Imperfect Gaps in Gap-ETH and PCPs

Mitali Bafna    Nikhil Vyas

Harvard        MIT
1. Introduction

2. Gap-ETH and Perfect Completeness

3. PCPs and Perfect Completeness
Introduction
Main Motivations

We study the role of perfect completeness:

• Hardness/Easiness of finding approximate solutions to satisfiable CSPs as compared to unsatisfiable ones?

• Is it easier to build PCPs with imperfect completeness as compared to perfect completeness?
Main Motivations

We study the role of perfect completeness:
Main Motivations

We study the role of perfect completeness:

- Hardness/Easiness of finding approximate solutions to satisfiable CSPs as compared to unsatisfiable ones?
Main Motivations

We study the role of perfect completeness:

- Hardness/Easiness of finding approximate solutions to satisfiable CSPs as compared to unsatisfiable ones?
- Is it easier to build PCPs with imperfect completeness as compared to perfect completeness?
Gap-ETH and Perfect Completeness
Constraint Satisfaction Problems (CSPs)

MAX\textsuperscript{k}-CSP(\(c, s\)):

- Given a \(k\)-width Boolean CSP, the problem of deciding
  - there exists an assignment satisfying more than a \(c\)-fraction of the clauses or
  - every assignment satisfies at most a \(s\)-fraction of the clauses.

We will also refer to this as Gap-\(k\)-CSP.

For this presentation, we will think of a Gap-CSPs on \(n\) variables and \(m = O(n)\) clauses.
MAX $k$-CSP$(c, s)$: Given a $k$-width Boolean CSP, the problem of deciding whether

- there exists an assignment satisfying more than a $c$-fraction of the clauses or
MAX $k$-CSP$(c, s)$: Given a $k$-width Boolean CSP, the problem of deciding whether

- there exists an assignment satisfying more than a $c$-fraction of the clauses or
- every assignment satisfies at most $s$ fraction of the clauses.
Constraint Satisfaction Problems (CSPs)

MAX $k$-CSP($c, s$): Given a $k$-width Boolean CSP, the problem of deciding whether

- there exists an assignment satisfying more than a $c$-fraction of the clauses or
- every assignment satisfies at most $s$ fraction of the clauses.
MAX \( k \)-CSP\((c, s)\): Given a \( k \)-width Boolean CSP, the problem of deciding whether

- there exists an assignment satisfying more than a \( c \)-fraction of the clauses or
- every assignment satisfies at most \( s \) fraction of the clauses.

We will also refer to this as Gap-\( k \)-CSP.
MAX $k$-CSP($c, s$): Given a $k$-width Boolean CSP, the problem of deciding whether

- there exists an assignment satisfying more than a $c$-fraction of the clauses or
- every assignment satisfies at most $s$ fraction of the clauses.

We will also refer to this as Gap-$k$-CSP.

For this presentation, we will think of a Gap-CSPs on $n$ variables and $m = O(n)$ clauses.
# Our Problems

**Problem (1)**

Is \( \text{MAX 3-SAT}(1, .98) \) “easier” than \( \text{MAX 3-SAT}(0.99, .97) \)?
Conjecture (Gap-ETH (Dinur’16 and MR’17))

For some constant $\tau > 0$, MAX 3-SAT$(1, 1 - \tau)$ does not have a $2^{o(n)}$ randomized algorithm.
The Gap-ETH Conjecture

**Conjecture (Gap-ETH(Dinur’16 and MR’17))**

*For some constant $\tau > 0$, MAX 3-SAT$(1, 1 - \tau)$ does not have a $2^{o(n)}$ randomized algorithm.*

**Conjecture (Gap-ETH without perfect completeness)**

*For some constants $\epsilon > \gamma > 0$, MAX 3-SAT$(1 - \gamma, 1 - \epsilon)$ does not have a $2^{o(n)}$ randomized algorithm.*
The Gap-ETH conjecture is equivalent to the Gap-ETH conjecture without perfect completeness i.e.

For all constants $\tau > 0$, $\text{MAX 3-SAT}(1, 1 - \tau)$ has a $2^{o(n)}$ time algorithm $\iff$ for all constants $\epsilon > \gamma > 0$, $\text{MAX 3-SAT}(1 - \gamma, 1 - \epsilon)$ has a $2^{o(n)}$ time algorithm.

Theorem
Equivalence of Gap-ETH conjectures

Theorem

The Gap-ETH conjecture is equivalent to the Gap-ETH conjecture without perfect completeness i.e.

For all constants $\tau > 0$, $\text{MAX 3-SAT}(1, 1 - \tau)$ has a $2^{o(n)}$ time algorithm $\iff$ for all constants $\epsilon > \gamma > 0$, $\text{MAX 3-SAT}(1 - \gamma, 1 - \epsilon)$ has a $2^{o(n)}$ time algorithm.

