Isomorphism Testing of Boolean Functions Computable by Constant-Depth Circuits

V. Arvind, Yadu Vasudev
The Institute of Mathematical Sciences, Chennai

Abstract

Given two n-variable Boolean functions f and g, we study the problem of computing an *ε-approximate isomorphism* between them. I.e. a permutation π of the n variables such that $f(x_1, x_2, \ldots, x_n)$ and $g(x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)})$ differ on at most an ε fraction of all Boolean inputs $\{0,1\}^n$. We give a randomized $2^{O(\sqrt{n}\log(n/\varepsilon)^{O(d)})}$ time algorithm that computes an ε-approximate isomorphism between two isomorphic Boolean functions f and g that are given by depth d circuits of poly(n) size, where d is a constant independent of n, for any positive ε. In contrast, the best known algorithm for computing an *exact isomorphism* between n-ary Boolean functions has running time $2^{O(n)}$ [12] even for functions computed by poly(n) size DNF formulas. Our algorithm is based on a result for hypergraph isomorphism with bounded edge size [4] and the classical Linial-Mansour-Nisan result on approximating small depth and size Boolean circuits by small degree polynomials using Fourier analysis.

1 Introduction

Given two Boolean functions $f, g : \{0,1\}^n \to \{0,1\}$ the *Boolean function isomorphism* is the problem of checking if there is a permutation π of the variables such that the Boolean functions $f(x_1, x_2, \ldots, x_n)$ and $g(x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)})$ are equivalent. The functions f and g could be given as input either by Boolean circuits that compute them or simply by black-box access to them. This problem is known to be coNP-hard even when f and g are given by DNF formulas (there is an easy reduction from $\overline{\mathsf{CNFSAT}}$). The problem is in Σ_2^p but not known to be in coNP. Furthermore, Agrawal and Thierauf [1] have shown that the problem is not complete for Σ_2^p unless the polynomial hierarchy collapses to Σ_3^p.

On the other hand, the best known algorithm for Boolean function isomorphism runs in time $2^{O(n)}$ where n is the number of variables in f and g. This algorithm works even when f and g are given by only black-box access: First, the truth-tables of the functions f and g can be computed in time $2^{O(n)}$. The truth tables for f and g can be seen as hypergraphs G_f and G_g where $S \subseteq [n]$ is an edge in G_f if $f(x_S) = 1$ where x_S is the characteristic vector corresponding to S. Hypergraph Isomorphism for n-vertex and m-edge hypergraphs has a $2^{O(n)}m^{O(1)}$ algorithm due to Luks [12] which yields the claimed $2^{O(n)}$ time algorithm for testing if f and g are isomorphic. This is the current best known algorithm for general hypergraphs and hence the current fastest algorithm for Boolean function isomorphism as well. Indeed, a hypergraph on n vertices and m edges can be represented as a DNF formula on n variables with m terms. Thus, even when f and g are DNF formulas the best known isomorphism test takes $2^{O(n)}$ time. In contrast, Graph Isomorphism has a $2^{O(\sqrt{n \log n})}$ time algorithm due to Luks and Zemlyachenko (see [5]). More recently, Babai and Codenotti [4] have shown for hypergraphs of edge size bounded by k that isomorphism testing can be done in $2^{\widetilde{O}(k^2\sqrt{n})}$ time.

Our results

Since the exact isomorphism problem for Boolean functions is as hard as Hypergraph Isomorphism, and it appears difficult to improve the $2^{O(n)}$ bound, we investigate the problem of computing *approximate* isomorphisms (which we define below). An interesting question is whether the *circuit complexity* of f and g can be exploited to give a faster *approximate* isomorphism test. Specifically, in this paper we study the approximation version of Boolean function isomorphism for functions computed by *small size and small depth* circuits and give a faster algorithm for computing approximate isomorphisms. Before we explain our results we give some formal definitions.

Let $\mathcal{B}_n$ denote the set of all n-ary Boolean functions $f : \{0,1\}^n \to \{0,1\}$. Let $g : \{0,1\}^n \to \{0,1\}$ be a Boolean function and let $\pi : [n] \to [n]$ be any permutation. The Boolean function $g^\pi : \{0,1\}^n \to \{0,1\}$ obtained by applying the permutation π to the function g is defined as follows: $g^\pi(x_1, x_2, \ldots, x_n) = g(x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)})$.

This defines a (faithful) group action of the permutation group S_n on the set $\mathcal{B}_n$. I.e. $g^{(\pi\psi)} = (g^\pi)^\psi$ for all $g \in \mathcal{B}_n$ and $\pi, \psi \in S_n$, and $g^\pi = g^\psi$ for all $g \in \mathcal{B}_n$ if and only if $\pi = \psi$.

Definition 1.1. *Two Boolean functions $f, g \in \mathcal{B}_n$ are said to be isomorphic (denoted by $f \cong g$) if there exists a permutation $\pi : [n] \to [n]$ such that $\forall x \in \{0,1\}^n, f(x) = g^\pi(x)$.*

Our notion of approximate isomorphism of Boolean functions is based on the notion of *closeness* of Boolean functions which we now recall.

Definition 1.2. *Two Boolean functions f, g are $\frac{1}{2^\ell}$-close if $\Pr_{x \in \{0,1\}^n}\big[f(x) \neq g(x)\big] \leq \frac{1}{2^\ell}$.*

Definition 1.3. *Two Boolean functions f, g are $\frac{1}{2^\ell}$-approximate isomorphic if there exists a permutation $\pi : [n] \to [n]$ such that the functions f and g^π are $\frac{1}{2^\ell}$-close.*

Let $\mathcal{AC}_{s,d,n}$ denote the class of n-ary Boolean functions computed by circuits of depth d and size s, where the gates allowed are unbounded fan-in AND and

OR gates, and negation gates. The properties of $\mathcal{AC}_{s,d,n}$ circuits are well-studied in literature. Furst, Saxe and Sipser [9] proved that for any constant d, any $\mathcal{AC}_{s,d,n}$ circuit computing the parity of n variables required super-polynomial sized circuits. This was improved by Håstad [10] to obtain optimal lower bounds for computing the parity of n variables using the *switching lemma*. The approximability of Boolean functions computable by $\mathcal{AC}_{s,d,n}$ circuits by polynomials was studied in a series of papers [11, 14, 15]. In particular, we use the polynomials approximating $\mathcal{AC}_{s,d,n}$ circuits given by Linial, Mansour and Nisan, in [11], to find an approximate isomorphism.

