Reprinted from: Proceedings 31st ACM Symposium on Theory of Computing, 1999

Hypergraph Isomorphism and Structural Equivalence of Boolean Functions

Eugene M. Luks
Department of Computer and Information Science
University of Oregon
Eugene, OR 97403
luks@cs.uoregon.edu

Abstract

We show that hypergraph isomorphism can be tested in time $O(c^n)$, where n is the size of the vertex set. In general, input of a hypergraph could require $\Omega(2^n)$ space, in which case the isomorphism test is in polynomial time. As a consequence, we put into polynomial time the classic problem of testing whether two Boolean functions, given by truth tables, are related via permutations and complementations of the variables, and therefore have structurally identical network realizations. In fact, the method is parallelizable and we put the problem even into NC. We obtain similarly an NC test of equivalence of truth tables under permutation of variables alone.

1 Introduction

In the synthesis and verification of Boolean circuits, it is often necessary to test whether two functions, f, g, have the same physical network. Clearly, this is the case if g is derived from f by relabelling input leads or transposing "0" and "1" states at any lead [36]. With this in mind, we say Boolean functions are *(structurally) equivalent*[1] if they are in the same orbit of the group generated by permutations and complementations of the n variables, as well as complementation of the function values [16, 18], a group of order $2^{n+1}n!$. Treating this issue at length in a 1965 text, Harrison observed [18, p. 152]:

> ... *we do not know a nonexhaustive way to* [test equivalence]

Taking a step in that direction, he offered an algorithm for computation of a system of Boolean-function invariants that had been described by Golomb [16]. Although it has practical advantage over enumeration of the group, the method requires consideration of up to $n!$ variable orderings when there is a coincidence of "first-order" invariants. In terms of the problem size[2], $m = \Omega(2^n)$, its time complexity has the same form as that of brute force, namely, $O(m^{c \log\log m})$. Although there has been much study, and implementation, of practical heuristics for special cases (e.g., [6, 18, 19, 31, 33, 39]) the question of whether there is a polynomial-time ($O(m^c)$) alternative for the general problem has remained open.

Consider also the question that arose following Goldberg's demonstration of the first subfactorial algorithm for testing graph isomorphism [15, 8]; using a combinatorial "sectioning" technique, he described a *simply-exponential*, $O(c^n)$, algorithm for n-vertex graphs. This raised the issue of whether even *hypergraph* isomorphism could be tested in similar time (see [4, Section 6]). Graph isomorphism has since been improved to *moderately-exponential* time (that is, $\exp(n^\alpha)$ with $\alpha < 1$ [40, 1], the best-known bound being $\exp(c\sqrt{n \log n})$ [3]). However, direct application of the techniques to hypergraph isomorphism achieves $O(c^n)$ timing only when the rank (= maximum cardinality of hyperedges) is bounded (see Section 7).

It is not difficult to reduce the Boolean-function question to the hypergraph question and the main results of this paper include the resolution of both (Sections 4,5).

Theorem 1.1 *For hypergraphs on n vertices, isomorphism can be tested in time $O(c^n)$, where c is constant.*

And, as a consequence,

Corollary 1.2 *Given truth tables for two Boolean functions, testing structural equivalence is in polynomial time.*

Also of some interest for a variety of applications is the matter of equivalence of Boolean functions just under permutations of variables (see, e.g., [9, 11, 19]). We show

Corollary 1.3 *Testing equivalence of truth tables under permutation of variables is in polynomial time.* □

Another application is to isomorphism-testing of bipartite graphs:

[1] This relation has also been termed *Boolean matching* or *NPN-equivalence*.

[2] In general, the functions would be specified by truth tables or the equivalent.

Corollary 1.4 *Let $\mathcal{B} = (\Phi \dot\cup \Psi, E)$, $\mathcal{B}' = (\Phi' \dot\cup \Psi', E')$ be bipartite graphs (i.e., the edge sets satisfy $E \subseteq \Phi \times \Psi$, $E' \subseteq \Phi' \times \Psi'$). Then isomorphism of $\mathcal{B}$ with $\mathcal{B}'$ can be tested in $O(c^{|\Phi|}|\Psi|^{c'})$ time (c, c' constant).*

But we can state even more in regard to Theorem 1.1. The method is strongly parallelizable (Section 6).

Theorem 1.5 *There are constants c, c' such that, using $O(c^n)$ parallel processors, isomorphism of hypergraphs on n vertices can be tested in time $O(n^{c'})$.*

Thus,

Corollary 1.6 *Testing structural equivalence of Boolean functions given by truth tables is in NC.*

Similarly, testing equivalence of truth tables under permutation of variables is in NC.

Among other things, our demonstration of Theorem 1.1 makes use of a procedure for intersecting cosets of permutation groups. Simply-exponential approaches to coset intersection were already given in [3]. However, we offer (Section 3) a self-contained version that provides the reader with the additional bonus of a succinct method for testing graph isomorphism in simply-exponential time.

While groups are employed in our procedures, it is noteworthy that their use is elementary in comparison to their involvement in several earlier applications to graph isomorphism (cf. [3], and even [25]). No knowledge of internal group structure is required: groups appear essentially as bookkeeping devices, recording, in a combinatorial divide-and-conquer, isomorphisms of substructures. In fact, one of the outcomes is a considerable simplification of methods for some problems. The aforementioned $O(c^n)$ bounded-rank hypergraph-isomorphism test required citation of the monumental simple groups classification; our more general result has no such need.

The NC results are also self-contained. Because the groups act faithfully on sets whose cardinality is logarithmic in the input size, we can avoid the deeper NC machinery of [5], which also depended on the simple groups classification.

Finally, we note that, in the interest of exposition, we make no attempt to minimize the constant c in our $O(c^n)$ algorithms. But, even with minimum c, it would not yet be prudent to claim practical applicability in testing Boolean equivalence: in applications of significant size it may already be prohibitive to represent Boolean functions by truth tables or the like. Heuristics are typically restricted to functions having compact representations (e.g., [6, 20, 21]) that are, say, polynomial in the number of variables, their applicability limited by the NP-hardness of the problem.

2 Notation and preliminaries

A *hypergraph* $\mathcal{H} = (\Sigma, E)$ consists of a set Σ together with a collection $E \subseteq 2^\Sigma$ of subsets of Σ. The *rank* of $\mathcal{H}$ is $\max_{e \in E}(|e|)$. Thus, *graphs* are hypergraphs of rank 2.

