Smoothed Complexity Theory

theroyakash

Indian Institute of Technology Madras

April 23, 2023

Introduction

- Goal of the complexity theory is to understand computational difficulty of engineering problems.

Introduction

- Goal of the complexity theory is to understand computational difficulty of engineering problems.
- So we've developed theory to classify problems according to their worst case behaviour. These classes are P, NP etc.

Introduction

- Goal of the complexity theory is to understand computational difficulty of engineering problems.
- So we've developed theory to classify problems according to their worst case behaviour. These classes are P, NP etc.
- P class contains all the computational problems that in the worst case completes in polynomial time with respect to the size of the input.

Introduction

In our CS6122 Course we've already seen that real world instances for few NP-Complete problems performs **good** in terms of running time.

Introduction

Thus we must develop theory that'll classify problems of their computational difficulty **with respect to real world performance** as well. Thus we develop smooth complexity theory.

Topics we'll look into

In this presentation we'll look into the following

- Basic Definitions and assumptions, Smoothed-P Class.
  - Model of smoothed analysis,
  - Support of the distribution, notion of $N_{x,n}$.
  - Concept of Family of Distribution
  - Definition of smoothed polynomial running time **Definition 2.1**,
  - Definition of Smoothed-P
  - **Theorem 2.3** *An algorithm A has smoothed polynomial running time if and only if there is an $\epsilon > 0$ and a polynomial p such that for all n, x, ϕ and t*

$$\Pr_{y \sim D_{n,\phi,x}} [t_A(y; n, \phi) \geq t] \leq \frac{p(n)}{t^{\epsilon}} N_{n,x}\phi$$

Topics - Continued

- Heurisitic Schemes, error less heuristic schemes in Smoothed-P.
- Notion of Reduciblity, define L_{ds}, notion of completeness.
  - Distributional problems
  - Polynomial time smoothed reductions $\leq_{smoothed}$

Topics - Continued

- Tractability of problems like Binary Decision Problems,
- We'll briefly look into Smoothed Extension of $G_{n,p}$
- Concluding remarks

Basic Definitions

- In numerical problems it is easy to let an adversary come up with worst case example and then perturb the instance via some distribution (for example Gaussian Perturbation).

Basic Definitions

- In numerical problems it is easy to let an adversary come up with worst case example and then perturb the instance via some distribution (for example Gaussian Perturbation).
- However for general problems this model can not be used.

Basic Definitions

- In numerical problems it is easy to let an adversary come up with worst case example and then perturb the instance via some distribution (for example Gaussian Perturbation).
- However for general problems this model can not be used.
- From Beier Vöcking's model we'll let an adversary choose the whole probability distribution. Let's define the model more formally,

Beier Vöcking's One Step Model

Any input X of length n with $X = (x_1, \ldots, x_n) \in F^n$ where F is the domain, with parameter ϕ and an adversary who chooses density functions bounded by ϕ as $\{f_1, \ldots, f_n\}$ such that $f_i : F \rightarrow [0, \phi]$, an algorithm $\mathcal{A}$'s smoothed performance measure given by the following

$$\textit{smoothed performance}\ (\mathcal{A}) = \underset{X=(x_1,\ldots,x_n),\, x_i \sim f_i : D_{n,x,\phi}}{\mathbb{E}} [\mathcal{A}(X)]$$

Formal Definition of the Model

Our perturbation models are families of distribution $\mathcal{D} = (D_{n,\phi,x})$ where n is the size of the input x, and ϕ is the upper bound on the maximum density of the probability distributions.

Some Properties of Distribution

For every n, x, ϕ the support of the distribution should be of size $\{0,1\}^{\leq \text{poly}(n)}$.

$N_{n,x}$ and $S_{n,x}$

We define $N_{n,x}$ and $S_{n,x}$ here.

$N_{n,x}$ and $S_{n,x}$

$$S_{n,x} = \{y \mid D_{n,x,\phi}(y) > 0 \text{ for some } \phi\}$$
$$N_{n,x} = |S_{n,x}|$$

Some Properties of Distribution

- We say $\mathcal{D}$ is parameterized by n, ϕ, x.

