New Arenas in Hardness Amplification

Karthik C. S.
(Weizmann Institute of Science)

Joint work with

Elazar Goldenberg
(The Academic College of Tel Aviv-Yaffo)

Necessity is the Mother of Invention

Necessity is the Mother of Invention

- Average Case Complexity

Necessity is the Mother of Invention

- Average Case Complexity
  - Hardness of Problems in Practice

Necessity is the Mother of Invention

- Average Case Complexity
  - Hardness of Problems in Practice
- Modern Cryptography

Necessity is the Mother of Invention

- Average Case Complexity
  - Hardness of Problems in Practice
- Modern Cryptography
  - Hard on average function in NP

Necessity is the Mother of Invention

- Average Case Complexity
  - Hardness of Problems in Practice
- Modern Cryptography
  - Hard on average function in NP

Modest Goal:

Necessity is the Mother of Invention

- Average Case Complexity
  - Hardness of Problems in Practice
- Modern Cryptography
  - Hard on average function in NP

Modest Goal: Hardness Amplification

Necessity is the Mother of Invention

- Average Case Complexity
  - Hardness of Problems in Practice
- Modern Cryptography
  - Hard on average function in NP

Modest Goal: Hardness Amplification

Mild average case $\Rightarrow$ Sharp average case

The Utopic Theorem of Hardness Amplification

- Family of functions $\{f_n\}_{n \in \mathbb{N}}$

The Utopic Theorem of Hardness Amplification

- Family of functions $\{f_n\}_{n\in\mathbb{N}}$
- Every algorithm $\mathscr{A}$ running in time $t(n)$,
  fails on $p(n)$ fraction of inputs

The Utopic Theorem of Hardness Amplification

- Family of functions $\{f_n\}_{n\in\mathbb{N}}$
- Every algorithm $\mathscr{A}$ running in time $t(n)$, fails on $p(n)$ fraction of inputs

$$\Downarrow$$

The Utopic Theorem of Hardness Amplification

- Family of functions $\{f_n\}_{n\in\mathbb{N}}$
- Every algorithm $\mathscr{A}$ running in time $t(n)$,
  fails on $p(n)$ fraction of inputs

$$\Downarrow$$

- Family of functions $\{g_n\}_{n\in\mathbb{N}}$

The Utopic Theorem of Hardness Amplification

- Family of functions $\{f_n\}_{n\in\mathbb{N}}$
- Every algorithm $\mathscr{A}$ running in time $t(n)$,
  fails on $p(n)$ fraction of inputs

$$\Downarrow$$

- Family of functions $\{g_n\}_{n\in\mathbb{N}}$
- Every algorithm $\mathscr{A}'$ running in time $t'(n)$,
  fails on $p'(n)$ fraction of inputs

The Utopic Theorem of Hardness Amplification

- Family of functions $\{f_n\}_{n\in\mathbb{N}}$
- Every algorithm $\mathcal{A}$ running in time $t(n)$, fails on $p(n)$ fraction of inputs

$$\Downarrow$$

- Family of functions $\{g_n\}_{n\in\mathbb{N}}$
- Every algorithm $\mathcal{A}'$ running in time $t'(n)$, fails on $p'(n)$ fraction of inputs

- $p'(n) \gg p(n)$

The Utopic Theorem of Hardness Amplification

- Family of functions $\{f_n\}_{n \in \mathbb{N}}$
- Every algorithm $\mathcal{A}$ running in time $t(n)$, fails on $p(n)$ fraction of inputs

$$\Downarrow$$

- Family of functions $\{g_n\}_{n \in \mathbb{N}}$
- Every algorithm $\mathcal{A}'$ running in time $t'(n)$, fails on $p'(n)$ fraction of inputs

- $p'(n) \gg p(n)$
- $f_n = g_n$

The Utopic Theorem of Hardness Amplification

- Family of functions $\{f_n\}_{n \in \mathbb{N}}$
- Every algorithm $\mathscr{A}$ running in time $t(n)$, fails on $p(n)$ fraction of inputs

$$\Downarrow$$

- Family of functions $\{g_n\}_{n \in \mathbb{N}}$
- Every algorithm $\mathscr{A}'$ running in time $t'(n)$, fails on $p'(n)$ fraction of inputs

- $p'(n) \gg p(n)$
- $f_n = g_n$
- f is "interesting"

The Big Question

Can we do hardness amplification

The Big Question

Can we do hardness amplification
for problems

The Big Question

Can we do hardness amplification
for problems
we care about and

The Big Question

Can we do hardness amplification
for problems
we care about and
we believe are hard on average?

The Story so far

- #P (Lipton'89):

The Story so far

- #P (Lipton'89): If `Permanent` can be computed:
  - deterministically in polynomial time
  - on 1/2 the matrices

The Story so far

- #P (Lipton'89): If `Permanent` can be computed:
  - deterministically in polynomial time
  - on 1/2 the matrices

  then `Permanent` is in BPP.

The Story so far

- #P (Lipton'89): If `Permanent` can be computed:
  - deterministically in polynomial time
  - on $1/2$ the matrices

  then `Permanent` is in BPP.

