS(F, L, l, k). According to Lemma 8, the height of the search tree is at most $\alpha(F, L, l, k) \leqslant n + k$. Theorem 5 states that the number of leaves of $ST(F, L, l, k)$ is at most $\sqrt{5}^{t}$ where $t = \beta(F, L, l, k)$. Taking into account that $t \leqslant 2k$, the number of leaves of $ST(F, L, l, k)$ is at most 5^k. Consequently, the number of nodes of the search tree is at most $5^k \times (n + k)$. The complexity of FindCS(F, L, l, k) can be represented as the number of nodes multiplied by the complexity of the operations performed *within* the given recursive call.

Let us evaluate the complexity of FindCS(F, L, l, k) without taking into account the complexity of the subsequent recursive calls. First of all note that each literal of F belongs to a clause and each clause contains at most 2 distinct literals. Consequently, the number of clauses of F is at least half of the number of literals of F and, as a result, at least half of the number of variables. Note that this is important because most of the operations of FindCS(F, L, l, k) involve doing Depth-First Search (DFS) or Breadth-First Search (BFS) on graph D, which take $\mathcal{O}(V + A)$. In our case $|V| = \mathcal{O}(n + |L|)$ and $|A| = \mathcal{O}(m)$. Since $n = \mathcal{O}(m)$, $\mathcal{O}(V + A)$ can be replaced by $\mathcal{O}(m + |L|)$.

The first operation performed by FindCS(F, L, l, k) is checking whether SWRT$(F, L \cup \{l\})$ is true. Note that this is equivalent to checking the satisfiability of a 2-CNF F' which is obtained from F by adding clauses $(l' \vee l')$ for each $l' \in L \cup \{l\}$. It is well known [18] that the given 2-CNF formula F' is *not* satisfiable if and only if there are literals l' and $\neg l'$ that belong to the same strongly connected component of the implication graph of F'. The implication graph D' of F' can be obtained from D by adding arcs that correspond to the additional clauses. The resulting graph has $\mathcal{O}(m + |L|)$ vertices and $\mathcal{O}(m + |L|)$ arcs. The partition into the strongly connected components can be done by a constant number of applications of the DFS algorithm. Hence the whole Step 1 takes $\mathcal{O}(m + |L|)$. Steps 2 and 3 take $\mathcal{O}(1)$. According to Theorem 2, Step 4 can be performed by assigning all the arcs of D a unit flow, contracting all the vertices of L into a source s, identifying $\neg l$ with the sink t, and checking whether $k + 1$ units of flow can be delivered from s to t. This can be done by $\mathcal{O}(k)$ iterations of the Ford–Fulkerson algorithm, where each iteration is a run of BFS and hence can be performed in $\mathcal{O}(m + |L|)$. Consequently, Step 4 can be performed in $\mathcal{O}((m + |L|) \times k)$. Checking the condition of Step 5 can be done by BFS and hence takes $\mathcal{O}(m + |L|)$. Moreover, if the required walk exists, BFS finds the shortest one and, as noted in the proof of Lemma 4, a required clause is the first clause of this walk. Hence, the whole Step 5 can be performed in $\mathcal{O}(m + |L|)$. The proof of Lemma 4 also outlines an algorithm implementing Step 6: choose an arbitrary walk w from $\neg l$ to $\neg l$ in F (which, as noted in the proof of Theorem 2, corresponds to a walk from l to $\neg l$ in D), find a satisfying assignment P of F which does not intersect with $\neg L$ and choose a clause of w whose both literals are satisfied by P. Taking into account the above discussion, all the operations take $\mathcal{O}(m + |L|)$, hence Step 6 takes this time. Note that preparing an input for a recursive call takes $\mathcal{O}(1)$ because this preparation includes removal of one clause from F or adding one literal to L (with introducing appropriate changes to the implication graph). Therefore Steps 7 and 8 take $\mathcal{O}(1)$. Step 9 takes $\mathcal{O}((m + |L|) \times k)$ on the account of neutrality checking: $\mathcal{O}(k)$ iterations of the Ford–Fulkerson algorithm are sufficient because $SepSize(F, \neg L, \neg l) \leqslant k$ due to dissatisfaction of the condition of Step 4. Step 10 takes $\mathcal{O}(1)$ on the account of input preparation for the recursive call. Thus the complexity of processing (F, L, l, k) is $\mathcal{O}((m + |L|) \times k)$.

