Journal on Satisfiability, Boolean Modeling, and Computation 10 (2019) 37-58

Solution-Graphs of Boolean Formulas and Isomorphism*

Patrick Scharpfenecker† patrick.scharpfenecker@uni-ulm.de
Jacobo Torán jacobo.toran@uni-ulm.de
University of Ulm
Institute of Theoretical Computer Science
Ulm, Germany

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.

In this paper we study the complexity of the isomorphism problem of solution-graphs of Boolean formulas. We consider the classes of formulas that arise in the CSP-setting and investigate how the complexity of the isomorphism problem depends on the formula type.

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 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.

KEYWORDS: *Solution-Graph, Isomorphism, Counting, Partial Cube*

Submitted March 2017; revised January 2019; published July 2019

1. Introduction

Schaefer provided in [18] a well known dichotomy result for the complexity of the satisfiability problem on different classes of Boolean formulas. He showed that for formulas constructed from specific Boolean relations (now called Schaefer relations), 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 [10, 21] uncovered a similar behaviour 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

∗ A preliminary version of this paper appeared in the conference SAT 2016 [20].
† Supported by DFG grant TO 200/3-1.

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 [16, 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. [9]). The motivation behind the works of [10] and [21] 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 relations, the authors in [10], were able to prove a dichotomy result for the *st*-connectivity problem and Schwerdtfeger proved a trichotomy result for connectivity [21]. 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.

Schwerdtfeger [21] resolved some problems in the work of Gopalan [10] by restricting the definitions of some sets of relations to include a *safely componentwise* restriction. Such a restriction added to a property requires the property to hold even after identifying arbitrary variables, yielding a smaller set of relations satisfying this additional property. As these new definitions were required for proving the mentioned trichotomy, we will use the same definitions as Schwerdtfeger, namely *safely tight* and *constraint-projection separation Schaefer (CPSS)* sets of relations, for our results.

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 formulas, 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 [14]), 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 completely characterise 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 assignments. This result also holds for the more general class of CPSS formulas (definitions in the preliminaries section) introduced by Schwerdtfeger [21], showing that for this class of formulas isomorphism and counting have the same complexity. For the upper bound we use a recent result on the isometric dimension of partial cubes [19], the fact that GI is low for the class $C_{=}P$ [13], as well as the closure of this class under universal quantification [11]. The hardness property uses a result of Curticapean [8], 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), the isomorphism problem of CPSS solution-graphs, are complete for $C_=P$.

For the other types of formulas used in [10, 21], built from Schaefer, safely tight and general relations, we observe that the corresponding solution-graph isomorphism problems can be solved in EXP, thus improving the trivial NEXP upper bound.

For classes of relations 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.

Table 1 summarises the complexity results for isomorphism of solution-graphs for specific classes of formulas. The hardness lower bound means that there is some family of Boolean formulas in the corresponding class for which the hardness result holds. Lower bounds from classes of formulas can of course be transferred to super-classes of formulas while the other direction works for upper bounds.

Table 1. Classification of Isomorphism problems.

Relations	**Lower bound**	**Upper bound**
CPSS	$C_=P$ (Theorem 8)	$C_=P$ (Theorem 4)
Schaefer, not CPSS	$C_=P$ (Corollary 4)	EXP (Theorem 1)
safely tight, not Schaefer	$C_=P$ (Corollary 4)	EXP (Theorem 1)
not safely tight	PSPACE (Theorem 5)	EXP (Theorem 1)

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. This means that in the graph induced by such set of solutions the minimum distance between any two vertices coincides with the Hamming distance between the corresponding solutions. Therefore a solution-graph can be seen as taking a partial cube for every locally maximal solution and gluing them together.

While there is no direct connection between the isomorphism problem for solution-graphs and SAT-solving methods, the study of isomorphism questions provides new insights on the structure of solution-graphs and on the number of satisfying assignments for certain formula classes that might be useful in further SAT related research.

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.

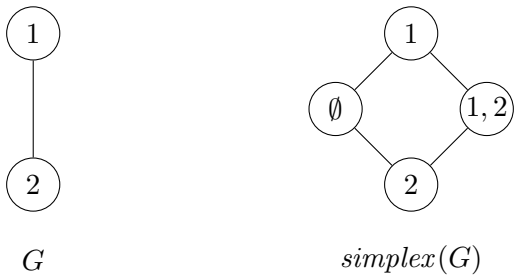

Figure 1. A graph G and its simplex graph $simplex(G)$.

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. [5]) 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 if and only if 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, $|V'| = |V| + |E| + 1$. The graph G' contains all original vertices V, a vertex $u = \{i, j\}$ for every edge $\{i, j\} \in E$ which is connected to $\{i\}$ and $\{j\}$ and a new vertex $o = \emptyset$ which is connected to all original vertices. Figure 1 gives an example for a graph G and its simplex graph $simplex(G)$.

Two graphs $G = (V, E)$ and $H = (V', E')$ with $V = V' = [n]$ are isomorphic if and only if there is a bijection $\pi\colon V \to V'$ such that for all $u, v \in V : \{u, v\} \in E \iff \{\pi(u), \pi(v)\} \in E'$. 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 signed permutation[1.] π of the n variables such that for all $x \in \{0,1\}^n$, $F(x_1, \ldots, x_n) = G(\pi(x_1), \ldots, \pi(x_n))$. For example, for the two formulas $F(x_1) = x_1$ and $G(x_1) = \neg x_1$ there exists the Boolean isomorphism $\pi : x_1 \to \neg x_1$.

