Hardness Self-Amplification: Simplified, Optimized, and Unified

Nobutaka Shimizu
Tokyo Institute of Technology

joint work with Shuichi Hirahara
Average-Case Complexity

• How many **hard instances**?
  
  ▶ complexity of **random** instance

• **Motivation**: pessimism of worst-case complexity, derandomization, crypto

Reductions in worst-case hardness:

- ▲ may have “structure” due to the gadget construction.
  - hardness of “structural” instances
### Average-Case Complexity

- **How many hard instances?**
  - complexity of random instance

- **Motivation:** pessimism of worst-case complexity, derandomization, crypto

---

**Hardness of unstructured instances?**

- ▲ may have “structure” due to the gadget construction.
  - → hardness of “structural” instances
**Average-Case Complexity**

- **Algo $A$ computes $f$ with success probability $\gamma$ if** $\Pr_x[A(x) = f(x)] \geq \gamma$
  - $x$ is chosen from some distribution (over inputs of fixed size)

- **$f$ is worst-case hard** $\overset{\text{def}}{\iff}$ **efficient algo $A$, $\exists x$, $A(x) \neq f(x)$**

- **$f$ is strongly-hard** $\overset{\text{def}}{\iff}$ **efficient algo has success prob $\leq 0.01$**

\[
\begin{align*}
\text{f is strongly-hard} & \iff \text{trivial} \\
\text{strong random self-reduction} & \iff \text{f is worst-case hard}
\end{align*}
\]
**Average-Case Complexity**

- **Algo** $A$ computes $f$ with **success probability** $\gamma$ if $\Pr_{x}[A(x) = f(x)] \geq \gamma$
  - $x$ is chosen from some distribution (over inputs of fixed size)

- **$f$ is worst-case hard**

- **$f$ is strongly-hard**

---

**possible for some problems!**

Permanent [Cai, Pavan, Sivakumar, 1999]
Matrix multiplication [Asadi, Golovnev, Gur, Shinkar, 2022]
Discrete logarithm (folklore)
etc…

---

$f$ is strongly-hard **strong random self-reduction**

---

$f$ is worst-case hard
Algo $A$ computes $f$ with success probability $\gamma$ if $\Pr[A(x) = f(x)] \geq \gamma$

- $x$ is chosen from some distribution over inputs of fixed size.

- $f$ is worst-case hard.

- $f$ is strongly-hard.

**Possible for some problems!**

**What problems admit strong RSR?**

- Permanent [Cai, Pavan, Sivakumar, 1999]
- Matrix multiplication [Asadi, Golovnev, Gur, Shinkar, 2022]

$f$ is strongly-hard $\iff$ strong random self-reduction $\iff$ $f$ is worst-case hard.
Hardness Self-Amplification

- $f$ is weakly-hard $\overset{\text{def}}{\iff}$ \mbox{$\forall$ efficient algo has success prob $\leq 0.99$}
Our Results

Our Paper: hardness self-amplification for popular problems
- matrix multiplication (MM)
- online matrix-vector problem (OMv)
- triangle counting (TC)
- planted clique (PC)

Corollary
- new strong RSR for MM, OMv, TC
- search-to-decision reduction of PC
- improves and simplifies previous RSR for those problems
Our Results

- **Our Paper: hardness self-amplification for popular problems**
  - matrix multiplication (MM)
  - online matrix-vector problem (OMv)
  - triangle counting (TC)
  - planted clique (PC)

- **Our Ingredient**
  - A framework of hardness amplification using **expanders (samplers)**
  - The same framework was previously used to obtain Direct Product Theorem

[Impagliazzo, Jaiswal, Kabanets, Wigderson (2010)]
Matrix Multiplication
Matrix Multiplication (MM)

- Task: Multiply given $A, B \in \mathbb{F}^{n\times n}$ (finite field $\mathbb{F}$)
- Input distribution: Uniform

**Theorem (Blum, Luby, Rubinfeld, 1993)**

If we can solve MM with success prob 0.99 in time $T(n)$, then we can solve MM in time $O(T(n))$ for any input.
Matrix Multiplication (MM)

**Theorem (Asadi, Golovnev, Gur, Shinkar, 2022)**

If we can solve MM with success prob $\epsilon$ in time $T(n)$, then we can solve MM in time $2^{O(\log^5(1/\epsilon))} \cdot T(n)$ for any input.

