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

Eugene Goldberg (egold@cadence.com)
Cadence Berkeley Labs, 2001 Addison Str., 3^{rd} floor, Berkeley, California, 94704-1103, USA
phone: (510)-647-2825,fax: (510)-486-0205

Abstract. We show that a conjunctive normal form (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. So testing the satisfiability of a CNF formula reduces to looking for a stable set of points (SSP). We give some properties of SSPs and describe a simple algorithm for constructing an SSP for a CNF formula. Building 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 F 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 set of points whose size is quadratic in the number of holes. Finally, we introduce the notion of an SSP with excluded directions and sketch a procedure of satisfiability testing based on the construction of such SSPs.

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 a search space. For example, in the DPLL procedure [5], which is the basis of many algorithms used in practice, the 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 believe that SSPs can serve as a basis for constructing algorithms that traverse the search space in a "natural" way. 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 constructing an SSP. We list some properties of SSPs and describe a simple algorithm for constructing an SSP. Interestingly, this algorithm is, in a sense, an extension of Papadimitriou's algorithm [11] (or a similar algorithm that is used in the well-known program called Walksat [13]).

A very 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 (i.e. disjunctions of literals) F consists of. So the process of constructing an SSP can be viewed as a "natural" way of traversing the search space when checking F for satisfiability. In particular, 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 stable modulo symmetry set of points whose size is linear in the number of holes. The unsatisfiability of pigeon-hole CNF formulas can be proven by examining only a set of points of quadratic size.

If, for a class of formulas, SSPs are exponentially large, computing a monolithic SSP point by point is too time and memory consuming. In the paper we experimentally show that this is the case for hard random CNFs formulas. One of the possible solutions to this problem is to exclude some directions (i.e. variables) from consideration when computing an SSP. Such a set of points is stable only with respect to "movements" in the allowed directions. By excluding directions one can always get an SSP of small size. In this paper we sketch a procedure of satisfiability testing in which computing a monolithic SSP is replaced with constructing a sequence of small SSPs with excluded directions.

The notion of an SSP is the development of the idea of 1-neighborhood exploration introduced in [6]. There we described two proof systems based on the fact that for proving the unsatisfiability of a CNF formula F it suffices to examine the 1-neighborhood of all the clauses of F. (The 1-neighborhood of a clause C is the set of all points of the Boolean space that satisfy, i.e. set to 1, exactly one literal of C.) In this paper we show that it is not even necessary to examine the whole 1-neighborhood of clauses. It is sufficient to consider only a fraction of the 1-neighborhood that is an SSP. 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 that an SSP can be constructed as the set of points reachable from a point of the Boolean space. A few simple properties of SSPs are described in Section 4. In Section 5 we introduce a simple algorithm for constructing an SSP. In Section 6 we give some background on testing the satisfiability of symmetric CNF formulas. In Section 7 we show that our algorithm for constructing SSPs can be easily modified to take into account formula's symmetry. In Section 8 we apply the modified algorithm to a class of highly symmetric formulas called pigeon-hole CNF formulas. In Section 9 we introduce the notion of SSPs with excluded directions and sketch a procedure of satisfiability testing based on using such SSPs. We conclude in Section 10 with a summary of results and directions for future research.

2. Stable Set of Points

In this section, we introduce the notion of an SSP. Let F be a CNF formula of n variables $x_1, \ldots, x_n$. Denote by B the set $\{0, 1\}$ of values taken by a Boolean variable. Denote by B^n the set of points of the Boolean space specified by variables x_1,...,x_n. A point of B^n is an assignment of values to all 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, the 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 the point*** p with respect to the clause C (written $\boldsymbol{Nbhd(p,C)}$) is the set of points that are at Hamming distance 1 from p and that satisfy C.

Remark. It is not hard to see that the number of points in $Nbhd(p, C)$ is equal to that of literals in C.

Example 1. Let $C = x_1 \vee \overline{x_3} \vee x_6$ be a clause specified in the Boolean space of 6 variables $x_1, \ldots, x_6$. 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: $p_1 = (\boldsymbol{x_1}\mathbf{=1}, x_2 = 1, x_3 = 1, x_4 = 0, x_5 = 1, x_6 = 0)$, $p_2 = (x_1 = 0, x_2 = 1, \boldsymbol{x_3}\mathbf{=0}, x_4 = 0, x_5 = 1, x_6 = 0)$, $p_3 = (x_1 = 0, x_2 = 1, x_3 = 1, x_4 = 0, x_5 = 1, \boldsymbol{x_6}\mathbf{=1})$. 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 takes value 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 $p \in P$, the clause $g(p) \in F$ is falsified by p. In other words, a transport function $g{:}P \to F$ is meant to assign each point $p \in P$ a clause that is falsified by p.

Remark. We call mapping $P \to F$ 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 $Z(F)$, F be a CNF formula, and g: $P \to F$ be a transport function. The set P is called ***stable*** with respect to F and g if $\forall p \in P$, $Nbhd(p, g(p)) \subseteq P$. As it was mentioned before, "stable set of points" abbreviates to ***SSP***.

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 function g:$P \to F$ such that P is stable with respect to F and g.

Example 2. Consider an unsatisfiable CNF formula F consisting of the following 7 clauses: $C_1 = x_1 \vee x_2$, $C_2 = \overline{x_2} \vee x_3$, $C_3 = \overline{x_3} \vee x_4$, $C_4 = \overline{x_4} \vee x_1$, $C_5 = \overline{x_1} \vee x_5$, $C_6 = \overline{x_5} \vee x_6$, $C_7 = \overline{x_6} \vee \overline{x_1}$. Clauses of F are composed of literals of 6 variables: x_1,...,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 specified 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.) The 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, for every point p_i of P, the condition $Nbhd(p, g(p)) \subseteq P$ of Definition 5 holds. Consider, for example, point p_{10}=010011. The value of $g(p_{10})$ is C_2, $C_2 = \overline{x_2} \vee x_3$ and $Nbhd(p_{10}, C_2) = \{p_{11} = 000011, p_9 = 011011\}$, 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.