- EXP (Trevisan-Vadhan'07):

The Story so far

- #P (Lipton'89): If `Permanent` can be computed:
  - deterministically in polynomial time
  - on 1/2 the matrices

  then `Permanent` is in BPP.

- EXP (Trevisan-Vadhan'07):If $\exists \Pi \in$EXP:
  - cannot be efficiently solved in the worst case by
  - uniform probabilistic algorithms

The Story so far

- #P (Lipton'89): If `Permanent` can be computed:
  - deterministically in polynomial time
  - on 1/2 the matrices

  then `Permanent` is in BPP.

- EXP (Trevisan-Vadhan'07):If $\exists \Pi \in$EXP:
  - cannot be efficiently solved in the worst case by
  - uniform probabilistic algorithms

  then $\exists \Lambda \in$EXP:
  - cannot be efficiently solved on random instances
  - noticeably better than guessing the answer at random.

What about NP?

- Non-uniform case (Healy-Vadhan-Viola'04):

What about NP?

- Non-uniform case (Healy-Vadhan-Viola'04):If $\exists f$ in NP
  - circuits of size $s(n)$ fails to compute f
  - on $1/\text{poly}(n)$ fraction of inputs,

What about NP?

- Non-uniform case (Healy-Vadhan-Viola'04):If $\exists f$ in NP
  - circuits of size $s(n)$ fails to compute f
  - on $1/\text{poly}(n)$ fraction of inputs,

then $\exists f'$ in NP

  - circuits of size $s'(n) = s(\sqrt{n})^{\Omega(1)}$ fails to compute f'
  - on $1/2 - 1/s'(n)$ fraction of inputs.

What about NP?

- Non-uniform case (Healy-Vadhan-Viola'04):If $\exists f$ in NP
  - circuits of size $s(n)$ fails to compute f
  - on $1/\mathrm{poly}(n)$ fraction of inputs,

  then $\exists f'$ in NP
  - circuits of size $s'(n) = s(\sqrt{n})^{\Omega(1)}$ fails to compute f'
  - on $1/2 - 1/s'(n)$ fraction of inputs.

- Uniform case (Trevisan'05):

What about NP?

- Non-uniform case (Healy-Vadhan-Viola'04):If $\exists f$ in NP
  - circuits of size $s(n)$ fails to compute f
  - on $1/\text{poly}(n)$ fraction of inputs,

  then $\exists f'$ in NP
  - circuits of size $s'(n) = s(\sqrt{n})^{\Omega(1)}$ fails to compute f'
  - on $1/2 - 1/s'(n)$ fraction of inputs.

- Uniform case (Trevisan'05):If every problem in NP
  - admits an efficient uniform algorithm
  - succeeds with probability at least $1/2 + 1/(\log n)^{O(1)}$

What about NP?

- Non-uniform case (Healy-Vadhan-Viola'04):If $\exists f$ in NP
  - circuits of size $s(n)$ fails to compute f
  - on $1/\text{poly}(n)$ fraction of inputs,

  then $\exists f'$ in NP
  - circuits of size $s'(n) = s(\sqrt{n})^{\Omega(1)}$ fails to compute f'
  - on $1/2 - 1/s'(n)$ fraction of inputs.

- Uniform case (Trevisan'05):If every problem in NP
  - admits an efficient uniform algorithm
  - succeeds with probability at least $1/2 + 1/(\log n)^{O(1)}$

  then for every problem in NP
  - there is an efficient uniform algorithm
  - succeeds with probability at least $1 - 1/\text{poly}(n)$

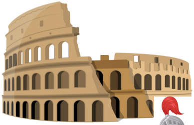
The Verona, Pompeii, Flavian, and Fiesole arenas
may not be as well known as the Colosseum,
but are just as impressive.

