Testing satisfiability of CNF formulas by computing a stable set of points

Eugene Goldberg
Cadence Berkeley Labs (USA)
egold@cadence.com

Abstract. *We show that a conjunctive normal form (CNF) formula F is unsatisfiable iff there is a set of points of the Boolean space that is stable with respect to F. So testing the satisfiability of a CNF formula reduces to looking for a stable set of points (SSP). We give a simple algorithm for constructing a set of points that is stable with respect to a given set of clauses. Constructing an SSP can be viewed as a "natural"”way of search space traversal. This naturalness of search space examination allows one to make use of the regularity of CNF formulas to be checked for satisfiability. We illustrate this point by showing that if a CNF formula is symmetric with respect to a group of permutations, it is very easy to make use of this symmetry when constructing an SSP. As an example, we show that the unsatisfiability of pigeon−hole CNF formulas can be proven by examining only a linear size set of points that can be constructed in quadratic time.*

1. Introduction

A common belief is that there is no polynomial time algorithm for the satisfiability problem. Nevertheless, many classes of "real−life"”CNF formulas have structural properties that reduce (or may potentially reduce) the complexity of checking these CNF formulas for satisfiability from exponential to polynomial. However, the existing algorithms are not very good at taking into account structural properties of CNF formulas. One of the reasons is that currently there is no "natural"”way of traversing search space. For example, in the DPLL procedure [4], that is the basis of many algorithms used in practice, search is organized as a binary tree. In reality, the search tree is used only to impose a linear order on the points of the Boolean space to avoid visiting the same point twice. However, this order may be in conflict with “natural” relationships between points of the Boolean space that are imposed by the CNF formula to be checked for satisfiability (for example, if this formula has some symmetries).

In this paper, we introduce the notion of a stable set of points (SSP). We show that a CNF formula F is unsatisfiable if and only if there is a set of points of the Boolean space that is stable with respect to F. If F is satisfiable then any subset of points of the Boolean space is unstable, and an assignment satisfying F will be found in the process of SSP construction. We describe a simple algorithm for constructing an SSP.

An important fact is that, generally speaking, a set of points that is stable with respect to a CNF formula F depends only on the clauses F consists of. So the process of constructing an SSP can be viewed as a "natural"”way of traversing search space when checking F for satisfiability. For instance, if F has symmetries, they can be easily taken into account when constructing an SSP. To illustrate this point, we consider the class of CNF formulas that are symmetric with respect to a group of permutations. We show that in this case for proving the unsatisfiability of a CNF formula it is sufficient to construct a set of points that is stable modulo symmetry. In particular, as it is shown in the paper for pigeon−hole CNF formulas there is a linear size set of points that is stable modulo symmetry. So the unsatisfiability of pigeon−hole CNF formulas can be proven by examining only a set of points of linear size.

The notion of an SSP is the development of the idea of 1−neighborhood exploration introduced in [5]. From the practical point of view the notion of an SSP (and, more generally, the notion of 1−neighborhood exploration) is important because it gives a new criterion for algorithm termination. Namely, once it is proven that the examined part of the Boolean space is an SSP (or contains an SSP) one can claim that the CNF under test is unsatisfiable.

The rest of the paper is organized as follows. In Section 2 we introduce the notion of an SSP. In Section 3 we show the relationship between SSPs and sets of reachable points. In Section 4 we describe a simple algorithm for constructing an SSP. In Section 5 we show that our algorithm for constructing SSPs can be easily modified to take into account formula's symmetry. In Section 6 we apply the modified algorithm to a class of highly symmetric formulas called pigeon−hole CNF formulas. In Section 7 we give a summary of results and directions for future research. In Appendix A we describe background on testing the satisfiability of symmetric CNF formulas. In appendix B proofs of the propositions are given. In appendix C some experimental results are shown.

2. Stable set of points

In this section, we introduce the notion of an SSP. Let F be a CNF formula of n variables $x_1,...,x_n$. Denote by B the set {0,1} of values taken by a Boolean variable. Denote by B^n the set of points the Boolean space specified by variables $x_1,...,x_n$. A point of B^n is an assignment of values to all the n variables.

Definition 1. A disjunction of literals (also called a clause) C is called ***satisfied*** by a value assignment (point) p if $C(p)$=1. Otherwise, clause C is called ***falsified*** by p.

Definition 2. Let F be a CNF formula. The ***satisfiability problem*** is to find a value assignment (point) satisfying all the clauses of F. This assignment is called a ***satisfying assignment***.

Definition 3. Let p be a point of the Boolean space falsifying a clause C. The ***1−neighborhood of point*** p with respect to clause C (written $Nbhd(p,C)$) is the set of points that are at Hamming distance 1 from p and that satisfy C.

Example 1. Let $C=x_1 \vee \neg x_3 \vee x_6$ be a clause specified in the Boolean space of 6 variables $x_1,\ldots,x_6$. (Symbol '¬' means negation.) Let $p=(x_1=0,x_2=1,x_3=1,x_4=0,x_5=1,x_6=0)$ be a point falsifying C. Then $Nbhd(p,C)$ consists of the following three points: $p_1=(\boldsymbol{x_1=1},x_2=1,x_3=1,x_4=0,x_5=1,x_6=0)$, $p_2=(x_1=0,x_2=1,\boldsymbol{x_3=0},x_4=0,x_5=1,x_6=0)$, $p_3=(x_1=0,x_2=1, x_3=1,x_4=0,x_5=1,\boldsymbol{x_6=1})$. Points p_1,p_2,p_3 are obtained from p by flipping the value of variables x_1,x_3,x_6 respectively i.e. the variables whose literals are in C.

Denote by $Z(F)$ the set of points at which F = 0. If F is unsatisfiable, $Z(F) = B^n$.

Definition 4. Let F be a CNF formula and P be a subset of $Z(F)$. Mapping g of P to F is called a ***transport function*** if, for any $p \in P$, clause $g(p) \in F$ is falsified by p. In other words, a transport function $g{:}P \rightarrow F$ is meant to assign each point $p \in P$ a clause that is falsified by p. We call mapping $P \rightarrow F$ a transport function because, as it is shown in section 3, such a mapping allows one to introduce some kind of "movement" of points in the Boolean space.

Definition 5. Let P be a nonempty subset of $Z(F)$ and F be a CNF formula. Set P is called ***stable*** with respect to a CNF formula F and transport function g: $P \rightarrow F$, if

$\forall\, p \in$ P, $Nbhd(p,g(p)) \subseteq P$.

Remark. Henceforth, if we say that a set of points P is stable with respect to a CNF formula F without mentioning a transport function, we mean that there is a mapping g:$P \rightarrow F$ such that P is stable with respect to F and g.