Proof. 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 the 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 can not 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 values 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. □

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 $Z(F) \to F$.

Proof. Since F is unsatisfiable, then $Z(F) = B^n$. For each point $p \in B^n$, condition $Nbhd(p, g(p)) \subseteq B^n$ holds. □

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

3. SSP as a reachable set of points

In this section, we introduce the notion of reachability that will be used in Section 5 to formulate an algorithm for constructing an SSP. Our main objective here is to show that the set of points reachable from a point of the Boolean space is an SSP unless this set contains a satisfying assignment.

Definition 6. Let F be a CNF formula and g: $Z(F) \to F$ be a transport function. A sequence of k points $p_1, \ldots, p_k$, $k \geq 2$ is called a ***path*** from p_1 to p_k in a set P with a transport function g if points p_1,...,p_{k-1} are in P and $p_i \in Nbhd(p_{i-1}, g(p_{i-1}))$, $2 \leq i \leq k$. (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 appears 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. The sequence of points p_1,p_{14},p_{13},p_{12} forms 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 set $Nbhd(p'', g(p''))$ where p'' is the preceding point.

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

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

Proof. 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 the 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 to 1 by s. Then C contains a literal of a variable y from Y. Denote by p' the point obtained from p by flipping the value of y in p. The point p' is reachable from p by means of the transport function g. If $|Y| = 1$, then p' is the satisfying assignment s. If $|Y| > 1$, then p' cannot be a satisfying assignment since, by our assumption,

the satisfying assignment s is the closest to p. Then after applying the same reasoning to the point p', we conclude that the clause assigned to p' by g must contain a literal of a variable y' from $Y \setminus \{y\}$. Flipping the value of y' in p' we produce a point p'' that is either the satisfying assingmnet s or is at distance $|Y| - 2$ from s. Going on in this manner we reach the satisfying assignment s in $|Y|$ steps. □

Proposition 4. Let P be a set of points that is stable with respect to a CNF formula F and a transport function $g : P \to F$. Then $\forall p \in P$, $Reachable(p, g) \subseteq P$.

Proof. Assume the contrary, i.e. that there is a point $p^* \in Reachable(p, g)$ that is not in P. Let H be a path from p to p^*. Denote by p'' the first point in the sequence of points specified by H that is not in P. (Points are numbered from p to p^*). Denote by p' the point preceding p'' in H. The point p' is in P and the latter is stable with respect to F and g. So $Nbhd(p', g(p')) \subseteq P$. The point p'' is in $Nbhd(p', g(p'))$ and so it has to be in P. We have a contradiction. □

Proposition 5. Let F be a CNF formula, $g : Z(F) \to F$ be a transport function, and p be a point from $Z(F)$. If $P = Reachable(p, g)$ does not contain a satisfying assignment for F, then P is stable with respect to F and g, and so F is unsatisfiable.

Proof. 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. □

Remark. From Proposition 5 it follows that a CNF F is satisfiable if and only if, given a point $p \in Z(F)$ and a transport function $g : Z(F) \to F$, the set $Reachable(p, g)$ contains a satisfying assignment.

4. Some Properties of SSPs

In this section, we describe some properties of SSPs. We show that the union of two SSPs is an SSP. The intersection of two SSPs is also proven to be an SSP unless this intersection is an empty set. Though the propositions of this section are not used in the rest of the paper, we list them hoping that they may come useful in developing an algebra of SSPs.

Proposition 6. Let F be a CNF formula, $P, P' \subseteq B^n$ be two sets of points, and $g : P \rightarrow F$, $g' : P' \rightarrow F$ be transport functions. Let P, P' be stable with respect to F and transport functions g and g' respectively. Then set $P'' = P \cup P'$ is also stable with respect to F and some transport function g''.

Proof. Denote by g'' the transport function $P'' \rightarrow F$ such that $g''(p) = g(p)$ if $p \in P$ and $g''(p) = g'(p)$ if $p \in P''\backslash P$. Let p be a point of P''. Consider the following two cases.

1) $p \in P$. Since P is stable with respect to g then $Nbhd(p, g(p)) \subseteq P$. Since $g''(p) = g(p)$ it is also true that $Nbhd(p, g''(p)) \subseteq P$ and so $Nbhd(p, g''(p)) \subseteq P''$.

2) $p \in P''\backslash P$. Since $P'' = P \cup P'$ then $P''\backslash P \subseteq P'$ and so $p \in P'$. Since P' is stable with respect to g' then $Nbhd(p, g'(p)) \subseteq P'$. Since $g''(p) = g'(p)$ it is also true that $Nbhd(p, g''(p)) \subseteq P'$ and so $Nbhd(p,g"(p)) \subseteq P''$.

Since for any point $p \in P''$ it is true that $Nbhd(p, g''(p)) \subseteq P''$, then P'' is stable with respect to F and transport function g''. □

The following proposition may be useful in minimizing the size of SSPs.

Proposition 7. Let F be an unsatisfiable CNF formula, and P be the set of points reachable from a point p by means of transport function $g : P \rightarrow F$. Denote by P^* a subset of P consisting of the points of P from which there is no path to p. (In particular, the point p itself is not included in P^*). If P^* is not empty, then it is stable with respect to the CNF formula F and the transport function g.

Proof. Assume the contrary, i.e. that the set P^* is not stable. Then there is a point $p' \in P^*$ such that some point p'' from $Nbhd(p', g(p'))$ is not in P^*. Since P is stable, then $p'' \in P$. Since $p'' \in P\backslash P^*$, the point p is reachable from p'' (all the points from which there is no path to p are in P^*). On the other hand, there is a path from p' to p''. This means that there is a path from p' to p going through p'', which contradicts the fact that $p' \in P^*$. □

The following two auxilliary propositions are used in the proof of Proposition 10.

Proposition 8. Let F be an unsatisfiable CNF formula and $P = \{p_1, .., p_k\}$ be a subset of B^n that is stable with respect to F and a transport function g:$P \rightarrow F$. Then $P = Reachable(p_1, g) \cup \ldots \cup Reachable(p_k, g)$.