– *Roman history trivia*

Arenas in Hardness Amplification

NP

Arenas in Hardness Amplification

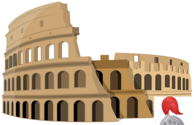


#P

EXP

NP

Arenas in Hardness Amplification



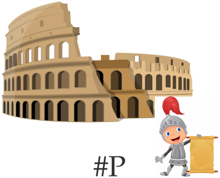
#P

EXP

NP

P

Arenas in Hardness Amplification

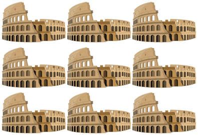


#P

EXP

NP

P



Optimization Problems

Focus of this Talk

Optimization Problems

Focus of this Talk

Optimization Problems

- NP-hard problems

Focus of this Talk

Optimization Problems

- NP-hard problems
- Subquadratic-hard problems

Focus of this Talk

Optimization Problems

- ◎ NP-hard problems
- ◎ Subquadratic-hard problems
- ◎ Total Problems

Maximum Clique

Maximum Clique

Input: A graph G

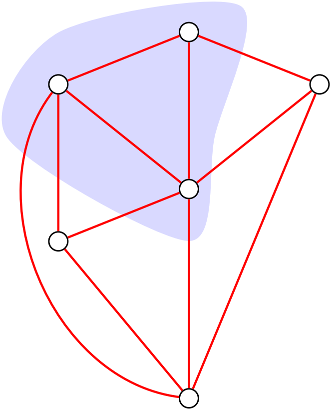
Maximum Clique

Input: A graph G

Output: Clique of maximum size in G

Input: A graph G

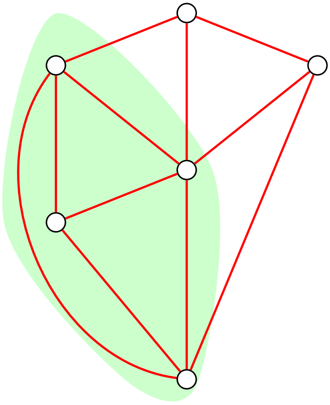
Output: Clique of maximum size in G



Maximum Clique

Input: A graph G

Output: Clique of maximum size in G



Our Result for Maximum Clique

Theorem (Goldenberg-K'19)

Let $\mathscr{D}$ be poly(n) time samplable distribution over graphs on n vertices

Our Result for Maximum Clique

Theorem (Goldenberg-K'19)

Let $\mathscr{D}$ be poly(n) time samplable distribution over graphs on n vertices such that for every randomized algorithm $\mathscr{A}$ running in time poly(n),

Our Result for Maximum Clique

Theorem (Goldenberg-K'19)

Let $\mathscr{D}$ be poly(n) time samplable distribution over graphs on n vertices such that for every randomized algorithm $\mathscr{A}$ running in time poly(n), we have:

$$\Pr_{G \sim \mathscr{D}} \left[\mathscr{A} \text{ finds max-clique in } G \text{ w.p. } \geq 2/3 \right] \leq 1 - 1/n.$$

Our Result for Maximum Clique

Theorem (Goldenberg-K'19)

Let $\mathscr{D}$ be poly(n) time samplable distribution over graphs on n vertices such that for every randomized algorithm $\mathscr{A}$ running in time poly(n), we have:

$$\Pr_{G\sim\mathscr{D}}\left[\mathscr{A} \text{ finds max-clique in } G \text{ w.p. } \geq 2/3\right] \leq 1 - 1/n.$$

Then there is $\mathscr{D}'$ a poly(n) time samplable distribution over graphs on poly(n) vertices

Our Result for Maximum Clique

Theorem (Goldenberg-K'19)

Let $\mathscr{D}$ be poly(n) time samplable distribution over graphs on n vertices such that for every randomized algorithm $\mathscr{A}$ running in time poly(n), we have:

$$\Pr_{G \sim \mathscr{D}} \left[\mathscr{A} \text{ finds max-clique in } G \text{ w.p.} \geq 2/3 \right] \leq 1 - 1/n.$$

Then there is $\mathscr{D}'$ a poly(n) time samplable distribution over graphs on poly(n) vertices such that for every randomized algorithm $\mathscr{A}'$ running in time poly(n),

Our Result for Maximum Clique

Theorem (Goldenberg-K'19)

Let $\mathscr{D}$ be poly(n) time samplable distribution over graphs on n vertices such that for every randomized algorithm $\mathscr{A}$ running in time poly(n), we have:

$$\Pr_{G \sim \mathscr{D}} \left[\mathscr{A} \text{ finds max-clique in } G \text{ w.p. } \geq 2/3 \right] \leq 1 - 1/n.$$

Then there is $\mathscr{D}'$ a poly(n) time samplable distribution over graphs on poly(n) vertices such that for every randomized algorithm $\mathscr{A}'$ running in time poly(n), we have:

$$\Pr_{G' \sim \mathscr{D}'} \left[\mathscr{A}' \text{ finds max-clique in } G' \text{ w.p. } \geq 2/3 \right] \leq 0.01.$$

Proof Overview

Proof Overview

1. Define new distribution $\mathscr{D}'$

Proof Overview

1. Define new distribution $\mathscr{D}'$
2. Given $\mathscr{A}'$ for $\mathscr{D}'$ design $\mathscr{A}$ for $\mathscr{D}$

Proof Overview

1. Define new distribution $\mathscr{D}'$
2. Given $\mathscr{A}'$ for $\mathscr{D}'$ design $\mathscr{A}$ for $\mathscr{D}$
3. Argue that if $\mathscr{A}'$ is correct on 0.01 fraction of inputs
   then $\mathscr{A}$ is correct on $1 - 1/n$ fraction of inputs

Construction of New Distribution

$\mathscr{D}'$ samples a graph H as follows:

Construction of New Distribution

$\mathscr{D}'$ samples a graph H as follows:

1. Independently sample $G_1, \ldots, G_k$ from $\mathscr{D}$ ($k :=$poly(n))

Construction of New Distribution

$\mathscr{D}'$ samples a graph H as follows:

1. Independently sample $G_1, \ldots, G_k$ from $\mathscr{D}$ ($k :=$poly(n))
2. Define $H := G_1 \dot\cup \cdots \dot\cup G_k$

Construction of New Distribution

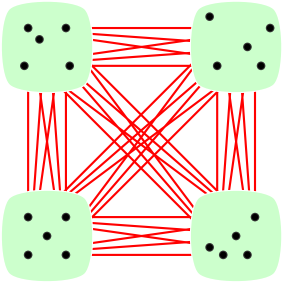
$\mathcal{D}'$ samples a graph H as follows:

1. Independently sample $G_1, \ldots, G_k$ from $\mathcal{D}$ ($k :=$poly(n))
2. Define $H := G_1 \dot\cup \cdots \dot\cup G_k$
3. For every $i \neq j$ insert every edge between G_i and G_j

Construction of New Distribution

$\mathscr{D}'$ samples a graph H as follows:

1. Independently sample $G_1, \ldots, G_k$ from $\mathscr{D}$ ($k :=$poly(n))
2. Define $H := G_1 \dot\cup \cdots \dot\cup G_k$
3. For every $i \neq j$ insert every edge between G_i and G_j
4. Output H



Construction of New Distribution

$\mathscr{D}'$ samples a graph H as follows:

1. Independently sample $G_1, \ldots, G_k$ from $\mathscr{D}$ ($k :=$poly(n))
2. Define $H := G_1 \dot\cup \cdots \dot\cup G_k$
3. For every $i \neq j$ insert every edge between G_i and G_j
4. Output H

Sampling time: poly(n)

Algorithm for Original Distribution

Algorithm for Original Distribution

Algorithm $\mathscr{A}$

Input: A graph G sampled from $\mathscr{D}$

Output: A maximum clique in G

Algorithm $\mathcal{A}$

Input: A graph G sampled from $\mathcal{D}$

Output: A maximum clique in G

1. Set `Solution` to be empty.

Algorithm for Original Distribution

Algorithm $\mathscr{A}$

Input: A graph G sampled from $\mathscr{D}$

Output: A maximum clique in G

1. Set `Solution` to be empty.
2. Repeat following $O(1)$ times.

Algorithm for Original Distribution

Algorithm $\mathscr{A}$

Input: A graph G sampled from $\mathscr{D}$

Output: A maximum clique in G

1. Set `Solution` to be empty.
2. Repeat following $O(1)$ times.
   2.1 Pick randomly $i \in [k]$

Algorithm for Original Distribution

Algorithm $\mathscr{A}$

Input: A graph G sampled from $\mathscr{D}$

Output: A maximum clique in G

1. Set `Solution` to be empty.
2. Repeat following $O(1)$ times.
   2.1 Pick randomly $i \in [k]$
   2.2 Independently sample $G_1, \ldots, G_{i-1}, G_{i+1}, \ldots G_k$ from $\mathscr{D}$

Algorithm $\mathscr{A}$

Input: A graph G sampled from $\mathscr{D}$

Output: A maximum clique in G

1. Set `Solution` to be empty.
2. Repeat following $O(1)$ times.
   - 2.1 Pick randomly $i \in [k]$
   - 2.2 Independently sample $G_1, \ldots, G_{i-1}, G_{i+1}, \ldots G_k$ from $\mathscr{D}$
   - 2.3 Construct H setting G_i to be G

Algorithm for Original Distribution

Algorithm $\mathscr{A}$

Input: A graph G sampled from $\mathscr{D}$

Output: A maximum clique in G

1. Set `Solution` to be empty.
2. Repeat following $O(1)$ times.
   2.1 Pick randomly $i \in [k]$
   2.2 Independently sample $G_1, \ldots, G_{i-1}, G_{i+1}, \ldots G_k$ from $\mathscr{D}$
   2.3 Construct H setting G_i to be G
   2.4 Find clique in H using $\mathscr{A}'$

Algorithm for Original Distribution

Algorithm $\mathscr{A}$

Input: A graph G sampled from $\mathscr{D}$

Output: A maximum clique in G

1. Set `Solution` to be empty.
2. Repeat following $O(1)$ times.
   2.1 Pick randomly $i \in [k]$
   2.2 Independently sample $G_1, \ldots, G_{i-1}, G_{i+1}, \ldots G_k$ from $\mathscr{D}$
   2.3 Construct H setting G_i to be G
   2.4 Find clique in H using $\mathscr{A}'$
   2.5 Restrict clique in H to G and add to `Solution`

Algorithm $\mathscr{A}$

Input: A graph G sampled from $\mathscr{D}$

Output: A maximum clique in G

1. Set `Solution` to be empty.
2. Repeat following $O(1)$ times.
   2.1 Pick randomly $i \in [k]$
   2.2 Independently sample $G_1, \ldots, G_{i-1}, G_{i+1}, \ldots G_k$ from $\mathscr{D}$
   2.3 Construct H setting G_i to be G
   2.4 Find clique in H using $\mathscr{A}'$
   2.5 Restrict clique in H to G and add to `Solution`
3. Output the largest clique in `Solution`

Structure of Optimal Solutions

Claim

If S is a maximum clique of H then for any $i \in [k]$ its restriction to vertices of G_i gives a maximum clique of G_i.

Correctness of Algorithm

- $\mathscr{A}_0$ be one iteration of Step 2 of $\mathscr{A}$

Correctness of Algorithm

- $\mathscr{A}_0$ be one iteration of Step 2 of $\mathscr{A}$
- If $\mathscr{A}_0$ outputs maximum clique w.p. ε on $1 - 1/n$ fraction of samples from $\mathscr{D}$ then,

Correctness of Algorithm

- $\mathcal{A}_0$ be one iteration of Step 2 of $\mathcal{A}$
- If $\mathcal{A}_0$ outputs maximum clique w.p. ε on $1 - 1/n$ fraction of samples from $\mathcal{D}$ then,

  $\mathcal{A}$ outputs maximum clique w.p. $2/3$ on $1 - 1/n$ fraction of samples from $\mathcal{D}$.