- strong RSR 😊
- Tool: Additive Combinatorics (quasi-polynomial Bogolyubov-Ruzsa lemma)
Matrix Multiplication (MM)

**Theorem (this work)**
If we can solve MM with success prob $\epsilon$ in time $T(n)$, then we can solve MM in time $\frac{\text{polylog}(1/\epsilon)}{\epsilon} \cdot T(n)$ for any input.

- **Proof:** Hardness self-amplification + BLR93
- Matrices can be over finite ring
- **Improved** overhead ($2^{O(\log^5(1/\epsilon))} \rightarrow \tilde{O}(1/\epsilon)$)

**Strong hardness** → **this work** → **weak hardness** → **worst-case hardness**

---

[BLR93]
**Proof Sketch**

**Assumption:** \( \exists \text{ algo } M \text{ that solves MM with success prob } \epsilon \)

**Goal:** compute \( AB \) for any \( A, B \in \mathbb{F}^{n \times n} \)
Proof Sketch

Given $A, B \in \mathbb{F}^{n \times n}$ (worst-case instance)
Divide $A, B$ into $k$ submatrices $A_i \in \mathbb{F}^{(n/k) \times n}$ and $B_j \in \mathbb{F}^{n \times (n/k)}$. 

Proof Sketch
Proof Sketch

Product $AB$ has $k \times k$ blocks
Proof Sketch

We focus on MM for $A_iB_j$
(downward self-reduction)
Proof Sketch

By [BLR93], \( A_i, B_j \) can be random matrices (random self-reduction)
**Input:** random matrices $R, S$

**Goal:** compute $RS$ (with success prob 0.99)
If we can solve MM with success prob $\epsilon$ in time $T(n)$, then we can compute $RS$ with success prob $0.99$ for some $k = O(\log(1/\epsilon))$.

Lemma.

- Hardness (Self-) Amplification for MM
- idea: "upward-reduction"
- Essentially same as the proof of Direct Product Theorem
Proof Sketch

Sample \( n \times n \) random matrices \( \overline{R}, \overline{S} \)
Proof Sketch

Divide $R, S$ into $k$ submatrices: $R_i, S_j$ ($i, j \in [k]$)
Proof Sketch

Choose random $i \sim [k]$

$R_i \leftarrow R$ and $S_i \leftarrow S$
Proof Sketch

Let $\mathcal{R}^\mathcal{M}(R, S)$ be the algorithm that outputs $\mathcal{M}(\overline{R}, \overline{S})$
Proof Sketch

Output $T_{i,i}$ if $RS = T_{i,i}$
We can verify in time $O(n^2)$
Proof Sketch

Our algo: Run $R^M (R, S)$ until we find $RS$
Lemma

For 0.99-fraction of $(R, S)$, # of iteration is at most $O(1/e)$ if $k \geq 100 \log(1/e)$

Proof

- Expansion property (sampler) of query graph
Query Graph

\[ X = \text{set of all of inputs } (R, S) \]
$Y = \text{set of all pairs } (A, B) \text{ of } n \times n \text{ matrices}$
Query Graph

Edge weight = \( \Pr[\mathcal{R}(R, S) \text{ produces query } (A, B)] \)
Query Graph

Query = random neighbor of \((R, S)\)
Lemma (informal)

The query graph \((X, Y, E)\) has an \textbf{expansion} property if \(k \geq 100 \log(1/\epsilon)\)
**Sampler**