We denote by $\mathrm{Sym}(\Sigma)$ the group of all permutations of the set Σ, and for $n > 0$, $\mathrm{Sym}(n) = \mathrm{Sym}(\{1, 2, \ldots, n\})$. For $\sigma \in \Sigma$ and $x \in \mathrm{Sym}(\Sigma)$, the image of σ under the permutation x is denoted σ^x; if $\Delta \subseteq \Sigma$ then $\Delta^x = \{\delta^x \mid \delta \in \Delta\}$. For $A \subseteq \mathrm{Sym}(\Sigma)$ (A is not necessarily a subgroup), the *(set-)stabilizer* of $\Delta \subseteq \Sigma$ in A is $A_\Delta = \{x \in A \mid \Delta^x = \Delta\}$ and the *(point-)stabilizer* of $\sigma \in \Sigma$ is $A_\sigma = A_{\{\sigma\}}$. We indicate by $H \leq G$ that H is a subgroup of G. For $H \leq G$, we deal often with the set $\{Hx \mid x \in G\}$ of *right cosets* of H in G; the number of distinct right cosets is the *index* of H in G and is denoted $|G : H|$; a *right transversal* for H in G is a set $T \subseteq G$ containing precisely one element from each right coset of H. For $G \leq \mathrm{Sym}(\Sigma)$, $\sigma \in \Sigma$, the *orbit* of σ under G is $\sigma^G = \{\sigma^x \mid x \in G\}$; then $|G : G_\sigma| = |\sigma^G|$; in fact, if T is a subset of G such that, for each $\phi \in \sigma^G$, there is a unique $t \in T$ such that $\phi = \sigma^t$, then T is a right transversal for G_σ in G.

For any sets Σ and Φ, $\mathrm{Sym}(\Sigma) \times \mathrm{Sym}(\Phi)$ acts naturally on $\Sigma \times \Phi$ via $(\sigma, \phi)^{(x,y)} = (\sigma^x, \phi^y)$ for $\sigma \in \Sigma$, $\phi \in \Phi$, $x \in \mathrm{Sym}(\Sigma)$, $y \in \mathrm{Sym}(\Phi)$. Also, $\mathrm{Sym}(\Sigma)$ acts naturally on the power set 2^Σ with Δ^x for $\Delta \in 2^\Sigma$ defined as above. For any set Σ, the *diagonal* of $\Sigma \times \Sigma$ is $\mathrm{diag}(\Sigma \times \Sigma) = \{(\sigma, \sigma) \mid \sigma \in \Sigma\}$.

We require only rudimentary machinery for group-theoretic computation as given, say, in [26, Section 3]. In algorithms, groups are specified (input or output) by generators; a coset Hx is specified by generators for H along with a representative element x. Polynomial-time membership-testing is fundamental to all computation in permutation groups; this is guaranteed by Sims's method [35, 12, 26, 34]. We assume that the number of generators retained for $G \leq \mathrm{Sym}(\Sigma)$ is $O(|\Sigma|^c)$ (variations of Sims's method maintain $c = 1$, e.g., [22, 23]); thus, polynomial time for permutation groups means polynomial in $|\Sigma|$. Membership-testing, and other basic algorithms, make use of *Schreier generators* for subgroups: given generators S for G and a right transversal T for H in G, one obtains $|S||T|$ generators for the subgroup H; specifically, for each $s \in S$, $t \in T$, find $\tilde{t}$ such that $Hts = H\tilde{t}$ and thus form the Schreier generator $ts\tilde{t}^{-1}$ of H. This is particularly useful for finding point-stabilizers G_σ since T is readily obtained in a transitive-closure computation of σ^G. We note, however, that in principal applications, we apply this method to find subgroups other than G_σ: if $H \leq G$ is *any* subgroup of G for which only a membership test is available (notably, H might be a set-stabilizer in G) then we can consider the induced action of G on right cosets of H (i.e., $x \in G$ maps $Ht \mapsto Htx$) and find generators for the stabilizer of the "point" H in time that is polynomial in $|G : H|$.

On occasion, we will have generators S for $H \leq \mathrm{Sym}(\Sigma)$ along with the knowledge that $\bigcup_{t \in T} Ht$ is a coset C of some group G; in such case, we can conclude that G is generated by $S \cup \{tt_0^{-1} \mid t \in T\}$ and that $C = Gt_0$, where t_0 is any fixed element of T.

3 Coset intersection (and graph isomorphism)

We consider the following problems.

COSET_INTERSECTION.

Input: $G, H \leq \mathrm{Sym}(\Sigma)$, $x, y \in \mathrm{Sym}(\Sigma)$.

Output: $Gx \cap Hy$.

The output is either $\emptyset$ or a right coset of $G \cap H$.

GRAPH_ISOMORPHISM.

Input: *Graphs* $\mathcal{G} = (\Sigma, E)$, $\mathcal{G}' = (\Sigma', E')$

Question: *Is* $\mathcal{G}$ *isomorphic to* $\mathcal{G}'$*?*

Our main result requires a simply-exponential solution for COSET_INTERSECTION, and either of two methods outlined in [3] would suffice.[3] In fact, the method of [3, Section 10], due to Babai, only needs moderately-exponential, $\exp(n^{1/2+o(1)})$, time. Instead, we keep our discussion self-contained and present a straightforward specialization of a method, due to the present author, in [3, Section 9]. Its key role in [3] is its contribution to the $\exp(c\sqrt{n \log n})$ test for GRAPH_ISOMORPHISM. (The application of Babai's method for intersecting cosets to a graph-isomorphism test does not seem to beat $O(n!)$.) However, with only a simply-exponential goal, we do not need to get involved with the deeper aspects of that test. The approach herein has the dual advantages of simplicity and wider applicability (see [2] for another application).

Simply-exponential ($O(c^{|\Sigma|})$ COSET_INTERSECTION and GRAPH_ISOMORPHISM will both follow from a simply-exponential solution to the following problem. Recall (Section 2) the natural embedding of $\mathrm{Sym}(\Gamma) \times \mathrm{Sym}(\Delta)$ in $\mathrm{Sym}(\Gamma \times \Delta)$

Problem I.

Input: $L \leq \mathrm{Sym}(\Gamma) \times \mathrm{Sym}(\Delta)$; $z \in \mathrm{Sym}(\Gamma \times \Delta)$; $\Pi \subseteq \Gamma \times \Delta$.