We will present:

Theorem

If for all constants $\tau > 0$, $\text{MAX 3-SAT}(1, 1 - \tau)$ has a $2^{o(n)}$ time randomized algorithm, then for all constants $\delta > 0$, $\text{MAX 3-SAT}(0.99, 0.97)$ has a $2^{\delta n}$ time randomized algorithm.
Proof Sketch

Lemma

For large enough constant $k$, there exists a randomized reduction from \( \text{MAX 3-SAT}(0.99, 0.97) \) on $n$ variables and $O(n)$ clauses to \( \text{MAX 3k-CSP}(1, 1/2) \) on $n$ variables and $O(n)$ clauses, such that:

- YES instances reduce to YES instances with probability $\geq 2^{-n/k}$.
- NO instances reduce to NO instances with probability $\geq 1 - 2^{-n}$. 
Getting Perfect Completeness starting from a YES case

\[ X_1 \cdot \cdot \cdot \cdot X_n \]
Getting Perfect Completeness starting from a YES case

\[ C_1 \cdots C_2 \cdots C_i \cdots C_j \cdots C_m \]

\[ \text{frac of 1's} > .99 \]

\[ \Pr[\text{Thr}.98 = 0] \leq 2^{-\Omega(k)} \]

Note that this gives us a $3^k$-CSP.
Getting Perfect Completeness starting from a YES case

\[\text{frac of 1's} > .99\]
Getting Perfect Completeness starting from a YES case

\[ \Pr[\text{Thr}_{.98} = 0] \leq 2^{-\Omega(k)} \]

\[
\begin{array}{ccccccc}
\text{(Thr}_{0.98})_1 & \cdots & \text{Thr}_{0.98} & \cdots & (\text{Thr}_{0.98})_n \\
k & \rightarrow & C_1 & \rightarrow & C_2 & \rightarrow & \cdots & \rightarrow & C_i & \rightarrow & \cdots & \rightarrow & C_j & \rightarrow & \cdots & \rightarrow & C_m \\
\end{array}
\]

\[ \text{frac of 1's} > .99 \]
Getting Perfect Completeness starting from a YES case

\[ \Pr[\text{Thr}.98 = 0] \leq 2^{-\Omega(k)} \]

\[ \text{wp} \geq 2^{-n/k} \]

frac of 1’s = 1

frac of 1’s > .99
Getting Perfect Completeness starting from a YES case

\[ \Pr[Thr_{.98} = 0] \leq 2^{-\Omega(k)} \]

Note that this gives us a 3\(k\)-CSP.
Soundness starting from a NO case

\[ X_1 \ldots X_n \]
Soundness starting from a NO case

\[ \frac{\text{frac of } 1\text{'s} < .97}{X_1 \rightarrow C_1 \rightarrow \cdots \rightarrow C_i \rightarrow \cdots \rightarrow C_j \rightarrow \cdots \rightarrow C_m \rightarrow X_n} \]
Soundness starting from a NO case

\[ \frac{1}{k} \frac{1}{s} \frac{1}{s} \leq 2 - \Omega(k) \]

\[ \frac{1}{s} < \frac{1}{2} \]

\[ k \rightarrow (\text{Thr}_{0.98})_{1} \rightarrow \cdots \rightarrow \text{Thr}_{0.98} \rightarrow \cdots \rightarrow (\text{Thr}_{0.98})_{n} \]

\[ C_{1} \rightarrow C_{2} \rightarrow \cdots \rightarrow C_{i} \rightarrow \cdots \rightarrow C_{j} \rightarrow \cdots \rightarrow C_{m} \]

\[ X_{1} \rightarrow \cdots \rightarrow X_{n} \]

frac of 1's < .97
Soundness starting from a NO case

\[ \Pr[\text{Thr}.98 = 1] \leq 2^{-\Omega(k)} \]

\[ \frac{\text{frac of } 1's}{.97} \]
Soundness starting from a NO case

\[ \Pr[\text{Thr}_{.98} = 1] \leq 2^{-\Omega(k)} \]

\[ \text{wp} \geq 1 - 2^{-n} \]

frac of 1's < 1/2

frac of 1's < 0.97
Proof Sketch

Lemma

For large enough constant $k$, there exists a randomized reduction from
MAX 3-SAT($0.99, 0.97$) on $n$ variables and $O(n)$ clauses to MAX
3-$k$-CSP($1, 1/2$) on $n$ variables and $O(n)$ clauses, such that:

- YES instances reduce to YES instances with probability $\geq 2^{-n/k}$.
- NO instances reduce to NO instances with probability $\geq 1 - 2^{-n}$. 
Proof Sketch

Lemma

For large enough constant $k$, there exists a randomized reduction from MAX 3-SAT(.99, .97) on $n$ variables and $O(n)$ clauses to MAX $3k$-CSP(1, 1/2) on $n$ variables and $O(n)$ clauses, such that:

- YES instances reduce to YES instances with probability $\geq 2^{-n/k}$.
- NO instances reduce to NO instances with probability $\geq 1 - 2^{-n}$.

