Computational Complexity
计算复杂性

张方国

中山大学计算机学院

isszhfg@mail.sysu.edu.cn

Lecture 8. Variations on P and NP

- Non-uniform polynomial-time (P/poly)
- The Polynomial-time Hierarchy (PH)
- The P/poly-versus-NP Question and PH

Non-uniform polynomial-time (P/poly)

Non-uniform polynomial-time(P/poly) captures efficient computations that are carried out by devices that can each only handle inputs of a specific length. The basic formalism ignore the complexity of constructing such devices (i.e., a uniformity condition). A finer formalism that allows to quantify the amount of non-uniformity refers to so called machines that take advice.

The two models of non-uniform computing devices are Boolean circuits and "machines that take advice".

The main motivation for considering non-uniform polynomial-size circuits is that their computational limitations imply analogous limitations on polynomial-time algorithms.

The main motivation for considering polynomial time algorithms that take polynomially bounded advice is that such devices are useful in modeling auxiliary information that is available to possible efficient strategies that are of interest to us.

Boolean Circuits

Definition

A Boolean circuit(n input m output) induces(or computes) a function from $\{0,1\}^n$ to $\{0,1\}^m$: For any fixed string $x \in \{0,1\}^n$, we iteratively define the value of vertices in the circuit such that the input terminals are assigned the corresponding bits in $x = x_1 \cdots x_n$ and the values of other vertices are determined in the natural manner. That is:

- An input terminal with label $i \in \{1, 2, \ldots, n\}$ is assigned the i^{th} bit of x (i.e., the value x_i).
- If the children of a gate(of in-degree d) that is labeled $\wedge$ have values $v_1, v_2, \ldots, v_d$, then the gate is assigned the value $\wedge_{i=1}^{d} v_i$. The value of a gate labeled $\vee$(or $\neg$) is determined analogously.

Definition (Circuit families)

We say that a family of circuits $(C_n)_{n\in\mathbb{N}}$ computes a function $f : \{0,1\}^* \to \{0,1\}^*$ if for every n the circuit C_n computes the restriction of f to strings of length n. In other words, for every $x \in \{0,1\}^*$, it must hold that $C_{|x|}(x) = f(x)$.

Theorem (circuit evaluation)

There exists a polynomial-time algorithm that, given a circuit $C : \{0,1\}^n \to \{0,1\}^m$ and an n-bit long string x, returns $C(x)$.

The size of a circuit is the number of its edges. When considering a family of circuits $(C_n)_{n\in\mathbb{N}}$ that computes a function $f : \{0,1\}^* \to \{0,1\}^*$, we are interested in the size of C_n as a function of n. Specifically, we say that this family has size complexity $s : \mathbb{N} \to \mathbb{N}$ if for every n the size of C_n is $s(n)$.

Definition (Circuit complexity)

The circuit complexity of $f : \{0,1\}^* \to \{0,1\}^*$ is the function $s_f : \mathbb{N} \to \mathbb{N}$ such that $s_f(n)$ is the size of the smallest circuit that computes the restriction of f to n-bit strings.

We will be interested in the class of problems that are solvable by families of polynomial-size circuits, That is, a problem is solvable by polynomial-size circuits if it can be solved by a function f that has polynomial circuit complexity.

Definition (uniformity)

A family of circuits $(C_n)_{n\in\mathbb{N}}$ is called uniform if there exists an algorithm A that on input n outputs C_n within a number of steps that is polynomial in the size of C_n.

Proposition

If a problem is solvable by **a uniform family** *of polynomial-size circuits then it is solvable by a polynomial-time algorithm.*

这里是多项式规模电路的一致类，不是多项式规模电路。

Proof: On input x, the algorithm operates in two stages. In the first stage, it invokes the algorithm guaranteed by the uniformity condition, on input $n = |x|$, and obtains the circuit C_n. Next, it invokes the circuit evaluation algorithm on input C_n and x, and obtains $C_n(x)$. Since the size and the description length of C_n are polynomial in n, it follows that each stage of our algorithm runs in polynomial time (i.e., polynomial in $n = |x|$) Thus, the algorithm emulates the computation of $C_{|x|}(x)$, and does so in time polynomial in the length of its own input (i.e., x).

Machines that take advice

General(non-uniform) families of polynomial-size circuits and uniform families of polynomial-size circuits are two extremes with respect to the "amounts of non-uniformity" in the computing device. Intuitively, in the former, non-uniformity is only bounded by the size of the device, whereas in the latter the amounts of non-uniformity is zero.

