Non-Uniform Complexity

A. Antonopoulos

Theoretical Computer Science I

Computation and Reasoning Laboratory
National Technical University of Athens

January 2012

Boolean Circuits

- A Boolean Circuit is a natural model of *nonuniform* computation, a generalization of hardware computational methods.
- A non-uniform computational model allows us to use a different "algorithm" to be used for every input size, in contrast to the standard (or *uniform*) Turing Machine model, where the same T.M. is used on (infinitely many) input sizes.
- Each circuit can be used for a fixed input size, which limits or model.

Definition (Boolean circuits)

For every $n \in \mathbb{N}$ an n-input, single output Boolean Circuit C is a directed acyclic graph with n sources and *one* sink.

- All nonsource vertices are called *gates* and are labeled with one of $\wedge$ (and), $\vee$ (or) or $\neg$ (not).
- The vertices labeled with $\wedge$ and $\vee$ have *fan-in* (i.e. number or incoming edges) 2.
- The vertices labeled with $\neg$ have *fan-in* 1.
- The *size* of C, denoted by $|C|$, is the number of vertices in it.
- For every vertex v of C, we assign a value as follows: for some input $x \in \{0,1\}^n$, if v is the i-th input vertex then $val(v) = x_i$, and otherwise $val(v)$ is defined recursively by applying v's logical operation on the values of the vertices connected to v.
- The *output* $C(x)$ is the value of the output vertex.
- The *depth* of C is the length of the longest directed path from an input node to the output node.

- To overcome the fixed input length size, we need to allow families (or sequences) of circuits to be used:

Definition

Let $T : \mathbb{N} \to \mathbb{N}$ be a function. A $T(n)$*-size circuit family* is a sequence $\{C_n\}_{n\in\mathbb{N}}$ of Boolean circuits, where C_n has n inputs and a single output, and its size $|C_n| \leq T(n)$ for every n.

- These infinite families of circuits are defined arbitrarily: There is **no** pre-defined connection between the circuits, and also we haven't any "guarantee" that we can construct them efficiently.
- Like each new computational model, we can define a complexity class on it by imposing some restriction on a *complexity measure*:

Definition

We say that a language L is in $\mathbf{SIZE}(T(n))$ if there is a $T(n)$*-size circuit family* $\{C_n\}_{n\in\mathbb{N}}$, such that $\forall x \in \{0,1\}^n$:

$$x \in L \Leftrightarrow C_n(x) = 1$$

Definition

$\mathbf{P}_{/\mathbf{poly}}$ is the class of languages that are decidable by polynomial size circuits families. That is,

$$\mathbf{P}_{/\mathbf{poly}} = \bigcup_{c\in\mathbb{N}} \mathbf{SIZE}(n^c)$$

Theorem (Nonuniform Hierarchy Theorem)

For every functions $T, T' : \mathbb{N} \rightarrow \mathbb{N}$ *with* $\frac{2^n}{n} > T'(n) > 10T(n) > n$,

$$\mathbf{SIZE}(T(n)) \subsetneq \mathbf{SIZE}(T'(n))$$

Turing Machines that take advice

Definition

Let $T, \alpha : \mathbb{N} \to \mathbb{N}$. The class of languages decidable by $T(n)$-time Turing Machines with $a(n)$ bits of advice, denoted

$$\mathbf{DTIME}\left(T(n)/a(n)\right)$$

containts every language L such that there exists a secuence $\{a_n\}_{n\in\mathbb{N}}$ of strings, with $a_n \in \{0,1\}^{a(n)}$ and a Turing Machine M satisfying:

$$x \in L \Leftrightarrow M(x, a_n) = 1$$

for every $x \in \{0,1\}^n$, where on input (x, a_n) the machine M runs for at most $\mathcal{O}(T(n))$ steps.

Turing Machines that take advice

Theorem (Alternative Definition of $P_{/poly}$)

$$\mathbf{P}_{/\mathbf{poly}} = \bigcup_{c,d \in \mathbb{N}} \mathbf{DTIME}(n^c / n^d)$$

Turing Machines that take advice

Theorem (Alternative Definition of $P_{/poly}$)

$$\mathbf{P}_{/\mathbf{poly}} = \bigcup_{c,d \in \mathbb{N}} \mathbf{DTIME}(n^c / n^d)$$

Proof: $(\subseteq)$ Let $L \in \mathbf{P}_{/\mathbf{poly}}$. Then, $\exists \{C_n\}_{n \in \mathbb{N}} : C_{|x|} = L(x)$.
We can use C_n 's encoding as an advice string for each n.

