Almost 2-SAT Is Fixed-Parameter Tractable (Extended Abstract)

Igor Razgon and Barry O'Sullivan

Cork Constraint Computation Centre
Computer Science Department, University College Cork, Ireland
{i.razgon,b.osullivan}@cs.ucc.ie

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, and 2-SAT deletion. The status of the fixed-parameter tractability of this problem is a long-standing open question in the area of parameterized complexity. We resolve this open question by proposing an algorithm that solves this problem in $O(15^k * k * m^3)$ time showing that this problem is fixed-parameter tractable.

1 Introduction

We consider the following problem. Given a 2-CNF formula, is it possible to remove at most k clauses so that the resulting 2-CNF formula is satisfiable? This problem is known to different research communities in theoretical computer science under the names Almost 2-SAT, All-but-k 2-SAT, 2-CNF deletion, and 2-SAT deletion. The status of the fixed-parameter tractability of this problem is a long-standing open question in the area of parameterized complexity. The question regarding the fixed-parameter tractability of this problem was first raised in 1997 by Mahajan and Raman [11,12]. This question has been posed in the book of Niedermeier [15], being referred as one of central challenges for parameterized algorithms design. Finally, in July 2007, this question was included by Fellows in the list of open problems of the Dagstuhl seminar on Parameterized Complexity [5]. In this paper we resolve this open question by proposing an algorithm that solves this problem in $O(15^k * k * m^3)$ time. Thus we show that this problem is fixed-parameter tractable (FPT).

Regarding the name of this problem, we call *Almost 2-SAT* (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 formula is satisfiable. The *parameterized* 2-ASAT problem gets as additional input a parameter k, and the corresponding decision problem is to determine whether at most k clauses can be removed so that the resulting formula becomes satisfiable. The algorithm proposed in this paper solves the parameterized 2-ASAT problem.

Overview of the Algorithm. We define a variation of the 2-ASAT problem called the *Annotated* 2-ASAT *problem with a single literal* (2-ASLASAT). The

© Springer-Verlag Berlin Heidelberg 2008

input of this problem is a triple (F, L, l), where F is a 2-CNF formula, L is a set of literals such that F is *satisfiable with respect to* L (i.e. $F \wedge \bigwedge_{l' \in L} l'$ is satisfiable), l is a single literal. The task is to find a smallest subset of clauses of F such that after their removal the resulting formula is satisfiable with respect to $(L \cup \{l\})$. The description of the algorithm for the parameterized 2-ASAT problem is divided into two parts. In the first, and most important part we provide an $O(5^k * k * m^2)$ time algorithm that solves the parameterized 2-ASLASAT problem, where the parameter k is the maximum number of clauses to be removed and m is the number of clauses of F. In the second part we show that the parameterized 2-ASAT problem can be solved by $O(3^k * m)$ applications of the algorithm solving the parameterized 2-ASLASAT problem. The resulting runtime follows from the product of the last two complexity expressions. The transformation of the 2-ASAT problem into the 2-ASLASAT problem is based on *iterative compression* and can be seen as an adaptation of the method employed in [8] in order to solve the graph bipartization problem. In the following we overview the first part.

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

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

Related Work. As said above, the parameterized 2-ASAT problem has been introduced in [11]. In [10], this problem was shown to be a generalization of the parameterized graph bipartization problem, which was also an open problem at that time. The latter problem was resolved in [16]. The additional contribution of [16] was introducing a method of iterative compression that has had a considerable impact on the design of parameterized algorithms. The most recent algorithms based on this method are currently the best one for the undirected Feedback Vertex Set [1] and the first parameterized algorithm for the famous Directed Feedback Vertex Set problem [3]. For earlier results based on iterative compression, we refer the reader to a survey article [9].

The study of parameterized graph separation problems was initiated in [13]. The technique introduced by the author allowed him to design fixed-parameter algorithms for the multiterminal cut problem and for a more general multicut problem. The latter assumed that the number of pairs of terminals to be separated was also a parameter. The latter result was extended in [7] where fixed-parameter algorithms for multicut problems on several classes of graphs were proposed. The first $O(c^k * poly(n))$ algorithm for the multiterminal cut problem was proposed in [2]. A reformulation of the main theorem of [2] is an essential part of the parameterized algorithm for the Directed FVS problem [3] mentioned in the previous paragraph. In the present paper, we applied the strategy of proof of this theorem in order to show that adding a *neutral* literal to the set of literals of the input does not increase the solution size. Along with computing the separators, the methods of computing disjoint paths have been investigated. The research led to intractability results [17] and parameterized approximability results [6].