Here we consider a model that allows to decouple the size of the computing device from the amount of non-uniformity, which may indeed range from zero to the device's size. Specifically, we consider algorithms that "take a non-uniform advice" that depends only on the input length.

Turing machines that "take advice" : such a machine has, for each n, an advice string α_n, which it is allowed to use in its computation whenever the input has size n.

Definition (taking advice)

We say that algorithm A computes the function f using advice of length $l : \mathbb{N} \to \mathbb{N}$ if there exists an infinite sequence $(\alpha_n)_{n\in\mathbb{N}}$ such that:

- For every $x \in \{0,1\}^*$, it holds that $A(\alpha_{|x|}, x) = f(x)$.
- For every $n \in \mathbb{N}$, it holds that $|\alpha_n| = l(n)$.

The sequence$(\alpha_n)_{n\in\mathbb{N}}$ is called the advice sequence.

Definition 3.5 (non-uniform polynomial-time and $\mathcal{P}/\text{poly}$): *We say that a function f is* computed in polynomial-time with advice of length $\ell : \mathbb{N} \to \mathbb{N}$ *if these exists a polynomial-time algorithm A and an infinite* advice sequence $(a_n)_{n\in\mathbb{N}}$ *such that*

1. *For every $x \in \{0,1\}^*$, it holds that $A(a_{|x|}, x) = f(x)$.*
2. *For every $n \in \mathbb{N}$, it holds that $|a_n| = \ell(n)$.*

We say that a computational problem can be solved in polynomial-time with advice of length ℓ if a function solving this problem can be computed within these resources. We denote by $\mathcal{P}/\ell$ the class of decision problems that can be solved in polynomial-time with advice of length ℓ, and by $\mathcal{P}/\text{poly}$ the union of $\mathcal{P}/p$ taken over all polynomials p.

Relation to families of polynomial-size circuits:

The class of problems solvable by polynomial-time algorithms with polynomially bounded advice equals the class of problems solvable by families of polynomial-size circuits.

Theorem

A decision problem is in $\mathcal{P}/poly$ if and only if it can be solved by a family of polynomial-size circuits.

Proof Sketch: Suppose that a problem can be solved by a polynomial-time algorithm A using the polynomially bounded advice sequence $(a_n)_{n\in\mathbb{N}}$. We obtain a family of polynomial-size circuits that solves the same problem by adapting the proof of Theorem 2.20. Specifically, we observe that the computation of $A(a_{|x|}, x)$ can be emulated by a circuit of poly($|x|$)-size, *which incorporates* $a_{|x|}$ *and is given* x *as input*. That is, we construct a circuit C_n such that $C_n(x) = A(a_n, x)$ holds for every $x \in \{0,1\}^n$ (analogously to the way C_x was constructed in the proof of Theorem 2.20, where it holds that $C_x(y) = M_R(x, y)$ for every y of adequate length).

On the other hand, given a family of polynomial-size circuits, we obtain a polynomial-time algorithm for emulating this family *using advice that provide the description of the relevant circuits*. Specifically, we use the evaluation algorithm asserted in Theorem 3.1, while using the circuit's description as advice. That is, we use the fact that a circuit of size s can be described by a string of length $O(s \log s)$, where the log factor is due to the fact that a graph with v vertices and e edges can be described by a string of length $2e \log_2 v$. □

$\mathcal{P}/1 \subseteq \mathcal{P}/poly$的两个定义：

1，被多项式规模的电路族计算的问题

2，被带一个额外的多项式大小“advice”的图灵机多项式时间计算的问题。

Theorem (the power of advice)

There exist functions that can be computed using one-bit advice but cannot be computed without advice.

Theorem (the power of advice, revisited)

The class $\mathcal{P}/1 \subseteq \mathcal{P}/poly$ contains $\mathcal{P}$ as well as some undecidable problems.

Proof: Clearly, $\mathcal{P} = \mathcal{P}/0 \subseteq \mathcal{P}/1 \subseteq \mathcal{P}/\text{poly}$. To prove that $\mathcal{P}/1$ contains some undecidable problems, we review the proof of Theorem 1.13. The latter proof established the existence of uncomputable Boolean function that only depend on their input length. That is, there exists an undecidable set $S \subset \{0,1\}^*$ such that for every pair of equal length strings (x, y) it holds that $x \in S$ if and only if $y \in S$. In other words, for every $x \in \{0,1\}^*$ it holds that $x \in S$ if and only if $1^{|x|} \in S$. But such a set is easily decidable in polynomial-time by a machine that takes one bit of advice; that is, consider the algorithm A and the advice sequence $(a_n)_{n\in\mathbb{N}}$ such that $a_n = 1$ if and only if $1^n \in S$ and $A(a, x) = a$ (for $a \in \{0,1\}$ and $x \in \{0,1\}^*$). Note that indeed $A(a_{|x|}, x) = 1$ if and only if $x \in S$. ■

The inclusion $\mathcal{P} \subseteq \mathcal{P}/poly$ is proper.

