

k -Approximating Circuits

Marco Cadoli, Francesco M. Donini,
Paolo Liberatore, and
Marco Schaerf, *Member, IEEE*

Abstract—In this paper, we define and study the k -approximating circuits. A circuit accepting a given set of inputs A is k -approximated by accepting inputs that differ from one of A by at most k bits. We show that the existence of polynomial-size k -approximating circuits depends on the relation between k and the number of inputs.

Index Terms—Reliability and testing, complexity measures and classes, models of computation.

1 INTRODUCTION

GIVEN two points in an n -dimensional Boolean space, their Hamming distance is the number of coordinates on which they differ. The Hamming distance is often regarded as the “natural” distance measure in such spaces. Viewing a point as a vector of bits, the distance between two points is the number of bits on which they differ: the smaller the distance, the more similar the two vectors.

We use the Hamming distance as a measure of approximation for Boolean functions. It is well known that there are some Boolean functions with n inputs whose optimal-size representing circuits are not polynomial with respect to n . In such cases, approximation can obviously be useful. Pippenger [16] introduced an approximation method for Boolean circuits in which points that are evaluated as false by a Boolean function are allowed to be evaluated to true by the approximating function, as long as their Hamming distance from a “true” point is less than or equal to a given bound value. In particular, points in such a condition can be evaluated in any way by the approximating function: The value of these points can be chosen so that the size of the representing circuit is minimized.

The approximation concept proposed in this paper is formally similar, but has different applications and different computational properties. Given an integer k , we define the k -approximation of a function f as the function f^k that is true on exactly all points whose Hamming distance from points that are evaluated to true by f is at most k . Formally, we denote by $x\Delta y$ the set of coordinates on which x and y differ and, with $|\cdot|$, the cardinality of a set; therefore, $|x\Delta y|$ is the Hamming distance between x and y . The k -approximation of a function f is therefore the function f^k defined as follows:

$$f^k(x) = \begin{cases} \text{true} & \text{if there exists } y \text{ such that } |x\Delta y| \leq k \\ & \text{and } f(y) = \text{true}, \\ \text{false} & \text{otherwise.} \end{cases} \quad (1)$$

Fig. 1 is a graphical representation of this concept: f is represented by the set of points it evaluates to true; f^k is true on the same points, plus all other points that are at most k far from

them. The points in the “border” (whose width is k) are evaluated to false by f and to true by f^k . Note that, on the contrary, Pippenger’s circuit approximation is free in the evaluation of these points, that is, it can evaluate them to either true or false.

We remark that, while Pippenger’s circuit approximation and k -approximation have similar definitions, their aims are completely different. Circuit approximation changes the truth evaluation of some points with the aim of representing the function with a Boolean circuit of polynomial size. k -approximation evaluates to true all points which are not too different from points that the original function evaluates to true. The aim of circuit approximation is to obtain a computational advantage (size of representing circuit) at the expense of allowing some “mistakes” in the truth evaluation of points. k -approximation is done for obtaining a different function that evaluates to true some additional points.

A k -approximation circuit can be seen as a solution for the problem of checking the distance between a point and a language (a set of points). This problem has been considered in different fields such as error-correcting codes (check the distance between a string and the closest one encoding an input string [7], [1], [2], which is in turn an issue relevant to cryptography [18]), recovery from syntax errors (check the distance between a string and the closest one recognized by a grammar or an automata [12], [15]), computational biology [8], [13] (approximate match of a string with a structural pattern), and temporal databases (approximate match of the sequence representing the evolution of a database over time with an automata [17]). The edit distance [11] is often used instead of the Hamming distance. However, Manthey and Reischuk [12] have shown that these distance measures are equivalent from a theoretical-complexity point of view.

In all these cases, a k -approximation circuit allows for solving the problem of checking whether the distance from a point to the language is less than or equal to k . If the language can be recognized in time polynomial in some parameter (such as the number of dimensions of the space), it can be represented by a circuit of size polynomial in that parameter. The assumption that the language is represented by a circuit therefore formalizes the polynomiality in time of exact string membership to the language. A polynomial-size k -approximation circuit allows for finding out the distance of a string to the language in polynomial time.

The constraints imposed by k -approximation result in a computational cost: As is proven in Section 3, there are Boolean functions that can be represented by polynomial-size circuits while their k -approximation (for some k) cannot unless the Polynomial Hierarchy collapses. It is also possible to prove the converse, that is, there are Boolean functions that cannot be represented by polynomial circuits while some of their k -approximations can. This is actually easy to prove: Consider a family of Boolean functions $\langle f_n \rangle$, where f_n has n arguments that cannot be represented by a family of polynomial circuits (the proof of existence of such families is due to Shannon’s counting argument), and their k -approximation, where $k = n$. The approximation of f_n is the function that evaluates to true on all points of the space and can be easily represented with a very small circuit.