We deal with different classes of formulas. 2CNF denotes the class of formulas in conjunctive normal form and with exactly two literals per clause[2.]. For a 2CNF formula $F(x_1, \ldots, x_n)$ we define the directed implication graph $I(F) = (V, E)$ on vertices $V = \{x_1, \ldots, x_n, \overline{x_1}, \ldots, \overline{x_n}\}$ and edges $(k, l) \in E$ with $k, l \in V$ if and only if there is no solution to F which falsifies the clause $(k \to l)$. This coincides with the implication graph considered by Aspvall, Plass and Tarjan in [3], after considering all possible 2-clauses that are logically implied by F. 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 $I(F) = RI(F)$. Observe that the computation of $I(F)$ and $RI(F)$ can be done in polynomial time in $|F|$ since the number of possible 2-clauses is quadratic in the number of variables.

1. A signed permutations may map variables to variables and may flip variables.
2. These formulas are also known as Krom formulas.

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 $C_=P$ [24]. This consists of the class of problems A for which there is a non-deterministic 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 $C_=P$ is ExactSAT: given a Boolean formula F and a number k, does F have exactly k satisfying assignments?

The class $C_=P$ is closely related to the class #P as $C_=P$ contains for every function problem in #P (compute the number of satisfying solutions) a related decision problem (decide if the number of satisfying solutions is exactly k for some $k \in \mathbb{N}$).

2.1 Solution-graphs of Boolean formulas

Intuitively, a solution-graph for a given Boolean formula is the induced subgraph on all satisfying solutions 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 $E = \{\{u,v\} \mid \Delta(u,v) = 1\}$.

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 if and only if their Hamming distance is one. For a set of Boolean formulas D (for example $D = 2\text{CNF}$), $\text{Iso}(D)$ denotes the isomorphism problem on the class of solution-graphs of D-formulas.

Furthermore, observe that we can apply both isomorphism concepts to solution-graphs. A Boolean isomorphism of two solution-graphs encoded by formulas F, F' on variables $x_1,\ldots,x_n$ asks for a (signed) permutation π of the variables such that for all $x \in \{0,1\}^n$, $F(x_1,x_2,\ldots,x_n) = 1$ if and only if $F'(x_{\pi(1)},x_{\pi(2)},\ldots,x_{\pi(n)}) = 1$. A general isomorphism asks for a bijection between the sets of vertices of the two solution-graphs encoded by F and F'. Observe that every Boolean isomorphism is an isomorphism between the two solution-graphs. The other direction may not be true. This may be the case for solution graphs which are not partial cubes or even for solution graphs which are the disjoint union of several partial cubes as every such connected component may require a different Boolean isomorphism for a bijection to another solution graph. Boolean isomorphisms are therefore more restricted than general isomorphisms of solution-graphs.

Given a graph G and two vertices u, v, $d_G(u,v)$ is the length of the shortest path between u and v in G or ∞ if there is no such path. When the graph G is clear from the context we will just write $d(u,v)$.

Observe that in an induced subgraph G of H_n, for every pair of vertices x, y in G, $d_G(x,y) \geq \Delta(u,v)$.

Definition 2 *An induced subgraph G of H_n is a partial cube if and only if for all $x, y \in G$, $d(x,y) = \Delta(x,y)$. We call such an induced subgraph "isometric".*

Definition 3 *The isometric dimension of a graph G is the smallest n such that G embeds isometrically into H_n or ∞ if no such embedding exists for any n.*

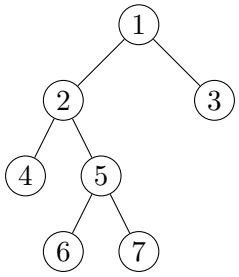

Figure 2. Example of a median graph: the median of vertices 3, 4 and 7 is the vertex 1.

Definition 4 *An undirected graph $G = (V, E)$ is a median graph if and only if for all vertices $u, v, w \in V$ there is a unique $b \in V$ which lies on the shortest paths between (u, v), (u, w) and (v, w). Then b is called the median of u, v and w.*

Figure 2 gives an example of a median graph. While this is only one example of a tree which is a median graph, it is known that all trees are median graphs.

Any Boolean function $F\colon \{0,1\}^n \to \{0,1\}$ can be represented by the subset of all its satisfying assignments in $\{0,1\}^n$ as an n-ary Boolean relation. An n-ary Boolean relation $F \subseteq \{0,1\}^n$ is closed under a ternary operation $\odot\colon \{0,1\}^3 \to \{0,1\}$ if and only if $\forall x, y, z \in F\colon \odot(x, y, z) := (\odot(x_1, y_1, z_1), \ldots, \odot(x_n, y_n, z_n)) \in F$. 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 a set of Boolean relations R with arbitrary arities (for example, $R = \{(\overline{x} \vee y), (x \oplus y), (x \oplus y \oplus z)\}$), we define $\mathrm{SAT}(R)$ to be the satisfiability problem for all Boolean formulas which are conjunctions of instantiations of relations 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-relation. Similarly, $\mathrm{Conn}(R)$ ($\mathrm{stConn}(R)$) is the following connectivity (reachability) problem: given a conjunction F of R-relations (and s, t), is the solution-graph connected (is there a path from s to t)? We mostly use F for Boolean formulas/functions/relations and R, S for sets of Boolean relations.

Note that $r \in R$ can be an arbitrary Boolean relation as for example $r = (x \oplus y)$ or $r = (x \vee \overline{y} \vee \overline{z}) \wedge (\overline{x} \vee z)$. With Horn_k we define the set of all Horn clauses, that is, the set of clauses with at most one positive literal, of size up to $k \in \mathbb{N}$. The ternary majority function $maj\colon \{0,1\}^3 \to \{0,1\}$ 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 concepts introduced in [21] and [10].