The parameterized MAX-SAT problem (a complementary problem to the one considered in the present paper) where the goal is to satisfy at least k clauses of arbitrary sizes also received a considerable attention from the researchers. The best currently known algorithm for this problem runs in $O(1.37^k + |F|)$, where $|F|$ is the size of the given formula [4].

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 that the 2-ASLASAT problem is fixed-parameter tractable. In Section 4 we show that the 2-ASAT problem is fixed-parameter tractable. We conclude by mentioning a number of problems known to be FPT-equivalent to parameterized 2-ASAT and notice that the fixed-parameter tractability of these problems follows as a by-product of our main result. Due to space constraints, proofs are either omitted or replaced by sketches.[1]

2 Terminology

A CNF formula F is called a 2-CNF *formula* if each clause of F is of size at most 2. Throughout the paper we make two assumptions regarding the 2-CNF formulas we consider. Firstly, we assume that all the clauses are of size 2. If a formula has a clause (l) of size 1 then this clause is represented as $(l \vee l)$. Secondly,

[1] A manuscript available at `http://arxiv.org/abs/0801.1300` contains a complete description of the result of the present paper with all the proofs and technical details.

everywhere except the very last theorem, we assume that all the clauses of any formula are pairwise distinct, i.e. no two clauses have the same set of literals. This assumption allows us to represent the operation of removing clauses from a formula in a set-theoretical manner. In particular, let S be a set of clauses. Then $F \setminus S$ is a 2-CNF formula that is the conjunction of clauses of F that are not contained in S. The result of removing a single clause C is denoted by $F \setminus C$ rather than $F \setminus \{C\}$.

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

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

Let F be a 2-CNF formula and L be a set of literals. F is *satisfiable with respect to* L if $F \wedge \bigwedge_{l' \in L} l'$ is satisfiable. The notion of satisfiability of a 2-CNF formula with respect to the given set of literals will be very frequently used in the paper, hence, in order to save the space, we introduce a special notation for this notion. In particular, we say that $\textsc{swrt}(F, L)$ 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 $\textsc{swrt}(F, l)$ rather than $\textsc{swrt}(F, \{l\})$.

Definition 1 (Walk of a 2-CNF). *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}. The walk w is a* path *if all its clauses are pairwise distinct.*

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 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. In general a literal of a clause may be both a first and a second with respect to the given walk.

Definition 2 (Culprit Sets, 2-ASAT and 2-ASLASAT Problems)

– *A <u>Culprit Set</u> (CS) of a* 2-CNF *formula F is a subset S of $Clauses(F)$ such that $F \setminus S$ is satisfiable. We call the problem of finding a* Smallest CS *(SCS) of F the <u>Almost 2-SAT Problem</u> (*2-ASAT *problem).*

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

In this paper we consider the parameterized versions of the 2-ASAT and 2-ASLASAT problems. In particular, the input of the *parameterized* 2-ASAT problem is (F, k), where F is a 2-CNF formula and k is a non-negative integer. The output is a CS of F of size at most k, if one exists. Otherwise, the output is 'NO'. The input of the *parameterized* 2-ASLASAT problem is (F, L, l, k) where (F, L, l) is as specified in Definition 3. The output is a CS of (F, L, l) of size at most k, if one exists. Otherwise, the output is 'NO'.

3 Parameterized Algorithm for the 2-ASLASAT Problem

We begin our analysis from the following Theorem.

Theorem 1. *Given a* 2-ASLASAT *problem instance (F, L, l), 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 $\neg l$.*

Theorem 1 allows us to view the 2-ASLASAT problem as a *separation* problem. In particular, a SCS of (F, L, l) can be viewed as the smallest number of clauses whose removal separates all walks from $\neg L$ to $\neg l$ and from $\neg l$ to $\neg l$. Our next step is to introduce a polynomially computable lower bound for the size of a SCS of (F, L, l).