**Definition**

$Q = (X, Y, E)$ is a $(\delta, c)$-sampler for density $\epsilon$ if, for any $W \subseteq Y$ of $|W| \geq \epsilon |Y|$,\

$$\Pr_{x \sim X} \left[ \frac{|\Gamma(x) \cap W|}{|\Gamma(x)|} \geq (1 - c)\epsilon \right] \geq 1 - \delta,$$

where $\Gamma(x) = \{ \text{neighbors of } x \}$. 
Query Graph

\[ W = \{ y \in Y : \mathcal{M}(y) \text{ succeeds} \} \]
$W$ has density $\epsilon$ inside $Y$
For 99% of \((R, S)\), 
\(\epsilon/2\)-fraction of neighbors are in \(W\)
If we sample $O(1/\epsilon)$ random neighbors, one of them is in $W$.
The query graph \((X, Y, E)\) of MM is a \((\delta, c)\)-sampler for density \(\epsilon\) if

\[
k \geq \frac{8}{c^2 \delta} \log \left( \frac{2}{c \epsilon} \right).
\]

In MM, we set \(\delta = 0.99\) and \(c = 1/2\):

- \(k = O(\log(1/\epsilon))\) suffices
Proof Summary

- **downward self-reduction**

- **random self-reduction**

- **upward self-reduction**
Triangle Counting
Triangle Counting (TC)

- **Task:** How many triangles (3-cycles) in a given graph?
- **Input:** $G_{n,p}$ (for $p$ const)

**Theorem (Boix-Adserà, Brennan, Bresler, 2019)**

If we can solve TC with success prob $1 - 1/\text{polylog}(n)$ in time $T(n)$, then we can solve TC in time $T(n) \cdot \text{polylog}(n)$ for any input.

- weak RSR
- strong RSR: open
Triangle Counting (TC)

**Theorem (this work)**

\[ \exists T(n)\text{-time error-less algo for TC with success prob } \epsilon, \]

\[ \exists \frac{T(n)\text{polylog}(n)}{\epsilon}\text{-time nonuniform randomized algo that solves TC for any input.} \]

- **error-less algo**: output \( \in \{\text{answer, } \bot\} \)
  - never output a wrong value

- **nonuniform algo**: receives advice string \( \alpha \) as additional input
  - \( \alpha \) depends on input size \( n \) & random seed

- **Proof**: Hardness self-amplification + BBB19
Related Work

- **Counting (over $G_{n,p}$)**
  - k-clique [Boix-Adserà, Brennan, Bresler, 2019]
  - general [Dalirrooyfard, Lincoln, Williams, 2020]
  - low-error regime

- **Counting Mod 2**
  - k-clique (low-error regime) [Boix-Adserà, Brennan, Bresler, 2019], [Goldreich, 2020]
  - triangle *(nonuniform, strong RSR)* [Hirahara, S, 2022]

We simplified & improved this reduction
Proof Summary

- **downward self-reduction**
  - $n$ vertices $\rightarrow \frac{n}{k}$ vertices

- **random self-reduction**
  - [Boix-Adserà, Brennan, Bresler, 2019]

- **upward self-reduction**
  - $\frac{n}{k}$ vertices $\rightarrow n$ vertices
  - we use errorless + nonuniformity
We have an algo $M$ that solves TC with success prob $\epsilon$ over $G_{n,p}$

**Goal:** solve TC with success prob $1 - 1/\text{polylog}(n)$ over $G_{n/k,p}$
Upward Reduction

Input: $G \sim G_{n/k,p}$
Upward Reduction

\[ G_1 \]
\[ G_2 \]
\[ \vdots \]
\[ G_k \]

generate \( k \) graphs \( G_1, \ldots, G_k \sim G_{n/k,p} \)
Upward Reduction

Select $i \sim [k]$
Upward Reduction

\[
\begin{align*}
G_1 \\
G \\
\vdots \\
G_k
\end{align*}
\]

\[G_i \leftarrow G\]
Upward Reduction

Add random edges between two groups (with prob $p$)
Let $\overline{G}$ be the resulting graph

$(\overline{G} \sim G_{n,p} \text{ since } G \sim G_{n/k,p})$
Upward Reduction

Run $\mathcal{M}(\overline{G})$. 
Upward Reduction

- **How to obtain** \#Triangle(G)?
  - In reduction, we have \#Triangle(\overline{G}) if \mathcal{M} succeeds
  - this counts unnecessary triangles
Upward Reduction

