5. The bottleneck counting method

References

- Armin Haken, Counting bottlenecks to show monotone $P \neq NP$, 36th FOCS (1995), 36-40.
- Armin Haken, Stephen A. Cook, An exponential lower bound for the size of monotone real circuits, JCSS 58 (1999), 326 – 335.
- Christer Berg, Staffan Ulfberg, Symmetric approximation arguments for monotone lower bounds without sunflowers, Comput. complex. 8 (1999), 1-20.
- Stasys Jukna, Combinatorics of monotone computations, Combinatorica 19 (1999), 65-85.
- Danny Harnik, Ran Raz, Higher lower bounds on monotone size, 32nd STOC (2000), 191-201.
- Kazuyuki Amano, Akira Maruoka, The potential of the approximation method, SIAM J. Comput. 33 (2004), 433-447.
- Janos Simon, Shi-Chun Tsai, On the bottleneck counting argument, TCS 237 (2000), 429 – 437.

In 1995, Armin Haken has introduced the so-called "bottleneck counting method" for proving lower bounds for the monotone network complexity of a Boolean function which resembles the clique function. Instead of approximating the behaviour of the network directly he has defined a function μ which maps inputs to the gates of the network. He has shown that the total number of inputs mapped to the gates by μ has to be large and also so that only few inputs are being mapped to each gate. Hence, the network must contain many gates. Shortly after that, Berg and Ulfberg and also Amano and Maruoka have shown that Haken's approach is in fact an approximation argument in disguise. They have translated Haken's proof method into an approximation proof method which uses CNF - and DNF - approximators. We shall present Berg and Ulfberg's description of the bottleneck counting method. Before doing this we shall investigate the structures of the functions which are computed at the nodes of a monotone Boolean network β which computes a function $f \in M_n$.

Let u be any node in β. The function $res_\beta(u)$ can be written as a DNF formula; i.e.,

$$res_\beta(u) = \bigvee_{j=1}^{t} m_j$$

where each m_j is a monomial.

Starting at the input nodes of the network, we can compute these DNF formulas in the obvious way by applying the properties of the Boolean operations.

We call this representation of $res_\beta(u)$ the DNF representation of $res_\beta(u)$.

Next we characterize the DNF representation of the output node u_0 of β. Let $p_1, p_2, \ldots, p_r$ be the prime implicants of the function f. For the DNF formula

$$res_\beta(u_0) = \bigvee_{j=1}^{t_0} m_j$$

each monomial m_j is an implicant of the function f. Otherwise, it would exist an input $(a_1, a_2, \ldots, a_n) \in \{0,1\}^n$ such that

$$f(a_1, a_2, \ldots, a_n) = 0 \text{ but } res_\beta(u_0)(a_1, a_2, \ldots, a_n) = 1.$$

This means that each monomial of the DNF representation of $res_\beta(u_0)$ contains at least one prime implicant of the function f.

Moreover, for each prime implicant p_i, $1 \leq i \leq r$ there is a $j \in \{1, 2, \ldots, t_0\}$ with $m_j = p_i$. Otherwise, there is an input $(a_1, a_2, \ldots, a_n) \in \{0,1\}^n$ such that

$f(a_1, a_2, \ldots, a_n) = 1$ but $res_\beta(u_0)(a_1, a_2, \ldots, a_n) = 0$.

Hence, we can restrict our considerations to the prime implicants contained in the DNF representation of $res_\beta(u_0)$. (P-DNF representation)

Furthermore, for each node u in β, we can restrict our consideration to the prime implicants of $res_\beta(u)$. Note that all these prime implicants are contained in the DNF representation of $res_\beta(u)$ defined above.

The disjunction of some variables is called a clause. The empty clause is the constant function 0. If we delete some variables from a clause c then we obtain a subclause of c

A clause c is called f-clause if for all $a \in \{0,1\}^n$

$$c(a) = 0 \Rightarrow f(a) = 0.$$

A prime f-clause c is a f-clause where no proper subclause of c is a f-clause.