Correctness of Algorithm

- $\mathscr{A}_0$ be one iteration of Step 2 of $\mathscr{A}$
- If $\mathscr{A}_0$ outputs maximum clique w.p. ε on $1 - 1/n$ fraction of samples from $\mathscr{D}$ then,

  $\mathscr{A}$ outputs maximum clique w.p. $2/3$ on $1 - 1/n$ fraction of samples from $\mathscr{D}$.
- Suffices to show: $\mathscr{A}'$ outputs maximum clique in Step 2.5 w.p. ε on $1 - 1/n$ fraction of samples from $\mathscr{D}$.

A Direct Product Lemma

A Direct Product Lemma

Lemma (Feige-Kilian'94)

Let $\mathcal{T}$ be a distribution over X. Let $f : X^k \to \{0,1\}$.

A Direct Product Lemma

Lemma (Feige-Kilian'94)

Let $\mathcal{T}$ be a distribution over X. Let $f : X^k \to \{0,1\}$. Then,

$$\Pr_{\substack{x\sim\mathcal{T}\\ i\in[k]}}\left[|\mu_{i,x} - \mu| \geq k^{-1/3}\right] \leq k^{-1/3},$$

A Direct Product Lemma

Lemma (Feige-Kilian'94)

Let $\mathcal{T}$ be a distribution over X. Let $f : X^k \to \{0,1\}$. Then,

$$\Pr_{\substack{x \sim \mathcal{T} \\ i \in [k]}} \left[|\mu_{i,x} - \mu| \geq k^{-1/3} \right] \leq k^{-1/3},$$

where

$$\mu = \mathop{\mathbb{E}}_{x^k \sim \mathcal{T}^k} \left[f\left(x^k\right) \right],$$

A Direct Product Lemma

Lemma (Feige-Kilian'94)

Let $\mathcal{T}$ be a distribution over X. Let $f : X^k \to \{0,1\}$. Then,

$$\Pr_{\substack{x \sim \mathcal{T} \\ i \in [k]}} \left[|\mu_{i,x} - \mu| \geq k^{-1/3} \right] \leq k^{-1/3},$$

where

$$\mu = \mathop{\mathbb{E}}_{x^k \sim \mathcal{T}^k} \left[f\left(x^k\right) \right],$$

$$\mu_{i,x} = \mathop{\mathbb{E}}_{x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_k \sim \mathcal{T}} \left[f(x_1, \ldots, x_{i-1}, x, x_{i+1}, \ldots x_k) \right].$$

A Direct Product Lemma

Lemma (Feige-Kilian'94)

Let $\mathcal{T}$ be a distribution over X. Let $f : X^k \to \{0,1\}$. Then,

$$\Pr_{\substack{x\sim\mathcal{T}\\ i\in[k]}}\left[\left|\mu_{i,x} - \mu\right| \geq k^{-1/3}\right] \leq k^{-1/3},$$

where

$$\mu = \mathop{\mathbb{E}}_{x^k\sim\mathcal{T}^k}\left[f\left(x^k\right)\right],$$

$$\mu_{i,x} = \mathop{\mathbb{E}}_{x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_k\sim\mathcal{T}}\left[f(x_1,\ldots,x_{i-1},x,x_{i+1},\ldots x_k)\right].$$

$f(x^k) = 1 \iff \mathcal{A}'$ outputs maximum clique w.p. $2/3$

Proof Summary

Proof Summary

- New Distribution: Direct Product of Old Distribution with solution preserving property

Proof Summary

- New Distribution: Direct Product of Old Distribution with solution preserving property
- Invoke Feige-Kilian lemma to show amplification of hardness

Hardness Amplification for Optimization Problems

Hardness Amplification for Optimization Problems

An optimization problem Π is the quadruple $(\mathsf{I}_\Pi, \mathsf{Sol}_\Pi, \Delta_\Pi, \mathsf{goal}_\Pi)$:

Hardness Amplification for Optimization Problems

An optimization problem Π is the quadruple $(\mathsf{I}_\Pi, \mathsf{Sol}_\Pi, \Delta_\Pi, \mathsf{goal}_\Pi)$:

- I_Π: set of instances of Π;

Hardness Amplification for Optimization Problems

An optimization problem Π is the quadruple $(\mathsf{I}_\Pi, \mathsf{Sol}_\Pi, \Delta_\Pi, \mathsf{goal}_\Pi)$:

- I_Π: set of instances of Π;
- Sol_Π: function from I_Π to set of feasible solutions;

An optimization problem Π is the quadruple $(\mathsf{I}_\Pi, \mathsf{Sol}_\Pi, \Delta_\Pi, \mathsf{goal}_\Pi)$:

- I_Π: set of instances of Π;
- Sol_Π: function from I_Π to set of feasible solutions;
- Δ_Π: assigns $(x \in \mathsf{I}_\Pi,\ y \in \mathsf{Sol}_\Pi(x))$ a non-negative integer;

Hardness Amplification for Optimization Problems

An optimization problem Π is the quadruple $(\mathsf{I}_\Pi, \mathsf{Sol}_\Pi, \Delta_\Pi, \mathsf{goal}_\Pi)$:

- I_Π: set of instances of Π;
- Sol_Π: function from I_Π to set of feasible solutions;
- Δ_Π: assigns $(x \in \mathsf{I}_\Pi,\ y \in \mathsf{Sol}_\Pi(x))$ a non-negative integer;
- $\mathsf{goal}_\Pi \in \{\min, \max\}$.