Definition 5 *A Boolean relation 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* $a \wedge (b \vee c)$*, and*
- *IHSB+, iff it is closed under* $a \vee (b \wedge c)$*.*

A relation has such a property componentwise, if and only if every connected component in the solution-graph is closed under the corresponding operation. A relation F has the additional property "safely", if and only if the property still holds for every relation F' obtained by identification of variables[3]*. Further, observe that IHSB− relations are always Horn relations and IHSB+ relations are always dual-Horn relations while the other direction of these statements may not be true.*

Observe that IHSB− relations are always Horn relations and IHSB+ relations are always dual-Horn relations while the other direction of these statements may not be true. In the case of Horn formulas, the fact that any clause contains at most one positive literal implies that the represented relations are Horn.

Definition 6 *A finite set of relations R is Schaefer if at least one of the following conditions holds:*

1. *every relation in R is bijunctive, or*
2. *every relation in R is Horn, or*
3. *every relation in R is dual-Horn, or*
4. *every relation in R is affine.*

Based on this definition for Schaefer formulas, we define a more restricted class of formulas by replacing only the second and third conditions.

Definition 7 *A finite set of relations R is CPSS (constraint-projection separation Schaefer) if at least one of the following conditions holds:*

1. *every relation in R is bijunctive, or*
2. *every relation in R is safely componentwise IHSB−, or*
3. *every relation in R is safely componentwise IHSB+, or*
4. *every relation in R is affine.*

We assume all sets of relations to be finite. If we have a Boolean formula F which is built from a set R of CPSS relations then we say that F is CPSS. Clearly, every CPSS formula is Schaefer. We later use a bigger class of relations which is called *safely tight*. This class properly contains all Schaefer sets of relations.

Definition 8 *A set R of relations is (safely) tight if at least one of the following conditions holds:*

3. Identifying two variables corresponds to replacing one of them with the other variable.

- *every relation in R is (safely) componentwise bijunctive, or*
- *every relation in R is (safely) OR-free, or*
- *every relation in R is (safely) NAND-free.*

A relation is OR-free if we cannot derive $(x \vee y)$ by substituting variables by constants and reducing the formula. Similarly, a relation is NAND-free if we cannot derive $(\overline{x} \vee \overline{y})$ by substituting variables by constants and reducing the formula.

3. Isomorphism for solution-graphs

We now turn our attention to the isomorphism problem on solution-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 directly tanslates into a NEXP upper bound for the isomorphism of solution-graphs.

Based on the celebrated new algorithm from Babai for graph isomorphism [4] running in time $n^{\log^{O(1)}(n)}$, 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 explicitly 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 [15] suffices.

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

Proof: Luks [15] 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. 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 formulas is in $C_=P$, a subclass of PSPACE. For this, we need the following two results.

Theorem 2 ([19]) *Given a CPSS relation $F(x_1, \ldots, x_n)$, every connected component of the solution-graph of F is a partial cube of isometric dimension at most n.*

Theorem 3 ([17], Theorem 5.72) *For any two finite isomorphic partial cubes G_1 and G_2 on H_n, there is a signed permutation[4] of H_n that maps one of the partial cubes onto the other. Moreover, for any isomorphism $\alpha\colon G_1 \to G_2$, there is a signed permutation σ such that σ applied to the vertices of G_1 is exactly α.*

In other words, two partial cubes G_1 and G_2 are isomorphic if and only if there exists a (signed) permutation on the variables of these partial cubes which transforms the hypercube

4. What we call a signed permutation of H_n, Ovchinnikov [17] (Section 3.8) calls an automorphism of the cube $H(X)$ of a set X: the automorphism group of $H(X)$ (which is the same as H_n if $|X| = n$) is the semidirect product of all permutations of X and the elementary Abelian 2-group on X. In other words, the automorphism group of $H(X)$ consists of all signed permutations on $|X|$ variables.

in which G_1 is embedded into the hypercube in which G_2 is embedded while still mapping G_1 to G_2. Therefore, although partial cubes of H_n might be of size 2^n, an isomorphism between such graphs can be described by a Boolean isomorphism, which can be described by only $n\log(n)$ bits.

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

Proof: As median graphs are a restricted set of graphs, the upper bound is trivial. For the lower bound, we need 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 $\mathit{simplex}(G') \cong \mathit{simplex}(H')$ if and only if $G \cong H$.

To see this we first suppose that for two given general graphs G, H we know that $|E| \neq |V|$. This could easily be enforced in an isomorphism-preserving logarithmic space 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 if and only if G and H were isomorphic. But then $\mathit{simplex}(G') \cong \mathit{simplex}(H')$ if and only if $G \cong H$.

Finally, the upper bound follows from the observation that for any graph, its simplex graph is a median graph. □

This of course implies that the isomorphism of (explicitly given) partial cubes is already GI-complete as median graphs are partial cubes (see e.g. [17], Theorem 5.75).

Note that it is known that median graphs can be exactly embedded as a solution-graph of a reduced 2CNF formula (see for example Theorem 17 in [6]). Accordingly, Lemma 1 gives an alternative reduction to the one given in [7] Claim 22, 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 if and only if there is a signed permutation of the variables mapping one partial cube to the other. But such a signed permutation 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 Thm. 17 in [7]) by creating a unique normal form and looking for a syntactic isomorphism of the formulas.