Output: $(Lz)_\Pi = \{x \in Lz \mid \Pi^x = \Pi\}$.

As indicated in the proof of Proposition 3.1, the output to Problem I is again either $\emptyset$ or a right coset (of the group L_Π).

Proposition 3.1 *Problem I can be solved in* $O(c^{|\Gamma|+|\Delta|})$ *time (c constant).*

Proof. We may assume that $|\Gamma|$ and $|\Delta|$ are powers of 2 (e.g., we can augment $\Gamma \subseteq \Gamma'$ and $\Delta \subseteq \Delta'$, letting z act trivially on $\Gamma' \times \Delta' \setminus \Gamma \times \Delta$ and, for $(x, y) \in L$, letting x act trivially on $\Gamma' \setminus \Gamma$ and y act trivially on $\Delta' \setminus \Delta$; since Π is contained in $\Gamma \times \Delta$, which is stabilized by L and z, the augmentation does not change $(Lz)_\Pi$).

To accommodate recursion, we state the following more general problem

Problem II.

Input: $L \leq \mathrm{Sym}(\Gamma) \times \mathrm{Sym}(\Delta)$; $z \in \mathrm{Sym}(\Gamma \times \Delta)$; $\Pi \subseteq \Gamma \times \Delta$; $\Theta = \Phi \times \Psi \subseteq \Gamma \times \Delta$ *with* $|\Theta|$ *a power of 2 and such that* $L_\Theta = L$.

Output: $(Lz)_\Pi[\Theta] = \{x \in Lz \mid (\Pi \cap \Theta)^x = \Pi \cap \Theta^x\}$.

(Problem I is the special case $\Theta = \Gamma \times \Delta$.) Since Θ is stabilized by L, if $x, y \in Lz$ then $\Theta^x = \Theta^y$. Hence, $x, y \in (Lz)_\Pi[\Theta]$ implies $xy^{-1} \in L_\Pi[\Theta] = L_{\Pi \cap \Theta}$. Thus, the output to Problem II is either $\emptyset$ or a right coset of $L_\Pi[\Theta]$.

Since Π will be fixed throughout, we may write $(Ny)[\Theta]$ in lieu of $(Ny)_\Pi[\Theta]$ for any subcoset Ny of $\mathrm{Sym}(\Gamma \times \Delta)$.

If $|\Pi \cap \Theta| \neq |\Pi \cap \Theta^z|$, then $(Lz)[\Theta] = \emptyset$ (note that $\Theta^x = \Theta^z$ for all $x \in Lz$); if both $\Pi \cap \Theta$ and $\Pi \cap \Theta^z$ are empty, then $(Lz)[\Theta] = Lz$; hence, we may assume both are nonempty and of the same cardinality.

If $|\Pi \cap \Theta| = 1$, then the problem reduces to a standard orbit/point-stabilizer problem: suppose $\Pi \cap \Theta = \{\alpha\}$ and $\Pi \cap \Theta^z = \{\beta\}$; Let $C = \{y \in L \mid \alpha^y = \beta^{z^{-1}}\}$, so that $(Lz)[\Theta] = Cz$; now, $C \neq \emptyset$ iff $\beta^{z^{-1}} \in \alpha^L$ and, in the latter case, we take any $y \in L$ such that $\alpha^y = \beta^{z^{-1}}$ and output $C = L_\alpha y$.

We may assume $|\Pi \cap \Theta| > 1$.

If $|\Phi| > 1$, then fix $\Phi_1 \subset \Phi$ with $|\Phi_1| = |\Phi|/2$ and set $\Phi_2 = \Phi \setminus \Phi_1$, $\Theta_i = \Phi_i \times \Psi$ for $i = 1, 2$. Otherwise, fix $\Psi_1 \subset \Psi$ with $|\Psi_1| = |\Psi|/2$ and set $\Psi_2 = \Psi \setminus \Psi_1$ and $\Theta_i = \Phi \times \Psi_i$ for $i = 1, 2$.

Let $M = L_{\Theta_1}$, the subgroup of L stabilizing Θ_1. Generators for M are obtainable (see Section 2) in time that is polynomial in $|L : M|$, which is the number of images of Θ_1 under L (this number is $\leq \binom{|\Phi|}{|\Phi|/2}$ when we bisect Φ and $\leq \binom{|\Psi|}{|\Psi|/2}$ when we bisect Ψ). In similar time we can express $L = \bigcup_t Mt$. Since Θ_1 is stabilized by M, $(Mtz)[\Theta_1]$ is a coset (if not $\emptyset$) and so we can proceed recursively for each t:

$$(Mtz)[\Theta] = ((Mtz)[\Theta_1])[\Theta_2]. \tag{3.1}$$

This entails $\leq 2^{|\Phi|}$ recursive calls to subproblems with $|\Phi_i| = |\Phi|/2$ or $\leq 2^{|\Psi|}$ recursive calls to subproblems with $|\Psi_i| = |\Psi|/2$. Finally, $(Lz)[\Theta] = \bigcup_t (Mtz)[\Theta]$. Each nonempty subanswer $Mtz[\Theta]$ is a coset Nx_t where $N = M[\Theta]$, the subgroup of M stabilizing $\Pi \cap \Theta$; form the union of these cosets (as indicated in Section 2). □

Problem I subsumes both issues in the section title:

Corollary 3.2 *COSET_INTERSECTION can be solved in* $O(c^{|\Sigma|})$ *time (c constant).*

Proof. Consider the special case of Problem I in which $L = G \times H$, $z = (x, y)$, $\Gamma = \Delta = \Sigma$, and $\Pi = \mathrm{diag}(\Sigma \times \Sigma)$. For $(u, v) \in \mathrm{Sym}(\Sigma) \times \mathrm{Sym}(\Sigma)$, $\Pi^{(u,v)} = \Pi$ if and only if $u = v$. Thus, $Gx \cap Hy$ is the first (or second) coordinate projection of $(Lz)_\Pi$. □

Corollary 3.3 *GRAPH_ISOMORPHISM can be solved in time* $O(c^{|\Sigma|})$ *(c constant).*

Proof. We may assume $|\Sigma|$ and $|\Sigma'|$ are disjoint and of the same cardinality. Set $\widehat{\Sigma} = \Sigma \dot\cup \Sigma'$, $\Pi = \{(\sigma, \tau) \mid \{\sigma, \tau\} \in E \dot\cup E'\}$, and $G = \mathrm{Sym}(\widehat{\Sigma})_\Sigma$. Generators for G are easily specified ($G \simeq \mathrm{Sym}(\Sigma) \times \mathrm{Sym}(\Sigma')$). Let $t \in \mathrm{Sym}(\widehat{\Sigma})$ be a fixed permutation that transposes Σ and Σ'; then Gt is comprised of all permutations of $\widehat{\Sigma}$ that transpose Σ and Σ'. Consider the special case of Problem I in which $\Gamma = \Delta = \widehat{\Sigma}$, $L = \mathrm{diag}(G \times G)$, $z = (t, t)$, and $\Pi = \{(\sigma, \tau) \mid \{\sigma, \tau\} \in E \dot\cup E'\}$.