Suppose $f, g \in \mathcal{AC}_{s,d,n}$ are isomorphic Boolean functions. As a consequence of the main result, in Section 2, we show that there is a randomized algorithm that computes an ε-approximate isomorphism between f and g in time $2^{\log(n/\varepsilon)^{O(d)}\sqrt{n}}$ for any positive ε. This is substantially faster than the $2^{O(n)}$ time algorithm for computing an exact isomorphism. We show how to achieve this running time by combining the Fourier analytic properties of Boolean functions with the Babai-Codenotti algorithm mentioned above.

Isomorphism testing of functions computable by restricted circuit classes have been studied in the literature and we point to a couple of recent works in this direction [3, 13]. In a different context, approximate Boolean function isomorphism has been studied in the framework of *property testing*, and nearly matching upper and lower bounds are known ([2, 6, 8]). In property testing the objective is to test whether two given Boolean functions are close to being isomorphic or far apart. The goal is to design a property tester with low *query* complexity. In contrast, our result is algorithmic and the goal is to efficiently compute a good approximate isomorphism.

We also note that approximate versions of Graph Isomorphism have been studied in the literature as graph edit distance, graph similarity and graph matching with respect to various distance measures (e.g. [7]). There are various heuristic algorithms for the problem. These results do not appear related to the topic of our paper.

The rest of the paper is organized as follows. In Section 2 we explain our approximate isomorphism algorithm for constant-depth small size Boolean circuits. Finally, in Section 3 we study a general problem: given two n-variable Boolean functions f and g consider the optimization problem where the objective is to find a permutation π that maximizes $|\{x \in \{0,1\}^n \mid f(x) = g^\pi(x)\}|$. We prove that this problem is coNP-hard under Turing reductions. We also give a simple $2^{O(n)}$ time deterministic approximation algorithm that, when given as input Boolean functions f and g such that f and g^π agree on a constant fraction of the inputs for some permutation π, outputs a permutation σ such that f and g^σ agree on an $O(\frac{1}{\sqrt{n}})$ fraction of the inputs.

2 Main Result

In this section we focus on the problem of computing an approximate isomorphism for two Boolean functions $f, g \in \mathcal{AC}_{s,d,n}$. We first recall the required basics from Fourier analysis of Boolean functions which is an important ingredient in our algorithm.

In the rest of the paper we will consider Boolean functions with domain $\{-1,1\}^n$ and range $\{-1,1\}$. The range $\{-1,1\}$ makes it convenient to define the Fourier basis.

The set $\mathcal{F} = \{f : \{-1,1\}^n \to \mathbb{R}\}$ of real-valued functions forms a 2^n-dimensional vector space over $\mathbb{R}$, where vector addition is defined as $(f+g)(x) = f(x)+g(x)$. The vector space $\mathcal{F}$ forms an inner product space with inner product defined as:

$$\langle f, g\rangle = \mathbb{E}_{x\in\{-1,1\}^n}\left[f(x)g(x)\right] = \frac{1}{2^n}\sum_{x\in\{-1,1\}^n} f(x)g(x).$$

The ℓ_2-norm of a function $f \in \mathcal{F}$ is $\|f\|_2 = \sqrt{\langle f, f\rangle}$. Clearly any Boolean function $f : \{-1,1\}^n \to \{-1,1\}$ has unit norm under this inner product.

The standard basis for the vector space consists of the functions $\{f_a \mid a \in \{0,1\}^n\}$ where $f_a(x) = 1$ if and only if $x = a$ and zero otherwise. For analyzing the properties of Boolean functions, it is useful to define the *Fourier basis*, which is the set $\{\chi_S \big| S \subseteq [n]\}$ defined as $\chi_S(x) = \prod_{i\in S} x_i$. It is easy to observe that $\mathbb{E}_x[\chi_S(x)] = 0$ for nonempty sets S and $\mathbb{E}_x[\chi_\emptyset(x)] = 1$. Furthermore, $\langle\chi_S, \chi_T\rangle = \mathbb{E}_x[\chi_{S\triangle T}(x)]$. It follows that the Fourier basis is an orthonormal basis with respect to the inner product. Thus, any $f \in \mathcal{F}$ can be written as $f = \sum \widehat{f}_S\chi_S$. This is the *Fourier representation* of f, and the numbers $\widehat{f}_S = \langle f, \chi_S\rangle$ are the *Fourier coefficients* of f. The orthonormality of the Fourier basis yields *Parseval's identity:* $\langle f, f\rangle = \sum_{S\subseteq[n]} \widehat{f}(S)^2$. In particular, since any Boolean function $f : \{-1,1\}^n \to \{-1.1\}$ has unit norm, we note that $\sum_{S\subseteq[n]} \widehat{f}(S)^2 = 1$.

In the next two propositions we relate the isomorphism of Boolean functions f and g to their Fourier coefficients.

Proposition 2.1. *Let $\pi : [n] \to [n]$ be any permutation, g any Boolean function and $S \subseteq [n]$, then $\widehat{g^\pi}(S) = \widehat{g}(S^\pi)$ where $S^\pi = \{i|\pi(i) \in S\}$.*

Proof. From the definition, we have

$$\begin{aligned}\widehat{g^\pi}(S) &= \frac{1}{2^n}\sum_{x\in\{-1,1\}^n} g^\pi(x)\chi_S(x)\\ &= \frac{1}{2^n}\sum_{x_1,\dots,x_n\in\{-1,1\}} g(x_{\pi(1),\dots,\pi(n)})\chi_S(x_1,\dots,x_n).\end{aligned}$$

The permutation π defines a bijection from $\{-1,1\}^n$ to itself where $\pi(b_1,\dots,b_n) = (b_{\pi(1)},\dots,b_{\pi(n)})$. Hence

$$\sum_{x_1,\dots,x_n\in\{-1,1\}} g(x_{\pi(1),\dots,\pi(n)})\chi_S(x_1,\dots,x_n) = \sum_{x_1,\dots,x_n\in\{0,1\}} g(x_1,\dots,x_n)\prod_{i\in S} x_{\pi^{-1}(i)}.$$

This completes the proof since χ_{S^π} is exactly $\prod_{i\in S} x_{\pi^{-1}(i)}$. □

Proposition 2.2. *Two Boolean functions $f, g : \{-1.1\}^n \to \{-1,1\}$ are isomorphic via permutation π if and only if $\widehat{f}(S) = \widehat{g}(S^\pi)$ for each subset S.*

Proof. Suppose $\pi : [n] \to [n]$ is an isomorphism. I.e. $f(x) = g^\pi(x)$ for all $x \in \{-1,1\}^n$. Consider any subset $S \subseteq [n]$

$$\widehat{f}(S) = \frac{1}{2^n} \sum_{x\in\{-1,1\}^n} f(x)\chi_S(x) = \frac{1}{2^n} \sum_{x\in\{-1,1\}^n} g^\pi(x)\chi_S(x) = \widehat{g^\pi}(S)$$

Since $\widehat{g^\pi}(S) = \widehat{g}(S^\pi)$ from the previous proposition, this completes the proof of the forward direction of the proposition.