For 2CNF formulas this can easily be done using only logarithmic space. [7] (Theorem 23) creates for a given formula F a normal form F' by including all possible clauses of size 2 into F' which are valid under F. This can for 2CNF formulas be done in logarithmic space. By doing this for two formulas F and G to get F' and G', they observe that F has an (unsigned) Boolean isomorphism if and only if F' and G' are syntactically isomorphic (using a specific representation as graphs).

For our purposes, we observe that we can represent such a formula F' with a graph using the three vertices $x_i, +x_i, -x_i$ for all variables x_i in F'. We connect all triples $x_i, +x_i, -x_i$

with the edges $(x_i, +x_i), (x_i, -x_i)$, colour all x_i with the same colour (which is not used elsewhere) and connect for every clause $(u, v) \in F'$ the vertices representing the literals u and v (if $u = x_i$ we use $+x_i$, if $u = \neg x_i$ we use $-x_i$). By applying this construction to F' and G', yielding the graphs F^* and G^*, we observe that $F^* \cong G^*$ if and only if there is a Boolean isomorphism between F and G. All of this can be done using only logarithmic space. □

This basically tells us that even if we look at two exponentially sized, isomorphic partial cubes embedded in 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 (a single formula could represent several components). For the isomorphism of solution-graphs of CPSS relations we will show an upper bound of $\mathrm{C}_{=}\mathrm{P}$. 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 *Let F be a CPSS formula with satisfying solution x and let x^I for $I \subseteq [n]$ be another solution obtained from x by flipping all bit-positions $i \in I$ of x. If x^I and, for all $i \in I$, $x^{\{i\}}$ satisfy F, then there is a path from x to x^I.*

Proof: Let us assume w.l.o.g. that $I = [k]$ for some $k \leq n$. Obviously, for all $i \in I$, $x^{\{i\}}$ is connected to x as $d(x, x^{\{i\}}) = 1$. We show by induction on j that for all j with $1 \leq j \leq k$, $x^{[j]}$ satisfies F. As their consecutive distances are 1, the statement follows.

For $j = 1$ we know by the assumption of this theorem that $x^{[1]}$ satisfies F. For the induction step assume $x^{[j]}$ satisfies F. If F is componentwisely bijunctive, then $maj(x^I, x^{[j]}, x^{\{j+1\}}) = x^{[j+1]}$ satisfies F by its closure property. If F is not componentwise bijunctive (and therefore not affine), it is Horn and componentwise IHSB- (or the dual case). We now handle the two cases where $x^{\{j+1\}}$ flips a bit either from 1 to 0 or from 0 to 1. W.l.o.g., we assume that the order the flips occur in is such that we first only flip from 1 to 0 and then only from 0 to 1. In other words, we rename the indices of the variables such that there exists an $l \in [k]$ such that for all $j < l$, $x^{\{j\}}$ flips the j-th bit in x from 1 to 0 and for all $j \geq l$, $x^{\{j\}}$ flips the j-th bit in x from 0 to 1.

First, if $x^{\{j+1\}}$ flips the $(j+1)$-th bit in x from 1 to 0, then $x^{[j]} \wedge (x^{\{j+1\}} \vee x^I) = x^{[j+1]}$. To see this, we first note that $x^{[j+1]}$ differs from $x^{[j]}$ only by setting the $(j+1)$-th bit from 1 to 0. This will be achieved as the $(j+1)$-th bit of $x^{\{j+1\}} \vee x^I$ is 0 (this bit has been flipped by both $x^{\{j+1\}}$ and x^I). Additionally, we have to make sure that no other bit than the $(j+1)$-th bit gets flipped from 1 to 0 by the conjunction with $x^{[j]}$. But for a bit in $x^{[j]}$ to get flipped from 1 to 0 in this conjunction, it has to be 0 in $(x^{\{j+1\}} \vee x^I)$, implying this bit has to be 0 in x and in x^I, which is only the case for bits not touched by I. As such bits are already 0 in $x^{[j]}$, the conjunction does not flip these bits. Therefore $x^{[j]} \wedge (x^{\{j+1\}} \vee x^I) = x^{[j+1]}$ and the closure property implies that $x^{[j+1]}$ satisfies F.

Second, if $x^{\{j+1\}}$ flips the $(j+1)$-th bit in x from 0 to 1. Then $(x^{[j]} \vee x^{\{j+1\}})$ sets the $(j+1)$-th bit to 1 and possibly some bits which were flipped from 1 to 0 in $x^{[j]}$. But then $x^I \wedge (x^{[j]} \vee x^{\{j+1\}})$ restores these flips by setting them back to 0, except the $(j+1)$-th bit which is 1 in x^I. Therefore $x^I \wedge (x^{[j]} \vee x^{\{j+1\}}) = x^{[j+1]}$, implying that $x^{[j+1]}$ satisfies F.