The conjunction of clauses is a formula in conjunctive normal form (CNF).

Goal:
Derivation from the P-DNF representation of f an equivalent expression which we call

P-CNF representation of f:

The following algorithm constructs the P-CNF representation of f.

Algorithm P-DNF $\rightsquigarrow$ P-CNF

Input: $PIM(f) = \{p_1, p_2, \ldots, p_r\}$

Output: P-CNF representation of f

Method:

(1) Compute all prime f-clauses $c_1, c_2, \ldots, c_t$.

(2) Output := $\bigwedge_{j=1}^{t} c_j$.

Exercise:

a) Develop an algorithm for the computation of all prime f-clauses

b) Is it possible that a prime f-clause contains more than one variable of a prime implicant? Prove your answer.

Exercise:

a) Develop an algorithm which given the P-CNF representation of f computes the P-DNF representation of f.

b) Let f be a monotone Boolean function. Show that the P-DNF representation and also the P-CNF representation are unique.

Now, we are prepared to describe the bottleneck counting method.

Let β be a monotone network which computes a function $f \in M_n$. The idea is to approximate the function $res_\beta(u)$ by two functions

$$D_u^r = \bigvee_{i=1}^{t_1} m_i \quad \text{and} \quad C_u^s = \bigwedge_{j=1}^{t_2} c_j$$

where each m_i is a monomial of size $< r$ and each c_j is a clause of size $< s$.

The size of a monomial m_i can be the number of distinct variables in m_i or another measure. The size of a clause can be the number of distinct variables or another measure.

We do not restrict the number of monomials in D_u^r and also not the number of clauses in C_u^s. Let u_0 be the output gate of β. The approximators associated with the gates have the following properties:

1) The approximators $C_{u_0}^s$ and $D_{u_0}^r$ fail to compute correctly the value $f(x)$ for many inputs x.

2) For each gate u in β, the number of

inputs for which the approximators introduce an error is small.

For each $a \in \{0,1\}^n$ for which $C^s_{u_0}(a) \neq f(a)$ or $D^r_{u_0}(a) \neq f(a)$, the error has to be introduced by at least one of the gates in β. Hence, the number of gates in β has to be large.

We start in the output node u_0 and run backwards through β applying the following rule with respect to a given input $a \in \{0,1\}^n$:

- The edge (v,w) is traversed iff
  - w is visited during the backward search and $res_\beta(v)(a) = f(a)$.

β_a is that subnetwork of β which contains exactly the visited nodes and the traversed edges. Note that in some sence, the correct value $f(a)$ is computed by the subnetwork β_a.

For an input $a \in \{0,1\}^n$ and a node $u \in \beta_a$, we say that the approximator C^s_u (D^r_u) fails for a if

$$C^s_u(a) \neq res_\beta(u)(a)$$

$$\left(D^r_u(a) \neq res_\beta(u)(a) \right).$$

The approximator C^s_u or D^r_u introduces an error for the input a, $u \in \beta_a$ if it fails for a but none of the approximators for the input

of u in β_a fail for the input a.

Now, we shall define the approximators of the nodes in β.

If the node is an input node $x_{i,j}$ then the approximators are defined by

$$C^s_{x_{i,j}} := x_{i,j} \quad \text{and} \quad D^r_{x_{i,j}} := x_{i,j}.$$

For the definition of the approximators of the other nodes in β, we consider the nodes in β in a topological order such that always the approximators with respect to both direct predecessors of the considered node are defined. Let u be the considered node and let v and w be the direct predecessors of u. Let

$$D^r_v = \bigvee_{i=1}^{t_1'} m_i' \quad \text{and} \quad C^s_v = \bigwedge_{j=1}^{t_2'} c_j'$$

$$D^r_w = \bigvee_{i=1}^{t_1''} m_i'' \quad \text{and} \quad C^s_w = \bigwedge_{j=1}^{t_2''} c_j''.$$

Note that all monomials and all clauses contained in the approximators have size $< r$ and $< s$, respectively.

