Electronic Colloquium on Computational Complexity, Report No. 24 (2016)

Solution-Graphs of Boolean Formulas and Isomorphism

Patrick Scharpfenecker* and Jacobo Torán

University of Ulm, Institute of Theoretical Computer Science,
Ulm, Germany
patrick.scharpfenecker@uni-ulm.de,
jacobo.toran@uni-ulm.de

February 22, 2016

Abstract

The solution graph of a Boolean formula on n variables is the subgraph of the hypercube H_n induced by the satisfying assignments of the formula. The structure of solution graphs has been the object of much research in recent years since it is important for the performance of SAT-solving procedures based on local search. Several authors have studied connectivity problems in such graphs focusing on how the structure of the original formula might affect the complexity of the connectivity problems in the solution graph [9, 19].

In this paper we study the complexity of the isomorphism problem of solution graphs of Boolean formulas. We investigate how this complexity depends on the formula type, considering for this the classes of formulas introduced in [9, 19].

We observe that for general formulas the solution graph isomorphism problem can be solved in exponential time while in the cases of 2CNF formulas as well as for CPSS formulas, the problem is in the counting complexity class $C_=P$, a subclass of PSPACE. We also prove a strong property on the structure of solution graphs of Horn formulas showing that they are just unions of partial cubes.

In addition we give a PSPACE lower bound for the problem on general Boolean functions. We use a recent result on the complexity of testing the number of perfect matchings [7] to prove that for 2CNF as well as for CPSS formulas the solution graph isomorphism problem is hard for $C_=P$ under polynomial time many one reductions, thus matching the given upper bound.

1 Introduction

Schaefer provided in [17] a well known dichotomy result for the complexity of the satisfiability problem on different classes of Boolean formulas. He showed that

*Supported by DFG grant TO 200/3-1.

ISSN 1433-8092

for formulas constructed from specific Boolean functions (now called Schaefer functions), satisfiability is in P while for all other classes, satisfiability is NP-complete. Surprisingly, there are no formulas of intermediate complexity.

More recently, Gopalan et al. and Schwerdtfeger [9, 19] uncovered a similar behavior for connectivity problems on solution graphs of Boolean formulas. The solution graph of a Boolean formula on n variables is the subgraph of the n-dimensional hypercube induced by all satisfying assignments. The study of solution graphs of Boolean formulas has been the object of important research in recent years, especially for the case of random formula instances. It has been observed both empirically and analytically that the solution space breaks in many small connected components as the ratio between variables and clauses in the considered formulas approaches a critical threshold [15, 1]. This phenomenon explains the better performance on random formulas of SAT-solvers based on message passing with decimation than those based on local search or DPLL procedures (see e.g. [8]). The motivation behind the works of [9] and [19] was to obtain new information about the connectivity properties of the solution space for different types of Boolean formulas. Introducing some new classes of Boolean functions, they were able to prove a dichotomy result for the *st*-connectivity problem [9] as well as a trichotomy result for connectivity [19]. For different formula classes the complexity of the connectivity problem is either in P, or complete for coNP or for PSPACE while for *st*-connectivity it is either in P or PSPACE-complete.

In this paper we look further in the solution space of Boolean formulas studying the complexity of the isomorphism of their solution graphs. In other words, we consider the following natural questions: given two Boolean formulas, how hard is it to test if their solution graphs are isomorphic? Does the complexity of the problem depend on the structure of the formula? Observe that isomorphism of solution graphs is a very strong concept of equivalence between fromulas, stronger than Boolean isomorphism [2] and stronger than saying that both formulas have the same number of satisfying assignments. Since the complexity of the general graph isomorphism problem, GI, is not completely settled (see [13]), one might expect that it would be hard to obtain a complete classification for solution graph isomorphism. We show in fact that for different types of Boolean formulas, the complexity of the isomorphism problem on their solution graphs varies. We also characterize completely the complexity of the problem for some types of Boolean formulas. For solution graphs of 2CNF formulas, isomorphism of a single connected component is exactly as hard as testing Graph Isomorphism. For a collection of such components (encoded by a single 2CNF formula), the isomorphism problem is complete for the complexity class $C_=P$, a complexity class defined in terms of exact counting. This means that deciding isomorphism of the solution graphs of 2CNF formulas is exactly as hard as testing if two such formulas have the same number of satisfying assignment. This result also holds for the more general class of CPSS formulas (definitions in the preliminaries section). Again showing that for this class of formulas isomorphism and counting have the same complexity. For the upper bound we use a recent result on isometric dimension of partial cubes [18], the fact that GI is low for the class $C_=P$ [12] as well as the closure of this class under universal quantification [10]. The hardness property uses a result of Curticapean [7], where it is proven that $SamePM$, the problem to decide if two given graphs have the same number of perfect matchings is complete for $C_=P$. We show that this problem can

be reduced to the verification of whether two 2CNF formulas have the same number of satisfying solutions, implying that this problem and even *Iso(CPSS)* are complete for $C_=P$.

For the other types of formulas used in [9, 19], built from Schaefer, safely tight and general functions, we observe that the corresponding solution graph isomorphism problems can be solved in EXP, thus improving the trivial NEXP upper bound.

For classes of functions that are not safely tight, we can also improve the $C_=P$ lower bound and show that the isomorphism problem for their solution graphs is in fact hard for PSPACE.

Figure 1 summarizes the complexity results for isomorphism of solutions graphs for specific classes of formulas.

Function	**Hard for**	**Upper bound**
CPSS	$C_=P$	$C_=P$
Schaefer, not CPSS	$C_=P$	EXP
safely tight, not Schaefer	$C_=P$	EXP
not safely tight	PSPACE	EXP

Figure 1: Classification of Isomorphism problems.