For instance, there are unary problems that are undecidable and hence are not in $\mathcal{P}$, whereas every unary problem is in $\mathcal{P}/poly$.

Claim

Let $L \subseteq \{0,1\}^*$ be a unary problem, then $L \in \mathcal{P}/poly$.

A problem is called unary if every string in it is of the form 1^i (the string of i ones) for some $i > 0$.

The Polynomial-time Hierarchy (PH)

Sets in $\mathcal{NP}$ can be viewed as sets of valid assertions that can be expressed as quantified Boolean formulae using only existential quantifiers.

That is, a set S is in $\mathcal{NP}$ if there is a Karp-reduction of S to the problem of deciding whether or not an existentially quantified Boolean formula is valid (i,e,. an instance x is mapped by this reduction to a formula of the form $\exists y_1 ... \exists y_{m(x)} \phi_x(y_1 ... y_{m(x)}))$

Alternation of quantifiers

Definition 3.8 (the class Σ_k): *For a natural number k, a decision problem $S \subseteq \{0,1\}^*$ is in Σ_k if there exists a polynomial p and a polynomial time algorithm V such that $x \in S$ if and only if*

$$\exists y_1 \in \{0,1\}^{p(|x|)} \forall y_2 \in \{0,1\}^{p(|x|)} \exists y_3 \in \{0,1\}^{p(|x|)} \cdots Q_k y_k \in \{0,1\}^{p(|x|)} \quad \text{s.t. } V(x, y_1, ..., y_k) = 1$$

where Q_k is an existential quantifier if k is odd and is a universal quantifier otherwise.

Note that $\Sigma_1 = \mathcal{NP}$ and $\Sigma_0 = \mathcal{P}$. **The Polynomial-time Hierarchy**, denoted $\mathcal{PH}$, is the union of all the aforementioned classes (i.e., $\mathcal{PH} = \cup_k \Sigma_k$ and Σ_k is often referred to as the k-th level of $\mathcal{PH}$. The levels of the Polynomial-time Hierarchy can also be defined inductively, by defining Σ_{k+1} based on $\Pi_k = co\Sigma_k$ where $co\Sigma_k = \{\{0,1\}^* \backslash S : S \in \Sigma_k\}$.

Proposition 3.9 *For every $k \geq 0$, a set S is in Σ_{k+1} if and only if there exists a polynomial p and a set $S' \in \Pi_k$ such that $S = \{x : \exists y \in \{0,1\}^{p(|x|)} \text{ s.t. } (x,y) \in S'\}$.*

Proof: Suppose that S is in Σ_{k+1} and let p and V be as in Definition 3.8. Then define S' as the set of pairs (x,y) such that $|y| = p(|x|)$ and

$$\forall z_1 \in \{0,1\}^{p(|x|)} \exists z_2 \in \{0,1\}^{p(|x|)} \cdots Q_k z_k \in \{0,1\}^{p(|x|)} \text{ s.t. } V(x,y,z_1,...,z_k) = 1\,.$$

Note that $x \in S$ if and only if there exists $y \in \{0,1\}^{p(|x|)}$ such that $(x,y) \in S'$, and that $S' \in \Pi_k$ (see Exercise 3.6).

On the other hand, suppose that for some polynomial p and a set $S' \in \Pi_k$ it holds that $S = \{x : \exists y \in \{0,1\}^{p(|x|)} \text{ s.t. } (x,y) \in S'\}$. Then, for some p' and V', it holds that $(x,y) \in S'$ if and only if $|y| = p(|x|)$ and

$$\forall z_1 \in \{0,1\}^{p'(|x|)} \exists z_2 \in \{0,1\}^{p'(|x|)} \cdots Q_k z_k \in \{0,1\}^{p'(|x|)} \text{ s.t. } V'(x,y,z_1,...,z_k) \neq 1$$

(see Exercise 3.6 again). By suitable encoding (of y and the z_i's as strings of length $\max(p(|x|), p'(|x|))$) and a trivial modification of V', we conclude that $S \in \Sigma_{k+1}$.

