CS 710: Complexity Theory	Date: Feb. 27th, 2020
Lecture 12: Chee Yap Theorem, Mahaney's theorem, and Tail bound	
Instructor: Jin-Yi Cai	Scribe: Jialu Bao

The document has not been scrutinized as peer-reviewed works would have. The scribe is responsible for any errors within it.

1 Sparse Set

We first discussed sparse set, which gives an equivalent construction of $P/_{Poly}$.

Definition 1. *Sparse set S is a set of strings such that there exists a polynomial $p(n)$, the number of strings of length n in S is bounded by $p(n)$ for all $n \in \mathbb{N}$. Let $S^{=n} = \{x \in S \mid |x| = n\} \subseteq \Sigma^n$, then being sparse means that $|S^{=n}|$ is bounded by a polynomial p on n.*

Then we made the following claim,

Claim 1. *The condition $SAT \in P/_{Poly}$ in Karp-Lipton Theorem is equivalent to say that there exists a sparse set S such that $SAT = L(M^S)$ where M is a deterministic polynomial time Turing machine.*

Proof. If $SAT \in P/_{Poly}$, then there exists polynomial sized advice $y(n)$, and a deterministic polynomial time Turing Machine M such that M accepts $(x, y(|x|))$ if and only if $x \in SAT$. Then we can design the following set S,

$$S = \{1^n \# p \mid p \textbf{ is the prefix of } \mathbf{y(n)}\}$$

Given n, we can query the oracle S if $1^n\#0$ in S, if so, continue querying $1^n\#00$, and otherwise query $1^n\#01$... By $|y(n)|$ such queries to the oracle, we can recover $y(n)$ and use it as the advice, and as the size $y(n)$ is bounded by polynomial, a deterministic polynomial time machine suffices. S is sparse because for every string size m, there is at most one string in S with the prefix $1^n\#$ for any $n < m - 1$, so the total number of size m string in S is less than m.

In the other direction, there exists a sparse set S such that $SAT = L(M^S)$ and $|S^{=n}| \leq p(n)$, we want to design some advice $y(n)$ such that M accepts $(x, y(|x|))$ if and only if $x \in SAT$. Given input x, any input M can make to ask for its membership in S is of size polynomial of $|x|$ – say it's bounded by $q(n)$. Since $|S^{=n}| \leq p(n)$ implies that $|S^{\leq n}| \leq n \times p(n)$, the number of strings of size at most $q(n)$ that is in S is at most $n \times p(q(n))$, which is also polynomial time to n. Thus, we can list all such strings with polynomial number of n bits. This list $y'(n)$ can be used as the advice – when trying to decide x, whenever needed to query the oracle for the membership of some s, if suffices to scan $y'(|x|)$ to see if s is listed in $y'(|x|)$. By doing that, it accepts $(x, y'(n))$ if and only if $x \in L(M^S)$. □

With the same technique, we can show that $SAT \in$ co-NP if and only if there exists a polynomial time non-deterministic Turing machine N_0 such that $\overline{SAT} = L(N_0^S)$.

2 Chee Yap Theorem

Chee Yap used similar techniques used in Karp-Lipton theorem and proved the following

Theorem 1. *If $NP \subseteq$ co-$NP/_{Poly}$, then $\Sigma_3^P = \Pi_3^P$, and consequently, the whole polynomial hierarchy collapsed to the third level.*

Proof. $NP \subseteq$ co-$NP/_{Poly}$ is equivalent to $SAT \in$ co-$NP/_{Poly}$, which is also equivalent to there exists a polynomial time non-deterministic Turing machine N_0 and a sparse set oracle S such that $\overline{SAT} = L(N_0^S)$.In first-order logic language, we have

$$\begin{aligned}\exists N_0 \in NTM,\ \text{sparse set } S \\ \forall^P \Psi\ \Big(\big(\exists^P \sigma_1 \Psi\ |_{\sigma_1} = 1 \wedge \forall^P (\text{path } p)\ N_0(\Psi, p, S^{\leq p^*(|\Psi|)})\ \text{rejects}\big) \\ \vee\ \big(\forall^P \sigma_2 \Psi\ |_{\sigma_2} = 0 \wedge \exists^P (\text{path } q)\ N_0(\Psi, q, S^{\leq p^*(|\Psi|)})\ \text{accepts}\big)\Big)\end{aligned}$$

where $p^*(n)$ is a generic notation for polynomials and bounds the size of query that N_0 makes to S. We can move all quantifiers ahead and transform the formula to the following Prenex normal form,

$$\begin{aligned}\exists N_0 \in NTM,\ \text{sparse set } S \\ \forall^P \Psi\ \forall^P \text{path } p\ \forall^P \sigma_2\ \exists^P \sigma_1 \exists^P \text{path } q\ \Big(\big(\Psi\ |_{\sigma_1} = 1 \wedge N_0(\Psi, p, S^{\leq p^*(|\Psi|)})\ \text{rejects}\big) \\ \vee\ \big(\Psi\ |_{\sigma_2} = 0 \wedge N_0(\Psi, q, S^{\leq p^*(|\Psi|)})\ \text{accepts}\big)\Big)\end{aligned}$$

