Searching for circuit complexity lower bounds using SAT solvers

August Taylor

December 27, 2021

Abstract

Where the abstract will go

Contents

1 Introduction

1.1 Motivation

Finding lower bounds on the size of Boolean circuits computing given functions is a fundamental problem in computer science. Despite the fact that the majority of functions have exponential circuits, the best lower bounds proven on unrestricted circuits are only linear [2] and we still do not have optimal circuits or close upper and lower bounds for many important functions. Results in this area can be used to separate computational complexity classes, possibly even P and NP [1].

A heuristic approach is to find minimal circuits for small instances of the functions. This can be used to prove new bounds [6] and may lead to theoretical insights on the structure of their optimal circuits generally [9]. Finding efficient circuits is also necessary in practice when designing electronic systems.

One approach to automating the search for efficient circuits is to reduce the problem of designing a correct circuit (logical design synthesis) to the Boolean satisfiability problem (SAT). Existing algorithms (SAT solvers) can then be used to solve the problem, either finding a correct circuit of fixed size or generating a proof that none exists [5].

1.2 Previous work

Kamath et al [4] propose a reduction from logical design synthesis to SAT which fixes the DNF structure of the circuit and number of disjunctions used. Experiments using this reduction are reported in [3]. Kojevnikov et al [5] demonstrate a more general reduction allowing any circuit with a fixed number of gates. In [7] the same reduction was used to improve large circuits by searching for smaller versions of their sub-circuits.

*Mention valiant universal circuits?

*Should look for other things

1.3 Our contributions

We investigate feasible ways to find small circuits using SAT reductions. Previous research in this area has focused on a small number of target Boolean functions and a single reduction at a time. Instead we use multiple combinations of reductions and SAT solvers to see which leads to the best performance, testing them on a range of different functions.

We also present a new reduction *not sure what to say about this*

*Extension axioms - should actually try them first

2 Background

2.1 General setting

Denote by $B_{n,m}$ the set of all Boolean functions $f : \{0,1\}^n \to \{0,1\}^m$ and let $B_n = B_{n,1}$. A circuit over the basis $A \subseteq B_2$ is a directed acyclic graph with nodes of in-degree 0 or 2. Nodes of in-degree 0 are called inputs. Nodes of in-degree 2 are assigned functions from A and are called gates. Some nodes may be also be marked as outputs.

Without loss of generality, we can also assert that every gate has a larger index than both of its predecessors (i.e. the gates are sorted topologically w.r.t. the used numbering); that every gate's second predecessor has a larger index than its first predecessor; and that the last gate is an output. [5]

A circuit c of size N is said to compute the function $f \in B_{n,m}$ if, for all input vectors $v \in \{0,1\}^n$ in the range of f, $c(v) = f(v)$, where $c(v) = (o_1, ...o_m)$ are the values of the output gates of c, each input x_i takes value v_i, and each gate takes the value of its assigned function when applied to the value of its two predecessor nodes.

In this paper, we mainly consider circuits over the De Morgan basis

$$C_2 = \{\vee, \wedge, \neg\}$$

where $\neg$ denotes the Boolean function which outputs the negation of its first argument.

The size of a circuit is its number of gates. We define logical design synthesis as the problem of whether, given a truth table for a function $f : \{0,1\}^n \to \{0,1\}^m$, there exists a circuit of size N computing f.

2.2 Reduction to SAT

We encoded the logical design synthesis problem as a Boolean formula using three reductions.

2.2.1 Existing reductions

The first is due to Kojevnikov et al [5] who present it together with bounds on the growth rate of the formula. The basis of the circuit can be specified as any subset of B_2. The number of clauses generated is $O(N^3 \cdot 2^n)$ and the number of variables is $O(2^n \cdot N)$

The second is due to Razborov [8] and requires a circuit over the basis C_2: *not sure how much to describe this one but I definitely need to give the growth rates..*

Also am bothered that this is the only reduction that allows constants in the circuit - it wouldn't be hard to remove that aspect.

2.2.2 A new reduction

The third *Don't know how to credit Jan* also requires a circuit over the basis C_2.