Any bijection $f: \Sigma \to \Sigma'$ induces a permutation $\hat{f} \in Gt$ such that $\hat{f}|_\Sigma = f$ and $\hat{f}|_{\Sigma'} = f^{-1}$. Such f is an isomorphism $\mathcal{X} \simeq \mathcal{X}'$ iff $(\hat{f}, \hat{f}) \in (Lz)_\Pi$. Conversely, if $(xt, xt) \in$

[3] Despite the title of [3], neither method requires the simple groups classification to achieve the timing needed herein.

$(Lz)_\Pi$, then $xt|_\Sigma$ induces an isomorphism $\mathcal{X} \simeq \mathcal{X}'$. Thus, by restricting to Σ the first (or second) coordinate projection of $(Lz)_\Pi$, we obtain all the isomorphisms from $\mathcal{X}$ to $\mathcal{X}'$. □

Remarks. Reductions of GRAPH_ISOMORPHISM to COSET_INTERSECTION are well-known (see, e.g., [26]). For the goal of polynomial-time reduction, it has usually been simpler to let the permutation domain consist of pairs of graph-vertices, thus seeming to square the set size. Our proof of Corollary 3.3 is just a more careful restatement of the method.

A similar observation is in order with regard to the proof of Corollary 3.2. It has been observed that COSET_INTERSECTION is polynomial-time equivalent to finding set stabilizers in permutation groups (see again [26]). Furthermore, it is easy to compute set-stabilizers in simply-exponential time: for $G \leq \mathrm{Sym}(\Sigma)$ and $\Delta \subseteq \Sigma$, one can consider the action of G on 2^Σ and so, as described in Section 2, find the stabilizer of the "point" Δ in time that is polynomial in $2^{|\Sigma|}$. However, in the straightforward conversion of COSET_INTERSECTION to a set-stabilizer problem, the size of the permutation domain is squared.

4 Hypergraph isomorphism

We consider the decision problem

HYPERGRAPH_ISOMORPHISM.

Input: *Hypergraphs* $\mathcal{H} = (\Sigma, E)$, $\mathcal{H}' = (\Sigma', E')$

Question: *Is* $\mathcal{H}$ *isomorphic to* $\mathcal{H}'$*?*

Thus, $E \subseteq 2^\Sigma$, $E' \subseteq 2^{\Sigma'}$ and we ask whether there is a bijection $\Sigma \to \Sigma'$ inducing a bijection $E \to E'$.

Our test of HYPERGRAPH_ISOMORPHISM involves the natural representation of a hypergraph as a bipartite graph, i.e., vertices on one side, hyperedges on the other, new edges recording incidence. However, the exponential number of "points" on the "hyperedge side" prohibits the use of the GRAPH_ISOMORPHISM method of Section 3. Using the divide-and-conquer just on the "vertex side," results in a simply-exponential number of base (1-element) cases, each of which translates into a HYPERGRAPH_ISOMORPHISM instance, but with only a 1-element reduction in vertex size. We avoid this combinatorial explosion by noting that, in recursions such as (3.1), we make numerous visits to the same subsets but with different cosets.[4] Essentially, we now ignore the input coset and solve a single problem at each node. In effect, this means replacing sequential calls analogous to (3.1) with independent (possibly parallel) calls followed by an intersection of the resulting cosets (and, by Section 3, we know how to do COSET_INTERSECTION). As indicated below, a *dynamic-programming* scheme keeps track of the subresults.

Testing isomorphism is reducible to finding automorphism groups by well-known methods ([27], see also proof of Corollary 1.4 below). However, our algorithm for the following will already subsume an isomorphism test.

[4] See [14] for an exploitation of a similar phenomenon in the context of low-order complexity.

HYPERGRAPH_AUTOMORPHISM.

Input: *Hypergraph* $\mathcal{H} = (\Sigma, E)$.

Output: $\mathrm{Aut}(\mathcal{H})$

Thus, we want (generators) for the subgroup of $\mathrm{Sym}(\Sigma)$ that, in its natural action on 2^Σ, stabilizes E.

The algorithm will refer to an induced collection of hypergraphs on Σ: for any $\Delta \subseteq \Sigma$, let $\mathcal{H}^\Delta = (\Sigma, E^\Delta)$ where

$$E^\Delta = \{\Theta \cap \Delta \mid \Theta \in E,\ \Theta \cup \Delta = \Sigma\}.$$

In turn, each hypergraph $\mathcal{H}^\Delta$ induces a collection of bipartite graphs: for any $\Gamma, \Delta \subseteq \Sigma$, let $\mathcal{B}^\Delta_\Gamma$ denote the bipartite graph on $\Sigma \dot\cup 2^\Sigma$ with edge set

$$\mathcal{E}(\mathcal{B}^\Delta_\Gamma) = \{(\gamma, \Phi) \mid \Phi \in E^\Delta,\ \gamma \in \Phi \cap \Gamma\}.$$

The following properties are straightforward. (For (1), note that $\Phi \in E^{\Delta \setminus \{\gamma\}}$ implies $\gamma \notin \Phi$.)

Lemma 4.1

(1) *For* $\gamma \in \Delta$, $\mathcal{E}(\mathcal{B}^\Delta_{\{\gamma\}}) = \{(\gamma, \{\gamma\} \dot\cup \Phi) \mid \Phi \in E^{\Delta \setminus \{\gamma\}}\}$.

