

# Approximating Succinct MaxSat

Christian Schallhart\* and Luca Trevisan†

## Abstract

We study the approximability of the version of MAXSAT where exponentially large instances are succinctly represented using circuits. First, we prove that the **NP**-hardness for approximating MAXSAT can be lifted to a corresponding **NEXP**-hardness for approximating circuit-succinct MAXSAT for some constant performance ratio. Second, we consider the approximability of circuit-succinct MAXSAT with respect to lower complexity classes: In particular, we prove that computing  $(2 - \epsilon)$ -approximate solutions for circuit-succinct MAXSAT is at least as hard as inverting one-way permutations. On the other hand, a simple randomized approximation algorithm computes a  $(2 + \epsilon)$ -approximate solution with high probability.

Recall that the standard (not succinctly represented) version of the MAXSAT problem is approximable to within a .78 factor [AW00] and that the MAX3SAT problem is approximable to within a 7/8 factor [KZ97].

## 1 Introduction

There are different concrete definitions of the problem CIRCUITMAXSAT in the literature. But all these approaches define an instance of CIRCUITMAXSAT as a circuit  $C$  describing a CNF-formula  $\phi = \text{expand}[C]$ . To compute  $\phi$ , it is usually necessary to enumerate all possible input vectors  $\bar{c}$  and to concatenate the corresponding outputs  $C(\bar{c})$ . The reason for this vague treatment of the problem definition is that each of these representations leads to **NEXP**-hardness of the associated succinct MAXSAT-problem (see for example [Pap94], Chapter 20).

We define CIRCUITMAXSAT such that its circuits work as functions mapping a clause number to a clause description. This functional behavior gives a polynomial time approximation algorithm access to a polynomially sized sample of the clauses contained within the instance, so this definition is a very “approximation-friendly” one.

### Definition 1 CircuitMaxSat

*An instance of CIRCUITMAXSAT is a circuit  $C$  which maps a bit string of length  $n$  to a bit string of length  $k(m+1)$  where  $n$ ,  $m$  and  $k$  are positive integers.  $C$  represents a CNF-formula  $\phi = \text{expand}[C]$  where the clause with number  $\bar{c}$  is described by*

$$C(\bar{c}) = \langle \bar{v}_1, s_1, \dots, \bar{v}_k, s_k \rangle$$

---

\*schallha@in.tum.de. Technische Universität München

†luca@cs.berkeley.edu. U. C. Berkeley.

with  $|\bar{c}| = n$  and  $|\bar{v}_i| = m$  for all  $1 \leq i \leq k$ . This description yields the clause  $l_1 \vee \dots \vee l_k$  where

$$l_i = \begin{cases} x_{\bar{v}_i} & \text{if } s_i = \text{TRUE} \\ \neg x_{\bar{v}_i} & \text{if } s_i = \text{FALSE} \end{cases}$$

Given a **CIRCUITMAXSAT**-instance  $C$ , the corresponding optimization problem is to find a truth assignment for  $\phi = \text{expand}[C]$  which maximizes the number of satisfied clauses. •

To interpret a circuit  $C$  in the way described above, it is necessary to know the integer  $k$ . However, we will only mention the circuit explicitly when we talk about a **CIRCUITMAXSAT**-instance and assume that the corresponding integer  $k$  is given implicitly to complete the instance description.

An algorithm is an  $r$ -approximation algorithm for **MAXSAT**, if this algorithm computes an assignment  $\tau$  on any CNF-formula  $\phi$  such that  $\text{opt}(\phi)/\mathbf{m}(\phi, \tau) \leq r$  holds, where  $\text{opt}(\phi)$  is the maximum number of simultaneously satisfiable clauses in  $\phi$  and  $\mathbf{m}(\phi, \tau)$  is the number of clauses in  $\phi$  which are satisfied by  $\tau$ . Similarly, an  $r$ -approximation algorithm for **CIRCUITMAXSAT** must compute the description of an assignment  $\tau$  with  $\text{opt}(\text{expand}[C])/\mathbf{m}(\text{expand}[C], \tau) \leq r$ . In case of an exponential time algorithm, the exponentially large assignment can be represented directly. However, a polynomial time algorithm must represent the assignment succinctly.

In Section 2, we prove that the **NP**-hardness for approximating **MAXSAT** can be lifted to a corresponding **NEXP**-hardness for approximating **CIRCUITMAXSAT** for some constant performance ratio. In Section 3, we prove that a simple randomized approximation algorithm computes a  $(2 + \epsilon)$ -approximate solution with high probability. On the other hand, we prove that computing  $(2 - \epsilon)$ -approximate solutions for circuit-succinct **MAXSAT** is at least as hard as inverting one-way permutations.