Consider a smallest subset S of clauses such that $F \setminus S$ has no path from $\neg L$ to $\neg l$. We denote $|S|$ by $SepSize(F, \neg L, \neg l)$. From Theorem 1, $SepSize(F, \neg L, \neg l)$ is a lower bound on the size of an SCS of (F, L, l). In order to prove polynomial computability of $SepSize(F, \neg L, \neg l)$, we recall a well known notion of the *implication graph* of F. This is a digraph D whose set $V(D)$ of vertices 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 $(\neg l_1 \vee l_2) \in Clauses(F)$. It is easy to establish a one-to-one correspondence between the walks of F and the walks of D. In particular, a walk $w = (l_1 \vee \neg l_2), (l_2 \vee \neg l_3), \dots, (l_{t-1} \vee \neg l_t)$ of F from l_1 to $\neg l_t$ corresponds to the walk $w(D) = (\neg l_1, \neg l_2), (\neg l_2, \neg l_3), \dots, (\neg l_{t-1}, \neg l_t)$ in D from $\neg l_1$ to $\neg l_t$. The opposite correspondence holds as well. Observe that in the above walk each clause $C_i = (l_i \vee \neg l_{i+1})$ is *represented* by arc $e_i = (\neg l_i, \neg l_{i+1})$ and the first and the second literals of C_i with respect to w correspond to the tail and the head of e_i, respectively.

This correspondence suggests that a Menger-like dependence in F might hold, i.e. the number of clause-disjoint paths from $\neg L$ to $\neg l$ equals $SepSize(F, \neg L, \neg l)$, and that $SepSize(F, \neg L, \neg l)$ might be computed by a Ford-Fulkerson-like procedure. This statement would immediately follow if one established a one-to-one correspondence between the sets of clause-disjoint paths of F and the sets of clause-disjoint paths of D. The subtle point is that this correspondence does not hold in general. The reason is that a clause $(l_1 \vee l_2)$ where l_1 and l_2 are distinct is represented by two arcs of D: $(\neg l_1, l_2)$ and $(\neg l_2, l_1)$. It follows that a path of D may correspond to a walk of F which is not a path (i.e. has repeated clauses). Moreover, a set of arc-disjoint paths of D may correspond to a set of walks of F which are not clause-disjoint.

Fortunately, it is sufficient for us to establish the correspondence only between the paths from $\neg L$ in F and the paths from L in D. Taking into account that SWRT(F, L) is true, by definition of the 2-ASLASAT problem, this correspondence can be shown based on the following lemma.

Lemma 1. *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 with respect to w. Then l_1 is* not *a second literal of C with respect to any walk from $\neg L$.*

It immediately follows from this lemma that there are no two paths p_1 and p_2 of D starting at L such that $(\neg l_1, l_2)$ participates in p_1, while $(\neg l_2, l_1)$ participates in p_2 because it would mean that in the corresponding walks in F, l_1 is the first literal of C with respect to one of them and the second literal with respect to the other. Thus Lemma 1 eliminates the above obstacle to establishing the desired correspondence. Formalizing this argument allows us to prove the following theorem.

Theorem 2. *Given a 2-*ASLASAT *problem instance (F, L, l), $SepSize(F, \neg L, \neg l)$ equals the largest number of clause-disjoint paths from $\neg L$ to $\neg l$ in F as well as the largest number of arc-disjoint paths from L to $\neg l$ in D.*