- MAX $3k$-CSP(1, 1/2) on $n$ variables and $O(n)$ clauses can be converted to MAX 3-SAT(1, $1 - \Omega_k(1)$) on $n' = O_k(n)$ variables and clauses.
Lemma

For large enough constant $k$, there exists a randomized reduction from MAX 3-SAT(0.99, 0.97) on $n$ variables and $O(n)$ clauses to MAX 3$\cdot$k-CSP(1, 1/2) on $n$ variables and $O(n)$ clauses, such that:

- YES instances reduce to YES instances with probability $\geq 2^{-n/k}$.
- NO instances reduce to NO instances with probability $\geq 1 - 2^{-n}$.

- MAX 3$\cdot$k-CSP(1, 1/2) on $n$ variables and $O(n)$ clauses can be converted to MAX 3-SAT(1, 1 - $\Omega_k(1)$) on $n' = O_k(n)$ variables and clauses.
- Run the above reduction $2^{n/k}n^2$ times.
Lemma

For large enough constant $k$, there exists a randomized reduction from $\text{MAX }3\text{-SAT}(0.99, 0.97)$ on $n$ variables and $O(n)$ clauses to $\text{MAX }3k\text{-CSP}(1, 1/2)$ on $n$ variables and $O(n)$ clauses, such that:

- YES instances reduce to YES instances with probability $\geq 2^{-n/k}$.
- NO instances reduce to NO instances with probability $\geq 1 - 2^{-n}$.

- $\text{MAX }3k\text{-CSP}(1, 1/2)$ on $n$ variables and $O(n)$ clauses can be converted to $\text{MAX }3\text{-SAT}(1, 1 - \Omega_k(1))$ on $n' = O_k(n)$ variables and clauses.
- Run the above reduction $2^{n/k}n^2$ times.
- Run the $2^{o(n')}$ algorithm on the $\text{MAX }3\text{-SAT}(1, 1 - \Omega_k(1))$ instances and output YES if the algorithm outputs YES on any of the produced instances.
Lemma

For large enough constant $k$, there exists a randomized reduction from MAX 3-SAT($0.99, 0.97$) on $n$ variables and $O(n)$ clauses to MAX $3k$-CSP($1, 1/2$) on $n$ variables and $O(n)$ clauses, such that:

- YES instances reduce to YES instances with probability $\geq 2^{-n/k}$.
- NO instances reduce to NO instances with probability $\geq 1 - 2^{-n}$.

- MAX $3k$-CSP($1, 1/2$) on $n$ variables and $O(n)$ clauses can be converted to MAX 3-SAT($1, 1 - \Omega_k(1)$) on $n' = O_k(n)$ variables and clauses.
- Run the above reduction $2^{n/k}n^2$ times.
- Run the $2^{o(n')}$ algorithm on the MAX 3-SAT($1, 1 - \Omega_k(1)$) instances and output YES if the algorithm outputs YES on any of the produced instances.
- Total running time $2^{n/k}n^2 \cdot 2^{o(n')} = 2^{n/k+o(n)} \leq 2^{\delta n}$ for large enough constant $k$. 

Proof Sketch
One-sided derandomization using samplers. We use LLL to handle the completeness case.
PCPs and Perfect Completeness
Definition of PCPs

PCP_{c,s}[r, q] with proof size n:
Definition of PCPs

$\text{PCP}_{c,s}[r,q]$ with proof size $n$:

YES ($x \in L$): $\exists \Pi, \Pr_i[Q_i(\Pi) = 1] \geq c$

NO ($x \notin L$): $\forall \Pi, \Pr_i[Q_i(\Pi) = 1] \leq s$
Definition of PCPs

\[ \text{PCP}_{c,s}[r, q] \text{ with proof size } n: \]

YES \( (x \in L) \): \( \exists \ \Pi, \Pr_i[Q_i(\Pi) = 1] \geq c \)

NO \( (x \notin L) \): \( \forall \ \Pi, \Pr_i[Q_i(\Pi) = 1] \leq s \)
Definition of PCPs

PCP_{c,s}[r, q] with proof size $n$:

YES ($x \in L$): $\exists \, \Pi, \Pr_i[Q_i(\Pi) = 1] \geq c$

NO ($x \notin L$): $\forall \, \Pi, \Pr_i[Q_i(\Pi) = 1] \leq s$
Definition of PCPs

PCP_{c,s}[r, q] with proof size $n$:

YES ($x \in L$): $\exists \Pi, \Pr_i[Q_i(\Pi) = 1] \geq c$

NO ($x \not\in L$): $\forall \Pi, \Pr_i[Q_i(\Pi) = 1] \leq s$
PCP results

• PCP theorem [ALMSS]: For some constant $s < 1$, $\text{NTIME}[O(n)] \subseteq \text{PCP}_{1,s}[O(\log n), O(1)]$.

