arXiv:0801.1300v2 [cs.DS] 9 Jan 2008

Almost 2-SAT is Fixed-Parameter Tractable

Igor Razgon and Barry O'Sullivan
{i.razgon,b.osullivan}@cs.ucc.ie

Computer Science Department, University College Cork, Ireland

Abstract. 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', '2-SAT deletion'. The status of 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 which solves this problem in $O(15^k * k * m^3)$ and thus we show 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', '2-SAT deletion'. The status of fixed-parameter tractability of this problem is a long-standing open question in the area of Parameterized Complexity. To the best of our knowledge, the question regarding the fixed-parameter tractability of this problem has been first asked in 2000 by Khot and Raman [11]. This question has been posted in the book of Niedermeier [13] being referred as one of central challenges for parameterized algorithms design. Finally, in July 2007, this question has been posted 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 the considered problem in $O(15^k * k * m^3)$ time. Thus we show that this problem is fixed-parameter tractable.

The rest of the introduction is partitioned into three subsections. In the first subsection we overview the proposed algorithm. In the second subsection we discuss the related work. The third subsection outlines the structure of the rest of the paper.

1.1 Overview of the algorithm

We start from the terminology we adopt regarding the names of the considered problems. We call *Almost 2-SAT* (abbreviated as *2-ASAT*) 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 2-CNF formula is satisfiable. The *parameterized 2-ASAT* problem gets as additional input a parameter k and the output of this problem is a set of at most k clauses whose removal makes the given 2-CNF formula satisfiable, in case such a set exists. If there is no such a set, the output is 'NO'. So, the algorithm proposed in this paper solves the parameterized 2-ASAT problem.

We introduce two variations of the 2-ASAT problem. The first of them is the *2-ASAT problem with a single literal* problem abbreviated as *2-SLASAT*. The input of this problem is a pair (F, l), where F is a *satisfiable* 2-CNF formula and l is a literal. The task is to find a smallest subset of clauses to be removed from F so that the resulting formula is *satisfiable w.r.t.* l (i.e. has a satisfying assignment which does not include $\neg l$). The second variations of 2-ASAT is called the *annotated 2-SLASAT problem* abbreviated as *2-ASLASAT*. The input of this problem is (F, L, l), where F is a 2-CNF formula, L is a set of literals such that F is *satisfiable w.r.t.* L (i.e. has a satisfying assignment which does not include negations of literals of L), 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 w.r.t. $(L \cup \{l\})$. Clearly, the 2-SLASAT problem is a special case of the 2-ASLASAT problem obtained by setting $L = \emptyset$. The parameterized versions of 2-SLASAT and 2-ASLASAT problems are defined analogously to the parameterized 2-ASAT problem.

The description of the algorithm for the parameterized 2-ASAT problem is divided into two parts. In the first part (which is the most important one) we provide an algorithm which solves the parameterized 2-SLASAT problem in $O(5^k * k * m^2)$ time. In the second part we show that the parameterized 2-ASAT problem can be solved by $O(3^k * m)$ applications of the algorithm solving the parameterized 2-SLASAT problem. The resulting runtime follows from the product of the last two complexity expressions. The transformation of the 2-ASAT problem into the 2-SLASAT problem is based on the *iterative compression* and can be seen as an adaptation of the method employed in [9] in order to solve the graph bipartization problem. In the rest of the subsection we overview the first part.

The algorithm for the parameterized 2-SLASAT problem in fact solves the parameterized 2-ASLASAT problem. In order to design the algorithm, we represent the 2-ASLASAT problem as a *separation* problem and prove a number of theorems based on this view. In particular, we introduce a notion of a *walk* from a literal l' to a literal l'' in a 2-CNF formula F. We define the walk as a sequence $(l' \vee l_1), (\neg l_1 \vee l_2), \dots, (\neg l_{k-1} \vee l_k), (\neg l_k \vee l'')$ of clauses of F such that literals are ordered within each clause so that the *second* literal of each clause except the last one is the negation of the *first* literal of the next clause. Then we prove that, given an instance (F, L, l) of the 2-ASLASAT problem, F is insatisfiable w.r.t. $L \cup \{l\}$ 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.

Next we define the notion of a *path* of F as a walk of F with no repeated clauses. Based on this notion we prove a Menger's like theorem. In particular, given an instance (F, L, l) of the 2-ASLASAT problem, we show that the smallest number of clauses whose removal breaks all the paths from $\neg L$ to $\neg l$ equals the largest number of clause-disjoint paths from $\neg L$ to $\neg l$ (for this result it is essential that F is satisfiable w.r.t. L). Based on this result, we show that the size of the above *smallest separator* of $\neg L$ from from $\neg l$ can be computed in a polynomial time by a Ford-Fulkerson-like procedure. Thus this size is a polynomially computable *lower bound* on the size of the solution of (F, L, l).

Next we introduce the notion of a *neutral literal* l^* of (F, L, l) whose main property is that the number of clauses which separate $\neg(L \cup \{l^*\})$ from $\neg l$ equals the number of clauses separating $\neg L$ from $\neg l$. Then we prove a theorem stating that in this case the size of a solution of $(F, L \cup \{l^*\}, l)$ does not exceed the size of a solution of (F, L, l). The strategy of the proof is similar to the strategy of the proof of the main theorem of [2].

Having proved all the above theorems, we present the algorithm solving the parameterized 2-ASLASAT problem on input (F, L, l, k). The algorithm selects a clause C. If C includes a neutral literal l^* then the algorithm applies itself recursively to $(F, L \cup \{l^*\}, l, k)$ (this operation is justified by the theorem in the previous paragraph). 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 literals of C to L and applies itself recursively without changing the size of the parameter. 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.

1.2 Related Work

As said above, the parameterized 2-ASAT problem has been introduced in [11]. The authors showed that this problem is a generalization of the parameterized graph bipartization problem, which was also an open problem at that time. The latter problem has been resolved in [15]. Apart from resolving an open problem, the additional contribution of [15] was introducing a method of iterative compression which has had a considerable impact on the design of parameterized algorithms. The most recent algorithms based on this method are currently the best algorithm for the undirected Feedback Vertex Set [3] and the first parameterized algorithm for the famous Direct FVS problem [4]. For earlier results based on the iterative compression, we refer the reader to a survey article [10].

The study of parameterized graph separation problems has been initiated in [12]. 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 to be separated was also a parameter. The latter result has been extended in [8] where fixed-parameter algorithms for multicut problems on several classes of graphs have been proposed. The first $O(c^k * poly(n))$ algorithm for the multiterminal cut problem has been proposed in [2]. A reformulation of the main theorem of [2] is an essential part of the parameterized algorithm for the Directed FVS problem [4] mentioned in the previous paragraph. Along with computing the separators, the methods of computing disjoint paths have been investigated. The research led to intractability results [16] and parameterized approximability results [7].

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 received a considerable attention from the researchers resulted in a series of improvements of the worst-case upper bound on the runtime of this problem. Currently the best algorithm is given in [5] and solves this problem in $O(1.37^k + |F|)$, where $|F|$ is the size of the given formula.