Direct Product Feasibility

Let $S, T : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$.

Direct Product Feasibility

Let $S, T : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$.

We say $\Pi(\mathsf{I}_\Pi, \mathsf{Sol}_\Pi, \Delta_\Pi, \mathsf{goal}_\Pi)$ is (S, T)-direct product feasible

Direct Product Feasibility

Let $S, T : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$.

We say $\Pi(\mathsf{I}_\Pi, \mathsf{Sol}_\Pi, \Delta_\Pi, \mathsf{goal}_\Pi)$ is (S, T)-direct product feasible

if there exists deterministic $(\mathsf{Gen}, \mathsf{Dec})$:

Direct Product Feasibility

Let $S, T : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$.

We say $\Pi(\mathsf{I}_\Pi, \mathsf{Sol}_\Pi, \Delta_\Pi, \mathsf{goal}_\Pi)$ is (S, T)-direct product feasible

if there exists deterministic $(\mathsf{Gen}, \mathsf{Dec})$:

- Gen:
  - Input: $x_1, \ldots, x_k \in \mathsf{I}_\Pi(n)$
  - Output: $x' \in \mathsf{I}_\Pi(S(n, k))$

Direct Product Feasibility

Let $S, T : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$.

We say $\Pi(\mathsf{I}_\Pi, \mathsf{Sol}_\Pi, \Delta_\Pi, \mathsf{goal}_\Pi)$ is (S, T)-direct product feasible

if there exists deterministic $(\mathsf{Gen}, \mathsf{Dec})$:

- Gen:
  - Input: $x_1, \ldots, x_k \in \mathsf{I}_\Pi(n)$
  - Output: $x' \in \mathsf{I}_\Pi(S(n, k))$
- Dec:
  - Input: $i \in [k], x_1, \ldots, x_k \in \mathsf{I}_\Pi(n)$, and optimal $y' \in \mathsf{Sol}_\Pi(x')$
  - Output: optimal $y \in \mathsf{Sol}_\Pi(x_i)$

Direct Product Feasibility

Let $S, T : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$.

We say $\Pi(\mathsf{I}_\Pi, \mathsf{Sol}_\Pi, \Delta_\Pi, \mathsf{goal}_\Pi)$ is (S, T)-direct product feasible

if there exists deterministic $(\mathsf{Gen}, \mathsf{Dec})$:

- Gen:
    - Input: $x_1, \ldots, x_k \in \mathsf{I}_\Pi(n)$
    - Output: $x' \in \mathsf{I}_\Pi(S(n,k))$
- Dec:
    - Input: $i \in [k]$, $x_1, \ldots, x_k \in \mathsf{I}_\Pi(n)$, and optimal $y' \in \mathsf{Sol}_\Pi(x')$
    - Output: optimal $y \in \mathsf{Sol}_\Pi(x_i)$
- Gen and Dec run in $T(n,k)$ time.

Our General Result

Theorem (Goldenberg-K'19)

Let Π be (S, T)-direct product feasible. Let D be $s(n)$ time samplable distribution over $\mathsf{I}_{\Pi}(n)$ such that for every randomized algorithm $\mathscr{A}$ running in time $t(n)$, we have:

$$\Pr_{x \sim D}\left[\mathscr{A} \text{ finds optimal solution of } x \text{ w.p.} \geq 2/3\right] \leq 1 - \frac{1}{p(n)}.$$

*Conditions apply.

Our General Result

Theorem (Goldenberg-K'19)

Let Π be (S, T)-direct product feasible. Let D be $s(n)$ time samplable distribution over $\mathsf{I}_\Pi(n)$ such that for every randomized algorithm $\mathscr{A}$ running in time $t(n)$, we have:

$$\Pr_{x \sim D}\left[\mathscr{A} \text{ finds optimal solution of } x \text{ w.p. } \geq 2/3\right] \leq 1 - \frac{1}{p(n)}.$$

Then for $k = \text{poly}(p(n))$ there is D' a $\tilde{O}(k \cdot s(n) + T(n, k))$ time samplable distribution over $\mathsf{I}_\Pi(S(n, k))$ such that for every randomized algorithm $\mathscr{A}'$ running in time* $\tilde{O}(t(n))$, we have:

$$\Pr_{x' \sim D'}\left[\mathscr{A}' \text{ finds optimal solution of } x' \text{ w.p. } \geq 2/3\right] \leq 0.01.$$

*Conditions apply.