Proof. According to Proposition 4, for any point $p_i \in P$ the set $Reachable(p_i, g)$ is a subset of P. So $(Reachable(p_1, g) \cup \ldots \cup Reachable(p_k, g)) \subseteq P$. On the other hand, the set $Reachable(p_i, g)$ contains the point p_i. So $Reachable(p_1, g) \cup \ldots \cup Reachable(p_k, g) \supseteq P$. □

Proposition 9. Let F be an unsatisfiable CNF formula and $g{:}B^n \to F$ be a transport function. Let p and p' be two points from B^n. Then, if set $P = Reachable(p, g) \cap Reachable(p', g)$ is not empty, it is stable with respect to F and g.

Proof. Let p'' be a point of P. Then $Reachable(p'', g) \subseteq Reachable(p, g)$ and $Reachable(p'', g) \subseteq Reachable(p', g)$ because any point reachable from p'' is also reachable from p and p'. Hence $Reachable(p'', g) \subseteq P$. Then P can be represented as $Reachable(p_1, g) \cup \ldots \cup Reachable(p_m, g)$ where $p_1, \ldots, p_m$ are the points from which P consists of. Hence P can be represented as a union of stable sets of points. According to Proposition 6, the set P is stable as well. □

Proposition 10. Let F be unsatisfiable and $g : B^n \to F$ be a transport function. Let P and P' be two sets that are stable with respect to F and g. Then , if $P'' = P \cap P'$ is not empty, it is stable with respect to F and g.

Proof. Let $P = \{p_1, \ldots, p_k\}$ and $P' = \{p'_1, \ldots, p'_d\}$. From Proposition 8 it follows that $P = Reachable(p_1, g) \cup \ldots \cup Reachable(p_k, g)$ and $P' = Reachable(p'_1, g) \cup \ldots \cup Reachable(p'_d, g)$. Then the set $P \cap P'$ can be represented as the union of $k * d$ sets $Reachable(p_i, g) \cap Reachable(p'_j, g)$, $i = 1, \ldots, k$, $j = 1, \ldots, d$. According to Proposition 9, the set $Reachable(p_i, g) \cap Reachable(p'_j, g)$ is either empty or stable . Then the set $P \cap P'$ is either empty (if every set $Reachable(p_i, g) \cap Reachable(p'_j, g)$ is empty) or it is the union of stable sets. In the latter case, according to Proposition 6, the set $P \cap P'$ is stable. □

5. Testing Satisfiability of CNF Formulas by SSP Construction

In this section, we describe a simple algorithm for constructing an SSP that is based on Proposition 5. 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 the set $Reachable(p, g)$. Since no transport function g :

$Z(F) \rightarrow F$ is known beforehand, it is built on the fly. In the description of the algorithm given below, the set $Reachable(p, g)$ is broken down into two parts: *Boundary* and *Body*. *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 $Body \cup Boundary$) are added to the *Boundary*. This goes on until a stable set is constructed (F is unsatisfiable) or a satisfying assignment is found (F is satisfiable).

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

Interestingly, the algorithm described above can be viewed as an extension of Papadimitriou's algorithm [11] (or a similar algorithm used in the program Walksat [13]) to the case of unsatisfiable CNF formulas. Papadimitriou's algorithm (and Walksat) can be applied only to satisfiable CNF formulas since it does 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.

Below we 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 a small SSP.

Proposition 11. There is a subclass of unsatisfiable 2-CNF formulas such that for each formula of this subclass there is an SSP of linear size (in the number of variables).

The *proof* of this proposition and the other proofs that are skipped in the body of the paper are given in the appendix.

A natural question to ask is: "What is the size of SSPs for "hard" CNF formulas?". One example of such formulas are random CNFs for which general resolution was proven to have exponential complexity [2]. Table I gives the results of computing SSPs for CNF formulas from the "hard" domain (the number of clauses is 4.25 times the number of variables [9]). For computing SSPs we used the algorithm described above enhanced by the following heuristic. When picking a clause to be assigned to the current point p' of *Boundary* (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 we have to add to *Boundary*.

We generated 10 random CNFs of each size (number of variables). The starting point was chosen randomly. Table I 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 the SSP constructed by the described algorithm steadily decreases as the number of variables grows.

Table I. SSPs of "hard" random CNF formulas

number of variables	SSP size	#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

The poor performance of the proposed algorithm on random CNF formulas suggests that computing a "monolythic" SSP point by point

is too time and memory consuming. There are at least three ways of solving the problem. One of them (that is described in the paper in full detail) concerns computing SSPs for symmetric CNF formulas. In Sections 7 and 8 we show that to prove that a symmetric CNF formula is unsatisfiable it suffices to build a set of points that is stable modulo symmetry. Such a set of points can be very small. For example, for the pigeon-hole formulas that are considered in Section 8 there is a set of points stable modulo symmetry whose size is linear in the number of holes.

Another way of dealing with the exponential blow-up of SSPs is briefly described in Section 9. The idea is to exclude some directions (i.e. variables) from consideration when computing an SSP. This way the size of an SSP can be drastically reduced. By constructing an SSP with excluded directions one obtains a new implicate of the formula. By adding this implicate to the formula we make it "simpler" (in terms of the size of its SSPs). By computing SSPs with excluded directions and adding the corresponding implicates we replace the compution of a monolithic SSP with the construction of a sequence of small size SSPs. The third (and probably most promising) way of making SSP computation more efficient is to build SSP in big "chunks" clustering "similar" points. That is, instead of computing the 1-neighborhood of a single point we construct the 1-neighborhood of a big cluster of points. We do not study this idea in the paper leaving it for future research.

6. Testing Satisfiability of Symmetric CNF Formulas

In this section, 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 [8] 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 one to "shortcut" the deduction of implicates that are symmetric to ones deduced before. Pigeon-hole formulas was the first class of CNF formulas for which the resolution proof system was shown to have exponential complexity [7]. In [15] 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. Refutations of polynomial size can be also produced in other proof systems e.g. the cutting planes refutation system [3] and extended resolution [7]. Unfortunately, all these systems give only non-

deterministic proofs and so the obtained results are not very helpful in designing deterministic algorithms.