2 PRELIMINARIES

2.1 Boolean Circuits

A Boolean circuit is a directed acyclic graph containing a single node with out-degree equal to 0, which is called the output. The nodes of in-degree 0 are called *inputs* (and are labeled either with a number or with a constant true or false), while the other nodes are called *gates* and are labeled with an (unbounded fan-in) Boolean connective [3], e.g., \wedge , \vee . Given a set of Boolean values, one for each input, a circuit determines the truth value induced at its output. The output that is produced on the tuple of Boolean values I by the

• M. Cadoli, P. Liberatore, and M. Schaerf are with the Dipartimento di Informatica e Sistemistica, Università di Roma “La Sapienza,” Via Salaria 113, I-00198, Roma, Italy.

E-mail: {cadoli, liberatore, scharf}@dis.uniroma1.it.

• F.M. Donini is with the Università della Tuscia, Facoltà di Scienze Politiche, Via San Carlo 32, 01100 Viterbo, Italy.

E-mail: donini@unitus.it.

Manuscript received 23 Dec. 2004; revised 16 Aug. 2005; accepted 19 Aug. 2005; published online 22 May 2006.

For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-0426-1204.

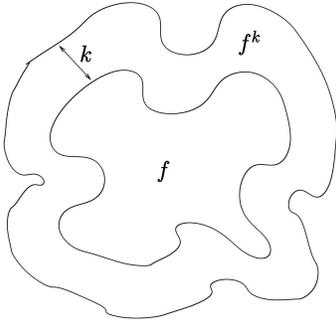


Fig. 1. A function and its approximation: Points in the k -wide border are evaluated to false by f , to true by f^k , and can be evaluated to any value by Pippenger's approximation.

circuit C is denoted by $C(I)$. The size of a Boolean circuit C is the number of its gates and is denoted by $\|C\|$ and it also determines the space it occupies. In general, the notation $\|\cdot\|$ denotes the size of an object; $|\cdot|$ represents its cardinality.

A Boolean function of n arguments is a function from n -tuples of Boolean values to $\{\text{true}, \text{false}\}$. Boolean circuits and Boolean functions are clearly related. Indeed, both functions and circuits determine a truth value for any n -tuple of Boolean values. As a result, a circuit with n inputs represents a Boolean function of n arguments and vice versa.

Formally, an n -input circuit computes a Boolean function f if, for all assignments $x \in \{0, 1\}^n$ to the n inputs, the value induced at the output gate is $f(x)$. On the other hand, a function f represents a circuit C if and only if $f(x)$ is equal to the output of C when the values x are set as the input gates. Since Boolean circuits are suitable for "implementing" Boolean functions, in the following, we refer to circuits, rather than functions.

We can indeed extend the notion of k -approximation to circuits: If C is a circuit representing f , then C^k is a k -approximation of C if it represents f^k . Note that, given a function f and an integer k , there is a unique function f^k that is the k -approximation of f . On the other hand, both f and f^k can be represented by many circuits. A function can be represented in polynomial space if and only if there exists a polynomial p such that a circuit C represents f and $\|C\| \leq p(n)$, where n is the number of input gates of C .

A polynomial circuit representing f might or might not exist. The same holds for f^k ; hence, there are four possible combinations. In particular, if $k = 0$, then $f = f^k$. It is therefore possible that both functions are represented by polynomial circuits and it is also possible that both are not. In Section 1, we have already shown a function f and a value of k such that f cannot be polynomially represented while f^k can. In the following, we prove that there are functions f that can be represented by polynomial circuits while f^k cannot.

This is perhaps the most interesting of the four cases: We may indeed assume that the function f is already represented by a circuit, so what we have is actually C . Approximating means finding a new circuit that approximates C . The main result of this paper is that, in some cases, there is no k -approximating circuit that is only polynomially larger than C .

2.2 Nonuniform Complexity Classes

We assume the reader is familiar with (uniform) classes of the Polynomial Hierarchy, like P, NP, coNP, Σ_2^P , etc. [10]. We only briefly introduce nonuniform classes [10, p. 116], which we use in some proofs. In the context of strings, the notation $\|s\|$ denotes the length of the string s .