Turing Machines that take advice

Theorem (Alternative Definition of $P_{/poly}$)

$$\mathbf{P}_{/\mathbf{poly}} = \bigcup_{c,d \in \mathbb{N}} \mathbf{DTIME}(n^c/n^d)$$

Proof: ($\subseteq$) Let $L \in \mathbf{P}_{/\mathbf{poly}}$. Then, $\exists \{C_n\}_{n \in \mathbb{N}} : C_{|x|} = L(x)$.
We can use C_n 's encoding as an advice string for each n.
($\supseteq$) Let $L \in \mathbf{DTIME}(n^c)/n^d$. Then, since CVP is $\mathbf{P}$-complete, we construct for every n a circuit D_n such that, for $x \in \{0,1\}^n, a_n \in \{0,1\}^{a(n)}$:

$$D_n(x, a_n) = M(x, a_n)$$

Then, let $C_n(x) = D_n(x, a_n)$ (We hard-wire the advice string!)
Since $a(n) = n^d$, the circuits have polynomial size. $\square$.

Theorem

$$\mathbf{P} \subsetneq \mathbf{P}_{/\mathbf{poly}}$$

- For "$\subseteq$", recall that $\mathtt{CVP}$ is **P**-complete.
- But why proper inclusion?
- Consider the following language:

$$\mathtt{U} = \{1^n | n\text{'s binary expression encodes a pair } < M, x > \text{ s.t. } M(x)\downarrow\}$$

- It is easy to see that $\mathtt{U} \in \mathbf{P}_{/\mathbf{poly}}$, but....

Theorem (Karp-Lipton Theorem)

If $\mathbf{NP} \subseteq \mathbf{P}_{/\mathbf{poly}}$, *then* $\mathbf{PH} = \Sigma_2^p$.

Theorem (Meyer's Theorem)

If $\mathbf{EXP} \subseteq \mathbf{P}_{/\mathbf{poly}}$, *then* $\mathbf{EXP} = \Sigma_2^p$.

Uniform Families of Circuits

- We saw that $\mathbf{P}_{/\mathbf{poly}}$ contains an undecidable language.
- The root of this problem lies in the "weak" definition of such families, since it suffices that $\exists$ a circuit family for L.
- We haven't a way (or an algorithm) to construct such a family.
- So, may be useful to restric or attention to families we can construct efficiently:

Theorem (P-Uniform Families)

A circuit family $\{C_n\}_{n\in\mathbb{N}}$ is $\mathbf{P}$-uniform if there is a polynomial-time T.M. that on input 1^n outputs the description of the circuit C_n.

- But...

Theorem

A language L is computable by a $\mathbf{P}$-uniform circuit family iff $L \in \mathbf{P}$.

Theorem

$$\mathbf{BPP} \subset \mathbf{P}_{/\mathbf{poly}}$$

Proof: Recall that if $L \in \mathbf{BPP}$, then $\exists$ PTM M such that:

$$\mathbf{Pr}_{r \in \{0,1\}^{poly(n)}} [M(x,r) \neq L(x)] < 2^{-n}$$

Then, taking the union bound:

$$\mathbf{Pr}[\exists x \in \{0,1\}^n : M(x,r) \neq L(x)] = \mathbf{Pr}\left[\bigcup_{x \in \{0,1\}^n} M(x,r) \neq L(x)\right] \leq$$

$$\leq \sum_{x \in \{0,1\}^n} \mathbf{Pr}[M(x,r) \neq L(x)] < 2^{-n} + \cdots + 2^{-n} = 1$$

So, $\exists r_n \in \{0,1\}^{poly(n)}$, s.t. $\forall x \{0,1\}^n$: $M(x,r) = L(x)$.
Using $\{r_n\}_{n \in \mathbb{N}}$ as advice string, we have the non-uniform machine. $\square$

Definition (Circuit Complexity or Worst-Case Hardness)

For a finite Boolean Function $f : \{0,1\}^n \rightarrow \{0,1\}$, we define the (circuit) *complexity* of f as the size of the smallest Boolean Circuit computing f (that is, $C(x) = f(x), \forall x \in \{0,1\}^n$).

Definition (Average-Case Hardness)

The minimum S such that there is a circuit C of size S such that:

$$\mathbf{Pr}\left[C(x) = f(x)\right] \geq \frac{1}{2} + \frac{1}{S}$$

is called the (average-case) hardness of f.

