Imperfect Gaps in Gap-ETH and PCPs

Mitali Bafna Nikhil Vyas

Harvard MIT

Table of contents

Introduction

Main Motivations

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)

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

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.

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.

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.

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

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.

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 MAX 3-SAT$(1, .98)$ *"easier" than MAX 3-SAT*$(.99, .97)$*?*

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.

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.

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

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

We will present:

Theorem

If for all constants $\tau > 0$, *MAX 3-SAT*$(1, 1 - \tau)$ *has a* $2^{o(n)}$ *time randomized algorithm, then for all constants* $\delta > 0$, *MAX 3-SAT*$(.99, .97)$ *has a* $2^{\delta n}$ *time randomized algorithm.*

Proof Sketch

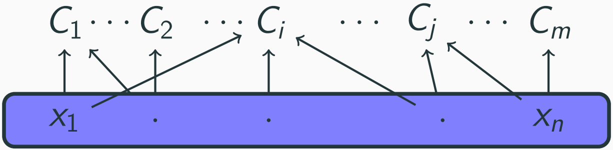
Lemma

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 $\geq 1 - 2^{-n}$.*

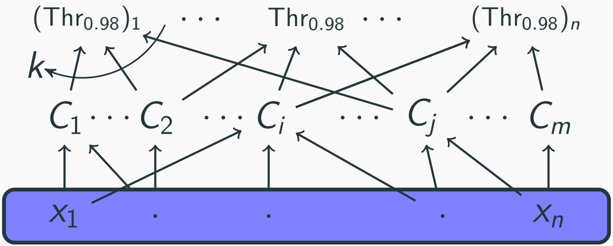
Getting Perfect Completeness starting from a YES case

Getting Perfect Completeness starting from a YES case



frac of $1's > .99$

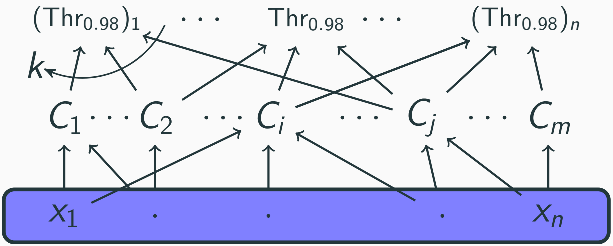
Getting Perfect Completeness starting from a YES case



frac of $1's > .99$

Getting Perfect Completeness starting from a YES case

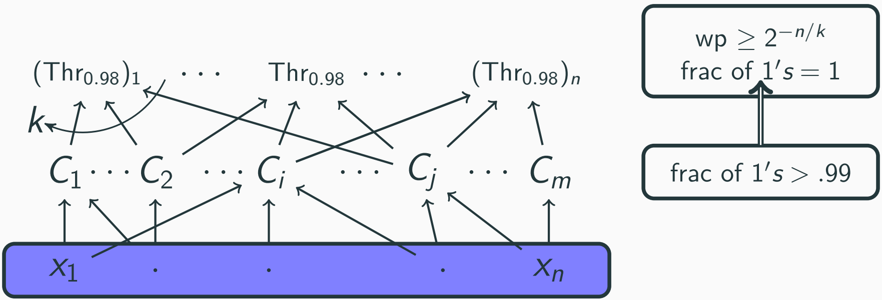
$\Pr[\mathrm{Thr}_{.98} = 0] \leq 2^{-\Omega(k)}$



frac of $1's > .99$

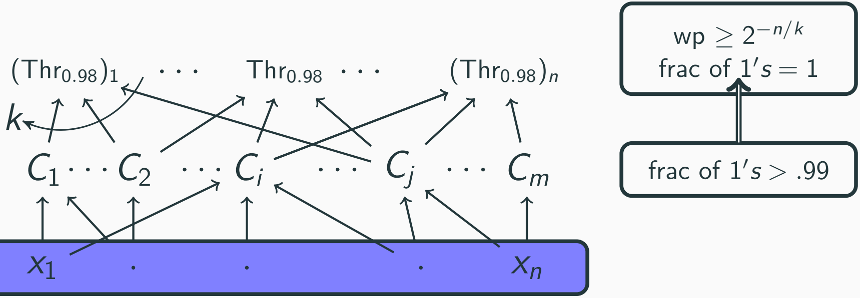
Getting Perfect Completeness starting from a YES case

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



Getting Perfect Completeness starting from a YES case

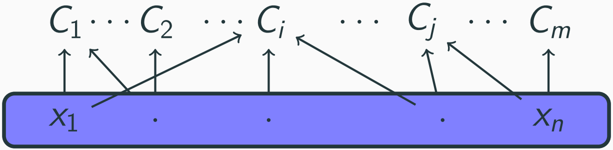
$\Pr[\text{Thr}_{.98} = 0] \leq 2^{-\Omega(k)}$



Note that this gives us a $3k$-CSP.