## 2 $r$ -Approximating CircuitMaxSat is NEXP-hard for some $r > 1$

The original proof for the **NP**-hardness of  $s$ -approximating **MAXSAT**, for some constant  $s > 1$ , relies directly on the **PCP** Theorem [ALM<sup>+</sup>98]. This proof uses the **PCP**-verifier as black box procedure. Similarly, we will use a characterization of **NEXP** in terms of a **PCP**-class and integrate the computation of the underlying verifier into a circuit. This will lead to a corresponding result for **CIRCUITMAXSAT**, i.e., **NEXP**-hardness of approximating **CIRCUITMAXSAT** within some other constant performance ratio  $r > 1$ . First recall the definition of probabilistic checkable proofs and the class **PCP**( $r(n), q(n)$ ):

### Definition 2 Probabilistic Checkable Proofs: The class **PCP**( $r(n), q(n)$ )

Given an input  $x$  of length  $n$  and a proof string  $\Pi$ , a **PCP**( $r(n), q(n)$ )-verifier  $V$  first reads  $s = \mathcal{O}(r(n))$  random bits. Using  $x$  and these random bits  $\bar{r}$ , the verifier computes  $t = \mathcal{O}(q(n))$  bits to be read from the proof string  $\Pi$ , i.e.,  $t$  positions of  $\Pi$  are read non-adaptively. Finally, based on  $x$ , the random bits  $\bar{r}$ , and the queried locations of  $\Pi$ , the verifier outputs its final verdict  $V_{\bar{r}}^x(\Pi)$ .  $V$  run in polynomial time.

A language  $L$  is in the class  $\mathbf{PCP}(r(n), q(n))$  if there is a  $\mathbf{PCP}(r(n), q(n))$ -verifier  $V$  such that

- if  $x \in L$ , then there is a proof  $\Pi$  such that the verifier accepts with probability 1, i.e.,

$$\exists \Pi : \Pr_{\bar{r} \in \{0,1\}^{r(n)}} [V_{\bar{r}}^x(\Pi) = \text{accept}] = 1$$

- if  $x \notin L$ , then the verifier accepts all possible proofs  $\Pi$  with a probability smaller than or equal to  $1/2$ , i.e.,

$$\forall \Pi : \Pr_{\bar{r} \in \{0,1\}^{r(n)}} [V_{\bar{r}}^x(\Pi) = \text{accept}] \leq \frac{1}{2}$$

•

We will use the following characterization of nondeterministic time computations to “lift” the  $\mathbf{NP}$ -hardness of approximating  $\mathbf{MAXSAT}$  to exponential time:

**Theorem 3 [Arora’s PhD Thesis [Aro94] Theorem 8.3]**

If  $t(n)$  is a function of at least polynomial growth, i.e,  $t(n) = \Omega(\text{poly}(n))$ ,

$$\mathbf{NTIME}(t(n)) = \mathbf{PCP}(\log t(n), 1)$$

•

This implies in particular that for each  $L \in \mathbf{NEXP}$  there is a constant  $c$ , such that  $L \in \mathbf{PCP}(n^c, 1)$ , which will be used as the starting point for the proof of the following theorem.

**Theorem 4**

There is a constant  $r > 1$  such that  $\mathbf{CIRCUITMAXSAT}$  is  $\mathbf{NEXP}$ -hard to approximate within performance ratio  $r$ .

•

**Proof:** Fix some language  $L \in \mathbf{NEXP}$ . Given an input  $x$ , we will construct a  $\mathbf{CIRCUITMAXSAT}$ -instance  $C_x$  such that an  $r$ -approximate assignment to  $\text{expand}[C_x]$  can be used to decide whether  $x \in L$  holds or not.

Assume without loss of generality that the  $\mathbf{PCP}(n^c, 1)$ -verifier for  $L$  uses precisely  $n^c$  random bits and queries at most  $q$  bits from the proof string  $\Pi$ . Corresponding to  $\Pi$  we define an assignment  $\tau$  such that  $\tau(x_i) = \text{TRUE}$  iff the  $i$ th bit of  $\Pi$  equals 1.