Example 2. Consider an unsatisfiable CNF formula F consisting of 7 clauses: $C_1 = x_1 \vee x_2$, $C_2 = \neg\, x_2 \vee x_3$, $C_3 = \neg\, x_3 \vee x_4$, $C_4 = \neg\, x_4 \vee x_1$, $C_5 = \neg\, x_1 \vee x_5$, $C_6 = \neg\, x_5 \vee x_6$, $C_7 = \neg\, x_6 \vee \neg\, x_1$. Clauses of F are composed of literals of 6 variables: $x_1,\dots,x_6$. The following 14 points form an SSP P: p_1=000000, p_2=010000, p_3=011000, p_4=011100, p_5=111100, p_6=111110, p_7=111111, p_8=011111, p_9=011011, p_{10}=010011, p_{11}=000011, p_{12}=100011, p_{13}=100010, p_{14}=100000. (Values of variables are given in the order variables are numbered. For example, p_4 consists of assignments x_1=0, x_2=1, x_3=1, x_4=1, x_5=0, x_6=0.) Set P is stable with respect to the transport function g specified as: g(p_1)=C_1, g(p_2)=C_2, g(p_3)=C_3, g(p_4)=C_4, g(p_5)=C_5, g(p_6)=C_6, g(p_7)=C_7, g(p_8)=C_4, g(p_9)=C_3, g(p_{10})=C_2, g(p_{11})=C_1, g(p_{12})=C_7, g(p_{13})=C_6, g(p_{14})=C_5. It is not hard to see that g indeed is a transport function i.e. for any point p_i of P it is true that $C(p_i)$=0 where C=$g(p_i)$. Besides, every point p_i of P satisfies the condition $Nbhd(p,g(p)) \subseteq P$ of Definition 5. Consider, for example, point p_{10}=010011. The value of $g(p_{10})$ is C_2, $C_2 = \neg\, x_2 \vee x_3$ and the value of $Nbhd(p_{10},C_2)$ is {p_{11}=000011, p_9=011011}, the latter being a subset of P.

Proposition 1. If there is a set of points that is stable with respect to a CNF formula F, then F is unsatisfiable.

Proofs of all the propositions are given in Appendix B.

Proposition 2. Let F be an unsatisfiable CNF formula of n variables. Then set $Z(F)$ is stable with respect to F and any transport function $Z(F) \rightarrow F$.

Remark. From Proposition 1 and Proposition 2 it follows that a CNF formula F is unsatisfiable if and only if there is a set of points stable with respect to F.

3. SSPs as sets of reachable points

In this section, we show the relationship between SSPs and sets of reachable points.

Definition 6. Let F be a CNF formula and g: $Z(F) \rightarrow F$ be a transport function. A sequence of points $p_1,\dots,p_k$ is called a ***path*** from point p_1 to point p_k in set P with transport function g, if points $p_1,\dots,p_{k-1}$ are in P and $p_i \in Nbhd(p_{i-1},g(p_{i-1}))$, $2 \le i \le k$. (Note that the last point of the path, i.e. p_k, does not have to be in P.) We will assume that no point repeats twice (or more) in a path.

Example 3. Consider the CNF formula and transport function of Example 2. Let P be the set of points specified in Example 2. Sequence of points p_1,p_{14},p_{13},p_{12} form a path from p_1 to p_{12}. Indeed, it is not hard to check that $Nbhd(p_1,g(p_1))=\{p_2,p_{14}\}$, $Nbhd(p_{14},g(p_{14}))=\{p_{13},p_1\}$, $Nbhd(p_{13},g(p_{13}))=\{p_{14},p_{12}\}$, $Nbhd(p_{12},g(p_{12}))=\{p_{13},p_{11}\}$. So each point p' of the path (except the starting point i.e. p_1) is contained in the $Nbhd(p'',g(p''))$ where p'' is the preceding point.

Definition 7. Let F be a CNF formula. Point p'' is called ***reachable*** from point p' *by* means of transport function g:$Z(F) \rightarrow F$ if there is a path from p' to p'''with transport function g. Denote by *Reachable*(p,g) the set that consists of point p and all the points that are reachable from p by means of transport function g.

Proposition 3. Let F be a satisfiable CNF formula, p be a point of $Z(F)$, and s be the closest to p (in Hamming distance) satisfying assignment. Let $g{:}Z(F)\rightarrow F$ be a transport function. Then in $Z(F)$ there is a path from p to s with transport function g i.e. solution s is reachable from p.

Proposition 4. Let F be a CNF formula, $g{:}Z(F)\rightarrow F$ be a transport function, and p be a point from $Z(F)$. If P=*Reachable*(p,g) does not contain a satisfying assignment, then P is stable with respect to F and g and so F is unsatisfiable.

4. Testing satisfiability of CNF formulas by SSP construction

In this section, we describe a simple algorithm for constructing an SSP. It is based on Proposition 3 and Proposition 4. Let F be a CNF formula to be checked for satisfiability. The idea is to pick a point p of the Boolean space and construct set *Reachable*(p,g). According to Proposition 4 if *Reachable*(p,g) does not contain a solution, it is stable and so F is unsatisfiable. On the other hand, if F is satisfiable, then according to Proposition 3 set *Reachable*(p,g) must contain a solution. Since transport function $g{:}Z(F)\rightarrow F$ is not known beforehand, it is built on the fly. In the description of the algorithm given below, set *Reachable*(p,g) is broken down into two parts: *Boundary* and *Body*. The *Boundary* consists of those points of the current set *Reachable*(p,g) whose 1−neighborhood has not been explored yet. At each step of the algorithm a point p' of the *Boundary* is extracted and a clause C falsified by p' is assigned as the value of $g(p')$. Then the set *Nbhd*(p',C) is generated and its points (minus those that are already in the *Body* or *Boundary*) are added to the *Boundary*. This goes on until the *Boundary* is empty (F is unsatisfiable) or a satisfying assignment is found (F is satisfiable).

Generate a starting point p. *Boundary*=$\{p\}$. *Body*=$\emptyset$, g=$\emptyset$.

1. If the *Boundary* is empty, then the *Body* is an SSP and F is unsatisfiable. The algorithm terminates.
2. Pick a point $p' \in$ *Boundary*. *Boundary*=*Boundary* \ $\{p'\}$.
3. Find a set M of clauses that are falsified by point p'. If M=$\emptyset$, then CNF formula F is satisfiable and p' is a satisfying assignment The algorithm terminates.
4. Pick a clause Ñ from Ì. Take C as the value of $g(p')$. Generate *Nbhd*(p',C). *Boundary*=*Boundary* $\cup$ (*Nbhd*(p',C) \ *Body*). *Body* = *Body* $\cup$ $\{p'\}$.
5. Go to step 2.