According to the type of the node u, we distinguish two cases.

Case 1: u is an $\wedge$-gate.

We obtain the approximators

$$D_u^r = \bigvee_{i=1}^{t'} \bar{m}_i \quad \text{and} \quad C_u^s = \bigwedge_{j=1}^{t''} \bar{c}_j$$

in the following way:

$$C_u^s := C_v^s \wedge C_w^s.$$

Since each clause in C_v^s and in C_w^s has size less than s, all clauses in C_u^s have still size less than s.

If none of C_v^s and C_w^s fail for an input a then C_u^s does not fail for the input a as well.

$\Rightarrow$

The approximator C_u^s does not introduce an error for any input a.

The approximator D_u^r is constructed from the approximator C_u^s by transforming the CNF expression first into an equivalent DNF expression $\tilde{D}_u^{ttt}$. This does not introduce an error for any input $a \in \{0,1\}^n$. Then D_u^r is obtained from $\tilde{D}_u^r$ by removing all monomials of size $\geq r$.

page 160 a)

First, we shall describe the construction of $\tilde{D}_u^r$. For doing this, we first define the

Case 2: u is an $\vee$-gate.

We obtain the approximators

$$D_u^r = \bigvee_{i=1}^{t'} \bar{m}_i \quad \text{and} \quad C_u^s = \bigwedge_{j=1}^{t''} \bar{c}_j$$

in the following way.

$$D_u^r := D_v^r \vee D_w^r .$$

Since each monomial in D_v^r and in D_w^r has size less than r, all monomials in D_u^r have still size less than r.

If none of D_v^r and D_w^r fail for an input a then D_u^r does not fail for the input a as well.

$\Rightarrow$

The approximator D_u^r does not introduce an error for any input a.

The approximator C_u^s is constructed from the approximator D_u^r by transforming the DNF expression first into an equivalent CNF expression $\tilde{C}_u^s$. This does not introduce an error for any input $a \in \{0,1\}^n$. Then C_u^s is obtained from $\tilde{C}_u^s$ by removing all clauses of size $\geq s$.

$\Rightarrow$

By construction, no error with respect to an input in $cl_{n,2}^{-1}(0)$ can be introduced at

an $\wedge$-gate and no error with respect to an input in $Cl^{-1}_{4,2}(1)$ can be introduced at an $\vee$-gate.

sizes of monomials and clauses used in the lower bound proof for the clique function $cl_{n,k}$.

We say that a monomial m_e (clause c_e) touches a node $v \in V$ iff there is at least one variable in m_e (in c_e) which corresponds to an edge which has an end node v. The size of the monomial m_e is the number of different nodes in V which are touched by m_e.

The variables in a clause c_e correspond to an edge set $E(c_e)$. These edges define a graph $G(c_e) := (V, E(c_e))$. The size of the clause c_e is n minus the number of connected components in $G(c_e)$.

For the approximators D_n^r and C_n^s we use the values

$$r := \lfloor \sqrt{k} \rfloor \quad \text{and} \quad s := \lfloor \frac{n}{8k} \rfloor .$$

Since $r = \lfloor \sqrt{k} \rfloor$, less than $\lfloor \sqrt{k} \rfloor$ different nodes in V can be touched by a monomial. Hence, the number of variables in such a monomial is bounded by

$$\frac{r^2}{2} \leq \frac{k}{2} .$$

$s = \lfloor \frac{n}{8k} \rfloor$ implies that the graphs corresponding to the clauses have more than $n - \lfloor \frac{n}{8k} \rfloor$

connected components. The number of different end nodes of the edges in such a graph is maximized if the number of connected components with exactly two nodes is maximized.