Definition 1. An advice-taking Turing machine is a single-tape Turing machine enhanced by an "advice oracle" A , which is a function from integers to strings (not necessarily recursive). On

input s , the tape is automatically loaded with $A(\|s\|)$ and, from then on, the computation proceeds as normal, based on the two inputs s and $A(\|s\|)$.

Definition 2. An advice-taking Turing machine uses polynomial advice if its advice oracle A satisfies $\|A(n)\| \leq p(n)$ for some fixed polynomial p and all nonnegative integers n .

Definition 3. If C is a class of languages defined in terms of resource-bounded Turing machines, then C/poly is the class of languages defined by Turing machines with the same resource bounds, but augmented by polynomial advice.

As an example, P/poly is the class of languages recognizable by a Turing machine working in polynomial time using polynomial advice, NP/poly uses a Turing machine working in nondeterministic polynomial time, etc. A class C/poly is also known as nonuniform C , where nonuniformity is due to the presence of the advice. The nonuniform and uniform complexity classes are related: The collapse of the nonuniform hierarchy implies the collapse of the uniform one at some level [10]. The noncollapse of the polynomial hierarchy is considered very likely by most researchers, but has not been proven so far.

3 APPROXIMATION BASED ON THE HAMMING DISTANCE

We investigate whether k -approximation is actually feasible in polynomial space or not. Intuitively, we want to find out whether every circuit C has a k -approximating circuit C^k whose size is bounded by a polynomial in the size of C .

Question 1 (Polynomiality of Approximation). Is there a polynomial p such that, for any circuit C and any $k \geq 0$, the circuit C has a k -approximation C^k of size $\|C^k\| \leq p(\|C\|)$?

The answer to this very general question is no (Theorem 2). Nevertheless, it is interesting to study this problem when k depends on C and, in particular, when k depends on the number of inputs n of C . Several dependencies are possible:

1. $k \in O(1)$, i.e., k is a constant.
2. $k \in O(\log n)$, i.e., it is a logarithmic function of n .
3. $k \in O(n^\alpha)$ with $\alpha < 1$, i.e., it is a sublinear function in n .
4. k is linear in n .

We discuss each case separately.

3.1 Fixed k -Approximation

We can positively answer Question 1 for the case in which k is a constant. Hence, we can say, for example, that 2-approximation is feasible, that 3-approximation is feasible, etc.

Theorem 1 (Polynomiality of Fixed k -approximation). For each positive integer k , there exists a polynomial p such that, for every circuit C , it holds $\|C^k\| \leq p(\|C\|)$, where C^k is the smallest k -approximation of C .

Proof. The proof is based on the fact that, for each k , we can choose a different polynomial, denoted p_k . Let C be a circuit of size m with n inputs; by definition, $n \leq m$. Since k is to be regarded as a constant, $p_k(m) = m^{k+1}$ is a polynomial. We prove that $p_k(m) = m^{k+1}$ indeed bounds the size of k -approximations of any circuit C . The following circuit k -approximates C .

$$C^k = \bigvee_{X' \subseteq X, |X'| \leq k} C[X'/\neg X']$$

We use the notation $C[X'/\neg X']$ to refer to the circuit in which a negation gate is introduced on every input in X' . We have to

show that this circuit has size bounded by $p_k(n)$, and that it is a k -approximation of C .

The circuit C^k is composed of a number of copies of C , one for each subset of X composed of k elements. Since C has n inputs, $|X| = n$. The number of subsets of k elements of a set of n elements is less than n^k . Therefore, C^k is made of at most n^k copies of C . Its size is therefore bounded by $m \cdot n^k$. Since $n \leq m$, the size of C^k is also bounded by m^{k+1} . We now prove that C^k is a k -approximation of C . For the first direction, let I be an arbitrary tuple whose Hamming distance is t away from a tuple J that makes C output true. Let J be the tuple that makes C output true and such that $|I\Delta J| = t \leq k$. Let $X' = I\Delta J$. By definition, $C(J) = \text{true}$, which implies $C[X'/\neg X'](I) = C[J] = \text{true}$. Since the latter is part of the disjunction that forms C^k , we have $C^k(I) = \text{true}$.