• Almost-linear proofs [Ben-Sasson, Sudan] and [Dinur]: $\text{NTIME}[O(n)] \subseteq \text{PCP}_{1,s}[\log n + O(\log \log n), O(1)]$.

• Linear-sized PCP with long queries [BKKMS’13]: $\text{NTIME}[O(n)] \subseteq \text{PCP}_{1/2, 1/2}[\log n + O(\epsilon), n \epsilon]$, with a $O(\epsilon n)$ proof size.
PCP results

- PCP theorem [ALMSS]: For some constant $s < 1$,

\[ \text{NTIME}[O(n)] \subseteq \text{PCP}_{1,s}[O(\log n), O(1)] \]
PCP results

- PCP theorem [ALMSS]: For some constant $s < 1$,

\[ \text{NTIME}[O(n)] \subseteq \text{PCP}_{1,s}[O(\log n), O(1)] \]

- Almost-linear proofs [Ben-Sasson, Sudan] and [Dinur]:

\[ \text{NTIME}[O(n)] \subseteq \text{PCP}_{1,s}[\log n + O(\log \log n), O(1)] \]
PCP results

- **PCP theorem** [ALMSS]: For some constant $s < 1$,

\[ NTIME[O(n)] \subseteq \text{PCP}_{1,s}[O(\log n), O(1)] \]

- Almost-linear proofs [Ben-Sasson, Sudan] and [Dinur]:

\[ NTIME[O(n)] \subseteq \text{PCP}_{1,s}[\log n + O(\log \log n), O(1)] \]

- Linear-sized PCP with long queries [BKKMS’13]:

\[ NTIME[O(n)] \subseteq \text{PCP}_{1,1/2}[\log n + O_\epsilon(1), n^\epsilon], \]

with a $O_\epsilon(n)$ proof size.
**Conjecture (Linear-sized PCP conjecture)**

$\text{NTIME}[O(n)]$ has linear-sized PCPs, i.e.

$\text{NTIME}[O(n)] \subseteq \text{PCP}_{1,s}[\log n + O(1), O(1)]$ for some constant $s < 1$. 
Our Question

• What is the role of completeness in PCPs? Can one build better PCPs with imperfect completeness?

• Can we convert an imperfect PCP to a perfect completeness PCP in a blackbox manner?
What is the role of completeness in PCPs? Can one build better PCPs with imperfect completeness?
What is the role of completeness in PCPs? Can one build better PCPs with imperfect completeness?

Can we convert an imperfect PCP to a perfect completeness PCP in a blackbox manner?
Ways to transfer gap

- One can just apply the best known PCPs for \( \text{NTIME}[O(n)] \), for example \( \text{MAX 3-SAT} \) \( \in \text{PCP}_{1,1-\Omega(1)}(\log n + O(\log \log n), O(1)) \).

- Bellare Goldreich and Sudan [1] studied many such black-box reductions between PCP classes. Their result for transferring the gap to \( \text{PCP}_{c,s}[r,q] \leq R_{\text{PCP}} 1,rs/c \leq R_{\text{PCP}} 1,qr/c \).
Ways to transfer gap

- One can just apply the best known PCPs for NTIME[O(n)], for example
  MAX 3-SAT(.99, .97) \(\in\) PCP\(_{1,1-\Omega(1)}(\log n + O(\log \log n), O(1))\)
Ways to transfer gap

- One can just apply the best known PCPs for $\NTIME[O(n)]$, for example $\text{MAX 3-SAT}(.99, .97) \in \text{PCP}_{1,1-\Omega(1)}(\log n + O(\log \log n), O(1))$

- Bellare Goldreich and Sudan [1] studied many such black-box reductions between PCP classes. Their result for transferring the gap to 1:
Ways to transfer gap

- One can just apply the best known PCPs for NTIME[O(n)], for example
  \[ \text{MAX 3-SAT(.99, .97)} \in \text{PCP}_{1,1-\Omega(1)}(\log n + O(\log \log n), O(1)) \]
- Bellare Goldreich and Sudan [1] studied many such black-box reductions between PCP classes. Their result for transferring the gap to 1:
  \[ \text{PCP}_{c,s}[r, q] \leq_R \text{PCP}_{1,rs/c}[r, qr/c]. \]
Our Result

We show a blackbox way to transfer a PCP with imperfect completeness to one with perfect completeness, while incurring a small loss in the query complexity, but maintaining other parameters of the original PCP.