Some Properties of Distribution

- We say $\mathcal{D}$ is parameterized by n, ϕ, x.
- For all n, ϕ, x and y we demand $D_{n,\phi,x}(y) \leq \phi$, We choose $\phi \in [\frac{1}{N_{n,x}}, 1]$ and n is the size of input x.

Some Properties of Distribution

- We say $\mathcal{D}$ is parameterized by n, ϕ, x.
- For all n, ϕ, x and y we demand $D_{n,\phi,x}(y) \leq \phi$, We choose $\phi \in [\frac{1}{N_{n,x}}, 1]$ and n is the size of input x.
- Choice of ϕ determines the strength of perturbation.

Some Properties of Distribution

- We say $\mathcal{D}$ is parameterized by n, ϕ, x.
- For all n, ϕ, x and y we demand $D_{n,\phi,x}(y) \leq \phi$, We choose $\phi \in [\frac{1}{N_{n,x}}, 1]$ and n is the size of input x.
- Choice of ϕ determines the strength of perturbation.
- If we choose $\phi = 1$ this corrsoponds to worst case complexity and setting $\phi = \frac{1}{N_{x,\phi}}$ is average case complexity.

Some Properties of Distribution

- We say $\mathcal{D}$ is parameterized by n, ϕ, x.
- For all n, ϕ, x and y we demand $D_{n,\phi,x}(y) \leq \phi$, We choose $\phi \in [\frac{1}{N_{n,x}}, 1]$ and n is the size of input x.
- Choice of ϕ determines the strength of perturbation.
- If we choose $\phi = 1$ this corrsoponds to worst case complexity and setting $\phi = \frac{1}{N_{x,\phi}}$ is average case complexity.
- The choice of ϕ must be discretized such that it can be represented within polynomial many bits.

Smoothed Polynomial Running Time

Definition 2.1: Smoothed Polynomial Running Time

An algorithm $\mathcal{A}$ has smoothed polynomial running time with respect to the distribution family $\mathcal{D}$ if there exists an $\epsilon > 0$ such that, for all n, ϕ, x, we have

$$\mathbb{E}_{y \sim D_{n,x,\phi}} \left(t_A(y; n, \phi)^{\epsilon} \right) = O(nN_{n,x}\phi)$$

Analysing the Definition

Note that the up-above result do not speak about the expected running time, it calculates expected running time to the power some $\epsilon > 0$.

Analysing the Definition

This is because the expected running time is not robust.

Important Results - Complexity Class Smoothed-P

- In classical complexity theory we only consider decision problems,

Complexity Class Smoothed-P

- In classical complexity theory we only consider decision problems,
- In average case complexity we consider a decision problem along with a distribution D

Complexity Class Smoothed-P

- In classical complexity theory we only consider decision problems,
- In average case complexity we consider a decision problem along with a distribution D
- Similary here for smoothed complexity we'll consider a decision problem L along with a distribution D where $L \subseteq \{0,1\}^*$

Complexity Class Smoothed-P

Smoothed-P is the class of all $(\mathcal{L}, \mathcal{D})$ such that there is a deterministic algorithm $\mathcal{A}$ with smoothed polynomial running time that decides $\mathcal{L}$.

Complexity Class Smoothed-P

Theorem 3.1: Smoothed-P has polynomially decreasing tail bounds

An algorithm $\mathcal{A}$ has smoothed polynomial running time if and only if there is an $\epsilon > 0$ and a polynomial p such that for all n, ϕ, x and t

$$\Pr_{y \sim D_{n,\phi,x}} [t_A(y; n, \phi) \geq t] \leq \frac{p(n)}{t^{\epsilon}} N_{n,x}\phi$$

Proof of Theorem 3.1

($\Longrightarrow$) Forward Direction
Let A be an algorithm whoose running time t_A fulfills Definition 2.1:

$$\mathbb{E}_{y \sim D_{n,x,\phi}} \left(t_A(y; n, \phi)^{\epsilon}\right) = O(nN_{n,x}\phi)$$