Finally, note that for any subsequent recursive call (F', L', l, k') the implication graph of (F', L', l) is a subgraph of the graph of (F, L, l): every change in the implication graph on the path from (F, L, l, k) to (F', L', l, k') is caused by the removal of a clause or adding to the second parameter a literal of a variable of F. Consequently, the complexity of any recursive call is $\mathcal{O}((m + |L|) \times k)$ and the time taken by the entire run of FindCS(F, L, l, k) is $\mathcal{O}(5^k \times k(n + k) \times (m + |L|))$ as required. □

5. Fixed-parameter tractability of 2-ASAT problem

In this section we prove the main result of the paper, fixed-parameter tractability of the 2-ASAT problem.

Theorem 7. *The* 2-ASAT *problem with input (F, k) where F is a* 2-CNF *formula with possible repeated occurrences of clauses,*[6] *can be solved in $\mathcal{O}(15^k \times k \times m^3)$, where m is the number of clauses of F.*

Proof. We introduce the following 2 intermediate problems.

Problem I1
Input: A satisfiable 2-CNF formula F, a non-contradictory set of literals L, a parameter k.
Output: A set $S \subseteq Clauses(F)$ such that $|S| \leqslant k$ and SWRT$(F \setminus S, L)$ is true, if there is such a set S; 'NO' otherwise.

Problem I2
Input: A 2-CNF formula F, a parameter k, and a set $S \subseteq Clauses(F)$ such $|S| = k + 1$ and $F \setminus S$ is satisfiable.
Output: A set $Y \subseteq Clauses(F)$ such that $|Y| < |S|$ and $F \setminus Y$ is satisfiable, if there is such a set Y; 'NO' otherwise.

The following two claims prove the fixed-parameter tractability of Problem I1 by transforming an arbitrary instance of it into an instance of the 2-ASLASAT problem, and of Problem I2 by transforming an arbitrary instance of Problem I2 into an instance of Problem I1. Then we will show that the 2-ASAT problem, with no repeated occurrence of clauses, can be solved using a transformation into Problem I2. Finally, we show that the 2-ASAT problem, with repeated occurrences of clauses, is FPT by transforming it to the 2-ASAT problem without repeated occurrences of clauses.

Claim 8. *Problem* I1 *with the input (F, L, k) can be solved in $\mathcal{O}(5^k \times k \times m^2)$, where, $m = |Clauses(F)|$.*

[6] The case where repeated occurrences of clauses are allowed is considered in the last three paragraphs of the proof. Before that, all 2-CNF formulas are assumed to contain no repeated occurrences of clauses.

Proof. Observe that we may assume that $Var(L) \subseteq Var(F)$. Otherwise we can take a subset L' such that $Var(L') = Var(F) \cap Var(L)$ and solve Problem I1 with respect to the instance (F, L', k). It is not hard to see that the resulting solution applies to (F, L, k) as well.

Let P be a satisfying assignment of F. If $L \subseteq P$ then the empty set can be immediately returned. Otherwise partition L into two subsets L_1 and L_2 such that $L_1 \subseteq P$ and $\neg L_2 \subseteq P$.

We apply a two stage transformation of formula F. In the first stage we assign each clause of F a unique index from 1 to m, introduce new literals $l_1, \dots, l_m$ of distinct variables which do not intersect with $Var(F)$, and replace the ith clause $(l' \vee l'')$ by two clauses $(l' \vee l_i)$ and $(\neg l_i \vee l'')$. Denote the resulting formula by F'. In the second stage we introduce two new literals l_1^* and l_2^* such that $Var(l_1^*) \notin Var(F')$, $Var(l_2^*) \notin Var(F')$, and $Var(l_1^*) \neq Var(l_2^*)$. Then we replace in the clauses of F' each occurrence of a literal of L_1 by l_1^*, each occurrence of a literal of $\neg L_1$ by $\neg l_1^*$, each occurrence of a literal of L_2 by l_2^*, and each occurrence of a literal of $\neg L_2$ by $\neg l_2^*$. Let F^* be the resulting formula.