From now on, we will take $(c, s) = (9/10, 6/10)$. Let $L$ have a PCP with $c = 0.9$, $s = 0.6$, with total verifier queries $m$. We will show how to build a new proof system (specify proof bits and verifier queries) for $L$ that has completeness 1 and soundness $< 1$. 
Our Result

Gap-Transfer theorem

We show a blackbox way to transfer a PCP with imperfect completeness to one with perfect completeness, while incurring a small loss in the query complexity, but maintaining other parameters of the original PCP.
Our Result

**Gap-Transfer theorem**

We show a blackbox way to transfer a PCP with imperfect completeness to one with perfect completeness, while incurring a small loss in the query complexity, but maintaining other parameters of the original PCP.

From now on, we will take \((c, s) = (9/10, 6/10)\). Let \(L\) have a PCP with \(c = 0.9, s = 0.6\), with total verifier queries \(= m\).
Our Result

Gap-Transfer theorem

We show a blackbox way to transfer a PCP with imperfect completeness to one with perfect completeness, while incurring a small loss in the query complexity, but maintaining other parameters of the original PCP.

From now on, we will take \((c, s) = (9/10, 6/10)\). Let \(L\) have a PCP with \(c = 0.9, s = 0.6\), with total verifier queries \(= m\).

We will show how to build a new proof system (specify proof bits and verifier queries) for \(L\) that has completeness 1 and soundness \(< 1\).
A Robust Circuit using Thresholds

$$\prod_1 \cdots \cdot \prod_n$$

We can derandomize this using samplers.
A Robust Circuit using Thresholds

\[ \prod_1 \cdots \prod_n \]

\[ C_1 \cdots C_2 \cdots C_i \cdots C_j \cdots C_m \]

We can derandomize this using samplers.
A Robust Circuit using Thresholds

We can derandomize this using samplers.
A Robust Circuit using Thresholds

We can derandomize this using samplers.
A Robust Circuit using Thresholds

We can derandomize this using samplers.
A Robust Circuit using Thresholds

$\Pi_1 \cdots \Pi_n$

$C_1 \cdots C_2 \cdots C_i \cdots C_j \cdots C_m$

$(\text{Thr}_{0.8})_1 \cdots \text{Thr}_{0.8} \cdots (\text{Thr}_{0.8})_{m/2}$

$O(1)$

$\log m$ layers

We can derandomize this using samplers.
A Robust Circuit using Thresholds

We can derandomize this using samplers.
Increasing fraction of 1’s

\[ \prod_{1}^{n} \cdot \cdot \cdot \cdot \cdot \prod_{n} \]
Increasing fraction of 1’s

\[ \prod_{1} \ldots \prod_{n} \]

\[ C_{1} \ldots C_{2} \ldots C_{i} \ldots C_{j} \ldots C_{m} \]

\[ \text{frac of } 0\text{’s } < \frac{1}{2} \]

\[ \text{frac of } 0\text{’s } < \frac{1}{2} \text{ i.e., log } m \text{ layers} \]
Increasing fraction of 1’s

\[
\prod_{1}^{n} C_i \cdot \cdot \cdot C_j \cdot \cdot \cdot C_m
\]

\[
(\text{Thr}_{0.8})_1 \cdot \cdot \cdot \text{Thr}_{0.8} \cdot \cdot \cdot (\text{Thr}_{0.8})^{m/2}
\]

\[
O(1) \cdot \cdot \cdot \frac{\text{frac of 0’s} < 0.1}{\cdot \cdot \cdot \frac{\text{frac of 0’s} < 0.1/2}{\cdot \cdot \cdot \frac{\text{frac of 0’s} = 0}{\cdot \cdot \cdot \frac{\text{frac of 0’s} = 0.19}{\cdot \cdot \cdot \frac{\text{frac of 0’s} = 0.2}{}}}{}_{\cdot \cdot \cdot \frac{\text{frac of 0’s} = 0.21}{}}}{}
\]
Increasing fraction of 1’s

\[ \prod_{1} \cdots C_{2} \cdots C_{i} \cdots C_{j} \cdots C_{m} \]

\[ O(1) \]

\[ (\text{Thr}_{0.8})_{1} \cdots \text{Thr}_{0.8} \cdots (\text{Thr}_{0.8})^{m/2} \]