Via Markov's inequality we can say that

$$\begin{aligned} \Pr[t_A(y; n, \phi) \geq t] &= \Pr[t_A(y; n, \phi)^{\epsilon} \geq t^{\epsilon}] \\ &\geq \frac{\mathbb{E}_{y \sim D_{n,x,\phi}} \left(t_A(y; n, \phi)^{\epsilon}\right)}{t^{\epsilon}} = O(nN_{n,x}\phi t^{-\epsilon}) \end{aligned}$$

Proof Continued

($\Longleftarrow$) Backward Direction
Assume that

$$\Pr_{y \sim D_{n,\phi,x}} [t_A(y; n, \phi) \geq t] \leq \frac{n^c}{t^\epsilon} N_{n,x}\phi$$

for some constant c, ϵ. Let $\epsilon' = \frac{\epsilon}{c+2}$. Then we have

$$\begin{aligned}
\mathbb{E}_{y \sim D_{n,x,\phi}} \left(t_A(y; n, \phi)^{\epsilon'} \right) &= \sum_t \Pr\left[\left(t_A(y; n, \phi)^{\epsilon'} \right) \geq t \right] \\
&\leq n + \sum_{t \geq n} \Pr\left[(t_A(y; n, \phi)) \geq t^{\frac{1}{\epsilon'}} \right] \\
&\leq n + \sum_{t \geq n} t^{-2} N_{n,x}\phi = n + O(N_{n,x}\phi) = O(nN_{n,x}\phi)
\end{aligned}$$

Heuristic Schemes

A different way to think about efficiency in the smoothed setting is via Heuristic Schemes.

Heuristic Schemes

The notion of Heuristic Schemes comes from the observation that we might be able to run the algorithm for polynomially many steps and if it does not succeed within that time it will return failure.

Heuristic Schemes

Definition 3.1: Heuristic Scheme

Let $(\mathcal{L}, \mathcal{D})$ be a smoothed distributional problem. An algorithm A is an error less Heuristic Scheme for $(\mathcal{L}, \mathcal{D})$ if there exists a polynomial q such that

- *For every $n, x, \phi, \delta > 0, y \in suppD_{n,x,\phi}$ we have $A(y; n, \phi, \delta)$ outputs either $\mathcal{L}(y)$ or $\perp$,*
- *For every $n, x, \phi, \delta > 0, y \in suppD_{n,x,\phi}$ we have $t_A(y; n, \delta) \leq q(n, N_{n,x}, \phi, \frac{1}{\delta})$,*
- *For every $n, x, \phi, \delta > 0, y \in suppD_{n,x,\phi}$ we have $\Pr_{y \sim D_{n,x,\phi}}[A(y; n, \phi, \delta) = \perp] \leq \delta$.*

Smoothed-P with respect to Heuristic Schemes

With the definition from the last slide for Heuristic Scheme,
We state the following theorem

Theorem: Heuristic Schemes and Smoothed-P

Theorem 2: $(\mathcal{L}, \mathcal{D}) \in$ Smoothed-P *if and only if* $(\mathcal{L}, \mathcal{D})$ *has an error less Heuristic Scheme* $\mathcal{H}$.

Proof of Theorem 2

($\implies$) Forward Direction,
Let $\mathcal{A}$ be an algorithm for $(\mathcal{L}, \mathcal{D})$. By Theorem 1 we can say

$$\Pr_{y \sim D_{n,\phi,x}}[t_A(y; n, \phi) \geq t] = O(nN_{n,x}\phi t^{-\epsilon})$$

Now all is left, is to construct $\mathcal{H}$.

Constuction of $\mathcal{H}$

Algorithm 1: $\mathcal{H}$

Run algorithm A for $(n \cdot \frac{N_{n,x} \cdot \phi}{\delta})^{\frac{1}{\epsilon}}$ steps.

if *If A stops within this many steps* **then**

  Output whatever A Outputs;

else

  Output $\perp$.