Practical (and hence deterministic) algorithms for testing satisfiability of symmetric formulas were considered in [1, 4, 12, 14]. 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 [4] it was suggested to add to the CNF formula F to be tested for satisfiability a set Q of "symmetry-breaking" clauses. The idea is to find such a set Q of clauses that only one point of each symmetry class satisfies all the clauses of Q. This way the search space in symmetric portions of the Boolean space is pruned earlier than without adding clauses of Q (if a clause of Q is falsified before any clause of F). The generation of symmetry-breaking clauses Q 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 this approach is that the problem of generating a full set of symmetry-breaking clauses is NP-hard [4]. Moreover, for some groups the number of all clauses that have to be generated to break all the symmetris of the group is exponential [12]. This leads to the next 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. However, since symmetry processing and satisfiability testing are performed separately, at the symmetry processing step we do not know which symmetries should be broken. This suggests that even though incorporating symmetry processing into the current backtracking algorithms is difficult, satisfiability testing and symmetry processing should be tightly linked. 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 property 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.

7. Testing Satisfiability of Symmetric CNF Formulas by SSP Construction

In this section, we introduce the notion of a set of points that is stable modulo symmetry. This notion allows one to modify the algorithm of SSP construction given in Section 5 to take into account a formula's symmetry. The modification itself is described at the end of the section. In this paper, we consider only the case of permutations. However, a similar approach can be applied to a more general class of symmetries e.g. to the case when a CNF formula is symmetric under permutations combined with the negation of some variables.

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

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

Definition 9. A CNF formula F is called ***symmetric*** with respect to permutation π if the CNF formula $\pi(F)$ consists of the same clauses as F. In other words, F is symmetric with respect to π if each clause $\pi(C_i)$ of $\pi(F)$ is identical to a clause $C_k \in F$.

Proposition 12. Let p be a point of B^n and C be a clause falsified by p i.e. $C(p)$=0. Let π be a permutation of variables $\{x_1, \ldots, x_n\}$ and $C' = \pi(C)$ and $p' = \pi(p)$. Then $C'(p') = 0$.

Proof. Let $\delta(x_i)$ be the literal of a variable x_i that is present in C. This literal is set to 0 by the value of x_i in p. The variable x_i is mapped to $\pi(x_i)$ in the clause C' and the point p'. Then the value of $\pi(x_i)$ in the point p' is the same as that of x_i in p. So the value of literal $\delta(\pi(x_i))$ in the point p' is the same as the value of $\delta(x_i)$ in p i.e. 0. Hence, the clause C' is falsified by p'. □

Remark. From Proposition 12 it follows that if F is symmetric with respect to a permutation π then $F(p) = F(\pi(p))$. In other words, F takes the same value at points p and $\pi(p)$.

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. Let B^n be the Boolean space specified by variables X=$\{x_1,....,x_n\}$ and G be a group of permutations specified on X. Denote by ***symm(p,p',G)*** the following binary relation between points of B_n. A pair of points (p, p') is in $symm(p, p', G)$ if and only if there is $\pi \in G$ such that $p' = \pi(p)$.

Remark. $symm(p, p', G)$ is an equivalence relation and so breaks B^n into equivalence classes. In group theory the set of points that can be produced by applying to p the elements of a group G (i.e. the set of points that are in the same equivalence class as p) is called the orbit of p.

Definition 11. Points p and p' are called ***symmetric points*** if they are in the same equivalence class of $symm(p,p',G)$.

Definition 12. Let F be a CNF formula that is symmetric with respect to a group of permutations G and P be a subset of $Z(F)$. The set P is called ***stable modulo symmetry*** with respect to F and a 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 13. Let B^n be the Boolean space specified by variables $X = \{x_1, \ldots, x_n\}$. Let p be a point of B^n, C be a clause falsified by p, and a point $q \in Nbhd(p, C)$ be obtained from p by flipping the value of a 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 $Nbhd(p', C')$ that is symmetric to q.

Proof. The value of a variable x_k, $k \neq i$ in q is the same as in p. Besides, the value of the variable $\pi(x_k)$ in q' is the same as in p' (q' is obtained from p' by changing the value of the 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 the variable $\pi(x_i)$ in q' is obtained by the negation of the value of $\pi(x_i)$ in p'. Hence the values of the variable x_i in q and the variable $\pi(x_i)$ in q' are the same. So $q' = \pi(q)$. □

Proposition 14. Let F be a CNF formula, P be a subset of $Z(F)$, and $g : P \to F$ be a transport function. If P is stable modulo symmetry with respect to F and g, then the CNF formula F is unsatisfiable.

Proof. Denote by $K(p)$ the set of all points that are symmetric to the point p i.e. that are in the same equivalence class of the relation *symm* as p. 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. Then there is a point $p \in P$ that is symmetric to p' and so $p' = \pi(p)$, $\pi \in G$. 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.

Now we show that $K(P)$ is stable with respect to F and g: $K(P) \to F$. Let p' be a point of $K(P)$. Then there is a point p of P that is symmetric to p' and so $p' = \pi(p)$. Then from Proposition 13 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 $q' \in K(P)$. This means that $Nbhd(p', g(p')) \subseteq K(P)$ and so $K(P)$ is stable. Then according to Proposition 1, the CNF formula F is unsatisfiable. □

Remark. The idea of the proof was suggested to the author by Howard Wong-Toi [16].

Proposition 15. Let $P \subseteq B^n$ be a set of points that is stable with respect to a CNF formula F and transport function $g : P \to F$. 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.

Proof. 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 the 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'. □

Definition 13. Let F be a CNF formula, G be its group of permutations, p be a point of $Z(F)$, and g: $P \to F$ be a transport function. A set $Reachable(p, g, G)$ is called the set of points ***reachable from p modulo***

symmetry if a) the point p is in $Reachable(p, g, G)$ b) each point p' that is reachable from p by means of the transport function g is either in $Reachable(p, g, G)$ or there exists a point $p'' \in Reachable(p, g, G)$ that is symmetric to p'.