Problems in P

Theorem (Goldenberg-K'19)

Let D be $\tilde{O}(n)$ time samplable distribution over LCS/Edit Distance such that for every randomized algorithm $\mathscr{A}$ running in time $n^{2-\varepsilon}$, we have:

$$\Pr_{x \sim D}\left[\mathscr{A} \text{ finds optimal alignment of } x \text{ w.p.} \geq 2/3\right] \leq 1 - {}^{1}/{}_{n^{o(1)}}.$$

Problems in P

Theorem (Goldenberg-K'19)

Let D be $\tilde{O}(n)$ time samplable distribution over LCS/Edit Distance such that for every randomized algorithm $\mathscr{A}$ running in time $n^{2-\varepsilon}$, we have:

$$\Pr_{x\sim D}\left[\mathscr{A} \text{ finds optimal alignment of } x \text{ w.p.} \geq 2/3\right] \leq 1 - 1/n^{o(1)}.$$

Then there is D' a $\tilde{O}(n)$ time samplable distribution over LCS/Edit Distance such that for every randomized algorithm $\mathscr{A}'$ running in time $n^{2-2\varepsilon}$, we have:

$$\Pr_{x'\sim D'}\left[\mathscr{A}' \text{ finds optimal alignment of } x' \text{ w.p.} \geq 2/3\right] \leq 0.01.$$

Theorem (Goldenberg-K'19)

Let D be $\tilde{O}(n)$ time samplable distribution over LCS/Edit Distance such that for every randomized algorithm $\mathscr{A}$ running in time $n^{2-\varepsilon}$, we have:

$$\Pr_{x\sim D}\left[\mathscr{A} \text{ finds optimal alignment of } x \text{ w.p.} \geq 2/3\right] \leq 1 - {}^{1}/_{n^{o(1)}}.$$

Then there is D' a $\tilde{O}(n)$ time samplable distribution over LCS/Edit Distance such that for every randomized algorithm $\mathscr{A}'$ running in time $n^{2-2\varepsilon}$, we have:

$$\Pr_{x'\sim D'}\left[\mathscr{A}' \text{ finds optimal alignment of } x' \text{ w.p.} \geq 2/3\right] \leq 0.01.$$

What about Fréchet Distance?

Theorem (Goldenberg-K'19)

Let D be $\tilde{O}(n)$ time samplable distribution over LCS/Edit Distance such that for every randomized algorithm $\mathscr{A}$ running in time $n^{2-\varepsilon}$, we have:

$$\Pr_{x \sim D}\left[\mathscr{A} \text{ finds optimal alignment of } x \text{ w.p.} \geq 2/3\right] \leq 1 - 1/n^{o(1)}.$$

Hardness Amplification for

Matrix Multiplication

Then there ... LCS/Edit Distance su... unning in time $n^{2-2\varepsilon}$, we have:

$$\Pr_{x' \sim D'}\left[\mathscr{A}' \text{ finds optimal alignment of } x' \text{ w.p.} \geq 2/3\right] \leq 0.01.$$

What about Fréchet Distance?

Connection to Max-SAT

Theorem (Goldenberg-K'19)

Let D be $\mathrm{poly}(n)$ time samplable distribution over Max-SAT such that for every randomized algorithm $\mathscr{A}$ running in time $2^{o(n)}$, we have:

$$\Pr_{x \sim D}\left[\mathscr{A} \text{ finds optimal assignment of } x \text{ w.p. } \geq 2/3\right] \leq 1 - 1/2^{n^{1-o(1)}}.$$

Connection to Max-SAT

Theorem (Goldenberg-K'19)

Let D be $\mathrm{poly}(n)$ time samplable distribution over Max-SAT such that for every randomized algorithm $\mathscr{A}$ running in time $2^{o(n)}$, we have:

$$\Pr_{x \sim D}\left[\mathscr{A} \text{ finds optimal assignment of } x \text{ w.p.} \geq 2/3\right] \leq 1 - {}^{1}\!/\!{}_{2^{n^{1-o(1)}}}.$$

Then there is D' a $\mathrm{poly}(n)$ time samplable distribution over Max-SAT such that for every randomized algorithm $\mathscr{A}'$ running in time $n^{\omega(1)}$, we have:

$$\Pr_{x' \sim D'}\left[\mathscr{A}' \text{ finds optimal assignment of } x' \text{ w.p.} \geq 2/3\right] \leq 0.01.$$

Connection to Max-SAT

Theorem (Goldenberg-K'19)

Let D be p … Max-SAT such that … g in time $2^{o(n)}$, we ha …

Can be extended to Vertex Cover, Dominating Set, etc

$$\Pr_{x \sim D}\left[\mathscr{A} \text{ finds optimal assignment of } x \text{ w.p.} \geq 2/3\right] \leq 1 - {}^{1}/_{2^{n^{1-o(1)}}}.$$

Then there is D' a poly(n) time samplable distribution over Max-SAT such that for every randomized algorithm $\mathscr{A}'$ running in time $n^{\omega(1)}$, we have:

$$\Pr_{x' \sim D'}\left[\mathscr{A}' \text{ finds optimal assignment of } x' \text{ w.p.} \geq 2/3\right] \leq 0.01.$$

