Information and Computation 239 (2014) 3–12

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

Isomorphism testing of Boolean functions computable by constant-depth circuits

V. Arvind, Yadu Vasudev *

The Institute of Mathematical Sciences, CIT Campus, Taramani, Chennai-600113, India

ARTICLE INFO

Article history:
Received 24 July 2013
Available online 20 August 2014

Keywords:
Constant-depth circuits
Boolean functions
Isomorphism

ABSTRACT

Given two n-variable Boolean functions f and g, we study the problem of computing an ε-*approximate isomorphism* between them. An ε-*approximate isomorphism* is a permutation π of the n Boolean 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 [11].

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 $\overline{\text{CNFSAT}}$). 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

* Corresponding author.
E-mail addresses: arvind@imsc.res.in (V. Arvind), yadu@imsc.res.in (Y. Vasudev).

http://dx.doi.org/10.1016/j.ic.2014.08.003
0890-5401/

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.

1.1. 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 $\Pr_{x\in\{0,1\}^n}[f(x) \neq g(x)] \leq \frac{1}{2^\ell}$.

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 Boolean circuits of depth d and size s, where the circuit gates allowed are unbounded fan-in AND and OR gates, and negation gates. We recall that constant depth unbounded fan-in circuits are well-studied in complexity theory. The class AC^0 consists of languages $L \subseteq \{0,1\}^*$ for which there is a nonuniform family of circuits $\{C_n\}_{n>0}$ such that: (i) For each n the circuit C_n takes n input bits and accepts precisely the length n strings in L, and (ii) There are a constant d and a polynomial $p(n)$ such that C_n is an unbounded fan-in circuit of depth bounded by d and size bounded by $p(n)$ for each n. Furst, Saxe and Sipser [9] proved that the language of all strings of odd parity (i.e. the range of the parity function) is not in AC^0. A far reaching improvement of this result was due to Håstad [10] who obtained essentially optimal lower bounds for computing the parity of n variables. A different approach due to Razborov and Smolensky was to show that AC^0 circuits can be well approximated by polylogarithmic degree polynomials [14,15]. This technique was powerful enough to prove lower bounds even for AC^0 circuits that are allowed unbounded fan-in Mod p gates for prime p.

Linial, Mansour and Nisan [11], in the context of learnability of AC^0 computable functions, gave a different approximation of AC^0 circuits by polylogarithmic degree polynomials based on the Fourier analytic properties of AC^0 computable Boolean functions. This technique of [11] is a crucial ingredient in our algorithm for computing an approximate isomorphism between f and g given by $\mathcal{AC}_{s,d,n}$ circuits. If π is an isomorphism from f to g then we can show that π must map the approximating polynomial of f, defined via Fourier coefficients [11], to the approximating polynomial of g. This property is not true for the Razborov–Smolensky polynomial approximations of f and g; it is because those are probabilistic polynomials and we can only say that π maps the polynomial distribution corresponding to f to the one corresponding to g.

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.