$\Rightarrow$

The number of different end nodes of the edges in such a graph is less than

$$2s \leq \frac{n}{4k}.$$

Therefore, a clause c_ℓ touches less than $\frac{n}{4k}$ nodes of the graph.

The following lemma shows that the approximators for the output node u_0 of β fail for a large amount of the inputs

Lemma 5.1

$C^s_{u_0}$ fails for all inputs in $cl^{-1}_{n,k}(0)$ or $C^s_{u_0}$ fails for at least half of the inputs which correspond exactly to the prime implicants of the function $cl_{n,k}$.

Proof:

If $C^s_{u_0}$ fails for all inputs in $cl^{-1}_{n,k}(0)$ then nothing is to prove. Otherwise, there is $b \in cl^{-1}_{n,k}(0)$ such that $C^s_{u_0}(b) = 0$.

Hence, $C^S_{u_0}$ is not the constant function 1. If $C^S_{u_0}$ is the constant function 0 then $C^S_{u_0}$ fails for all inputs in $cl^{-1}_{u,k}(1)$ such that nothing is to prove.

Otherwise, $C^S_{u_0}$ contains at least one non-empty clause c_e.

Let a be an input which corresponds exactly to a prime implicant p of $cl_{u,k}$. This means that $a_{i,j} = 1$ iff (i,j) is an edge of the clique $cl(p)$ corresponding to p. ↗ PI - input

Now we shall derive an upper bound for the number of such inputs a for which $C^S_{u_0}$ does not fail. Note that

$C^S_{u_0}(a) = 1 \Rightarrow$ c_e contains a variable $x_{i,j}$ such that (i,j) is an edge of $cl(p)$.

This implies that i is a node of the clique $cl(p)$.

Each node is contained in a fraction $\frac{k}{n}$ of the possible k-cliques of a graph with n nodes.

As observed above, c_e touches less than $\frac{n}{4k}$ of the nodes in V.

$\Rightarrow$

The fraction of PI-inputs for which $C^S_{u_0}$ does not fail is

$$< \frac{n}{4k} \cdot \frac{2k}{n} = \frac{1}{4}.$$

This proves the lemma.

□

Note that all errors with respect to an input in $cl^{-1}_{u,k}(1)$ are introduced at an $\wedge$-gate of β. The following lemma bounds the number of prime implicants p_e such that an error with respect to the corresponding PI-input can be introduced at an $\wedge$-gate u.

Lemma 5.2

Let u be an $\wedge$-gate in β. Then the following holds.

a) The number of prime implicants p_e such that the approximator D^r_u introduces an error for the corresponding PI-input is bounded by $\binom{n-r}{k-r}\left(\frac{n}{4k}\right)^r$.

b) D^r_u introduces no error for any input in $cl^{-1}_{u,k}(0)$.

Proof:
Our goal is to get an upper bound for the prime implicants p_e such that the approximator D^r_u introduces an error for the corresponding PI-input

which is as small as possible. Hence, we shall organize the construction of $\tilde{D}_u^r$ from C_u^s very carefully.

Goal:
Construction of a set of monomials such that at least one of the monomials is satisfied by an (PI-) input or iff all clauses in C_u^s are satisfied by or.

The disjunction of all these monomials gives us $\tilde{D}_u^r$.

Assume that the clauses in C_u^r are given in any fixed order $\bar{C}_1, \bar{C}_2, \ldots, \bar{C}_{t^u}$. We organize the construction by building a tree T.

- Each edge in T is labeled by an input variable of β or has no label. To each node v in T corresponds that monomial which we obtain by the conjunction of the variables on the unique path from the root of T to v.
- At the root of T, we create for each of the variables in $\bar{C}_1$ one labeled edge to a new son. We say that the clause $\bar{C}_1$ is expanded under the root.
- Suppose that w is a leaf that was created while expanding $\bar{C}_i$ and that $m(w)$ is the monomial corresponding to w.

Then a PI-input a satisfies $\overline{c}_1, \overline{c}_2, \ldots, \overline{c}_i$ iff a satisfies the monomial $m(w)$.