- **How to obtain** \#Triangle(G)?
  - In reduction, we have \#Triangle(\overline{G}) if \(\mathcal{M}\) succeeds
  - this counts unnecessary triangles

- **Two types of edges**
  - type 1: edges inside \(G\)
  - type 2: others
Upward Reduction

- **How to obtain \#Triangle(\(G\))?**
  - In reduction, we have \#Triangle(\(\overline{G}\)) if \(M\) succeeds
  - this counts unnecessary triangles

- **Two types of edges**
  - type 1: edges inside \(G\)
  - type 2: others

- **Three types of triangles**
  - 
  - 
  - 
  -
Upward Reduction

• **How to obtain** \#Triangle(G)?
  - In reduction, we have \#Triangle(\overline{G}) if \mathcal{M} succeeds
  - this counts unnecessary triangles

• **Two types of edges**
  - type 1: edges inside \( G \)
  - type 2: others

• **Three types of triangles**
  - type1 + type1 + type1 (we want to count)
Upward Reduction

● **How to obtain** \( \#\text{Triangle}(G) \)?
  ‣ In reduction, we have \( \#\text{Triangle}(\overline{G}) \) if \( \mathcal{M} \) succeeds
  ‣ this counts unnecessary triangles

● **Two types of edges**
  ‣ type 1: edges inside \( G \)
  ‣ type 2: others

● **Three types of triangles**
  ‣ type1 + type1 + type1
  ‣ type1 + type2 + type2 (unnecessary)
Upward Reduction

**How to obtain** \( \#\text{Triangle}(G) \)?
- In reduction, we have \( \#\text{Triangle}(\overline{G}) \) if \( \mathcal{M} \) succeeds
- this counts unnecessary triangles

**Two types of edges**
- type 1: edges inside \( G \)
- type 2: others

**Three types of triangles**
- type1 + type1 + type1
- type1 + type2 + type2
- type2 + type2 + type2 (unnecessary)
Upward Reduction

- **How to obtain \#Triangle(G)?**
  - In reduction, we have \#Triangle(\bar{G}) if \(M\) succeeds
  - this counts unnecessary triangles

- **Two types of edges**
  - type 1: edges inside \(G\)
  - type 2: others

- **Three types of triangles**
  - type1 + type1 + type1
  - type1 + type2 + type2
  - type2 + type2 + type2 (unnecessary)

all edges independent of input we can give # of such triangle as **advice**!
Upward Reduction

- **How to obtain** $\#\text{Triangle}(G)$?
  - In reduction, we have $\#\text{Triangle}(\overline{G})$ if $M$ succeeds
  - this counts unnecessary triangles

- **Two types of edges**
  - type 1: edges inside $G$
  - type 2: others

- **Three types of triangles**
  - type 1 + type 1 + type 1
  - type 1 + type 2 + type 2
  - type 2 + type 2 + type 2 (unnecessary)

For each pink edge $uv$, we can give # of green $uv$-paths of length two as advice.
Upward Reduction

- **How to obtain** \( \#\text{Triangle}(G) \)?:
  - In reduction, we have \( \#\text{Triangle}(\overline{G}) \) if \( M \) succeeds
  - this counts unnecessary triangles

- **Two types of edges**
  - type 1: edges inside \( G \)
  - type 2: others

- **Three types of triangles**
  - type1 + type1 + type1
  - type1 + type2 + type2
  - type2 + type2 + type2 (unnecessary)

We can count in time \( O(n^2) \) using nonuniform advice!
Upward Reduction

- **How to obtain \#Triangle(G)?**
  - In reduction, we have \#Triangle(\overline{G}) if \mathcal{M} succeeds
  - this counts unnecessary triangles

**Lemma (informal)**

We can compute \#Triangle(G) in nonuniform time \(O(n^2)\) given \#Triangle(\overline{G}).