If  $x$  has length  $n$ , then there are  $N = 2^{n^c}$  choices for the random string  $\bar{r}$ , denoted as  $\bar{r}_1, \dots, \bar{r}_N$ . Corresponding to these choices we construct a set of 3CNF-formulas  $\phi_1, \dots, \phi_N$ . The result of the reduction on input  $x$  will be a circuit  $C_x$  such that  $\text{expand}[C_x] = \phi_x = \bigwedge_{i=1}^N \phi_i$ . Each formula  $\phi_i$  is defined as follows:

- $\phi_i$  has at most  $q$  variables. If the verifier queries the bits at positions  $p_1, \dots, p_q$  on random string  $\bar{r}_i$ ,  $\phi_i$  has the variables  $x_{p_1}, \dots, x_{p_q}$ .
- $\phi_i$  is satisfied by the assignment  $\tau$  iff the **PCP**-verifier would have accepted  $x$  together with the random string  $\bar{r}_i$  and the proof string  $\Pi$  represented by  $\tau$ .

This construction has the following implications:

- Each  $\phi_i$  involves at most  $q$  different variables, thus there exists a constant  $d$  such that no  $\phi_i$  contains more than  $d$  clauses. By repeating some clauses, we can fix the size of each  $\phi_i$  to exactly  $d$  clauses.
- If  $x \in L$ , we know that there exists a proof string  $\Pi$  such that the **PCP**-verifier accepts  $x$  and  $\Pi$  for all random strings  $\bar{r}_1, \dots, \bar{r}_N$ . Consequently, there is an assignment  $\tau$  such that all formulas  $\phi_1, \dots, \phi_N$  are satisfied by  $\tau$ .
- If  $x \notin L$  then only one half of the formulas  $\phi_1, \dots, \phi_N$  can be entirely satisfied at the same time, since any proof string  $\Pi$  is rejected by the verifier for at least half of the random inputs.

Summarized, for an input  $x \in L$  we get exactly  $dN$  clauses and all of them are satisfiable simultaneously. For an input  $x \notin L$  we get the same number of clauses but any assignment  $\tau$  can satisfy at most one half the formulas  $\phi_1, \dots, \phi_N$  entirely and thus in at least one half of the formulas  $\phi_1, \dots, \phi_N$  at least one clause remains unsatisfied. Therefore at most a fraction of  $\frac{1}{2} + \frac{1}{2} \frac{d-1}{d} = r' < 1$  of the clauses in  $\phi_x = \bigwedge_{i=1}^N \phi_i$  can be satisfied at the same time.

By simulating the **PCP**-verifier, we can generate on input  $x, i, j$  the description of the  $j$ th clause of  $\phi_i$  for the verification of input  $x$  in polynomial time. To get a **CIRCUITMAXSAT**-instance, we use this polynomial time algorithm, fix the input  $x$ , and transform this algorithm into a polynomially sized circuit with inputs  $i$  and  $j$ . This circuit computes the description of the corresponding clause. This transformation can be done in polynomial time and we obtain  $C_x$  with  $\text{expand}[C_x] = \phi_x$ .

If any assignment  $\tau$  satisfies more than a fraction  $r'$  of the clauses in  $\text{expand}[C_x]$  then  $x \in L$ . On the other hand, if  $x \in L$ , then  $\text{expand}[C_x]$  is satisfiable entirely. Thus it is **NEXP**-hard to  $r$ -approximate **CIRCUITMAXSAT** within any performance ratio  $r < 1/r'$ . •

### 3 Approximating CircuitMaxSat in Randomized Polynomial Time

#### 3.1 A Trivial Approximation Algorithm

Knowing that it is **NEXP**-hard to approximate **CIRCUITMAXSAT** within some constant  $r > 1$ , it is not surprising that we cannot compute good approximate solutions using only polynomial time or even randomized polynomial time. On the other hand, a random assignment which assigns each variable **TRUE** with probability  $1/2$  is expected to satisfy at least one half of all clauses. A useful probabilistic algorithm must compute with high probability a solution of guaranteed quality. This can be achieved by a simple application of sampling. We will describe a randomized polynomial time algorithm which produces with high probability a truth assignment satisfying almost half of all clauses. To talk about approximate algorithms for **CIRCUITMAXSAT** which run in polynomial time

we need a suitable definition of the solutions constructed by these algorithms since a solution  $\tau$  for a given instance  $C$  might be of exponential size. That is to say, we need a succinct representation for truth assignments:

**Definition 5 Succinct Assignments for CircuitMaxSat**

A succinct approximate solution for a CIRCUITMAXSAT-instance  $C$  is another circuit  $A$ , such that that the truth value of the variable  $x_{\bar{v}}$  is computed by  $A(\bar{v})$ , i.e.,  $A$  represents the assignment  $\tau$  with  $\tau(x_{\bar{v}}) = A(\bar{v})$ . •

The algorithm in figure 3.1 on page 6 outputs the all-TRUE assignment  $\tau(\text{TRUE})$  or the all-FALSE assignment  $\tau(\text{FALSE})$ . One of these assignments satisfies at least half of the clauses. The algorithm chooses the better assignment with high probability unless the gap between the quality of those two alternatives is very small.