Now we wish to treat the clause $\overline{c}_{i+1}$ with respect to the leaf w. We distinguish two cases.

Case 1: $\overline{c}_{i+1}$ contains a variable $x_{u,v}$ for which both end nodes are already touched by $m(w)$.

Then a PI-input that satisfies $m(w)$ also satisfies $\overline{c}_{i+1}$

$\rightsquigarrow$

We only create one son w' for w and label the edge (w,w') with the variable $x_{u,v}$.

Case 2: $\neg$ Case 1

Then the clause $\overline{c}_{i+1}$ separates into two set Var_0 and Var_1 of variables such that

$$Var_0 = \{ x_{u,v} \mid \text{both end nodes } u,v \text{ are not touched by } m(w) \}$$

$$Var_1 = \{ x_{u,v} \mid \text{exactly one of } u \text{ and } v \text{ is touched by } m(w) \}$$

First we consides the variables in Var_1. Let

$N(Var_1) := \{u \in V \mid u$ is not touched by $m(w)$ and $\exists$ variable $x_{u,v} \in Var_1\}$.

For each $u \in N(Var_1)$ choose any variable $x_{u,v} \in Var_1$ and create a son w_u of the node w. Label the edge (w, w_u) with the variable $x_{u,v}$.

Note that two monomials touching the same nodes accept the same PI-inputs. Hence, it suffice to choose for each node $u \in N(Var_1)$ exactly one variable $x_{u,v} \in Var_1$ and to take away the other such variables from the consideration.

The total number of sons created in this way is less

$$2s \leq \frac{n}{4k}$$

since $\bar{c}_{i+1}$ touches at most $2s$ nodes.

Now we consider the variables in Var_0.

As long as there is an edge in Var_0 such that none of its two end nodes is chosen, we choose an end node of such an edge and create a son for w with respect to this node. The corresponding edges are not labeled

$\Rightarrow$

$$< 2s \leq \frac{n}{4k}$$

sons are created.

Then, for each son of w we create

$$< 2s = \frac{n}{4k}$$

sons for the other end node of the edges. These edges are labeled with the corresponding variable.

Construction of T $\Rightarrow$

a) No node in T has more than

$$2s \leq \frac{n}{4k}$$

sons.

b) When descending to a son from a node with more than one son, the number of nodes touched by the corresponding monomial increases by one.

$\Rightarrow$

There are at most

(*) $$\left(\frac{n}{4k}\right)^r$$

monomials touching exact r nodes in V.

r nodes are contained in exact $\binom{n-r}{k-r}$ k-cliques.

$\Rightarrow$

The deletion of such a monomials introduces

an error for at most

(**) $\binom{n-r}{k-r}$

PI - inputs.

(*) and (**) $\Rightarrow$

After the deletion of all monomials touching exact r variables, an error with respect

$$\leq \binom{n-r}{k-r}\left(\frac{n}{4k}\right)^r$$

PI - inputs is introduced

b) ~~exercise~~.

Now we wish to bound the number of inputs in $cl_{n,k}^{-1}(0)$ for which an error is introduced at an v-gate of β. For doing this we shall only consider inputs corresponding to complete $(k-1)$-partite graphs; i.e., inputs corresponding to graphs $G(h)$ where

$$h: V \to \{1, 2, \ldots, k-1\}$$

is a colouring of the nodes in V by $k-1$ colours. We call such an input $G(h)$-input.

Lemma 5.3

Let u be a v-gate in β. Then the following holds:

a) The number of $G(n)$-inputs for which the approximator C_u^s introduces an ~~error~~ is bounded by

$$\left(\frac{k}{2}\right)^s (k-1)^{n-s}.$$

b) C_u^s introduces no error for any input in $cl_{u,k}^{-1}(1)$.