Collapsing $\forall$'s and $\exists$', we can transform our assumption to the following formula,

$$\begin{aligned}\exists N_0 \in NTM,\ \text{sparse set } S \\ \forall^P \Psi, \text{path } p, \sigma_2\ \exists^P \sigma_1, \text{path } q\ \Big(\big(\Psi\ |_{\sigma_1} = 1 \wedge N_0(\Psi, p, S^{\leq p^*(|\Psi|)})\ \text{rejects}\big) \\ \vee\ \big(\Psi\ |_{\sigma_2} = 0 \wedge N_0(\Psi, q, S^{\leq p^*(|\Psi|)})\ \text{accepts}\big)\Big) \qquad (1)\end{aligned}$$

where quantification p, q, σ_2, σ_1 can be freely reordered. Then we can use eq. (1) to show that $\Pi_3^P \subseteq \Sigma_3^P$.

Any problem T in Π_3^P can be formulated as

$$T = \{w \mid \forall^P x\ \exists^P y\ \forall^P z\ D(w, x, y, z,)\}$$

where D is a polynomial time decidable problem by a deterministic Turing machine. By Cook reduction, given any w, x, y, we can find a formula $\Psi'_{w,x,y}$ that is unsatisfiable if and only if $\forall^P z\ D(w, x, y, z)$. Thus, there exists $N_0 \in NTM$, sparse set S such that for any w, x, y, there exists path in N_0^S accepts $\Psi_{w,x,y}$ if and only if the formula $\Psi_{w,x,y}$ is unsatisfiable, or equivalently, $\forall^P z\ D(w, x, y, z)$. Thus, by eq. (1), T can be rewriten as

$$\begin{aligned}T = \{w \mid \exists N_0 \in NTM, \exists^P \text{ sparse set } S, \\ \forall^P \Psi,\ \text{path } p, \sigma_2, \exists^P \sigma_1, \text{path } q\ \Big(\big(\Psi\ |_{\sigma_1} = 1 \wedge N_0(\Psi, p, S^{\leq p^*(|\Psi|)})\ \text{rejects}\big) \\ \vee\ \big(\Psi\ |_{\sigma_2} = 0 \wedge N_0(\Psi, q, S^{\leq p^*(|\Psi|)})\ \text{accepts}\big)\Big) \\ \wedge\ \big(\forall^P x\ \exists^P y,\ \text{path } r,\ N_0^S(\Psi_{w,x,y}, r, S^{\leq p^*(|\Psi_{w,x,y}|)})\ \text{accepts }\big)\}\end{aligned}$$

This simplifies to,

$$T = \{w \mid \exists N_0 \in NTM, \exists^P \text{ sparse set } S, \\ \forall^P x, \exists^P y, \text{path } q \ \forall^P \sigma_2, \big(\Psi_{w,x,y} \mid_{\sigma_2=0} \wedge N_0(\Psi_{w,x,y}, , q, S^{\leq p^*(|\Psi|)}) \text{ accepts}\big)$$

Instead of checking whether $\Psi_{\sigma_2=0}$ for all σ_2, we can make a curcuit that generates a certificate every time N_0 accepts using self-reducibility and verify the circuit using the certificate. Thus,

$$T = \{w \mid \exists^P C, \exists^P \text{ sparse set } S, \\ \forall^P x, \exists^P y, \text{path } q \ \big(\Psi_{w,x,y} \mid_{\sigma_2=0} \wedge N_0(\Psi_{w,x,y}, , q, S^{\leq p^*(|\Psi|)}) \text{ accepts}\big) \\ \subseteq \Sigma_3^P$$

Σ_3^P is the complement of Π_3^P, so $\Pi_3^P \subseteq \Sigma_3^P$ implies $\Pi_3^P = \Sigma_3^P$. □

3 Generalization

In the polynomial hierarchy,

Definition 2.

$$\Theta_i^P := \Sigma_i^P \cap \Pi_i^P$$
$$\Delta_k^P := P^{\Sigma_{k-1}^P}$$

Equivalently, $\Delta_k^P = P^{\Pi_{k-1}^P}$ as the polynomial Turing machine can always flip the input to and the answer given by the oracle. Furthermore, by this definition, for all i,

$$\Delta_i^P = P^{\Sigma_{k-1}^P} \subseteq NP^{\Sigma_{k-1}^P} \\ = \underbrace{NP^{NP^{\cdots^{NP}}}}_{k} = \Sigma_k^P \Delta_i^P \qquad\qquad = P^{\Pi_{k-1}^P} \subseteq \text{co-}NP^{\Pi_{k-1}^P} = \Pi_k^P$$

So $\Delta_k^P \subseteq \Theta_k^P$.