Conversely, if $\widehat{f}(S) = \widehat{g}(S^\pi)$ for each subset S, again by the previous proposition we have $\widehat{f}(S) = \widehat{g^\pi}(S)$ which implies that $f = g^\pi$. □

2.1 Approximate Isomorphism for $\mathcal{AC}_{s,d,n}$

In this subsection, we prove that the problem of checking if two Boolean functions $f, g \in \mathcal{AC}_{s,d,n}$ are $1/2^\ell$-approximately isomorphic can be reduced to checking if two "low-degree" polynomials (whose degrees depend on ℓ) are isomorphic. We first outline this reduction using the Fourier spectrum of the Boolean functions f and g.

A crucial theorem that we will use is the celebrated result of Linial-Mansour-Nisan [11] which gives the distribution of Fourier coefficients for Boolean functions computed by small depth circuits.

Theorem 2.3 ([11]). *Let $f : \{-1,1\}^n \to \{-1,1\}$ be computed by an $\mathcal{AC}_{s,d,n}$ circuit. Then for all $t > 0$,*

$$\sum_{S\subseteq[n],|S|>t} \widehat{f}(S)^2 \leq 2s2^{-t^{1/d}/20}.$$

Consequently, for $\widetilde{f} = \sum_{S\subseteq[n],|S|\leq t} \widehat{f}(S)\chi_S$ we have $\|f - \widetilde{f}\|_2^2 \leq 2s2^{-t^{1/d}/20}$.

Notice that each $\chi_S = \prod_{i\in S} x_i$ is a monomial, and hence $\widetilde{f}$ is a degree-t polynomial that approximates f. Given Boolean functions $f, g \in \mathcal{AC}_{s,d,n}$ as an instance of Boolean function isomorphism, our aim is to work with the polynomials $\widetilde{f}$ and $\widetilde{g}$:

$$\widetilde{f} = \sum_{S\subseteq[n],|S|\leq t} \widehat{f}(S)\chi_S \quad \text{and} \quad \widetilde{g} = \sum_{S\subseteq[n],|S|\leq t} \widehat{g}(S)\chi_S. \tag{1}$$

This is because $\widetilde{f}$ and $\widetilde{g}$ are of degree t and have only n^t terms. I.e. we will check if there is an approximate isomorphism between $\widetilde{f}$ and $\widetilde{g}$. Since the polynomials $\widetilde{f}, \widetilde{g} : \{-1,1\}^n \to \mathbb{R}$ are no longer Boolean-valued functions, we define an appropriate notion of isomorphism here.

Definition 2.4. *Let $f', g' : \{-1,1\}^n \to \mathbb{R}$ be two functions from $\mathcal{F}$. We say that f' and g' are $\frac{1}{2^\ell}$-approximate isomorphic witnessed by a permutation $\pi : [n] \to [n]$ if $\|f' - g'^{\pi}\|_2^2 \le \frac{1}{2^\ell}$.*

Notice that this definition of $\frac{1}{2^\ell}$-approximate isomorphism for non-Boolean functions differs from the notion of $\frac{1}{2^\ell}$-approximate isomorphism for Boolean functions in Definition 1.3. The next proposition shows that if Boolean functions f and g are $\frac{1}{2^\ell}$-close then $\|f - g\|$ is small.

Proposition 2.5. *If $f, g : \{-1,1\}^n \to \{-1,1\}$ are $\frac{1}{2^\ell}$-close, then $\|f-g\|_2^2 \le 4\frac{1}{2^\ell}$*

Proof. From the definition, we have,

$$\|f - g\|_2^2 = \frac{1}{2^n} \sum_{x \in \{-1,1\}^n} (f(x) - g(x))^2.$$

Observing that $(f(x) - g(x))^2$ is 4 if $f(x) \neq g(x)$ and zero otherwise, we have

$$\|f - g\|_2^2 = \frac{4|\{x \mid f(x) \neq g(x)\}|}{2^n}$$

This completes the proof. □

We now explain the connection between $\frac{1}{2^\ell}$-approximate isomorphism of two functions and their Fourier coefficients.

Lemma 2.6. *Let f and g be two Boolean functions that are $\frac{1}{2^\ell}$-approximate isomorphic via permutation $\pi : [n] \to [n]$. Then $\forall S \subseteq [n]$: $\left|\widehat{f}(S) - \widehat{g^\pi}(S)\right| \le \frac{2}{2^{\ell/2}}$.*

Proof. Notice that $\sum_{S \subseteq [n]} (\widehat{f}(S) - \widehat{g^\pi}(S))^2 = \|f - g^\pi\|_2^2$. Suppose f, g are $\frac{1}{2^\ell}$-approximate isomorphic via permutation π. By Proposition 2.5 we know that $\sum_{S \subseteq [n]} (\widehat{f}(S) - \widehat{g^\pi}(S))^2 = \|f - g^\pi\|_2^2 \le \frac{4}{2^\ell}$. Hence for each subset $S \subseteq [n]$ we have $\left(\widehat{f}(S) - \widehat{g^\pi}(S)\right)^2 \le \frac{4}{2^\ell}$. □

Let f and g be two Boolean functions that are $\frac{1}{2^\ell}$-approximate isomorphic via permutation $\pi : [n] \to [n]$. By the above proposition $|\widehat{f}(S) - \widehat{g^\pi}(S)|$ is bounded by $\frac{2}{2^{\ell/2}}$. Furthermore, since both $\widehat{f}(S)$ and $\widehat{g^\pi}(S)$ are Fourier coefficients of Boolean functions f and g^π, we have $0 \le |\widehat{f}(S)| \le 1$ and $0 \le |\widehat{g^\pi}(S)| \le 1$. Hence, the bound implies that the $\lfloor \ell/2 \rfloor - 1$ most significant positions in the binary representation of $\widehat{f}(S)$ and $\widehat{g^\pi}(S)$ are identical.

For each subset S, let $\widehat{f}_\ell(S)$ denote the truncation of $\widehat{f}(S)$ to the first $\lfloor \ell/2 \rfloor - 1$ bits. Thus, $|\widehat{f}_\ell(S) - \widehat{f}(S)| \le \frac{1}{2^{\lfloor \ell/2 \rfloor - 1}}$ for each S. Similarly, $\widehat{g}_\ell(S)$ denotes the truncation of $\widehat{g}(S)$ to the first $\lfloor \ell/2 \rfloor - 1$ bits. We define the following two functions f_ℓ and g_ℓ from $\{-1,1\}^n \to \mathbb{R}$:

$$f_\ell = \sum_{S \subseteq [n]} \widehat{f}_\ell(S) \chi_S \quad \text{and} \quad g_\ell = \sum_{S \subseteq [n]} \widehat{g}_\ell(S) \chi_S. \tag{2}$$

The following lemma summarizes the above discussion and gives us a way to go from approximate isomorphism to exact isomorphism.