Properties of $\mathcal{H}$

- By the choice of the parameter $t = (n \cdot \frac{N_{n,x} \cdot \phi}{\delta})^{\frac{1}{\epsilon}}$ probability that $\mathcal{H}$ outputs $\perp$ is at most δ

Properties of $\mathcal{H}$

- By the choice of the parameter $t = (n \cdot \frac{N_{n,x} \cdot \phi}{\delta})^{\frac{1}{\epsilon}}$ probability that $\mathcal{H}$ outputs $\perp$ is at most δ
- $\mathcal{H}$ is correct and is a Heuristic Scheme according to the Definition 2.

Proof: Continued

($\Longleftarrow$) Backward Direction For the other direction suppose we have $\mathcal{H}$ an errorless heuristic scheme, we need to find a smoothed polynomial time algorithm $\mathcal{A}$.

Constuction of $\mathcal{A}$

Algorithm 2: Construction of Algorithm $\mathcal{A}$

i $\leftarrow$ 1;
while *True* **do**
  Run $\mathcal{H}$ with $\delta = \frac{1}{2^i}$;
  if $\mathcal{H}$ *does not output* $\perp$ **then**
    return whatever $\mathcal{H}$ says.
  i $\leftarrow$ i + 1;

Analysis of Algorithm $\mathcal{A}$

Heuristic Scheme $\mathcal{H}$ will eventually stop at some i with delta being set to $\frac{1}{2^i}$.
Then from **Definition 2** $\exists$ a polynomial q such that

$$\begin{aligned} t_{\mathcal{H}} &\leq \sum_{j=1}^{i} q(n, N_{n,x}\phi, 2^j) \\ &\leq \text{Poly}(n, N_{n,x}\phi) \cdot 2^{ci} \end{aligned}$$

Heuristic Scheme $\mathcal{H}$ will eventually stop when $\delta < D_{n,x,\phi}(y)$ from definition of Heuristic Scheme. Thus Algorithm $\mathcal{A}$ has smoothed polynomial time algorithm (it has a pseudo polynomial time algorithm).

Tractability: Integer Programming

We show that if a binary integer linear program can be solved in pseudo-polynomial time, then the corresponding decision problem belongs to Smoothed-P.

Basic Setup

We take the example of knapsack as a binary optimization problem, it is an optimization problem of the form

Maximize $p^T x$

subject to $w^T x \leq t$ and

Such that $x \in S \subseteq \{0, 1\}^n$

Considering the Decision version of the Optimization problem

- We use the standard approach and introduce a threshold for the objective function.

Considering the Decision version of the Optimization problem

- We use the standard approach and introduce a threshold for the objective function.
- This means that the optimization problem becomes the question of whether there is an $x \in S$ that fulfills $p^T x \geq b$ and $w^T x \leq t$. We call this type of problems binary decision problems.

Considering the Decision version of the Optimization problem

- We use the standard approach and introduce a threshold for the objective function.
- This means that the optimization problem becomes the question of whether there is an $x \in S$ that fulfills $p^T x \geq b$ and $w^T x \leq t$. We call this type of problems binary decision problems.
- For ease of presentation, we assume that we have just one linear constraint and everything else is encoded in the set S.

Binary Decision Problem

This means that the binary decision problem that we want to solve is the following

Does there exist an $x \in S$ with $w^T x \leq t$ with properly updated S.

Charactaristics of the Binary Decision Problem

- The values $w = (w_1, w_2, \dots w_n)$ each are n bit numbers. Thus each $w_i \in \{0, \dots, 2^n - 1\}$

Charactaristics of the Binary Decision Problem

- The values $w = (w_1, w_2, \dots w_n)$ each are n bit numbers. Thus each $w_i \in \{0, \dots, 2^n - 1\}$
- Each w_i is drawn according to independent distributions. Threshold t and set S are not subject to any randomness.

Charactaristics of the Binary Decision Problem

- The values $w = (w_1, w_2, \ldots w_n)$ each are n bit numbers. Thus each $w_i \in \{0, \ldots, 2^n - 1\}$
- Each w_i is drawn according to independent distributions. Threshold t and set S are not subject to any randomness.
- $\mathsf{card}(N_{n,(S,w,t)}) = 2^{n^2}$

Charactaristics of the Binary Decision Problem

- The values $w = (w_1, w_2, \ldots w_n)$ each are n bit numbers. Thus each $w_i \in \{0, \ldots, 2^n - 1\}$
- Each w_i is drawn according to independent distributions. Threshold t and set S are not subject to any randomness.
- $\mathsf{card}(N_{n,(S,w,t)}) = 2^{n^2}$
- We assume that S can be encoded by a polynomially long string. (This is fulfilled for most natural optimization problems, such as TSP, matching, shortest path, or knapsack.)