\[ \frac{\text{frac of } 0's}{2} < .1 \]

\[ \frac{\text{frac of } 0's}{2} < .1 \]
Increasing fraction of 1’s

\[ \prod_{i=1}^{n} \prod_{j=1}^{m} (\text{Thr}_{0.8})_i \cdot \ldots \cdot \text{Thr}_{0.8} \cdot \ldots \cdot (\text{Thr}_{0.8})_{m/2} \]

\[ O(1) \rightarrow \frac{\text{frac of } 0's < \frac{1}{2^i}}{\log m \text{ layers}} \]

\[ \frac{\text{frac of } 0's < \frac{1}{2}}{\frac{\text{frac of } 0's < \frac{1}{2}}{i \log m \text{ layers}} = 0} \]
Increasing fraction of 1’s

\[ \prod_{1} \cdots \prod_{n} \]

\[ \text{Thr}_{0.8} \]

\[ \text{log } m \text{ layers} \]

\[ O(1) \]

\[ (\text{Thr}_{0.8})_1 \rightarrow \cdots \rightarrow \text{Thr}_{0.8} \rightarrow \cdots \rightarrow (\text{Thr}_{0.8})_{m/2} \]

\[ C_1 \cdots C_2 \cdots C_i \cdots C_j \cdots C_m \]

\[ \text{frac of 0’s} = 0 \]

\[ \text{frac of 0’s} < \frac{1}{2^i} \]

\[ \text{frac of 0’s} < \frac{1}{2} \]

\[ \text{frac of 0’s} < 0.1 \]
Maintaining fraction of 1’s

\[ \prod_{1}^{n} \frac{c_{1} \ldots c_{i} \ldots c_{j} \ldots c_{m}}{\text{Thr}_{0.8} \ldots \text{Thr}_{0.8}} \frac{\text{log} m \text{ layers}}{\frac{1}{10}} < \frac{7}{10} \]

\[ O(1) \]
Maintaining fraction of 1’s

\[ \prod_{i=1}^{n} \phantom{\prod} \quad C_1 \cdots C_2 \cdots C_i \cdots C_j \cdots C_m \]

\[ \frac{\text{frac of 1's}}{\text{< 7/10}} \]
Maintaining fraction of 1’s

\[
\prod_{1}^{n}
\]

\[
C_1 \cdots C_2 \cdots C_i \cdots C_j \cdots C_m
\]

\[
O(1) \quad (\text{Thr}_{0.8})_1 \quad \cdots \quad \text{Thr}_{0.8} \quad \cdots \quad (\text{Thr}_{0.8})^{m/2}
\]

\[
\frac{\text{frac of 1’s}}{< \frac{7}{10}}
\]
Maintaining fraction of 1’s

\[ \Pi_1 \cdots \Pi_n \]

\[ C_1 \cdots C_2 \cdots C_i \cdots C_j \cdots C_m \]

\[ (\text{Thr}_{0.8})_1 \cdots \text{Thr}_{0.8} \cdots (\text{Thr}_{0.8})_{m/2} \]

\[ O(1) \]

frac of 1’s < 6/10

frac of 1’s < 7/10
Maintaining fraction of 1’s

\[ \prod_{i=1}^{n} C_i \]

\[ \text{Thr}_{0.8} \]

\[ \text{log } m \text{ layers} \]

\[ \frac{\text{frac of 1’s}}{< \frac{7}{10}} \]

\[ \frac{\text{frac of 1’s}}{< \frac{6}{10}} \]
In a single query, we will verify all included gates: check whether each gate’s output is consistent with its inputs and the top gate evaluates to 1.
In a single query, we will verify all included gates: check whether each gate’s output is consistent with its inputs and the top gate evaluates to 1.
Parameters of the Reduction

This gives us a PCP that has the following properties:

- **Completeness**: 1
- **Soundness**: $\frac{9}{10}$
- **Queries**: $q + O(\log m) = q + O(r)$
- **Randomness complexity**: $r$ (stays the same)
- **Size**: $O(m)$
Parameters of the Reduction

This gives us a PCP that has the following properties:
Parameters of the Reduction

This gives us a PCP that has the following properties:

- Completeness: 1
Parameters of the Reduction

This gives us a PCP that has the following properties:

- Completeness: 1
- Soundness: 9/10
This gives us a PCP that has the following properties:

- Completeness: 1
- Soundness: $9/10$
- Queries: $q + O(\log m) = q + O(r)$
Parameters of the Reduction

This gives us a PCP that has the following properties:

- Completeness: 1
- Soundness: 9/10
- Queries: $q + O(\log m) = q + O(r)$
- Randomness complexity: $r$ (stays the same)
Parameters of the Reduction

This gives us a PCP that has the following properties:

- Completeness: 1
- Soundness: 9/10
- Queries: $q + O(\log m) = q + O(r)$
- Randomness complexity: $r$ (stays the same)
- Size: $O(m)$
Theorem

For all constants, $c$, $s$, $s' \in (0, 1)$ with $s < c$, we have that,

$$\text{PCP} \left[ c, s \right] \subseteq \text{PCP} \left[ 1, s' \right] \cup O \left( 1 \right), q + O \left( r \right)$$

We have a similar “randomized reduction” between PCP classes where the new randomness and query complexities have better dependence on the initial $r$, $q$. 