Conversely, let us assume that $C^k(I) = \text{true}$ and let us prove that there exists a tuple J such that $C(J) = \text{true}$ and $|I\Delta J| \leq k$. By definition of C^k , we have $C^k(I) = \text{true}$ if and only if there exists a set $X' \subseteq X$ with $|X'| \leq k$ and such that $C[X'/\neg X'](I) = \text{true}$. If J is the tuple that differs from I for the elements in X' , we have $C(J) = \text{true}$ and $|I\Delta J| = |X'| \leq k$. \square

This theorem shows, for example, that 2-approximation can be done without an exponential increase in size. However, the proof also shows an annoying exponential dependency on k , as the k -approximating circuit C^k is n^k larger than the original one, n being the number of inputs of C . No harm is done if k is fixed, for example, if $k = 2$, when the approximation increase the size of only n^2 times. However, this means that we are always doing the approximation regardless of the number of inputs of C . While 2-approximating a circuit of 10 variables may be reasonable, it may not be when the inputs are 1,000. In such cases, the bound on the approximation k should increase with the number of inputs.

3.2 Logarithmic k -Approximation

We consider a value of k that depends on n , but only moderately increases with it. In this case, the bound for the approximation is not a constant value, but the result of a function. In particular, we consider a logarithmic function. The construction of the last section leads to a k -approximation that is n^k times the size of the circuit. While this is still subexponential (i.e., it is $n^{\log n}$ if $k = \log n$), it is not a polynomial any more.

3.3 Sublinear k -Approximation

For this case, we have a negative answer to Question 1. The following theorem shows a uniform family of circuits $\{C_0, C_1, C_2, \dots\}$ such that C_n is an n -input circuit of size polynomial in n , and the size of their k -approximations increases more than every polynomial function in n , unless the Polynomial Hierarchy collapses. Note that this result is not proven by exhibiting the minimal-size k -approximating circuits of the above family and showing that their size is (in the worst case) nonpolynomial.

Theorem 2. *Let $k(n) \in \theta(n^\alpha)$ with $\alpha = \frac{1}{3}$. There exists a uniform family of circuits $\{C_0, C_1, C_2, \dots\}$ such that, if there exists a polynomial p for which $\|C_n^{k(n)}\| \leq p(\|C_n\|)$ for all $n \geq 0$, then $NP \subseteq P/Poly$.*

Proof. Since the proof is rather long, we first give an outline to improve its readability. The proof consists of the following steps:

1. choice of an NP-complete problem π ;
2. definition of the family in such a way that the n th circuit of the family C_n is polynomial in n and, for each instance F of π of size m , the answer to F is "yes" if and only if there exists an n -bit input I_F (computable in polynomial time) such that $C_n^{k(n)}$ has output true on I_F , with $k(n) = m \in O(\sqrt[3]{n})$;

3. proof that, for $k = m \in O(\sqrt[3]{n})$, if, for every n , C_n admits a k -approximating circuit $C_n^{k(n)}$ that is polynomial in $\|C_n\|$ (hence, polynomial in n), then NP is contained in P/poly.

Step 1: We choose the NP-complete problem 3sat. Let F be an instance of 3sat, i.e., a 3CNF formula, with $\|F\| = m$. The number of propositional letters contained in F is bounded by m . It will be useful to have an easy way to determine the number of variables of a formula. To this aim, we assume that any formula F of size m is built over the alphabet $X_F = \{x_1, \dots, x_m\}$, even if F only contains some of these variables. This way, we have $\|F\| = |X_F|$ for any formula F . From now on, we omit the subscript F in X for simplicity.

Step 2: Given n , we show how to build the circuit C_n in such a way that C_n depends only on n and its size will be polynomial in n . Moreover, we want to enforce that a 3CNF formula F (with $\|F\| = m$) is satisfiable if and only if there is a set of input values I_F that makes true the output gate of C_n , where $n \in O(m^3)$.

Let $Y = \{y_1, \dots, y_m\}$ be a set of new letters in one-to-one correspondence with letters of X and let G be a set of new letters one-to-one with the set of the three-literal clauses over X , i.e., $G = \{g_i \mid \gamma_i \text{ is a three-literal clause of } X\}$. Finally, let L be the set $X \cup Y \cup G$ and $n = |L|$. Notice that $n \in O(m^3)$. We define C_n as the conjunction of two formulae:

$$C_n = \Delta_m \wedge \Gamma_m. \quad (2)$$

Δ_m states nonequivalence between atoms in X and their corresponding atoms in Y , while Γ_m codes every possible 3CNF formula over X , using the atoms in G as "enabling gates."

$$\begin{aligned} \Delta_m &= \bigwedge_{i=1}^m (x_i \neq y_i), \\ \Gamma_m &= \bigwedge_{g_i \in G} \gamma_i \vee \neg \gamma_i. \end{aligned} \quad (3)$$