We claim that $(F^*, \{l_1^*\}, l_2^*)$ is a valid instance of the 2-ASLASAT problem. To show this we have to demonstrate that all the clauses of F^* are pairwise different and that SWRT(F^*, l_1^*) is true.

For the former, notice that all the clauses of F^* are pairwise different because each clause is associated with the unique literal l_i or $\neg l_i$. This also allows us to introduce new notation. In particular, we denote the clause of F^* containing l_i by $C(l_i)$ and the clause containing $\neg l_i$ by $C(\neg l_i)$.

For the latter let P^* be a set of literals obtained from P by replacing L_1 by l_1^* and $\neg L_2$ by $\neg l_2^*$. Observe that for each i, P^* satisfies either $C(l_i)$ or $C(\neg l_i)$. Indeed, let $(l' \vee l'')$ be the *origin* of $C(l_i)$ and $C(\neg l_i)$, i.e. the clause which is transformed into $(l' \vee l_i)$ and $(\neg l_i \vee l'')$ in F', then $(l' \vee l_i)$ and $(\neg l_i \vee l'')$ become, respectively, $C(l_i)$ and $C(\neg l_i)$ in F^* (with possible replacement of l' or l'' or both). Since P is a satisfying assignment of F, $l' \in P$ or $l'' \in P$. Assume the former. Then if $C(l_i) = (l' \vee l_i)$, $l' \in P^*$. Otherwise, $l' \in L_1$ or $l' \in \neg L_2$. In the former case $C(l_i) = (l_1^* \vee l_i)$ and $l_1^* \in P^*$ by definition; in the latter case $C(l_i) = (\neg l_2^* \vee l_i)$ and $\neg l_2^* \in P^*$ by definition. So, in all the cases P^* satisfies $C(l_i)$. It can be shown analogously that if $l'' \in P$ then P^* satisfies $C(\neg l_i)$. Now, let P_2^* be a set of literals which includes P^* and for each i exactly one of $\{l_i \neg l_i\}$ selected as follows. If P^* satisfies $C(l_i)$ then $\neg l_i \in P_2^*$. Otherwise $l_i \in P_2^*$. Thus P_2^* satisfies all the clauses of F^*. By definition $l_1^* \in P^* \subseteq P_2^*$. It is also not hard to show that P_2^* is non-contradictory and that $Var(P_2^*) = Var(F^*)$. Thus P_2^* is a satisfying assignment of F^* containing l_1^* which witnesses SWRT(F^*, l_1^*) is true.

We show that there is a set $S \subseteq Clauses(F)$ such that $|S| \leqslant k$ and SWRT$(F \setminus S, L)$ is true if and only if $(F^*, \{l_1^*\}, l_2^*)$ has a CS of size at most k.

Assume that there is a set S as above. Let $S^* \subseteq Clauses(F^*)$ be the set consisting of all clauses $C(l_i)$ such that the clause with index i belongs to S. It is clear that $|S^*| = |S|$. Let us show that S^* is a CS of $(F^*, \{l_1^*\}, l_2^*)$. Let P' be a satisfying assignment of $F \setminus S$ which does not intersect with $\neg L$. Let P_1 be the set of literals obtained from P' by replacing the set of all the occurrences of literals of L_1 by l_1^* and the set of all the occurrences of literals of L_2 by l_2^*.

Observe that for each i, at least one of $\{C(l_i), C(\neg l_i)\}$ either belongs to S^* or is satisfied by P_1. In particular, assume that for some i, $C(l_i) \notin S^*$. Then the origin of $C(l_i)$ and $C(\neg l_i)$ belongs to $F \setminus S$ and it can be shown that P_1 satisfies $C(l_i)$ or $C(\neg l_i)$ similar to the way we have shown that P^* satisfies $C(l_i)$ or $C(\neg l_i)$ three paragraphs above.