(2) $\mathcal{E}(\mathcal{B}^\Delta_\Sigma) = \{(\sigma, \Phi) \mid \sigma \in \Phi \in E^\Delta\}$. □

Theorem 4.2 $\mathrm{Aut}(\mathcal{H})$ *can be computed in* $O(c^{|\Sigma|})$ *time (*c *constant).*

Proof. We may assume $|\Sigma|$ is a power of 2 (e.g., we can solve the problem on an augmented Σ and then cut to the subgroup that fixes the added points; alternately, this can be done with combinatorial gadgets).

We describe a dynamic-programming computation of $\mathrm{Aut}(\mathcal{H})$. This will determine, for all $\Delta, \Delta', \Gamma, \Gamma' \subseteq \Sigma$ such that $|\Delta| = |\Delta'|$ and $|\Gamma| = |\Gamma'| =$ a power of 2, the subset $\mathrm{Iso}(\Gamma, \Delta; \Gamma', \Delta')$ of $\mathrm{Sym}(\Sigma)$ mapping Γ to Γ' and inducing isomorphisms from $\mathcal{B}^\Delta_\Gamma$ to $\mathcal{B}^{\Delta'}_{\Gamma'}$ (that is, mapping $\mathcal{E}(\mathcal{B}^\Delta_\Gamma)$ to $\mathcal{E}(\mathcal{B}^{\Delta'}_{\Gamma'})$). Thus, $\mathrm{Iso}(\Gamma, \Delta; \Gamma', \Delta')$ is either $\emptyset$ or a right coset of the group $\mathrm{Iso}(\Gamma, \Delta; \Gamma, \Delta)$. A table (of size $\exp(O(|\Sigma|))$) of all these cosets is filled in order of increasing $|\Delta|$ and, for each $|\Delta|$, in order of increasing $|\Gamma|$. For $|\Delta| = 0$, $\mathrm{Iso}(\Gamma, \Delta; \Gamma', \Delta')$ consists of the permutations that map Γ to Γ' (easily specified in polynomial time for each Γ, Γ'). From the completed table, we read $\mathrm{Aut}(\mathcal{H}) = \mathrm{Iso}(\Sigma, \Sigma; \Sigma, \Sigma)$.

Computation of $\mathrm{Iso}(\Gamma, \Delta; \Gamma', \Delta')$:

For $|\Gamma| > 1$, we follow a naïve 'halving' divide-and-conquer. Fix $\Gamma_1 \subset \Gamma$ with $|\Gamma_1| = |\Gamma|/2$. Then for each $\Gamma'_1 \subset \Gamma'$ with $|\Gamma'_1| = |\Gamma'|/2$ (of course, there are $\exp(O(|\Sigma|))$ such Γ'_1), determine $\mathrm{Iso}(\Gamma_1, \Delta; \Gamma'_1, \Delta') \cap \mathrm{Iso}(\Gamma \setminus \Gamma_1, \Delta; \Gamma' \setminus \Gamma'_1, \Delta')$ (note that $\mathcal{E}(\mathcal{B}^\Delta_\Gamma) = \mathcal{E}(\mathcal{B}^\Delta_{\Gamma_1}) \dot\cup \mathcal{E}(\mathcal{B}^\Delta_{\Gamma \setminus \Gamma_1})$). The subisomorphisms are determined by table lookup; the intersection of cosets is carried out in $\exp(O(|\Sigma|))$ time by the method of Section 3. Take the union over all Γ'_1 of the results (as indicated in Section 2).

Suppose $|\Gamma| = 1$ with $\Gamma = \{\gamma\}$, $\Gamma' = \{\gamma'\}$. If one of $\mathcal{E}(\mathcal{B}^\Delta_{\{\gamma\}})$, $\mathcal{E}(\mathcal{B}^{\Delta'}_{\{\gamma'\}})$ is empty (no edges in the graph) but the other is not, then $\mathrm{Iso}(\{\gamma\}, \Delta; \{\gamma'\}, \Delta') = \emptyset$. If both are empty, then $\mathrm{Iso}(\{\gamma\}, \Delta; \{\gamma'\}, \Delta')$ is just the set of all permutations mapping γ to γ'. We may assume that there

are edges in both graphs. In particular, $\gamma \in \Delta$, $\gamma' \in \Delta'$. By Lemma 4.1(1), $\mathrm{Iso}(\{\gamma\}, \Delta; \{\gamma'\}, \Delta')$ consists of those permutations that map γ to γ' and $E^{\Delta\setminus\{\gamma\}}$ to $E^{\Delta'\setminus\{\gamma'\}}$. By Lemma 4.1(2), the permutations that map $E^{\Delta\setminus\{\gamma\}}$ to $E^{\Delta'\setminus\{\gamma'\}}$ are precisely those that induce isomorphisms from $\mathcal{B}_\Sigma^{\Delta\setminus\{\gamma\}}$ to $\mathcal{B}_\Sigma^{\Delta'\setminus\{\gamma'\}}$, i.e., $\mathrm{Iso}(\Sigma, \Delta\setminus\{\gamma\}; \Sigma, \Delta'\setminus\{\gamma'\})$, which is determined by a table lookup. The subcoset then mapping γ to γ' is then an orbit/point-stabilizer problem, solvable in polynomial-time. □

Remark. The reader who is concerned about a specific value for c in Theorem 4.2 should note that there are some easy improvements to the value implied by the above. For example, it is clear that, for values of Γ, only a linear number of subsets need be considered. Also, with some modifications, one can restrict to $\Gamma \subseteq \Delta$, $\Gamma' \subseteq \Delta'$.

Proof of Theorem 1.1. Let $\mathcal{H}_1 = (\Sigma_1, E_1)$, $\mathcal{H}_2 = (\Sigma_2, E_2)$ be given hypergraphs; we may assume Σ_1 and Σ_2 are disjoint and $|\Sigma_1| = |\Sigma_2|$ = a power of 2. Let $\mathcal{H} = (\Sigma_1 \dot\cup \Sigma_2, E_2 \dot\cup E_2)$. The isomorphisms from $\mathcal{H}_1$ to $\mathcal{H}_2$ comprise $\mathrm{Iso}(\Sigma_1, \Sigma_1; \Sigma_2, \Sigma_2)$, which is computed in the course of the above. □

5 Applications

We deal with the consequences indicated in Section 1.