Advice: \(O(n^2 \log n)\) bits (# of green 2-paths)
Upward Self-Reduction

- **upward self-reduction**
  - $n/k$ vertices $\rightarrow n$ vertices
  - we use errorless + nonuniformity

**Lemma**
Query graph is a sampler.

- **We can boost the success prob of $M$ by repetition.**
Proof Summary

• **downward self-reduction**
  - $n$ vertices $\rightarrow n/k$ vertices

• **random self-reduction**
  - [Boix-Adserà, Brennan, Bresler, 2019]

• **upward self-reduction**
  - $n/k$ vertices $\rightarrow n$ vertices
  - we use errorless + nonuniformity
Planted Clique
Random Graph with Planted Clique

- **Input**: random $k$-clique + $G_{n,1/2}$ (Erdős-Rényi graph)
  - Sample $G_{n,1/2}$
  - Randomly choose a set $C \subseteq V$ of $k$ vertices
  - Make $C$ a $k$-clique by adding edges
  - let $G_{n,1/2,k}$ be the resulting graph

- **Maximum clique of** $G_{n,1/2} \approx 2 \log_2 n$
  - We assume $k \gg \log n$
  - Then, $C$ is the unique $k$-clique (whp)
**Search Planted Clique**

**Def (Search Planted Clique Problem)** [Jerrum, 92][Kučera, 95]

**Input:** \(G_{n,1/2,k}\)

**Output:** any \(k\)-clique (not necessarily be the planted one)

- If \(k = \Omega(\sqrt{n})\), \(\exists\) poly-time algo with success prob \(1 - 2^{-n^{0.1}}\)
  - the larger \(k\), the easier it is to solve

- open problem: poly-time algo for \(\log n \ll k \ll \sqrt{n}\)

[Alon, Krivelevich, Sudakov, 98]
[Deke, Gurel-Gurevich, Peres, 2014]
Decision Planted Clique

**Def (Decision Planted Clique Problem)**

Input: \( G_{n,1/2,k} \) (with prob 1/2) or \( G_{n,1/2} \) (with prob 1/2)

Output: “Yes” if the input contains a \( k \)-clique. “No” otherwise.

- **\( \mathcal{A} \) has advantage \( \gamma \) if** \( \Pr_{G}[\mathcal{A}(G) \text{ is correct}] \geq \frac{1 + \gamma}{2} \)
  - Random guess: \( \gamma = 0 \)
  - Goal: \( \gamma \approx 1 \)

- **Algo for Search Planted Clique \( \Rightarrow \) Algo for Decision Planted Clique**

- **Does converse hold?**
Previous Work

**Theorem (Alon, Andoni, Kaufman, Matulef, Rubinfeld, Xie, 2007).**

If we can decide $G_{n,1/2,k}$ or $G_{n,1/2}$ with advantage $1 - 1/n^2$,
then, we can find a $k$-clique in $G_{n,1/2,k}$ with success prob $1 - 1/n$.

- for low-error regime 😊
  - reduction has $n$ queries + union bound
Our Result

**Theorem.**

If we can decide $G_{N,1/2,k}$ or $G_{N,1/2}$ with advantage $\epsilon(N) \geq N^{-1/2+c}$, then, we can find a $k$-clique in $G_{n,1/2,k}$ with success prob $1 - 1/n$, where $N = n^{O(1/c)}$.

- **high-error regime!**
- **Blow-up in instance size 😞**
Proof Outline

- decision algo with adv $\epsilon$
  - hardness amplification polynomial blow-up in $n$
- decision algo with adv $1 - 1/n^2$
  - Search-to-Decision by [Alon, Andoni, Kaufman, Matulef, Rubinfeld, Xie, 07]
- search algo with success prob $1 - 1/n$
Our Reduction

• For simplicity we focus on Search Planted Clique

• $A$ : algo with success prob $\epsilon$

• $G$ : input (chosen from $G_{n,1/2,k}$)
Our Reduction

- For $N = \text{poly}(n)$, randomly embed $G$ into $G_{N,1/2}$. Let $\overline{G}$ be the resulting graph.
  - Let $R^A(G)$ be the randomized reduction that outputs $A(\overline{G})$

- Repeat $R^A(G)$ until $A(\overline{G})$ outputs a $k$-clique in $G$

- $\overline{G}$ contains a unique $k$-clique since $k \gg \log N$
**Def (Query Graph)**

