A Superpolynomial Lower Bound for Clique Function Circuits with at most $\frac{1}{6}\log\log n$ Negation Gates

Kazuyuki Amano and Akira Maruoka

Sajin Koroth

Department of Computer Science
IIT Madras

Advanced Complexity Theory Course - Spring 2012

Outline

1. Main Result : C with at most $\frac{1}{6}\log\log n$ negations requires $2^{\frac{1}{5}(\log m)^{(\log m)^{\frac{1}{2}}}}$ gates for detecting cliques of size $(\log m)^{3(\log m)^{\frac{1}{2}}}$ in a graph with m vertices
2. A better monotone lower bound for clique function : (using bottleneck counting) $\exp^{\Omega\left(m^{\frac{1}{3}}\right)}$ for appropriate parameters

Are we there yet ? (NP $\not\subset$ P/poly)

- Thanks to Fischer Limited Negation lower bounds for circuits with at most $\log n$ negations would do
- The monotone lowerbounds could be misleading : Perfect matching inspite of being in P has a superpolynomial lowerbound for monotone circuits

1. From the limited-negation non-monotone circuit for clique obtain an "appropriate" family of monotone functions
2. Obtain lowerbounds for each function in the family (which approximates clique function)
3. Use the lowerbounds on each function in the family to extend the lowerbound to any such family
4. Transfer this lowerbound to the limited-negation circuit for clique.

1. From the limited-negation non-monotone circuit for clique obtain an "appropriate" family of monotone functions
2. Obtain lowerbounds for each function in the family (which approximates clique function)
3. Use the lowerbounds on each function in the family to extend the lowerbound to any such family
4. Transfer this lowerbound to the limited-negation circuit for clique.

1. From the limited-negation non-monotone circuit for clique obtain an "appropriate" family of monotone functions
2. Obtain lowerbounds for each function in the family (which approximates clique function)
3. Use the lowerbounds on each function in the family to extend the lowerbound to any such family
4. Transfer this lowerbound to the limited-negation circuit for clique.

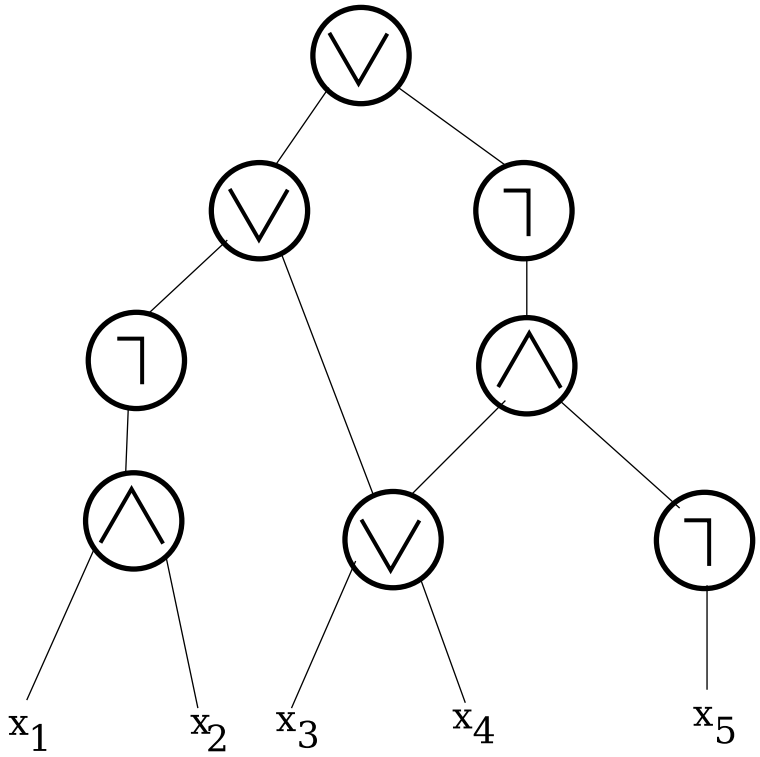
1. From the limited-negation non-monotone circuit for clique obtain an "appropriate" family of monotone functions
2. Obtain lowerbounds for each function in the family (which approximates clique function)
3. Use the lowerbounds on each function in the family to extend the lowerbound to any such family
4. Transfer this lowerbound to the limited-negation circuit for clique.

Outline

Idea

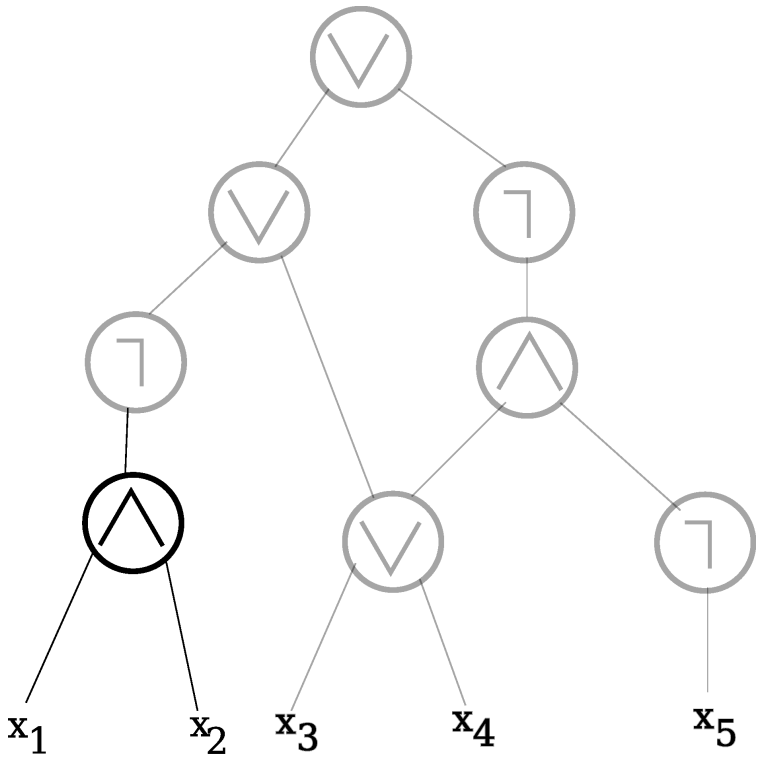
Use restrictions to eliminate negation gates by fixing their outputs