Based on Theorems 1 and 2, we can give an informal outline of a parameterized algorithm for the 2-ASLASAT problem. For the main case of this algorithm we have an instance (F, L, l) of the problem where L is non-empty and select a clause $C = (l_1 \vee l_2)$ such that $l_1 \in \neg L$. We branch on the removal/non-removal of this clause. If this clause is not removed then any satisfying assignment with respect to L must contain $\neg l_1$ and, hence, also must contain l_2 in order to satisfy C. Therefore, we can say that we branch on the removal of this clause, or the addition of l_2 to L. The first branch decreases the parameter but the second branch does not. Hence the second branch is problematic for the design of the parameterized algorithm. One fortuitous case occurs if $SepSize(F, \neg(L \cup \{l_2\}), \neg l) > SepSize(F, \neg L, \neg l)$. According to the combination of Theorem 1 and Theorem 2, this condition means that adding l_2 to L increases a polynomially computable lower bound on the size of a SCS of (F, L, l). Therefore if C satisfies this fortuitious case then both branches *decrease* the gap between parameter and the lower bound: one by decreasing the parameter, the other by increasing the lower bound. It can be shown that if only such fortuitious clauses are selected then the algorithm terminates in $O^*(c^k)$. However what about the case where adding l_2 to L does not increase the lower bound? The following Theorem proves that in this case the size of a SCS of (F, L, l) is *not increased* as well and, hence, the non-removal decision can be safely made regarding C. That is, the branching is applied only for the fortuitious case, which leads to a parameterized algorithm.

Definition 3 (Neutral Literal). *Let (F, L, l) be an instance of the* 2-ASLASAT *problem. A literal l^* is a* neutral literal *of (F, L, l) if $(F, L \cup \{l^*\}, l)$ is an instance of the* 2-ASLASAT *problem and $SepSize(F, \neg L, \neg l) = SepSize(F, \neg(L \cup \{l^*\}), \neg l)$.*

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).*

Proof. (Sketch) We begin by introducing a number of sets. Let X be a SCS of (F, L, l), SP be a set of clauses of F such that $F \setminus SP$ has no path from $\neg(L \cup \{l^*\})$ to $\neg l$ and $|SP| = SepSize(F \neg(L \cup \{l^*\}), \neg l)$ (due to the neutrality of l^*, $|SP| = SepSize(F, \neg L, \neg l)$). Let R be the subset of all clauses C of $F \setminus SP$ such that $F \setminus SP$ has a walk $w_R(C)$ from $\neg L$ ending by C. We denote the rest of the clauses of $F \setminus SP$ by NR. We denote $X \cap R$ by XR. Finally, we denote by Y the set of all clauses C of $SP \setminus X$ such that there is a walk $w(C)$ having the following properties. The first clause of $w(C)$ is C, the last literal of $w(C)$ is $\neg l$, all the literals of $w(C)$ except C belong to $NR \setminus X$, and there is a walk from $\neg(L \cup \{l^*\})$ to $\neg l$ having $w(C)$ as a suffix. We are going to prove that $X^* = (X \setminus XR) \cup Y$ is a CS of $(F, L \cup \{l^*\}, l)$ and that $|Y| \leq |XR|$. The present theorem immediately follows from the combination of these two facts.

By Theorem 1, to prove that X^* is a CS of $(F, L \cup \{l^*\}, l)$ all we need to show is that $F \setminus X^*$ has no walk from $\neg(L \cup \{l^*\})$ to $\neg l$ and from $\neg l$ to $\neg l$. We show here only the former. Assume that $F \setminus X^*$ has a walk w^* from $\neg(L \cup \{l^*\})$ to $\neg l$ in contradiction with the considered statement. Then w^* intersects with SP by definition. Fix the *last* clause C of w^* such that $C \in SP$. We claim that $C \in Y$, which leads to a contradiction with $Y \subseteq X^*$ confirming the statement. To show this we assume the opposite and fix an entry $C' \in R$ of w^* following C. According to Lemma 1, C' has *the same* 'orientation' in both w^* and $w_R(C')$, hence we may replace the prefix of w^* ending with C' by $w_R(C')$. As a result we get a new walk w'' from $\neg L$ to $\neg l$ in $F \setminus X^*$ which meets SP *after* C'. That is, w^* meets SP after C' and hence after C in contradiction to the definition of C. This shows that $C \in Y$.