For each i, add to P_1 an appropriate l_i or $\neg l_i$ so that the remaining clauses of $F^* \setminus S^*$ are satisfied, let P_2 be the resulting set of literals. Add to P_2 one arbitrary literal of each variable of $Var(F^* \setminus S^*) \setminus Var(P_2)$, taking into account, if needed, that $Var(l_1^*)$ is represented by l_1^* and $Var(l_2^*)$ is represented by l_2^*. It is not hard to see that the resulting set of literals P_3 is a satisfying assignment of $F^* \setminus S^*$, which does not contain $\neg l_1^*$ nor $\neg l_2^*$. It follows that S^* is a CS of $(F^*, \{l_1^*\}, l_2^*)$ of size at most k.

Conversely, let S^* be a CS of $(F^*, \{l_1^*\}, l_2^*)$ of size at most k. Let S be a set of clauses of F such that the clause of index i belongs to S if and only if $C(l_i) \in S^*$ or $C(\neg l_i) \in S^*$. Clearly $|S| \leqslant |S^*|$. Let $S_2^* \subseteq Clauses(F^*)$ be the set of all clauses $C(l_i)$ and $C(\neg l_i)$ such that the clause of index i belongs to S. Since $S^* \subseteq S_2^*$, we can specify a satisfying assignment P_2^* of $F^* \setminus S_2^*$ which does not contain $\neg l_1^*$ nor $\neg l_2^*$.

Let P_2 be a set of literals obtained from P_2^* by the removal of all l_i, $\neg l_i$, the removal of l_1^* and l_2^*, and the addition of all the literals l' of L such that l' or $\neg l'$ appear in the clauses of $F \setminus S$. It is not hard to see that $Var(P_2) = Var(F \setminus S)$ and that P_2 does not intersect with $\neg L$.

To observe that P_2 is a satisfying assignment of $F \setminus S$, note that there is a bijection between the pairs $C(l_i), C(\neg l_i)$ of clauses of $F^* \setminus S_2^*$ and the clauses of $F \setminus S$. In particular, each clause of $F \setminus S$ is the origin of exactly one pair $\{C(l_i), C(\neg l_i)\}$ of $F^* \setminus S_2^*$ in the form described above and each pair $\{C(l_i), C(\neg l_i)\}$ of $F^* \setminus S_2^*$ has exactly one origin in $F \setminus S$.

Now, let $(l' \vee l'')$ be a clause of $F \setminus S$ which is the origin of $C(l_i) = (t' \vee l_i)$ and $C(\neg l_i) = (\neg l_i \vee t'')$ of $F^* \setminus S_2^*$, where $l' = t'$ or t' is the result of replacing l', t'' has the analogous correspondence with l''. By definition of P_2^*, either $t' \in P_2^*$ or $t'' \in P_2^*$. Assume the former. In this case if $l' = t'$ then $l' \in P_2$. Otherwise $t' \in \{l_1^*, l_2^*\}$ and, consequently $l' \in L$. By definition of P_2, $l' \in P_2$. It can be shown analogously that if $t'' \in P_2^*$ then $l'' \in P_2$. It follows that any clause of $F \setminus S$ is satisfied by P_2.

It follows from the above argumentation that Problem I1 with input (F, L, k) can be solved by solving the parameterized 2-ASLASAT problem with input $(F^*, \{l_1^*\}, l_2^*, k)$. In particular, if the output of the 2-ASLASAT problem on $(F^*, \{l_1^*\}, l_2^*, k)$ is a set S^*, this set can be transformed into S as shown above and S can be returned; otherwise 'NO' is returned. Observe that $|Clauses(F^*)| = \mathcal{O}(m)$ and $|Var(F^*)| = \mathcal{O}(m + |Var(F)|)$. Taking into account our note in the proof of Theorem 6 that $|Var(F)| = \mathcal{O}(m)$, $|Var(F^*)| = \mathcal{O}(m)$. Also note that we may assume that $k < m$ because otherwise the algorithm can immediately returns $Clauses(F^*)$.