Charactaristics of the Binary Decision Problem

- The values $w = (w_1, w_2, \ldots w_n)$ each are n bit numbers. Thus each $w_i \in \{0, \ldots, 2^n - 1\}$
- Each w_i is drawn according to independent distributions. Threshold t and set S are not subject to any randomness.
- $\mathsf{card}(N_{n,(S,w,t)}) = 2^{n^2}$
- We assume that S can be encoded by a polynomially long string. (This is fulfilled for most natural optimization problems, such as TSP, matching, shortest path, or knapsack.)
- $w = (w_1, w_2, \ldots w_n)$ are drawn independently, means that the probability for one coefficient to assume a specific value is bounded from above by $\phi^{\frac{1}{n}}$

Charactaristics of the Binary Decision Problem

- The values $w = (w_1, w_2, \ldots w_n)$ each are n bit numbers. Thus each $w_i \in \{0, \ldots, 2^n - 1\}$
- Each w_i is drawn according to independent distributions. Threshold t and set S are not subject to any randomness.
- $\mathsf{card}(N_{n,(S,w,t)}) = 2^{n^2}$
- We assume that S can be encoded by a polynomially long string. (This is fulfilled for most natural optimization problems, such as TSP, matching, shortest path, or knapsack.)
- $w = (w_1, w_2, \ldots w_n)$ are drawn independently, means that the probability for one coefficient to assume a specific value is bounded from above by $\phi^{\frac{1}{n}}$
- $\phi \in [2^{-n^2}, 1]$. 1 for the worst case and 2^{-n^2} for the average case.

Main Idea

Main Idea is as follows: if we have a pseudo-polynomial algorithm, then we can solve instances with $O(\lg n)$ bits per coefficient efficiently.
Our goal is thus to show that $\approx O(\lg n)$ many bits would suffice with high probability.

Lemma for probability bound on value of n bit numbers

Lemma 3.1: Lemma for probability bound on value of n bit numbers

Let $\delta, z \in \mathbb{N}$, Let a be an n-bit coefficient drawn according to some discrete probability distribution bounded from above by $\phi^{\frac{1}{n}}$ Then $\Pr[a \in [z, z+\delta]] \leq \phi^{\frac{1}{n}}\delta$

Proof

Proof: There are exactly δ outcomes of a that lead to $a \in [z, z+\delta]$, thus $\Pr[a \in [z, z+\delta]] \leq \phi^{\frac{1}{n}}\delta$.

Goal

If we could show there is roughly $O(\lg n)$ bits for each coefficient suffice to determine whether a solution exists then we have an algorithm in Smoothed-P. The algorithm goes like this with the pseudo-polynomial time algorithm $\mathcal{A}$

Algorithm 3: Algorithm for knapsack (as binary decision problem)

Input: A pseudo-polynomial time algorithm $\mathcal{A}$

$i \leftarrow 0$

while *We don't get a solution for true coefficients of* w **do**

 Run the algorithm $\mathcal{A}$ with highest $O(\lg n) + i$ many bits by rounding.

 if *We find a solution* **then**

  Then check it against the true coefficients of w

 else

  $i \leftarrow i + 1$ // take one more bit for each coefficient and continue.

We need to upper bound the running time of this while loop.

Some Lemmas to understand the runtime of Algorithm 3

- Suppose we take only first $\lg n$ bits, then it is not sufficient for an $x \in S$ to just satisfy $w^T \leq t$.

Some Lemmas to understand the runtime of Algorithm 3

- Suppose we take only first $\lg n$ bits, then it is not sufficient for an $x \in S$ to just satisfy $w^T \leq t$.
- Because of the rounding, we might find that x is feasible with respect to the rounded coefficients, whereas x is infeasible with respect to the true coefficients.