Figure: Motone Circuit's from Non-monotone one's using restrictions



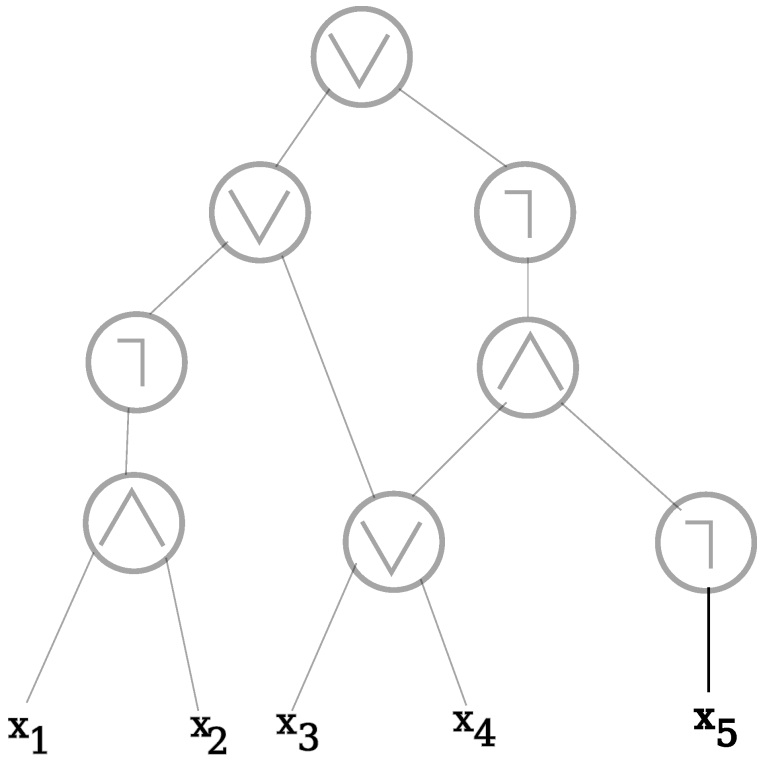
u=

Figure: Motone Circuit's from Non-monotone one's using restrictions



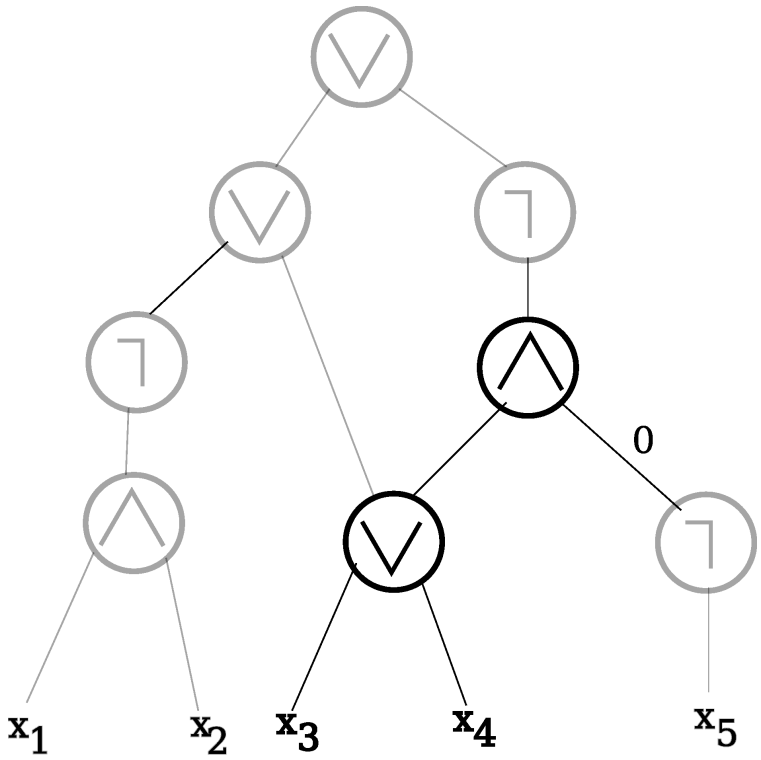
u=

Figure: Motone Circuit's from Non-monotone one's using restrictions



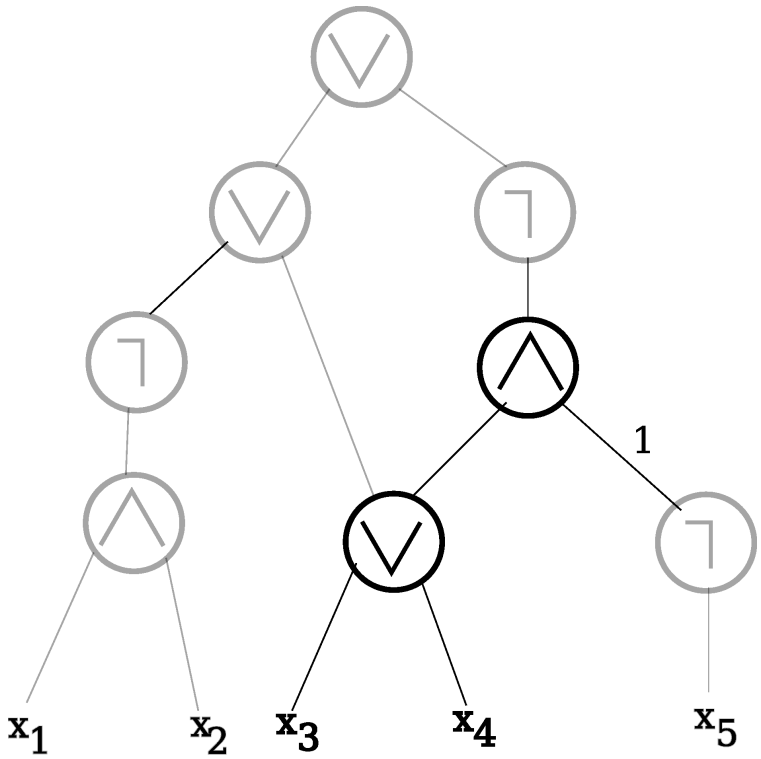
u= 0

Figure: Motone Circuit's from Non-monotone one's using restrictions



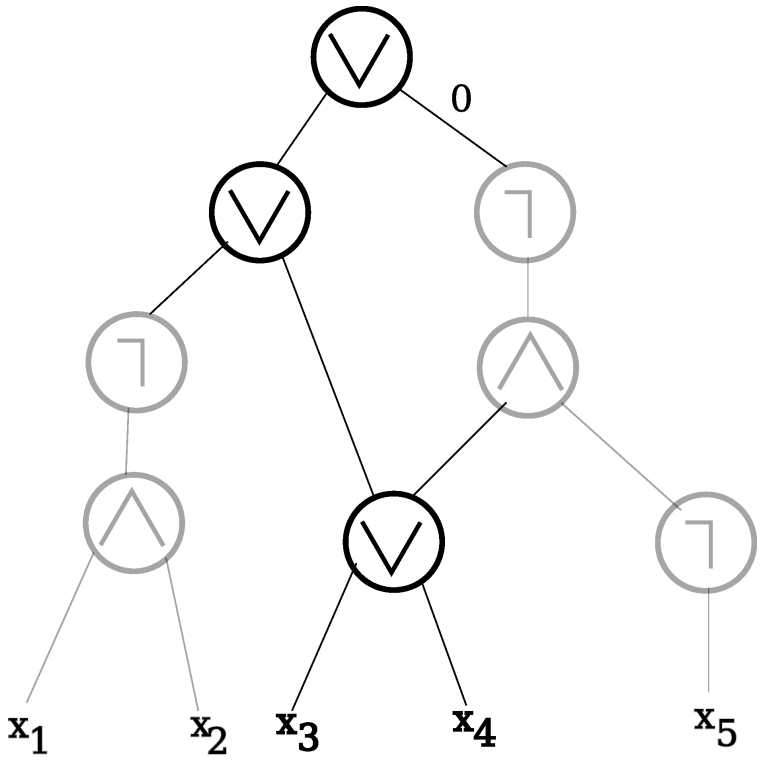
u= 00

Figure: Motone Circuit's from Non-monotone one's using restrictions



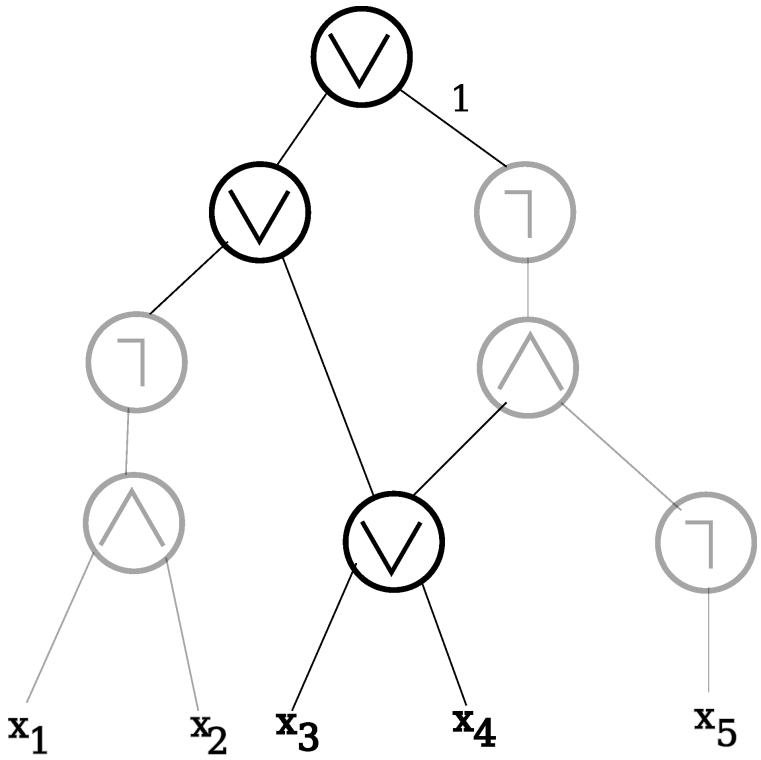
u= 01

Figure: Motone Circuit's from Non-monotone one's using restrictions



u= 000

Figure: Motone Circuit's from Non-monotone one's using restrictions



u= 001

Some Notation

- Negation gates appearing in the minimal at most t-negations circuit C will be labelled $g_1, \ldots, g_t$ (labelled in a topological sorting order of DAG C.
- The output gate of the circuit will be labelled g_{t+1}
- The functions corresponding to restrictions will be labelled f_i, $1 \leq i \leq 2^t$.
- For example f_3 corresponds to the restriction 11, i.e. $g_1 = 1, g_2 = 1$
- f_ε would correspond to no restrictions to NOT gates, which corresponds to the function evaluated at gate which is the input to g_1

Some Notation

- Negation gates appearing in the minimal at most t-negations circuit C will be labelled $g_1, \ldots, g_t$ (labelled in a topological sorting order of DAG C.
- The output gate of the circuit will be labelled g_{t+1}
- The functions corresponding to restrictions will be labelled f_i, $1 \leq i \leq 2^t$.
- For example f_3 corresponds to the restriction 11, i.e. $g_1 = 1, g_2 = 1$
- f_ε would correspond to no restrictions to NOT gates, which corresponds to the function evaluated at gate which is the input to g_1