Proposition 16. Let F be a CNF formula, G be its group of permutations, p be a point of $Z(F)$, and $g : P \to F$ be a transport function. If the set P=$Reachable(p, g, G)$ does not contain a satisfying assignment, then it is stable modulo symmetry with respect to F and g and so F is unsatisfiable.

Proof. Assume the contrary, i.e. that P is not stable modulo symmetry. 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. □

Remark. From Proposition 16 it follows that a CNF F that is symmetric with respect to a group of permutations G is satisfiable if and only if, given a point $p \in Z(F)$, a transport function $g : Z(F) \to F$, the set $Reachable(p, g, G)$ contains a satisfying assignment.

Let F be a CNF formula and G be its group of permutations. According to Proposition 16 when testing the satisfiability of F it is sufficient to construct a set $Reachable(p, g, G)$. This set can be built by the algorithm of Section 5 in which step 5 is modified in the following way. Before adding a point p'' from $Nbhd(p', C)\backslash(Body \cup Boundary)$ to $Boundary$ it is checked if there is a point p^* of $Boundary \cup Body$ that is symmetric to p''. If such a point exists, then p'' is not added to $Boundary$.

8. Computing SSPs for Pigeon-Hole CNF formulas

In this section, we apply the theory of Section 7 to a class of symmetric formulas called pigeon-hole formulas. Pigeon-hole CNF formulas, by means of propositional logic, describe the fact that if $n > m$, n objects (pigeons) cannot be placed in m holes so that no two objects occupy the same hole.

Definition 14. Let a Boolean variable $ph(i, k)$ specify if i-th pigeon is in k-th hole ($ph(i, k) = 1$ means that the pigeon is in the hole). ***A pigeon-hole CNF formula*** (written $PH(n, m)$) consists of the

following two sets of clauses (denote them by $H_1(n,m)$ and $H_2(n,m)$). The set $H_1(n,m)$ consists of n clauses $ph(i,1) \vee ph(i,2) \vee ... \vee ph(i,m)$, $i = 1,\ldots,n$, i-th clause encoding the fact that i-th pigeon has to be in at least one hole. The set $H_2(n,m)$ consists of $m*n*(n-1)/2$ clauses $\overline{ph(i,k)} \vee \overline{ph(j,k)}$, $i < j$, $1 \leq i,j \leq n$, $1 \leq k \leq m$, each clause encoding the fact that i-th and j-th pigeons , $i \neq j$, cannot be placed in the same k-th hole.

Remark. Henceforth, we consider only the unsatisfiable CNF formulas $PH(n,m)$ i.e. those of them for which $n > m$.

The 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. The entry $M(i,j)$ of the matrix corresponds to the 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 a 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 by $S_1(n,m)$ and $S_2(n,m)$. A point p is included into the 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 into the subset $S_2(n,m)$ if and only if the matrix $M(p)$ has exactly one column containing two 1-entries and the rest of the columns have at most one 1-entry. Besides, $M(p)$ 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 the clause $ph(i,1) \vee ph(i,2) \vee ... \vee ph(i,m)$ of $H_1(n,m)$. For each point p of $S_2(n,m)$ there is exactly one clause of $H_2(n,m)$ 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 $H_2(n,m)$ the point p does not satisfy is $\overline{ph(i,k)} \vee \overline{ph(j,k)}$.

Definition 15. 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 17. The set of points $S(n,m) = S_1(n,m) \cup S_2(n,m)$ is stable with respect to the set of clauses $PH(n,m) = H_1(n,m) \cup H_2(n,m)$ and the transport function g specified by Definition 15.

Proposition 18. Let p be the point in which all the variables are assigned 0. Let $g : B^{n*m} \rightarrow PH(n,m)$ be a transport function. Then the set $Reachable(p,g)$ constructed by the algorithm described in Section 5 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 *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 the CNF formula $PH(n,m)$ (denote it by $G(PH(n,m))$) is the direct product of groups $S(n)$ and $S(m)$ where $S(n)$ is the group of all the permutations of n pigeons and $S(m)$ is the group of all the permutations of m holes.

Definition 16. Let p be a point of the Boolean space B^{n*m} of $n*m$ variables in which $PH(n,m)$ is specified. The 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 the matrix representation $M(p)$ of p, is called the ***column signature*** of p. We will say that column signatures v' of p' and v'' of p'' are ***identical modulo permutation*** if vector v' can be transformed to v'' by a permutation.

Proposition 19. Let p' and p'' be points of B^{n*m} such that their column 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.

Proof. Assume the contrary. Let points p and p' be symmetric but their signatures are not identical modulo permutation. Since p and p' are symmetric, then matrix $M(p')$ can be obtained by a permutation of rows and/or columns of $M(p)$. A permutation of rows cannot change the column signature of $M(p)$ while a permutation of columns can only permute components of the column signature of $M(p)$. So we have a contradiction. □

Proposition 20. Let p and p' be points of $S(n,m)$ such that their column signatures are identical modulo permutation. Then there is a permutation $\pi^* \in G(PH(n,m))$ such that $p' = \pi^*(p)$ i.e. points p and p' are symmetric

Proposition 21. The set $S(n,m)$ contains $2*m+1$ equivalence classes of the relation $symm(p,p',G(PH(n,m)))$.

Proposition 22. There is a set of points that is stable with respect to $PH(n,m)$ and a transport function g (specified by Definition 15) modulo symmetry, and that consists of $2*m+1$ points.

Proof. According to Proposition 21,the 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 a different equivalence class. According to Proposition 15,the set S' is stable with respect to F and g modulo symmetry. □

Proposition 23. 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 at the end of Section 7 where the construction of the transport function is guided by the heuristic described in Proposition 18. Then the 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 $O(m^3 * f)$ where f is the complexity of checking if two points of $S(n, m)$ are symmetric. The number of points visited by the algorithm is $O(m^2)$.