Some Lemmas to understand the runtime of Algorithm 3

- Suppose we take only first $\lg n$ bits, then it is not sufficient for an $x \in S$ to just satisfy $w^T \leq t$.
- Because of the rounding, we might find that x is feasible with respect to the rounded coefficients, whereas x is infeasible with respect to the true coefficients.
- Thus we need an x such that $w^T x$ is sufficiently smaller than t so the rounding does not affect the feasibility.

Some Lemmas to understand the runtime of Algorithm 3

- Suppose we take only first $\lg n$ bits, then it is not sufficient for an $x \in S$ to just satisfy $w^T \leq t$.
- Because of the rounding, we might find that x is feasible with respect to the rounded coefficients, whereas x is infeasible with respect to the true coefficients.
- Thus we need an x such that $w^T x$ is sufficiently smaller than t so the rounding does not affect the feasibility.
- But, we cannot rule out the existence of solutions $x \in S$ that are very close to the threshold, because there are exponentially large numbers of solutions so some might be in that neighborhood.

Some True Statement

It is possible to **prove** the following.

- Assume that there is some ranking among the solutions $x \in S$. Let the winner be the solution $x^* \in S$ that fulfills $w^T x^* \leq t$ is ranked highest among all such solutions.

Some True Statement

It is possible to **prove** the following.

- Assume that there is some ranking among the solutions $x \in S$. Let the winner be the solution $x^* \in S$ that fulfills $w^T x^* \leq t$ is ranked highest among all such solutions.
- It is likely that $t - w^T x$ is not too small.

Some True Statement

It is possible to **prove** the following.

- Assume that there is some ranking among the solutions $x \in S$. Let the winner be the solution $x^* \in S$ that fulfills $w^T x^* \leq t$ is ranked highest among all such solutions.
- It is likely that $t - w^T x^*$ is not too small.
- Now, any solution that is ranked higher than x^* must be infeasible because it violates the weight constraint.

Some True Statement

It is possible to **prove** the following.

- Assume that there is some ranking among the solutions $x \in S$. Let the winner be the solution $x^* \in S$ that fulfills $w^T x^* \leq t$ is ranked highest among all such solutions.
- It is likely that $t - w^T x^*$ is not too small.
- Now, any solution that is ranked higher than x^* must be infeasible because it violates the weight constraint.
- Suppose $\hat{x}$ be the solution that minimizes $w^T \hat{x} - t$ among all solutions ranked higher than x^*.

Some True Statement

It is possible to **prove** the following.

- Assume that there is some ranking among the solutions $x \in S$. Let the winner be the solution $x^* \in S$ that fulfills $w^T x^* \leq t$ is ranked highest among all such solutions.
- It is likely that $t - w^T x^*$ is not too small.
- Now, any solution that is ranked higher than x^* must be infeasible because it violates the weight constraint.
- Suppose $\hat{x}$ be the solution that minimizes $w^T \hat{x} - t$ among all solutions ranked higher than x^*.
- Then, it is also unlikely that $w^T \hat{x} - t$ is extremely small that is $\hat{x}$ violates the weight constraint by only a small margin.

More about the Ranking of solutions

- In analysis done in class the ranking was given by the objective function (the profitable solution).
- We do not have an objective function here because we deal with decision problems,
- Thus, we have to introduce a ranking artificially.

Strategy for Ranking of solutions

We'll use the lexicographic ordering, which satisfies the following monotonicity property,

$$\text{If } x > y \; \forall x, y \in S \text{ then } \exists\, i \text{ with } x_i = 1, y_i = 0$$

Some Basic Definitions

Definition 3.2: Winner Gap

Let x^* be the winner if it exists that is, the highest ranked solution among all feasible solutions then we define winner gap to be $\Gamma(t)$ as

$$\Gamma(t) = \begin{cases} t - w^T x^*, & \text{if there exists a feasible solution and} \\ \perp, & \text{otherwise} \end{cases}$$

Some Basic Definitions

Definition 3.3: Looser Gap

Let $\hat{x}$ be the solution (looser) that is ranked higher than x^* but cut off by the weight constraint $w^T\hat{x} \leq t$. It is the solution with minimal $w^T x - t$ among all such solutions. Now we define

$$\Lambda(t) = \begin{cases} w^T\hat{x} - t, & \text{if there exists a looser } \hat{x} \text{ and} \\ \perp, & \text{otherwise} \end{cases}$$

Winner gap is not too small

The goal is to show that the value of $\Gamma(t)$ is not too small.