Recall that Karp-Lipton theorem is saying that, if $\Sigma_1^P \subset \Delta_1^{P,S}$ for some sparse set S, then the polynomial hierarchy collapsed to $\Sigma_2^P = \Pi_2^P$. So a natural generalization is what $\Sigma_k^P \subset \Delta_k^P$ implies for other $k \in \mathbb{N}$?

Similarly, the generalized version of Chee Yap theorem asks what is the implication if $\Sigma_k^P \subset \Pi_k^{P,S}$ for some sparse set oracle S.

Some results in computational complexity could be surprising, for example, if $NP \cap \text{co-}NP \neq P$, then there are problems that can be decided in polynomial time but we can't extract a certificate of in polynomial time. In other word, self-reducibility breaks down.

4 Steve Maheney's theorem

As illustrated in claim 1, $NP \subseteq P/_{Poly}$ is equivalent to saying every NP problem can be reduced to some sparse set problem under Cook reduction, which is also equivalent to there exists a sparse set S that is NP-complete under *Cook reduction*. While Karp-Lipton theorem answeres the implications

of that condition, Maheney asks what's the implications that there exists a sparse set S that is NP-complete under *Karp reduction*. Unlike Cook reduction that may call O multiple times in reduction to problem O, Karp recuction makes only one call to O and use the answer from O directly as the answer for the original problem. Formally,

Definition 3 (Karp reduction)**.** *Karp reduction from decision problem A to B is a polynomial-time function f : alphabet$(A) \to$ alphabet(B) such that x is accepted in problem A if and only if $f(x)$ is accepted in B. It is sometimes denoted as $A \leq_m^P B$ or $A \leq_p B$.*

Mahaney shows the following

Theorem 2. *There exists sparse set S that is NP-complete under Karp-reduction if and only if $NP = P$.*

Below we present a simplified proof credit to Ogihara-Watanabe.

Proof. ($\Leftarrow$:) If $NP = P$, then there exists polynomial time algorithm f for all NP-complete problems. Thus, a trivial sparse set $\{1\}$ suffices.

($\Rightarrow$:) Assuming there exists sparse set S that is NP-complete under Karp-reduction, we want to show $NP = P$.

Define total order $\prec$ such that for $\tau, \sigma \in \{0,1\}^*$, $\tau \prec \sigma$ when either σ is a prefix of τ, or there exists $i \geq 1$ such that for all $1 \leq j < i$ $\sigma_j = \tau_j$ and $\sigma_i = 0$, $\tau_i = 1$.

Define the decision problem

$$LSAT : \{\langle \Psi, \sigma \rangle \mid \text{there exists full assignment } \tau \prec \sigma \text{ such that } \tau \text{ is a satisfying assignment of } \Psi \}$$

Note that $LSAT$ is an NP-complete problem, because for any given Ψ, whether Ψ in SAT could be reduced to whether (Ψ, ϵ) is in $LSAT$, and a satisfying full assignment can provide a polynomial time certificate.

By assumption, there exists a sparse set S and a polynomial-time function f, such that $\langle \Psi, \sigma \rangle \in LSAT$ if and only if $f(\langle \Psi, \sigma \rangle) \in S$. Since f is polynomial time algorithm, the size of $f(x)$ is also bounded by polynomial of $|x|$ for any input x. Say $|f(x)| \leq p(|x|)$. By sparseness of S, there exists polynomial $q(n)$ such that $|S^{\leq n}| \leq q(n)$. Given formula Ψ with k variables, let $N = q(p(|\Psi| + k))$, then N upper bounds the number of element in S with size at most $p(|\Psi| + k)$, and thus upper bounds the number of element in S such that there exists partial assignments σ with $f(\Psi, \sigma) \in S$.

Now we can devise a polynomial time algorithm that solves SAT by exploring partial assignments of a formula ordered in a tree and pruning the tree at each level. Let each node in the tree corresponds to a partial assignment. At the root of the tree sits the empty assignment ϵ, and at leaves sit full assignments. Each node v that is not leaf have two children, $v0$ and $v1$, which extends v with assignments to the next unassigned variable.