Some Notation

- Negation gates appearing in the minimal at most t-negations circuit C will be labelled $g_1, \ldots, g_t$ (labelled in a topological sorting order of DAG C.
- The output gate of the circuit will be labelled g_{t+1}
- The functions corresponding to restrictions will be labelled f_i, $1 \leq i \leq 2^t$.
- For example f_3 corresponds to the restriction 11, i.e. $g_1 = 1, g_2 = 1$
- f_ε would correspond to no restrictions to NOT gates, which corresponds to the function evaluated at gate which is the input to g_1

Some Notation

- Negation gates appearing in the minimal at most t-negations circuit C will be labelled $g_1, \ldots, g_t$ (labelled in a topological sorting order of DAG C.
- The output gate of the circuit will be labelled g_{t+1}
- The functions corresponding to restrictions will be labelled f_i, $1 \leq i \leq 2^t$.
- For example f_3 corresponds to the restriction 11, i.e. $g_1 = 1, g_2 = 1$
- f_ε would correspond to no restrictions to NOT gates, which corresponds to the function evaluated at gate which is the input to g_1

Outline

Properties of these functions

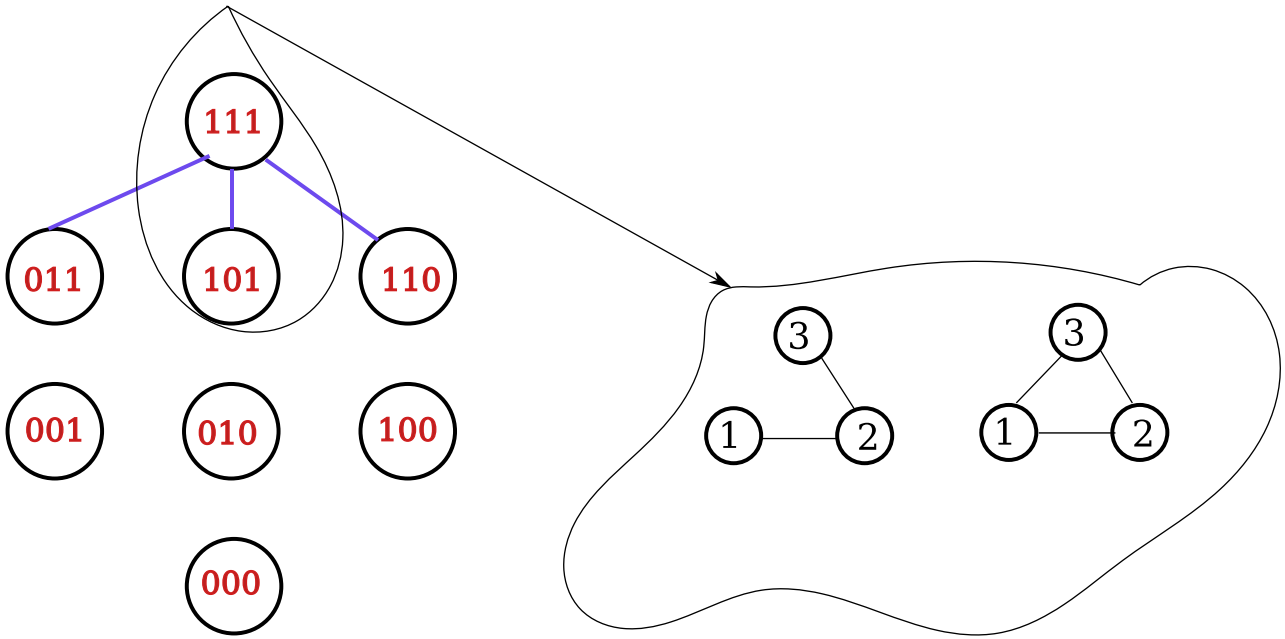
- They are monotone
- There are $1 + \sum_{i=1}^{t} 2^i = 2^{t+1} - 1$ of them
- Union of their sensitive graphs is a super graph of sensitive graph of f the original function

Properties of these functions

- They are monotone
- There are $1 + \sum_{i=1}^{t} 2^i = 2^{t+1} - 1$ of them
- Union of their sensitive graphs is a super graph of sensitive graph of f the original function

What the hell is a sensitive graph ?

Figure: Sensitve graph for Clique function



Sensitivity graph for clique's of size 3
Input ordering : (1,2),(1,3),(2,3)

Sensitive graphs of restrictions cover sensitive graph of f

Theorem

$\bigcup_{f_i} G_{f_i} \supseteq G_f$

- Take a $(w, w') \in G_f$, i.e. $f(w) = 0$ and $f(w') = 1$
- Look at $f_i(w)$ vs $f_i(w')$ starting from $i = 1$
- If $f_\varepsilon(w) = f_\varepsilon(w')$ then the sub-circuit of C rooted at input to g_1 evaluates to same on w, w'
- If $\forall i < j$, $f_i(w) = f_i(w)$, then the sub-circuit of C rooted at input to g_i evaluates to the same on w, w'
- If all $f_i(w) = f_i(w')$ then we get that $f_{t+1}(w) = f_{t+1}(w)$ then sub-circuit of C rooted at g_{t+1} evaluates to the same on w, w'. But....
- This is a contradiction, g_{t+1} is the root gate of C.

Sensitive graphs of restrictions cover sensitive graph of f

Theorem

$\bigcup_{f_i} G_{f_i} \supseteq G_f$

- Take a $(w, w') \in G_f$, i.e. $f(w) = 0$ and $f(w') = 1$
- Look at $f_i(w)$ vs $f_i(w')$ starting from $i = 1$
- If $f_\varepsilon(w) = f_\varepsilon(w')$ then the sub-circuit of C rooted at input to g_1 evaluates to same on w, w'
- If $\forall i < j$, $f_i(w) = f_i(w)$, then the sub-circuit of C rooted at input to g_i evaluates to the same on w, w'
- If all $f_i(w) = f_i(w')$ then we get that $f_{t+1}(w) = f_{t+1}(w)$ then sub-circuit of C rooted at g_{t+1} evaluates to the same on w, w'. But....
- This is a contradiction, g_{t+1} is the root gate of C.