While we could not improve the EXP upper bound for the isomorphism of solution graphs corresponding to Horn formulas, we prove a strong new property for the structure of such graphs which might help to develop a non-trivial isomorphism algorithm. We show that the set of solutions between a locally minimal and locally maximal solution is a partial cube. Therefore a solution graph can be seen as taking a partial cube for every locally maximal solution and glueing them together.

2 Preliminaries

For two words $x, y \in \{0,1\}^n$, $\Delta(x,y)$ denotes the Hamming-distance between them. We associate words in $\{0,1\}^n$ with subsets of $[n] = \{1,\ldots,n\}$ in the standard way.

We mostly deal with undirected graphs without self-loops. For such a graph $G = (V,E)$ with vertex set $V = [n]$ and edge set $E \subseteq \binom{V}{2}$, its simplex graph (see e.g. [4]) is defined as $simplex(G) = (V', E')$ with V' as the set of all cliques (including the empty clique) in G and $E' = \{\{u,v\} \in \binom{V'}{2} \mid \Delta(u,v) = 1\}$. So $G' = simplex(G)$ is the set of all cliques of G and two cliques are connected iff they differ (considered as strings of $\{0,1\}^n$) in one element. We will only consider the simplex graph of bipartite graphs. As these graphs have only cliques of size at most 2, $|V'| = |V| + |E| + 1$. G' contains all original nodes V, a node $u = \{i,j\}$ for every edge $\{i,j\} \in G$ which is connected to $\{i\}$ and $\{j\}$ and a new node $o = \emptyset$ which is connected to all original nodes.

Two graphs $G = (V,E)$ and $H = (V',E')$ with $V = V' = [n]$ are isomorphic iff there is a bijection $\pi : V \to V'$ such that for all $u, v \in V : (u,v) \in E \Leftrightarrow (\pi(u), \pi(v)) \in E'$. If such a bijection exists we write $G \cong H$, if not, $G \ncong H$. The graph isomorphism problem (GI) is the decision problem of whether two given

graphs are isomorphic. Given a class of graphs C, *Iso(C)* denotes the graph isomorphism problem on graphs in C.

The Boolean isomorphism problem consists in deciding, given two Boolean formulas F and G on variables $x_1, \ldots, x_n$, whether there is a permutation $\pi \in S_n$ of the variables such that for all $x \in \{0,1\}^n$, $F(x_1, \ldots, x_n) = G(x_{\pi(1)}, \ldots, x_{\pi(n)})$.

We deal with different classes of formulas. 2CNF denotes the class of formulas in conjunctive normal form and with exactly two literals per clause. For a 2CNF formula $F(x_1, \ldots, x_n)$ we define the directed implication graph $I(F) = (V, E)$ on nodes $V = \{x_1, \ldots, x_n, \overline{x_1}, \ldots, \overline{x_n}\}$ and edges $(k, l) \in E$ with $k, l \in V$ iff there is no solution to F which falsifies the clause $(k \to l)$. By replacing all variables in a cycle with a single variable we get the reduced implication graph $RI(F)$. We say that a 2CNF formula F is reduced if $I(F) = RI(F)$.

We deal mostly with standard complexity classes like P, NP, EXP and NEXP. A class that might not be so familiar is the counting class $\mathrm{C_{=}P}$ [22]. This consists of the class of problems A for which there is a nondeterministic polynomial time Turing machine M and a polynomial time computable function f such that for each $x \in \{0,1\}^*$, $x \in A$ if and only if the number of accepting paths of $M(x)$ is exactly $f(x)$. The standard complete problem for $\mathrm{C_{=}P}$ is ExactSAT: given a Boolean formula F and a number k, does F have exactly k satisfying assignments?

2.1 Solution graphs of Boolean formulas

Intuitively, a solution graph for a given Boolean formula is the induced subgraph on all satisfying solution represented in a host graph. In this paper we only consider induced subgraphs of the n-dimensional hypercube H_n which is the graph with $V = \{0,1\}^n$ and $E = \{\{u, v\} | \Delta(u, v) = 1\}$.

Definition 1. *Let $F(x_1, \ldots, x_n)$ be an arbitrary Boolean formula. Then the solution graph G_F is the subgraph of the n-dimensional hypercube H_n induced by all satisfying solutions x of F.*

Note that two satisfying solutions are connected by an edge iff their Hamming distance is one. For a set of Boolean formulas D (for example D =2CNF), *Iso(D)* denotes the isomorphism problem on the class of solution graphs of D-formulas.

Given a graph G and two nodes u, v, $d(u, v)$ is the length of the shortest path between u and v in G or ∞ if there is no such path.

Definition 2. *An induced subgraph G of H_n is a partial cube iff for all $x, y \in G$, $d(x, y) = \Delta(x, y)$. We call such an induced subgraph "isometric". The isometric dimension of a graph G is the smallest n such that G embeds isometrically into H_n.*

Definition 3. *A graph $G = (V, E)$ is a median graph iff for all nodes $u, v, w \in V$ there is a unique median $a \in V$ which lies on the shortest paths between (u, v), (u, w) and (v, w).*

For any Boolean function $F : \{0,1\}^n \to \{0,1\}$ we can represent F with the subset of all its satisfying assignments in $\{0,1\}^n$. A Boolean function $F \subseteq \{0,1\}^n$ is closed under a ternary operation $\odot : \{0,1\}^3 \to \{0,1\}$ iff $\forall x, y, z \in F : \odot(x, y, z) := (\odot(x_1, y_1, z_1), \ldots, \odot(x_n, y_n, z_n)) \in F$. Note that we abuse the notation of a ternary operation to an operation on three bit-vectors by applying

the operation bitwise on the three vectors. For R a set of Boolean functions with arbitrary arities (for example $R = \{(\overline{x} \vee y), (x \oplus y), (x \oplus y \oplus z)\}$, we define $SAT(R)$ to be the satisfiability problem for all Boolean formulas which are conjunctions of instantiations of functions in R. For the given example R, $F(x, y, z) = (\overline{z} \vee y) \wedge (x \oplus y)$ is a formula in which every clause is an instantiation of an R-function. Similarly, *Conn(R)* (*stConn(R)*) is the connectivity (reachability) problem, given a conjunction F of R-functions (and s, t), is the solution graph connected (is there a path from s to t). We mostly use F for Boolean formulas/functions and R, S for sets of functions.