1.3 Structure of the Paper

In Section 2 we introduce the terminology which we use in the rest of the paper. In Section 3 we prove the theorems mentioned in the above overview subsection. In Section 4 we present an algorithm for the parameterized 2-ASLASAT problem, prove its correctness and evaluate the runtime. In Section 5 we present the iterative compression based transformation from parameterized 2-ASAT problem to the parameterized 2-SLASAT problem. For the sake of simplicity we assume throughout the paper that all clauses of the given formula are distinct. Therefore at the very end of Section 5 we provide a theorem which shows how the proposed result can be generalized (without changing the runtime) to 2-CNF formulas with repeated occurrences of clauses.

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 considered 2-CNF formulas. First, we assume that all the clauses of the considered formulas are of size 2. If a formula has a clause (l) of size 1 then this clause is represented as $(l \vee l)$. Second, everywhere except the very last theorem, we assume that all the clauses of any considered formula are pairwise distinct. [1] This assumption allows us to represent the operation of removal clauses from a formula in a set-theoretical manner. In particular, let S be a set of clauses [2]. Then $F \setminus S$ is a 2-CNF formula which is the AND of clauses of F that are not contained in S. The result of removal a single clause C is denoted by $F \setminus C$ rather than $F \setminus \{C\}$.

[1] Note that the clause $(l_1 \vee l_2)$ is *the same* as the clause $(l_2 \vee l_1)$.

[2] We implicitly assume that all the clauses considered in this paper have size 2

Let F, S, C, L be a 2-CNF formula, a set of clauses, a single clause, and a set of literals. Then $Var(F)$, $Var(S)$, $Var(C)$, $Var(L)$ 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-contradictional* 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-contradictional 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 has a satisfying assignment P which does not intersect with $\neg L$ [3]. 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 the space, we introduce a special notation for this notion. In particular, we say that $SWRT(F, L)$ is true (false) if F is, respectively, satisfiable (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. *A walk of the given 2-CNF formula F is a non-empty sequence $w = (C_1, \dots, C_q)$ 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}.*

Let $w = (C_1, \dots, C_q)$ 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.r.t.) 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 $(C_q, \dots, C_1)$ in which the first and the second literals of each entry are exchanged w.r.t. 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

[3] We do not say 'P contains L' because generally $Var(L)$ may be not a subset of $Var(F)$

walk whose first literal is $\neg l''$. Then $w+w'$ is the walk obtained by concatenation of w' to the end of w with the first and second literals of all entries in w and w' preserving their roles in $w+w'$.

Definition 2. *A path of a 2-CNF formula F is a walk of F all clauses of which are pairwise distinct.*

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 $(\neg l_2 \vee l_3)$ w.r.t. 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 $(\neg l_2 \vee l_3)$ w.r.t. 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 $(l_1 \vee l_2)$ w.r.t. 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 Almost 2-SAT problem and its variations.

Definition 3. *1. A Culprit Set (CS) of a 2-CNF formula F is a subset S of $Clauses(F)$ such that $F \setminus S$ is satisfiable.*
2. Let (F, l) be a pair where F is a satisfiable 2-CNF formula and l is a literal. A CS of (F, l) is a subset of $Clauses(F)$ such that $SWRT(F \setminus S, l)$ is true.
3. Let (F, L, l) where F is a 2-CNF formula, L is a non-contradictionary set of literals such that $SWRT(F, L)$ is true and l s 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.

Having defined a CS with respect to three different structures, we define problems of finding a smallest CS (SCS) with respect to these structures. In particular *Almost 2-SAT problem* (2-ASAT problem) is defined as follows: given a 2-CNF formula F, find an SCS of F. The *Almost 2-SAT problem with a single literal* (2-SLASAT problem) is defined as follows: given a pair (F, l) as in the second item of Definition 3, find an SCS of (F, l). Finally, the *Annotated Almost 2-SAT problem with single literal* (2-ASLASAT problem) is defined as follows: given the triplet (F, L, l) as in the last item of Definition 3, find an SCS of (F, L, l). Note that the 2-ASLASAT problem is a generalization of the 2-SLASAT problem. In particular, an instance (F, l) of the 2-SLASAT problem is equivalent to the instance $(F, \emptyset, l)$ of the 2-ASLASAT problem.

Now we introduce parameterized versions of the 2-ASAT, 2-SLASAT, and 2-ASLASAT problem, where the parameter restricts the size of a CS. 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 3. The output is a CS of (F, L, l) of size at most k, if there is such one. Otherwise, the output is 'NO'. Finally, the instance (F, l, k) of the parameterized 2-SLASAT problem is equivalent to the instance $(F, \emptyset, l, k)$ of the parameterized 2-ASLASAT problem.

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 $SWRT(F, \{\neg l_x, \neg l_y\})$ is false. In particular, if $l_x = l_y$ then $SWRT(F, \neg l_x)$ 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 $SWRT(F, \{\neg l_x, \neg l_y\})$ 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 is clear 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 $SWRT(F, \{\neg l_x, l_t\})$ is false and hence any satisfying assignment containing $\neg l_x$ contains $\neg l_t$ and hence contains l_y. As we noted above in the proof, this implies that $SWRT(F, \{\neg l_x, \neg l_y\})$ is false. ■

Lemma 2. *Let F be a 2-CNF formula and let L be a set of literals such that $SWRT(F, L)$ 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 w.r.t. w. Then l_1 is* not *a second literal of C w.r.t. 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 w.r.t. 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 $SWRT(F, \{\neg l_1, \neg l'\})$ is false. Therefore if $l_1 \in \neg L$ then $SWRT(F, L)$ 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', both belong to $\neg L$. According to Lemma 1, $SWRT(F, \{\neg l^*, \neg l'\})$ is false and hence $SWRT(F, L)$ 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 $SWRT(F, L)$ 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 statement is clear 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 $w = (C_1, \ldots, C_q)$ and assume that $C_i = C_j$ where $1 \leq i < j \leq q$. 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 $w' = (C_1, \ldots, C_i, C_{j+1}, C_q)$. 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 $SWRT(F, L \cup \{l\})$ 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, $SWRT(F, l)$ is false or $SWRT(F, \{l', l\})$ is false, respectively. Clearly in both cases $SWRT(F, L \cup \{l\})$ is false as $L \cup \{l\}$ is, by definition, a superset of both $\{l\}$ and $\{l', l\}$.

Assume now that $SWRT(F, L \cup \{l\})$ 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. Observe that $Var(I) \subseteq Var(S)$. Really, at least one clause C of F includes $\neg l$ because otherwise that fact that $SWRT(F, L)$ is true (which is a part of definition of 2-ASLASAT problem) causes that $SWRT(F, L \cup \{l\})$ is true in contradiction to our assumption. Any clause C including $\neg l$ belongs to S since the other literal of C belongs to I by definition. It follows that $Var(l) \in Var(S)$. For any other $l' \in I$, F has a walk from $\neg l$ to l', the last clause of this walk contains l' and hence belongs to S. Consequently $Var(l') \in Var(S)$.

