Average Case Complexity

Mohith Raju N and Umang Bhat

The case for Average Case Complexity

Real Life Problems

- In our course, we've looked at the time taken on the "worst" possible inputs, i.e. worst-case complexity
- In practice, an algorithm might have vastly different runtimes for different inputs
- Might have algorithms that run quickly "almost" always
- Notion of an "average" runtime might be useful

Towards average-case complexity

• Several NP-hard problems are actually easy on many input instances and have small expected running time.

• The 'expected' running time of a problem depends on the frequency in which the inputs appear, i.e. the distribution in which the inputs appear.
Example. Uniform distribution $U = \{U_n\}_{n \geq 1}$: All inputs $x \in \{0,1\}^n$ appear with the same frequency, i.e. U_n is the uniform distribution on $\{0,1\}^n$.

• Thus, when we talk about the average case complexity of a problem we are really talking about the average case complexity of $(L, \{D_n\}_{n \geq 1})$ where L is the language associated with the problem and D_n is the distribution in which the inputs of length n appear.
In regard to this we make the following definition.

Distributional problems

Defn. Distributional Problem

A distributional problem is a pair $\langle L, D\rangle$ such that L is a language (or equivalently, a decision problem) and $D = \{D_n\}$ is a sequence of distributions, with D_n being a distribution on $\{0,1\}^n$ (i.e D_n is a random variable taking values in $\{0,1\}^n$).

Examples:

- (CLIQUE,$\{G(n,p)\}_n$)
- (3-COLOR, $\{G(n,p)\}_n$)
- (3-SAT, $\{U_{m,n}\}_{m,n}$)

A distribution on undirected graphs

$G(n, p)$ is a distribution on undirected graphs with n vertices obtained as follows:

- A random graph can be generated by randomly deciding which edges to include.
- For an undirected graph with n vertices, there are $\binom{n}{2}$ edges
- For each edge, include that edge with probability p.
- The obtained distribution is called $G(n, p)$

Note: Taking $p = 1/2$ will give us the uniform distribution.

Given this setup, we can think about the expected running time of algorithms for 3-COLOR, CLIQUE, INDSET.

Facts:

• 3-COLOR has an algorithm which is linear time with high probability when inputs are sampled using $G(n, p)$.

• CLIQUE and INDSET have algorithms which run in $n^{2 \log n}$ with high probability (where $n^{2 \log n}$ is only a little more than polynomial).

Distributional 3-SAT problem

Let $U_{m,n}$ be a uniform distribution on 3-CNFs having m clauses and n variables. The distribution $U_{m,n}$ is obtained as follows:

- For each clause, pick 3 literals randomly from the $2n$ available literals (n variables $+$ n negated variables) and take their disjunction.

Intuitively as m increases, we expect the fraction of satisfiable 3-CNFs having m clauses and n variables to go down.

Fact: There exist constants $c_1 < c_2$ such that if $m < c_1 n$, then the fraction of satisfiable 3-CNFs is very low and if $m > c_2 n$, then the fraction of satisfiable 3-CNFs is very high.

Defining dist**P**: average-case analog of P

We now define the average-case analog of P that aims to capture the set of distributional problems (L, D) that are efficiently solvable, i.e. "polynomial time on average".

Given a distributional problem (L, D) and an algorithm A, let $time_A(x)$ denote the number of steps A takes on input x.

Defn. Class distP** (First attempt)**

A distributional problem (L, D) is in dist**P** if there is a constant c and an algorithm A which decides L and satisfies:

$$\mathbb{E}[time_A(D_n)] \leq O(n^c)$$

This definition seems appealing at first but it has a few issues.

Defining dist**P**: average-case analog of P

Defn. Class dist**P** (First attempt)

A distributional problem (L, D) is in dist**P** if there is a constant c and an algorithm A which decides L and satisfies:

$$\mathbb{E}[time_A(D_n)] \leq O(n^c)$$

Issues with the definition:

- This definition is not robust. If we change the model of computation to a different model with quadratic slow down (for example, change from multiple tape Turing machines to one-tape Turing machines), then a polynomial-time algorithm can suddenly turn into an exponential-time algorithm as we shall soon see.
- Using this definition it is difficult to define average-case reductions.

Defining dist**P**: average-case analog of P

Defn. Class dist**P** (First attempt)

A distributional problem (L, D) is in dist**P** if there is a constant c and an algorithm A which decides L and satisfies:

$$\mathbb{E}[time_A(D_n)] \leq O(n^c)$$

Claim: There is an algorithm A such that for every n we have $\mathbb{E}[time_A(U_n)] \leq n+1$ but $\mathbb{E}[time_A^2(U_n)] \geq 2^n$.