Note that $r \in R$ can be an arbitrary Boolean function as for example $r = (x \oplus y)$ or $r = (x \vee \overline{y} \vee \overline{z}) \wedge (\overline{x} \vee z)$. With $Horn_n$ we define the set of all Horn-clauses of size up to n. The ternary majority function $maj : \{0,1\}^3 \to \{0,1\}$ is defined as $maj(a, b, c) = (a \wedge b) \vee (a \wedge c) \vee (b \wedge c)$.

In the next definitions we recall some terms introduced in [19] and [9].

Definition 4. *A Boolean function F is*

- *bijunctive, iff it is closed under $maj(a, b, c)$.*
- *affine, iff it is closed under $a \oplus b \oplus c$.*
- *Horn, iff it is closed under $a \wedge b$.*
- *dual-Horn, iff it is closed under $a \vee b$.*
- *IHSB−, iff it is closed under $a \wedge (b \vee c)$.*
- *IHSB+, iff it is closed under $a \vee (b \wedge c)$.*

A function has such a property componentwise, iff every connected component in the solution graph is closed under the corresponding operation. A function F has the additional property "safely", iff the property still holds for every function F' obtained by identification of variables[1]*.*

In the case of Horn-formulas, the usual definition (the conjunction of Horn-clauses, clauses with at most one positive literal) implies that the represented functions are Horn.

Definition 5. *A set of functions R is Schaefer (CPSS) if at least one of the following conditions holds:*

- *every function in R is bijunctive.*
- *every function in R is Horn (and safely componentwise IHSB−).*
- *every function in R is dual-Horn (and safely componentwise IHSB+).*
- *every function in R is affine.*

If we have a Boolean formula F which is build from a set of CPSS functions R we say that F is CPSS. Clearly, every CPSS formula is Schaefer. We later use a bigger class of functions called *safely tight*. This class properly contains all Schaefer sets of functions.

[1] Identifying two variables corresponds to replacing one of them with the other variable.

Definition 6. *A set R of functions is (safely) tight if at least one of the following conditions holds:*

- *every function in R is (safely) componentwise bijunctive.*
- *every function in R is (safely) OR-free.*
- *every function in R is (safely) NAND-free.*

A function is OR-free if we can not derive $(x \vee y)$ by fixing variables. Similarly, a function is NAND-free if we can not derive $(\overline{x} \vee \overline{y})$ by fixing variables.

3 Isomorphism for solution graphs

We now turn our attention to the isomorphism problem on solutions graphs. In general the solution graph of a formula can have an exponential number of connected components and each component might be of exponential size (in the formula size). The NP upper bound for GI translates directly into a NEXP upper bound for the isomorphism of solution graphs.

Based on the celebrated new algorithm from Babai for Graph Isomorphism [3] running in time $n^{\log^{O(1)}}$, it is not hard to see that the isomorphism of solution graphs is in EXP: for two given Boolean formulas on n variables, we can construct explicitely their solution graphs in time $O(2^n)$ and then apply Babai's algorithm on them, resulting in a $2^{n^{O(1)}}$ algorithm. But we do not need such a strong result, the algorithm of Luks for testing isomorphism of bounded degree graphs [14] suffices.

Theorem 1. *The problem to decide for two given Boolean formulas whether their respective solution graphs are isomorphic is in EXP.*

Proof. Luks [14] gave an algorithm for graph isomorphism with time-complexity $|V|^{deg(G)}$. A solution graph embedded in the hypercube H_n has degree at most $n-1$. The running time of Luks algorithm on such graphs is bounded by 2^{n^2}. □

By restricting the encoding formula, we can get better upper bounds. Theorem 4 will show that the isomorphism problem for $CPSS$ encoding formulas is in $\mathrm{C_{=}P}$, a subclass of PSPACE. For this, we need the following two results.

Theorem 2 ([18])**.** *Given a CPSS function $F(x_1, \ldots, x_n)$, every connected component of F is a partial cube of isometric dimension at most n.*

Theorem 3 ([16], Theorem 5.72)**.** *For any two finite isomorphic partial cubes G_1 and G_2 on a set X, there is an automorphism of the cube $H(X)$ that maps one of the partial cubes onto the other. Moreover, for any isomorphism $\alpha : G_1 \to G_2$, there is a boolean automorphism $\sigma : H(X) \to H(X)$ such that σ on G_1 is exactly α.*

We note that isomorphism of (explicitly given) partial cubes is already GI-complete. The hardness follows from the observation that for any graph, its simplex is a median graph. The other two facts we need is that median graphs are partial cubes (see e.g. [16], Theorem 5.75), and the fact that a given pair of graphs G, H, can be transformed in logarithmic space into a pair of bipartite graphs G', H' so that $simplex(G') \cong simplex(H')$ iff $G \cong H$.

To see this we first suppose that for two given general graphs G, H we know that $|E| \neq |V|$. This could easily enforced in an isomorphism-preserving logspace reduction. In a next step, we replace each edge (u, v) in both graphs with the gadget $(u, z_{u,v}), (z_{u,v}, v)$ where $z_{u,v}$ is a new vertex. This yields two new bipartite graphs G', H' which are isomorphic iff G and H were isomorphic. But then $simplex(G') \cong simplex(H')$ iff $G \cong H$. This implies the following Lemma.