Lemma 2.7. *Let f and g be two Boolean functions that are $\frac{1}{2^\ell}$-approximate isomorphic via permutation $\pi : [n] \to [n]$. Then $f_\ell = g_\ell^\pi$, i.e. the functions f_ℓ and g_ℓ are (exactly) isomorphic via the permutation π.*

Lemma 2.6 and Proposition 2.5 yield the following observation.

Lemma 2.8. *Suppose f, g are two Boolean functions that are $\frac{1}{2^\ell}$-approximate isomorphic via permutation π. Then $\|\widetilde{f} - \widetilde{g^\pi}\|_2^2 \leq \frac{4}{2^\ell}$. I.e. $\widetilde{f}$ and $\widetilde{g}$ are $\frac{4}{2^\ell}$-approximate isomorphic via the same permutation π. Furthermore, $|\widehat{f}(S) - \widehat{g^\pi}(S)| \leq \frac{2}{2^{\ell/2}}$ for all $S : |S| \leq t$.*

Proof. By Lemma 2.6 and Proposition 2.5 we have $\sum_{S\subseteq[n]} \big(\widehat{f}(S) - \widehat{g^\pi}(S)\big)^2 \leq \frac{4}{2^\ell}$, which implies $\|\widetilde{f} - \widetilde{g^\pi}\|_2^2 = \sum_{|S|\leq t} \big(\widehat{f}(S) - \widehat{g^\pi}(S)\big)^2 \leq \frac{4}{2^\ell}$. It follows that $|\widehat{f}(S) - \widehat{g^\pi}(S)| \leq \frac{2}{2^{\ell/2}}$ for all $S : |S| \leq t$. □

Now, if $|\widehat{f}(S) - \widehat{g^\pi}(S)| \leq \frac{2}{2^{\ell/2}}$ for all $S : |S| \leq t$, it implies that $\widehat{f_\ell}(S) = \widehat{g_\ell^\pi}(S)$ for all $S : |S| \leq t$, where $\widehat{f_\ell}(S)$ and $\widehat{g_\ell}(S)$ are defined in Equation 2. Indeed, if we truncate the coefficients of the polynomials $\widetilde{f}$ and $\widetilde{g}$ also to the first $\lfloor \ell/2 \rfloor - 1$ bits we obtain the polynomials

$$\widetilde{f_\ell}(x_1, \ldots, x_n) = \sum_{S:|S|\leq t} \widehat{f_\ell}(S)\chi_S \quad \text{and} \quad \widetilde{g_\ell}(x_1, \ldots, x_n) = \sum_{S:|S|\leq t} \widehat{g_\ell}(S)\chi_S. \tag{3}$$

It clearly follows that π is an exact isomorphism between $\widetilde{f_\ell}$ and $\widetilde{g_\ell}$. We summarize the above discussion in the following lemma which is crucial for our algorithm.

Lemma 2.9. *Suppose f, g are two Boolean functions that are $\frac{1}{2^\ell}$-approximate isomorphic via permutation π. Then:*

1. *$\|\widetilde{f} - \widetilde{g^\pi}\|_2^2 \leq \frac{4}{2^\ell}$. I.e. $\widetilde{f}$ and $\widetilde{g}$ are $\frac{4}{2^\ell}$-approximate isomorphic via the same permutation π, and hence $|\widehat{f}(S) - \widehat{g^\pi}(S)| \leq \frac{2}{2^{\ell/2}}$ for all $S : |S| \leq t$.*

2. *Consequently, π is an exact isomorphism between the polynomials $\widetilde{f_\ell}$ and $\widetilde{g_\ell}$.*

This lemma provides a reduction of the approximate isomorphism problem for two Boolean functions f and g to an exact isomorphism problem between two polynomials $\widetilde{f_\ell}$ and $\widetilde{g_\ell}$. Now, if π is an exact isomorphism between $\widetilde{f_\ell}$ and $\widetilde{g_\ell}$ what can we infer about π as an approximate isomorphism between f and g? The following lemma quantifies it.

Lemma 2.10. *Suppose f and g are Boolean functions in $\mathcal{AC}_{s,d,n}$ such that π is an exact isomorphism between $\widetilde{f}_\ell$ and $\widetilde{g}_\ell$ (where $\widetilde{f}$ and $\widetilde{g}$ are given by Equation 1). Then π is an $(\delta+\varepsilon)^2$-approximate isomorphism between f and g, where $\delta = 2s2^{-t^{1/d}/20}$ and $\varepsilon = \frac{2n^{t/2}}{2^{(\ell-1)/2}}$.*

Proof. Since π is an exact isomorphism between $\widetilde{f}_\ell$ and $\widetilde{g}_\ell$ we have $\widetilde{f}_\ell = \widetilde{g^\pi_\ell}$. Now consider $\|f - g^\pi\|_2^2$. By triangle inequality

$$\begin{aligned}\|f-g^\pi\|_2 &\le \|f-\widetilde{f}\| + \|\widetilde{f}-\widetilde{f}_\ell\| + \|\widetilde{f}_\ell - \widetilde{g^\pi}_\ell\| + \|\widetilde{g^\pi} - \widetilde{g^\pi}_\ell\| + \|g^\pi - \widetilde{g^\pi}\| \\ &= \|f-\widetilde{f}\| + \|\widetilde{f}-\widetilde{f}_\ell\| + \|\widetilde{g^\pi} - \widetilde{g^\pi}_\ell\| + \|g^\pi - \widetilde{g^\pi}\|.\end{aligned}$$

By Theorem 2.3, both $\|f-\widetilde{f}\|$ and $\|g^\pi - \widetilde{g^\pi}\|$ are bounded by δ. Furthermore,

$$\|\widetilde{f}-\widetilde{f}_\ell\|_2^2 = \sum_{S:|S|\le t} (\widehat{f}(S) - \widehat{f}_\ell(S))^2 \le \sum_{S:|S|\le t} \frac{4}{2^{\ell-1}} \le \frac{4n^t}{2^{\ell-1}}.$$

Hence, $\|\widetilde{f}-\widetilde{f}_\ell\| \le \frac{2n^{t/2}}{2^{(\ell-1)/2}}$ and, likewise, $\|\widetilde{g^\pi} - \widetilde{g^\pi}_\ell\| \le \frac{2n^{t/2}}{2^{(\ell-1)/2}}$. Putting it together with Proposition 2.5 we get

$$4\Pr[f \ne g^\pi] = \|f-g^\pi\|_2^2 \le \left(2\delta + \frac{4n^{t/2}}{2^{(\ell-1)/2}}\right)^2.$$