It uses the following variables:

1. e^i_a $(n \leq i < N, a \in \{0,1\}^n)$ is the value of the i-th gate when the input to the circuit is a. It gives $O(N \cdot 2^n)$ variables.

2. $x_{i,j}$ $(0 \leq i < N+n, 0 \leq j < i)$ is true iff j is a predecessor of i. It gives $O(N \cdot N + n)$ variables.

3. g^0_i, g^1_i $(n \leq i < N)$ together encode the function assigned to the i-th gate by their truth values, giving $2N$ variables:

g^0_i	g^1_i	Function
0	0	$\neg$
0	1	$\vee$
1	0	$\wedge$
1	1	$\neg$

We encode the requirements for the circuit by the following clauses:

1. For all (i,j,k) with $n \leq i < n+N$, $0 \leq j < i$ and $j < k < i$, and for all $a \in \{0,1\}^n$, if j and k are both predecessors of i, then e^i_a must take the value computed by the ith gate's assigned function on e^j_a and e^k_a. A clause

must be added for each function in the basis. For example, the clause for when the ith gate is an $\wedge$-gate:

$$(g_i^0 \wedge \neg g_i^1) \wedge (x_{i,j} \wedge x_{i,k}) \rightarrow (e_a^i \leftrightarrow e_a^j \wedge e_a^k)$$

(Note that the gate must compute the conjunction of any two predecessors' variables, so it can have a maximum of two predecessors.) This leads to $O(2^n \cdot N^3)$ 7-clauses.

2. For all i with $n \leq i < n + N$, gate i must have at least one predecessor. This leads to $O(N)$ $O(N)$-clauses.

3. For all (i, j) with $n \leq i < n+N$, $0 \leq j < i$, if i is an $\vee$-gate or $\wedge$-gate with j its predecessor, it must have another distinct predecessor (i.e. it must have at least 2 overall). For example, the clause for when the ith gate is an $\wedge$-gate:

$$(g_i^0 \wedge \neg g_i^1) \rightarrow \left(x_{i,j} \rightarrow \bigvee_{j<k<i} x_{i,k} \right)$$

This leads to $O(N^2)$ $O(N)$-clauses.

4. For all i with $0 \leq i < n$, $a \in \{0,1\}^n$, the i-th input must equal a_i. This leads to $O(n \cdot 2^n)$ 1-clauses.

5. For all $a \in \{0,1\}^n$, the final m gates (i.e. the outputs) should agree with the output values given for a in the target truth table. This leads to $O(2^n)$ 1-clauses.

2.3 Target Boolean functions

2.4 Extension axioms

3 Methodology

4 Results

5 Discussion

5.1 Reflection

5.2 Conclusions

5.3 Further directions

References

[1] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach.* Cambridge University Press, 2009.

[2] Ravi B Boppana and Michael Sipser. The complexity of finite functions. In *Algorithms and complexity*, pages 757–804. Elsevier, 1990.

[3] Giovani Gomez Estrada. A note on designing logical circuits using sat. In *International Conference on Evolvable Systems*, pages 410–421. Springer, 2003.

[4] AP Kamath, NK Karmarkar, KG Ramakrishnan, and MGC Resende. An interior point approach to boolean vector function synthesis. In *Proceedings of 36th Midwest Symposium on Circuits and Systems*, pages 185–189. IEEE, 1993.

[5] Arist Kojevnikov, Alexander S Kulikov, and Grigory Yaroslavtsev. Finding efficient circuits using sat-solvers. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 32–44. Springer, 2009.

[6] Alexander S Kulikov. Improving circuit size upper bounds using sat-solvers. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 305–308. IEEE, 2018.

[7] Alexander S. Kulikov and Nikita Slezkin. Sat-based circuit local improvement, 2021.

[8] Alexander A Razborov. Pseudorandom generators hard for k-dnf resolution and polynomial calculus resolution. *Annals of Mathematics*, pages 415–472, 2015.

[9] Ryan Williams. Applying practice to theory, 2008.