Lemma 1. *Isomorphism for median graphs is GI-complete under logarithmic space many-one reductions.*

Note that it is known that median graphs can be exactly embedded as a solution graph of a reduced 2CNF formula (see e.g. [5]). Lemma 1 gives therefore an alternative reduction to the one given in [6] between Boolean isomorphism for 2CNF formulas and GI. With Theorem 3 we get:

Corollary 1. *The Isomorphism Problem for reduced 2CNF solution graphs is GI-complete under logarithmic space many-one reductions.*

Proof. The hardness part follows from the observation given above. By Theorem 3 two partial cubes are isomorphic iff there is an automorphism of the whole hypercube mapping one partial cube to the other. But such an automorphism is just a Boolean automorphism of the Boolean function. For general boolean formulas this problem is hard for NP and in Σ_2 [2], but for Schaefer-formulas, which contain 2CNF formulas, this problem can be reduced in polynomial time to GI (see [6]) by creating a unique normal form and looking for a syntactic isomorphism of the formulas. □

This basically tells us that even if we look at two exponentially sized, isomorphic partial cubes embedded in the hypercube H_n, finding an isomorphism is as easy as finding a Boolean isomorphism. The problem is more complex when the solution graphs might have more than one connected component. We face the additional problem that single connected components may not have a single formula representing just this component. For the isomorphism of solution graphs of CPSS functions we will show an upper bound of $C_=P$. For this we need the following Lemma showing that the problem of testing if there is an isomorphism between two connected components which maps a given solution to another given solution, can be reduced to GI.

Lemma 2. *Given CPSS functions F and G and two satisfying solutions s and t. Deciding if there is an isomorphism π between the connected components containing s and t with $\pi(s) = t$ can be reduced to GI.*

Proof. We know by Theorem 3 that if two partial cubes are isomorphic then there is always a Boolean isomorphism[2]. One could easily guess a candidate permutation of variables for the isomorphism. But it is not clear how to verify that this permutation is in fact an isomorphism. To reduce this problem to GI we would have to extract a single connected component and create a formula which contains only this subgraph. In general, this is not possible. We use the construction depicted in Figure 2 to achieve such an extraction which is enough in the case of isomorphism.

[2]Boolean isomorphisms are signed permutations: they may map variables to variables and may flip variables.

We describe this construction which basically performs a walk on the solution graph beginning at a given node s. We use several blocks of variables. Given the original variables $x = (x_1, \ldots, x_n)$, we create new blocks of variables x^i and $x^{i,j}$ for $i \in \{0, \ldots, w\}$ and $j \in \{1, \ldots, n\}$ (w to be fixed later), each containing n variables. For example $x^0 = (x_1^0, \ldots, x_n^0)$ and $x^{0,4} = (x_1^{0,4}, \ldots, x_n^{0,4})$. We fix the first block of variables x^0 to $s \in \{0,1\}^n$. We then add n new blocks $x^{0,1}, \ldots, x^{0,n}$ such that every $x^{0,j}$ may only differ from x^0 in bit j. If $x_j^0 = 0$ we add the clause $(x_j^0 \to x_j^{0,j})$, if $x_j^0 = 1$ we add the clause $(\overline{x_j^0} \to \overline{x_j^{0,j}})$. This will not be relevant in the first step as these clauses are obviously satisfied but this ensures that a walk never returns to s: if we add these clauses for all later steps and for example for the case $x_j^0 = 0$ there is an i such that $x_j^i = 1$ then for all $i' > i$, $x_j^{i'} \neq 0$.

All other variables have to be equivalent to the variables in the previous block (or the previous variables could get reused in $x_j^{0,j}$, except x_j^0). In addition we add for every j the clauses $F(x^{0,j})$ to ensure that every following block of x^0 satisfies F. Obviously, every $x^{0,j}$ has distance at most 1 from $x^0 = s$ and is a node in the solution graph. We then add a new block x^1 such that $x_j^1 = x_j^{0,j}$. This performs all steps of the previous branching-step in parallel and we require x^1 to satisfy F.

Although all nodes visited in the branching-step have distance at most 1 from s, the nodes described in x^1 have distance $\sum_{j \in [n]} d(x^0, x^{0,j})$. Therefore x^1 may not be in the same connected component as x^0. We now show that, in the case of CPSS functions, this can never happen. Let us assume w.l.o.g. that exactly the first k blocks have distance 1.

Claim 1. *Let S be a CPSS function with satisfying solution x. If x^i for $1 \leq i \leq k \leq n$ is equal to x with the i-th bit flipped, $x' = x_1^1, \ldots, x_k^k, x_{k+1}, \ldots, x_n$ and $x^1, \ldots, x^k, x'$ all satisfy S, then there is a path from x to x'.*

Proof: Obviously, x^i is connected to x as $d(x, x^i) = 1$. We show by induction on j that for all j with $1 \leq j \leq k$, $y^j = x_1^1, \ldots, x_j^j, x_{k+1}, \ldots, x_n$ satisfies S. As their consecutive distances are 1 the statement follows. For $j = 1$ we know that $x^1 = y^1$. Now let y^j satisfy S. If S is componentwise bijunctive, then $maj(x', y^j, x^{j+1}) = y^{j+1}$ satisfies S by its closure property. If S is not componentwise bijunctive it is Horn and componentwise IHSB- (or dual). Again the closure property gives $x' \wedge (y^j \vee x^{j+1}) = y^{j+1}$. ■

Note that the given construction on (F, x) creates a formula F' such that every satisfying solution is a walk on F of length $w := n$ starting at x (we use n branch and reduce blocks). Therefore, the set of all satisfying solutions to F' is the set of all walks on F where every step is the traversal of a complete subhypercube and in every step the walk may refuse to take a step and remain at the previous node. Obviously, if there is an isomorphism π mapping the two components onto each other such that $\pi(x) = y$, then there is an isomorphism mapping the sets of paths onto each other. This isomorphism just has to use π for every block and has to exchange the parallel steps according to π.