Substituting this data into the runtime of 2-ASLASAT problem following from Theorem 6, we obtain that Problem I1 can be solved in time $\mathcal{O}(5^k \times k \times m \times (m + |\{l_1^*\}|)) = \mathcal{O}(5^k \times k \times m^2)$. □

Claim 9. *Problem* I2 *with input* (F, S, k) *can be solved in time* $\mathcal{O}(15^k \times k \times m^2)$*, where,* $m = |Clauses(F)|$.

Proof. We solve Problem I2 using the following algorithm. Explore all possible subsets E of S of size at most k. For the given set E explore all the sets of literals L obtained by choosing l_1 or l_2 for each clause $(l_1 \vee l_2)$ of $S \setminus E$ and creating L as the set of all chosen literals. For all the resulting pairs (E, L) such that L is non-contradictory, solve Problem I1 for input $(F^*, L, k - |E|)$ where $F^* = F \setminus S$. If for at least one pair (E, L) the output is a set S^* then return $E \cup S^*$. Otherwise return 'NO'. Assume that this algorithm returns $E \cup S^*$ such that S^* has been obtained for a pair (E, L). Let P be a satisfying assignment of $F^* \setminus S^*$ which does not intersect with $\neg L$. Observe that $P \cup L$ is non-contradictory, that $P \cup L$ satisfies all the clauses of $Clauses(F^* \setminus S^*) \cup (S \setminus E)$ and that $Clauses(F^* \setminus S^*) \cup (S \setminus E) = Clauses(F \setminus (S^* \cup E))$. Let L' be a set of literals, one for each variable of $Var(S \setminus E) \setminus Var(P \cup L)$. Then $P \cup L \cup L'$ is a satisfying assignment of $F \setminus (S^* \cup E)$, i.e. the output $(S^* \cup E)$ is valid. Assume that the output of Problem I1 is 'NO' for all inputs, but there is a set $Y \subseteq Clauses(F)$ such that $|Y| \leqslant k$ and $F \setminus Y$ is satisfiable. Let $E = Y \cap S$, $S^* = Y \setminus S$. Let P be a satisfying assignment of $F \setminus Y$ and let L be a set of literals obtained by selecting for each clause C of $S \setminus E$ a literal of C which belongs to P. Then the subset of P on the variables of $F^* \setminus S^*$ witnesses that SWRT$(F^* \setminus S^*, L)$ is true that is the output of Problem I1 on (E, L) cannot be 'NO'. This contradiction shows that when the proposed algorithm returns 'NO' this output is valid, i.e. the proposed algorithm correctly solves Problem I2.

In order to evaluate the complexity of the proposed algorithm, we bound the number of considered combinations (E, L). Each clause $C = (l_1 \vee l_2) \in S$ can be taken to E or l_1 can be taken to L or l_2 can be taken to L. That is, there are 3 possibilities for each clause, and hence there are at most 3^{k+1} possible combinations (E, L). Multiplying 3^{k+1} to the runtime of solving Problem I1 following from Claim 8, we obtain the desired runtime for Problem I2. □

Let (F, k) be an instance of 2-ASAT problem without repeated occurrences of clauses. Let $C_1, \ldots, C_m$ be the clauses of F. Let $F_0, \ldots, F_m$ be 2-CNF formulas such that F_0 is the empty formula and for each i from 1 to m, $Clauses(F_i) = \{C_1, \ldots, C_i\}$. We solve (F, k) by the method of iterative compression [17]. In particular we solve the 2-ASAT problems $(F_0, k), \ldots, (F_m, k)$ in the given order. For each (F_i, k), the output is either a CS S_i of F_i of size at most k or 'NO'. If 'NO' is returned for any (F_i, k), $i \leqslant m$, then clearly 'NO' can be returned for (F, k). Clearly, for (F_0, k), $S_0 = \emptyset$. It remains to be shown how to get S_i from S_{i-1}. Let $S'_i = S_i \cup \{C_i\}$. If $|S'_i| \leqslant k$ then $S_i = S'_i$. Otherwise, we solve Problem I2 with input (F_i, S'_i, k). If the output of this problem is a set then this set is S_i, otherwise the whole iterative compression procedure returns 'NO'. The correctness of this procedure can be easily shown by induction on i. It follows that the 2-ASAT problem with input $(F, k) = (F_m, k)$ can be solved by at most m applications of an algorithm solving Problem I2. According to Claim 9, Problem I2 can be solved in $\mathcal{O}(15^k \times k \times m^2)$, so 2-ASAT problem with input (F, k) can be solved in $\mathcal{O}(15^k \times k \times m^3)$.