We explore partial assignments given by nodes on the tree level by level. At each level h , we evaluate $f(v_i)$ for each v_i in that level,

- If the number of nodes in that level, 2^h, does not exceed N, then we go to the next level without pruning.
- Otherwise, we prune in the following ways,
  - If $f(v_i) = f(v_j)$ for $v_i \prec v_j$, then prune v_j and its descendent. Repeat until there's no more duplicates.

– If after deduplication, there are still more than N distinct $f(v_i)$'s, then prune the left-most, i.e. , smallest w.r.t. order $\prec$ and and its descedents. Repeat until there's no more than N branches remained.

We claim that this pruning procedure maintain the following invariant: if there exists satisfying assignments, the smallest (w.r.t. order $\prec$) satisfying assignment's ancestor at every level is not pruned. This invariant is maintained because

- When we pruned v_j because $f(v_i) = f(v_j)$ for some $i \prec j$, if v_j has satisfying descendents, then $f(v_j) \in S$, and thus $f(v_i) \in S$, indicating v_i also has satisfying descendents, which would be smaller than any of v_j's descendents.

- When we pruned v_i at level h because at least N nodes at level h bigger than v_i are mapped to distinct values in S, if v_i is the ancestor of the smallest satisfying assignments, then $\langle \Psi, v_i \rangle \in LSAT$ and $f(v_i) \in S$. As the unpruned nodes v_j all satisfies $v_i \prec v_j$, by definition of $LSAT$, $\langle \Psi, v_i \rangle \in LSAT$ implies that $\langle \Psi, v_j \rangle \in LSAT$ and $f(v_j) \in LSAT$. At least N such v_j with distinct $f(v_j)$ then implies that there exists more than N elements in S with size at most $p(|\psi| + k)$, contracting to the assumption that S is sparse.

  Thus, the left most/smallest v_i must not be the ancestor of the smallest satisfying assignments, and is safe to pruned.

With this invariant maintained, the smallest satisfying assignment will not get pruned throughout the process. Thus, when reaching the leaf level, there exists any unpruned full assignments satisfying Ψ if and only if there exists any full assignments satisfying Ψ.So this algorithm gives correct answer.

Also, its runtime is polynomial. At each level, it calculates $f(v)$ for at most $2N$ vertices v, and prunes at most N times – there are at most N unpruned vertices at the last level and thus at most $2N$ vertices at this level are remained to get scrutinized. Each run of f is polynomial time, and pruning any one vertex is at most $N \log(N)$ time, with N polynomial to the size of Ψ, so the computations at each level are in polynomial time. There are $k < |\Psi|$ levels in total, so the total time is also polynomial to the input size.

□

5 Useful inequalities in probability

In preparation of learning probabilistic complexity class, we learned about inequalities frequently used in reasoning about probability.

The bound gets tighter when the random variable is the sum of a bunch of independent random variables. Consider a coin flip with probability $\frac{1}{2}$ to get a head $+1$, and probability $\frac{1}{2}$ to get a tail -1. Tossing the coin n times, and let X_i records outcome in toss i. Let $S_n = X_1 + ... + X_n$. The expectation follows from the linearity of expectation: $E[S_n] = \sum_{i=1}^{n} E[X_i] = 0$. To study how concentrated X is around 0, we may use the knowledge that $\frac{S_n}{n}$ is a bionomial distrbution and converges towards normal distribution $\mathcal{N}(0, 1)$ as n converges to infinity. Then,

$$\lim_{n\to\infty} Pr\Big[\frac{S_n}{\sqrt{n}} \in (a, b)\Big] = \int_a^b \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}$$

However, the limit $n \to \infty$ here is not satisfying when we want to know how concentrate is X for small n. Thus, tail probability estimate such as Markov Inequallity.

Theorem 3 (Markov Inequality)**.** *Given non-negative random variable x whose expected value is finite,*

$$Pr[x \geq a] \leq \frac{E[x]}{a}$$

for all $a > 0$.

The proof is straightforward,

Proof. By definition,

$$\begin{aligned} E[x] &= \int_{\mathbb{R}} x \cdot Pr[x]dx \\ &\geq \int_{a} x \cdot Pr[x]dx \textbf{ because } x \geq 0 \\ &\geq a \cdot \int_{a} Pr[x]dx \textbf{ because } a > 0 \\ &= a \cdot Pr[x \geq a] \end{aligned}$$

Thus, $Pr[x \geq a] \leq \frac{E[x]}{a}$. □

However, Markov inequality requires the random variable to be non-negative and thus do not apply to our motivating example. Furthermore, Markov inequality does not use the condition that S_n is the sum of independent variables. With that assumption, we can achieve a sharper bound.