We can now reduce the Boolean isomorphism question between F' and G' to GI (again, using [6]) implementing the additional properties with graph gadgets. We therefore force all blocks to be mapped internally in the same way and we force the n parallel blocks to be mapped exactly as each block is mapped internally. The result are two graphs which are isomorphic iff there is

an isomorphism mapping the components rooted at x and y in F and G onto each other so that x gets mapped to y. □

Theorem 4. *Iso(CPSS)* $\in C_{=}P$.

Proof. The proof uses the fact that GI is low for the class $\mathrm{C}_{=}\mathrm{P}$ [12]. This means that a nondeterministic polynomial time algorithm with a $\mathrm{C}_{=}\mathrm{P}$ acceptance mechanism and having access to an oracle for GI, can be simulated by an algorithm of the same kind, but without the oracle. In symbols $\mathrm{C}_{=}\mathrm{P}^{\mathrm{GI}} = \mathrm{C}_{=}\mathrm{P}$.

We already know that the solution graphs of CPSS functions consist of at most an exponential number of connected components and every such component is a partial cube. For two solution graphs F and G to be isomorphic there has to be a bijection mapping each connected component of F onto an isomorphic component of G.

One way to check the existence of such a bijection is by looking at each possible partial cube and count the number of connected components isomorphic to it in both graphs. If the numbers match for all partial cubes, the graphs are isomorphic. Instead of checking all possible partial cubes, which would be too many, one only has to check the ones which occur in the graphs. For $x \in \{0,1\}^n$ let A_x and B_x be the sets

$$A_x = \{y \in \{0,1\}^n \mid F(y) = 1 \wedge F_x \cong F_y \text{with an isomorphism mapping } x \text{ to } y\}$$

$$B_x = \{y \in \{0,1\}^n \mid G(y) = 1 \wedge F_x \cong G_y \text{with an isomorphism mapping } x \text{ to } y\}$$

The existence of an isomorphism between F_x and F_y (or G_y) mapping x to y can be checked with a GI oracle (as proven in Lemma 2). Our algorithm checks for every $x \in \{0,1\}^n$ satisfying F, whether $||A_x|| = ||B_x||$. The same test is performed for all x satisfying G. Both tests are successful if and only if the graphs are isomorphic. Clearly the graphs are isomorphic if and only if both tests succeed.

This procedure shows that the problem is in the class $\forall\mathrm{C}_{=}\mathrm{P}^{GI}$.[3] Using the mentioned fact that GI is low for $\mathrm{C}_{=}\mathrm{P}$, this class coincides with $\forall\mathrm{C}_{=}\mathrm{P}$. In addition, Green showed [10] that $\mathrm{C}_{=}\mathrm{P}$ is closed under universal quantification, i.e. $\forall\mathrm{C}_{=}\mathrm{P} = \mathrm{C}_{=}\mathrm{P}$. We conclude that *Iso(CPSS)* $\in \mathrm{C}_{=}\mathrm{P}$. □

In Theorem 4 we exploited the fact that CPSS functions consist of partial cubes of small isometric dimension. But for general Schaefer functions this property does not hold. The solution graph might have an exponential isometric dimension or the connected subgraphs might even not be partial cubes. Therefore it seems improbable that the $\mathrm{C}_{=}\mathrm{P}$-algorithm can be adapted for general Schaefer solution graphs. These graphs should admit a better lower bound. Unfortunately, we can only provide such a lower bound for the more powerful class of Boolean functions that are not safely tight.

Theorem 5. *Let S be a set of functions which is not safely tight. Then Iso(S) is hard for PSPACE under logarithmic-space reductions.*

Proof. The proof is based on the reduction from s,t-connectivity to GI from [11]. We know that the s,t-connectivity problem for functions that are not safely tight

[3] We use the same quantifier notation which is common for the classes in the polynomial time hierarchy.

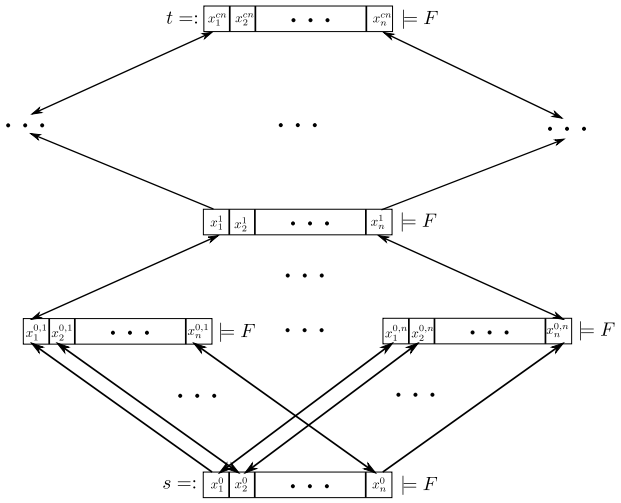

Figure 2: A walk on solution graphs.

is PSPACE-complete [9]. We give a construction of solution graphs that have colored vertices as a way to distinguish some vertices. Later we show how the formulas can be modified to produce the colors in their solution graphs. Given a formula F built on functions from S, as well as satisfying assignments s and t, we create two copies of G_F (which is the solution graph defined by F) and color vertex s in one of the copies with color white and with black in the second copy. Let $G_{F'}$ be the disjoint union of the two copies. Now we consider two copies G_{F_1} and G_{F_2} of $G_{F'}$. We color in G_{F_1} one of the copies of vertex t with the grey color while in G_{F_2}, the second copy of t is colored grey. All other nodes have no color. There is a path from s to t in G_F if and only if G_{F_1} and G_{F_2} are isomorphic.