It follows that f and g are $(\delta+\varepsilon)^2$-approximate isomorphic via the permutation π. ☐

Suppose f and g are in $\mathcal{AC}_{s,d,n}$ and are given by circuits C_f and C_g. Our goal now is to design an efficient algorithm that will compute the polynomials $\widetilde{f}_\ell$ and $\widetilde{g}_\ell$, where ℓ will be appropriately chosen in the analysis. In order to compute $\widetilde{f}_\ell$ and $\widetilde{g}_\ell$ we need to estimate to $\lfloor \ell/2 \rfloor - 1$ bits of precision, the Fourier coefficients $\widehat{f}(S)$ and $\widehat{g}(S)$ for each subset $S : |S| \le t$. Now, by definition, $\widehat{f}(S)$ is the average of $f(x)\chi_S(x)$ where x is uniformly distributed in $\{-1,1\}^n$. Hence, following a standard Monte-Carlo sampling procedure, we can estimate $\widehat{f}(S)$ quite accurately from a random sample of inputs from $\{-1,1\}^n$ and with high probability we can exactly compute $\widehat{f}_\ell(S)$ for all $S : |S| \le t$. We formally explain this in the next lemma.

Lemma 2.11. *Given $f : \{-1,1\}^n \to \{-1,1\}$ computed by an $\mathcal{AC}_{s,d,n}$ circuit, there is a randomized algorithm $\mathcal{C}$ with running time $\mathrm{poly}(s, n^t, 2^\ell)$ that outputs the set $\{\widehat{f}_\ell(S) \mid |S| \le t\}$ with probability $1 - \frac{1}{2^{\Omega(n)}}$.*

Proof. We use the same technique as [11] to estimate the required Fourier coefficients.

1. For each subset $S \subset [n]$ such that $|S| \le t$ do the following two steps:
2. Pick $x_i \in_r \{-1,1\}^n$ and compute the value $f(x_i)\chi_S(x_i)$ for $i \in [m]$.

3. Estimate the Fourier coefficient as $\alpha_f(S) = \frac{1}{m}\sum_{i=1}^{m} f(x_i)\chi_S(x_i)$.

Applying Chernoff bounds, for each subset S we have $\Pr\left[\left|\widehat{f}(S) - \alpha_f(S)\right| \geq \lambda\right] \leq 2e^{-\lambda^2 m/2}$. In our case we set $\lambda = \frac{1}{2^{\lfloor \ell/2 \rfloor - 1}}$. In order to estimate $\widehat{f}(S)$ for each $S : |S| \leq t$ within the prescribed accuracy and with small error probability, we set $m = tn \log n 2^{\ell}$. The entire procedure runs in $\mathrm{poly}(s, n^t, 2^{\ell})$ time. Furthermore, by a simple union bound it follows that with probability $1 - 2^{-\Omega(n)}$ we have $\alpha_f(S) = \widehat{f_\ell}(S)$ for each $S : |S| \leq t$ with high probability. Thus, the randomized algorithm computes the polynomial $\widetilde{f_\ell}$ with high probability. $\Box$

2.2 Exact isomorphism test for low degree polynomials

We now focus on the problem of checking if the polynomials $\widetilde{f_\ell} = \sum_{S:|S|\leq t} \widehat{f_\ell}(S)\chi_S$ and $\widetilde{g_\ell} = \sum_{S:|S|\leq t} \widehat{g_\ell}(S)\chi_S$ are isomorphic, and if so to compute an exact isomorphism π. To this end, we shall encode f_ℓ and g_ℓ as *weighted* hypergraphs G_f and G_g, respectively.

The vertex sets for both graphs is $[n]$. Let E denote the set of all subsets $S \subset [n]$ of size at most t. The weight functions for the edges are w_f and w_g for G_f and G_g defined as follows

$$w_f(S) = \begin{cases} \widehat{f_\ell}(S) & \forall S \subseteq [n], |S| \leq t \\ 0 & \textit{otherwise}, \end{cases} \quad \text{and} \quad w_g(S) = \begin{cases} \widehat{g_\ell}(S) & \forall S \subseteq [n], |S| \leq t \\ 0 & \textit{otherwise}. \end{cases}$$

The isomorphism problem for the polynomials f_ℓ and g_ℓ is now the edge-weighted hypergraph isomorphism problem, where G_f and G_g are the two edge-weighted graphs, and the problem is to compute a permutation on $[n]$ that maps edges to edges (preserving edge weights) and non-edges to non-edges. Our aim is to apply the Babai-Codenotti isomorphism algorithm for hypergraphs with hyperedge size bounded by k [4]. Their algorithm has running time $2^{\widetilde{O}(k^2\sqrt{n})}$. We need to adapt their algorithm to work for hypergraphs with edge weights. Since the edge weights for the graphs G_f and G_g can be encoded by $\lfloor \ell/2 \rfloor - 1$ length bit strings, we can encode the weights into the hyperedges by introducing new vertices.

More precisely, we create new graphs G'_f and G'_g corresponding to f and g, where the number of vertices is now $n + O(\ell)$. Let the set of new vertices be $\{v_1, \ldots, v_r\}$, where $r = O(\ell)$. Let $S \subset [n]$ be a hyperedge in the original graph G_f. A subset $T \subset \{v_1, \ldots, v_r\}$ encodes an r-bit string via a natural bijection (the j^{th} bit is 1 if and only if $v_j \in T$). Let $T(S) \subset \{v_1, \ldots, v_r\}$ denote the encoding of the number $\widehat{f_\ell}(S)$ for each hyperedge $S \in E$. Similarly, let $T'(S) \subset \{v_1, \ldots, v_r\}$ denote the encoding of the number $\widehat{g_\ell}(S)$. The hyperedge $S \cup T(S)$ encodes S along with its weight $\widehat{f_\ell}(S)$ for each S in G'_f. Similarly, $S \cup T'(S)$ encodes S along with its weight $\widehat{g_\ell}(S)$ for each S in G'_g. The following lemma is an easy consequence of our construction of the hypergraphs.

Lemma 2.12. *There exists a permutation* $\pi : [n] \to [n]$ *such that* $\widetilde{f_\ell} = \widetilde{g_\ell}^{\pi}$ *if* *and only if $\mu : [n+r] \to [n+r]$ defined by*

$$\mu(i) = \begin{cases} \pi(i) & \text{if } i \le n \\ i & \text{if } i > n \end{cases}$$

is a hypergraph isomorphism between G'_f and G'_g.