Interestingly, the described algorithm can be viewed as an extension of Papadimitriou's algorithm [10] and Walksat [12] to the case of unsatisfiable CNF formulas. Papadimitriou's algorithm and Walksat can be applied only to satisfiable CNF formulas since they do not store visited points of the Boolean space. The remarkable fact is that the number of points that one has to explore to prove the unsatisfiability of a CNF formula can be very small. For instance, in Example 2 an SSP of a CNF formula of 6 variables consists only of 14 points while the Boolean space of 6 variables consists of 64 points. In is not hard to show that for a subclass of the class of 2−CNF formulas (a clause of a 2−CNF formula contains at most 2 literals) there is always an SSP of linear size. This subclass consists of formulas analogous to the one described in Example 2.

However, we have not proved (or disproved) this claim for the whole class of 2−CNF formulas yet.

From the practical point of view the described algorithm has two substantial flaws. First, an SSP is constructed point−by−point while computing an SSP in larger chunks of points (clustering "similar" points of the Boolean space) should be more efficient. Second, the algorithm looks for a set of points that is stable with respect to the initial set of clauses. On the other hand, if an algorithm is allowed to resolve clauses of the initial CNF, it may find a much smaller set of points that is stable with respect to a set of resolvents. Nevertheless, for symmetric CNF formulas (considered in Sections 5 and 6) even point−by−point SSP computation can be efficient.

5. Testing satisfiability of symmetric CNF formulas by SSP construction

In this section we show how the algorithm of Section 4 should be modified to take into account formula's symmetry.

Definition 8. Let $X=\{x_1,....,x_n\}$ be a set of Boolean variables. ***Permutation*** π defined on set X is a bijective mapping of X onto itself.

Let $F=\{C_1,...,C_k\}$ be a CNF formula. Let $p=(x_1,...,x_n)$ be a point of B_n. Denote by $\pi(p)$ point $(\pi(x_1),...,\pi(x_n))$. Denote by $\pi(C_i)$ the clause that is obtained from $C_i \in F$ by replacing variables $x_1,..,x_n$ with variables $\pi(x_1),...,\pi(x_n)$ respectively. Denote by $\pi(F)$ the CNF formula obtained from F by replacing each clause C_i with $\pi(C_i)$.

Definition 9. CNF formula F is called ***symmetric*** with respect to permutation π if each clause $\pi(C_i)$ of $\pi(F)$ is identical to a clause $C_k \in F$.

The set of the permutations, with respect to which a CNF formula is symmetric, forms a group. Henceforth, we will denote this group by G. The fact that a permutation π is an element of G will be denoted by $\pi \in G$. Denote by 1 the identity element of G.

Definition 10. Points p and p' are called ***symmetric*** if there is a permutation $\pi \in G$ such that $p'=\pi(p)$ or $p=\pi(p')$.

Remark. The binary relation introduced by Definition 10 is an equivalence relation. This relation partitions the Boolean space B^n into equivalence classes. Each class consists of a set of pairwise symmetric points. It is not hard to show that if G is a group of permutation of a CNF formula F, then for any pair p,p' of symmetric points $F(p)=F(p')$.

Definition 11. Let F be a CNF formula and P be a subset of $Z(F)$. Set P is called ***stable modulo symmetry*** with respect to F and transport function $g: P \rightarrow F$ if for each $p \in P$ every point $p' \in Nbhd(p,g(p))$ is either in P or there is a point p'' of P that is symmetric to p'.

Proposition 5. Let B^n be the Boolean space specified by variables $X=\{x_1,....,x_n\}$. Let p be a point of B^n, C be a clause falsified by p, and $q \in Nbhd(p,C)$ be obtained from p by flipping the value of variable x_i. Let π be a permutation of variables from X, p' be equal to $\pi(p)$, C' be equal to $\pi(C)$, and $q' \in Nbhd(p',C')$ be obtained from p' by flipping the value of variable $\pi(x_i)$. Then $q'=\pi(q)$. In other words, for each point q of $Nbhd(p,C)$ there is a point q' of $Nbhd(p',C')$ such that $q'=\pi(q)$.

Proposition 6. Let F be a CNF formula, P be a subset of $Z(F)$, and $g{:}P \rightarrow F$ be a transport function. If P is stable modulo symmetry with respect to F and g, then CNF formula F is unsatisfiable.

Proposition 7. Let $P \subseteq B^n$ be a set of points that is stable with respect to a CNF formula F and transport function g: $P \rightarrow F$. Let P' be a subset of P such that for each point p of P that is not in P' there is a point $p' \in P'$ symmetric to p. Then P' is stable with respect to F and g modulo symmetry.

Definition 12. Let F be a CNF formula, G be its group of permutations, p be a point of $Z(F)$, and g: $P \rightarrow F$ be a transport function. Set $Reachable(p,g,G)$ is called the set of points ***reachable from p modulo symmetry*** if a) it includes point p; b) each point p' that is reachable from p by means of transport function g is either in $Reachable(p,g,G)$ or there exists point $p'' \in Reachable(p,g,G)$ that is symmetric to p'.

Proposition 8. Let F be a CNF formula, G be its group of permutations, p be a point of $Z(F)$, and g: $P \rightarrow F$ be a transport function. If set $P=Reachable(p,g,G)$ does not contain a satisfying assignment, then it is stable modulo symmetry with respect to F and g.

Proposition 9. Let F be a CNF formula, G be its group of permutations, $g{:}Z(F) \rightarrow F$ be a transport function, and p be a point of $Z(F)$. CNF formula F is satisfiable if and only if $Reachable(p,g,G)$ contains a satisfying assignment.

Let F be a CNF formula and G be its group of permutations. According to Proposition 9 when testing the satisfiability of F it is sufficient to construct set $Reachable(p,g,G)$. This set can be built by the algorithm of Section 4 in which step 5 is modified in the following way. Before adding a point p''"from $Nbhd(p',C) \setminus Body$ to the *Boundary* it is checked if there is a point of $Boundary \cup Body$ that is symmetric to p''. If such a point exists, then p'' is not added to the *Boundary*.

6. Computing SSPs of pigeon−hole CNF formulas

In this section, we apply the theory of Section 5 to a class of symmetric CNF formulas called pigeon−hole formulas. Pigeon−hole CNF formulas, by means of propositional logic, describe the fact that n objects (pigeons) cannot be placed in m holes so that no two objects occupy the same hole if $n > m$. Pigeon−hole formulas was the first class of CNF formulas for which resolution was proven to be exponential [6].