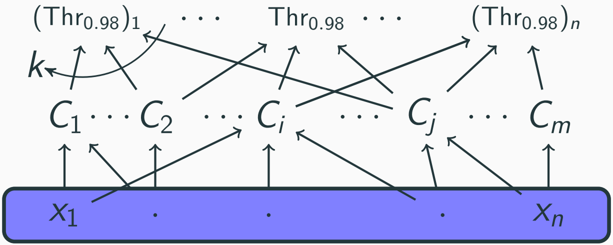
Soundness starting from a NO case

Soundness starting from a NO case



frac of $1's < .97$

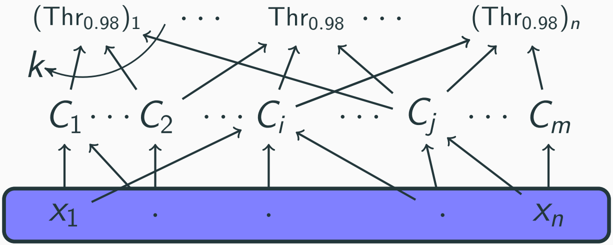
Soundness starting from a NO case



frac of $1's < .97$

Soundness starting from a NO case

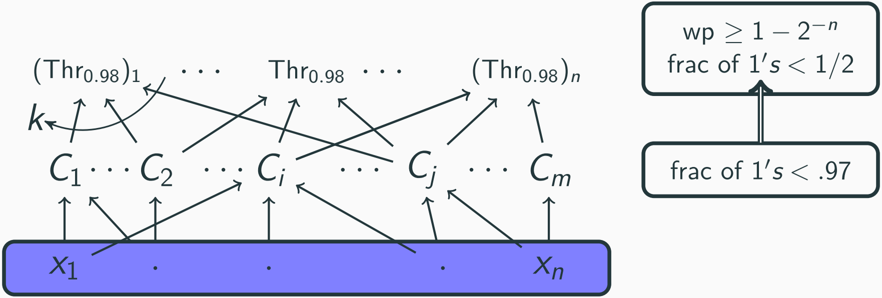
$\Pr[\mathrm{Thr}_{.98} = 1] \leq 2^{-\Omega(k)}$



frac of $1's < .97$

Soundness starting from a NO case

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



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 $\geq 1 - 2^{-n}$.*

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.

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.
- Run the above reduction $2^{n/k} n^2$ times.

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.
- 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.

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.
- 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.

Derandomization using samplers

- One-sided derandomization using samplers. We use LLL to handle the completeness case.

PCPs and Perfect Completeness

Definition of PCPs

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

Definition of PCPs

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

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

Definition of PCPs

$\mathrm{PCP}_{c,s}[r, q]$ 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

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

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

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

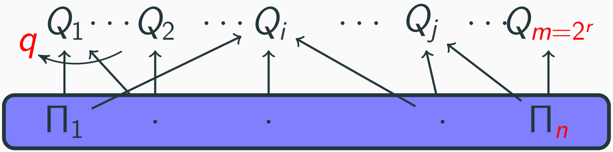
$$\Pi_1 \quad \cdot \quad \cdot \quad \cdot \quad \Pi_n$$

Definition of PCPs

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

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

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



PCP results

PCP results

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

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

PCP results

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

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

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

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

PCP results

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

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

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

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

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

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

with a $O_\epsilon(n)$ proof size.

Linear-Sized PCP conjecture

Conjecture (Linear-sized PCP conjecture)

$NTIME[O(n)]$ has linear-sized PCPs, i.e.
$NTIME[O(n)] \subseteq PCP_{1,s}[\log n + O(1), O(1)]$ for some constant $s < 1$.

Our Question

Our Question

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

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?

Ways to transfer gap

Ways to transfer gap

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

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
  MAX 3-$SAT(.99, .97) \in \mathrm{PCP}_{1,1-\Omega(1)}(\log n + O(\log\log n), O(1))$
- Bellare Goldreich and Sudan [1] studied many such black-box reductions between PCP classes. Their result for transferring the gap to 1:

$$\mathrm{PCP}_{c,s}[r, q] \leq_R \mathrm{PCP}_{1,rs/c}[r, qr/c].$$

Our Result

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.