Theorem

For all constants, $c$, $s$, $s' \in (0, 1)$ with $s < c$, we have that,

$$\text{PCP} \left[ c, s \right] \leq R \text{PCP} \left[ 1, s' \right] \cup O \left( 1 \right), q + O \left( \log r \right)$$
Theorem

For all constants, \( c, s, s' \in (0, 1) \) with \( s < c \), we have that,

\[
PCP_{c,s}[r, q] \subseteq PCP_{1,s'}[r + O(1), q + O(r)].
\]
Main theorem

**Theorem**

For all constants, $c, s, s' \in (0, 1)$ with $s < c$, we have that,

$$PCP_{c, s}[r, q] \subseteq PCP_{1, s'}[r + O(1), q + O(r)].$$

We have a similar “randomized reduction” between PCP classes where the new randomness and query complexities have better dependence on the initial $r, q$. 
Main theorem

**Theorem**

For all constants, $c, s, s' \in (0, 1)$ with $s < c$, we have that,

$$PCP_{c,s}[r, q] \subseteq PCP_{1,s'}[r + O(1), q + O(r)].$$

We have a similar “randomized reduction” between PCP classes where the new randomness and query complexities have better dependence on the initial $r, q$.

**Theorem**

For all constants, $c, s, s' \in (0, 1)$ with $s < c$, we have that,

$$PCP_{c,s}[r, q] \leq_{R} PCP_{1,s'}[r + O(1), q + O(\log r)].$$
Comparison to Best-Known PCPs

We get the following result for $\text{NTIME}[O(n)]$:

**Corollary**

For all constants, $c, s, s', \text{if } \text{NTIME}[O(n)] \subseteq \text{PCP}_{c, s}[\log n + O(1), q]$, then $\text{NTIME}[O(n)] \subseteq \text{PCP}_{1, s'}[\log n + O(1), q + O(\log n)]$.

While the current best known linear-sized PCP is:

$\text{NTIME}[O(n)] \subseteq \text{PCP}_{1, s}[\log n + O(1), n^{\epsilon}]$.
We get the following result for $\text{NTIME}[O(n)]$:

**Corollary**

For all constants, $c, s, s'$, if $\text{NTIME}[O(n)] \subseteq \text{PCP}_{c,s}[\log n + O(1), q]$, then $\text{NTIME}[O(n)] \subseteq \text{PCP}_{1,s'}[\log n + O(1), q + O(\log n)]$. 
We get the following result for $NTIME[O(n)]$:

**Corollary**

For all constants, $c, s, s'$, if $NTIME[O(n)] \subseteq PCP_{c,s}[\log n + O(1), q]$, then $NTIME[O(n)] \subseteq PCP_{1,s'}[\log n + O(1), q + O(\log n)]$.

While the current best known linear-sized PCP is:

$$NTIME[O(n)] \subseteq PCP_{1,s}[\log n + O_\epsilon(1), n^\epsilon],$$
Conclusion

Our results imply that building linear-sized PCPs with minimal queries for \( \text{NTIME} \left[ O(n) \right] \) and perfect completeness should be nearly as hard (or easy!) as linear-sized PCPs with minimal queries for \( \text{NTIME} \left[ O(n) \right] \) and imperfect completeness.

We show the equivalence of Gap-ETH under perfect and imperfect completeness, i.e. Max-3SAT with perfect completeness has \( 2^{o(n)} \) randomized algorithms iff Max-3SAT with imperfect completeness has \( 2^{o(n)} \) algorithms.
Our results imply that building linear-sized PCPs with minimal queries for $\text{NTIME}[O(n)]$ and perfect completeness should be nearly as hard (or easy!) as linear-sized PCPs with minimal queries for $\text{NTIME}[O(n)]$ and imperfect completeness.
Our results imply that building linear-sized PCPs with minimal queries for $\text{NTIME}[O(n)]$ and perfect completeness should be nearly as hard (or easy!) as linear-sized PCPs with minimal queries for $\text{NTIME}[O(n)]$ and imperfect completeness.