To show that $|Y| \leq |XR|$, we take a set $\mathbf{P}$ of $|SP|$ clause-disjoint paths of F from $\neg L$ to $\neg l$ guaranteed to exist according to Theorem 2. We observe that every path of $\mathbf{P}$ contains exactly one clause of SP and every clause of SP is contained in exactly one path of $\mathbf{P}$. Let $p \in \mathbf{P}$ be the path containing a clause $C \in Y$. We claim that C is preceded in C by a clause of XR. Otherwise, applying Lemma 1, we can replace the suffix of p starting from C by $w(C)$. As a result we get a walk from $\neg L$ to $\neg l$ that does not intersect with X, i.e. a walk from $\neg L$ to $\neg l$ in $F \setminus X$ which is impossible according to Theorem 1. Since $\mathbf{P}$ has $|Y|$ *clause-disjoint* paths containing the clauses of Y, $|XR| \geq |Y|$ as required. □

We provide the formal description of the algorithm below.

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

 1. if SWRT$(F, L \cup \{l\})$ is true then return $\emptyset$
 2. if $k = 0$ then Return 'NO'
 3. if $k \geq |Clauses(F)|$ 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 $Var(l_2) \notin Var(L)$
 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 [2]
 7. if Both $l_1$ and $l_2$ belong to $\neg(L \cup \{l\})$ then
    7.1 $S \leftarrow$ FINDCS$(F \setminus C, L, l, k-1)$
    7.2 if $S$ is not 'NO' then Return $S \cup \{C\}$
    7.3 Return 'NO'
 8. if Both $l_1$ and $l_2$ do not belong to $\neg(L \cup \{l\})$ then
    8.1 $S_1 \leftarrow$ FINDCS$(F, L \cup \{l_1\}, l, k)$
    8.2 if $S_1$ is not 'NO' then Return $S_1$
    8.3 $S_2 \leftarrow$ FINDCS$(F, L \cup \{l_2\}, l, k)$
    8.4 if $S_2$ is not 'NO' then Return $S_2$
    8.5 $S_3 \leftarrow$ FINDCS$(F \setminus C, L, l, k-1)$
    8.6 if $S_3$ is not 'NO' then Return $S_3 \cup \{C\}$
    8.7 Return 'NO'
    (In the rest of the algorithm we consider the cases where exactly one literal of $C$
    belongs to $\neg(L \cup \{l\})$. W.l.o.g. we assume that this literal is $l_1$)
 9. if $l_2$ is not neutral in $(F, L, l)$ then
    9.1 $S_2 \leftarrow$ FINDCS$(F, L \cup \{l_2\}, l, k)$
    9.2 if $S_2$ is not 'NO' then Return $S_2$
    9.3 $S_3 \leftarrow$ FINDCS$(F \setminus C, L, l, k-1)$
    9.4 if $S_3$ is not 'NO' then Return $S_3 \cup \{C\}$
    9.5 Return 'NO'
10. Return FINDCS$(F, L \cup \{l_2\}, l, k)$
```

The algorithm is presented as a function $FindCS(F, L, l, k)$. The first part of the algorithm (lines 1-4) is processing the stopping conditions. Lines 1-3 are trivial. Line 4 is correct because $SepSize(F \neg L, \neg l)$ is a lower bound on the size of a SCS of (F, L, l) according to Theorem 1. The second part of the algorithm is selecting the clause C to be considered during the branching process. The selection procedure is designed so that it guarantees that if a literal l' is added to the set L then $Var(l') \notin Var(L)$. This ensures that on any path from the root of the search tree to the leaves there may be at most n nodes that add literals to L, where $n = |Vars(F)|$. Taking into account that along a path in a search tree at most k nodes that remove clauses can occur, we derive that the height of the search tree is at most $n + k$.

The remaining part of the algorithm describes the process of applying an appropriate branching rule depending on the literals of clause C. The correctness

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

of the branching rules is based on the observation that if C does not belong to the CS S of (F, L, l) being constructed, then any satisfying assignment of $F \setminus S$, including the one that does not intersects with $\neg(L \cup \{l\})$, has to satisfy C. It follows that on the branches where C is not removed, at least one literal of C is added to L. There are two cases where branching is not performed at all. The first case occurs if the condition of line 7 is satisfied: in this case C itself is not satisfiable with respect to $L \cup \{l\}$. Consequently, C belongs to any CS of (F, L, l). The second case occurs in line 10. The correctness of this step follows from Theorem 3 by taking into account that l_2 is a neutral literal with respect to (F, L, l).