The case for dual-Horn and componentwise IHSB+ follows by duality. □

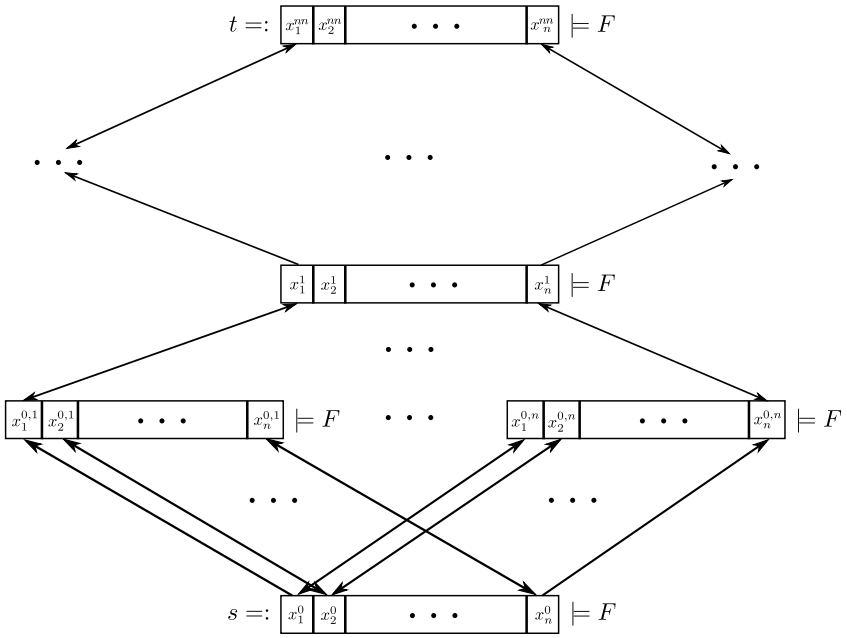

Figure 3. A walk on solution-graphs.

Lemma 3 *The following problem is reducible to* GI *in polynomial time: given two CPSS formulas F and G and satisfying solutions s of F and t of G, decide whether there is an isomorphism π between the connected components containing s and t with $\pi(s) = t$.*

Proof: We know by Theorem 3 that if two partial cubes are isomorphic, then there always is a Boolean isomorphism. One could easily guess a candidate signed permutation of variables for the Boolean isomorphism. But it is not clear how to verify that this signed permutation is in fact a Boolean isomorphism. A way to reduce this problem to GI would be to extract a single connected component and create a formula which contains only this subgraph, but in general, it is not clear how to do this. We use the construction depicted in Figure 3 to achieve a different kind of extraction which is enough in the case of isomorphism.

The purpose of this construction is to give a formula which captures the structure of the connected component C (identified by a given vertex in this component) instead of the connected component itself. Every satisfying solution of the constructed formula will represent a walk on the solution-graph, starting in the given vertex and having a fixed maximum length. If this construction is isomorphism-invariant and every vertex of C may be reached by such a walk, then the constructed formula encodes the structure of only the connected component C.

We describe this walk on a solution-graph beginning at a given vertex u in a branch-and-combine construction. We use several blocks of variables. Beginning from a vertex u in the solution-graph, we have a block of variables for every one of the n bits such that the i-th block can be assigned to u or the neighbour of u which differs from u in bit i (if

this neighbour exists). We combine these at most n vertices into a new vertex u' which accumulates all bit-flips these vertices used.

Given the original variables $x = (x_1, \ldots, x_n)$, we create new blocks of variables x^i and $x^{i,j}$ for $i \in \{0, \ldots, n\}$ and $j \in \{1, \ldots, n\}$, each containing n variables. For example, the block x^0 contains the variables $(x_1^0, \ldots, x_n^0)$ and the block $x^{0,4}$ contains the variables $(x_1^{0,4}, \ldots, x_n^{0,4})$. 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. This can be achieved by either reusing the variables which are not allowed to change or by adding a clause implementing $(x_j^0 \leftrightarrow x_j^{0,i})$ for all $i \neq j$. By not adding this constraint for $x_j^{0,j}$ with $j \in [n]$, this variable may change compared to x_j^j.

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 vertex in the solution-graph. We then add a new block x^1 such that $x_j^1 = x_j^{0,j}$ by again either reusing the variable $x_j^{0,j}$ for x_j^1 or adding a clause $(x_j^1 \leftrightarrow x_j^{0,j})$. This combines all steps of the previous branching step and we require x^1 to satisfy F.

Although all vertices visited in the branching step have distance at most 1 from x^0, the vertices described in x^1 have distance $\sum_{j\in[n]} d(x^0, x^{0,j})$ from x^0. Therefore x^1 may in principle not be in the same connected component as x^0. Lemma 2 showed that, in the case of CPSS formulas, this can never happen.

Note that the given construction on (F, s) creates a formula F' in which every satisfying solution is a walk on F of length n starting at s (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 sub-hypercube and in every step the walk may refuse to take a step and remain at the previous vertex.

Obviously, if there is an isomorphism π mapping the two components onto each other such that $\pi(s) = t$, then there is an isomorphism mapping the sets of paths onto each other. This isomorphism just has to use π for every block x^i and has to exchange the parallel steps $x^{i,0}, \ldots, x^{i,n}$ according to π. The other direction is true if we restrict the bijection between the paths to follow from a signed permutation π on the set of the original variables of F and G by applying π to all blocks x^i, $x^{i,j}$ (for $i, j \in [n]$) and interchanging $x^{i,1}, \ldots, x^{i,n}$ (for all $i \in [n]$) according to π.

We can now reduce the Boolean isomorphism question between F' and G' to GI (see Corollary 1 which uses a modified construction of [7]) implementing these additional properties with graph gadgets. Given (F, s) and (G, t), we construct the two formulas F' and G' as described earlier. We then apply the reduction of [7] and create the formulas $F^* = nf(F')$ and $G^* = nf(G')$ as normal forms of F' and G' such that there is a Boolean isomorphism between F' and G' if and only if there is a signed permutation on the variables of F^* and G^* mapping the formulas to each other. Observe that this construction does not modify the set of variables. Further, these formulas can be interpreted as graphs by using for each clause a vertex and for each variable three vertices, one for the variable and one for the positive and one for the negative literal of this variable. Additionally, there are vertices representing constants 0 and 1. We then connect the vertices representing clauses to the vertices representing the literals or constants occurring in this clause and for every variable x we connect the vertex representing this variable with the two vertices representing its

literals. [7] proved that these two graphs (we still call them F^* and G^*) are isomorphic (while mapping vertices representing variables/literals/clauses only to vertices representing variables/literals/clause) if and only if there is a Boolean isomorphism between F' and G'.