This construction can easily be performed with solution graphs. Given the formula $F(x_1, \ldots, x_n)$, two disjoint copies of the encoded graph are defined by the formula

$$F(a, b, x_1, \ldots, x_n) = (a \leftrightarrow b) \wedge F(x_1, \ldots, x_n)$$

using two new variables a and b. Coloring a vertex by attaching a gadget to it can be done with the following construction. We assume w.l.o.g. that the node we want to color is 0^n in G_F. We add to 0^n the graph H_m with $m > n$ as neighbor. Then $F'(x_1, \ldots, x_n, y_1, \ldots, y_m) = F(x_1, \ldots, x_n) \wedge \bigwedge_{i \leq n, j \leq m}(x_i \rightarrow \overline{y_j})$. The new graph can be described as the old solution graph of F but 0^n now is the minimal node of a new, complete hypercube on m variables. Note that 0^n is the only node of the original solution graph which is part of a hypercube of dimension m. In addition, it is the only node of the hypercube of dimension m which is connected to some of the old nodes. This completes the reduction. □

The given construction uses new clauses which are Horn and 2CNF and can even be applied to simpler classes of formulas. The following statements use the hardness results of [18] with the reduction in Theorem 5.

Corollary 2. *Iso(2CNF) is hard for NL and Iso($Horn_3$) is hard for P under logspace reductions.*

Note that the resulting solution graphs in this corollary can not have more than two connected components. The isomorphism for these $2CNF$ graphs is therefore polynomial time reducible to GI.

4 Structure of solution graphs of Horn formulas

While [18] showed that CPSS formulas contain only partial cubes of small isometric dimension as connected components, Horn formulas may encode partial cubes of exponential isometric dimension or graphs which are not even partial cubes. So for the isomorphism question, things seem to get more complicated. We give an interesting property for Horn solution graphs which suggests that *Iso(Horn)* might be easier than general solution graph isomorphism.

Let $d_m(a, b)$ denote the monotone distance between a and b. So $d_m(a, b) < \infty$ iff there is a strictly monotone increasing path from a to b or vice versa. In [9] it is shown that in OR-free formulas there is a unique minimal satisfying assignment in every connected component. As Horn-formulas are OR-free, given an assignment y satisfying a Horn-formula F, the connected component of y contains a unique minimal satisfying assignment. For the next result we will assume w.l.o.g. that this minimal satisfying assignment in the connected component of y is 0^n. If this is not true, we could modify F setting all variables to 1 which are 1 in y and get a formula F' on less variables where $0^{n'}$ is the required minimal satisfying assignment. The resulting formula satisfies this property and still contains the connected component corresponding to y in F. With $[y]_F := \{a \in \{0,1\}^n \mid d_m(a, y) < \infty\}$ we denote the set of all nodes a lying between 0^n and y for which there is a monotone increasing path from a to y.

Theorem 6. *For every solution y to a Horn-formula F, $[y]_F$ is a partial cube.*

Proof. Let $a, b \in [y]_F$ be two arbitrary nodes. We show that $d(a, b) = \Delta(a, b)$. In case the two monotone increasing paths $a = a_1, \ldots, a_k = y$ from a to y and $b = b_1, \ldots, b_l = y$ from b to y are already of total length $\Delta(a, b)$, then we are done. Otherwise, suppose that there is at least one variable x_i which gets increased to 1 in both paths. The positions in the path where such variables are increased may differ. Every variable can be classified as either not changed in any of the paths, changed in only one path (and therefore contributing to $\Delta(a, b)$, or changed in both paths. We can now construct the shorter path from $a_1 \wedge y = a_1$ over $a_1 \wedge b_{l-1}$ and $a_1 \wedge b_1 = a \wedge b = b_1 \wedge a_1$ back to $b_1 \wedge a_2$ and $b_1 \wedge a_k = b_1$. Figure 3 illustrates in the first row the original path and in the second row the new path.

a_1	a_2	$\ldots$	$a_k = b_l = y$	b_{l-1}	$\ldots$	b_2	b_1
$a_1 \wedge b_l = a_1$	$a_1 \wedge b_{l-1}$	$\ldots$	$a_1 \wedge b_1 = a \wedge b$	$b_1 \wedge a_2$	$\ldots$	$b_1 \wedge a_{k-1}$	$b_1 \wedge a_k = b_1$

Figure 3: Original and shorted paths from a_1 to b_1 over $y = b_l = a_k$.

Note that all these nodes are in G_F as Horn-formulas are closed under conjunction and the overall sum of nodes in this sequence is the same as in the original path. But as the first half is the conjunction of a_1 with every node in the second half, every variable which gets increased in both halfs (0 in a_1) will lead to two identical consecutive nodes in the first half. By symmetry, the

same happens in the new second half. This path is now two nodes shorter for every variable which was changed in both paths. All remaining flips are still present. □

Figure 4 gives a minimal example (with repeated y node in the middle) which illustrates how an increasing/decreasing path can be transformed to a shortest path of the same length as the Hamming distance between the source and target nodes. The original path has length 6 with one common variable in both halfs while the shortcut has length 4, which is optimal.

Long path	$a = 11000$	11010	11011	11111	11111	01111	00111	$00011 = b$
Optimal path	$a = 11000$	01000	00000	00000	00000	00010	00011	$00011 = b$

Figure 4: Finding shortcuts in Horn solution graphs.

This result shows that Horn solution graphs encode for every locally maximal solution y a partial cube $[y]_F$ and every intersection of two such partial cubes $[y]_F \cap [y']_F = [z]_F$ is also a partial cube. We point out that a similar statement holds for dual-Horn-formulas.

5 *Iso(2CNF)* and the number of perfect matchings

We showed in Theorem 4 that *Iso(2CNF)* $\in$ C$_=$P. In this section we show that *Iso(2CNF)* is also hard for C$_=$P. For this we will consider several reductions involving the following decision problems:

*Same*2*SAT*: Given two 2CNF-formulas F and F', does the number of satisfying assignments for F and F' coincide?

SamePM: Given two graphs, does the number of perfect matchings in each of the graphs coincide?