Finally we show that if (F, k) contains repeated occurrences of clauses then the 2-ASAT problem remains FPT and can even be solved in the same runtime. In order to do that, we transform F into a formula F^* with all clauses being pairwise distinct and show that F can be made satisfiable by removing at most k clauses if and only if F^* can.

Assign each clause of F a unique index from 1 to m. Introduce new literals $l_1, \ldots, l_m$ of distinct variables that do not intersect with $Var(F)$. Replace the ith clause $(l' \vee l'')$ by two clauses $(l' \vee l_i)$ and $(\neg l_i \vee l'')$. Denote the resulting formula by F^*. It is easy to observe that all the clauses of F^* are distinct. Let I be the set of indices of the clauses of F such that the formula resulting from their removal is satisfiable and let P be a satisfying assignment of this resulting formula. Let $S^* = \{(l' \vee l_i) \mid i \in I\}$. Clearly, $|S^*| = |I|$. Observe that $F^* \setminus S^*$ is satisfiable. In particular, for every pair of clauses $(l' \vee l_i)$ and $(\neg l_i \vee l'')$ at least one clause is either satisfied by P or belongs to S^*. Hence $F^* \setminus S^*$ can be satisfied by an assignment that is obtained from P by adding for each i either l_i or $\neg l_i$ so that the remaining clauses are satisfied. Conversely, let S^* be a set of clauses of F^* of size at most k such that $F^* \setminus S^*$ is satisfiable and let P^* be a satisfying assignment of $F^* \setminus S^*$. Then for the set of indices I which consists of those i's such that the clause containing l_i or the clause containing $\neg l_i$ belong to S^*. Clearly $|S^*| \geqslant |I|$. Let F' be the formula obtained from F by removing the clauses whose indices belong to I. Observe that a clause $(l' \vee l'')$ belongs to $Clauses(F')$ if and only if both $(l' \vee l_i)$ and $(\neg l_i \vee l'')$ belong to $Clauses(F^* \setminus S^*)$. It follows that either l' or l'' belong to P^*. Consequently the subset of P^* consisting of the literals of variables of F' is a satisfying assignment of F'.

The argumentation in the previous paragraph shows that the 2-ASAT problem with input (F, k) can be solved by solving the 2-ASAT problem with input (F^*, k). If the output on (F^*, k) is a set S^* then S^* is transformed into a set of indices I as shown in the previous paragraph and the multiset of clauses corresponding to this set of indices is returned. If the output of the 2-ASAT problem on input (F^*, k) is 'NO' then the output on input (F, k) is 'NO' as well. To obtain the desired runtime, note that F^* has $2m$ clauses and $\mathcal{O}(m)$ variables and substitute this data to the runtime for 2-ASAT problem without repeated occurrences of literals. □

6. Concluding remarks

We conclude the paper by presenting a number of immediate by-products of the main result. It is noticed in [6] that the parameterized 2-ASAT problem is FPT-equivalent to the vertex cover problem parameterized above the prefect matching

(VC-PM). It is shown in [16] that the VC-PM problem is FPT-equivalent to the vertex cover problem parameterized above the size of a maximum matching and that the latter problem is FPT-equivalent to the problem of finding whether at most k vertices can be removed from the given graph so that the size of the minimum vertex cover of the resulting graph equals the size of its maximum matching. It follows from Theorem 7 that all these problems are FPT.