The query graph is the edge-weighted bipartite graph $Q = (X, Y, P)$ defined by:

- $X =$ set of all $n$-vertex graph having a $k$-clique
- $Y =$ set of all $N$-vertex graph having a $k$-clique
- $P(G, H) = \Pr[\mathcal{R}^A(G) \text{ produces query } H]$
**Def (Query Graph)**

The query graph is the edge-weighted bipartite graph $Q = (X, Y, P)$ defined by:

- $X = \text{set of all } n\text{-vertex graph having a } k\text{-clique}$
- $Y = \text{set of all } N\text{-vertex graph having a } k\text{-clique}$
- $P(G, H) = \Pr[\mathcal{R}^A(G) \text{ produces query } H]$
Analysis

**Def (Query Graph)**

The query graph is the edge-weighted bipartite graph $Q = (X, Y, P)$ defined by

- $X =$ set of all $n$-vertex graph having a $k$-clique
- $Y =$ set of all $N$-vertex graph having a $k$-clique
- $P(G, H) = \Pr[\mathcal{R}^A(G) \text{ produces query } H]$

**Lemma**

The query graph $Q$ is a $(\delta, c)$-sampler for density $\epsilon$ if $N \geq \frac{n}{c^2\delta\epsilon}$
**Sampler and Expander**

- **Let** $P = [0,1]^{X \times Y}$ **be** $P(x, y) = \frac{1}{|N(x)|}$
  - $P(x, \cdot) = \text{upward random walk}$

- **Let** $P^\dagger \in [0,1]^{Y \times X}$ **be** $P^\dagger(y, x) = \frac{1}{|N(y)|}$
  - $P^\dagger(y, \cdot) = \text{downward random walk}$

**Lemma (informal)**

*If $\lambda_2(PP^\dagger)$ is small, then $Q$ is a sampler*
To bound $\lambda_2(PP^\dagger)$, we need **rapid mixing** of RW according to $PP^\dagger$
Up-Down Walk

This can be done by coupling technique of Markov chain
Why $\epsilon(N) \gg N^{-1/2}$?

**Theorem.**

If we can decide $G_{N,1/2,k}$ or $G_{N,1/2}$ with advantage $\epsilon(N) \geq N^{-1/2+c}$, then, we can find a $k$-clique in $G_{n,1/2,k}$ with success prob $1 - 1/n$, where $N = n^{O(1/c)}$.

- **Blow-up in instance size**
  - For MM and TC, we used downward self-reduction to preserve instance size.

- **For decision PC problem, we need a $(\delta, \epsilon/2)$-sampler for density $1/2 + \epsilon$**
  - Query graph is sampler if $N \geq \Theta \left( \frac{n}{\delta \epsilon^2} \right)$

- Here, $\epsilon = \epsilon(N)$ and thus, $\epsilon^2 \geq n/N = N^{-1+c}$ if $N = n^c$
Why $\epsilon(N) \gg N^{-1/2}$?

**Theorem.**

If we can decide $G_{N,1/2,k}$ or $G_{N,1/2}$ with advantage $\epsilon(N) \geq N^{-1/2+c}$, then, we can find a $k$-clique in $G_{n,1/2,k}$ with success prob $1 - 1/n$, where $N = n^{O(1/c)}$.

- **Blow-up in instance size**
  - For MM and TC, we used downward self-reduction to preserve instance size.

- **For decision PC problem, we need a $(\delta, \epsilon/2)$-sampler for density $1/2 + \epsilon$**

**Open Question.**

Can we improve the dependency of $N$ on $1/\epsilon$?

(in particular, we are interested in $\log(1/\epsilon)$)
Conclusion

- query graph is sampler $\implies$ hardness amplification
  - Matrix Multiplication
  - Online Matrix-Vector Multiplication
  - Triangle Counting
  - Planted Clique
- **Reduction**: Downward/Upward/Random Self-Reduction + Sampler
- **Further Application**: other “planted” problems (e.g., planted k-SUM)
- **Open**:
  - Improve the blow-up of $N = \text{poly}(n)$
  - Uniform reduction for triangle
  - General subgraph counting