Curticapean [7] showed recently that *SamePM* is C$_=$P-complete. In a series of reductions, Valiant [20, 21] proved that the (functional) problem of computing the permanent can be Turing reduced to computing the number of satisfying assignments of a 2CNF formula. We take ideas from these reductions to show that *SamePM* is many-one reducible to *Same*2*SAT* and to *Iso(2CNF)*.

Theorem 7. *SamePM is polynomial time many one reducible to Same*2*SAT.*

Proof. Valiant [20] gave a way to Turing reduce the problem of counting perfect matchings to the problem of counting satisfying assignments of a 2CNF formula by counting all matchings as an intermediate step.

Reducing the number of matchings (perfect or not) of a given graph B to the number of satisfying solutions of a formula is easy. We define a variable x_e for each edge e in B and for each pair of edges e, e' with a common vertex we create a clause $(\overline{x_e} \vee \overline{x_{e'}})$. If F_B is the conjunction of all these clauses, the set of satisfying assignments for F_B coincides with the set of matchings in B.

The number of perfect matchings of a graph B with n vertices can be computed from the number of all matchings in B and some derived graphs B_k. For this, let b_i be the number of matchings with exactly i unmatched nodes.

Then b_0 is the number of perfect matchings that we want to compute, while b_{n-2} is the number of edges in B. Let us define a modification B_k of B ($1 \leq k \leq n$) consisting of a copy of B and for every node u in B, k otherwise isolated nodes $u_1, \ldots, u_k$ with a single edge connecting each of them to u. Now each matching in B can be extended in B_k by matching each non-matched node of B to one of its k new neighbors. Each original matching of B with i unmatched nodes corresponds to $(k+1)^i$ matchings in B_k. The total number of matchings c_k in B_k is $\sum_{i=0}^{n} b_i \cdot (k+1)^i$. The following equation system describes the relation between matchings in B_k graphs and in B.

$$\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & 2 & 4 & \cdots & 2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & (n+1) & (n+1)^2 & \cdots & (n+1)^n \end{pmatrix} \times \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{pmatrix} = \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{pmatrix}$$

The $(n+1) \times (n+1)$ matrix V is a Vandermonde-matrix and can therefore be inverted in polynomial time. The c coefficients are numbers of matchings, that can be reduced to numbers of satisfying assignments of 2CNF formulas. The first entry of $V^{-1} \times (c_0, \ldots, c_n)^T$ is b_0, the number of perfect matchings in B that we want to compute. Given V^{-1} and 2CNF formulas $F_0, \ldots, F_n$ having respectively $c_0, \ldots, c_n$ satisfying assignments (the formulas can be created from $B_0, \ldots, B_n$ with the aforementioned reduction), b_0 can be computed as the sum and difference of c_i's multiplied by coefficients defined by V^{-1}.

If we are given two graphs B_1 and B_2, on n vertices by doing the same construction we get two sets of coefficients (c^1 and c^2) and the number of perfect matchings in B_1 and B_2 coincide if and only if the following statement holds:

$$(V_{1,1}^{-1}, \ldots, V_{1,n+1}^{-1}) \times (c_0^1, \ldots, c_n^1)^T = (V_{1,1}^{-1}, \ldots, V_{1,n+1}^{-1}) \times (c_0^2, \ldots, c_n^2)^T$$

The c coefficients in the equation can be expressed as numbers of solutions of 2CNF formulas, while the other numbers are rational numbers. Inverting the Vandermonde matrix leads to rational numbers of length at most polynomial in n. Therefore, using an appropriate factor, we can multiply both sides of this equation by the same factor and reduce every rational number to an integer of polynomial length. This equation can be transformed so that both sides contain only additions and multiplications of positive numbers. These can be implemented as numbers of satisfying assignments of 2CNF formulas using the following gadgets. Note that input formulas are all anti-monotone and therefore have the satisfying solution 0^n and we maintain this solution through all constructions.

Multiplying the number of satisfying assignments of 2CNF-formulas can be achieved by the conjunction of both formulas (with disjoint sets of variables).

The sum of the solution sets is again a conjunction of both formulas (with disjoint sets of variables) with the following modification: For two fixed satisfying assignments 0^n of F and 0^m of F', we add the clauses $\bigwedge_{i \in [n], j \in [m]} (x_i \to \overline{y_j})$. So for every solution $v' \neq 0^n$ in F, the variables of F' get fixed to 0^m. By symmetry the same holds for all $v' \neq 0^m$ satisfying F'. This corresponds to the disjoint union of the solution sets except for 0^{n+m} which occurs only once. So we add a new variable b and add the clauses $\bigwedge_{i \in [n]} (x_i \to \overline{b}) \wedge \bigwedge_{j \in [m]} (y_j \to \overline{b})$ in the same way as before. This duplicates 0^{n+m} as b is allowed to be 1 or 0 but if we

deviate from this assignment, we fix b to 0. The number of satisfying solutions is therefore the sum of F and F' and 0^{n+m+1} is still a satisfying solution.

For encoding the coefficients of the inverse Vandermonde matrix we need a way to transform a positive integer k into a 2CNF-formula G with exactly k satisfying solutions. This can be achieved by looking at the binary encoding of $k = (k_1, \ldots, k_l)_2$. For every i with $k_i = 1$ we create the 2CNF formula $G_i = \bigwedge_{j \in [i]} (x_i \vee \overline{x_i})$ on i variables and take the sum of all these formulas (as described before) where every formula has its own set of variables and contains 0^i as satisfying solution. This new formula G has exactly k satisfying solutions.