Proof of Corollary 1.2. Let $\Sigma = \{\sigma_{1,0}, \sigma_{1,1}, \sigma_{2,0}, \sigma_{2,1}, \ldots \sigma_{n,0}, \sigma_{n,1}\}$, a $2n$-element set. For $x \in \mathrm{Sym}(n)$, let $x_\Sigma \in \mathrm{Sym}(\Sigma)$ be the permutation mapping $\sigma_{i,j} \mapsto \sigma_{i^x,j}$ for $1 \le i \le n$, $j = 0, 1$; let $G \le \mathrm{Sym}(\Sigma)$ be generated by $\{x_\Sigma \mid x \in \mathrm{Sym}(n)\}$ and the transpositions $\{(\sigma_{i,0}, \sigma_{i,1}) \mid 1 \le i \le n\}$.[5] A Boolean function f of n variables can be associated with the hypergraph $\mathcal{H}_f = (\Sigma, E_f)$ where $E_f = \{\{\sigma_{i,a_i}\}_{1\le i\le n} \mid (a_1, \ldots, a_n) \in \{0,1\}^n, f(a_1, \ldots, a_n) = 1\}$. Then Boolean functions f and g are structurally equivalent iff G contains an isomorphism from $\mathcal{H}_f$ to $\mathcal{H}_g$ or an isomorphism from $\mathcal{H}_f$ to $\mathcal{H}_{\bar g}$. The coset of all isomorphisms from $\mathcal{H}_f$ to $\mathcal{H}_g$, respectively, from $\mathcal{H}_f$ to $\mathcal{H}_{\bar g}$, can be found as above. An application of COSET_INTERSECTION (Section 3) finds those that are in G. Since the size of input is $\Omega(2^n)$, the polynomial-time claim follows. □

Proof of Corollary 1.3. The problem of testing equivalence under permutation of variables alone can be obtained as above by changing G accordingly, although a more direct construction follows from consideration of a truth table as a hypergraph $\mathcal{H} = (\Sigma, E)$, on the set, Σ, of variables where E consists of the characteristic sets of assignments (0, 1-vectors) mapping to TRUE. □

Remark. The hypergraph automorphism/isomorphism algorithm generalizes quite easily to "colored" hypergraphs (coloring vertices and/or coloring hyperedges and/or coloring edges in the induced bipartite graph on $\Sigma \dot\cup 2^\Sigma$). Among the applications is the generalization of Corollary 1.3 to *multi-valued logics.*

We approach Corollary 1.4 via the automorphism version of the problem.

[5] G is the *wreath product* $\mathrm{Sym}(2) \wr \mathrm{Sym}(n)$ in its *imprimitive* action. [10, ch. 2]

Corollary 5.1 *Let $\mathcal{B} = (\Phi \dot\cup \Psi, E)$ be a bipartite graph. Then* $\mathrm{Aut}(\mathcal{B})$ *can be computed in* $O(c^{|\Phi|}|\Psi|^{c'})$ *time.*

Note: We assume for now that automorphisms of $\mathcal{B}$ preserve Φ and Ψ. Such restriction can be lifted when isomorphism is resolved.

Proof. Define $f: \Psi \to 2^\Phi$ by $f(\psi) = \{\phi \in \Phi \mid (\phi, \psi) \in E\}$ and consider the hypergraph $\mathcal{H} = (\Phi, f(\Psi))$. Consider also the induced $\mathrm{Aut}(\mathcal{B})$-invariant equivalence relation R_f on Ψ where $\psi_1 R_f \psi_2$ iff $f(\psi_1) = f(\psi_2)$. It is immediate that the set P_f of equivalence classes of R_f is in 1-1 correspondence with $f(\Psi)$ and the action of $\mathrm{Aut}(\mathcal{B})$ on P_f corresponds precisely to the action of $\mathrm{Aut}(\mathcal{H})$ on $f(\Psi)$. Thus, to find generators of $\mathrm{Aut}(\mathcal{B})$: (i) we find generators S of $\mathrm{Aut}(\mathcal{H})$, taking arbitrary liftings of each to $\mathrm{Sym}(\Psi)$; (ii) we augment this collection with generators of the subgroup of $\mathrm{Sym}(\Psi)$ that stabilizes the classes (the subgroup induces the full symmetric group in each class), extending these generators to act trivially on Φ

□

Proof of Corollary 1.4. To test isomorphism of $\mathcal{B} = (\Phi \dot\cup \Psi, E)$, $\mathcal{B}' = (\Phi' \dot\cup \Psi', E')$, one can follow a familiar reduction of the problem to finding automorphism groups, e.g., reduce to the case when $\mathcal{B}, \mathcal{B}'$ are connected and then see whether any generators of the automorphism group of $((\Phi \dot\cup \Phi') \dot\cup (\Psi \dot\cup \Psi')), E \dot\cup E'))$ switch the connected components. (Alternately, one can construct a direct proof analogous to that of Corollary 5.1.) □

6 Parallelization

We describe the modifications that strengthen our polynomial-time results to NC.

First we parallelize Problem I.

Proposition 6.1 *There are constants c, c' such that, using* $O(c^{|\Gamma|+|\Delta|})$ *parallel processors, Problem I can be solved in time* $O((|\Gamma| + |\Delta|)^{c'})$.

Proof. We revisit the critical steps in the proof of Proposition 3.1.

Given L, Θ_1, we need to find $M = L_{\Theta_1}$. We consider the case where $|\Phi| > 1$ is bisected and $\Theta_1 = \Phi_1 \times \Psi$, the other case being similar. This set-stabilizer problem becomes a point-stabilizer problem by considering the action of L on the L-orbit of Θ_1. Parallel computation of orbits [29] is done by a parallel transitive-closure algorithm which assumes work at, and therefore enumeration of, all points of the permutation domain. Since, it would be prohibitive to enumerate $2^{\Phi\times\Psi}$, we determine only the orbit $\mathcal{O}$ of Φ_1 under the first coordinate action of L. For each element $\Pi \in \mathcal{O}$, the computation also returns a group element $x_\pi \in L$ such that $\Phi_1^{x_\Pi} = \Pi$. The collection $\{x_\Pi\}_{\Pi\in\mathcal{O}}$ together with a known generating set, S, enables us to write $|S||\mathcal{O}|$ *Schreier generators* (Section 2) of M. But we must take care to keep the numbers of group generators of polynomial size before proceeding to the next round, and we want to do this without resorting to the deeper machinery of [3]. The Schreier generators can be pruned to a polynomial size by a parallel application of "sifting" (see, e.g., [26, (3.3)]) through a "point-stabilizer tower" (here the "points" are again the elements of $\Gamma \times \Delta$, since the group is faithfully

represented therein); at each level, at most one generator remains to represent any coset mod the next subgroup, while others in the same coset are sifted down.