Definition 13. Denote by $ph(i,k)$ the Boolean variable whose value indicates if i−th pigeon is in k−th hole ($ph(i,k)$=1 means that the pigeon is in the hole). ***Pigeon−hole CNF formula*** (written $PH(n,m)$) consists of the following two sets of clauses (denote them by $H_1(n,m)$ and $H_2(n,m)$). Set $H_1(n,m)$ consists of n clauses $ph(i,1) \vee ph(i,2) \vee \ldots \vee ph(i,m)$, i=1,...,n, i−th clause encoding the fact that i−th pigeon has to be in at least one hole. Set $H_2(n,m)$ consists of $m*n*(n-1)/2$ clauses $\neg ph(i,k) \vee \neg ph(j,k)$, $i < j$, $1 \leq i,j \leq n$, $1 < k < m$. Clause $\neg ph(i,k) \vee \neg ph(j,k)$ encodes the fact that i−th and j−th pigeons $i \neq j$ cannot be placed in the k−th hole together.

CNF formula $PH(n,m)$ has $n*m$ variables. To "visualize"”points of the Boolean space B^{n*m} we will assume that the variables of $PH(n,m)$ are represented by entries of a matrix M of n rows and m columns. Entry $M(i,j)$ of the matrix corresponds to variable $ph(i,j)$. Then each point of the Boolean space can be viewed as a matrix $n \times m$ whose entries take values 0 or 1. Denote by $M(p)$ the matrix representation of point p. Denote by $S(n,m)$ the following set of points of the Boolean space. $S(n,m)$ consists of two subsets of points denoted $S_1(n,m)$ and $S_2(n,m)$. A point p is included in subset $S_1(n,m)$ if and only if each row and column of $M(p)$ contains at most one 1–entry. A point p is included in subset $S_2(n,m)$ if and only if a) matrix $M(p)$ has exactly one column containing two 1–entries and the rest of the columns have at most one 1–entry; b) $M(p)$ contains at most 1–entry per row.

It is not hard to see that for a point p from $S_1(n,m)$ there is a clause of $H_1(n,m)$ that p does not satisfy. The latter is true because, since $n > m$ and every column has at most one 1–entry, there is at least one row (say i–th row) of $M(p)$ consisting only of 0–entries. Then p does not satisfy clause $ph(i,1) \vee ph(i,2) \vee \ldots \vee ph(i,m)$ of $H_1(n,m)$. For each point p from $S_2(n,m)$ there is exactly one clause from $H_2(n,m)$ that p does not satisfy (and maybe some clauses of $H_1(n,m)$). Suppose for example, that in $M(p)$ entries $M(i,k)$ and $M(j,k)$ are equal to 1 (i.e. k–th column is the one containing two 1–entries). Then the only clause of $H_2(n,m)$ point p does not satisfy is $\neg ph(i,k) \vee \neg ph(j,k)$.

Definition 14. Denote by g the following transport function mapping $S(n,m)$ to $PH(n,m)$. If $p \in S_1(n,m)$ then $g(p)$ is equal to a clause from $H_1(n,m)$ that p does not satisfy (no matter which). If $p \in S_2(n,m)$ then $g(p)$ is equal to the clause from $H_2(n,m)$ that p does not satisfy.

Proposition 10. Set of points $S(n,m) = S_1(n,m) \cup S_2(n,m)$ is stable with respect to the set of clauses $H_1(n,m) \cup H_2(n,m)$ and transport function g specified above.

Proposition 11. Let p be the point in which every variable is assigned value 0. Let g: $B^{n*m} \rightarrow PH(n,m)$ be a transport function. Then set $Reachable(p,g)$ constructed by the algorithm described in Section 4 is a subset of $S(n,m)$ if the following heuristic is used when constructing an SSP. If a new point p to be added to the *Boundary* falsifies clauses from both $H_1(n,m)$ and $H_2(n,m)$, then a clause of $H_2(n,m)$ is selected as the value of $g(p)$.

The group of permutations of CNF formula $PH(n,m)$ (denote it by $G(PH(n,m))$) is the direct product of the group of all the permutations of n pigeons and the group of all the permutations of m holes.

Definition 15. Let p be a point of the Boolean space B^{n*m} in which $PH(n,m)$ is specified. Vector $(c_1,\ldots,c_m)$ where c_j , $1 \leq j \leq m$ is the number of 1–entries in the j–th column of $M(p)$, is called the ***signature*** of p. Signature v' of p' and v'' of p'' are said to be ***identical modulo permutation*** if v' can be transformed to v'' by a permutation.

Proposition 12. Let p and p' be points of B^{n*m} such that their signatures are not identical modulo permutation. Then there is no permutation $\pi \in G(PH(n,m)))$ such that $p' = \pi(p)$ i.e. points p and p' are not symmetric.

Proposition 13. Let p and p' be points of $S(n,m)$ such that their signatures are identical modulo symmetry. Then points p and p' are symmetric

Remark. From Proposition 12 and Proposition 13 it follows that points p and p' of $S(n,m)$ are symmetric with respect to $G(PH(n,m))$ iff the signature v' of p' and signature v'' of p'' are identical modulo permutation.

Proposition 14. Set $S(n,m)$ contains $2*m+1$ equivalence classes.

Proposition 15. There is a set of points that is stable with respect to $PH(n,m)$ and transport function g (specified by Definition 14) modulo symmetry, and that consists of $2*m+1$ points.

Proposition 16. Let $p \in S_1(n,m)$ be the point in which all variables are assigned 0. Let $Reachable(p,g,G(PH(n,m)))$ be the SSP built by the algorithm described in the end of Section 5 where the construction of the transport function is guided by the heuristic described in Proposition 11. Then set $Reachable(p,g,G(PH(n,m)))$ contains no more than $2*m+1$ points. The time taken by the algorithm for constructing such a set is $O(m^2*f)$ where f is the complexity of checking if two points of $S(n,m)$ are symmetric.

7. Conclusions

We show that satisfiability testing of a CNF formula reduces to constructing a stable set of points (SSP). An SSP of a CNF formula can be viewed as an inherent characteristic of this formula. We give a simple procedure for constructing an SSP. As a practical application we show that the proposed procedure of SSP construction can be easily modified to take into account symmetry (with respect to variable permutation) of CNF formulas. In particular, we consider a class of symmetric CNF formulas called pigeon−hole formulas. We show that the proposed algorithm can prove their unsatisfiability in quadratic time and there is a stable (modulo symmetry) set of points of linear size.