Since, as candidate isomorphisms between G'_f and G'_g we wish to consider only permutations on $[n] \cup \{v_1, \ldots, v_r\}$ that fix each $v_i, 1 \le i \le r$, we perform the following operation on G'_f and G'_g so that any isomorphism between G'_f and G'_g is an identity on the set $\{v_1, \ldots, v_r\}$. For each vertex V_i, we add a $n+i$ length path starting at v_i. Since the hypergraphs G'_f and G'_g did not have paths of length more than n originally, any isomorphism between the hypergraphs cannot map v_i to a vertex other than v_i.

With this construction, it is sufficient to construct an isomorphism between the hypergraphs G'_f and G'_g to compute an isomorphism between $\widetilde{f_\ell}$ and $\widetilde{g}_\ell$. Now we invoke the algorithm of [4] on G'_f and G'_g which will yield an isomorphism ψ between $\widetilde{f_\ell}$ and $\widetilde{g}_\ell$. In summary, the algorithm for isomorphism testing f_ℓ and g_ℓ carries out the following steps.

Isomorphism Test for Polynomials

1. Construct the hypergraphs G'_f and G'_g as defined above.

2. Run the algorithm of Babai and Codenotti [4] on the hypergraphs G'_f and G'_g and output isomorphism ψ or report they are non-isomorphic.

Lemma 2.13. *The isomorphism of polynomials $\widetilde{f_\ell}$ and $\widetilde{g}_\ell$ (defined by Equation 2) can be tested in time $2^{O(\sqrt{n})(\ell+t)^2 \log^{O(1)} n}$. If the polynomials are isomorphic, then an exact isomorphism can be computed in the same running time bound.*

2.3 The approximate isomorphism algorithm

We now give an outline of the entire algorithm.

> **Input**: $f, g \in \mathcal{AC}_{s,d,n}$ given by circuits of size s along with parameters t and ℓ.
>
> **Step 1.** Compute the polynomials $\widetilde{f_\ell}$ and $\widetilde{g}_\ell$ using the randomized algorithm of Lemma 2.11.
>
> **Step 2.** Check if $\widetilde{f_\ell}$ and $\widetilde{g}_\ell$ are isomorphic using the polynomial isomorphism algorithm described above. If they are not isomorphic *reject* else **output** the computed exact isomorphism π.

Suppose π is an exact isomorphism between $\widetilde{f}_\ell$ and $\widetilde{g}_\ell$ computed by the above algorithm. By Lemma 2.10 π is a $(\delta+\varepsilon)^2$-approximate isomorphism between f and g, where $\delta = 2s2^{-t^{1/d}/20}$ and $\varepsilon = \frac{2n^{t/2}}{2^{(\ell-1)/2}}$. From Lemmas 2.11 and 2.13 it follows that the overall running time of the algorithm is $\mathrm{poly}(s, n^t, 2^\ell) + 2^{O(\sqrt{n})(\ell+t)^2 \log^{O(1)} n}$ and the error probability, as argued in Lemma 2.11, is at most $2^{-\Omega(n)}$.

We now set parameters to obtain the main result of the paper. Suppose f and g are $\frac{1}{2^\ell}$-approximate isomorphic, where $\ell = (\log n + \log s)^{kd}$ for a suitably large constant $k > 1$. Then we choose $t = (\log n + \log s)^{O(d)}$ so that $(\delta+\varepsilon)^2$ is bounded by $2^{-(\log n)^{O(1)}}$.

Theorem 2.14. *Given two Boolean functions $f, g \in \mathcal{AC}_{s,d,n}$ which are $\frac{1}{2^{(\log n)^{O(d)}}}$-isomorphic, there is a randomized algorithm running in time $2^{O(\log^{O(d)}(n)\sqrt{n})}$ to compute a permutation π such that f, g are $\frac{1}{2^{(\log n)^{O(1)}}}$-approximate isomorphic with respect to π.*

If the two Boolean functions f and g are isomorphic, then we get the following theorem about approximate isomorphism for f and g.

Theorem 2.15. *Given two isomorphic Boolean functions f and g computable by circuits in $\mathcal{AC}_{s,d,n}$, there is a randomized $2^{\log(n/\bar{\varepsilon})^{O(d)}\sqrt{n}}$-time algorithm that computes a permutation π such that f and g are $\bar{\varepsilon}$-approximately isomorphic with respect to π.*

Proof. Notice first that if f and g are isomorphic, then for any $\ell > 0$, f and g are $\frac{1}{2^\ell}$-approximately isomorphic. Also, by Lemma 2.10, we know that if π is an isomorphism between between $\widetilde{f}_\ell$ and $\widetilde{g}_\ell$, then π is a $(\delta+\varepsilon)^2$-approximate isomorphism between f and g. For $\delta + \varepsilon \leq \sqrt{\bar{\varepsilon}}$, it is enough to set δ and ε such that $\delta \leq \sqrt{\bar{\varepsilon}}/2$ and $\varepsilon \leq \sqrt{\bar{\varepsilon}}/2$. For this, fix $\ell = O(\log(n/\bar{\varepsilon})^d)$ and $t = O(\log(n/\bar{\varepsilon})^d)$ in Lemma 2.10. Using Lemma 2.11, we can compute $\widetilde{f}_\ell$ and $\widetilde{g}_\ell$ in time $\mathrm{poly}(s, 2^{(\log(n/\bar{\varepsilon})^d)})$ and construct an isomorphism between $\widetilde{f}_\ell$ and $\widetilde{g}_\ell$ using Lemma 2.13 in randomized time $2^{\log(n/\bar{\varepsilon})^{O(d)}\sqrt{n}}$. $\square$

3 A general approximate isomorphism algorithm

In this section we study the approximate isomorphism problem for general Boolean functions. Given two n-variable Boolean functions f and g (either by Boolean circuits computing them or just by black-box access) consider the optimization problem of finding a permutation π that minimizes $|\{x \in \{0,1\}^n \mid f(x) \neq g^\pi(x)\}|$, which we will call MinBooleanIso. A brute-force search that runs in $n!$ time by cycling through all permutations yields a trivial algorithm for this optimization problem.

We show that MinBooleanIso is coNP-hard under Turing reductions. We give a polynomial-time Turing reduction from the coNP-complete problem TAUTOLOGY (checking if a propositional formula is a tautology) to MinBooleanIso.

Lemma 3.1. TAUTOLOGY *is polynomial-time Turing reducible to* MinBooleanIso.

Proof. Given $f : \{0,1\}^n \to \{0,1\}$ as an n-variable propositional formula, we define functions $g_i : \{0,1\}^n \to \{0,1\}$ for $i \in [n]$ such that

$$g_i(x) = \begin{cases} 0 & \text{if } x = 1^i 0^{n-i} \\ 1 & \text{otherwise} \end{cases}$$

Notice that if f is a tautology then for each i, $|\{x \in \{0,1\}^n \mid f(x) \neq g_i^\pi(x)\}| = 1$ for all permutations π.