We now want to find an isomorphism mapping the formula F^* (as a graph) to the formula G^* which satisfies some additional constraints. These additional constraints can be implemented with gadgets. As there are several copies of the original variables $x_1, \ldots, x_n$ in F^* and G^*, we connect all vertices representing copies of x_k (for $k \in [n]$) with each other (using a special colour for the edges) such that they form a clique. Therefore, in all blocks of variables x^i and $x^{i,j}$ we have to apply the same permutation of the original variables $x_1, \ldots, x_n$. If we do the same for the vertices representing positive and negative occurrences of the original variables, we additionally force that every block applies the same signed permutation to the original variables. Further, we connect for every $j \in [n]$ the vertex representing the original variable x_j to all vertices representing variables in $x^{i,j}$ (for all i) with an edge using an unused colour. This forces an isomorphism to apply the permutation used on $x_1, \ldots, x_n$ to the blocks $x^{i,1}, \ldots, x^{i,n}$ (for all $i \in [n]$).

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. Finally, we replace the colours applied to edges with gadgets. The result is two graphs which are isomorphic if there is an isomorphism mapping the components rooted at s and t in F and G onto each other so that s gets mapped to t. This construction can be done in time at most polynomial in the size of the formulas F and G, so in time polynomial in $|F|$ and $|G|$. □

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

Proof: The proof uses the fact that GI is low for the class $\mathrm{C_{=}P}$ [13]. This means that a non-deterministic polynomial time algorithm with a $\mathrm{C_{=}P}$ 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 $\mathrm{C_{=}P}^{\mathrm{GI}} = \mathrm{C_{=}P}$.

We already know that the solution-graphs of CPSS relations 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 counting 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 are a connected component in one of 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 3). 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 $\forall C_{=}P^{GI}$.[5] Using the mentioned fact that GI is low for $C_{=}P$, this class coincides with $\forall C_{=}P$. In addition, Green showed [11] that $C_{=}P$ is closed under universal quantification, i.e. $\forall C_{=}P = C_{=}P$. We conclude that $\mathrm{Iso(CPSS)} \in C_{=}P$. □

In Theorem 4 we exploited the fact that CPSS formulas encode graphs which consist of partial cubes of small isometric dimension. But for general Schaefer formulas 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 $C_{=}P$-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 relations that are not safely tight.

Theorem 5 *Let S be a set of relations which is not safely tight. Then* $\mathrm{Iso}(S)$ *is hard for PSPACE under logarithmic space reductions.*

Proof: First notice that, as S is not safely tight, there has to be a relation in S which is not OR-free and there has to be a relation in S which is not NAND-free. Therefore we can apply partial assignments for these relations to construct the two relations $(x \vee y)$ and $(\overline{x} \vee \overline{y})$, implying we can use these two relations, as well as $(x \oplus y)$. We first prove the weaker result that if S is not safely tight, then $\mathrm{Iso}(S \cup \{(\overline{x} \vee y)\})$ is hard for PSPACE. Later, we argue that we can replace the usage of $(\overline{x} \vee y)$ with a relation which is definitely present in S.

The proof is based on the reduction from s,t-connectivity to GI from [12] which uses coloured graphs. We know that the s,t-connectivity problem for formulas that are not safely tight is PSPACE-complete [10]. We give a construction of solution-graphs that have coloured vertices as a way to distinguish some vertices. Later we show how the formulas can be modified to mimic the colours in their solution-graphs.

Given a formula F built on relations 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 colour vertex s in one of the copies with white and in the second copy with black. 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 colour in G_{F_1} one of the copies of vertex t with grey while in G_{F_2} the other copy of t is coloured grey. All other vertices have no colour. There is a path from s to t in G_F if and only if G_{F_1} and G_{F_2} are not isomorphic. This is because in the case of an s,t path in G_F the two white copies of the s vertex cannot be mapped to each other, while this would be possible if there is no such a path since then they would not be in the same connected components as the t vertices.

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

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

using two new variables a and b. Colouring a vertex $s \in \{0,1\}^n$ by attaching a gadget to it can be done by adding to s the graph H_m with $m > n$ as neighbour. For a given formula G and the vertex s, we construct

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

$$\begin{aligned} G'_s(x_1,\ldots,x_n,y_1,\ldots,y_m) &= G(x_1,\ldots,x_n) \\ &\wedge \bigwedge_{i\le n, s[i]=0, j\le m} (x_i \to \overline{y_j}) \\ &\wedge \bigwedge_{i\le n, s[i]=1, j\le m} (\overline{x_i} \to \overline{y_j}) \end{aligned}$$

The new graph can be described as the old solution-graph of G but now $s0^m$ is the minimal vertex of a new complete hypercube on 2^m vertices using the m new variables $y_1,\ldots,y_m$. To see this, observe that if any of the variables $x_1,\ldots,x_n$ differs from s, then $y_1,\ldots,y_m$ all have to be set to 0. But if $x_1,\ldots,x_n$ are assigned to the bit-string s, then the $n \cdot m$ new clauses (of the form $(x_i \to \overline{y_j})$ and $(\overline{x_i} \to \overline{y_j})$) are satisfied, allowing $y_1,\ldots,y_m$ to be assigned arbitrary.