Assume that I is not-contradictionary 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-contradictionary. Really, P' is non-contradictionary as being a subset of a satisfying assignment P of F, I is non-contradictionary 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 $P' \cup I$. Really, if $C \in S$ then C is satisfied by I, by definition of I. If $C \in Clauses(F \setminus S)$ then, $Var(C) \subseteq Var(F \setminus S) = Var(F) \setminus Var(S) \subseteq Var(F) \setminus Var(I) = Var(P')$. 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 its definition. 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 $\neg(L \cup l)$. Really, 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-contradictionary, hence $\neg l \notin P' \cup I$. Thus $P' \cup I$ witnesses that $SWRT(F, L \cup \{l\})$ is true in contradiction to our assumption. Thus our assumption regarding I made in the beginning of the present paragraph is incorrect.

It follows from the previous paragraph 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 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 tot he 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 $SWRT(F, L \cup \{l\})$ 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 4. *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 have a path from L to l_y.*

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

Given the above definition, 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 w.r.t. $\{l', l\}$ 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), 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. The *arc separator* of D w.r.t. 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. Similarly to the case with 2-CNF formulas, we denote by $ArcSepSize(D, L, l)$ the size of the smallest arc separator of D w.r.t. L and l.

Theorem 2. *Let F be a 2-CNF formula, let L be a set of literals such that $SWRT(F, \neg L)$ 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 the largest number of arc-disjoint paths from $\neg L$ to l_y in D.*
2. *$SepSize(F, L, l_y) = ArcSepSize(D, \neg L, l_y)$.*
3. *The largest number of clause-disjoint paths from L to l_y in F equals $SepSize(F, L, l_y)$.*

Note that generally (if there is no requirement that $SWRT(F, \neg L)$ 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 $SWRT(F, \neg L)$ is true rules out this possibility.

Proof of Theorem 2. We may 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.

Let $\mathbf{P} = \{p_1, \dots, p_t\}$ be a set of clause-disjoint paths from L to l_y. A simple inductive argument shows that a path p from l_1 to l_2 in F corresponds to a walk from $\neg l_1$ to l_2 in D whose set of arcs represent the set of clauses of p. Assume first that $p = (l_1 \vee l_2)$, i.e. p consists of a single clause. Then D has an arc $(\neg l_1, l_2)$, so the statement holds for p of length 1. Assume that p has length $i > 1$ and the statement holds for all shorter paths. Then p can be represented in the form $p' + (\neg l'_2 \vee l_2)$, where l'_2 is the last literal of p'. By the induction assumption, there is a walk q from $\neg l_1$ to l'_2 in D and also $(l'_2, l_2) \in A(D)$ by definition of D. The walk $g + (l'_2, l_2)$ is the desired walk from $\neg l_1$ to l_2 in D, where '+' means that the arc (l'_2, l_2) is appended to the end of g. It follows that there is a set $\mathbf{W} = \{w_1, \dots, w_t\}$ of walks from $\neg L$ to l_y in D such that the set of arcs of each w_i represent the set of clauses of p_i. Observe that the walks of $\mathbf{W}$ are clause-disjoint because if an arc e belongs to w_i and w_j then e represents two different clauses, one for p_i and one for p_j, a contradiction. Since every walk w' in D includes a directed path p' with the same first and last nodes as w' and whose set of arcs is a subset of the set of arcs of w', we can specify a set of paths $\mathbf{P'} = \{p'_1, \dots, p'_t\}$ of D from $\neg L$ to l_y such that the set of arcs of each p'_i is a subset of the set of arcs of w_i. In the light of the previous discussion it is clear that the paths of $\mathbf{P'}$ are arc-disjoint. Hence largest number of arc-disjoint directed paths from $\neg L$ to l_y in D is at least as large as the largest number of clause-disjoint paths from L to l_y in F.

Let p' be a path of length t (for some arbitrary t) from a node $\neg l_1$ to a node l_2 in D. Applying induction similarly to the previous paragraph, we observe that there is a *corresponding* walk w in F from l_1 to l_2 of the same length such that for each i from 1 to t the i-th clause of w is represented by i-th arc of p'.

Moreover, let the i-th arc be (l'_1, l'_2). Then $\neg l'_1$ and l'_2 are, respectively, the first and the second literals of the i-th clause of w (in other words, $\neg l'_1$ and l_2 are, respectively, the first and the second literals of $(\neg l'_1 \vee l'_2)$ w.r.t. w). If p' consists of the single arc $(\neg l_1, l_2)$, this arc represents clause $(l_1 \vee l_2)$, so the statement clearly holds. Assume that p' has length $i > 1$ and that the statement holds for all shorter paths. Let (l'_2, l_2) be the last arc of p'. The prefix p'' of p' preceding this arc is a path in D from $\neg l_1$ to l'_2. By the induction assumption, F has a walk w'' from l_1 to l'_2 corresponding to p'', which satisfies the required properties. Then $w'' + (\neg l'_2 \vee l_2)$ is the walk corresponding to p' and satisfying the required properties.

Let us say that an arc (l_1, l_2) of D is reachable from the set of nodes $\neg L$ (where $\neg L$ is as in the statement of the lemma) if $l_1 \in \neg L$ or there is a path from $\neg L$ to l_1 in D.

Claim 1 *If two distinct arcs are reachable from $\neg L$ then these arcs represent different clauses of F*

Proof. Assume by contradiction that F has a clause $C = (l_1 \vee l_2)$ such that both arcs $(\neg l_1, l_2)$ and $(\neg l_2, l_1)$ are reachable from $\neg L$. Then D has paths p_1 and p_2 both starting from nodes of $\neg L$ and including $(\neg l_1, l_2)$ and $(\neg l_2, l_1)$, respectively. According to the second paragraph of the proof of the present theorem, p_1 and p_2 correspond to walks w_1 and w_2 from L in F such that l_1 is a first literal of C w.r.t. w_1 and a second literal of C w.r.t. w_2. However, this contradicts Lemma 2, and hence proves validity of the present claim. □

Let $\mathbf{P'} = \{p'_1, \ldots, p'_t\}$ be a set of arc-disjoint paths of D from $\neg L$ to l_y. Let $\mathbf{W'} = \{w'_1, \ldots, w'_t\}$ be the set of walks from L to l_y in F where w'_i corresponds to p'_i as shown in the second paragraph of the present proof. Observe that each w'_i is a path in F because otherwise two different arcs of p_i, which are reachable from $\neg L$, correspond to the same clause of F in contradiction to the statement of Claim 1. Furthermore, any two w'_i and w'_j are clause-disjoint because otherwise an arc of p'_i and an arc of p'_j which are distinct by definition and both reachable from $\neg L$ correspond to the same clause again contradicting Claim 1. Thus the largest number of clause-disjoint paths from L to l_y in F is at least as large as the largest number of arc-disjoint paths from $\neg L$ to l_y in D. Combining this statement with the statement proven in the first paragraph of the present proof we see that these two numbers are equal.