An interesting direction for future research is to relate the size of SSPs of a CNF formula to the complexity of proving its unsatisfiability. In particular, it is important to identify classes of CNF formulas having SSPs of polynomial size. A natural candidate is the class of 2−CNF formulas. On the practical side, it is important to develop methods that a) are able to construct an SSP in “chunks” clustering points that are “similar”; b) can use resolution to reduce the size of SSPs by producing “better” sets of clauses.

References

1. C.A.Brown,L.Finkelstein,P.W.Purdom. *Backtrack searching in the presence of symmetry*. In “Applied algebra, algebraic algorithms and error correcting codes”. Sixth international conference, P. 99−110. Springer−Verlag,1988.
2. V. Chvatal, E. Szmeredi. *Many hard examples for resolution.* J. of the ACM,vol. 35, No 4, pp.759−568.
3. M. Crawford, M. Ginsberg, E. Luks, A. Roy. *Symmetry breaking predicates for search problems*. Fifth International Conference on Principles of Knowledge Representation and Reasoning (KR’96).
4. M.Davis, G.Logemann, D.Loveland. *A Machine program for theorem proving*. Communications of the ACM. −1962. −V.5. −P.394−397.
5. E.Goldberg. *Proving unsatisfiability of CNFs locally*. Proceedings of LICS 2001 Workshop on Theory and Applications of Satisfiability Testing.
6. A.Haken. *The intractability of resolution*. Theor. Comput. Sci. 39 (1985),297−308.
7. B.Krishnamurthy. *Short proofs for tricky formulas.* Acta Informatica 22 (1985) 253−275.
8. D.Mitchell, B.Selman ,and H.J.Levesque. *Hard and easy distributions of SAT problems.* Proceedings AAAI−92, San Jose,CA, 459−465.
9. M.Moskewicz,C.Madigan,Y.Zhao,L.Zhang,S.Malik. *Chaff: Engineering an Efficient*

SAT Solver. Proceedings of DAC−2001.
10. C.Papadimitriou. *On selecting a satisfying truth assignment*. Proceedings of FOC−91.
11. A.Roy. *Symmetry breaking and fault tolerance in Boolean satisfiability*. PhD thesis. Downloadable from http://www.cs.uoregon.edu/~aroy/
12. B.Selman,H.Kautz,B.Cohen. *Noise strategies for improving local search*. Proceedings of AAAI−94.
13. A.Urquhart. *The symmetry rule in propositional logic*. Discrete Applied Mathematics 96−97(1999):177−193,1999.
14. H.Wong−Toi. *Private communication*.

Appendix A . Some background on testing satisfiability of symmetric CNF formulas

In this appendix, we give some background on testing the satisfiability of symmetric CNF formulas. Methods for simplifying satisfiability check for symmetric formulas have received substantial attention in the past. In [7] it was shown that if the resolution system is enhanced by a "symmetry rule" then the complexity of proofs for some classes of formulas reduces from exponential to polynomial. This extra rule allows to "shortcut" the deduction of implicates that are symmetric to ones deduced before. In [13] it was shown that in the resolution system with the symmetry rule, the satisfiability of pigeon−hole formulas can be refuted with a proof of length $(3n+1)n/2$ where n is the number of holes. Unfortunately, the resolution system (whether it has the symmetry rule or not) is non−deterministic and so these results are not very helpful in designing deterministic algorithms.

Practical (and hence deterministic) algorithms for testing satisfiability of symmetric formulas were considered in [1,3,11]. In [1] a backtracking algorithm with some machinery for prunning symmetric branches was introduced. The problem of such an approach is that the ability to prune symmetric branches is obtained at the expense of losing the freedom of search tree examination. So if a new scheme of backtracking is found in the future, a new algorithm would have to be designed to take into account symmetries of the CNF under test.

To solve the problem, in [3] it was suggested to add to the CNF formula F to be tested for satisfiability a set G of "symmetry breaking" clauses . The idea is to find such a set G of clauses that only one point of each symmetry class satisfies all the clauses of G. This way search in symmetric portions of the Boolean space is pruned earlier than without adding clauses of G (if a clause of G is falsified before any clause of F). The generation of symmetry−breaking clauses G is done by a separate procedure performed before actual satisfiability testing. So this procedure (used as a preprocessor) can be run in combination with any SAT−solver to be developed in the future.

One of the flaws of the approach is that the problem of generating a full set of symmetry breaking clauses is NP−hard [3]. Moreover, for some groups the number of all clauses that have to be generated to break all symmetris of the group is exponential [11]. This leads to the following problem. Since often one cannot break all the symmetries, it is reasonable to try to break only symmetries whose elimination would symplify satisfiability testing the most. (For example, if at the satisfiability testing step a search tree is explored, we would like symmetry breaking clauses to help prune largest subtrees of the search tree.) However, since symmetry processing and satisfiability testing are performed separately, at the symmetry processing step we do not know which symmetries should be broken. (In other words, when generating symmetry breaking clauses we do not have information about which subtrees of the future search tree are going to be large.)

This suggests that even though incorporating symmetry processing into the current backtracking algorithms is difficult, satisfiability testing and symmetry processing should be tightly linked. One more reason for such a conclusion is that non−symmetric (or having little symmetry) "real−life" CNF formulas may have highly symmetric subformulas (i.e. CNF formulas obtained from the initial CNF after making a number of value assignments). Such a kind of symmetry can be used only in the process of satisfiability testing. So, instead of separating symmetry processing and satisfiability testing steps it makes sense to try to find a search space traversal scheme that is more amenable to symmetry processing than backtracking. We believe that bulding an SSP could be such a scheme. The point is that an SSP of a CNF formula F is an inherent characteristic of F. So if F has some symmetries, an SSP has these symmetries as well, which makes it easy to use them during satisfiability testing.

Appendix B: Proofs of the propositions

Proof of Proposition 1**.** Assume the contrary. Let P be a set of points that is stable with respect to F and a transport function g, and p^* be a satisfying assignment i.e. $F(p^*) = 1$. It is not hard to see that $p^* \notin P$ because each point $p \in P$ is assigned a clause $C=g(p)$ such that $C(p)=0$ and so $F(p)=0$. Let p be a point of P that is the closest to p^* in Hamming distance. Denote by C the clause that is assigned to p by transport function g i.e. $C=g(p)$. Denote by Y the set of variables values of which are different in p and p^*.

Let us show that C cannot have literals of variables of Y. Assume the contrary, i.e. that C contains a literal of $x \in Y$. Then, since P is stable with respect to F and g, it has to contain the point p' which is obtained from p by flipping the value of x. But then p'' $\in P$ is closer to p^* than p. So we have a contradiction. Since $C(p)=0$ and C does not contain literals of variables whose value are different in p and p^* we have to conclude that $C(p^*)=0$. This means that p^* is not a solution and so we have a contradiction.