**Theorem 6**

The algorithm in Figure 3.1 on the following page computes a  $(2 + \frac{1}{n})$ -approximate solution for any CIRCUITMAXSAT-instance  $C$  with a probability of  $1 - e^{-\mathcal{O}(n)}$ , where  $n$  denotes the number of input gates of  $C$ . •

**Proof:** To achieve a performance ratio of  $(2 + \frac{1}{n})$ , the algorithm has to satisfy at least a fraction of  $1/(2 + \frac{1}{n}) = \frac{1}{2} - \frac{1}{4n+2}$  of all clauses in  $\phi = \text{expand}[C]$ . The algorithm takes a sample of  $t = n^3$  clauses and chooses the assignment which satisfies more clauses in this sample. We can distinguish two cases:

- If the worse assignment satisfies at least a fraction of  $\frac{1}{2} - \frac{1}{4n+2}$  of the clauses, then we are done, since both choices are good enough.
- On the other hand if the worse assignment satisfies less than a fraction of  $\frac{1}{2} - \frac{1}{4n+2}$  of the clauses, then we can use the Chernoff-Bound. Let  $X_i = 1$  if the  $i$ th selected clause evaluates to TRUE under the worse assignment, then we get

$$\begin{aligned} \Pr \left[ \sum X_i \geq \frac{1}{2}n^3 \right] &\leq \exp \left( -\frac{(1/(4n+2))^2 n^3}{6} \right) \\ &= \exp(-\mathcal{O}(n)) \end{aligned}$$

So in both cases we get with probability of at least  $1 - \exp(-\mathcal{O}(n))$  a solution which satisfies a fraction  $\frac{1}{2} - \frac{1}{4n+2}$  or more of all clauses. Note that  $\tau(\text{TRUE})$  and  $\tau(\text{FALSE})$  are easy to represent

```

1: Input:
2:   a circuit  $C$  representing a CIRCUITMAXSAT-instance with  $n$  input gates
3:   representing a CNF-formula  $\phi = \text{expand}[C]$  with  $N = 2^n$  clauses
4: Output:
5:   an  $(2 - \frac{1}{n})$ -approximate assignment  $\tau = \text{approxcircmaxsat}(C)$ 
6:   for  $\phi$  with probability  $1 - \exp(\mathcal{O}(n))$ 
7: Executes:
8:    $\bar{c}_i \in_r \{0, 1\}^n$  for  $1 \leq i \leq t = n^3$ 
9:   compute the clause set  $S = \bigcup_{i=1}^t C(\bar{c}_i)$ 
10:  if  $m(\tau(\text{TRUE}), S) > m(\tau(\text{FALSE}), S)$  then
11:    return  $\tau(\text{TRUE})$ 
12:  else
13:    return  $\tau(\text{FALSE})$ 
14:  end if

```

Figure 1: Approximating CIRCUITMAXSAT: `approxcircmaxsat()`

by a circuit of polynomial size. •

### 3.2 The Limit of Randomized Polynomial Time Approximations

Now we turn to the hardness of approximating CIRCUITMAXSAT using randomized polynomial time algorithms. We will show a matching hardness result, i.e., the impossibility of computing approximate solutions which satisfy more than  $1/2$  of all clauses with high probability within polynomial time. We will construct an instance  $C$  such that  $\text{expand}[C]$  consists entirely of unit clauses, i.e., our hard instance will be a function  $f$  which associates with each clause number one single literal. Since our algorithm (and our solution which will be a circuit) can take small samples of the form  $\langle x, f(x) \rangle$ , it must be hard to draw conclusions about the general behavior of  $f$  based on these samples. In particular it must be hard to find out whether a given variable is used more often positively or negatively. So we arrive at one way functions as the fundamental hardness to be used here.

#### Definition 7 One-Way Functions

A family of functions  $f = \{f_n\}$ ,  $f_n : \{0, 1\}^n \rightarrow \{0, 1\}^n$  is called one way if:

- $f$  is computable in polynomial time
- for every family of polynomial circuits  $C = \{C_n\}$

$$\Pr_{\bar{x} \in \{0,1\}^n} [C_n(f_n(\bar{x})) \in f_n^{-1}(f_n(\bar{x}))] \leq \epsilon_n$$

- $\lim_{n \rightarrow \infty} \epsilon_n n^c = 0$  for every constant  $c$

•

#### Definition 8 One-Way Permutations

A one-way permutation  $f$  is a function that is both one way, and one to one. Therefore  $f^{-1}$  is also a function and the above inequality can be written as follows:

$$\Pr_{y \in \{0,1\}^n} [C_n(y) = f^{-1}(y)] \leq \epsilon_n$$

•

We will use one-way permutations to associate a clause number with a variable number, so each variable is used only in one clause implying that the instance is entirely satisfiable. Following the preceding argumentation, we must provide a way to compute the sign of each variable from the clause number, such that it is hard to compute this sign out of the variable number even if we are allowed to take a polynomial number of samples  $\langle x, f(x) \rangle$  into account. The next theorem provides exactly such a computational device.