We form for both sides of the equation 2CNF-formulas implementing these computations, and get two formulas F, F' that have the same number of satisfying assignments iff B_1 and B_2 have the same number of perfect matchings. □

Theorem 8. *Same2SAT is polynomial time many-one reducible to Iso(2CNF).*

Proof. Two formulas having only isolated satisfying assignments have the same number of solutions if and only if their solution graphs are isomorphic. A formula F can be transformed into another one F' with the same number of solutions but having only isolated satisfying assignments. This can be done by duplicating each occurring variable x with a new variable x' and adding the restriction $(x \leftrightarrow x')$. The Hamming distance between two solutions in F' is then at least two. □

These reductions plus Theorem 4 imply:

Corollary 3. *Same2SAT and Iso(2CNF) are $C_=P$-complete.*

Corollary 4. *Iso(Horn) and Iso(safely tight) are hard for $C_=P$.*

This last result follows from the observation that all constructed 2CNF formulas, those for counting matchings as well as those for multiplication and summation constructions are also Horn.

6 Conclusions and Open Problems

We studied the isomorphism problem for solution graphs for the different types of Boolean formulas defined in [9, 19]. Although it is not clear how our results can have a direct application in the developement of SAT-solvers, we believe that it is worth exploring the structure of such graphs and its relationship to the formula structure. For example, all connected components of solution graphs of CPSS functions have a nice structure since they are partial cubes of small isometric dimension [18]. We showed that this implies that isomorphism for solution graphs of CPSS formulas (a class that includes 2CNF formulas) can be reduced to counting. It is open whether other formula classes provide other properties which can be exploited for isomorphism. The class of Horn formulas is such a candidate as we showed that their solution graphs have an interesting structure. We also proved that several natural problems like *Iso(2CNF)*, *Iso(CPSS)* and the problem to decide whether two 2CNF formulas have the same number of solutions are complete for $C_=P$, a class that did not have natural complete problems other than the standard counting versions of NP-complete problems. We achieved better lower bounds only for general Boolean functions. Our classification results

are summarized in the table in Figure 1. The natural open questions would be to match upper and lower bounds in all the function types, testing whether for the case of isomorphism, there is also a dichotomy (trichotomy) result with respect to the formula structure as in the cases of satisfiability and connectivity.

References

[1] Dimitris Achlioptas, Amin Coja-Oghlan, and Federico Ricci-Tersenghi. On the solution-space geometry of random constraint satisfaction problems. *Random Struct. Algorithms*, 38(3):251–268, May 2011.

[2] Manindra Agrawal and Thomas Thierauf. The Boolean isomorphism problem. In *Proceedings of 37th Conference on Foundations of Computer Science*, pages 422–430. IEEE Comput. Soc. Press, 1996.

[3] László Babai. Graph Isomorphism in Quasipolynomial Time. In *Proceedings of 48th Annual Symposium on the Theory of Computing, STOC*, 2016.

[4] H.-J. Bandelt and M. van de Vel. Embedding Topological Median Algebras in Products of Dendrons. *Proceedings of the London Mathematical Society*, s3-58(3):439–453, May 1989.

[5] Hans-Jurgen Bandelt and Victor Chepoi. Metric graph theory and geometry: a survey. *Contemporary Mathematics*, 453:49–86, 2008.

[6] Elmar Böhler, Edith Hemaspaandra, Steffen Reith, and Heribert Vollmer. Equivalence and isomorphism for boolean constraint satisfaction. In Julian Bradfield, editor, *Computer Science Logic*, volume 2471 of *Lecture Notes in Computer Science*, pages 412–426. Springer Berlin Heidelberg, 2002.

[7] Radu Curticapean. Parity separation: A scientifically proven method for permanent weight loss. *arXiv preprint arXiv:1511.07480*, 2015.

[8] Oliver Gableske. *SAT Solving with Message Passing.* PhD thesis, University of Ulm, 2016.

[9] Parikshit Gopalan, Phokion G. Kolaitis, Elitza Maneva, and Christos H. Papadimitriou. The Connectivity of Boolean Satisfiability: Computational and Structural Dichotomies. *SIAM Journal on Computing*, 38(6):2330–2355, January 2009.

[10] Frederic Green. On the power of deterministic reductions to C=P. *Mathematical Systems Theory*, 26(2):215–233, June 1993.

[11] Birgit Jenner, Johannes Köbler, Pierre McKenzie, and Jacobo Torán. Completeness results for graph isomorphism. *Journal of Computer and System Sciences*, 66(3):549–566, May 2003.

[12] Johannes Köbler, Uwe Schöning, and Jacobo Torán. Graph isomorphism is low for PP. *Computational Complexity*, 2(4):301–330, December 1992.

[13] Johannes Köbler, Uwe Schöning, and Jacobo Torán. *The graph isomorphism problem: its structural complexity.* Birkhauser Verlag Basel, aug 1993.

[14] Eugene M. Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. *Journal of Computer and System Sciences*, 25(1):42–65, August 1982.

[15] Marc Mézard, Thierry Mora, and Riccardo Zecchina. Clustering of solutions in the random satisfiability problem. *Physical Review Letters*, 94(19):197205, 2005.

[16] Sergei Ovchinnikov. *Graphs and Cubes*. Universitext, Springer, 2011.

[17] Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the tenth annual ACM symposium on Theory of computing - STOC '78*, pages 216–226, New York, New York, USA, May 1978. ACM Press.

[18] Patrick Scharpfenecker. On the structure of solution-graphs for boolean formulas. In *Fundamentals of Computation Theory - 20th International Symposium, FCT 2015, Gdańsk, Poland, August 17-19, 2015, Proceedings*, pages 118–130, 2015.

[19] Konrad W. Schwerdtfeger. A Computational Trichotomy for Connectivity of Boolean Satisfiability. page 24, December 2013.

[20] Leslie G. Valiant. The Complexity of Enumeration and Reliability Problems. *SIAM Journal on Computing*, 8:410–421, July 1979.

[21] L.G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, January 1979.

[22] Klaus W. Wagner. The complexity of combinatorial problems with succinct input representation. *Acta Informatica*, 23(3):325–356.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de