Hierarchies for Semantic Classes with advice

- We have argued why we can't obtain Hierarchies for semantic measures using classical diagonalization techniques. But using small advice we can have the following results:

Theorem ([Bar02], [GST04])

For $a, b \in \mathbb{R}$, with $1 \leq a < b$:

$$\mathbf{BPTIME}(n^a)/1 \subsetneq \mathbf{BPTIME}(n^b)/1$$

Theorem ([FST05])

For any $1 \leq a \in \mathbb{R}$ there is a real $b > a$ such that:

$$\mathbf{RTIME}(n^b)/1 \subsetneq \mathbf{RTIME}(n^a)/\log(n)^{1/2a}$$

Circuit Lower Bounds

- The significance of proving lower bounds for this computational model is related to the famous "**P** vs **NP**" problem, since:

$$\mathbf{NP} \setminus \mathbf{P}_{/\mathbf{poly}} \neq \emptyset \Rightarrow \mathbf{P} \neq \mathbf{NP}$$

- But...after decades of efforts, The best lower bound for an **NP** language is $5n - o(n)$, proved very recently (2005).
- There are better lower bounds for some special cases, i.e. some restricted classes of circuits, such as: bounded depth circuits, monotone circuits, and bounded depth circuits with "counting" gates.

Definition

Let $PAR : \{0,1\}^n \to \{0,1\}$ be the *parity* function, which outputs the modulo 2 sum of an n-bit input. That is:

$$PAR(x_1, ..., x_n) \equiv \sum_{i=1}^{n} x_i \pmod 2$$

Theorem

For all constant d, PAR has no polynomial-size circuit of depth d.

- The above result (improved by Håstad and Yao) gives a relatively tight lower bound of $\exp\left(\Omega(n^{1/(d-1)})\right)$, on the size of n-input PAR circuits of depth d.

Definition

For $x, y \in \{0,1\}^n$, we denote $x \preceq y$ if every bit that is 1 in x is also 1 in y. A function $f : \{0,1\}^n \to \{0,1\}$ is *monotone* if $f(x) \leq f(y)$ for every $x \preceq y$.

Definition

A Boolean Circuit is *monotone* if it contains only AND and OR gates, and no NOT gates. Such a circuit can only compute monotone functions.

Theorem (Monotone Circuit Lower Bound for CLIQUE)

Denote by $CLIQUE_{k,n} : \{0,1\}^{\binom{n}{2}} \to \{0,1\}$ the function that on input an adjacency matrix of an n-vertex graph G outputs 1 iff G contains an k-clique. There exists some constant $\epsilon > 0$ such that for every $k \leq n^{1/4}$, there is no monotone circuit of size less than $2^{\epsilon\sqrt{k}}$ that computes $CLIQUE_{k,n}$.

- So, we proved a significant lower bound ($2^{\Omega(n^{1/8})}$)
- The significance of the above theorem lies on the fact that there was some alleged connection between monotone and non-monotone circuit complexity (e.g. that they would be polynomially related). Unfortunately, Éva Tardos proved in 1988 that the gap between the two complexities is exponential.
- Where is the problem finally?
  Today, we know *that a result for a lower bound using such techniques would imply the inversion of strong one-way functions:*

*Natural Proofs [Razborov, Rudich 1994]

Definition

Let $\mathcal{P}$ be the predicate:

> *"A Boolean function $f : \{0,1\}^n \to \{0,1\}$ doesn't have n^c-sized circuits for some $c \geq 1$."*

$\mathcal{P}(f) = 0, \forall f \in \mathbf{SIZE}(n^c)$ for a $c \geq 1$. We call this n^c-*usefulness*.

A predicate $\mathcal{P}$ is natural if:

- There is an algorithm $M \in \mathbf{E}$ such that for a function $g : \{0,1\}^n \to \{0,1\}$: $M(g) = \mathcal{P}(g)$.
- For a random function g: $\mathbf{Pr}[\mathcal{P}(g) = 1] \geq \frac{1}{n}$

Theorem

If strong one-way functions exist, then there exists a constant $c \in \mathbb{N}$ such that there is no n^c-useful natural predicate $\mathcal{P}$.

References

- Sanjeev Arora and Boaz Barak, *Computational Complexity: A Modern Approach*, Cambridge University Press, 2009
- Ding-Zhu Du and Ker-I Ko, *Theory of Computational Complexity*, Wiley-Interscience, 2000
- Oded Goldreich, *Computational Complexity: A Conceptual Perspective*, Cambridge University Press, 1st edition, 2008
- *Ingo Wegener, *The Complexity of Boolean Functions*, John Wiley and Sons Ltd, and B. G. Teubner, Stuttgart, 1987
- Lance Fortnow, Rahul Santhanam, *Time Hierarchies: A Survey*, Electronic Colloquium on Computational Complexity, 2007 `http://eccc.hpi-web.de/report/2007/004/`

Thank You!