The recursive calls to problems on Mtz can be run for all t in parallel. For each, the two calls in (3.1) to problems on Θ_1 and Θ_2, respectively, have to remain serial, but these involve problems that may be considered half the size.

We need also take care to keep the numbers of group generators of manageable size when we form the union of cosets. Again, this a matter of parallel sifting.

Given the above, and observing that the depth of the recursion is only $O(\log|\Phi| + \log|\Psi|)$, we see that Problem I is solved with $O(c^{|\Gamma|+|\Delta|})$ processors in polynomial time. □

Proof of Theorem 1.5. In parallelizing the method for Theorem 1.1, generator-reduction as above is used for all subgroups visited. Coset intersections have already been parallelized. Also, for any fixed $|\Delta|$ and $|\Gamma|$, the computations of all $\mathrm{Iso}(\Gamma, \Delta; \Gamma', \Delta')$ can be run in parallel. We run through $|\Sigma|$ values of $|\Delta|$ and $\log|\Sigma|$ values of $|\Gamma|$, so the number of such "rounds" is $O(|\Sigma|\log|\Sigma|)$. In this manner, the problem is solved with $\exp(O(|\Sigma|))$ processors in polynomial time. □

Proof of Corollary 1.6. Following the construction in the proof of Corollary 1.2, The result follows from the above. □

7 Open problems

7.1 Equivalence under linear and affine transformations

A Boolean function on n variables may be viewed as a map $f: \mathrm{GF}(2)^n \to \{0,1\}$. Thus, there is an induced action on the set of such functions by the general linear group $\mathrm{GL}(n,2)$ and by the general affine linear group $\mathrm{AGL}(n,2)$. It is of some interest in both network and coding theory to determine when two functions are equivalent under the action of these groups [7, 17, 32, 37]. If one again considers the input size to be $m = 2^n$, then testing equivalence via enumeration of these groups requires $O(m^{c\log m})$ time.

> *Can equivalence of Boolean functions under the general and general-affine linear groups be tested in polynomial time?*

The question is also of interest in direct connection to GRAPH_ISOMORPHISM. Given f as above, construct a directed graph $\mathcal{G}_f = (\Sigma_f, E_f)$ with $|\Sigma_f| = \Theta(m^2)$ as follows: $\Sigma_f = \mathrm{GF}(2)^n \cup \{\{\alpha,\beta\} \mid \alpha,\beta \in \mathrm{GF}(2)^n,\ \alpha \neq \beta\}$;

$$\begin{aligned} E_f \;=\; & \{(\alpha,\{\alpha,\beta\}) \mid \alpha,\beta \in \mathrm{GF}(2)^n,\ \alpha \neq \beta\} \;\cup \\ & \{(\{\alpha,\beta\},\alpha+\beta) \mid \alpha,\beta \in \mathrm{GF}(2)^n,\ \alpha \neq \beta\} \;\cup \\ & \{(\alpha,\alpha) \mid \alpha \in \mathrm{GL}(2)^n,\ f(\alpha) = 1\}. \end{aligned}$$

Then Boolean functions f_1, f_2 are equivalent under the action of $\mathrm{GL}(n,2)$ iff $\mathcal{G}_{f_1}$ is isomorphic to $\mathcal{G}_{f_2}$. Hence, the question deals with an interesting subexponential obstruction to polynomial time for testing graph isomorphism (see [30] for questions with a similar flavor).

7.2 Canonical forms

Most advances in graph-isomorphism testing, both theoretical and practical, have depended upon, or have led to, finding canonical forms for the graphs under investigation, e.g., [4, 8, 13, 24, 28]. The method of Corollary 3.3 also falls in this class.

Similarly, much work in "Boolean matching" has involved computation of canonical elements in structural-equivalence classes, e.g., [18, 19, 33, 39]. (Limitations of the suggested techniques have been observed in [31].)

The question then arises as to whether canonical forms for hypergraphs and Boolean functions can be found within our indicated time bounds for isomorphism and equivalence.

> *Can canonical forms for hypergraphs on n vertices be found in $O(c^n)$ time?*

7.3 Bounded-rank hypergraph isomorphism

As indicated in Section 1, bounded-rank hypergraph isomorphism was known to be in $O(c^n)$ time via methods that called upon the simple groups classification. (The method makes use of a set-stabilizer algorithm outlined in [3, Section 9]: for $G \leq \mathrm{Sym}(m)$, set stabilizers can be found in time $m^{O(d/\log d)}$ provided the noncyclic composition factors of G are embeddable in $\mathrm{Sym}(d)$; for rank-r hypergraphs on an n-element vertex set, isomorphism testing is reducible to finding set stabilizers for $G \leq \mathrm{Sym}(n^r)$, but where G is actually embeddable in $\mathrm{Sym}(n)$.)

While we have extended and simplified the result, the bounded-rank case remains of interest, and we repeat a question posed in [4].

> *Can bounded-rank hypergraph-isomorphism be tested in $\exp(n^{1-\epsilon})$ time, for some $\epsilon > 0$?*

Rank-4 hypergraph isomorphism is reducible to isomorphism of graphs on $O(n^2)$ vertices. Hence, a moderately exponential rank-4 method is a necessary condition to improvement of graph-isomorphism testing to $\exp(n^{1/2-\epsilon})$.

Acknowledgment

The author thanks Nick Pippenger for a pointer to the Boolean equivalence problem including the observation that classical methods require $O(m^{c\log\log m})$ time.

References

[1] L. Babai, *Moderately exponential bound for graph isomorphism*, in: Fundamentals of Computation Theory, Lect. Notes in Math 117, Springer 1981, 34–50.

[2] L. Babai, R. Beals and P. Takácsi-Nagy, *Symmetry and complexity*, Proc. 24th ACM Symp. on Theory of Computing (1992), 438–449.

[3] L. Babai, W. M. Kantor and E. M. Luks, *Computational complexity and the classification of finite simple groups*, Proc. 24th IEEE Symp. on Foundations of Comp. Sci. (1983), 162–171.