Proof of Proposition 2**.** Since F is unsatisfiable, then $Z(F)=B^n$. Then, given a transport function $g{:}Z(F) \rightarrow F$, condition $Nbhd(p,g(p)) \subseteq B^n$ holds for every point $p \in B^n$.

Proof of Proposition 3**.** Denote by Y the set of variables whose values are different in p and s. Since $F(p)=0$, then $p \in Z(F)$ and function g assigns a clause C to p where $C(p)=0$. All literals of C are set to 0 by p. On the other hand, since s is a solution then at least one literal of C is set by s to 1. Then C has at least one literal of a variable from Y. Flipping the value of this variable of Y in p we obtain a point p' which is closer to point s by 1 (in Hamming distance). Point p' is reachable from p by means of transport function g. If $|Y|>1$, then p' cannot be a satisfying assignment since, by our assumption, s is the closest to p satisfying assignment. Going on in this manner we reach satisfying assignment s in $|Y|$ steps.

Proof of Proposition 4**.** Assume the contrary, i.e. that $Reachable(p,g)$ is not stable. Then there exists a point p' of *Reachable*(p,g) (and so reachable from p) such that a point p'' of *Nbhd*(p',g(p')) is not in *Reachable*(p,g). Since p'' is reachable from p' it is also reachable from p. We have a contradiction.

Proof of Proposition 5**.** The value of variable x_k, $k \neq i$ in q is the same as in p. Besides, the value of variable $\pi(x_k)$ in q' is the same as in p' (q' is obtained from p' by changing the value of variable $\pi(x_i)$ and since $k \neq i$ then $\pi(x_k) \neq \pi(x_i)$). Since $p'=\pi(p)$ then the value of x_k in q is the same as the value of variable $\pi(x_k)$ in q'. On the other hand, the value of variable x_i in q is obtained by negation of the value of x_i in p. The value of variable $\pi(x_i)$ in q' is obtained by negation of the value of $\pi(x_i)$ in p'. Hence the values of variable x_i in q and variable $\pi(x_i)$ in q' are the same. So $q'=\pi(q)$.

Proof of Proposition 6**.** The idea of the proof was suggested to the author by Howard Wong–Toi [14].

Denote by $K(p)$ the set of all points that are symmetric to point p i.e. that are in the same equivalence class. Denote by $K(P)$ the union of the sets $K(p)$, $p \in P$. Extend the domain of transport function g from P to $K(P)$ in the following way. Suppose p' is a point that is in $K(P)$ but not in P. Hence there is a point $p \in P$ that is symmetric to p'' and so $p'=\pi(p)$, $\pi \in G$. Then we assign C''$=\pi(C)$, $C=g(p)$ as the value of g at p'. If there is more than one point of P that is symmetric to p', we pick any of them.

To prove that F is unsatisfiable we show that $K(P)$ is stable with respect to F and g: $K(P) \rightarrow F$. Let p' be a point of $K(P)$. The idea is to show that any point q' from *Nbhd*(p',g(p')) is symmetric to a point of P and so $q' \in K(P)$. This would mean that *Nbhd*(p',g(p')) $\subseteq K(P)$ and so $K(P)$ is stable.

According to the definition of $K(P)$ there is a point p of P that is symmetric to p' and so $p'=\pi(p)$. Then from Proposition 5 it follows that for any point q of *Nbhd*(p,g(p)) there is a point $q' \in$ *Nbhd*(p',g(p')) such that $q'=\pi(q)$. On the other hand, since P is stable modulo symmetry then for any point q of *Nbhd*(p,g(p)) there is a point $q'' \in P$ symmetric to q and so $q=\pi^*(q'')$, $\pi^* \in G$ (π^* may be equal to $1 \in G$ if q is in P). Then $q'' = \pi(\pi^*(q''))$. Hence q' is symmetric to $q'' \in P$ and so $q' \in K(P)$.

Proof of Proposition 7**.** Let p' be a point of P'. Let q' be a point of *Nbhd*(p',g(p')). Point p' is in P because $P' \subseteq P$. Since P is a stable set then $q' \in P$. From the definition of set P' it follows that if q' is not in P' then there is a point $r' \in P'$ that is symmetric to q''. So each point q' of *Nbhd*(p',g(p')) is either in P' or there is a point of P' that is symmetric to q'.

Proof of Proposition 8**.** Assume the contrary, i.e. that P is not stable. Then there is a point $p' \in P$ (reachable from p modulo symmetry) such that a point p'' of *Nbhd*(p',g(p')) is not in P and P does not contain a point symmetric to p''. On the other hand, p'' is reachable from p' and so it is reachable from p modulo symmetry. We have a contradiction.

Proof of Proposition 9***.*** ***If part.*** If *Reachable*(p,g,G) contains a satisfying assignment then F is obviously satisfiable.
Only if part. Assume the contrary i.e. that F is satisfiable and *Reachable*(p,g,G) does not contain a satisfying assignment. From Proposition 8 it follows that *Reachable*(p,g,G) is stable modulo symmetry with respect to F and g. Then from Proposition 6 it follows that F is unsatisfiable. We have a contradiction.

Proof of Proposition 10***.*** Let p be a point from $S(n,m)$. Consider the following two alternatives.
1) $p \in S_1(n,m)$. Then the matrix representation $M(p)$ of p has at least one row (say i–th row) consisting only of 0–entries. Point p falsifies at least one clause from $H_1(n,m)$. A falsified clause of $H_1(n,m)$ (say, clause $C= ph(i,1) \vee ph(i,2) \vee \ldots \vee ph(i,m)$) is assigned to p by transport function g. Let us show that $Nbhd(p,C) \subseteq S_1(n,m) \cup S_2(n,m)$. Denote by p' the point obtained from p by flipping the value of variable $ph(i, j)$. By definition, no column of $M(p)$ contains more than one 1–entry. So we have two alternatives. First, if j–th column of $M(p)$ contains a 1–entry then the matrix representation $M(p')$ of p' contains exactly one column (namely, j–th column) that contains two 1–entries. Besides, all rows of $M(p')$ still contain at most one 1–entry. (We have added a 1–entry to the i–th row that did not contain any 1–entries in $M(p)$.) Then $p' \in S_2(n,m)$. Second, if j–th column of $M(p)$ does not contain a 1–entry, then $M(p')$ does not contain columns having two 1–entries and so $p' \in S_1(n,m)$.
2) $p \in S_2(n,m)$. Then the matrix representation $M(p)$ of p has exactly one column (say, j–th column) that has two 1–entries. Let us assume that j–th column $M(p)$ has 1–entries in i–th and k–th rows. Point p falsifies exactly one clause of $H_2(n,m)$, namely, clause $C=\neg ph(i, j) \vee \neg ph(k, j)$. This is the clause that is assigned to p by transport function g. Set $Nbhd(p,C)$ consists of two points obtained from p by flipping the value of $ph(i, j)$ or $ph(k, j)$. Let p' be either point of $Nbhd(p,C)$. Matrix $M(p')$ does not have columns containing two 1–entries (because one 1–entry of j–th column has disappeared). Besides, $M(p')$ has at most one 1–entry per row. Then $p' \in S_1(n,m)$. Hence $Nbhd(p,C) \subseteq S_1(n,m)$.