Proof: Let A be a problem where if $x = 00\cdots0$, the algorithm halts in 2^n steps. Else, it halts in n^2 steps
Then,

$$\mathbb{E}[time_A(D_n)] = 2^{-n} \cdot 2^n + \frac{2^n - 1}{2^n} \cdot n^2 \leq n^2$$

But,

$$\mathbb{E}[time_A^2(D_n)] = 2^{-n} \cdot 2^{2n} + \frac{2^n - 1}{2^n} \cdot n^4 \geq 2^n \quad \square$$

Defining dist**P**: average-case analog of P

Defn. Class dist**P** (First attempt)

A distributional problem (L, D) is in dist**P** if there is a constant c and an algorithm A which decides L and satisfies:

$$\mathbb{E}[time_A(D_n)] \leq O(n^c)$$

Claim: There is an algorithm A such that for every n we have $\mathbb{E}[time_A(U_n)] \leq n + 1$ but $\mathbb{E}[time_A^2(U_n)] \geq 2^n$.

Thus, this definition is not robust to model changes. More generally, in this definition, if we perform a polynomial time (in expectation) algorithm polynomially many times, the whole process is not poly-time. We now seek a more robust definition for dist**P**.

Defining dist**P**: average-case analog of P

Defn. Class dist**P**

A distributional problem (L, D) is in dist**P** if there is a constant c and an algorithm A which decides L and satisfies:

$$\mathbb{E}\left[time_A(D_n)^{\frac{1}{c}}\right] \leq O(n)$$

Defining dist**P**: average-case analog of P

Defn. Class dist**P**

A distributional problem (L, D) is in dist**P** if there is a constant c and an algorithm A which decides L and satisfies:

$$\mathbb{E}\left[time_A(D_n)^{\frac{1}{c}}\right] \leq O(n)$$

Observation 1: The definition is robust to changes in computational models: if the running times get squared, we just multiply c by 2 and satisfy the definition again. More clearly,

if $\exists c$ s.t. $\mathbb{E}\left[time_A(D_n)^{\frac{1}{c}}\right] \leq O(n)$,

then $\exists c' = 2c$ s.t. $\mathbb{E}\left[time_A^2(D_n)^{\frac{1}{c'}}\right] \leq O(n)$

Note: There is nothing special about 2 and the definition is robust to any polynomial change.

Observations about the class dist**P**

Defn. Class dist**P**
A distributional problem (L, D) is in dist**P** if there is a constant c and an algorithm A which decides L and satisfies:

$$\mathbb{E}\left[time_A(D_n)^{\frac{1}{c}}\right] \leq O(n)$$

Observation 2: P $\subseteq$ dist**P**, i.e. given L in P and given any distribution $\{D_n\}$, we have $(L, \{D_n\})$ is in dist**P**.
This is because given L in P, there is a constant c and an algorithm A s.t. $\mathrm{time}_A(x) \leq O(n^c)$ for all x of size n. Thus $\mathbb{E}\left[time_A(D_n)^{\frac{1}{c}}\right] \leq O(n)$ and $(L, \{D_n\})$ is in dist**P**.

Observations about the class dist**P**

Defn. Class dist**P**
A distributional problem (L, D) is in dist**P** if there is a constant c and an algorithm A which decides L and satisfies:

$$\mathbb{E}\left[time_A(D_n)^{\frac{1}{c}}\right] \leq O(n)$$

Observation 3: Given $(L, \{D_n\})$, if there is an algorithm A which decides L in expected $O(n^c)$ steps, then $(L, \{D_n\})$ is in dist**P** as by Hölder's inequality we have

$$\mathbb{E}\left[time_A(D_n)^{\frac{1}{c}}\right] \leq (\mathbb{E}[time_A(D_n)])^{\frac{1}{c}} = O(n)$$

Thus, our new definition is a relaxation of our previous attempt.