Γ_m contains $O(m^3)$ clauses. The overall circuit C_n is therefore polynomially large. It is an n -input circuit and it does not depend on the specific 3CNF formula F , but only on the size m of its alphabet. Therefore, we have proven that it satisfies all requirements but the last, that is, showing that, for $k(n) = m$, the satisfiability of F is equivalent to the existence of an input set that makes $C_n^{k(n)}$ output true. Let F be an arbitrary 3CNF formula over X and let G_F be the set of g_i s whose corresponding clauses are in F :

$$G_F = \{g_i \in G \mid \gamma_i \text{ is a clause of } F\}.$$

The input set I_F is defined as follows:

$$I_F(l) = \begin{cases} \text{true} & \text{if } l \in G_F, \\ \text{false} & \text{if } l \in (G \setminus G_F) \cup X \cup Y. \end{cases} \quad (4)$$

We now show that F is satisfiable iff $C_n^{k(n)}$ has output true on input I_F .

If part. Let F be satisfiable and M_F be an interpretation that satisfies F . We denote with X_F the set of literals that are mapped into true by M_F . Let $\overline{Y}_F = \{y_i \in Y \mid x_i \notin X_F\}$. We prove that $C_n^{k(n)}$ gives true on I_F by showing an input set I on which C_n gives true and $|I\Delta I_F| \leq m = k(n)$. This inputs set I is defined as follows:

$$I(l) = \begin{cases} \text{true} & \text{if } l \in (G_F \cup X_F \cup \overline{Y}_F), \\ \text{false} & \text{otherwise.} \end{cases} \quad (5)$$

We show that C_n gives true on I . The output of Δ_m on I is true by construction of \overline{Y}_F and also the output of Γ_m on I

is true because, for each clause $\gamma_i \vee \neg \gamma_i$ of Γ_m , either $I(g_i) = \text{false}$ or $I(\gamma_i) = \text{true}$ since X_F satisfies γ_i . Now, observe that $|I\Delta I_F| = |X_F \cup \overline{Y}_F| = m = k(n)$. Hence, $C_n^{k(n)}$ outputs true on input I_F .

Only if part. Suppose that $C_n^{k(n)}$ gives true on I_F . Then, there exists an input I that makes C_n give true, with $|I_F\Delta I| \leq m$. Note that $|I_F\Delta I| \geq m$ because, for all $1 \leq i \leq m$, the input I must assign false to exactly one of x_i and y_i , while I_F assigns false to all inputs in $X \cup Y$. Therefore, $|I_F\Delta I| = m$. Hence, I and I_F must assign the same value to all inputs in G . Let I_X be the input such that $I_X(l) = \text{true}$ if $I(l) = \text{true}$ and $l \in X$ (we remind that X is the set of variables of F), false otherwise. Since Γ_m outputs true on I , simplifying the circuit Γ_m by assigning to the inputs in G the value assigned by I_F , we obtain exactly the formula F . Thus, the model $M = \{l \mid l \in X, I(l) = \text{true}\}$ satisfies F .

Step 3: Let us assume that there exists a polynomial p with the properties claimed in the statement of the theorem. Then, for each circuit C_n , there exists a k -approximating circuit $C_n^{k(n)}$ with $\|C_n^{k(n)}\| < p(\|C_n\|)$. We define an advice-taking Turing machine that determines the satisfiability of propositional formulae in polynomial time in this way: Given a generic propositional formula F , with $\|F\| = m$, the machine loads the advice, that is, a representation of the circuit C_n^k , computes I_F , and then checks whether C_n^k gives true on I_F in polynomial time. Since $\|C_n^k\| = O(m^3)$, the advice has size $O(p(m^3))$, hence we would have shown that satisfiability of propositional formulae is in nonuniform P. Since the satisfiability of propositional formulae is an NP-complete problem, this implies $\text{NP} \subseteq \text{P/poly}$. \square

Recall that $\text{NP} \subseteq \text{P/poly}$ implies $\bigcup_{i>0} \Sigma_i^p \subseteq \Sigma_2^p$, i.e., the Polynomial Hierarchy collapses at the second level [10]. Although this is a relative argument, the collapse of the Polynomial Hierarchy is considered unlikely by current research in computational complexity.

The circuits C_n of (2) have depth 2 and each circuit can be built by an algorithm using only an amount of space that is logarithmic in n . As a result, there exists a language in the class AC^0 [10] that has a family of polynomial k -approximating circuits only if the polynomial hierarchy collapses.

A question that naturally arises is whether the exponential blow-up happens not only for the specific class of functions $k(n) \in \theta(n^\alpha)$ with $\alpha = \frac{1}{3}$, but also for a more general class of functions. In fact, we can enlarge the above results to smaller values of α .