So in both cases $Nbhd(p,C) \subseteq S_1(n,m) \cup S_2(n,m)$.

Proof of Proposition 11***.*** We prove the proposition by induction. Denote by *Boundary*(s) and *Body*(s) the *Boundary* and *Body* sets constructed after performing s steps of the algorithm. Denote by g_s the transport function after performing s steps. Our induction hypothesis is that after performing s steps of the algorithm set *Boundary*(s) $\cup$ *Body*(s) is a subset of $S(n,m)$ and g_s satisfies Definition 14 (at s points wherein the function g_s is specified). First we need to check that the hypothesis holds for s=1. The starting point p is in $S_1(n,m)$. Besides, p falsifies only clauses from $H_1(n,m)$. So if we assign a clause C of $H_1(n,m)$ as the value of g_1 at point p, then function g_1 satisfies Definition 14 .

Now we prove that from the fact the hypothesis holds after performing s steps of the algorithm, it follows that it also holds after s+1 steps of the algorithm. Let p' be the point of *Boundary*(s) chosen at step s+1. First, let us show that transport function g_{s+1} satisfies Definition 14. If p' is in $S_1(n,m)$ then it falsifies only clauses from $H_1(n,m)$. So no matter which falsified clause is picked as the value of transport function g_{s+1} at point p', g_{s+1} satisfies Definition 14. If p' is in $S_2(n,m)$ then it falsifies exactly one clause of $H_2(n,m)$ and maybe some clauses of $H_1(n,m)$. Our heuristic makes us select the falsified clause of $H_2(n,m)$ as the value of g at point p'. So again transport function g_{s+1} satisfies Definition 14. Then we can apply arguments of Proposition 10 to show that from $p' \in S(n,m)$ it follows that $Nbhd(p',g_{s+1}(p'))$ is a subset of $S(n,m)$. Hence $Boundary(s+1) \cup Body(s+1)$ is a subset of $S(n,m)$.

Proof of Proposition 12**.** Since signatures of p and p' are not identical modulo permutation then there is value $k \le n$ such that $M(p)$ and $M(p')$ have different number of columns containing k 1–entries. On the other hand, no permutation $\pi \in G(PH(n,m))$ can change the number of columns having k 1–entries. Indeed, let $M(p^*)$ be the matrix representation of a point p^* of B^{n*m}. Each permutation of $G(PH(n,m))$ consists of a permutation of m columns of $M(p^*)$ (i.e. holes) and n rows of $M(p^*)$ (i.e. pigeons). Any two entries e_1,e_2 of $M(p^*)$ that are initially in the same column (row) of $M(p^*)$ after any permutation $\pi \in G(PH(n,m))$ end up in the same column (row). (However the column (row) in which e_1,e_2 are put after permutation may be different from the initial column (row)). This means that if $M(p^*)$ has w columns with k 1–entries then after any permutation $\pi \in G(PH(n,m))$ the number of such columns in $M(\pi(p^*))$ is equal to w as well. So points p and p' cannot be symmetric.

Proof of Proposition 13**.** Let us show that there are permutations $\pi, \pi э \in G(PH(n,m)))$ such that $q=\pi(p)$ and $q=\pi э'(p')$, i.e. that p and q and p' and q are in the same equivalence class. (This would mean that p and p' have to be in the same equivalence class as well and so p and p' are symmetric.)

Since $p,p' \in S(n,m)$ then both p and p' have only columns containing no more than two 1–entries. Denote by $n_0(p),n_1(p),n_2(p)$ the numbers of columns of $M(p)$ containing zero, one and two 1–entries respectively ($n_2(p)$ can be equal only to 0 or 1). Since signatures of p and p' are identical modulo permutation then $n_0(p)= n_0(p'),n_1(p)= n_1(p'),n_2(p)= n_2(p')$. Since we want to find q such that $q=\pi(p)$ and $q=\pi'(p')$ then $n_0(q),n_1(q),n_2(q)$ must be the same as for points p and p'. Let q be the point of $S(n,m)$ such that in $M(q)$ all the columns with one 1–entry go first, then they are followed by a column of two 1–entries (if such a column exists in $M(q)$) and the rest of the columns of $M(q)$ do not contain 1–entries. Besides, if j–th column of $M(q)$ contains only one 1–entry, then this 1–entry is located in the j–th row. If j–th column of $M(q)$ contains two 1–entries then they are located in j–th and $(j+1)$–th rows. It is not hard to see that each row of $M(q)$ contains at most one 1–entry and so $q \in S(n,m)$.

Point p can be transformed to q by a permutation $\pi=\pi_1\pi_2$ where π_1 and π_2 are defined as follows. π_1 is a permutation of columns of matrix $M(p)$ that makes $n_1(p)$ columns having only one 1–entry the first columns of $M(\pi_1(p))$. Besides, permutation π_1 makes the column of $M(p)$ that has two 1–entries (if such a column exists) the $(n_1(p)+1)$–th column of $M(\pi_1(p))$. π_2 is a permutation of rows of matrix $M(\pi_1(p))$ that places the 1–

entry of j–th column, $1 \leq j \leq n_1(p)$ in the j–th row of $M(\pi_2(\pi_1(p)))$. Besides, permutation π_2 places the two 1–entries of the $(n_1(p)+1)$–th column of $M(\pi_1(p))$ (if such a column with two 1–entries exists) in $(n_1(p)+1)$–th and $(n_1(p)+2)$–th rows of $M(\pi_2(\pi_1(p)))$ respectively. Since all rows of $M(\pi_1(p))$ have at most one 1–entry, permutation π_2 always exists. It is not hard to see that $M(\pi_2(\pi_1(p)))$ is equal to $M(q)$ described above. The same procedure can be applied to point p'.