Observations about the class dist**P**

Defn. Class dist**P**
A distributional problem (L, D) is in dist**P** if there is a constant c and an algorithm A which decides L and satisfies:

$$\mathbb{E}\left[time_A(D_n)^{\frac{1}{c}}\right] \leq O(n)$$

Observation 4: There is a high probability that the algorithm runs in polynomial time.
Mathematically, using Markov's inequality, we have

$$\mathbb{P}[t_A(D_n) \geq T] = \mathbb{P}[t_A(D_n)^{\frac{1}{c}} \geq T^{\frac{1}{c}}] \leq \mathbb{E}[t_A(D_n)^{\frac{1}{c}}] \cdot \frac{1}{T^{\frac{1}{c}}}$$
$$= O(n) \cdot \frac{1}{T^{\frac{1}{c}}}$$

Formalizing “Real-life distributions”

Given a string x and a string length n, We define the cumulative probability

$$\mu_{D_n}(x) = \sum_{y \leq x} \mathbb{P}[D_n = y]$$

where y are all the strings $\leq x$ in lexicographic order. ($\leq$ being the lexicographic order)

Defn. P-computable distributions

We say that the distribution D is poly-time computable if given n, x, t, we can compute μ_{D_n} to t-digit precision in time polynomial in n and t

Note:
This restriction is as strong as requiring the probability $\mathbb{P}[D_n = x]$ to be polytime computable since $\mathbb{P}[D_n = x]$ can be computed as $\mu_{D_n}(x) - \mu_{D_n}(x-1)$, where $x - 1$ is the string before x in lexicographic order.

Formalizing "Real-life distributions"

Given a string x and a string length n, We define the cumulative probability

$$\mu_{D_n}(x) = \sum_{y \leq x} \mathbb{P}[D_n = y]$$

where y are all the strings $\leq x$ in lexicographic order. ($\leq$ being the lexicographic order)

Defn. P-computable distributions

We say that the distribution D is poly-time computable if given n, x, t, we can compute μ_{D_n} to t-digit precision in time polynomial in n and t

Examples:

• The distribution $U = \{U_n\}_n$ where U_n is the uniform distribution on $\{0,1\}^n$ is P-computable.

• Many other distributions that are defined using explicit formulas are P-computable.

Computable Distributions

Motivation for Definition

In the following section, we will define the NP version of average case problem. There, we will only look at Distributions that are poly-time computable.
This is because of the following reasons:

- It is a reasonable assumption that the problem instances were either generated by someone while inputting into a program and thus, it is reasonable to put computability restrictions on the generated distribution. (Since the generator, presumably does not have infinite computing power)

Computable Distributions

Motivation for Definition

In the following section, we will define the NP version of average case problem. There, we will only look at Distributions that are poly-time computable.
This is because of the following reasons:

- It is a reasonable assumption that the problem instances were either generated by someone while inputting into a program and thus, it is reasonable to put computability restrictions on the generated distribution. (Since the generator, presumably does not have infinite computing power)
- There exist distributions where every problem is as hard on average as the worst case. So, unless we restrict the kinds of distributions we take, average case analysis will be equivalent to worst case analysis for the most part and we'd have sort of wasted our time.

dist**NP** - average-case analog of NP

Defn. Class distNP

A distributional problem $\langle L, D \rangle$ is in dist**NP** if $L \in NP$ and D is P-computable.

Examples:

- (CLIQUE,$\{G(n,p)\}_n$)
- (3-COLOR, $\{G(n,p)\}_n$)
- (3-SAT, $\{U_{m,n}\}_{m,n}$)

Questions:

- What are the hardest problems in this class?
- Is there a dist**NP**-complete problem?
- Is dist**P** = dist**NP**?

Average-case reduction

We want to define average-case polynomial reduction (or simply average-case reduction).

Properties we want from a reduction

Let $\langle L, D\rangle$ and $\langle L', D'\rangle$ be two distributional problems and let $f : L \to L'$ be a reduction.
We would naturally expect the following properties:

- $x \in L \iff f(x) \in L'$
- $|f(x)| = p(|x|)$
- if $\langle L', D'\rangle$ is "easy", then so is $\langle L, D\rangle$.

Average-case reduction

To capture "if $\langle L', D' \rangle$ is "easy", then so is $\langle L, D \rangle$, we ask f to be poly-time computable.
Is this sufficient? Let's see.

Average-case reduction

To capture "if $\langle L', D' \rangle$ is "easy", then so is $\langle L, D \rangle$, we ask f to be poly-time computable.
Is this sufficient? Let's see.

- Assume there is an efficient algorithm A' for $\langle L', D' \rangle$.
- Let A be"obvious" algorithm for the problem $\langle L, D \rangle$, i.e. on input x compute $y = f(x)$ and run algorithm A' on y.

Is A an efficient algorithm for $\langle L, D \rangle$?

Average-case reduction

To capture "if $\langle L', D' \rangle$ is "easy", then so is $\langle L, D \rangle$, we ask f to be poly-time computable.
Is this sufficient? Let's see.