We now describe a polynomial-time algorithm for TAUTOLOGY with MinBooleanIso as oracle. For each g_i, we compute (with a query to the function oracle MinBooleanIso) a permutation π_i that minimizes $|\{x \in \{0,1\}^n \mid f(x) \neq g_i^\pi(x)\}|$. If $f(\pi_i^{-1}(1^i 0^{n-i})) = 1$ for each i, the algorithm describing the Turing reduction “accepts” f as a tautology and otherwise it “rejects” f.

We now show the correctness of the reduction. If f is a tautology, then clearly for each π_i we have $f(\pi_i^{-1}(1^i 0^{n-i})) = 1$. Conversely, suppose f is not a tautology. Then $f^{-1}(0) = \{x \in \{0,1\}^n \mid f(x) = 0\}$ is nonempty. Let $|f^{-1}(0)| = N$. Thus for any permutation π the cardinality $|\{x \in \{0,1\}^n \mid f(x) \neq g_i^\pi(x)\}|$ is either $N+1$ or $N-1$ for each i. Furthermore, suppose $x \in f^{-1}(0)$ has Hamming weight i. Then for any permutation π_i that maps x to $1^i 0^{n-i}$ we have $|\{x \in \{0,1\}^n \mid f(x) \neq g_i^{\pi_i}(x)\}| = N-1$. Hence, $f(\pi_i^{-1}(1^i 0^{n-i})) = f(x) = 0$. $\square$

Corresponding to the minimization problem, we look at the *maximization problem*: Find π that maximizes $|\{x \in \{0,1\}^n \mid f(x) = g^\pi(x)\}|$. Of course computing an optimal solution to this problem is polynomial-time equivalent to MinBooleanIso. In the remainder of this section we design a simple approximate isomorphism algorithm for the *maximization* problem. Our simple algorithm is based on the method of conditional probabilities. We first examine how good a random permutation is as an approximate isomorphism. Then we describe a deterministic algorithm for computing a permutation with the same solution quality.

3.1 Estimating the guarantee of a random permutation

For Boolean functions f and g, we now estimate the random variable $|\{x \mid f(x) = g^\pi(x)\}|$ where the permutation π is picked uniformly at random from S_n.

Lemma 3.2. *Let* $f : \{-1,1\}^n \to \{-1,1\}$ *and* $g : \{-1,1\}^n \to \{-1,1\}$ *be* δ*-close boolean functions for some* $\delta > 0$*. Then*

$$\mathbb{E}_\pi\left[|\{x \mid f(x) = g^\pi(x)\}|\right] \geq \frac{\delta^2 2^n}{64\sqrt{n}}$$

Proof. Let $s_i(f)$ denote the cardinality $|\{x \in \{0,1\}^n | \operatorname{wt}(x) = i, f(x) = 1\}|$ where $\operatorname{wt}(x)$ is the hamming weight of the Boolean string x. Clearly, $s_i(f) \leq \binom{n}{i}$. For each $u \in \{0,1\}^n$ define the 0-1 random variable X_u which takes value 1 if and only if $f(u) = g^\pi(u)$ for $\pi \in S_n$ picked uniformly at random. If $\operatorname{wt}(u) = i$, then

$$\Pr_\pi[X_u = 1] = \frac{s_i(g)}{\binom{n}{i}} f(u) + \frac{\binom{n}{i} - s_i(g)}{\binom{n}{i}} (1 - f(u)).$$

The sum $X = \sum_{u \in \{0,1\}^n} X_u$ is the random variable $|\{x | f(x) = g^\pi(x)\}|$ for a random permutation $\pi \in S_n$. We have

$$\begin{aligned}
\mathbb{E}_\pi[X] &= \sum_{i=0}^{n} \sum_{u:\operatorname{wt}(u)=i} \frac{s_i(g)}{\binom{n}{i}} f(u) + \sum_{i=0}^{n} \sum_{u:\operatorname{wt}(u)=i} \frac{\binom{n}{i} - s_i(g)}{\binom{n}{i}} (1 - f(u)) && (4) \\
&= \sum_{i=0}^{n} \frac{s_i(g) s_i(f)}{\binom{n}{i}} + \sum_{i=0}^{n} \frac{(\binom{n}{i} - s_i(g))(\binom{n}{i} - s_i(f))}{\binom{n}{i}} && (5) \\
&\geq \max_i \left(\frac{s_i(f) s_i(g)}{\binom{n}{i}}, \frac{(\binom{n}{i} - s_i(g))(\binom{n}{i} - s_i(f))}{\binom{n}{i}} \right). && (6)
\end{aligned}$$

Since f and g are δ-close for some constant $\delta > 0$, the fraction $\delta = \max_{\sigma \in S_n} |\{x | f(x) = g^\sigma(x)\}|/2^n$ is a constant (independent of n). For $0 \leq i \leq n$ let

$$\delta_i = \frac{|\{x \mid f(x) = g^\tau(x), \operatorname{wt}(x) = i\}|}{\binom{n}{i}}.$$

Thus $\sum_{i=0}^{n} \delta_i \binom{n}{i} = \delta 2^n$ which we can write as

$$\sum_{i=0}^{\sqrt{n}} (\delta_i + \delta_{n-\sqrt{n}+i}) \binom{n}{i} + \sum_{i=n/2-\sqrt{n}}^{n/2+\sqrt{n}} \delta_i \binom{n}{i} = \delta 2^n.$$

Since each $\delta_i \leq 1$, $\sum_{i=0}^{\sqrt{n}} (\delta_i + \delta_{n-\sqrt{n}+i}) \binom{n}{i} \leq 2n^{\sqrt{n}+1} \leq 2^{2\sqrt{n}\log n}$ for sufficiently large n.

Let A denote the sum $\sum_{i=n/2-\sqrt{n}}^{n/2+\sqrt{n}} \delta_i \binom{n}{i}$. Then

$$A \geq \delta 2^n \left(1 - \frac{2^{2\sqrt{n}\log n}}{\delta 2^n} \right) \geq \frac{\delta}{2} 2^n.$$

By averaging, there is some hamming weight i in the range $n/2 - \sqrt{n} \leq i \leq n/2 + \sqrt{n}$, such that

$$\delta_i \binom{n}{i} = |\{u \mid \operatorname{wt}(u) = i \text{ and } f(u) = g^\pi(u)\}| \geq \frac{\delta 2^n}{4\sqrt{n}}.$$

We fix this value of i and let S denote the set $\{u \mid \operatorname{wt}(u) = i$ and $f(u) = g^\pi(u)\}$. Assume without loss of generality that $|f^{-1}(1) \cap S| > \frac{\delta 2^n}{8\sqrt{n}}$ (Otherwise we consider $f^{-1}(0) \cap S$). Thus, we have $s_i(f) \geq |f^{-1}(1) \cap S| = |(g^\pi)^{-1}(1) \cap S| \geq \frac{\delta 2^n}{8\sqrt{n}}$. Similarly, $s_i(g) = \{u \mid \mathrm{wt}(u) = i \text{ and } g^\pi(u) = 1\} \supseteq (g^\pi)^{-1}(1) \cap S$. Hence $|(g^\pi)^{-1}(1) \cap S| \leq |\{u \mid \mathrm{wt}(u) = i \text{ and } g^\pi(u) = 1\}| = |\{u \mid \mathrm{wt}(u) = i \text{ and } g(u) = 1\}| = s_i(g)$. Therefore, both $s_i(f)$ and $s_i(g)$ are at least $\delta 2^n / 8\sqrt{n}$.