The key part of the runtime analysis is to prove that as a result of adding a literal l' of clause C to L on lines 8.1, 8.3, and 9.1, $SepSize(F, \neg(L \cup \{l'\}), \neg l) > SepSize(F, \neg L, \neg l)$. Regarding line 9.1, the algorithm explicitly states that $l' = l_2$ is not neutral and hence the required property follows from Definition 3. Regarding lines 8.1 and 8.3 our proof uses the following intuitive argument. A clause with both literals outside $\neg(L \cup \{l\})$ can be selected only on line 6 i.e in the case where $SepSize(F, \neg L, \neg l) = 0$. The selected clause $(l_1 \vee l_2)$ belongs to a walk w from $\neg l$ to $\neg l$ in F. We also show that $Var(l_1) \neq Var(l)$ and that $Var(l_2) \neq Var(l)$. Then we derive from the combination of these facts that there are subwalks of w that are walks from $\neg l_1$ to $\neg l$ and from $\neg l_2$ to $\neg l$. Consequently, F has a walk from $\neg(L \cup \{l_i\})$ to $\neg l$ for $i = 1, 2$. It can be shown that in this case, the respective paths also exist which implies by Theorem 2 that $SepSize(F \neg(L \cup \{l_i\}), \neg l) > 0 = SepSize(F, \neg L, \neg l)$.

To prove that the exponential part of the runtime of $FindCS$ depends only on k we show that the number of leaves of the search tree depends only on k. In order to do this we introduce a measure on (F, L, l, k) which is bounded from above by a function of k and which is decreased by each branch whenever a branching rule with 2 or 3 branches is applied. A measure $\beta = \beta(F, L, l, k) = 2k - SepSize(F, \neg L, \neg l)$ satisfies these requirements.[3] Indeed, if we add a non-neutral literal to L then the second item increases and hence the whole measure decreases. If we remove a clause from F (thus decreasing k) then the first item decreases by 2 while the second item decreases by at most 1, thus the whole measure decreases again. Taking into account that $\beta \leq 2k$, we obtain that on each path from the root of the search tree to a leaf at most $2k$ nodes with 2 or 3 outgoing branches. Since each node of the search node has at most 3 outgoing branches, the number of leaves of the search tree is at most $3^{2k} = 9^k$. This already implies the fixed-parameter tractability of the 2-ASLASAT problem. Using a more careful assessment we reduce the upper bound on the number of leaves to 5^k. We omit the details here due to the lack of space. As we noticed above, the height of the search tree is at most $n + k$, hence the number of nodes of the search tree is at most $(n + k)5^k$. It is also not hard to show that $(n + k) = O(m)$, where $m = |Clauses(F)|$. Therefore, the number of nodes of the search tree is bounded by $O(5^k m)$.

[3] In fact, we use the measure $max(0, 2k - SepSize(F, \neg L, \neg l))$. We demonstrate our argument on a simplified measure to make it more intuitive.

In the remaining part of the analysis we notice that the heaviest operations performed by $FindCS$ per node of the search tree are checking on line 4 whether the lower bound is exceeded or not and neutrality checking on line 9. According to Theorem 2, these operations can be performed by $O(k)$ iterations of the Ford-Fulkerson algorithm applied to the implication graph of F. The runtime of each iteration is $O(m + |L|)$, where the additional item $|L|$ takes into account the literals whose variables do not belong to $Vars(F)$. Combining this with the bound on the number of nodes obtained in the previous paragraph, we get the following theorem.

Theorem 4. *The* 2-ASLASAT *problem is fixed-parameter tractable. It particular it can be solved in* $O(5^k km(m + |L|)$ *time.*

4 Algorithm for the 2-ASAT Problem