- Assume there is an efficient algorithm A' for $\langle L', D' \rangle$.
- Let A be"obvious" algorithm for the problem $\langle L, D \rangle$, i.e. on input x compute $y = f(x)$ and run algorithm A' on y.

Is A an efficient algorithm for $\langle L, D \rangle$?
Not always. There is a possibility that A' is very slow on some input that is unlikely to be sampled according to distribution D' but has a high probability of showing up as $f(x)$ when we sample x according to D.
To prevent this from happening we introduce a domination condition in our definition of average-case reduction.

Average-case reduction

To capture "if $\langle L', D' \rangle$ is "easy", then so is $\langle L, D \rangle$, we ask f to be poly-time computable.
Is this sufficient? Let's see.

- Assume there is an efficient algorithm A' for $\langle L', D' \rangle$.
- Let A be"obvious" algorithm for the problem $\langle L, D \rangle$, i.e. on input x compute $y = f(x)$ and run algorithm A' on y.

Is A an efficient algorithm for $\langle L, D \rangle$?
Not always. There is a possibility that A' is very slow on some input that is unlikely to be sampled according to distribution D' but has a high probability of showing up as $f(x)$ when we sample x according to D.
To prevent this from happening we introduce a domination condition in our definition of average-case reduction.

Domination condition

For every $y \in \{0,1\}^{p(n)}$, $\mathbb{P}[f(D_n) = y] \leq q(n)\, \mathbb{P}[D'_{p(n)} = y]$

Average-case reduction

Defn. Average-case reduction

We say that a distributional problem $\langle L, D\rangle$ average-case reduces to a distributional problem $\langle L', D'\rangle$, denoted by $\langle L, D\rangle \leq_a \langle L', D'\rangle$, if there is a polynomial-time computable f and polynomials $p, q : \mathbb{N} \to \mathbb{N}$ satisfying

- (Correctness) For every $x \in \{0,1\}^*, x \in L \iff f(x) \in L'$
- (Length regularity) $|f(x)| = p(|x|)$
- (Domination) For every $n \in \mathbb{N}$ and $y \in \{0,1\}^{p(n)}$, $\mathbb{P}[f(D_n) = y] \leq q(n)\, \mathbb{P}[D'_{p(n)} = y]$

Theorem: If $\langle L, D\rangle \leq_a \langle L', D'\rangle$ and $\langle L', D'\rangle \in \text{dist}\mathbf{P}$ then $\langle L, D\rangle \in \text{dist}\mathbf{P}$.

“if (L', D') is easy, then so is (L, D)” when $\langle L, D\rangle \leq_a \langle L', D'\rangle$

Theorem: If $\langle L, D\rangle \leq_a \langle L', D'\rangle$ and $\langle L', D'\rangle \in \text{dist}\mathbf{P}$ then $\langle L, D\rangle \in \text{dist}\mathbf{P}$.

Proof: As $\langle L', D'\rangle \in \text{dist}\mathbf{P}$, there is an algorithm A' which decides L', and constants C', ϵ' s.t. for every m

$$\mathbb{E}[\text{time}_{A'}(D'_m)^{\epsilon'}] \leq C'm.$$

Let f be the reduction from $\langle L, D\rangle$ to $\langle L', D'\rangle$, and let A be the “obvious” algorithm for deciding L: Given input x it computes $f(x)$ and then outputs $A'(f(x))$. From defn, A decides L. Thus it remains to show that A is efficient, i.e. $\exists C, \epsilon$ s.t. for every n

$$\mathbb{E}[\text{time}_A(D_n)^{\epsilon}] \leq Cn.$$

Now note

$$\text{time}_A(x) = \text{time to compute } f(x) \ + \ \text{time}_{A'}(f(x))$$

"if (L', D') is easy, then so is (L, D)" when $\langle L, D\rangle \leq_a \langle L', D'\rangle$

Theorem: If $\langle L, D\rangle \leq_a \langle L', D'\rangle$ and $\langle L', D'\rangle \in \mathrm{dist}\mathbf{P}$ then $\langle L, D\rangle \in \mathrm{dist}\mathbf{P}$.

Proof:

- $$\begin{aligned} \mathrm{time}_A(x) &= \text{time to compute } f(x) \ + \ \mathrm{time}_{A'}(f(x)) \\ &\leq h(|x|)\ \mathrm{time}_{A'}(f(x)) \qquad \text{(for some polynomial } h(\cdot)\text{)} \end{aligned}$$

- For simplicity assume: $|f(x)| = |x|^d$, for every x.