Let $S \in \mathbf{OptSep}(F, L, l_y)$. For each clause $C \in S$, fix a path $p(C)$ from L to l_y in F which includes C (such a path necessarily exists due to the minimality of S). As shown in the first paragraph of the present proof, there is a walk $w(C)$ from $\neg L$ to l_y in D whose set of arcs represent clauses of $p(C)$. Let $e(C)$ be the arc of $w(C)$ which represents C. Let $D(S)$ be the set of $e(C)$ for each $C \in S$. Clearly $|D(S)| = |S|$ because each arc represents exactly one clause. We show that $D(S)$ separates $\neg L$ from l_y in D. Assume that this is not true and let p be a path from $\neg L$ to l_y in D which does not include any arc of $D(S)$. As shown above F has a path p' from L to l_y whose clauses are represented by the arcs of p. By definition of S, p' necessarily includes a clause C of S. Let

e' be an arc of p which represents C. By definition of p, $e' \neq e(C)$. In other words, e' and $e(C)$ are two arcs reachable from $\neg L$ which represent the same clause in contradiction to Claim 1. Thus $D(S)$ separates $\neg L$ from l_y in D and the $ArcSepSize(D, \neg L, l_y) \leq SepSize(F, L, l_y)$.

Let S be a minimal arc separator of $\neg L$ from l_y in D. For each $e \in S$, let $C(e)$ be the clause represented by e. Let $F(S)$ be the set of all clauses $C(e)$. Clearly $|F(S)| \leq |S|$. We show that $F(S)$ is a separator w.r.t. L and l_y in F. Assume that this is not true. Then F has a path p from L to l_y which does not include any clause of $F(S)$. Then, as shown in the first paragraph of the present proof, there is a path p' from $\neg L$ to l_y in D such that the arcs of the path represent a subset of clauses of p. Due to the definition of S, p' necessarily includes an arc e' of S. Let C' be the clause of p represented by e'. By definition of p, $C' \neq C(e')$, that is e' represent two different clauses in contradiction to the definition of an implication graph. Thus $F(S)$ is a separator w.r.t. L and l_y in F and $ArcSepSize(D, \neg L, l_y) \geq SepSize(F, L, l_y)$. Combining this statement with the one proven in the previous paragraph, we obtain that the sizes of these two separators are the same.

Let **PF** be a largest set of clause-disjoint paths from L to l_y in F and let **PD** be a largest set of clause-disjoint paths from $\neg L$ to l_y in D. It follows from the above proof that in order to show that $|\mathbf{PF}| = SepSize(F, L, l_y)$, it is sufficient to show that $|\mathbf{PD}| = ArcSepSize(D, \neg L, l_y)$. Taking into account that by our assumption $l_y \notin \neg L$, the latter can be easily derived by contracting the vertices of $\neg L$ into one vertex and applying the arc version of Menger's Theorem for directed graphs [1]. ■

3.4 Neutral Literals

Definition 5. *Let (F, L, l) be an instance of the 2-ASLASAT problem. Let l^* be a literal satisfying the following properties:*

1. *$Var(l^*) \notin Var(L)$ and $Var(l^*) \neq Var(l)$;*
2. *$SWRT(F, L \cup \{l^*\})$ is true;*
3. *$SepSize(F, L, l) = SepSize(F, L \cup \{l^*\}, l)$.*

Then l^* is called a neutral literal of (F, L, 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-ASALSAT 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 6. *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 w.r.t. w. Then l_1 is called* the main literal *of C w.r.t. (F, L, l).*

Given Definition 6, 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 w.r.t. (F, L, l). Then l_1 is not a second literal of C w.r.t. 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 w.r.t. $\neg L$ 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, 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, XNR respectively. Observe that the sets XR, XSP, 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) 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 w. By definition, $SP \setminus XP = SP \setminus X$ and $NR \setminus XNR = NR \setminus X$, hence the clauses of w do not intersect with X.

Claim 2 $|Y| \leq |XR|$.

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 one-to-one correspondence between paths of $\mathbf{P}$ and the clauses of SP they include. Let $\mathbf{PY}$ be the subset of $\mathbf{P}$ containing 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. 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 to intersect with X is intersection 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 w.r.t p, hence the suffix of p starting at C can be replaced by the witness walk of C 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 $\mathbf{PR}$ are clause-disjoint, $|XR| \geq |\mathbf{PY}| = |Y|$, as required. $\square$

Consider the set $X^* = Y \cup XSP \cup XNR$. Observe that $|X^*| = |Y| + |XSP| + |XNR| \leq |XR| + |XSP| + |XNR| = |X|$, the first equality follows from the mutual disjointness of Y, XSP and XNR by their definition, the inequality follows from Claim 2, 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 3 *$F \setminus X^*$ 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 w.r.t. $\neg(L \cap \{l^*\})$ 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 w.l.o.g. that l_1 is the main literal of C in $(F \setminus X^*, L \cup \{l^*\}, l)$. Let p^* be the suffix of p starting at C.

According to Claim 2, l_1 is the first literal of p^*. In the next paragraph we will show that no clause of R follows C is 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 clauses belong to X^*) we obtain the 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 w.l.o.g. that l'_1 is the main literal of C' in $(F \setminus X^*, L \cup \{l^*\}, l)$). 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 Lemma 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. $\square$

Claim 4 *$F \setminus X^*$ 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'. If l'_1 is a first literal of C' w.r.t. 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 3, 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 3. This contradiction shows that our initial assumption regarding the existence of w is incorrect and hence completes the proof of the present claim. □

It follows from Combination of Theorem 1, Claim 3, and Claim 4 that X^* is a CS of (F, L, l), which completes the proof of the present theorem. ■

4 Algorithm for the parameterized 2-ASLASAT problem and its analysis

4.1 The algorithm

$Solve2ASLASAT(F, L, l, k)$
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** $SWRT(F, L \cup \{l\})$ **then** return $\emptyset$
2. **if** $k = 0$ **then** Return 'NO'
3. **if** $k \geq |Clauses(F)|$ **then** return $Clauses(F)$
4. **if** $SepSize(F, \neg L, \neg l) > k$ **then** return 'NO'
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 $Var(l_2) \notin Var(L)$ (l_1 is selected w.l.o.g.)
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 [4]
7. **if** Both l_1 and l_2 belong to $\neg(L \cup \{l\})$ **then**
   7.1 $S \leftarrow Solve2ASLASAT(F \setminus C, L, l, k-1)$
   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 $S_1 \leftarrow Solve2ASLASAT(F, L \cup \{l_1\}, l, k)$
   8.2 **if** S_1 is not 'NO' **then** Return S_1
   8.3 $S_2 \leftarrow Solve2ASLASAT(F, L \cup \{l_2\}, l, k)$
   8.4 **if** S_2 is not 'NO' **then** Return S_2
   8.5 $S_3 \leftarrow Solve2ASLASAT(F \setminus C, L, l, k-1)$
   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 $\neg(L \cup \{l\})$. W.l.o.g. we assume that this literal is l_1)
9. **if** l_2 is not neutral in (F, L, l)
   9.1 $S_2 \leftarrow Solve2ASLASAT(F, L \cup \{l_2\}, l, k)$
   9.2 **if** S_2 is not 'NO' **then** Return S_2

[4] Doing the analysis, we will prove that on Steps 5 and 6 F has at least one clause with the required property

9.3 $S_3 \leftarrow Solve2ASLASAT(F \setminus C, L, l, k-1)$
9.4 **if** S_3 is not 'NO' **then** Return $S_3 \cup \{C\}$
9.5 Return 'NO'
10. Return $Solve2ASLASAT(F, L \cup \{l_2\}, l, k)$

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 $Solve2ASLASAT(F, L, l, k)$. The root of the tree is identified with (F, L, l, k). If $Solve2ASLASAT(F, L, l, k)$ 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 $Solve2ASLASAT(F, L, l, k)$. (For example, if $Solve2ASLASAT(F, L, l, k)$ 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 $Solve2ASLASAT$ that the third item of a valid input is not changed for its children hence in the rest of the 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. The $Solve2ASLASAT(F, L, l, k)$ succeeds to select a clause on Steps 5 and 6.*

Proof. Assume that F has a walk from $\neg L$ to $\neg l$ and let w be the shortest possible 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.