In the final part of our proof of the fixed-parameter tractability of the 2-ASAT problem we show that it can be solved by $O(3^k m)$ calls to a procedure solving the 2-ASLASAT problem. First we get rid of the repeated clauses by associating with each clause $C = (l' \vee l'')$ of the given formula a unique literal l_i and replacing C by a pair of clauses $(l' \vee l_i)$ and $(\neg l_i \vee l'')$. Note that the number of clauses of the resulting formula remains $O(m)$. Thus we may assume that in the given instance (F, k) of the 2-ASAT problem F has no repeated clauses. Next we observe that the 2-ASAT problem can be solved by $O(m)$ calls to a problem with input (F_1, S_1, k), where F_1 is a 2-CNF formula with $Clauses(F_1) \subseteq Clauses(F)$, and S_1 is a CS of F_1 of size $k + 1$. This transformation is known to the parameterized complexity community under the name *iterative compression* [9].

Next we observe that F_1 has a CS of size at most k if and only if there a set $I \subset S_1$ such that there is a subset S_2 of $Clauses(F_1 \setminus I)$ such that $S_2 \cap S_1 = \emptyset$, $|S_2| \leq k - |I|$, and S_2 is a CS of $F_1 \setminus I$. Thus in order to find the required CS of F_1, we explore 2^{k+1} subsets of S_1 and for each subset I we solve the problem with input (F_2, S_2, k_2), where $F_2 = F_1 \setminus I$, $S_2 = S_1 \setminus I$, $k_2 = k - |I|$. The output of the problem is a CS of F_2 having size at most k_2 and disjoint with S_2.

Set $F_3 = F_2 \setminus S_2$ and observe that the set Y is the required CS of F_2 if and only if $|Y| \leq k_2$ and there is a non-contradictory set of literals satisfying $F_3 \setminus Y$ and all the clauses of S_2. The last condition can be reformulated as follows: there is a non-contradictory set L_3 of literals satisfying the clauses of S_2 such that $F_3 \setminus Y$ is satisfiable with respect to L_3. Our next transformation is based on this view. In particular, we explore all possible non-contradictory sets of literals obtained by taking one literal from each clause of S_2 (there may be at most 2^{k_2+1} such combinations) and for each considered set L_3 we solve a problem with input (F_3, L_3, k_2) whose output is a set such that $Y \subseteq Clauses(F_3)$, $|Y| \leq k_2$, and SWRT$(F_3 \setminus Y, L_3)$ is true. Note that F_3 is satisfiable because $F_3 = F_2 \setminus S_2 = F \setminus S$ while S is a CS of F.

In the last stage, based on the satisfiability of F_3, we guess a satisfying assignment P_3 of F_3. If P_3 does not intersect with $\neg L_3$ we return the empty set. Otherwise we partition L_3 into the subsets L'_3 and L''_3 such that P_3 does not intersect with L'_3, while $\neg L''_3 \subseteq P_3$. Then we introduce two new literals l^*_1 and l^*_2 and transform F_3 into a 2-CNF F^* by replacing the literals of L'_3 by l^*_1, the literals of $\neg L'_3$ by $\neg l^*_1$, the literals of L''_3 by l^*_2, and the literals of $\neg L''_3$ by $\neg l^*_2$. We observe that the set of literals P^* obtained from P_3 by the analogous replacement is a satisfying assignment of F^* that does not contain $\neg l^*_1$, from which we conclude that SWRT(F^*, l^*_1) is true. It follows that $(F^*, \{l^*_1\}, l^*_2)$ is a valid instance of the 2-ASLASAT problem. Moreover, we show that $(F^*, \{l^*_1\}, l^*_2)$ has a CS Y^* of size at most k_2 if and only if there is a set Y, $|Y| \leq k_2$ such that SWRT$(F_3 \setminus Y, L)$ is true and show a transformation from Y^* to Y. Taking into account that the 2-ASLASAT problem is known to be FPT according to Theorem 4, it follows that the 2-ASAT problem is FPT as well.

It follows from the above description that 2-ASAT problem can be solved by $O(4^k m)$ calls to an algorithm solving the 2-ASLASAT problem. A simple combinatorial argument decreases the upper bound to $O(3^k m)$. Combining this with Theorem 4 we obtain the following theorem.

Theorem 5. *The* 2-ASAT *problem is fixed-parameter tractable. In particular, it can be solved in* $O(15^k k m^3)$ *time.*

5 Concluding Remarks