Λ is enough to know Γ

Lemma 3.2: To know $\Gamma(t)$ it is enough to know $\Lambda(t)$

For all t, δ *we have* $\Pr[\Gamma(t) \leq \delta] = \Pr[\Lambda(t - \delta) \leq \delta]$

Proof

Suppose a solution $x \in S$ is *Pareto-Optimal*.

Proof

Suppose a solution $x \in S$ is *Pareto-Optimal.*
Now a little observation is that

- First, we observe that both winners and losers are Pareto-optimal,
- For every Pareto-optimal solution x there exists a threshold t such that x is the loser for this particular threshold. To see this, simply set $t = w^T - 1$

Proof - Continued

Let $P \subseteq S$ be the Pareto-optimal set. Then

$$\Gamma(t) = \min\{t - w^T x \mid x \in P, w^T x \leq t\}$$
$$\Lambda(t) = \min\{w^T x - t \mid x \in P, w^T x > t\} = \min\{w^T x - t \mid x \in P, w^T x \geq t\}$$

Proof Continued

- If $\Lambda(t) \leq \delta$ if and only if there is an $x \in P$ with $t - w^T x \in \{0, \ldots, \delta - 1\}$

Proof Continued

- If $\Lambda(t) \leq \delta$ if and only if there is an $x \in P$ with $t - w^T x \in \{0, \ldots, \delta - 1\}$
- This is equivalent to $w^T x - t \in \{-\delta + 1, \ldots, 0\}$

Proof Continued

- If $\Lambda(t) \leq \delta$ if and only if there is an $x \in P$ with $t - w^T x \in \{0, \ldots, \delta - 1\}$
- This is equivalent to $w^T x - t \in \{-\delta + 1, \ldots, 0\}$
- Which is then equivalent to $w^T x - t + \delta \in \{1, \ldots, \delta\}$

Proof Continued

- If $\Lambda(t) \leq \delta$ if and only if there is an $x \in P$ with $t - w^T x \in \{0, \ldots, \delta - 1\}$
- This is equivalent to $w^T x - t \in \{-\delta + 1, \ldots, 0\}$
- Which is then equivalent to $w^T x - t + \delta \in \{1, \ldots, \delta\}$
- This is equivalent to $w^T x - (t - \delta) \in \{1, \ldots, \delta\} \equiv \Lambda(t - \delta) \leq \delta$

Analyze $\Lambda(t)$

Given that it is equivalent to analyze $\Lambda(t)$ we state the following lemma

Lemma 3.3: Separating Lemma

For every $\delta, t \in \mathbb{N}$, we have $\Pr[\Gamma < \delta] = \Pr[\Lambda(t) \leq \delta] \leq \phi^{\frac{1}{n}} n\delta$

Final Lemma

We'll not see a proof of the previous lemma. This is same as the separating lemma done in the class. Instead We'll move on the next lemma. For that We need two definitions.

Definition 3.4: $\lfloor a \rfloor_b$

We define $\lfloor a \rfloor_b$ *be the number obtained from* a *by only taking the* b *most significant bits. This means*

$$\lfloor a \rfloor_b = 2^{n-b} \cdot \left\lfloor \frac{a}{2^{n-b}} \right\rfloor$$

Lemma

Lemma 3.4: Probability that rounded solution is the solution

Assume that we use b bits for each coefficient of w. Let x^* be the winner with respect to the true coefficient of w without rounding. The probability that solving the problem with b bits for each coefficient yields a solution different from x^* is bounded above by $2^{n-b}\phi^{\frac{1}{n}}n^2$

Proof

- We only get a different from x^* if there exists a solution $\hat{x}$ ranked higher than x^* that is feasible with the linear constraint when the coefficients are rounded. For true coefficients $\hat{x}$ crosses the linear constraint.