Proof of Proposition 14**.** First of all, it is not hard to see that points from $S_1(n,m)$ and $S_2(n,m)$ have different signatures (for a point p of $S_2(n,m)$ matrix $M(p)$ has a column with two 1–entries, while points of $S_1(n,m)$ do not have such columns in their matrix representation). This means that no equivalence class contains points from both $S_1(n,m)$ and $S_2(n,m)$. For a point p of $S_1(n,m)$ matrix $M(p)$ can have k columns with one 1–entry where k ranges from 0 to m. From Proposition 12 and Proposition 13 it follows that points with the same value of k in their signatures are in the same equivalence class while points with different values of k in their signatures are in different equivalence classes. So there are m+1 equivalence classes in $S_1(n,m)$.

For a point p of $S_2(n,m)$ matrix $M(p)$ has exactly one column with two 1–entries. Besides, $M(p)$ can have k columns with one 1–entry where k ranges from 0 to m–1. Points with the same value of k in their signatures are in the same equivalence class while points with different value of k in their signatures are in different equivalence classes. So there are m equivalence classes in $S_2(n,m)$. Hence the total number of equivalence classes in $S(n,m)$ is $2*m+1$.

Proof of Proposition 15**.** According to Proposition 14 set $S(n,m)$ consists of $2*m+1$ equivalence classes. Let S' be a set consisting of $2*m+1$ points where each point is a representative of an equivalence class (one representative per class). According to Proposition 7 set S' is stable with respect to F and g modulo symmetry.

Proof of Proposition 16**.** The algorithm in question can have only two kinds of steps. At a step of the first kind at least one point of $Nbhd(p,g(p))$ (where p is the point of the *Boundary* picked at the current step) is added to the *Boundary*. At a step of the second kind no new point is added to the *Boundary* (because each point of p' of $Nbhd(p,g(p))$ is either in *Body* ∪ *Boundary* or the latter contains a point p'' that is symmetric to p'). The number of steps of the first kind is less or equal to $2*m+1$. Indeed, the total number of points contained in *Body* ∪ *Boundary* cannot exceed the number of equivalence classes (which is equal to $2*m+1$) because no new point is added to *Boundary* if it is symmetric to a point of *Body* ∪ *Boundary*. The number of steps of the second kind is also less or equal to $2*m+1$. The reason is that at each step of the second kind a point of the *Boundary* is moved to the *Body* and the total number of points that can appear in the *Boundary* is bounded by the number of equivalence classes in $S(n,m)$ i.e. by $2*m+1$. So the total number of steps in the algorithm is bounded by $2*(2*m+1)$. At each step of the algorithm it is checked if the current point is symmetric to a point of *Body* ∪ *Boundary*.

The complexity of this operation is bounded by $(2*m+1)*f$ where $2*m+1$ is the maximum number of points set *Body* $\cup$ *Boundary* can have and f is the complexity of checking whether two points of B^{n*m} are in the same equivalence class. So the time complexity of the algorithm is $O(m^2*f)$.

Appendix C . Some experimental results

In this appendix we compute SSPs for two classes of CNF formulas: random formulas and pigeon−hole formulas. These classes were shown to be exponentially hard for general resolution [2,6]. Table 1 gives the results of computing SSPs for random CNF formulas from the "hard" domain [8] (the number of clauses is 4.25 times the number of variables). For computing SSPs we used the algorithm described in Section 4 enhanced by the following heuristic. When picking a clause to be assigned to the current point *p'* of the *Boundary* at Step 5, we give preference to the clause *C* (falsified by *p'*) for which the maximum number of points of *Nbhd*(*p'*,*C*) are already in *Body* or *Boundary*. In other words, when choosing the clause *C* to be assigned to *p'*, we try to minimize the number of new points to be added to the *Boundary*.

We generated 10 random CNFs of each size (number of variables). Table 1 gives the average values of the SSP size and the share (percent) of the Boolean space taken by an SSP. It is not hard to see that the SSP size grows very quickly. So even for very small formulas it is very large. An interesting fact though is that the share of the Boolean space taken by SSPs constructed by the described algorithm steadily decreases as the number of variables grows.

Such a poor performance on random formulas can be explained by the following two flaws of the described algorithm. First, an SSP is constructed point−by−point while computing an SSP in larger chunks of points (clustering "similar" points of the Boolean space) should be much more efficient. Second, the algorithm looks for a set of points that is stable with respect to the initial set of clauses. On the other hand, if the algorithm is allowed to resolve clauses of the initial CNF, it may find a much smaller set of points that is stable with respect to a set of resolvents.

Table 2 shows, however, that even point−by−point SSP computation can be efficient. In Table 2 we compare the performance of SAT−solver Chaff [9] and the proposed algorithm of SSP computation on formulas $PH(n+1,n)$ (i.e. $n+1$ pigeons and n holes). Chaff, which is currently considered as the best SAT−solver based on the DPLL procedure [4], takes about 1 hour to finish the formula of 12 holes. It is not hard to see that Chaff's runtime grows up at least 5 times as the size of the instance increases just by one hole.

Table 1. SSPs of "hard" random CNF formulas

number of variables	the size of SSP	#SSP / #All_Space %
10	430	41.97
11	827	40.39
12	1,491	36.41

13	2,714	33.13
14	4,931	30.10
15	8,639	26.36
16	16,200	24.72
17	30,381	23.18
18	56,836	21.68
19	103,428	19.73
20	195,220	18.62
21	392,510	18.72
22	736,329	17.55
23	1,370,890	16.34

For each of the formulas of Table 2 a set of points that was stable modulo symmetry was computed using the algorithm described in the end of Section 5. This algorithm was implemented in a program written in C++. To check whether two points of the Boolean space were symmetric the algorithm just compared their signatures. Points with identical (modulo symmetry) signatures were assumed to be symmetric. This means that the runtimes for computing SSPs given in Table 2 do not take into account the time needed for symmetry checks. By a symmetry check we mean checking if a point p to be added to the *Boundary* is symmetric to a point p' of the current set *Boundary* $\cup$ *Body*. A more general version of the algorithm, instead of comparing signatures would have to check if there is a symmetry of $PH(n+1,n)$ that transforms p to a point of *Boundary* $\cup$ *Body* or vice versa. Nevertheless, Table 2 gives an idea of how easy formulas $PH(n,m)$ can be solved by constructing an SSP modulo symmetry.

Table 2. Solving $PH(n+1,n)$ formulas

Number of holes	Number of variables	Chaff Time (sec.)	Computing SSP modulo symmetry	
			Time (sec.)	Size of SSP modulo symmetry
8	72	2.2	0.05	17
9	90	10.6	0.07	19
10	110	51.0	0.09	21
11	132	447.9	0.13	23
12	156	3532.3	0.17	25
15	240	> 1 hour	0.38	31
20	420	> 1 hour	1.04	41
40	1640	> 1 hour	13.33	81