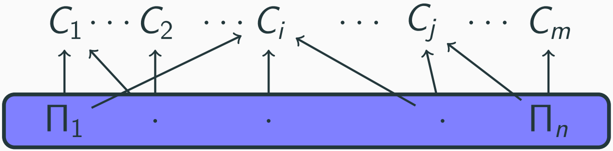
From now on, we will take $(c, s) = (9/10, 6/10)$. Let L have a PCP with $c = 0.9, s = 0.6$, with total verifier queries $= m$.

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.

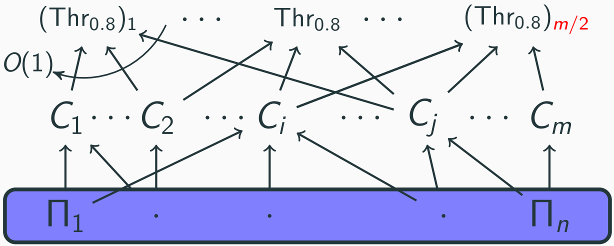
A Robust Circuit using Thresholds

Π_1 · · · Π_n

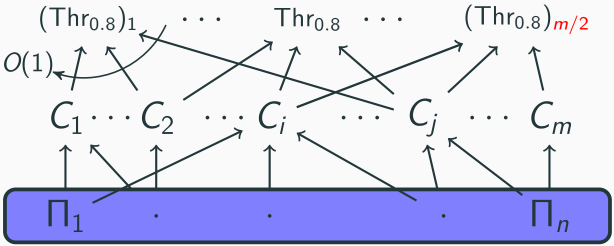
A Robust Circuit using Thresholds



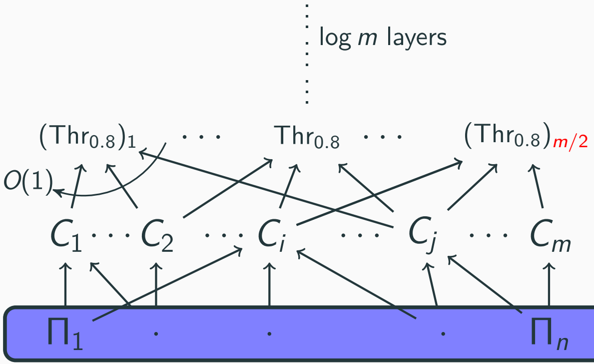
A Robust Circuit using Thresholds



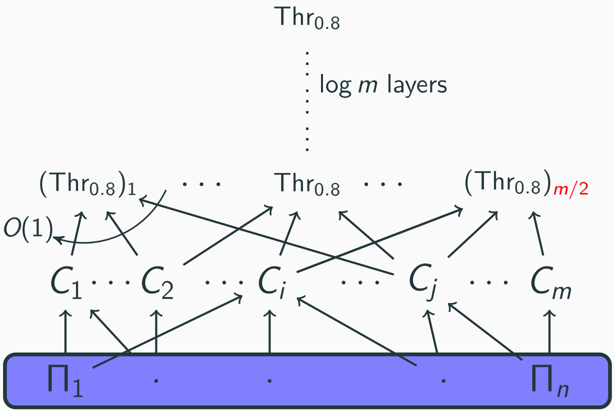
A Robust Circuit using Thresholds



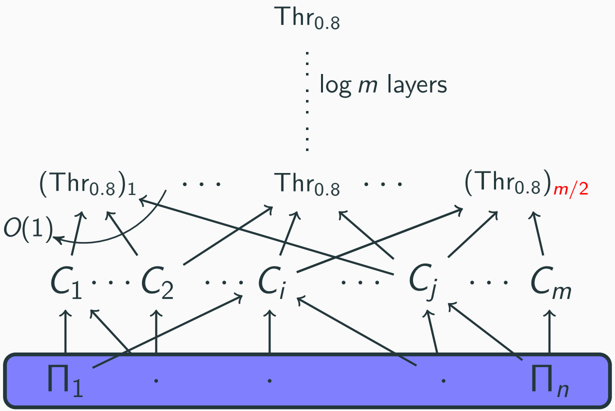
A Robust Circuit using Thresholds



A Robust Circuit using Thresholds



A Robust Circuit using Thresholds

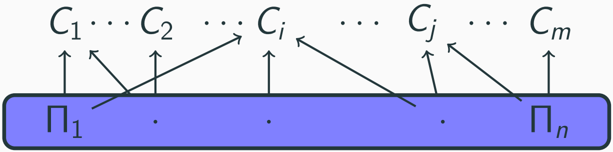


We can derandomize this using samplers.