Consider now the case where the condition of Step 5 is not satisfied. Note that $SWRT(F, L \cup \{l\})$ 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 $SWRT(F, \{l_1, l_2\})$ 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, $SWRT(F', l)$ 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. ■

Lemma 5. *Let (F, L, l, k) be a valid input and assume that $Solve2ASLASAT(F, L, l, k)$ applies itself recursively. Then all the children of (F, L, l, k) in the search tree are valid inputs.*

Proof. Let (F_1, L_1, l, k_1) be a child of (F, L, l, k) . Observe that $k_1 \geq k - 1$. Observe also that $k > 0$ because $Solve2ASLASAT(F, L, l, k)$ would not apply itself recursively if $k = 0$. It follows that $k_1 \geq 0$.

It remains to prove 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 on 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 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 on Steps 5 and 6. In particular, we are going to show that

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

That $L \cup \{l^*\}$ is non-contradictionary follows from 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 on Step 5, that is $l_1 \in \neg L$ and $l_2 = l$. Let P be a satisfying assignment of F which does not intersect with $\neg L$ (existing since ($SWRT(F, L)$ is true). Then $l_2 \in P$, i.e. $SWRT(F, L \cup \{l\})$ is true, which is impossible since in this the algorithm would stop at Step 1. The assumption that C is selected on Step 6 also leads to a contradiction because on the one hand $SWRT(F, l)$ is false by Lemma 1 due to existence of a walk from $\neg l$ to $\neg l$, on the other hand $SWRT(F, l)$ 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 w.l.o.g. that $l_1 \in \neg L$. Then, by the first statement, $l^* = l_2$. 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. $SWRT(F, L \cup \{l_2\})$ is true in the considered case. Assume that C is selected on 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 w.r.t. 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 $SWRT(F, L \cup \{l^*\})$ is false. Since $L \cup \{l^*\}$ is non-contradictionary 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 $SWRT(F, l^*)$ is true by selection of C. Let w'' be a walk from $\neg L$ to $\neg l^*$. Then

$w'' + w'$ is a walk from $\neg L$ to $\neg l$ in contradiction to our assumption that C is selected on Step 6. Thus $SWRT(F, L \cup \{l^*\})$ is true. The proof of the present lemma is now complete. ■

Now we introduce two measures of the input of the $Solve2ASLASAT$ 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 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 on Step 6. Really, if C is selected on 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-contradictional), 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, l, k), $l^* \neq \neg l$. 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 on 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 $\beta(F, L, l, k) \geq \beta(F_1, L_1, l, k_1)$. 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 $Solve2ASLASAT(F, L, l, k)$ 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 $\beta(F, L, l, k) \geq \beta(F_1, L_1, l, k_1)$ it is sufficient to show that $2k - SepSize(F, \neg L, \neg l) > 2k_1 - SepSize(F_1, \neg L_1, \neg l)$ or $2k - SepSize(F, \neg L, \neg l) \geq 2k_1 - SepSize(F_1, \neg L_1, \neg l)$, respectively.

Assume first that $(F_1, L_1, l, k_1) = (F \setminus C, L, l, k-1)$. Observe that $SepSize(F \setminus C, \neg L, \neg l) \geq SepSize(F, \neg L, \neg l) - 1$. Really assume the opposite and let S be a separator w.r.t. 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 w.r.t. $\neg L$ 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 $2(k-1) - SepSize(F \setminus C, \neg L, \neg l) = 2k - SepSize(F \setminus C, \neg L, \neg l) - 2 \leq 2k - SepSize(F, \neg L, \neg l) - 1 < 2k - SepSize(F, \neg L, \neg l)$.

Assume now that $(F_1, L_1, l, k_1) = (F, L \cup \{l^*\}, l, k)$ for some literal l^*. Clearly, $SepSize(F, \neg L, \neg l) \leq SepSize(F, \neg(L \cup \{l^*\}), \neg l)$ due to being $\neg L$ a subset of $\neg(L \cup \{l^*\})$. It follows that $2k - SepSize(F, \neg L, \neg l) \geq 2k - SepSize(F, \neg(L \cup$

$\{l^*\}), \neg l)$. It remains to show that $\geq$ can be replaced by $>$ in 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 $Solve2ASLASAT(F, L \cup \{l^*\}, l, k)$ is applied on Steps 8.2, 8.4, or 9.3. 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 on Step 6. That is, F has no walk from $\neg L$ to $\neg l$. According to Lemma 3, 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 again 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 $\alpha(F, L, l, k)$.* [5]
- *Each 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 $ST(F, L, l, k)$, $\beta(F', L', l, k') \leq \beta(F, L, l, k) - t$ 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 the size of $\alpha(F, L, l, k)$ 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, the height of $ST(F^*, L^*, l, k^*)$ of any child (F^*, L^*, l, k^*) of (F, L, l, k) is at most $\alpha(F^*, L^*, l, k^*)$. Hence the first statement follows by Lemma 6. Furthermore, any 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 $\alpha(F', L', l, k') \leq \alpha(F^*, L^*, l, k^*) < \alpha(F, L, l, k)$, the last inequality follows from Lemma 6. Furthermore $\beta(F', L', l, k') \leq \beta(F^*, L^*, l, k^*) - t^*$ 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 $ST(F, L, l, k)$ [6]. 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 $\beta(F', L', l, k') \leq \beta(F, L, l, k) - t^* - 1 = \beta(F, L, l, k) - t$. ■

[5] 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.

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

4.3 Correctness Proof

Theorem 4. *Let* (F, L, l, k) *be a valid input. Then* $Solve2ASLASAT(F, L, l, k)$ *correctly solves the parameterized 2-ASLASAT problem. That is, if* $Solve2ASLASAT(F, L, l, k)$ *returns a set, this set is a CS of* (F, L, l) *of size at most* k*. If* $Solve2ASLASAT(F, L, l, k)$ *returns 'NO' then* (F, L, l) *has no CS of size at most* k*.*

Proof. Let us prove first the correctness of $Solve2ASLASAT(F, L, l, k)$ for the cases when the procedure does not apply itself recursively. It is only possible when the procedure returns an answer on Steps 1-4. If the answer is returned on Step 1 then the validity is clear because nothing has to be removed from F to make it satisfiable w.r.t. L and l. If the answer is returned on 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 on 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 $Clauses(F)$ 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 on 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 correctness of $Solve2ASLASAT(F, L, l, k)$ 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, $Solve2ASLASAT(F, L, l, k)$ does not apply itself recursively (the output is returned on Step 1 or Step 2). Therefore, the correctness of $Solve2ASLASAT(F, L, l, k)$ 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 $Solve2ASLASAT(F, L, l, k)$ applies itself recursively, i.e. the node (F, L, l, k) has children in $ST(F, L, l, k)$.