Connection to Max-SAT

Theorem (Goldenberg-K'19)

Let D be p Max-SAT such that g in time $2^{o(n)}$, we ha

Can be extended to Vertex Cover,
Dominating Set, etc

$$\Pr_{x \sim D}\left[\mathscr{A} \text{ finds optimal assignment of } x \text{ w.p.} \geq 2/3\right] \leq 1 - 1/2^{n^{1-o(1)}}.$$

Then there tion over Max-SAT $\mathscr{A}'$ running in

Can even be extended to Knapsack,
and other maximization problems!

$$\Pr_{x' \sim D'}\left[\mathscr{A}' \text{ finds optimal assignment of } x' \text{ w.p.} \geq 2/3\right] \leq 0.01.$$

Connection to TFNP

Factoring

Connection to TFNP

Factoring

- Given $N \in [2^n]$ find all its prime factors

Connection to TFNP

Factoring

- Given $N \in [2^n]$ find all its prime factors
- Gen multiplies input integers

Connection to TFNP

Factoring

- Given $N \in [2^n]$ find all its prime factors
- Gen multiplies input integers
- Dec checks if candidate prime divides input integer

Connection to TFNP

Factoring

- Given $N \in [2^n]$ find all its prime factors
- Gen multiplies input integers
- Dec checks if candidate prime divides input integer

End of Line Problem

- Given $P, S : \{0,1\}^n \to \{0,1\}^n$ such that $P(0^n) = 0^n \neq S(0^n)$

Connection to TFNP

Factoring

- Given $N \in [2^n]$ find all its prime factors
- Gen multiplies input integers
- Dec checks if candidate prime divides input integer

End of Line Problem

- Given $P, S : \{0,1\}^n \to \{0,1\}^n$ such that $P(0^n) = 0^n \neq S(0^n)$
- Find x such that $P(S(x)) \neq x$ or $S(P(x)) = x \neq 0^n$

Connection to TFNP

Factoring

- Given $N \in [2^n]$ find all its prime factors
- Gen multiplies input integers
- Dec checks if candidate prime divides input integer

End of Line Problem

- Given $P, S : \{0,1\}^n \to \{0,1\}^n$ such that $P(0^n) = 0^n \neq S(0^n)$
- Find x such that $P(S(x)) \neq x$ or $S(P(x)) = x \neq 0^n$
- Gen concatenates input and output gates

Connection to TFNP

Factoring

- Given $N \in [2^n]$ find all its prime factors
- Gen multiplies input integers
- Dec checks if candidate prime divides input integer

End of Line Problem

- Given $P, S : \{0,1\}^n \to \{0,1\}^n$ such that $P(0^n) = 0^n \neq S(0^n)$
- Find x such that $P(S(x)) \neq x$ or $S(P(x)) = x \neq 0^n$
- Gen concatenates input and output gates
- Dec restricts on the corresponding block

Open Problem 1

Average case hard problems in P

Open Problem 1

Average case hard problems in P

- Can we show some natural problem in P is hard for the uniform distribution?

Open Problem 1

Average case hard problems in P

- Can we show some natural problem in P is hard for the uniform distribution?
- Can we construct a fine-grained one way function from worst case assumptions?

Gap Amplification vs. Hardness Amplification

Open Problem 2

Gap Amplification vs. Hardness Amplification

- Can we obtain a trade-off between gap and hardness?

Open Problem 2

Gap Amplification vs. Hardness Amplification

- Can we obtain a trade-off between gap and hardness?
- Can we say something stronger about Max-SAT assuming Gap-ETH?

Open Problem 3

Direct Product Feasibility

Open Problem 3

Direct Product Feasibility

- Can we characterize direct product feasible pairs?

Open Problem 3

Direct Product Feasibility

- Can we characterize direct product feasible pairs?
- Can we show Orthogonal Vectors is self direct product feasible?

Open Problem 3

Direct Product Feasibility

- Can we characterize direct product feasible pairs?
- Can we show Orthogonal Vectors is self direct product feasible?
- Can we show LCS is self direct product feasible?

Key Takeaways

- Hardness Amplification Technique

Key Takeaways

- Hardness Amplification Technique
  - for Optimization problems
  - via Direct Products
  - against Randomized algorithms

Key Takeaways

- Hardness Amplification Technique
  - for Optimization problems
  - via Direct Products
  - against Randomized algorithms
- Hardness Amplification meets Fine-Grained Complexity

Key Takeaways

- Hardness Amplification Technique
  - for Optimization problems
  - via Direct Products
  - against Randomized algorithms

- Hardness Amplification meets Fine-Grained Complexity
  - Amplify hardness from $1/n^{o(1)}$ to $1 - o(1)$
    for LCS, Edit Distance, etc.

Key Takeaways

- Hardness Amplification Technique
  - for Optimization problems
  - via Direct Products
  - against Randomized algorithms
- Hardness Amplification meets Fine-Grained Complexity
  - Amplify hardness from $1/n^{o(1)}$ to $1 - o(1)$ for LCS, Edit Distance, etc.
  - If ETH is true on mild worst case then Max-SAT is hard on average

THANK
YOU!