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

Number of holes	Number of variables	Chaff Time (sec.)	Computing SSPs 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	> 3600	0.38	31
20	420	> 3600	1.04	41
40	1640	> 3600	13.33	81

In Table II we compare the performance of the SAT-solver Chaff [10] and the proposed algorithm of SSP computation on formulas $PH(n + 1, n)$ (i.e. $n + 1$ pigeons and n holes). Chaff is a general-purpose SAT-solver that is currently considered as the best solver based on the DPLL procedure [5]. We use Chaff not to compare with the proposed algorithm that is specially designed for symmetric formulas but to show that even small pigeon-hole formulas cannot be solved by the best general-purpose SAT-solver. Chaff takes about 1 hour to finish the formula of 12 holes. Besides, 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. For each of the formulas of Table II a set of points that is stable modulo symmetry was computed using the algorithm described at the end of Section 7. 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 column signatures. Points with identical (modulo permutation) column signatures were assumed to be symmetric. This means that the runtimes for computing SSPs given in Table II 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 column signatures of p and points of $Boundary \cup Body$, 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 II gives an idea of how easy formulas $PH(n, m)$ can be solved by contructing an SSP modulo symmetry.

9. SSPs with Excluded Directions

Unfortunately, the theory developed in Sections 7 and 8 does not help in solving CNF formulas that have no (or have very few) symmetris. In this section, we describe a different way of reducing the size of SSPs. The idea is to replace the computation of a single SSP with the construction of a sequence of SSPs whose stability is "limited". These SSPs are called SSPs with excluded directions. The key point is that by excluding some directions from consideration one can drastically reduce the size of SSPs. The construction of an SSP with excluded directions allows one to generate a new clause that is an implicate of the initial CNF formula. This clause can be added to the current formula, which makes the obtained formula simpler in terms of the size of SSPs. For the new formula we can again build an SSP with excluded directions deducing a new implicate of the formula. A sketch of the procedure of satisfiability testing based on constructing SSPs with excluded directions is given at the end of the section.

Definition 17. Let F be a CNF formula. ***A set of excluded directions*** is a set E of literals that a) does not contain opposite literals of the same variable; b) there is no clause C of F such that all literals of C are in E.

Definition 18. Let F be a CNF formula and C be a clause of F. Let E be a set of excluded directions. Denote by $\boldsymbol{Nbhd(p,C,E)}$ the set of

points of $Nbhd(p, C)$ that set to 1 only the literals of C that are not in E.

Remark. Since, according to Definition 17, there is at least one literal of C that is not in E, then $Nbhd(p, C, E)$ is nonempty.

Example 4. Let a point p be equal to $(x_1 = 0, x_2 = 0, x_3 = 0, x_4 = 1, x_5 = 1, x_6 = 1)$. Let a clause C of a CNF F be equal to $x_1 \vee x_3 \vee \overline{x_6}$ and the set E of excluded directions be equal to $\{x_4, \overline{x_6}\}$. The set $Nbhd(p, C)$ consists of points p_1, p_2 and p_3 obtained from p by flipping the values of variables x_1, x_3, x_6 respectively. On the other hand, set $Nbhd(p, C, E)$ consists only of points p_1, p_2 because the point p_3 sets to 1 an "excluded" literal, namely the literal $\overline{x_6}$ of E.

Definition 19. Let P be a nonempty subset of $Z(F)$, F be a CNF formula, and g: $P \to F$ be a transport function. Let E be a set of excluded directions. The set P is called ***stable with respect to F, g and E*** if a) each point p of P sets all the literals of E to 0; b) for each point p of P, $Nbhd(p, g(p), E) \subseteq P$.

Proposition 24. If there is a set of points that is stable with respect to a CNF formula F and a set E of excluded directions, then any assignment satisfying F has to set to 1 at least one literal of E. In other words, the clause obtained by the disjunction of the literals of E is an implicate of F.

Proof. Let P be a set of points that is stable with respect to F, a transport function g and a set E of excluded directions. Make the assignments setting all the literals of E to 0. Remove from F all the clauses that are satisfied by these assignments and remove from the rest of the clauses all the literals that are in E (since they are set to 0). The obtained formula F' is unsatisfiable because the set P is stable with respect to F' and a transport function g'. Indeed, according to Definition 18, each point p of P sets all the literals of E to 0. Then the clause $C = g(p)$ of F cannot be satisfied by the assignment setting a literal l of E to 0. (If a clause C is satisfied by this assignment, it must contain the literal $\overline{l}$ but then C cannot be falsified by p.) So all the clauses assigned to the points of P by g are still in F'. Denote by g' the transport function that maps a point p of P to the clause C' obtained from the clause $C = g(p)$ by removing all the literals of E. It is not hard to see that $Nbhd(p, C') = Nbhd(p, C, E)$. So for each point p of P it is true that $Nbhd(p, g'(p)) \subseteq P$. □

Remark. A set of points stable with respect to a CNF F and a set E of excluded directions can be constructed by the algorithm of Section 5 modified in the following way. At step 1 the algorithm generates a starting point setting all the literals from E to 0. At step 5 it generates set $Nbhd(p', C, E)$ instead of $Nbhd(p', C)$.

Example 5. Let $p_1 = (x_1 = 0, x_2 = 0, x_3 = 0, x_4 = 0, x_5 = 0, x_6 = 0, x_7 = 0)$. Let F be a CNF formula containing clauses $C_1 = x_1 \vee x_2 \vee x_3, C_2 = \overline{x_1} \vee x_4 \vee x_5$ (and maybe some other clauses). Let the set E of excluded directions be equal to $\{x_2, x_3, x_4, x_5\}$. Denote by p_2 the point obtained from p_1 by flipping the value of x_1. Taking into account that p_1 falsifies clause C_1 and p_2 falsifies clause C_2 we can form the following transport function g: $g(p_1) = C_1, g(p_2) = C_2$. It is not hard to see that the set of points $P = \{p_1, p_2\}$ is stable with respect to clauses C_1, C_2, transport function g, and set E. Indeed, since literals x_2 and x_3 of C_1 are in E then $Nbhd(p_1, g(p_1), E) = \{p_2\} \subseteq P$. On the other hand, since literals x_4 and x_5 of C_2 are in E then $Nbhd(p_2, g(p_2), E) = \{p_1\} \subseteq P$. From Proposition 24 we conclude that the clause $C = x_2 \vee x_3 \vee x_4 \vee x_5$ equal to the disjunction of literals of E is an implicate of the formula F. On the other hand, it is not hard to see that C is actually the resolvent of clauses C_1 and C_2.