Claim 5 *Let* (F_1, l_1, l, k_1) *be a child of* (F, L, l, k)*. Then* $Solve2ASLASAT(F_1, L_1, l, k_1)$ *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 $Solve2ASLASAT(F, L, l, k)$ returns a set S. By description of the algorithm, S is returned by $Solve2ASLASAT(F, L \cup \{l^*\}, l, k)$ 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 $Solve2ASLASAT(F \setminus C, L, l, k-1)$ 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 5 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 5 that $|S_1| \leq k - 1$ 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 $Solve2ASLASAT(F, L, l, k)$ 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 on 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 5, the recursive call of Step 7.2. would not return 'NO'. However, this means that any satisfying assignment of $F \setminus S$ which does not intersect with $\neg(L \cup \{l\})$ (which exists by definition) cannot satisfy clause C, a contradiction. Assume now that 'NO' is returned on Step 10. By Claim 5, $(F, L \cup \{l_2\}, l)$ has no CS of size at most k. 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 on Step 8.7. or on Step 9.5. Assume first that the clause C selected on 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 $Solve2ASLASAT(F, L \cup \{l^*\}, l, k)$ 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 $Solve2ASLASAT(F, L \cup \{l^*\}, l, k)$ returned an incorrect answer in contradiction to Claim 5. 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 $Solve2ASLASAT(F \setminus C, L, l)$ contradicts Claim 5. Thus the answer 'NO' returned by $Solve2ASLASAT(F, L, l, k)$ 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 $\beta(F, L, l, k) \geq 0$ by definition, $\sqrt{5}^t \geq 1$. Hence if $Solve2ASLASAT(F, L, l, k)$ 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, $Solve2ASLASAT(F, L, l, k)$ does not apply itself recursively and hence the theorem holds by the 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 6 *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 6, is at most $\sqrt{5}^{t_1}$, where $t_1 = \beta(F_1, L_1, l, k_1)$. According to Lemma 7, $t_1 \leq t$ 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 6, 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 $(2/\sqrt{5}) * (\sqrt{5}^t) < \sqrt{5}^t$, 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 w.l.o.g. 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 on steps 5 and 6. Let $t_i = \beta(F_i, L_i, l, k_i)$ for $i = 1, 2, 3$.

Claim 7 $t \geq 2$ *and* $t_3 \leq t - 2$.

Proof. Note that $k > 0$ because otherwise $Solve2ASLASAT(F, L, l, k)$ does not apply itself recursively. Observe also that $SepSize(F, \neg L, \neg l) = 0$ because clause C can be selected only on 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 $2k - Sepsize(F, \neg L, \neg l) = 2k \geq 2$ and hence $t = \beta(F, L, l, k) = 2k \geq 2$. 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 w.l.o.g. 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 6 and 7, and Lemma 7, this is at most $\sqrt{5}^{t-1} + \sqrt{5}^{t-2} + 1$. Then $\sqrt{5}^t - \sqrt{5}^{t-1} - \sqrt{5}^{t-2} - 1 \geq \sqrt{5}^2 - \sqrt{5}^{2-1} - \sqrt{5}^{2-2} - 1 = 5 - \sqrt{5} - 2 > 0$, the first inequality follows from Claim 7. That is, the present theorem holds for the considered case.

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

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 $2 * \sqrt{5}^{t^*}$ 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, $t^* \leq t - 2$ 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 6, 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 proof of Claim 7, 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 $SepSize(F_i, \neg L_i, l) > 0$. 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 the pair of consecutive nodes (F', L', l, k') and (F'', L'', l, k''), being the former 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 change 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 $tt_i \leq t - 2$ by Lemma 8 and the present claims follows 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 7 and 8, is at most $5 * \sqrt{5}^{t-2} = \sqrt{5}^t$. ■

Let (F, L, l) be an instance of the 2-ASLASAT problem. We define the implication graph $D = (V, A)$ of (F, L, l) as a slight modification of the implication graph of F. In particular, $V(D)$ corresponds to all the literals of $Var(F) \cup Var(L \cup \{l\})$, the arcs are defined analogously to the implication graph of F.

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

Proof. According to assumptions of the theorem, (F, L, l, k) is a valid input. Assume that the instance is represented by the implication graph $D = (V, A)$ of (F, L, l) together with specifying parameter k and marking the vertices corresponding to L $\neg L$, l, and $\neg l$ (this representation can be obtained in polynomial time from any other reasonable representation). It follows from Theorem 4 that $Solve2ASLASAT(F, L, l, k)$ correctly solves the parameterized 2-ASLASAT problem with respect to the given input. Let us evaluate the complexity of $Solve2ASLASAT(F, L, l, k)$. According to Lemma 8, the height of the search tree is at most $\alpha(F, L, l, k) \leq n + k$. 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 $t \leq 2k$, 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 $5^k * (n + k)$. The complexity of $Solve2ASLASAT(F, L, l, k)$ 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 $Solve2ASLASAT(F, L, l, k)$ 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. This notice is important because most of operations of $Solve2ASLASAT(F, L, l, k)$ involve doing Depth-First Search (DFS) or Breadth-First Search (BFS) on graph D, which take $O(V + A)$. In our case $|V| = O(n + |L|)$ and $|A| = O(m)$. Since $n = O(m)$, $O(V + A)$ can be replaced by $O(m + |L|)$.

The first operation performed by $Solve2ASLASAT(F, L, l, k)$ 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 [14] that the given 2-CNF formula F' is *not* satisfiable if and only if there are literals l' and $\neg l'$ which 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 $O(m + |L|)$ vertices and $O(m + |L|)$ 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 $O(m + |L|)$. Steps 2 and 3 take $O(1)$. 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 $O(k)$ iterations of the Ford-Fulkerson algorithm, where each iteration is a run of BFS and hence can be performed on $O(m + |L|)$. Consequently, Step 4 can be performed in $O((m + |L|) * k)$. Checking the condition of Step 5 can be done by BFS and hence takes $O(m + |L|)$. 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 $O(m + |L|)$. 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 clearly corresponds to a walk from l to $\neg l$ in D), find a satisfying assignment 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 $O(m+|L|)$, hence Step 6 takes this time. Note that preparing an input for a recursive call takes $O(1)$ because this preparation includes removal of one clause from F or adding one literal to L. Therefore Steps 7 and 8 take $O(1)$. Step 9 takes $O((m + |L|) * k)$ on the account of neutrality checking: $O(k)$ iterations of the Ford-Fulkerson algorithm are sufficient because $SepSize(F, \neg L, \neg l) \leq k$ due to insatisfaction of the condition of Step 4. Step 10 takes $O(1)$ on the account of input preparation for the recursive call. Thus the complexity of processing (F, L, l, k) is $O((m + |L|) * k)$.

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 of graph in the path from (F, L, l, k) to (F', L', l, k') is caused by 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 $O((m+|L|)*k)$ and the time taken by the entire run of $Solve2ASLASAT(F,L,l,k)$ is $O(5^k*k(n+k)*(m+|L|)*k)$ as required. ■

Theorem 6 immediately implies the following Corollary.