Combined with Equation 4 and using the inequality $\binom{n}{i} \leq \frac{2^n}{\sqrt{n}}$ for large enough n, we get the desired lower bound on $\mathbb{E}[X]$:

$$\mathbb{E}[X] \geq \frac{s_i(f) s_i(g)}{\binom{n}{i}} \geq \frac{\delta^2 2^{2n}}{64n\binom{n}{i}} \geq \frac{\delta^2 2^n}{64\sqrt{n}}.$$

□

Using the estimate for the expected value of the size of the set $\{x \mid f(x) = g^\pi(x)\}$, where π is picked at random, we get an approximate isomorphism algorithm which is given in the next theorem.

Theorem 3.3. *There is a deterministic $2^{O(n)}$ time algorithm that takes two boolean functions $f, g : \{0,1\}^n \to \{0,1\}$ as input (give either by boolean circuits or by black-box access) and outputs a permutation σ with the following property: If f and g^π are δ-close for some permutation π and constant δ, then*

$$|\{x \mid f(x) = g^\sigma(x)\}| \geq \Omega\left(\frac{2^n}{\sqrt{n}}\right).$$

Proof. Let X denote the random variable $|\{x \mid f(x) = g^\pi(x)\}$ where π is picked uniformly at random from S_n. From the previous lemma we know that the $\mathbb{E}_\pi[X] \geq \Omega(2^n/\sqrt{n})$. Now we show how to compute a permutation σ such that $|\{x | f(x) = g^\sigma(x)\}| \geq \mathbb{E}_\pi[X]$,

Let σ_i be an injective map $\sigma_i : \{1, 2, \ldots, i\} \to [n]$ and let $S_n^{\sigma_i}$ denote the set $\{\pi \in S_n \mid \pi(l) = \sigma(j) \text{ for } 1 \leq l \leq i\}$. Given the partial permutation σ_i defined on $\{1, 2, \ldots, i\}$, define random variables $X_{\sigma_i,u}$ for each $u \in \{0,1\}^n$ where $X_{\sigma_i,u} = 1$ if $f(u) = g^\pi(u)$ for π picked uniformly at random from $S_n^{\sigma_i}$. Let $X_{\sigma_i} = \sum_u X_{\sigma_i,u}$. Similar to Equation 4, we can write an expression for $\mathbb{E}[X_{\sigma_i}]$ and compute it exactly in time $2^{O(n)}$ for a given σ_i. For $j \in [n] \setminus \{\sigma_i(1), \sigma_i(2), \ldots, \sigma_i(i)\}$ let $\sigma_{i,j}$ denote the extension of σ_i that maps $i+1$ to j. In time $2^{O(n)}$ we can compute $\mathbb{E}[X_{\sigma_{i,j}}]$ for every j, and choose the permutation σ_{i+1} as that $\sigma_{i,j}$ which maximizes $\mathbb{E}[X_{\sigma_{i,j}}]$. In particular, this will satisfy $\mathbb{E}[X_{\sigma_{i+1}}] \geq \mathbb{E}[X_{\sigma_i}]$. Continuing this process until $i = n$ yields $\sigma_n = \sigma$ such that $|\{x \mid f(x) = g^\sigma(x)\}| \geq \mathbb{E}_\pi[X]$, where π is randomly picked from S_n. □

Concluding Remarks. Motivated by the question whether Boolean function isomorphism testing has algorithms faster than Luks's $2^{O(n)}$ time algorithm [12], we initiate the study of approximate Boolean function isomorphism. As our main result we show a substantially faster algorithm that for Boolean functions having small depth and small size circuits computes an approximate isomorphism. Precisely characterizing the approximation threshold for this problem for various Boolean function classes based on their circuit complexity is an interesting direction of research.

Acknowledgment

We are grateful to Johannes Köbler and Sebastian Kuhnert for discussions on the topic, especially for their help with Lemma 3.1.

References

[1] Manindra Agrawal and Thomas Thierauf. The formula isomorphism problem. *SIAM Journal on Computing*, 30(3):990–1009, 2000.

[2] Noga Alon and Eric Blais. Testing boolean function isomorphism. In *APPROX-RANDOM*, pages 394–405, 2010.

[3] Giorgio Ausiello, Francesco Cristiano, and Luigi Laura. Syntactic isomorphism of cnf boolean formulas is graph isomorphism complete. *Electronic Colloquium on Computational Complexity (ECCC)*, 19:122, 2012.

[4] László Babai and Paolo Codenotti. Isomorphism of hypergraphs of low rank in moderately exponential time. In *FOCS*, pages 667–676, 2008.

[5] László Babai and Eugene M. Luks. Canonical labeling of graphs. In *STOC*, pages 171–183, 1983.

[6] Eric Blais and Ryan O'Donnell. Lower bounds for testing function isomorphism. In *IEEE Conference on Computational Complexity*, pages 235–246, 2010.

[7] H Bunke. *Graph matching: Theoretical foundations, algorithms, and applications*, volume 23, pages 82–88. 2000.

[8] Sourav Chakraborty, David García-Soriano, and Arie Matsliah. Nearly tight bounds for testing function isomorphism. In *SODA*, pages 1683–1702, 2011.

[9] Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. In *FOCS*, pages 260–270, 1981.

[10] Johan Håstad. Almost optimal lower bounds for small depth circuits. In *STOC*, pages 6–20, 1986.

[11] Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, fourier transform, and learnability. *J. ACM*, 40(3):607–620, 1993.

[12] Eugene M. Luks. Hypergraph isomorphism and structural equivalence of boolean functions. In *STOC*, pages 652–658, 1999.

[13] B. V. Raghavendra Rao and Jayalal M. N. Sarma. Isomorphism testing of read-once functions and polynomials. In *FSTTCS*, pages 115–126, 2011.

[14] Alexander A. Razborov. Lower bounds for the size of circuits of bounded depth with basis $\{\wedge, \oplus\}$. *Math. notes of the Academy of Sciences of the USSR*, 41(4):333–338, 1987.

[15] Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *STOC*, pages 77–82, 1987.