Remark. From Example 5 it follows that for an unsatisfiable formula F we can always choose a set E of excluded directions so that there is a set of two points that is stable with respect to F and E. Indeed, due to completeness of general resolution, in F there is always a pair of clauses C_1 and C_2 that produce a new resolvent. Then we form the set E of excluded directions consisting of all the literals of C_1 and C_2 except the literals of the variable in which the two clauses are resolved.

Below we sketch a procedure of satisfiability testing based on computing SSPs with excluded directions.

1. Compute an SSP P of a limited size trying to minimize the set E of excluded directions
2. Stop if a solution is found. The formula is satisfiable.
3. Stop if $E = \emptyset$. The formula is unsatisfiable.
4. Add the deduced clause (disjunction of the literals of E) to the current CNF formula.
5. Go to step 1.

The idea of the procedure is that adding new implicates gradually reduces the complexity of the inital formula F in terms of the size of "monolythic" SSPs. The claim that the size of SSPs decreases is based on the following observations. Any set of points that is stable with respect to a CNF formula F is also stable with respect to a CNF $F \cup \{C\}$ where C is a clause. So by adding clauses we preseve the best SSPs seen so far and may produce even smaller ones. The latter follows from the fact that by adding new implicates we will eventually produce an empty clause (at step 3 of the procedure above) and any set of clauses containing an empty clause has an SSP consisiting of only one point.

An important advantage of obtaining new implicates by computing SSPs with excluded directions is that directions can be excluded on the fly. The choice of directions to exclude should be aimed at the reduction of the size of the constructed SSP (that is the directions that may lead to the blow-up of the SSP should be excluded). Besides, when excluding directions one can make use of the information about the structure of the CNF formula to be tested for satisfiability.

10. 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. We describe a few operations on SSPs that produce new SSPs. These operations can serve the basis of an SSP algebra to be developed in the future. 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 cubic (in the number of holes) time and there is a stable (modulo symmetry) set of points of linear size. Finally, we introduce the notion of an SSP with excluded direction and describe a procedure of satisfiability testing based on constructing such SSPs.

An interesting direction for future research is to relate SSPs of a CNF formula to the complexity of proving its unsatisfiability. On the practical side, it is important to develop algorithms that a) implement the procedure sketched in Section 9; b) are able to construct an SSP in "chunks" clustering points that are "similar".

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. W.Cook, C.R.Coullard, G.Turan.*On the complexity of cutting planes proofs.* Discrete Applied Mathematics, 18,1987,25-38.
4. J.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).
5. M.Davis, G.Logemann, D.Loveland. *A Machine program for theorem proving.* Communications of the ACM. -1962. -V.5. -P.394-397.
6. E.Goldberg. *Proving unsatisfiability of CNFs locally.* Proceedings of LICS 2001 Workshop on Theory and Applications of Satisfiability Testing.
7. A.Haken. *The intractability of resolution.* Theor. Comput. Sci. 39 (1985),297-308.
8. B.Krishnamurthy. *Short proofs for tricky formulas.* Acta Informatica 22 (1985) 253-275.
9. D.Mitchell, B.Selman, H.J.Levesque. *Hard and easy distributions of SAT problems.* Proceedings AAAI-92, San Jose,CA, 459-465.
10. M.Moskewicz, C.Madigan, Y.Zhao, L.Zhang, S.Malik. *Chaff: Engineering an Efficient SAT Solver.* Proceedings of DAC-2001.
11. C.Papadimitriou. *On selecting a satisfying truth assignment.* Proceedings of FOC-91.
12. A.Roy. *Symmetry breaking and fault tolerance in Boolean satisfiability.* PhD thesis. Downloadable from http://www.cs.uoregon.edu/~aroy/
13. B.Selman, H.Kautz, B.Cohen. *Noise strategies for improving local search.* Proceedings of AAAI-94.
14. I. Shlyakhter. *Generating effective symmetry breaking predicates for search problems.* Proceedings of LICS 2001 Workshop on Theory and Applications of Satisfiability Testing
15. A.Urquhart. *The symmetry rule in propositional logic.* Discrete Applied Mathematics 96-97(1999):177-193,1999.
16. H.Wong-Toi. *Private communication.*

11. Appendix

Proof of Proposition 11. Consider the following subclass of 2-CNF formulas. Denote by $F_k(x_1,\ldots,x_n)$ a formula of n arguments from this subclass. The formula $F_k(x_1,\ldots,x_n)$ consists of the following two-literal clauses: $C_1 = x_1 \vee x_2$, $C_2 = \overline{x_2} \vee x_3$, $\ldots$,$C_{k-1} = \overline{x_{k-1}} \vee x_k$, $C_k = \overline{x_k} \vee x_1$, $C_{k+1} = \overline{x_1} \vee x_{k+1}$, $C_{k+2} = \overline{x_{k+1}} \vee x_{k+2}$,$C_{k+3} = \overline{x_{k+2}} \vee x_{k+3}$, $\ldots$,$C_n = \overline{x_{n-1}} \vee x_n$, $C_{n+1} = \overline{x_n} \vee \overline{x_1}$, $2 < k < n-1$. An instance of the subclass (for $n = 6, k = 4$) is given in Example 2. The formula

$F_k(x_1, \ldots, x_n)$ consists of $n+1$ clauses. The positive (or negative) literal of x_i, except $i = 1$, appears only in one clause. The positive (or negative) literal of x_1 appears in two clauses.