Claim 2 (Chernoff Bound)**.** *In the motivating example*

$$Pr[S_n \geq \Delta] \leq e^{-\frac{\Delta^2}{2n}}$$

Proof. Whether S_n is non-negative or not, $e^{\lambda S_n}$ is non-negative, and thus, by Markov inequality,

$$Pr[S_n \geq \Delta] = Pr\left[e^{\lambda S_n} \geq e^{\lambda \Delta}\right] \leq \frac{E[e^{\lambda S_n}]}{e^{\lambda \Delta}}$$

As $S_n = X_1 + ... + X_n$,

$$\begin{aligned} E[e^{\lambda S_n}] &= E[e^{\lambda X_1 + ... + \lambda X_n}] \\ &= E[e^{\lambda X_1} ... e^{\lambda X_n}] \end{aligned}$$

As X_i's are independent, $e^{\lambda X_i}$ are also independent, and thus,

$$E[e^{\lambda X_1} ... e^{\lambda X_n}] = E[e^{\lambda X_1}] \cdot E[e^{\lambda X_2}] \cdot ... \cdot E[e^{\lambda X_n}]$$

Recall that each X_i is a random variable with half probability to take value 1 and half probability to take value -1. Thus, for each i, $E[e^{\lambda X_i}] = \frac{1}{2}e^{\lambda} + \frac{1}{2}e^{-\lambda}$. Applying Taylor Expansion for e^x, we have

$$\begin{aligned} e^{\lambda} &= 1 + \lambda + \frac{1}{2!}\lambda^2 + \frac{1}{3!}\lambda^3 + ... \\ e^{-\lambda} &= 1 - \lambda + \frac{1}{2!}\lambda^2 - \frac{1}{3!}\lambda^3 + ... \end{aligned}$$

Thus,

$$\begin{aligned} E[e^{\lambda X_i}] &= \frac{1}{2}e^{\lambda} + \frac{1}{2}e^{-\lambda} \\ &= 1 + \frac{1}{2!}\lambda^2 + \frac{1}{4!}\lambda^4 + ... \\ &= e^{\frac{\lambda^2}{2}} \end{aligned}$$

Thus,

$$\begin{aligned} Pr[S_n \geq \Delta] &\leq \frac{E[e^{\lambda S_n}]}{e^{\lambda\Delta}} \\ &= \frac{e^{\frac{\lambda^2 \cdot n}{2}}}{e^{\lambda\Delta}} \\ &= e^{\frac{\lambda^2 \cdot n}{2} - \lambda\Delta} \end{aligned}$$

Fixed n, Δ, $\frac{\lambda^2 \cdot n}{2} - \lambda\Delta$ is minimized when $\lambda = \frac{\Delta}{n}$, and $\frac{\lambda^2 \cdot n}{2} - \lambda\Delta$ evaluates to $-\frac{\Delta^2}{2n}$. Thus,

$$Pr[S_n \geq \Delta] \leq e^{\frac{\Delta^2}{2n}}$$

□

See wikipedia([4]) for more general forms of Chernoff bound in both additive form and multiplicative form.

5.1 Application: Ramsey's theorem

Ramsey's theorem states that for any $k, l \geq 1$, if n is large enough, then for any complete graph G with more than n nodes colored with red or blue, either there exists a blue clique with k nodes or there exists red clique of size l. Let $R(k, l)$ be the smallest such n that is large enough.

Paul Erdos showed that

$$R(k, l) \leq R(k-1, l) + R(k, l-1)$$

And thus, $R(k, l) \leq \binom{k+l-1}{l-1}$.

Exercise left to readers to apply the tail bound: Color n nodes randomly with red, or blue, what is the possibility that the coloring is not (k, l) Ramsey, i.e., there's no blueclique of size k or red clique of size l?

References

[1] http://www.wisdom.weizmann.ac.il/ oded/CC/mahaney.pdf

[2] http://www.cs.umd.edu/ jkatz/complexity/f05/lecture6.pdf

[3] http://www.cs.cmu.edu/ odonnell/toolkit13/lecture03.pdf

[4] https://en.wikipedia.org/wiki/Chernoff_bound

[5] J.E. Hopcroft. Recent directions in algorithmic research. In *Theoretical Computer Science* 123-134, 1981, Springer

[6] Chee K, Yap. Some consequences of non-uniform conditions on uniform classes Theoretical computer science 287-300, 1983, Elsevier