Report

Amit Roy
CS18S022

Mystery of negations

Contents

1 Introduction

A monotone boolean function can be computed by a circuit without using any negation gates. There are hard functions like $CLIQUE$ which can be computed by an exponential size monotone circuit. There are a large class of boolean functions called the slice function for which monotone and non-monotone circuits have the same efficiency. The presence of negations in a circuit is the main difficulty in proving circuit lower bounds. We try to find answers to the following

1. What is the minimum number of negations required to compute any function f?
2. To what extent we can decrease the number of negations in a circuit without substantial increase in the circuit size?
3. What is the minimum number of negations required for superpolynomial saving in the size of the circuit?

2 Background

A circuit having no negation gates is called monotone circuit. The function computed by such circuit is called monotone function.

Monotonocity

We say $x \leq y$ if $x_i \leq y_i$ for all $i \in \{1, \ldots, n\}$. A function f is said to be monotone if for all $x \leq y \Leftrightarrow f(x) \leq f(y)$. A chain in the boolean hypercube is an increasing sequence $y^1 < y^2 < \ldots < y^k$ where $y^i \in \{0,1\}^n$ and y^i and y^{i+1} differs in only 1 bit.

Decrease

The decrease $d_Y(f)$ of a function f on a chain Y is the number of indices i in the chain such that $f(y^i) > f(y^{i+1})$. The decrease $d(f)$ of a function f is the maximum value of $d_Y(f)$ over all chains Y.

We define $M(f)$ to be the minimum number of negations required to compute a function f. Suppose that a function f can be computed by a circuit in polynomial size but with $M(f)$ negations the circuit requires superpolynomial size. What is then minimum number of negations $R(f)$ sufficient to compute f in polynomial size?

Connector Function

Let $\mu(i, i', x)$ be a boolean function on $n+2$ variables. μ is a connector function of two boolean functions f_0 and f_1 if $\mu(0,1,x) = f_0(x)$ and $\mu(1,0,x) = f_1(x)$

3 Main Results

Markov had an interesting observation about number of negations required to compute any boolean function.

Theorem 1 (Markov 1957)**.** *For every function f , the number of negations required in the circuit computing it is $M(f) = \lceil \log(d(f)+1) \rceil$ where $d(f)$ is the decrease of the function f.*

Proof Sketch : For the ***l*ower bound** i.e. $neg(f) \geq \lceil \log(d(f)+1) \rceil$ the idea is to keep removing the each negations from the circuit by replacing with a constant 0 or 1 based on the decrease $d(f)$ value , until it becomes a monotone circuit and then in the end count the decrease $d(f)$ again.

For the **upper bound** i.e. $neg(f) \leq \lceil \log(d(f)+1) \rceil$, the proof is based on induction on

$$M(f) = \lceil \log(d(f)+1) \rceil$$

Observe that the value $M(f)$ is on logarithmic scale. So to obtain a function f' with $M(f') = M(f) - 1$, we need to have the decrease value $d(f')$ half of the decrease value $d(f)$ of f. The idea is to construct a function f' with $M(f') = M(f) - 1$ to apply the induction hypothesis. We in fact construct two functions f_0 and f_1 which in some sense retains the property of function f i.e. on one half of the boolean hypercube it is equal to f and on other it is a constant. By induction hypothesis the $M(f_i)$ value will be $M(f) - 1$. The next idea is to combine these 2 functions cleverly to obtain the function f without substantial increase in the number of negations used. We call this connector function μ which connects f_0 and f_1 to obtain f and the number of negations used in the connector function is at most $max\{neg(f_0), neg(f_1)\}$. Finally we will need just one extra negation gate to compute f using the connector function μ. Hence $neg(f) \leq 1 + max\{neg(f_0), neg(f_1)\}$ which is $M(f)$.

We would need the following lemma for the completeness of the above proof.

Lemma 1 (Connector Function)**.** *For every function f_0 and f_1 we have a connector function μ which uses at most $max\{neg(f_0), neg(f_1)\}$.*

Theorem 2 (Fischer 1974)**.** *If a function $f : \{0,1\}^n \to \{0,1\}$ is computed by a circuit of size t , then restricting the circuit to use only $\lceil \log(n+1) \rceil$ negations entails only polynomial blow up in size which is atmost $2t + \mathcal{O}(n^2 \log^2 n)$.*

Proof Sketch : We bring down all the negations of the circuit to the inputs which makes the size of the circuit $2t$. Now our new goal is to show number of negations for the following circuit

$$NEG(x_1, \ldots, x_n) = (\neg x_1, \ldots, \neg x_n)$$

is $\mathcal{O}(n^2 \log^2 n)$.
We can observe that $\neg x_i = \bigwedge_{k=0}^{n} (\neg T_k^n(x)) \vee (T_{k,i}^n(x))$ where T_k^n is the threshold function and $T_{k,i}^n = T_k^{n-1}(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$. Now designing circuits for threshold function can be computed by a circuit of size $\mathcal{O}(n^2 \log^2 n)$.

4 Improvement in Fischer's Simulation

Suppose there is a function f which can be computed by a circuit of polynomial size but restricting it to use only $M(f)$ many negations make the circuit superpolynomial size. What

is then the minimum number of negations $R(f)$ required sufficient to compute f by polynomial size circuit. From Markov's and Fischer's result we have that the value $R(f)$ is between 0 and $\log n$.

Berkowitz and Valiant showed that there are a type of boolean functions called slice functions for which negation is powerless. In other words we don't have any superpolynomial savings. This raises the question that whether using NOT gates at all have superpolynomial saving in the size of the circuit?

Razborov solved this problem giving explicit monotone boolean function f which will have superpolynomial saving with use of negations. This function is characteristic function of bipartite graphs containing a perfect matching. Another example was given by Tardos. So the next natural question is: *how large is R(f)?* This question was considered by many authors with some restriction on type of circuit and use of NOT gates.

1. Okolonishnikova and Ajtai and Guruvich have shown that there exists monotone functions which can be computed with polynomial size , constant depth circuits but can't be computed by monotone polynomial size, constant depth circuit.

2. Satha and Wilson showed that for AC^0 circuits we need much more than $\lceil \log(n+1) \rceil$ negations by showing a multi output function which cannot be computed in constant depth with $o(\frac{n}{\log^{1+\epsilon} n})$ negations.

3. For logarithmic depth circuit, a lower bound of $R(f) = \Omega(n)$ was proved by Raz and Wigderson under the restriction that all negations are at the inputs.

Razborov in 1985 showed that we cannot have better than exponential size for $CLIQUE$ function computed by a monotone circuit. Amano and Maruoka in 2005 showed a stronger result showing that even using $\frac{1}{6} \log \log n$ negations , circuit computing $CLIQUE$ will have exponential size.

5 Conclusion

We saw several results on value of $R(f)$ when there were restriction on use of NOT gates or type of the circuit. However if we remove the restriction then only thing we know so far is that some functions requires $R(f) > 0$ many negations. Surprisingly the results of Razborov and Markov is used to move the threshold $R(f)$ closer to Markov's bound $\log n$.

Jukna in 2004 showed that there exists explicit monotone functions $f_n : \{0,1\}^n \to \{0,1\}^n$ such that $R(f_n) \geq \log n - 9 \log \log n$. That is if we move by more than $\log \log n$ from $\log n$ then we would have moved to the world of monotone circuit. But the function here is not a decision problem. We don't know about such bounds in the decision world. It would be really interesting to find an explicit monotone boolean function for which $R(f) = \Omega(\log n)$ and this is the research problem this paper motivates.