Sensitive graphs of restrictions cover sensitive graph of f

Theorem

$\bigcup_{f_i} G_{f_i} \supseteq G_f$

- Take a $(w, w') \in G_f$, i.e. $f(w) = 0$ and $f(w') = 1$
- Look at $f_i(w)$ vs $f_i(w')$ starting from $i = 1$
- If $f_\varepsilon(w) = f_\varepsilon(w')$ then the sub-circuit of C rooted at input to g_1 evaluates to same on w, w'
- If $\forall i < j$, $f_i(w) = f_i(w)$, then the sub-circuit of C rooted at input to g_i evaluates to the same on w, w'
- If all $f_i(w) = f_i(w')$ then we get that $f_{t+1}(w) = f_{t+1}(w)$ then sub-circuit of C rooted at g_{t+1} evaluates to the same on w, w'. But....
- This is a contradiction, g_{t+1} is the root gate of C.

Sensitive graphs of restrictions cover sensitive graph of f

Theorem

$\bigcup_{f_i} G_{f_i} \supseteq G_f$

- Take a $(w, w') \in G_f$, i.e. $f(w) = 0$ and $f(w') = 1$
- Look at $f_i(w)$ vs $f_i(w')$ starting from $i = 1$
- If $f_\varepsilon(w) = f_\varepsilon(w')$ then the sub-circuit of C rooted at input to g_1 evaluates to same on w, w'
- If $\forall i < j$, $f_i(w) = f_i(w)$, then the sub-circuit of C rooted at input to g_i evaluates to the same on w, w'
- If all $f_i(w) = f_i(w')$ then we get that $f_{t+1}(w) = f_{t+1}(w)$ then sub-circuit of C rooted at g_{t+1} evaluates to the same on w, w'. But....
- This is a contradiction, g_{t+1} is the root gate of C.

Sensitive graphs of restrictions cover sensitive graph of f

Theorem

$\bigcup_{f_i} G_{f_i} \supseteq G_f$

- Take a $(w, w') \in G_f$, i.e. $f(w) = 0$ and $f(w') = 1$
- Look at $f_i(w)$ vs $f_i(w')$ starting from $i = 1$
- If $f_\varepsilon(w) = f_\varepsilon(w')$ then the sub-circuit of C rooted at input to g_1 evaluates to same on w, w'
- If $\forall i < j$, $f_i(w) = f_i(w)$, then the sub-circuit of C rooted at input to g_i evaluates to the same on w, w'
- If all $f_i(w) = f_i(w')$ then we get that $f_{t+1}(w) = f_{t+1}(w)$ then sub-circuit of C rooted at g_{t+1} evaluates to the same on w, w'. But....
- This is a contradiction, g_{t+1} is the root gate of C.

Sensitive graphs of restrictions cover sensitive graph of f

Theorem

$\bigcup_{f_i} G_{f_i} \supseteq G_f$

- Take a $(w, w') \in G_f$, i.e. $f(w) = 0$ and $f(w') = 1$
- Look at $f_i(w)$ vs $f_i(w')$ starting from $i = 1$
- If $f_\varepsilon(w) = f_\varepsilon(w')$ then the sub-circuit of C rooted at input to g_1 evaluates to same on w, w'
- If $\forall i < j$, $f_i(w) = f_i(w)$, then the sub-circuit of C rooted at input to g_i evaluates to the same on w, w'
- If all $f_i(w) = f_i(w')$ then we get that $f_{t+1}(w) = f_{t+1}(w)$ then sub-circuit of C rooted at g_{t+1} evaluates to the same on w, w'. But....
- This is a contradiction, g_{t+1} is the root gate of C.

Outline

Putting it all together

Corollary

For any positive integer t and $\alpha = 2^{t+1} - 1$ the following is true

$$size_t(f) \geq \min_{F' = \{f_1, \ldots, f_\alpha\} \subseteq \mathscr{M}^n} \left\{ \max_{f \in F'} \left\{ size_0(f') \right\} \mid \bigcup_{f' \in F'} G(f') \supseteq G(f) \right\}$$

- They are monotone $\implies$ $\mathscr{M}^n$ is all n-input monotone functions
- There are $1 + \sum_{i=1}^{t} 2^i = 2^{t+1} - 1$ of them $\implies$ $\alpha = 2^{t+1} - 1$
- Union of their sensitive graphs is a super graph of sensitive graph of f $\implies$ $\bigcup_{f' \in F'} G(f') \supseteq G(f)$

Outline

- A better lowerbound for a special setting of parameters : $\exp\left(\Omega\left(m^{\frac{1}{3}}\right)\right)$ vs $\exp\left(\Omega\left(\frac{m}{\log m}\right)^{\frac{1}{3}}\right)$
- Uses only elementary combinatorics - Sun Flower Lemma not required

Outline of the proof

- Define appropriate approximations to AND and OR gates
- Prove that the approximated circuit makes a lot of errors
- Prove that approximated individual gates make only a small number of errors
- Hence there must be a large number of gates in the original circuit

Outline of the proof

- Define appropriate approximations to AND and OR gates
- Prove that the approximated circuit makes a lot of errors
- Prove that approximated individual gates make only a small number of errors
- Hence there must be a large number of gates in the original circuit

Outline of the proof

- Define appropriate approximations to AND and OR gates
- Prove that the approximated circuit makes a lot of errors
- Prove that approximated individual gates make only a small number of errors
- Hence there must be a large number of gates in the original circuit

Outline

Some Notation

Definition

End-Point SetLet t be a term or a clause. The **end-point** set of t is a set of all end-points of the edges in t. The **size(t)** is the cardinality of end-point set of t

- Circuit is layered and alternates between $\wedge, \vee$ gate layers
- Output gate is an $\wedge$ gate
- Let an OR gate be fed by two approximators represented by monotone DNF formulae's f_1^D, f_2^D.
- Approximate OR : Convert the DNF formula $f_1^D \vee f_2^D$ to a CNF formula and take away all clauses whose size exceed r.
- Let an AND gate be fed by two approximators represented by monotone CNF formulae's f_1^C, f_2^C.
- Approximate AND : Convert the CNF formula $f_1^C \wedge f_2^C$ to a DNF formula and take away all terms whose size exceed l.