Now we describe a linear size SSP for $F_k(x_1, \ldots, x_n)$. This SSP consists of $2 * (n+1)$ points $P = \{p_1, \ldots, p_{2*(n+1)}\}$. The point p_1 consists only of 0's and the rest of the points are specified inductively using p_1 and a function g described below. The function g mapping P to clauses of $F_k(x_1, \ldots, x_n)$ is specified in the following way: $g(p_i) = C_i$ for $1 \leq i \leq n+1$, $g(p_{n+1+i}) = C_{k+1-i}$ for $1 \leq i \leq k$, and $g(p_{n+k+1+i}) = C_{n+2-i}$ for $1 \leq i \leq n-k+1$. It is not hard to check that the function g is picked in such a way that any two consecutive clauses $g(p_i), g(p_{i+1})$ are orthogonal in exactly one variable. (That is, $g(p_i)$ and $g(p_{i+1})$ contain opposite literals of some variable x_p and there is only one such variable). For example, $g(p_{n+1}) = C_{n+1} = \overline{x_n} \vee \overline{x_1}$ and $g(p_{n+2}) = C_k = \overline{x_k} \vee x_1$ and clauses C_{n+1} and C_k are orthogonal in x_1. The point p_i is obtained from p_{i-1}, $2 \leq i \leq 2 * (n+1)$, by flipping the value of the variable in which $g(p_i)$ and $g(p_{i-1})$ are orthogonal. Using this rule one can specify all the points of the set P.

It is not hard to show that for any point p_i from P it is true that p_i falsifies the clause $g(p_i)$. So g is actually a transport function. (That is, the clause assigned to a point p_i of the domain of g is falsified by p_i.) Indeed, the point p_1 in which all the variables have the value 0 falsifies the clause $g(p_1) = C_1 = x_1 \vee x_2$. Since clauses $g(p_i), g(p_{i+1})$ are orthogonal only in one variable and the point p_{i+1} is obtained from p_i by flipping this variable, then from the fact that p_i falsifies $g(p_i)$ it follows that p_{i+1} falsifies $g(p_{i+1})$.

Finally, we show that the set $P = \{p_1, \ldots, p_{2*(n+1)}\}$ is an SSP. It is not hard to see that clauses $g(p_{2*(n+1)}) = C_{k+1} = \overline{x_1} \vee x_{k+1}$ and $g(p_1) = C_1 = x_1 \vee x_2$ are orthogonal in x_1. Besides, the point $p_{2*(n+1)}$ can be obtained from the point p_1 by flipping the value of x_1 (and vice versa). So $Nbhd(p_1, g(p_1)) = \{p_{2*(n+1)}, p_2\}$ and $Nbhd(p_{2*(n+1)}, g(p_{2*(n+1)})) = \{p_1, p_{2*n+1}\}$.Besides, for any point p_i such that $2 \leq i \leq 2 * n + 1$, $Nbhd(p_i, g(p_i)) = \{p_{i-1}, p_{i+1}\}$. Hence, for every point p_i of P, $Nbhd(p_i, g(p_i)) \subseteq P$. □

Proof of Proposition 17. 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. The point p falsifies at least one clause C from $H_1(n,m)$. According to Definition 15 one of the clauses of $H_1(n,m)$ falsified by p is assigned to p by the transport function g. Assume that it is the clause $C = ph(i,1) \vee ph(i,2) \vee \ldots \vee ph(i,m))$. 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 $ph(i,j)$, $1 \leq j \leq m$. By definition, no column of $M(p)$ contains more than one 1-entry. So we have two alternatives. a) 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 $p' \in S_2(n,m)$. b) 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 $p' \in S_1(n,m)$. In either case $Nbhd(p,C) \subseteq S_1(n,m) \cup S_2(n,m)$.

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, the clause $C = \overline{ph(i,j)} \vee \overline{ph(k,j)}$. According to Definition 15, this clause is assigned to p by the transport function g. The 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 a column of two 1-entries any more (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 $Nbhd(p,C) \subseteq S_1(n,m)$ and so $Nbhd(p,C) \subseteq S_1(n,m) \cup S_2(n,m)$. □

Proof of Proposition 18. We prove this proposition by induction. Denote by $Boundary(s)$ and $Body(s)$ the sets *Boundary* and *Body* 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 the set $Boundary(s) \cup Body(s)$ is a subset of $S(n,m)$ and besides, g_s satisfies Definition 15 (at s points wherein the function g_s has been 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 15.

Now we prove that from the fact that the hypothesis holds after performing s steps of the algorithm, it follows that it also holds after $s+1$ steps. Let p' be the point of $Boundary(s)$ chosen at step $s+1$. First let us show that the transport function g_{s+1} satisfies Definition 15. 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 the transport function g_{s+1} at point p', g_{s+1} satisfies Definition 15. 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 15. Then we can apply arguments of Proposition 17 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 20. Let us show that there are permutations π,$\pi' \in G(PH(n,m))$ such that $q = \pi(p)$ and $q = \pi'(p')$, 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 column 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, the 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 the 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, the 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, the 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 21. It is not hard to see that points from $S_1(n,m)$ and $S_2(n,m)$ have different column signatures (for a point p of $S_2(n,m)$ the 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 $S_1(n,m)$ the matrix $M(p)$ can have k columns with one 1-entry where k ranges from 0 to m. From Proposition 19 and Proposition 20 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 of $S_2(n,m)$ the 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 23. The algorithm 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 *Boundary* picked at the current step) is added to *Boundary*. At a step of the second kind no new points are added to *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 *Body* and the total number of points that can appear in *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 first kind at most m neighborhood points can be generated (because a clause of $PH(n,m)$ contains at most m literals). Each point is checked if it 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 the set *Body* ∪ *Boundary* can have and f is the complexity of checking whether two points of $S(n,m)$ are symmetric. So the time complexity of the algorithm is $O(m^3*f)$. Since at most m points can be reached at each step, then the total number of points reached by the algorithm is bounded by $m*2*(2*m+1)$. □