Theorem 3. *Question 1 has a negative answer for $k(n) \in O(n^\alpha)$ with $\alpha \leq \frac{1}{3}$.*

Proof. Let $t(m)$ be a polynomial such that $t(m) \in \Omega(m^3)$. We can modify the reduction of the previous theorem, "inflating" the number of inputs of the circuit as follows: Given a formula F with $\|F\| = m$, we build the following circuit C_n :

$$C_n = \Delta_m \wedge \Gamma_m \wedge \bigwedge_{i=1}^{t(m)} z_i.$$

Δ_m and Γ_m are as before and $Z = \{z_1, \dots, z_{t(m)}\}$ is a new set of variables, with $|Z| = t(m)$. The number of inputs of this circuit is $n = \Omega(m^3 + t(m))$. We still use $k(n) = m$, therefore $k(n) \in O(n^\alpha)$ with $\alpha \leq \frac{1}{3}$. This proves the claim. \square

3.4 Linear k -Approximation

As remarked in the introduction, if $k(n) = n$, a trivial k -approximation of any circuit having n inputs always exists. A similar result holds for any linear function $k(n) = n - h$, where h is a constant. This is a consequence of the fact that, for any x , there are less than $2n^h$ inputs that are at least $n - h + 1$ far from it. Therefore, for any circuit C that evaluates true on some input, C^k

evaluates false on at most $2n^h$ inputs; such circuits can be represented in linear space.

While Question 1 has a positive answer for $k(n) = n$ and $k(n) = n - h$ for any fixed h , it has a negative answer for $k(n) \in O(n)$ in general. Indeed, a statement similar to Theorem 2 can be proven for some functions $k(n)$ such that $k(n) \in O(n)$. The reduction in Theorem 2 is modified as follows: Let F be a formula with $\|F\| = m$ and let $r(m)$ be a polynomial in m . We build the following circuit C_n :

$$C_n = \Delta_m \wedge \Gamma_m \wedge \bigwedge_{i=1}^{r(m)} z_i \neq w_i.$$

The cardinality of both Z and W is $r(m)$. The inputs to C_n are now X, Y, G, W , and Z , so their number n is $n = 2m + m^3 + 2r(m)$. Hence, $n \in \theta(r(m))$. We use the input in which all variables in $X \cup Y \cup Z \cup W$ are false and a distance k (as a function of m) equal to $m + r(m)$. Following the same line of reasoning of Theorem 2, it can be shown that $C_n^{k(n)}$ outputs true iff F is satisfiable. Let $r(m) = \frac{1}{2}(m^\beta - m^3) - m$, with $\beta > 3$, so that $r(m) \geq 0$. Now, $n = m^\beta$, and $k = \frac{1}{2}(n - n^{3/\beta})$, hence $k \in O(n)$.

4 DISTANCE BASED ON SET CONTAINMENT

In the previous section, we took the Hamming distance between two points as a measure of the allowed error. Different measures can, however, be used instead. We now analyze a measure based on set containment. Given a circuit C on n inputs and a set of integers $S \subseteq \{1, 2, \dots, n\}$, an S -approximation of C is a circuit C^S that outputs 1 on input $x = (x_1, x_2, \dots, x_n)$ if and only if there exists another input $y = (y_1, y_2, \dots, y_n)$ such that C outputs true on x and $x_i = y_i$ for all $i \notin S$. In other words, C^S is the circuit that outputs 1 on all inputs that disagree only on the bits in S with inputs where C outputs 1.

Intuitively, if the circuit C outputs 1 on a set of n inputs x , then C^S will output 1 on all sets of n inputs that can be obtained from x by changing some of the bits in S , while bits not in S are fixed. On the contrary, k -approximation allows for changing any bit, provided that no more than k bits are changed. k -approximation is therefore based on the assumption that all bits have the same status or that errors in them have the same probability. On the contrary, S -approximation can be seen as a formalization of assuming that the bits in S can be wrong, but we are sure that the other ones are not. The study of S -approximations reported here is not as detailed as that of k -approximation. We only show that polynomial S -approximation is impossible in general, but feasible if S is a set of fixed cardinality, i.e., independent on n .

Theorem 4. *There exists a family of circuits $\{C_0, C_1, C_2, \dots\}$ such that, if there exists a polynomial p such that $\|C_n^S\| \leq p(\|C_n\|)$, where C_n^S is a minimal S -approximation of C_n , for all $n \geq 0$ and $S \subseteq \mathbb{N}$, then $\text{NP} \subseteq \text{P/Poly}$.*

Proof. The proof is similar to that of Theorem 2. We only point out the differences. The first step is the same. In the second one, we show that, for any integer m , there exists an n -input circuit C_n , depending only on m , of polynomial size with respect to m , such that, given any 3CNF formula F over an alphabet of m atoms, there exists an n -bit input I_F such that F is satisfiable iff I_F makes C_n^S output true. This circuit C_n is defined as in the other proof. We define $S = X \cup Y$.