Let $\epsilon := \epsilon'$ and observe:

$$\begin{aligned} \mathbb{E}[\mathrm{time}_A(D_n)^\epsilon] &= \sum_{x\in\{0,1\}^n} \mathbb{P}[D_n = x]\ \mathrm{time}_A(x)^\epsilon \\ &\leq \sum_{x\in\{0,1\}^n} \mathbb{P}[D_n = x]\ \ (h(n)\ \mathrm{time}_{A'}(f(x)))^\epsilon \\ &= \sum_{x\in\{0,1\}^n} \mathbb{P}[f(D_n) = f(x)]\ h(n)^\epsilon\ \mathrm{time}_{A'}(f(x))^\epsilon \end{aligned}$$

“if (L', D') is easy, then so is (L, D)” when $\langle L, D\rangle \leq_a \langle L', D'\rangle$

Theorem: If $\langle L, D\rangle \leq_a \langle L', D'\rangle$and $\langle L', D'\rangle \in$ dist**P** then $\langle L, D\rangle \in$ dist**P**.

Proof:

$$\begin{aligned}
\mathbb{E}[\text{time}_A(D_n)^\epsilon] &= \sum_{x\in\{0,1\}^n} \mathbb{P}[D_n = x]\ \text{time}_A(x)^\epsilon \\
&= \sum_{x\in\{0,1\}^n} \mathbb{P}[f(D_n) = f(x)]\ h(n)^\epsilon\ \text{time}_{A'}(f(x))^\epsilon \\
&= \sum_{y\in\{0,1\}^{n^d}} \mathbb{P}[f(D_n) = y]\ h(n)^\epsilon\ \text{time}_{A'}(y)^\epsilon \\
&\leq \sum_{y\in\{0,1\}^{n^d}} q(n)\mathbb{P}[D'_{n^d} = y]\ h(n)^\epsilon\ \text{time}_{A'}(y)^\epsilon \\
&= h(n)^\epsilon q(n)\mathbb{E}[\text{time}_{A'}(D'_{n^d})^\epsilon] \\
&\leq C' h(n)^\epsilon q(n) n^d
\end{aligned}$$

"if (L', D') is easy, then so is (L, D)" when $\langle L, D\rangle \leq_a \langle L', D'\rangle$

Lemma: Say A is an algorithm for a distributional problem (L, D) and say there are constants ϵ_1, C_1, d such that $\mathbb{E}[\text{time}_A(D_n)^{\epsilon_1}] \leq C_1 n^d$, then $\exists \epsilon_2, C_2$ s.t. $\mathbb{E}[\text{time}_A(D_n)^{\epsilon_2}] \leq C_2 n$

Proof:

Fact: If X is a non-negative random variable and $d \geq 1$, then $\mathbb{E}[X]^d \leq \mathbb{E}[X^d]$.

Note:

$$\mathbb{E}\left[\frac{\text{time}_A(D_n)^{\epsilon_1}}{n^d}\right] \leq C_1$$

$$\implies \mathbb{E}\left[\frac{\text{time}_A(D_n)^{\epsilon_1/d}}{n}\right]^d \leq \mathbb{E}\left[\frac{\text{time}_A(D_n)^{\epsilon_1}}{n^d}\right] \leq C_1$$

$$\implies \mathbb{E}\left[\frac{\text{time}_A(D_n)^{\epsilon_1/d}}{n}\right] \leq C_1^{1/d} \qquad \square$$

There exists a dist**NP**-complete problem

Theorem

There exists a distributional problem that is dist**NP**-complete.

Proof: We do this by defining a language NBH and a distribution U such that (NBH, U) is in dist**NP** and every (L, D) in dist**NP** can be average-case reduced to (NBH, U).

There exists a dist**NP**-complete problem

Theorem

There exists a distributional problem that is dist**NP**-complete.

Proof: We do this by defining a language NBH and a distribution U such that (NBH, U) is in dist**NP** and every (L, D) in dist**NP** can be average-case reduced to (NBH, U).

Non-deterministic bounded halting (NBH) problem

$NBH = \{\langle M, x, 1^t\rangle : \ M$ is a NTM that accepts x in atmost t steps$\}$

- NBH is in NP as we can use a universal NTM to simulate M on x for t steps of M using $t^{O(1)}$ steps of the universal NTM.
- NBH is NP-hard because given L in NP, there is NTM M_L which decides L in $p(|x|)$ time. Thus we can (Karp) reduce L to NBH using

$$x \mapsto (M_L, x, 1^{p(|x|)}).$$

There exists a dist**NP**-complete problem

A "roughly-uniform" distribution U on instances of the NBH problem

To sample from U_n, we first pick at random three positive integers a, b, c such that $a + b + c = n$, then we sample a random $M \in \{0,1\}^a$ and $x \in \{0,1\}^b$, and we output $(M, x, 1^c)$. So we have

$$\mathbb{P}[U_n = (M, x, 1^c)] = \Theta\left(\frac{1}{n^2}\right) \cdot \frac{1}{2^{|M|}} \cdot \frac{1}{2^{|x|}}$$

- U is P-computable, hence (NBH, U) is in dist**NP**.