- Circuit is layered and alternates between $\wedge, \vee$ gate layers
- Output gate is an $\wedge$ gate
- Let an OR gate be fed by two approximators represented by monotone DNF formulae's f_1^D, f_2^D.
- Approximate OR : Convert the DNF formula $f_1^D \vee f_2^D$ to a CNF formula and take away all clauses whose size exceed r.
- Let an AND gate be fed by two approximators represented by monotone CNF formulae's f_1^C, f_2^C.
- Approximate AND : Convert the CNF formula $f_1^C \wedge f_2^C$ to a DNF formula and take away all terms whose size exceed l.

- Circuit is layered and alternates between $\wedge, \vee$ gate layers
- Output gate is an $\wedge$ gate
- Let an OR gate be fed by two approximators represented by monotone DNF formulae's f_1^D, f_2^D.
- Approximate OR : Convert the DNF formula $f_1^D \vee f_2^D$ to a CNF formula and take away all clauses whose size exceed r.
- Let an AND gate be fed by two approximators represented by monotone CNF formulae's f_1^C, f_2^C.
- Approximate AND : Convert the CNF formula $f_1^C \wedge f_2^C$ to a DNF formula and take away all terms whose size exceed l.

Why these approximations ?

- Taking away terms from a DNF formula makes it accept possibly less number of inputs
- Taking away clauses from a CNF formula makes it accept possibly more number of inputs
- If there is a good graph on which $\bar{C}$ outputs 0 then there exists $\bar{\wedge}$ which outputs 0.
- If there is a bad graph on which $\bar{C}$ outputs 1 then there exists $\bar{\vee}$ which outputs 1.
- $(f_1 \bar{\vee} f_2) \geq (f_1 \vee f_2)$ and $(f_1 \bar{\wedge} f_2) \leq (f_1 \wedge f_2)$

Why these approximations ?

- Taking away terms from a DNF formula makes it accept possibly less number of inputs
- Taking away clauses from a CNF formula makes it accept possibly more number of inputs
- If there is a good graph on which $\bar{C}$ outputs 0 then there exists $\bar{\wedge}$ which outputs 0.
- If there is a bad graph on which $\bar{C}$ outputs 1 then there exists $\bar{\vee}$ which outputs 1.
- $(f_1 \bar{\vee} f_2) \geq (f_1 \vee f_2)$ and $(f_1 \bar{\wedge} f_2) \leq (f_1 \wedge f_2)$

Why these approximations ?

- Taking away terms from a DNF formula makes it accept possibly less number of inputs
- Taking away clauses from a CNF formula makes it accept possibly more number of inputs
- If there is a good graph on which $\bar{C}$ outputs 0 then there exists $\bar{\wedge}$ which outputs 0.
- If there is a bad graph on which $\bar{C}$ outputs 1 then there exists $\bar{\vee}$ which outputs 1.
- $(f_1 \bar{\vee} f_2) \geq (f_1 \vee f_2)$ and $(f_1 \bar{\wedge} f_2) \leq (f_1 \wedge f_2)$

Outline

Good and Bad Graphs

- Good graph : Clique on s vertices
- Bad graph : A complete $s-1$ partite graph

Proof

An approximator either identically outputs 0 or outputs 1 on at least half the fraction of bad graphs

- Output of C is an $\wedge$, hence is approximators ouput is represented by a DNF of whose terms are of size at most l
- If it is identically 0 then we are done, otherwise there exists a term t whose size is at most l and $\bar{f} \geq t$ holds.
- Reperesent bad graphs using a bijection from $\{v_1, \ldots, v_m\}$ to $\{(1,1), \ldots, (1, \frac{m}{s-1}), \ldots, (s-1,1), \ldots, (s-1, \frac{m}{s-1})\}$
- Term t outputs 0 iff there is a variable (u, v) such that u and v under the bijection are assigned to tuples whose first component differs
- The number of bad graphs where this happens is at most $\binom{l}{2}\left(\frac{\binom{m}{s-1}}{m}\right)$.

Proof

An approximator either identically outputs 0 or outputs 1 on at least half the fraction of bad graphs

- Output of C is an $\wedge$, hence is approximators ouput is represented by a DNF of whose terms are of size at most l
- If it is identically 0 then we are done, otherwise there exists a term t whose size is at most l and $\bar{f} \geq t$ holds.
- Reperesent bad graphs using a bijection from $\{v_1, \ldots, v_m\}$ to $\{(1,1), \ldots, (1, \frac{m}{s-1}), \ldots, (s-1,1), \ldots, (s-1, \frac{m}{s-1})\}$
- Term t outputs 0 iff there is a variable (u, v) such that u and v under the bijection are assigned to tuples whose first component differs
- The number of bad graphs where this happens is at most $\binom{l}{2}\left(\frac{\binom{m}{s-1}}{m}\right)$.

Proof

An approximator either identically outputs 0 or outputs 1 on at least half the fraction of bad graphs

- Output of C is an $\wedge$, hence is approximators ouput is represented by a DNF of whose terms are of size at most l
- If it is identically 0 then we are done, otherwise there exists a term t whose size is at most l and $\bar{f} \geq t$ holds.
- Reperesent bad graphs using a bijection from $\{v_1, \ldots, v_m\}$ to $\{(1,1), \ldots, (1, \frac{m}{s-1}), \ldots, (s-1, 1), \ldots, (s-1, \frac{m}{s-1})\}$
- Term t outputs 0 iff there is a variable (u, v) such that u and v under the bijection are assigned to tuples whose first component differs
- The number of bad graphs where this happens is at most $\binom{l}{2}\left(\frac{\binom{m}{s-1}}{m}\right)$.