Given a 3CNF formula F over X , we define $G_F = \{g_i \in G \mid \gamma_i \text{ is a clause of } F\}$. Given G_F , we define an n -bit input I_F as follows: $I_F(l) = \text{true}$ if $l \in G_F$, false otherwise. We now show that F is satisfiable iff C_n^S outputs true on input I_F .

If part. Let F be satisfiable and let X_F be a model of F . Let $\overline{Y}_F = \{y_i \mid x_i \notin X_F\}$ and let the input I be defined as follows:

$I(l) = \text{true}$ if $l \in (G_F \cup X_F \cup \overline{Y_F})$, false otherwise. We show that C_n outputs true on I . The output of Δ_m on I is true by construction of $\overline{Y_F}$, and also the output of Γ_m on I is true because, for each clause $\gamma_i \vee \neg g_i$ of Γ_m , either $I(g_i) = \text{false}$ or $I(\gamma_i) = \text{true}$ since X_F satisfies γ_i . Now, observe that $(I\Delta I_F) \subseteq X \cup Y = S$. Hence, C_n^S outputs true on input I_F .

Only if part. Suppose C_n^S outputs true on I_F . Then, there exists an input I such that makes C_n output true and $I\Delta I_F \subseteq S$. Therefore, I and I_F must assign the same value to all inputs in G . Let I_X be the input such that $I_X(l) = \text{true}$ if $I(l) = \text{true}$ and $l \in X$, false otherwise. Since Γ_m outputs true on I , simplifying the circuit Γ_m by assigning to the inputs in G the value assigned by I_F , we obtain exactly the formula F . Thus, the model $M = \{l \mid l \in X, I(l) = \text{true}\}$ satisfies F .

The third step of the proof (proving that the assumptions imply $\text{NP} \subseteq \text{P/Poly}$) is identical to the one of Theorem 2. \square

Let us now prove that, for any fixed $S \subseteq \mathbb{N}$, it is possible to S -approximate any circuit. The specific definition of this question is identical to the one given in the last section (substituting k with S).

Theorem 5. *For any fixed set $S \subseteq \mathbb{N}$, there exists a polynomial p such that, for any circuit C , it holds $\|C^S\| \leq p(\|C\|)$, where C^S is a minimal S -approximation of C .*

Proof. Given a circuit C , the following is an S -approximation of it:

$$C^S = \bigvee_{S' \subseteq S} C[S'/\neg S'],$$

where $C[S'/\neg S']$ is the circuit obtained modifying C by negating the variables in S' . This circuit is at most $2^{|S|}$ larger than C : Its size is therefore polynomial as S is a constant set. \square

5 DISCUSSION AND CONCLUSIONS

When we compare our results with similar ones in circuit approximation, a huge difference is apparent: While circuit approximation can make it possible to represent functions with polynomial circuits (which may otherwise be impossible), k -approximation seems to degrade the quality of circuit representation. As explained in the introduction, these two approximation methodologies have different definitions and aims. While the aim of the first one is to reduce the size of the representing circuits, the second one aims at increasing the confidence in the truth value of some points. This difference in aim leads to a difference in definition so that the first one admits a certain degree of freedom in choosing the truth values of some points if doing so shortens the representation; on the converse, the truth value of all points is specified in the k -approximation of a function. Removing this degree of freedom leads to the complexity result of this paper: The k -approximation of a function cannot be represented, in general, by a polynomial circuit.

The main result of this paper can be linked to other recent ones regarding how compactly information can be represented. According to various authors [4], [14], the space efficiency of a logical formalism is its efficiency in representing information with a small amount of space. Trivial examples are easy to find: First order logic formulas may need exponential space to be converted into equivalent propositional formulae, so the former one is more space efficient than the latter one. In many cases, however, a proof of equal/different space efficiency is more complicated (see the papers mentioned above for examples).

The results presented in this paper deal with space efficiency: A Boolean function may be represented with little space by saying it is the k -approximation (for a suitable number k) of another function which can be represented with exponentially less space. In other

words, we can represent a Boolean function by a pair $\langle C, k \rangle$, where C is a circuit and k is a number, such that the function is the k -approximation of the function represented by C . Clearly, any function can be represented with $k = 0$. However, as we have shown in this paper, in some cases, $\langle C, k \rangle$ represents a function whose smallest representing circuit cannot be represented in space polynomial in the size of the original circuit C . As a result, the "formalism" of using a pair can be more space efficient than the usual representation using circuits.