There exists a dist**NP**-complete problem

A "roughly-uniform" distribution U on instances of the NBH problem

To sample from U_n, we first pick at random three positive integers a, b, c such that $a + b + c = n$, then we sample a random $M \in \{0,1\}^a$ and $x \in \{0,1\}^b$, and we output $(M, x, 1^c)$. So we have

$$\mathbb{P}[U_n = (M, x, 1^c)] = \Theta\left(\frac{1}{n^2}\right) \cdot \frac{1}{2^{|M|}} \cdot \frac{1}{2^{|x|}}$$

- U is P-computable, hence (NBH, U) is in dist**NP**.

(NBH, U) is dist**NP**-complete

Let (L, D) be a distributional problem in dist**NP**, i.e. L is in NP and D is P-computable.We need to find an average-case reduction from (L, D) to (NBH, U). Will the reduction from before work as an average-case reduction? i.e. Does the map $x \mapsto (M_L, x, 1^{p(|x|)})$ satisfy the domination condition? (Spoiler: No)

There exists a dist**NP**-complete problem

We will soon see that the map $f : x \mapsto (M_L, x, 1^{p(|x|)})$ satisfies the domination condition only when for all x we have

$$\mathbb{P}[D_n = x] \leq O\left(\frac{n^{O(1)}}{2^{|x|}}\right)$$

There are many P-computable distributions for which the above is false.

Example: Consider a distribution D defined by:

$$\mathbb{P}[D_n = x] = \begin{cases} 1/2 & \text{if } x = 00\cdots0 \\ 1/2 & \text{if } x = 11\cdots1 \\ 0 & \text{else} \end{cases}$$

Here, clearly

$$\mathbb{P}[D_n = 00\cdots0] = \frac{1}{2} \nleq O\left(\frac{n^{O(1)}}{2^n}\right) \qquad \text{(for large enough } n\text{)}$$

There exists a dist**NP**-complete problem

• We will soon see that the map $f : x \mapsto (M_L, x, 1^{p(|x|)})$ satisfies the domination condition only when for all x we have

$$\mathbb{P}[D_n = x] \leq O\left(\frac{n^{O(1)}}{2^{|x|}}\right)$$

There are many distributions for which the above is false.

• Now if we instead use the map $f : x \mapsto (N_L, C(x), 1^{p'(|x|)})$, then f satisfies the domination condition only when for all x we have

$$\mathbb{P}[D_n = x] \leq O\left(\frac{n^{O(1)}}{2^{|C(x)|}}\right)$$

There exists a dist**NP**-complete problem

• Now if we instead use the map $f : x \mapsto (N_L, C(x), 1^{p'(|x|)})$, then f satisfies the domination condition only when for all x we have

$$\mathbb{P}[D_n = x] \leq O\left(\frac{n^{O(1)}}{2^{|C(x)|}}\right)$$

• We will soon construct an "invertible compression" algorithm C that satisfies the above identity. The compression algorithm will be such that

- $x \in L \iff (N_L, C(x), 1^{p'(|x|)}) \in NBH$
- $|C(x)|$ is "small" when $\mathbb{P}[D_n = x]$ is "big"
  (i.e. when $P[D_n = x] \geq 1/2^{|x|}$).
- $|C(x)| \approx |x|$ when $\mathbb{P}[D_n = x]$ is "small"
  (i.e. when $P[D_n = x] \leq 1/2^{|x|}$).

This reduction, f, will turn out to be a valid average-case reduction from (L, D) to (NBH, U).