Note that s is the only vertex of the original solution-graph which is part of a hypercube of dimension m. In addition, it is the only vertex of the hypercube of dimension m which is connected to some of the old vertices. Applying the colouring given above using this construction implies that $\overline{\text{Iso}(S \cup \{(\overline{x} \vee y)\})}$ is hard for PSPACE and therefore $\text{Iso}(S \cup \{(\overline{x} \vee y)\})$ is hard for coPSPACE = PSPACE.

We now show how to replace the clauses of the form $(\overline{a} \vee b)$ in this construction by using the relation $R(x,y,z) = (x \vee z) \wedge (\overline{y} \vee \overline{z}) = \{001, 100, 101, 110\}$. Observe that this relation is not equivalent to $(x \vee \overline{y})$ but similar: no satisfying assignment of R may set $xy = 01$ and for all partially satisfying assignments of xy there are either one or two possible ways to set z such that the complete assignment satisfies R.

Further, we replace the clauses $(\overline{x_i} \to \overline{y_j})$ with this relation R and use the same additional variable z for every clause involved in the creation of the same colour-gadget for both formulas. While the clause $(\overline{x_i} \to \overline{y_j})$ with satisfying solutions $\{00, 10, 11\}$ represents a solution graph which is a single path on 3 vertices, the relation R represents a solution graph which is a single path on 4 vertices. Therefore, by adding the relation $R(x,y,z)$ for existing variables x, y, we replace paths of length 3 by paths of length 4. Every vertex in the solution graph which was connected to a vertex setting xy to 10 is now connected to either a vertex setting xyz to 100 or 101. But as these two vertices are connected (they differ only by z), reachability is preserved. Moreover, no other edges are introduced except the ones connecting the copies of vertices with $xy = 10$ to their original neighbours or themselves.

So while this construction modifies the constructed graphs, we know that both graphs are modified in the same way while disconnected components still stay disconnected. Therefore this construction does not require clauses of type $(\overline{x} \vee y)$, proving our statement.

Observe that this reduction just uses copies of the input formula, adds 2 new variables and a polynomial amount of new clauses which only depend on n, m and the entries of s, t. This can be done using logarithmic space. □

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 [19] with the reduction in the proof of Theorem 5.

Corollary 2 Iso(2CNF) *is hard for NL and* $\text{Iso}(\text{Horn}_3)$ *is hard for P under logarithmic space reductions.*

4. Structure of solution-graphs of Horn formulas

While [19] 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 $\mathrm{Iso}(\mathrm{Horn}_3)$ might be easier than general solution-graph isomorphism.

Let $d_m(a,b)$ denote the monotone distance between a and b. So $d_m(a,b) < \infty$ if and only if there is a strictly monotone increasing path from a to b or vice versa. In [10, 21] it is shown that in safely OR-free formulas there is a unique minimal satisfying assignment in every connected component. As Horn formulas are safely OR-free ([10, 21]), 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, and y is the minimal satisfying assignment we could modify F setting all variables which are 1 in y to 1 and get a formula F' on less variables where $0^{n'}$ is the required minimal satisfying assignment. The resulting formula 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 vertices a between 0^n and y for which there is a monotonically 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 vertices. 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 4 illustrates in the first row the original path and in the second row the new path.

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

Figure 4. Original and shorted paths from a_1 to b_1 over $y = b_l = a_k$.

Note that all these vertices are in G_F as Horn formulas are closed under conjunction and the overall sum of vertices in this sequence is the same as in the original path. But as the first half is the conjunction of a_1 with every vertex in the second half, every variable which gets increased in both halves (0 in a_1) will lead to two identical consecutive vertices in the first half. By symmetry, the same happens in the new second half. This path is now two vertices shorter for every variable which was changed in both paths. All remaining flips are still present. □

Figure 5 gives a minimal example (with repeated y vertex 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 vertices. The original path has length 6 with one common variable in both halves while the shortcut has length 4, which is optimal.

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

Figure 5. 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.

While this seems to be a promising structure, connected components of solution-graphs encoded by Horn formulas may still have an exponential number of locally maximal solutions y, each yielding a partial cube $[y]_F$. It seems necessary to find additional structure between these locally maximal solutions to give improved upper bounds for the isomorphism problem on these graphs. Such a structure could lie in the intersection $[y]_F \cap [y']_F$ of two locally maximal solutions y and y'.

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:

Same2SAT: 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 [8] showed recently that SamePM is C$_=$P-complete. In a series of reductions, Valiant [23, 22] proved that the (functional) problem of computing the permanent of a matrix can be Turing reduced to computing the number of satisfying assignments of a 2CNF formula. This reduction queries a polynomial number of $\#SAT$ instances and uses the answers (which are numbers of polynomial length in the input matrix size) in a Chinese remainder fashion to compute the original number of perfect matchings. This argument does not work in the context of many-one reductions and decision problems because it is not clear how to combine all the queries into a single formula. We take ideas from these reductions to show that SamePM is many-one reducible to Same2SAT and to Iso(2CNF).

Theorem 7 SamePM *is polynomial time many-one reducible to* Same2SAT.

Proof: Valiant [23] gave a way to 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. We adapt some of his ideas for our result.