Proof:

Analogous to the proof of Lemma 5.2 we shall organize the construction of $\tilde{C}_u^s$ from D_u^Γ very carefully

Goal:

Construction of a set of clauses such that at least one of the clauses is falsified by an $G(n)$-input a iff all monomials in D_u^Γ are falsified by a.

Assume that the monomials in D_u^Γ are given in any fixed order $\bar{m}_1, \bar{m}_2, \dots, \bar{m}_{t'}$. We organize the construction by building a tree T.

- Each edge in T is labeled by an input variable of β. To each node v in T corresponds that clause which we obtain by the disjunction of the variables on the unique path from the root of T to v.

- Suppose that w is a leaf that was created while expanding $\overline{m}_i$ and that $c(w)$ is the clause corresponding to w.

Then a $G(h)$-input a falsifies all monomials $\overline{m}_1, \overline{m}_2, \ldots, \overline{m}_i$ iff a falsifies the clause $c(w)$.

Now we wish to treat the monomial $\overline{m}_{i+1}$ with respect to the leaf w. We distinguish two cases.

Case 1: $\overline{m}_{i+1}$ contains a variable $x_{u,v}$ for which both end nodes are in the same connected component of the graph

$$G(c(w)) = (V, E(c(w))).$$

Then, a $G(h)$-input which falsifies $c(w)$ also falsifies $\overline{m}_{i+1}$ since all nodes in the same connected component of $G(c(w))$ have to be in the same colour class.

$\Rightarrow$

It suffices to create one son w' for w and to label the edge (w, w') with the variable $x_{u,v}$.

Case 2: $\neg$ Case 1.

For each variable in $\overline{m}_{i+1}$ we create a son of w and label the corresponding edge with that variable.

Since $\overline{m}_{i+1}$ contains at most $\frac{r^2}{2} \leq \frac{n}{2}$ variables the number of created sons is bounded by

$$\frac{n}{2}.$$

Since for each such a variable $x_{u,v}$ both end nodes are contained in different connected components, the number of connected components decreases by one for each son.

$\Rightarrow$

We make s such expansions before we reach a node for which the graph of the corresponding clause contains $n-s$ connected components.

$\Rightarrow$

In total we obtain at most

$$\left(\frac{n}{2}\right)^d$$

such clauses.

It remains to bound the number of $G(4)$-inputs which are falsified by a clause c whose corresponding graph $G(c)$ contains $n-s$ connected components.

A $G(4)$-input is falsified by such a clause c iff all nodes within a connected

component are coloured with the same colour.

There are at most

$$(k-1)^{n-d}$$

such $G(k)$-inputs.

Remark:

$(k-1)^{n-d}$ is the number of different colourings of $n-d$ connected components. In reality different colouring corresponds to the same $G(k)$-input. This would only decrease the bound above.

□

Theorem 5.1

Let $k \leq n^{2/3}$. Then

$$C_{2m}(cl_{n,k}) = 2^{\Omega(\sqrt{k})}.$$

Proof:

We distinguish two cases.

Case 1: $C^S_{n_0}$ fails for at least half of the PI-inputs.

Lemma 5.2 $\Rightarrow$

$$C_{\Omega_m}(cl_{n,k}) \geq \frac{\binom{n}{k}}{2\binom{n-r}{k-r}\left(\frac{n}{4k}\right)^r}$$

$$\in 2^{\Omega(\sqrt{k})}$$

exercise

Case 2: $C^S_{n_0}$ fails for all $G(k)$-inputs.

Lemma 5.3 $\Rightarrow$

$$C_{\Omega_m}(cl_{n,k}) \geq \frac{(k-1)^n}{(k-1)^{n-s}\left(\frac{k}{2}\right)^s}$$

$$\in 2^{\Omega\left(\frac{n}{k}\right)}$$

exercise.

Since

$$2^{\Omega(\sqrt{k})} \subseteq 2^{\Omega\left(\frac{n}{k}\right)} \quad \text{for } k \leq n^{2/3}$$

the assertion follows.

□