There exists a dist**NP**-complete problem

Details part 1: $\mathbb{P}[D_n = x] \leq O\left(\frac{n^{O(1)}}{2^{|x|}}\right)$

Assume the map $f : x \mapsto (M_L, x, 1^{p(|x|)})$ satisfies the domination condition, i.e. there is a polynomial q s.t. for all y

$$\mathbb{P}[f(D_n) = y] \leq q(n)\, \mathbb{P}\left[U_{|M_L|+n+p(n)} = y\right]$$

Take $y = (M_L, x, 1^{p(|x|)})$ and note:

$$\mathbb{P}\left[U_\star = (M_L, x, 1^{p(|x|)})\right] = \Theta(1) \cdot \frac{1}{p(n)^2} \cdot \frac{1}{2^{M_L}} \cdot \frac{1}{2^{|x|}}$$

$$\mathbb{P}\left[f(D_n) = (M_L, x, 1^{p(|x|)})\right] = \mathbb{P}[D_n = x] \qquad \text{(As } f \text{ is one-to-one)}$$

Thus we get,

$$\mathbb{P}[D_n = x] \leq q(n) \cdot \Theta(1) \cdot \frac{1}{p(n)^2} \cdot \frac{1}{2^{M_L}} \cdot \frac{1}{2^{|x|}}$$

$$\leq O\left(\frac{n^{O(1)}}{2^{|x|}}\right)$$

There exists a dist**NP**-complete problem

Details part 2: the "invertible compression" algorithm C

Lemma: Let D is a polynomial-time computable distribution. Then there exists a polynomial-time algorithm C satisfying:

- $C(x) = (|x|, C_0(x))$ for all x.
- C is injective: $C(x) = C(y) \iff x = y$
- $|C(x)| \leq \log|x| + 1 + \min\left\{|x|, \log\left(\frac{1}{\mathbb{P}[D_n=x]}\right)\right\}$

We shall postpone the proof and finish the other details.

There exists a dist**NP**-complete problem

Details part 3: The NTM N_L

Let N_L be a NTM that on input (n, z) does the following:

- Guesses a string x of length n.
- Computes $C(x)$. If $C(x) \neq (n, z)$, then the machine halts and outputs 0.
- Else, It simulates M_L on input x and returns $M_L(x)$.

(i.e. N_L computes $C^{-1}(n, z)$ by guessing and simulates M_L on it.)

Observation 1: There is a polynomial $p'(\cdot)$ such that N_L halts in at most $p'(n)$ steps for input (n, z).

Observation 2: N_L accepts $C(x) = (|x|, C_0(x))$ in $p'(|x|)$ steps iff M_L accepts x in $p(|x|)$ steps.
Note: The $\Rightarrow$ direction of the proof relies on C being one-to-one.

There exists a dist**NP**-complete problem

Details part 4: $f : x \mapsto (N_L, C(x), 1^{p'(|x|)})$ works as an average-case reduction

Let $f(x) := (N_L, C(x), 1^{p'(|x|)})$.

Length regularity: $|f(x)| = |N_L| + |C(x)| + |p'(x)| = |x|^{O(1)}$

Correctness: $x \in L \iff f(x) \in NBH$. This is because:

$$\begin{aligned} x \in L &\iff M_L \text{ accepts } x \text{ in } p(|x|) \text{ steps} \\ &\iff N_L \text{ accepts } C(x) \text{ in } p'(|x|) \text{ steps} \\ &\iff (N_L, C(x), 1^{p'(|x|)}) \in NBH \end{aligned}$$

Domination condition: We want for all y:

$$\mathbb{P}[f(D_n) = y] \le q(n)\, \mathbb{P}[U_\star = y] \qquad (*)$$

If $y \neq (N_L, C(x), 1^{p'(n)})$ for some x, then $(*)$ trivially holds as LHS becomes 0.

Assume $y = (N_L, C(x), 1^{p'(n)})$ for some x.

There exists a dist**NP**-complete problem

Details part 4: $f : x \mapsto (N_L, C(x), 1^{p'(|x|)})$ works as an average-case reduction

Domination condition: We want for all y:

$$\mathbb{P}\left[f(D_n) = y\right] \leq q(n)\ \mathbb{P}\left[U_\star = y\right] \qquad (*)$$

Assume $y = (N_L, C(x), 1^{p'(n)})$ for some x.

Note:

$$\mathbb{P}\left[f(D_n) = (N_L, C(x), 1^{p'(n)})\right] = \mathbb{P}[D_n = x]$$

$$\begin{aligned}\mathbb{P}\left[U_\star = (N_L, C(x), 1^{p'(n)})\right] &= \Theta(1) \cdot \frac{1}{p'(|x|)^2} \cdot \frac{1}{2^{N_L}} \cdot \frac{1}{2^{|C(x)|}} \\ &\geq \Theta(1) \cdot \frac{1}{p'(|x|)^2} \cdot \frac{1}{2^{N_L}} \cdot \frac{1}{2^{\log|x|+1+\min\left\{|x|,\log\left(\frac{1}{\mathbb{P}[D_n=x]}\right)\right\}}} \\ &\geq \Theta(1) \cdot \frac{1}{p'(|x|)^2} \cdot \frac{1}{2^{N_L}} \cdot \frac{1}{2|x|} \cdot \mathbb{P}[D_n = x] \qquad \blacksquare\end{aligned}$$