An interesting open question is whether polynomial-size k -approximation circuits exist in the context of fixed-parameter approximation. The fixed-parameter complexity of checking the Hamming distance from a language has already been investigated [12], [6]; it would make sense to consider the problem of existence of k -approximating circuits whose size is parameterized polynomial when k is the parameter. Such an issue is related to the general framework of parameterized compilability [9]; a technical discussion on the differences between parameterized complexity and a framework closely related to the work reported in the present paper can be found in [5].

REFERENCES

- [1] S. Arora, L. Babai, J. Stern, and Z. Sweedyk, "The Hardness of Approximate Optima in Lattices, Codes, and Systems of Linear Equations," *J. Computer and System Sciences*, vol. 54, pp. 317-331, 1997.
- [2] E. Berlekamp, R. McEliece, and H. van Tilborg, "On the Inherent Intractability of Certain Coding Problems," *IEEE Trans. Information Theory*, vol. 24, pp. 384-386, 1978.
- [3] R. Boppana and M. Sipser, "The Complexity of Finite Functions," *Handbook of Theoretical Computer Science*, J. van Leeuwen, ed., vol. A, chapter 14, pp. 757-804, Amsterdam: Elsevier Science Publishers, 1990.
- [4] M. Cadoli, F.M. Donini, P. Liberatore, and M. Schaerf, "Space Efficiency of Propositional Knowledge Representation Formalisms," *J. Artificial Intelligence Research*, vol. 13, pp. 1-31, 2000.
- [5] M. Cadoli, F.M. Donini, P. Liberatore, and M. Schaerf, "Preprocessing of Intractable Problems," *Information and Computation*, vol. 176, no. 2, pp. 89-120, 2002.
- [6] R. Downey, M. Fellows, A. Vardy, and G. Whittle, "The Parameterized Complexity of Some Fundamental Problems in Coding Theory," *SIAM J. Computing*, vol. 29, no. 2, pp. 545-570, 1999.
- [7] I. Dumer, D. Micciancio, and M. Sudan, "Hardness of Approximating the Minimum Distance of a Linear Code," *IEEE Trans. Information Theory*, vol. 49, pp. 22-37, 2003.
- [8] I. Eidhammer, I. Jonassen, S. Grindhaug, D. Gilbert, and M. Ratnayake, "A Constraint Based Structure Description Language for Biosequences," *Constraints*, vol. 6, pp. 173-200, 2001.
- [9] H. Chen, "Parameterized Compilability," *Proc. 19th Int'l Joint Conf. Artificial Intelligence (IJCAI 2005)*, 2005.
- [10] D.S. Johnson, "A Catalog of Complexity Classes," *Handbook of Theoretical Computer Science*, J. van Leeuwen, ed., vol. A, chapter 2, pp. 67-161, Amsterdam: Elsevier Science Publishers, 1990.
- [11] V. Levenshtein, "Binary Codes Capable of Correcting Deletions, Insertions, and Reversals," *Soviet Physics Doklady*, vol. 10, 1966.
- [12] B. Manthey and R. Reischuk, "The Intractability of Computing the Hamming Distance," *Theoretical Computer Science*, vol. 337, pp. 331-346, 2005.
- [13] G. Mehltau and G. Myers, "A System for Pattern Matching Applications on Biosequences," *Computer Applications in the Biosciences*, vol. 9, pp. 299-314, 1993.
- [14] P. Penna, "Succinct Representations of Model Based Belief Revision," *Proc. 17th Ann. Symp. Theoretical Aspects of Computer Science (STACS 2000)*, pp. 205-216, 2000.
- [15] G. Pighizzini, "How Hard Is Computing the Edit Distance?" *Information and Computation*, vol. 165, pp. 1-13, 2001.
- [16] N. Pippenger, "Information Theory and the Complexity of Boolean Functions," *Math. Systems Theory*, vol. 10, pp. 129-167, 1977.
- [17] A. Prasad Sistla, T. Hu, and V. Chowdhry, "Similarity Based Retrieval from Sequence Databases Using Automata As Queries," *Proc. 2002 ACM CIKM Int'l Conf. Information and Knowledge Management*, pp. 237-244, 2002.
- [18] J. Stern, "A New Paradigm for Public Key Identification," *IEEE Trans. Information Theory*, vol. 42, pp. 1757-1768, 1996.

► For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.