#### Theorem 9 [Goldreich and Levin [GL89]]

If  $f$  is a one way permutation then for every family of polynomial circuits  $C = \{C_n\}$  and every constant  $d$ :

$$\Pr_{\bar{x}, \bar{y} \in \{0,1\}^n} [C_{2n}(\bar{x}, \bar{y}) = GL_n(f_n^{-1}(\bar{x}), \bar{y})] \leq \frac{1}{2} + \frac{1}{n^d}$$

where  $GL_n$  is the Goldreich-Levin function which is defined as

$$GL_n(\bar{x}, \bar{y}) \equiv \left( \sum_{1 \leq i \leq n} x_i y_i \right) \pmod{2}$$

•

Now we are ready to prove the following theorem:

**Theorem 10**

Under the assumption that one way permutations exist, for every constant  $\epsilon > 0$  and every polynomial  $p$ , there is a family of CIRCUITMAXSAT-instances  $\{C_n\}$ , for which no  $(2 - \epsilon)$ -approximate solutions of size  $p(|C_n|)$  exists. •

**Proof:** We define the circuit  $C_n$  in the following way:

$$\begin{aligned} C_n(\bar{x}, \bar{y}) &= (\bar{v}, s) \\ \text{where } \bar{v} &= (f_n(\bar{x}), \bar{y}) \\ s &= (GL_n(\bar{x}, \bar{y})) \end{aligned}$$

where  $\bar{x}, \bar{y}$  and  $\bar{v}$  are Boolean vectors of length  $n$  and  $s$  is a scalar.  $C_n(\bar{x}, \bar{y}) = (\bar{v}, s)$  means that the clause labeled with  $(\bar{x}, \bar{y})$  is a unit clause with variable  $x_{\bar{v}}$  and the sign  $s$  ( $s = \text{TRUE}$  means that the variable is positively used and  $s = \text{FALSE}$  that it is negated). This construction results in a CIRCUITMAXSAT-instance.

Given a polynomially sized circuit  $A$  which approximates  $C_n$  within  $2 - \epsilon$ , we can use this circuit to guess  $GL_n$ .

Since  $A$  satisfies a  $1/(2 - \epsilon) = 1/2 + \epsilon'$  fraction of all the clauses and all clauses are unit clauses, it must guess the sign of all the corresponding literals correctly. But this sign is determined by  $GL_n$ . Therefore

$$\Pr_{\bar{x}, \bar{y} \in \{0,1\}^n} [A(f_n(\bar{x}), \bar{y}) = GL_n(\bar{x}, \bar{y})] \geq \frac{1}{2} + \epsilon'$$

holds, which contradicts the assumption for large enough  $n$ . •

Since there are no polynomial sized succinct approximate solutions for CIRCUITMAXSAT-instances constructed in the way described above, there is no  $(2 - \epsilon)$ -approximation algorithm which runs in polynomial time, obtaining the next corollary.

**Corollary 11**

If one way permutations exist, it is impossible to approximate CIRCUITMAXSAT with a performance  $2 - \epsilon$  for every  $\epsilon > 0$  using randomized polynomial time. •

**References**

[ALM<sup>+</sup>98] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegdy. Proof verification and hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998.

- [Aro94] S. Arora. *Probabilistic Checking of Proofs and Hardness of Approximation Problems*. PhD thesis, CS Division, UC Berkeley, August 1994.
- [AW00] T. Asano and D. P. Williamson. Improved approximation algorithms for MAX-SAT. In *Proceedings of the 11th Symposium on Discrete Algorithms*, 2000.
- [GL89] O. Goldreich and L. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the 21st ACM Symposium on Theory of Computing*, pages 25–32, 1989.
- [KZ97] B. Karloff and U. Zwick. A  $(7/8 - \epsilon)$ -approximation algorithm for max3sat? In *Proceedings of the 29th ACM Symposium on Theory of Computing*, pages 406–415, 1997.
- [Pap94] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley Publishing Company, 1994.