Proof

An approximator either identically outputs 0 or outputs 1 on at least half the fraction of bad graphs

- Output of C is an $\wedge$, hence is approximators ouput is represented by a DNF of whose terms are of size at most l
- If it is identically 0 then we are done, otherwise there exists a term t whose size is at most l and $\bar{f} \geq t$ holds.
- Reperesent bad graphs using a bijection from $\{v_1, \ldots, v_m\}$ to $\{(1,1), \ldots, (1, \frac{m}{s-1}), \ldots, (s-1, 1), \ldots, (s-1, \frac{m}{s-1})\}$
- Term t outputs 0 iff there is a variable (u, v) such that u and v under the bijection are assigned to tuples whose first component differs
- The number of bad graphs where this happens is at most $\binom{l}{2}\left(\frac{\binom{m}{s-1}}{m}\right)$.

Proof

An approximator either identically outputs 0 or outputs 1 on at least half the fraction of bad graphs

- Output of C is an $\wedge$, hence is approximators ouput is represented by a DNF of whose terms are of size at most l
- If it is identically 0 then we are done, otherwise there exists a term t whose size is at most l and $\bar{f} \geq t$ holds.
- Reperesent bad graphs using a bijection from $\{v_1, \ldots, v_m\}$ to $\{(1,1), \ldots, (1, \frac{m}{s-1}), \ldots, (s-1, 1), \ldots, (s-1, \frac{m}{s-1})\}$
- Term t outputs 0 iff there is a variable (u, v) such that u and v under the bijection are assigned to tuples whose first component differs
- The number of bad graphs where this happens is at most $\binom{l}{2}\left(\frac{\binom{m}{s-1}}{m}\right)$.

Proof

An approximator either identically outputs 0 or outputs 1 on at least half the fraction of bad graphs

- Output of C is an $\wedge$, hence is approximators ouput is represented by a DNF of whose terms are of size at most l
- If it is identically 0 then we are done, otherwise there exists a term t whose size is at most l and $\bar{f} \geq t$ holds.
- Reperesent bad graphs using a bijection from $\{v_1, \ldots, v_m\}$ to $\{(1,1), \ldots, (1, \frac{m}{s-1}), \ldots, (s-1, 1), \ldots, (s-1, \frac{m}{s-1})\}$
- Term t outputs 0 iff there is a variable (u, v) such that u and v under the bijection are assigned to tuples whose first component differs
- The number of bad graphs where this happens is at most $\binom{l}{2}\left(\frac{\binom{m}{s-1}}{m}\right)$.

Outline

- Proof is very similar to the above
- For approximated OR gates the bijection above suffices for counting
- For approximated AND gates which will make an error on good graphs we need a bijection from good graphs
- Bijections for good graphs from $\{v_1, \ldots, v_m\}$ to $\{1, \ldots, s, s+1, \ldots, m\}$ such that vertices which are mapped to the first s co-ordinates form a clique

Extending to Approximate Clique functions

Let $s_1 \leq s_2$ with $s_1^{\frac{1}{3}} s_2 \leq \frac{m}{200}$, and suppose C is a monotone circuit and that the fraction of good graphs in $I(m, s_2)$ such that C outputs 1 at least $h = h(s_2)$. Then at least one of the following is true

1. The number of gates in C is at least $(h/2)2^{s_1^{\frac{1}{3}}/4}$
2. The fraction of bad graphs in $O(m, s_1)$ such that C outputs 0 is at most $2/s_1^{1/3}$

Layering Notation

- Let $l_0 < l_1 < \cdots < l_\alpha$ be some monotone increasing sequence with $l_0 = s$ and $l_\alpha = m$
- A graph v is called good in the i-th layer if v consists of a clique of size l_{i-1} and no other edges
- A graph u is called bad in the i-th layer if there exists a partition of some l_i vertices into $s-1$ sets with equal size such that any two vertices choosen between different sets have and edge between them and no other edges exist.
- Any edge ending in a good graph of **first** layer will be in G_f.
- A edge starting in a bad graph of **any** layer will be in G_f.

Layering Notation

- Let $l_0 < l_1 < \cdots < l_\alpha$ be some monotone increasing sequence with $l_0 = s$ and $l_\alpha = m$
- A graph v is called good in the i-th layer if v consists of a clique of size l_{i-1} and no other edges
- A graph u is called bad in the i-th layer if there exists a partition of some l_i vertices into $s-1$ sets with equal size such that any two vertices choosen between different sets have and edge between them and no other edges exist.
- Any edge ending in a good graph of **first** layer will be in G_f.
- A edge starting in a bad graph of **any** layer will be in G_f.

Layering Notation

- Let $l_0 < l_1 < \cdots < l_\alpha$ be some monotone increasing sequence with $l_0 = s$ and $l_\alpha = m$
- A graph v is called good in the i-th layer if v consists of a clique of size l_{i-1} and no other edges
- A graph u is called bad in the i-th layer if there exists a partition of some l_i vertices into $s-1$ sets with equal size such that any two vertices choosen between different sets have and edge between them and no other edges exist.
- Any edge ending in a good graph of **first** layer will be in G_f.
- A edge starting in a bad graph of **any** layer will be in G_f.

Layering Notation

- Let $l_0 < l_1 < \cdots < l_\alpha$ be some monotone increasing sequence with $l_0 = s$ and $l_\alpha = m$
- A graph v is called good in the i-th layer if v consists of a clique of size l_{i-1} and no other edges
- A graph u is called bad in the i-th layer if there exists a partition of some l_i vertices into $s-1$ sets with equal size such that any two vertices choosen between different sets have and edge between them and no other edges exist.
- Any edge ending in a good graph of **first** layer will be in G_f.
- A edge starting in a bad graph of **any** layer will be in G_f.