Corollary 3. *Let (F,l,k) be an instance of the parameterized 2-SLASAT problem. With respect to this input the problem can be solved in $O(5^k*k*(n+k)*m)$.*

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 parameterized 2-ASAT problem is FPT. In particular, let (F,k) be the input of this problem. Then it can be solved in $O(15^k*k*m^3)$, where $m=|Clauses(F)|$.*

Proof. We introduce the following 2 intermediate problems.

Problem I1
Input: A satisfiable 2-CNF formula F, a non-contradictionary set of literals L, a parameter k
Output: A set $S \subseteq Clauses(F)$ such that $|S| \leq k$ 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 through transformation of its instance into an instance of 2-SLASAT problem and of Problem I2 through transformation of its instance into an instance of Problem I1. Then we will show that the 2-SAT problem can be solved through transformation of its instance into an instance of Problem I2.

Claim 9 *Problem I1 with the input (F,L,k) can be solved in $O(5^k*k*m^2)$, where, $m=|Clauses(F)|$.*

Proof. Let S' be the set of all clauses of F both literals of which belong to $\neg L$. Clearly all such clauses have to be removed in order to make the resulting formula satisfiable w.r.t. L. Therefore if $|S'|>k$ we return 'NO'. Otherwise we solve the instance $(F \setminus S', L, k-|S'|)$. If the output is a set S, we return $S \cup S'$, otherwise we return 'NO'. Let S'' be the set of all clauses of F containing literals of L. Then the output of Problem I1 on $(F \setminus S'', L, k)$ is a valid output of this problem on input (F, L, k). Really, assume that the output of Problem I1 on instance $(F \setminus S'', L, k)$ is a set S. Then $SWRT(F \setminus S, L)$ is true because any satisfying assignment P of $(F \setminus S'') \setminus S$ which does not intersect with $\neg L$ can be extended by those literals of L that are included into the clauses of S'' (note that L is non-contradictionary) and by arbitrary literals of $Var(S'') \setminus (Var(P) \cup Var(L))$ to form a satisfying assignment of $F \setminus S$ which does not intersect with $\neg L$. If the output of Problem I1 on $(F \setminus S'', L, k)$ is 'NO' then, clearly, the output on (F, L, k) is 'NO' as well.

Assume that F does not contain clauses both literals of which are negations of L and does not contain clauses including literals of L. The argumentation in the previous paragraph shows that this assumption is valid. We may also assume that $SWRT(F, L)$ is false because otherwise we can immediately return the empty set. Let S^* be the set of clauses of F including literals of $\neg L$. For each $C \in S^*$ introduce a unique literal l_C such that $Var(l_C) \notin Var(F)$ and for any distinct C_1, C_2 of F, $Var(l_{C_1}) \neq Var(l_{C_2})$. Also introduce an additional literal l^* such that $Var(l^*) \neq Var(l_C)$ for any $C \in S^*$ and $Var(l^*) \notin Var(F)$. Let F^* be a 2-CNF formula obtained from F by the replacement of each $C \in S^*$ by two clauses done as follows. Let $C = (l_1 \vee l_2)$ and assume that $l_1 \in \neg L$. We replace C by clauses $(\neg l^* \vee l_C)$ and $(\neg l_C \vee l_2)$. Note that due to the uniqueness of l_C for each clause C, the proposed transformation does not produce clause repetitions. Observe that F^* is satisfiable. Really, let P be a satisfying assignment of F. Let P^* be an assignment obtained from P by replacement all the literals of $\neg L$ by $\neg l^*$ (note that P contains at least one literal of $\neg L$ because we assumed that $SWRT(F, L)$ is false) and adding l_C or $\neg l_C$ for each $C \in S^*$ as follows. For $C = (l_1 \vee l_2)$ such that $l_1 \in \neg L$, if $l_1 \in P$ then $\neg l_C \in P^*$. Otherwise $l_C \in P^*$. In other words, P^* is explicitly constructed so that both $(\neg l^* \vee l_C)$ and $(\neg l_C \vee l_2)$ are satisfied. It is easy to observe that P^* is a satisfying assignment of F^*.

Solve the parameterized 2-SLASAT problem with respect to the instance (F^*, l^*, k). Assume that the output is a set $T^* \subseteq Clauses(F^*)$ and such that $|T^*| \leq k$ and $SWRT(F^*, l^*)$ is true. Create a set $T \subseteq Clauses(F)$ having the following properties. $T \cap T^* = T^* \cap (Clauses(F) \cap Clauses(F^*))$ and a clause $C \in S^*$ belongs to T whenever the clause including l_C or the clause including $\neg l_C$ belong to T^*. No other clauses are contained in T. It is easy to see that $|T| \leq |T^*|$. Let us show that $SWRT(F \setminus T, L)$ is true. Let P^* be a satisfying assignment of $F^* \setminus T^*$ such that $\neg l^* \notin P^*$. Observe that for each $C = (l_1 \vee l_2)$ of S^* (we assume w.l.o.g. that $l_1 \in \neg L$) such that both $(\neg l^* \vee l_C)$ and $(\neg l_C \vee l_2)$ belong to $Clauses(F^* \setminus T^*)$, $l_C \in P^*$ and $l_2 \in P^*$. Thus the subset P' of P^* containing the literals of the variables of F satisfies all the clauses of $Clauses(F) \cap Clauses(F^*)$ and all the clauses of S^* which do not belong to T. In other words, P' satisfies all the clauses of $F \setminus T$. By definition P' does not intersect with $\neg L$. Let L' be the subset of literals of L such that for each $l' \in L'$ there is a clause including $\neg l'$ in $Clauses(F \setminus T)$. Then $Var(P' \cup L') = Var(F \setminus T)$ and hence $P' \cup L'$ is a satisfying assignment of $F \setminus T$ which does not contain $\neg L$. Thus in the considered case T is the output of Problem I1.

Assume that the output of the 2-SLASAT problem for instance (F^*, l^*, k) is 'NO'. We shall show that the output of Problem I1 for instance (F, L, k) is also 'NO'. Assume by contradiction that there is a set $T \subseteq Clauses(F)$ such that $|T| \leq k$ and $SWRT(F \setminus T, L)$ is true. Create a set $T^* \subseteq Clauses(F^*)$ having the following properties. $T \cap T^* = T^* \cap (Clauses(F) \cap Clauses(F^*))$ and whenever $C \in T \cap S^*$ where $C = (l_1 \vee l_2)$ and $l_1 \in \neg L$, $(\neg l_C \vee l_2) \in T^*$. No other clauses are contained in T^*. Clearly $|T^*| = |T|$. Let P^* be the set of literals obtained from P by removal all the literals of L, adding the literals l_C for each $C \in S^*$, and adding l^*. Observe that all the clauses of $F^* \setminus T^*$ are satisfied by P^*. In particular, the clauses of $Clauses(F) \cap Clauses(F^*)$ are satisfied by $P \cap P^*$ each clause containing l_C is satisfied by description, and each clause $(\neg l_C \vee l_2)$ is satisfied because $l_2 \in P \cap P^*$. For the last statement note that $(\neg l_C \vee l_2) \in Clauses(F^* \setminus T^*)$ whenever the respective clause $(l_1 \vee l_2)$ belongs to $Clauses(F \setminus T)$. The literal $l_1 \in \neg L$ cannot be contained in P, hence $l_2 \in P$. Taking into account that $Var(P^*) = Var(F^* \setminus T^*)$, P^* is a satisfying assignment of $F^* \setminus T^*$ which does not contain $\neg l^*$, which contradicts the output 'NO' of the 2-SLASAT problem with respect to input (F^*, l^*, k). It follows that the output of Problem I1 for input (F, L, k) is also 'NO'.