Reducing the number of all matchings (perfect or not perfect) 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 in some derived graphs B_k for $k \in \{1, \dots n\}$. For this, let b_i be the number of matchings with exactly i unmatched vertices in B. 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 vertex u in B, k otherwise isolated vertices $u_1, \dots, u_k$ with the additional set of edges $\{\{u, u_i\} \mid 1 \leq i \leq k\}$. Now each matching in B can be extended in B_k by matching each non-matched vertex of B to one of its k new neighbours. Each original matching of B with i unmatched vertices corresponds to $(k+1)^i$ matchings in B_k. Let c_k be the total number of matchings in B_k. $c_k = \sum_{i=0}^{n} b_i \cdot (k+1)^i$. The following equation system describes the relation between matchings in B_k graphs and in B.

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

Let us call by V the $(n+1) \times (n+1)$ matrix, and by b and c the two vectors in the equation. V is a Vandermonde matrix with in binary encoded entries of length bounded by $poly(n)$ and can therefore be inverted in polynomial time [23]. The coefficients c_i are numbers of matchings, that can be reduced to numbers of satisfying assignments of 2CNF formulas. The first entry of $V^{-1} \times (c_0, \dots, c_n)^T$ is b_0, the number of perfect matchings in B that we want to compute. Given V^{-1} and 2CNF formulas $F_0, \dots, F_n$ having respectively $c_0, \dots, c_n$ satisfying assignments (the formulas can be created from $B_0, \dots, B_n$ 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:

$$(V_{1,1}^{-1}, \dots, V_{1,n+1}^{-1}) \times (C_0^1, \dots, C_n^1)^T = (V_{1,1}^{-1}, \dots, V_{1,n+1}^{-1}) \times (C_0^2, \dots, C_n^2)^T.$$

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 reorganised 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 two 2CNF-formulas can be achieved by the conjunction of both formulas (with disjoint sets of variables).

The sum of the solution sets of two formulas F and F' 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. Observe that the number of of variables in the new formula is the sum of the variables in F and in F', plus 1 (for the new variable b).

For encoding the coefficients of the inverse Vandermonde matrix we need a way to transform a positive integer k into a 2CNF-formula H 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 $H_i = \bigwedge_{j\in[i]}(x_i \vee \overline{x_i})$ on i variables having exactly 2^i solutions and construct a formula having exactly as many solutions as the sum of solutions all these formulas (as described before). For this every formula has its own set of variables and contains 0^i as satisfying solution. This new formula H 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 if and only if 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 under polynomial many-one reductions.*

Corollary 4 Iso(*Schaefer*) *and* Iso(*safely tight*) *are hard for $C_=P$ under polynomial many-one reductions.*

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 [10, 21]. Although it is not clear how our results can have a direct application in the development 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 formulas have a nice structure since they are partial cubes of small isometric dimension [19]. 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 an open question whether other formula classes provide properties which can be exploited for testing isomorphism of their solution-graphs. 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 $C_=P$.

We achieved better lower bounds for the isomorphism problem of solution-graphs only for general Boolean formulas. Our classification results are summarised in Table 1. A natural open questions would be to match upper and lower bounds for all types of relations, 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 *37th Annual Symposium on Foundations of Computer Science, FOCS '96, Burlington, Vermont, USA, 14-16 October, 1996*, pages 422–430, 1996.

[3] Bengt Aspvall, Michael F. Plass, and Robert Endre Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Inf. Process. Lett.*, **8**(3):121–123, 1979.

[4] László Babai. Graph isomorphism in quasipolynomial time [extended abstract]. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 684–697, 2016.

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

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

[7] Elmar Böhler, Edith Hemaspaandra, Steffen Reith, and Heribert Vollmer. Equivalence and isomorphism for boolean constraint satisfaction. In *Computer Science Logic, 16th*

International Workshop, CSL 2002, 11th Annual Conference of the EACSL, Edinburgh, Scotland, UK, September 22-25, 2002, Proceedings, pages 412–426, 2002.

[8] Radu Curticapean. Parity separation: A scientifically proven method for permanent weight loss. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 47:1–47:14, 2016.

[9] Oliver Gableske. *SAT solving with message passing: a dissertation*. PhD thesis, University of Ulm, 2016.
https://www.gableske.net/downloads/gableske_thesis.pdf.

[10] Parikshit Gopalan, Phokion G. Kolaitis, Elitza N. Maneva, and Christos H. Papadimitriou. The connectivity of boolean satisfiability: Computational and structural dichotomies. *SIAM J. Comput.*, **38**(6):2330–2355, 2009.

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

[12] Birgit Jenner, Johannes Köbler, Pierre McKenzie, and Jacobo Torán. Completeness results for graph isomorphism. *J. Comput. Syst. Sci.*, **66**(3):549–566, 2003.

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

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

[15] Eugene M. Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. *J. Comput. Syst. Sci.*, **25**(1):42–65, 1982.

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

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

[18] Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing, May 1-3, 1978, San Diego, California, USA*, pages 216–226, 1978.

[19] 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.

[20] Patrick Scharpfenecker and Jacobo Torán. Solution-graphs of boolean formulas and isomorphism. In *Theory and Applications of Satisfiability Testing - SAT 2016 - 19th International Conference, Bordeaux, France, July 5-8, 2016, Proceedings*, pages 29–44, 2016.

[21] Konrad W. Schwerdtfeger. A computational trichotomy for connectivity of boolean satisfiability. *J. of Satisfiability*, **8**(3/4):173–195, 2014.

[22] Leslie G. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, **8**:189–201, 1979.

[23] Leslie G. Valiant. The complexity of enumeration and reliability problems. *SIAM J. Comput.*, **8**(3):410–421, 1979.

[24] Klaus W. Wagner. The complexity of combinatorial problems with succinct input representation. *Acta Inf.*, **23**(3):325–356, 1986.