We show the equivalence of Gap-ETH under perfect and imperfect completeness, i.e. Max-3SAT with perfect completeness has $2^{o(n)}$ randomized algorithms iff Max-3SAT with imperfect completeness has $2^{o(n)}$ algorithms.
Open Problems

• A query reduction on our result for PCPs, using [Dinur], gives that:

\[ \text{Corollary} \]

\[ \text{NTIME}\left[O(n)\right] \subseteq \text{PCP}_{c,s}[\log n, O(1)] \]

This is what one gets using the current PCPs for \( \text{NTIME}[O(n)] \).

Can one prove that, \( \text{PCP}_{c,s}[\log n + O(\log \log n), O(1)] \subseteq \text{PCP}_{1,s'[\log n + o(\log \log n), O(1)]} \)?

• Can we derandomize the reduction from Gap-ETH without perfect completeness to Gap-ETH?

• Blackbox reductions to get better parameters for MAX \( k \)-CSP?

Currently we know that MAX \( k \)-CSP \((1, 2^{O(k^{1/3})/2k})\) for satisfiable instances whereas for unsatisfiable instances MAX \( k \)-CSP \((1 - \epsilon, 2k/2k)\) (which is tight up to constant factors).
A query reduction on our result for PCPs, using [Dinur], gives that:

**Corollary**

If $\text{NTIME}[O(n)] \subseteq \text{PCP}_{c,s}[\log n, O(1)]$, then $\text{NTIME}[O(n)] \subseteq \text{PCP}_{1,s'}[\log n + O(\log \log n), O(1)]$. 

• Can one prove that, $\text{PCP}_{c,s}[\log n + O(1)] \subseteq \text{PCP}_{1,s'}[\log n + o(\log \log n), O(1)]$?

• Can we derandomize the reduction from Gap-ETH without perfect completeness to Gap-ETH?

• Blackbox reductions to get better parameters for MAX $k$-CSP?

Currently we know that MAX $k$-CSP $(1, 2^{O(k^{1/3})}/2^k)$ for satisfiable instances whereas for unsatisfiable instances MAX $k$-CSP $(1-\epsilon, 2^k/2^k)$ (which is tight up to constant factors).
A query reduction on our result for PCPs, using [Dinur], gives that:

**Corollary**

If $\text{NTIME}[O(n)] \subseteq \text{PCP}_{c,s}[\log n, O(1)]$, then $\text{NTIME}[O(n)] \subseteq \text{PCP}_{1,s'}[\log n + O(\log \log n), O(1)]$.

This is what one gets using the current PCPs for $\text{NTIME}[O(n)]$. Can one prove that, $\text{PCP}_{c,s}[\log n + O(1), O(1)] \subseteq \text{PCP}_{1,s'}[\log n + o(\log \log n), O(1)]$?
Open Problems

- A query reduction on our result for PCPs, using [Dinur], gives that:

**Corollary**

If $\text{NTIME}[O(n)] \subseteq \text{PCP}_{c,s}[\log n, O(1)]$, then
$\text{NTIME}[O(n)] \subseteq \text{PCP}_{1,s'}[\log n + O(\log \log n), O(1)]$.

This is what one gets using the current PCPs for $\text{NTIME}[O(n)]$.
Can one prove that,
$\text{PCP}_{c,s}[\log n + O(1), O(1)] \subseteq \text{PCP}_{1,s'}[\log n + o(\log \log n), O(1)]$?

- Can we derandomize the reduction from Gap-ETH without perfect completeness to Gap-ETH?
Open Problems

- A query reduction on our result for PCPs, using [Dinur], gives that:

**Corollary**

If $\text{NTIME}[O(n)] \subseteq \text{PCP}_{c,s}[\log n, O(1)]$, then $\text{NTIME}[O(n)] \subseteq \text{PCP}_{1,s'}[\log n + O(\log \log n), O(1)]$.

This is what one gets using the current PCPs for $\text{NTIME}[O(n)]$. Can one prove that, $\text{PCP}_{c,s}[\log n + O(1), O(1)] \subseteq \text{PCP}_{1,s'}[\log n + o(\log \log n), O(1)]$?

- Can we derandomize the reduction from Gap-ETH without perfect completeness to Gap-ETH?

- Blackbox reductions to get better parameters for $\text{MAX } k$-CSP? Currently we know that $\text{MAX } k$-CSP$(1, 2^{O(k^{1/3})}/2^k)$ for satisfiable instances whereas for unsatisfiable instances $\text{MAX } k$-CSP$(1 - \epsilon, 2k/2^k)$ (which is tight up to constant factors).
Thanks! Questions?
M. Bellare, O. Goldreich, and M. Sudan. 
**Free bits, pcps, and nonapproximability-towards tight results.**  