Proposition 3.10 *For every $k \geq 1$, if $\Sigma_k = \Pi_k$ then $\Sigma_{k+1} = \Sigma_k$, which in turn implies $\mathcal{PH} = \Sigma_k$.*

The converse also holds (i.e., $\mathcal{PH} = \Sigma_k$ implies $\Sigma_{k+1} = \Sigma_k$ and $\Sigma_k = \Pi_k$). Needless to say, Proposition 3.10 does not seem to hold for $k = 0$.

Proof: Assuming that $\Sigma_k = \Pi_k$, we first show that $\Sigma_{k+1} = \Sigma_k$. For any set S in Σ_{k+1}, by Proposition 3.9, there exists a polynomial p and a set $S' \in \Pi_k$ such that $S = \{x : \exists y \in \{0,1\}^{p(|x|)} \text{ s.t. } (x,y) \in S'\}$. Using the hypothesis, we infer that $S' \in \Sigma_k$, and so (using Proposition 3.9 and $k \geq 1$) there exists a polynomial p' and a set $S'' \in \Pi_{k-1}$ such that $S' = \{x' : \exists y' \in \{0,1\}^{p'(|x'|)} \text{ s.t. } (x',y') \in S''\}$. It follows that

$$S = \{x : \exists y \in \{0,1\}^{p(|x|)} \exists z \in \{0,1\}^{p'(|(x,y)|)} \text{ s.t. } ((x,y),z) \in S''\}.$$

By collapsing the two adjacent existential quantifiers (and using Proposition 3.9 yet again), we conclude that $S \in \Sigma_k$. This proves the first part of the proposition.

Turning to the second part, we note that $\Sigma_{k+1} = \Sigma_k$ (or, equivalently, $\Pi_{k+1} = \Pi_k$) implies $\Sigma_{k+2} = \Sigma_{k+1}$ (again by using Proposition 3.9), and similarly $\Sigma_{j+2} = \Sigma_{j+1}$ for any $j \geq k$. Thus, $\Sigma_{k+1} = \Sigma_k$ implies $\mathcal{PH} = \Sigma_k$. ■

Non-deterministic oracle machines

The Polynomial-time Hierarchy is commonly defined in terms of non-deterministic polynomial-time (oracle) machines that are given oracle access to a set in the lower level of the same hierarchy. Such machines are defined by combining the definitions of non-deterministic (polynomial-time) machines (cf. Definition 2.7) and oracle machines (cf. Definition 1.11). Specifically, for an oracle $f : \{0,1\}^* \to \{0,1\}^*$, a non-deterministic oracle machine M, and a string x, one considers the question of whether or not there exists an accepting (non-deterministic) computation of M on input x and access to the oracle f. The class of sets that can be accepted by non-deterministic polynomial-time (oracle) machines with access to f is denoted $\mathcal{NP}^f$. (We note that this notation makes sense because we can associate the class $\mathcal{NP}$ with a collection of machines that lends itself to be extended to oracle machines.) For any class of decision problems $\mathcal{C}$, we denote by $\mathcal{NP}^{\mathcal{C}}$ the union of $\mathcal{NP}^f$ taken over all decision problems f in $\mathcal{C}$. The following result provides an alternative definition of the Polynomial-time Hierarchy.

Proposition 3.11 *For every $k \geq 1$, it holds that $\Sigma_{k+1} = \mathcal{NP}^{\Sigma_k}$.*

The P/poly-versus-NP Question and PH

As stated, a main motivation for the definition of $\mathcal{P}/poly$ is the hope that it can serve to separate $\mathcal{P}$ from $\mathcal{NP}$ (by showing that $\mathcal{NP}$ is not even contained in $\mathcal{P}/poly$, which is a superset of $\mathcal{P}$).

Theorem

If $\mathcal{NP} \subseteq \mathcal{P}/poly$ *then* $\Sigma_2 = \Pi_2$.

$\mathcal{P} \subseteq \mathcal{P}/poly$. Karp-Lipton theorem 表明若$\mathcal{NP} \subseteq \mathcal{P}/poly$，则多项式层级（polynomial hierarchy）将会坍缩至第二层，这是一个不大可能的结果。 这两个结果结合起来表明，P/poly可以当作是分离NP与P的一个中间的工具，具体的途径就是证明任一个NP完全问题的电路大小的下界。

在1980年代，电路复杂性途径取得了一系列的成功，其中包括奇偶性函数（Parity function）在AC0 中的下界为指数，以及团问题（clique problem）在单调性电路（monotone circuit）中的下界为指数。

然而在1994年Razborov和Rudich的著名论文自然性证明（Natural proof）中指出，上面所用证明电路下界的方法，在单向函数存在的前提下是不可能分离NP和P的。该结果使很多专家对证明电路下界来分离NP和P的前景表示不乐观。

Thank You Very Much!