We conclude by noting a number of consequences of our main result. It was noticed in [5] that the parameterized 2-ASAT problem is FPT-equivalent to the following problem: given a graph G having a perfect matching, find whether G has a vertex cover of size at most $n/2 + k$. This problem is called vertex cover problem parameterized above the perfect matching (VC-PM). It is shown [14] that the VC-PM problem is FPT-equivalent to the vertex cover problem parameterized above the size of a maximum matching and that the latter problem is FPT-equivalent to a problem of finding whether at most k vertices can be removed from the given graph so that the size of the minimum vertex cover of the resulting graph equals the size of its maximum matching. It follows from Theorem 5 that all these problems are fixed parameter tractable.

Acknowledgements

We thank Venkatesh Raman for pointing out to several relevant references, Somnath Sikdar for his help in fixing a bug in an earlier version of the archived manuscript, and an anonymous reviewer for suggestions of improvement the presentation of the extended abstract. Finally, we acknowledge the Science Foundation Ireland for supporting our research through grant 05/IN/I886.

References

1. Chen, J., Fomin, F., Liu, Y., Lu, S., Villanger, Y.: Improved algorithms for the feedback vertex set problems. In: Dehne, F., Sack, J.-R., Zeh, N. (eds.) WADS 2007. LNCS, vol. 4619, pp. 422–433. Springer, Heidelberg (2007)

2. Chen, J., Liu, Y., Lu, S.: An improved parameterized algorithm for the minimum node multiway cut problem. In: Dehne, F., Sack, J.-R., Zeh, N. (eds.) WADS 2007. LNCS, vol. 4619, pp. 495–506. Springer, Heidelberg (2007)
3. Chen, J., Liu, Y., Lu, S., O'Sullivan, B., Razgon, I.: A fixed-parameter algorithm for the directed feedback vertex set problem. In: STOC 2008 (to appear, 2008)
4. Chen, J., Kanj, I.A.: Improved exact algorithms for $\mathrm{m_{ax}\text{-}s_{at}}$. Discrete Applied Mathematics 142(1-3), 17–27 (2004)
5. Demaine, E., Gutin, G., Marx, D., Stege, U.: Open problems from dagstuhl seminar 07281 (2007), `http://drops.dagstuhl.de/opus/volltexte/2007/1254/pdf/07281.SWM.Paper.1254.pdf`
6. Grohe, M., Grüber, M.: Parameterized approximability of the disjoint cycle problem. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 363–374. Springer, Heidelberg (2007)
7. Guo, J., Hüffner, F., Kenar, E., Niedermeier, R., Uhlmann, J.: Complexity and exact algorithms for multicut. In: SOFSEM, pp. 303–312 (2006)
8. Hüffner, F.: Algorithm engineering for optimal graph bipartization. In: Nikoletseas, S.E. (ed.) WEA 2005. LNCS, vol. 3503, pp. 240–252. Springer, Heidelberg (2005)
9. Hüffner, F., Niedermeier, R., Wernicke, S.: Techniques for practical fixed-parameter algorithms. The Computer Journal 51(1), 7–25 (2008)
10. Khot, S., Raman, V.: Parameterized complexity of finding subgraphs with hereditary properties. Theoretical Computer Science 289(2), 997–1008 (2002)
11. Mahajan, M., Raman, V.: Parametrizing above guaranteed values: Maxsat and maxcut. Electronic Colloquium on Computational Complexity (ECCC) 4(33) (1997)
12. Mahajan, M., Raman, V.: Parameterizing above guaranteed values: Maxsat and maxcut. Journal of Algorithms 31(2), 335–354 (1999)
13. Marx, D.: Parameterized graph separation problems. Theoretical Computer Science 351(3), 394–406 (2006)
14. Mishra, S., Raman, V., Saurabh, S., Sikdar, S., Subramanian, C.: The complexity of finding subgraphs whose matching number equals the vertex cover number. In: ISAAC, pp. 268–279 (2007)
15. Niedermeier, R.: Invitation to fixed-parameter algorithms. Oxford Lecture Series in Mathematics and Its Applications, vol. 31 (2006)
16. Reed, B., Smith, K., Vetta, A.: Finding odd cycle transversals. Operations Research Letters 32(4), 299–301 (2004)
17. Slivkins, A.: Parameterized tractability of edge-disjoint paths on directed acyclic graphs. In: Di Battista, G., Zwick, U. (eds.) ESA 2003. LNCS, vol. 2832, pp. 482–493. Springer, Heidelberg (2003)