Proof

- We only get a different from x^* if there exists a solution $\hat{x}$ ranked higher than x^* that is feasible with the linear constraint when the coefficients are rounded. For true coefficients $\hat{x}$ crosses the linear constraint.
- When rounding we change the coefficients by at most 2^{n-b}. So $w^T\hat{x} - \lfloor w \rfloor_b^T \hat{x} \leq 2^{n-b}n$.

Proof Continued

This says that we find a $\hat{x}$ instead of x if the looser gap is $\Lambda \leq 2^{n-b}n$. Probability of that happening via the Sparating Lemma is at most $2^{n-b}n^2\phi^{\frac{1}{n}}$.

Theorem: Binary Decision Problem is $\in$ Smoothed-P

Theorem 3.2: Binary Decision Problem is $\in$ Smoothed-P

If a binary decision problem can be solved in pseudo-polynomial time, then it is in Smoothed-P.

Proof

- We must show the running time of the Algorithm 3 has polynomially decreasing tail bound.
- Recall Algorithm 3 uses pseudo-polynomial time algorithm as a subroutine. Hence the theorem will be proved.
- If b bits for each coefficient are used, the running time of the pseudo-polynomial algorithm is bounded from above by $O((n \cdot 2^n)^c)$ for some constant c.

Proof Continued

Probability that more than $t = O((n2^b)^c)$ time required is upper bounded by $2^{n-b}\phi^{1/n}n^2$ according to the last lemma.

$$
\begin{aligned}
\Pr[\text{Algorithm 3 takes more than } O((n2^b)^c) \text{ time}] &\leq 2^{n-b}\phi^{1/n}n^2 \\
&= n^2 2^{-b}(2^{n^2}\phi)^{1/n} \\
&= \frac{n^3}{O(t^{1/c})} \cdot (2^{n^2}\phi)^{1/n} \\
&\leq \frac{n^3}{O(t^{1/c})} \cdot (2^{n^2}\phi)
\end{aligned}
$$

Hence the theorem is proved that this tail bound is sufficient to have an problem in Smoothed-P.

Other things not discussed here

- Some of the other Smoothed-P problems
  - Graph Coloring and Smoothed Extension of $G_{n,p}$.
  - K Coloring is in Smoothed-P for any $k \in \mathbb{N}$.
- Dist-NP$_{\text{para}}$ the NP equivalent class in Smoothed Complexity Theory.

Smoothed Extension of $G_{n,p}$

Model of $G_{n,p}$ in smoothed extension:

Given an adversarial graph $G = (V, E)$ and an $\epsilon \in \left(0, \frac{1}{2}\right]$, we obtain a new graph $G' = (V', E')$ on the same set of vertices by flipping each non-edge of G independently with probability ϵ.

This means that if $e = (u, v) \in G.E$ then $\Pr[e \in G'.E] \leq 1 - \epsilon$

$G_{n,p}$ model in our framework

- We represent a graph G on n vertices as a binary string of length $\binom{n}{2}$.
- Now we have $N_{n,G} = 2^{\binom{n}{2}}$.
- We choose $\epsilon \leq \frac{1}{2}$ such that $(1-\epsilon)^{\binom{n}{2}} = \phi$

Theorem

We'll not show proof and any other problems in Smoothed-P

Theorem 3.3: k-Coloring in Smoothed-P

k-Coloring is a known NP-Complete Problem for $k > 3$. For any $k \in \mathbb{N}$ k-Coloring $\in$ Smoothed-P

Proof idea

- A graph is k-colorable if and only if it does not contain a clique of size $k+1$.
- The Probability of a specific set of $k+1$ vertices form a $k+1$ clique is at least $\epsilon^{\binom{k+1}{2}}$
- Then the graph G does not contain a $k+1$ clique is at most $(1-\epsilon^{\binom{k+1}{2}})^{\frac{n}{k+1}}$
- For $\epsilon \geq 0.1$ we can show that expected running time to the power $\lg_k(\frac{1}{c})$ is bounded from above by some polynomial.
- For $\epsilon < 0.1$ we can do a brute force search in every run.