Layering Notation

- Let $l_0 < l_1 < \cdots < l_\alpha$ be some monotone increasing sequence with $l_0 = s$ and $l_\alpha = m$
- A graph v is called good in the i-th layer if v consists of a clique of size l_{i-1} and no other edges
- A graph u is called bad in the i-th layer if there exists a partition of some l_i vertices into $s-1$ sets with equal size such that any two vertices choosen between different sets have and edge between them and no other edges exist.
- Any edge ending in a good graph of **first** layer will be in G_f.
- A edge starting in a bad graph of **any** layer will be in G_f.

- Without loss of genarality f_1 covers $\frac{1}{\alpha}$ fraction of edges of G_f
- Thanks to the extension of monotone clique function hardness to monotone clique approximators on a "large" number of bad graphs f_1 outputs 1, whereas f would have output 0
- Hence for a large number of good graphs obtained by adding edges to the wrongly classified bad graphs f_1 outputs 1 (as f_1 is monotone)
- For these graphs $f_1(u) = 1, f_1(u^+) = 1$ but $f(u) = 0, f(u^+) = 1$ hence $(u, u^+) \in G_f$ but $(u, u^+) \notin G_{f_1}$
- $\frac{1}{\alpha}$ fraction of these edges must be covered without loss of generality by f_2. Hence $f_2(u) = 0$ and $f_2(u^+) = 1$.
- Let v be a good graph in the second layer such that $u^+ \leq v$. Hence $f_2(v) = 1$ (montonicity).

- Without loss of genarality f_1 covers $\frac{1}{\alpha}$ fraction of edges of G_f
- Thanks to the extension of monotone clique function hardness to monotone clique approximators on a "large" number of bad graphs f_1 outputs 1, whereas f would have output 0
- Hence for a large number of good graphs obtained by adding edges to the wrongly classified bad graphs f_1 outputs 1 (as f_1 is monotone)
- For these graphs $f_1(u) = 1, f_1(u^+) = 1$ but $f(u) = 0, f(u^+) = 1$ hence $(u, u^+) \in G_f$ but $(u, u^+) \notin G_{f_1}$
- $\frac{1}{\alpha}$ fraction of these edges must be covered without loss of generality by f_2. Hence $f_2(u) = 0$ and $f_2(u^+) = 1$.
- Let v be a good graph in the second layer such that $u^+ \leq v$. Hence $f_2(v) = 1$ (montonicity).

- Without loss of genarality f_1 covers $\frac{1}{\alpha}$ fraction of edges of G_f
- Thanks to the extension of monotone clique function hardness to monotone clique approximators on a "large" number of bad graphs f_1 outputs 1, whereas f would have output 0
- Hence for a large number of good graphs obtained by adding edges to the wrongly classified bad graphs f_1 outputs 1 (as f_1 is monotone)
- For these graphs $f_1(u) = 1, f_1(u^+) = 1$ but $f(u) = 0, f(u^+) = 1$ hence $(u, u^+) \in G_f$ but $(u, u^+) \notin G_{f_1}$
- $\frac{1}{\alpha}$ fraction of these edges must be covered without loss of generality by f_2. Hence $f_2(u) = 0$ and $f_2(u^+) = 1$.
- Let v be a good graph in the second layer such that $u^+ \leq v$. Hence $f_2(v) = 1$ (montonicity).

- Without loss of genarality f_1 covers $\frac{1}{\alpha}$ fraction of edges of G_f
- Thanks to the extension of monotone clique function hardness to monotone clique approximators on a "large" number of bad graphs f_1 outputs 1, whereas f would have output 0
- Hence for a large number of good graphs obtained by adding edges to the wrongly classified bad graphs f_1 outputs 1 (as f_1 is monotone)
- For these graphs $f_1(u) = 1, f_1(u^+) = 1$ but $f(u) = 0, f(u^+) = 1$ hence $(u, u^+) \in G_f$ but $(u, u^+) \notin G_{f_1}$
- $\frac{1}{\alpha}$ fraction of these edges must be covered without loss of generality by f_2. Hence $f_2(u) = 0$ and $f_2(u^+) = 1$.
- Let v be a good graph in the second layer such that $u^+ \leq v$. Hence $f_2(v) = 1$ (montonicity).

- Without loss of genarality f_1 covers $\frac{1}{\alpha}$ fraction of edges of G_f
- Thanks to the extension of monotone clique function hardness to monotone clique approximators on a "large" number of bad graphs f_1 outputs 1, whereas f would have output 0
- Hence for a large number of good graphs obtained by adding edges to the wrongly classified bad graphs f_1 outputs 1 (as f_1 is monotone)
- For these graphs $f_1(u) = 1, f_1(u^+) = 1$ but $f(u) = 0, f(u^+) = 1$ hence $(u, u^+) \in G_f$ but $(u, u^+) \notin G_{f_1}$
- $\frac{1}{\alpha}$ fraction of these edges must be covered without loss of generality by f_2. Hence $f_2(u) = 0$ and $f_2(u^+) = 1$.
- Let v be a good graph in the second layer such that $u^+ \leq v$. Hence $f_2(v) = 1$ (montonicity).

- Apply the approximator lowerbound again, as we have showed that f_2 outputs correctly for a large fraction of good graphs in layer 2. Hence we get that there are bad graphs in layer 2 where f_2 errs.
- If f_2 errs on a bad graph in layer 2 then so must f_1.
- Continuing the argument shows that every function $f_1, \ldots, f_\alpha$ errs on at least one bad graph.

- Apply the approximator lowerbound again, as we have showed that f_2 outputs correctly for a large fraction of good graphs in layer 2. Hence we get that there are bad graphs in layer 2 where f_2 errs.
- If f_2 errs on a bad graph in layer 2 then so must f_1.
- Continuing the argument shows that every function $f_1, \ldots, f_\alpha$ errs on at least one bad graph.

- Apply the approximator lowerbound again, as we have showed that f_2 outputs correctly for a large fraction of good graphs in layer 2. Hence we get that there are bad graphs in layer 2 where f_2 errs.
- If f_2 errs on a bad graph in layer 2 then so must f_1.
- Continuing the argument shows that every function $f_1, \ldots, f_\alpha$ errs on at least one bad graph.

Thank You

Q

Questions ?