[4] L. Babai and E.M. Luks, *Canonical labeling of graphs*, Proc. 15th ACM Symp. on Theory of Computing, 1983, 171–183.

[5] L. Babai, E. M. Luks and Á. Seress, *Permutation groups in NC*, Proc. 19th ACM Symp. on Theory of Computing, 1987, 409–420.

[6] R.E. Bryant, *Graph-based algorithms for Boolean function manipulation*, IEEE Trans. on CAD, 35, 1986, 677–318.

[7] J.D. Comerford, *Affine and general linear equivalences of Boolean functions*, Inform. and Control 45 (1980), 156–169.

[8] D. Corneil and M. Goldberg, *A non-factorial algorithm for canonical numbering of a graph*, J. Algorithms, 5 (1984), 345–362.

[9] J. Crawford, M. Ginsberg, E. M. Luks and A. Roy, *Symmetry-breaking predicates for search problems,* 5th Intern. Conf. on Principles of Knowledge Representation and Reasoning, 1996, 148–159.

[10] J.D. Dixon and B. Mortimer, Permutation Groups, Springer 1996.

[11] O. Ekin, S. Foldes, P.L. Hammer and L. Hellerstein, *Equational theories of Boolean functions*, DIMACS Tech. Rep. 97-79, 1997; to appear in Discrete Mathematics.

[12] M. Furst, J. Hopcroft and E. Luks, *Polynomial–time algorithms for permutation groups,* Proc. 21st IEEE. Symp. on Foundations of Comp. Sci., 1980, 36–41.

[13] M. Fürer, W. Schnyder and E. Specker, *Normal forms for trivalent graphs and graphs of bounded valence*, Proc. 15th ACM Symp. on Theory of Computing, 1983, 161–170.

[14] Z. Galil, C.M. Hoffmann, E.M. Luks, C.P. Schnorr and A. Weber, *An $O(n^3 \log n)$ deterministic and an $O(n^3)$ Las Vegas isomorphism test for trivalent graphs,* J. ACM, 1987, 513–531.

[15] M.K. Goldberg, *A nonfactorial algorithm for testing isomorphism of two graphs*, Disc. Appl. Math, 6 (1983), 229–236.

[16] S. Golomb, *On the classification of Boolean functions*, IRE Trans. Inform. Theory, IT-5 (1959), 176–186.

[17] M.A. Harrison, *On the classification of Boolean functions by the general linear and affine groups*, J. SIAM, 12, (1964), 285–299.

[18] M.A. Harrison, Introduction to Switching and Automata Theory, McGraw-Hill 1965.

[19] U. Hinsberger and R. Kolla, *Matching a Boolean function against a set of functions*, Tech. Rep. 185, 1997, Inst. für Informatik, Univ. Würzburg.

[20] U. Hinsberger, R. Kolla and T. Kunjan, *Approximative representation of Boolean functions by size controllable ROBDD's*, Tech. Rep. 182, 1997, Inst. für Informatik, Univ. Würzburg.

[21] J. Jain, J.R. Bitner, M.S. Abadir, J.A. Abraham and D.S. Fussell, *Indexed BDDs: algorithmic advances in techniques to represent and verify Boolean functions*, IEEE Trans. on Computers, 46 (1997) 1230–1245.

[22] M. Jerrum, *A compact representation for permutation groups,* J. Algorithms 7 (1986), 60–78.

[23] D.E. Knuth, *Notes on efficient representation of perm groups*, Combinatorica 11 (1991), 57–68.

[24] L. Kučera, *An $O(N^{1.5+\epsilon})$ expected time algorithm for canonization and isomorphism testing of trivalent graphs*, Proc. 2nd Symp. Theoretical Aspects Comp. Sci., 1985, 197–207.

[25] E. M. Luks, *Isomorphism of graphs of bounded valence can be tested in polynomial time*, J. Comp. Sys. Sci., 25 (1982) 42–65.

[26] E. M. Luks, *Permutation groups and polynomial-time computation*, in: Groups and Computation (Eds. L. Finkelstein and W. M. Kantor), Amer. Math. Soc. 1993, 139–175.

[27] R. Mathon, *A note on the graph isomorphism counting problem*, Inform. Proc. Letters, 8 (1979), 131–132.

[28] B. McKay, *Practical graph isomorphism*, Congressus Numerantium, 30, (1981), 45-87.

[29] P. McKenzie and S.A. Cook, *The parallel complexity of abelian permutation group problems,* SIAM J. Comp, 16, (1987), 880–909.

[30] G.L. Miller, *On the $n^{\log n}$ isomorphism technique*, Proc. Symp. on Theory of Computing 10 (1978), 51–58.

[31] J. Mohnke, P. Molitor and S. Malik, *Limits of using signatures for permutation independent Boolean comparison*, Proc,. IEEE/ACM Asia and South Pacific Design Automation Conference, 1995, 459–464.

[32] E.I. Nechiporuk, *On the synthesis of networks using linear transformations of variables*, Dokl. Akad. Nauk. SSSR, 123, (1958), 610–612. English translation in Automation Express (1959), 12–13.

[33] U. Schlichtmann, F. Brglez and P. Schneider. *Efficient Boolean matching based on unique variable ordering*, Proc. Intern. Workshop on Logic Synthesis, 1993, 3.6.1–3.6.6.

[34] Á. Seress, *An introduction to computational group theory*, Notices Amer. Math. Soc., 44, (1997), 671–684.

[35] C. C. Sims, *Some group-theoretic algorithms*, Lect. Notes in Math. 697, Springer 1978, 108–124.

[36] D. Slepian, *On the number of symmetry types of Boolean functions of n variables,* Canadian J. Math., 5 (1954) 185–193.

[37] D. Slepian, *Some further theory of group codes*, Bell System Tech. J., 39, (1960), 1219–1252.

[38] D.A. Spielman *Faster isomorphism testing of strongly regular graphs*, Proc. 28th Symp. on Theory of Computing, 1996, 576–584.

[39] C-C. Tsai and M. Marek-Sadowska, *Boolean functions classification via fixed polarity Reed-Muller forms*, IEEE Trans. on Computers 46, (1997), 173–186.

[40] V.M. Zemlyachenko, N.M. Kornienko and R.I. Tyshkevich, *Graph isomorphism problem*, (Russian), "The Theory of Computational Complexity I", Notes of Sci. Sem. LOMI, 118, (1982).