Finally, it is noted in [7] that the fixed-parameter tractability of the VC-PM problem implies the fixed parameter tractability of the following problem. Given a CNF formula F (not necessarily 2-CNF), is there a subset V of at most k variables of F so that after removing all their occurrences from the clauses of F, the resulting CNF formula is *Renamable Horn*, i.e. it can be transformed by renaming of the variables to a CNF formula with at most one positive literal in each clause. This subset V is known under the name RENAMABLE HORN DELETION BACKDOOR (RHORN-DB) and the problem of finding it can be referred to as the RHORN-DB problem.

Acknowledgments

This work was supported by Science Foundation Ireland through Grant 05/IN/I886. We thank Venkatesh Raman for pointing out to several relevant references, Somnath Sikdar for his help in fixing a bug in an earlier version of our manuscript, and Henning Fernau for drawing our attention to the Renamable Horn Deletion Backdoor problem.

References

[1] J. Bang-Jensen, G. Gutin, Digraphs: Theory, Algorithms and Applications, Springer-Verlag, 2001.
[2] J. Chen, F. Fomin, Y. Liu, S. Lu, Y. Villanger, Improved algorithms for the feedback vertex set problems, in: WADS, 2007, pp. 422–433.
[3] J. Chen, I. Kanj, Improved exact algorithms for MAX-SAT, Discrete Appl. Math. 142 (1–3) (2004) 17–27.
[4] J. Chen, Y. Liu, S. Lu, An improved parameterized algorithm for the minimum node multiway cut problem, in: WADS, 2007, pp. 495–506.
[5] J. Chen, Y. Liu, S. Lu, B. O'Sullivan, I. Razgon, A fixed-parameter algorithm for the directed feedback vertex set problem, in: STOC 2008, 2008, pp. 177–186.
[6] E. Demaine, G. Gutin, D. Marx, U. Stege, Open problems from Dagstuhl Seminar 07281, available electronically at http://drops.dagstuhl.de/opus/volltexte/2007/1254/pdf/07281.SWM.Paper.1254.pdf, 2007.
[7] G. Gottlob, S. Szeider, Fixed-parameter algorithms for artificial intelligence, constraint satisfaction, and database problems, Comput. J. 51 (3) (2008) 303–325.
[8] M. Grohe, M. Grüber, Parameterized approximability of the disjoint cycle problem, in: ICALP, 2007, pp. 363–374.
[9] J. Guo, F. Hüffner, E. Kenar, R. Niedermeier, J. Uhlmann, Complexity and exact algorithms for multicut, in: SOFSEM, 2006, pp. 303–312.
[10] F. Hüffner, Algorithm engineering for optimal graph bipartization, in: WEA, 2005, pp. 240–252.
[11] F. Hüffner, R. Niedermeier, S. Wernicke, Techniques for practical fixed-parameter algorithms, Comput. J. 51 (1) (2008) 7–25.
[12] S. Khot, V. Raman, Parameterized complexity of finding subgraphs with hereditary properties, Theoret. Comput. Sci. 289 (2) (2002) 997–1008.
[13] M. Mahajan, V. Raman, Parametrizing above guaranteed values: Maxsat and maxcut, Electron. Colloq. Computat. Complex. (ECCC) 4 (33) (1997).
[14] M. Mahajan, V. Raman, Parameterizing above guaranteed values: Maxsat and maxcut, J. Algorithms 31 (2) (1999) 335–354.
[15] D. Marx, Parameterized graph separation problems, Theoret. Comput. Sci. 351 (3) (2006) 394–406.
[16] S. Mishra, V. Raman, S. Saurabh, S. Sikdar, C. Subramanian, The complexity of finding subgraphs whose matching number equals the vertex cover number, in: ISAAC, 2007, pp. 268–279.
[17] R. Niedermeier, Invitation to Fixed-Parameter Algorithms, Oxford Lecture Ser. Math. Appl., vol. 31, 2006.
[18] C. Papadimitriou, Computational Complexity, Addison–Wesley, 1994.
[19] B. Reed, K. Smith, A. Vetta, Finding odd cycle transversals, Oper. Res. Lett. 32 (4) (2004) 299–301.
[20] A. Slivkins, Parameterized tractability of edge-disjoint paths on directed acyclic graphs, in: ESA, 2003, pp. 482–493.