Thus we have shown that Problem I1 with input (F, L, k) can be solved by solving the 2-ASLASAT problem with input (F^*, l^*, k). Observe that $|Clauses(F^*)| = O(m)$ and $|Var(F^*)| = O(m + |Var(F)|)$. Taking into account our note in the proof of Theorem 6 that $|Var(F)| = O(m)$, $|Var(F^*)| = O(m)$. Also note that we may assume that $k < m$ because otherwise the algorithm can immediately returns $Clauses(F^*)$ without recursive applications. Substituting this data into the runtime of 2-SLASAT problem following from Theorem 6, we obtain the desired runtime for Problem I1. □

Claim 10 *Problem I2 with input (F, S, k) can be solved in time $O(15^k * k * m^2)$, where, $m = |Clauses(F)|$.*

Proof We solve Problem I2 by 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 $(l_1 \vee l_2)$ and creating L as the set of all chosen literals. For all the resulting pairs (E, L) such that L is non-contradictionary, 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-contradictionary, 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 $|Y| \leq k$ 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 subsets 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 9, we obtain the desired runtime for Problem I2. □

Let (F, k) be an instance of 2-ASAT problem. Let $C_1, \dots, C_m$ be the clauses of F. Let $F_0, \dots, F_m$ be 2-CNF formulas such that F_0 is the empty formula and for each i from 1 to m, $Clauses(F_i) = \{C_1, \dots C_i\}$. We solve (F, k) by the method of iterative compression [13]. In particular we solve the 2-ASAT problems $(F_0, k), \dots (F_m, k)$ 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), $i \leq m$, then clearly 'NO' can be returned for (F, k). Clearly, for (F_0, k), $S_0 = \emptyset$. It remains to show how to get S_i from S_{i-1}. Let $S'_i = S_i \cup \{C\}$. If $|S'_i| \leq k$ 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.

In follows that 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 10, Problem I2 can be solved in $O(15^k * k * m^2)$, so 2-ASAT problem can be solved in $O(15^k * k * m^3)$. ■

Finally, we show that the 2-ASAT problem remains FPT if the formula at the input contains repeated occurrences of some clauses.

Theorem 8. *The 2-ASAT problem with input (F, k) where F is a 2-CNF formula with possible repeated occurrences of clauses, can be solved in $O(15^k * k * m^3)$, where m is the number of clauses of F.*

Proof. We transform F into a formula F^* with all clauses being pairwise distinct and show that F can be made satisfiable by removal of at most k clauses if and only if F^* can. Then we apply Theorem 7.

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 $Var(F)$. Replace the i-th 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 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 $S^* = \{(l' \vee l_i) | i \in I\}$. 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 assignment which 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 $|S^*| \geq |I|$. Let F' be the formula obtained from F by removal the clauses whose indices belong to I. Observe that a clause $(l' \vee l'')$ belongs to 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 out put on input (F, k) is 'NO' as well. To obtain the desired runtime, note that F^* has $2m$ clauses and $O(m)$ variables and substitute this data to the runtime for 2-ASAT problem stated in Theorem 7. ■

References

1. J. Bang-Jensen, G. Gutin *D*igraphs: Theory, Algorithms and Applications, Springer-Verlag, 2001.
2. J. Chen, Y. Liu, S. Lu *A*n Improved Parameterized Algorithm for the Minimum Node Multiway Cut Problem. Algorithms and Data Structures, 10th International Workshop, WADS 2007, Halifax, Canada, August 15-17, 2007, pp. 495-506.
3. J. Chen, F. Fomin, Y. Liu, S. Lu, Y. Villanger, *Improved Algorithms for the Feedback Vertex Set Problems.* Algorithms and Data Structures, 10th International Workshop, WADS 2007, Halifax, Canada, August 15-17, 2007, pp. 422-433.
4. J. Chen, Y. Liu, S. Lu, B. O'Sullivan, I. Razgon,*Directed Feedback Vertex Set is Fixed-Parameter Tractable*, submitted to STOC 2008.
5. J. Chen, I. Kanj, *Improved Exact Algorithms for MAX-SAT*, Discrete Applied Mathematics, 142(1-3),2004,pp. 17-27.
6. E. Demaine, G. Gutin, D. Marx, U. Stege,*Open Problems from Dagstuhl Seminar 07281*, avaiable electronically at `http://drops.dagstuhl.de/opus/volltexte/2007/1254/pdf/07281.SWM.Paper.1254.pdf`
7. M. Grohe, M. Grüber, *Parameterized Approximability of the Disjoint Cycle Problem.* Automata, Languages and Programming, 34th International Colloquium, ICALP 2007, Wroclaw, Poland, July 9-13, 2007, pp. 363-374.
8. J. Guo, F. Hüffner E. Kenar, R. Niedermeier, J. Uhlmann *Complexity and Exact Algorithms for Multicut.* SOFSEM 2006: Theory and Practice of Computer Science, 32nd Conference on Current Trends in Theory and Practice of Computer Science, Mern, Czech Republic, January 21-27, 2006, pp. 303-312.
9. F. Hüffner *Algorithm Engineering for Optimal Graph Bipartization.* Experimental and Efficient Algorithms, 4th InternationalWorkshop, WEA 2005, Santorini Island, Greece, May 10-13, 2005, pp.240-252.

10. F. Hüffner, R. Nidermeier, S. Wernicke, *Techniques for Practical Fixed-Parameter Algorithms.*, to appear in The Computer Journal. Available electronically at http://comjnl.oxfordjournals.org/cgi/reprint/bxm040?ijkey=OL6dCDrinntlpts&keytype=ref
11. S. Khot, V. Raman, *Parameterized Complexity of Finding Subgraphs with Hereditary Properties.*, Computing and Combinatorics, 6th Annual International Conference, COCOON 2000, Sydney, Australia, 2006, pp. 137-147.
12. D. Marx *Parameterized graph separation problems.* Theoretical Computer Science,351(3), 2006, pp. 394-406.
13. R. Niedermeier, *Invitation to fixed-parameter algorithms*, Oxford Lecture Series in Mathematics and its Applications, volume 31, 2006.
14. C. Papadimitriou, *Computational Complexity*, Addision-Wesley, 1994.
15. B. Reed, K. Smith, A. Vetta, *Finding odd cycle transversals.* Operations Research Letters 32(4),2004, pp.299-301.
16. A. Slivkins, *Parametrized Tractability of Edge-Disjoint Paths on DAGs*, Algorithms - ESA 2003, 11th Annual European Symposium, Budapest, Hungary, September 16-19, 2003, pp.482-493.