There exists a dist**NP**-complete problem

Lemma: Let D is a polynomial-time computable distribution. Then there exists a polynomial-time algorithm C satisfying:

- $C(x) = (|x|, C_0(x))$ for all x.
- C is injective: $C(x) = C(y) \iff x = y$
- $|C(x)| \leq \log|x| + 1 + \min\left\{|x|, \log\left(\frac{1}{\mathbb{P}[D_n=x]}\right)\right\}$

Proof:

Let

$$C_0(x) := \begin{cases} 1x & \text{when } \mathbb{P}[D_n = x] \leq 2^{-|x|} \\ 0\,h(x) & \text{when } \mathbb{P}[D_n = x] > 2^{-|x|} \end{cases}$$

where $h(x)$ is a poly-time computable function s.t. h is one-to-one on inputs of same length and $h(x)$ has "small" length.

Let $\mu_n(x) = \mathbb{P}[D_n \leq x]$ and let $h(x)$ be the longest common prefix of $\mu_n(x)$ and $\mu_n(x-1)$ when both are written out in binary.

There exists a dist**NP**-complete problem

$h(x)$ is the longest common prefix of $\mu_n(x)$ and $\mu_n(x-1)$ after the decimal point when both are written out in binary.

Observations about h:

- $h(x)$ is poly-time computable as $\mu_n(x)$ and $\mu_n(x-1)$ are poly-time computable.
- h is one-to-one on inputs of same length as given $h(x_1) = h(x_2) = s$ where $|x_1| = |x_2|$, we must have:

$$\begin{aligned} \mu_n(x_1 - 1) &= 0.s\,0 * * \cdots \\ \mu_n(x_1) &= 0.s\,1 * * \cdots \\ \mu_n(x_2 - 1) &= 0.s\,0 * * \cdots \\ \mu_n(x_2) &= 0.s\,1 * * \cdots \end{aligned}$$

  As μ_n is monotonic we have $x_1 - 1 < x_2$ and $x_2 - 1 < x_1$ from which it follows that $x_1 = x_2$.

There exists a dist**NP**-complete problem

$h(x)$ is the longest common prefix of $\mu_n(x)$ and $\mu_n(x-1)$ after the decimal point when both are written out in binary.

Observations about h:

- $h(x)$ is poly-time computable as $\mu_n(x)$ and $\mu_n(x-1)$ are poly-time computable.
- h is one-to-one on inputs of same length.
- $|h(x)| \leq \log\left(\frac{1}{\mathbb{P}[D_n=x]}\right)$
  This is because
  $$|h(x)| > \log\left(\frac{1}{\mathbb{P}[D_n=x]}\right)$$
  $$\Rightarrow \mu_n(x) - \mu_n(x-1) < 2^{-\log\left(\frac{1}{\mathbb{P}[D_n=x]}\right)}$$
  $$\Rightarrow \mu_n(x) - \mu_n(x-1) < \mathbb{P}[D_n=x]$$
  $$\Rightarrow \mathbb{P}[D_n=x] < \mathbb{P}[D_n=x]\text{, a contradiction.}$$

There exists a dist**NP**-complete problem

$h(x)$ is the longest common prefix of $\mu_n(x)$ and $\mu_n(x-1)$ after the decimal point when both are written out in binary.

Observations about h:

- $h(x)$ is poly-time computable as $\mu_n(x)$ and $\mu_n(x-1)$ are poly-time computable.
- h is one-to-one on inputs of same length.
- $|h(x)| \leq \log\left(\frac{1}{\mathbb{P}[D_n = x]}\right)$

Recall:

$$C(x) := \begin{cases} (|x|, 1x) & \text{when } \mathbb{P}[D_n = x] \leq 2^{-|x|} \\ (|x|, 0\,h(x)) & \text{when } \mathbb{P}[D_n = x] > 2^{-|x|} \end{cases}$$

Hence C is a poly-time computable injective function satisfying $|C(x)| \leq \log|x| + 1 + \min\left\{|x|, \log\left(\frac{1}{\mathbb{P}[D_n = x]}\right)\right\}$
$\square$

References

References

- Trevisan, L. 2014, *Notes for Lecture 8 & 9*, lecture notes, CS254: Computational Complexity, Stanford University. [web]
- Arora, Sanjeev. *Computational Complexity: A Modern Approach.*