Increasing fraction of 1's

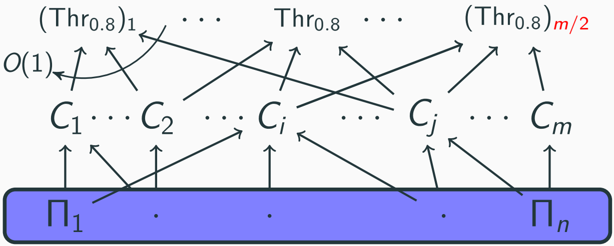
$\Pi_1 \quad \cdot \quad \cdot \quad \cdot \quad \Pi_n$

Increasing fraction of 1's



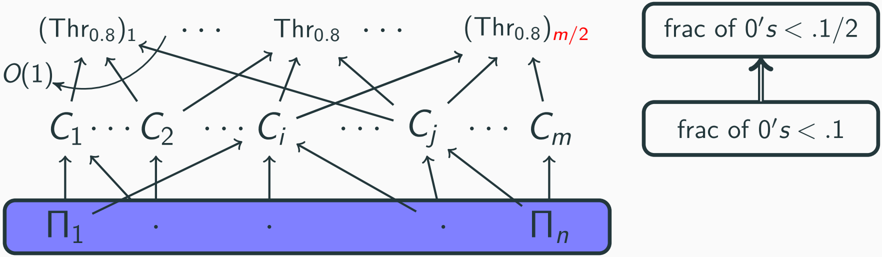
frac of $0's < .1$

Increasing fraction of 1’s

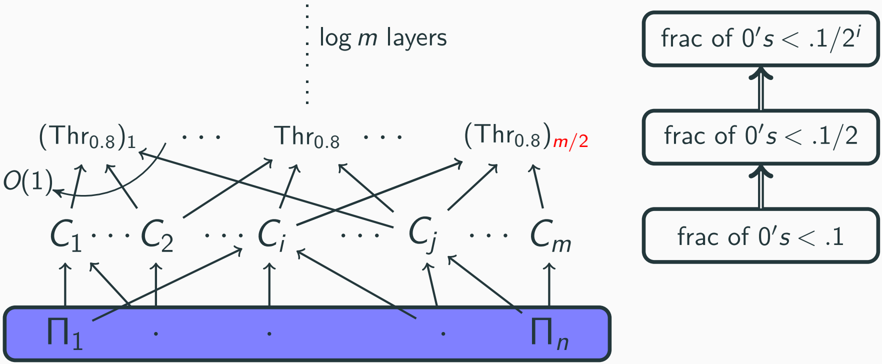


frac of $0's < .1$

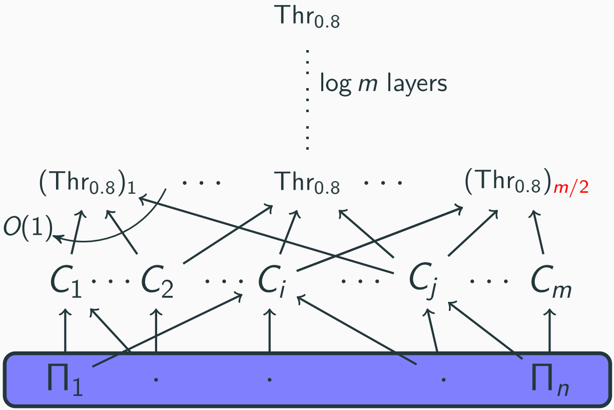
Increasing fraction of 1's

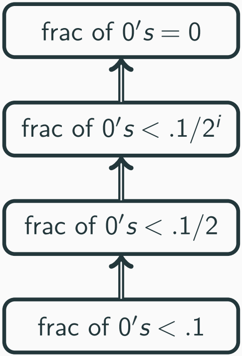


Increasing fraction of 1's



Increasing fraction of 1's

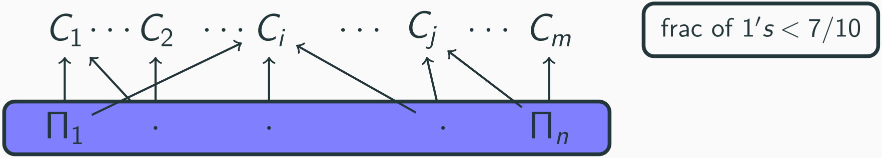




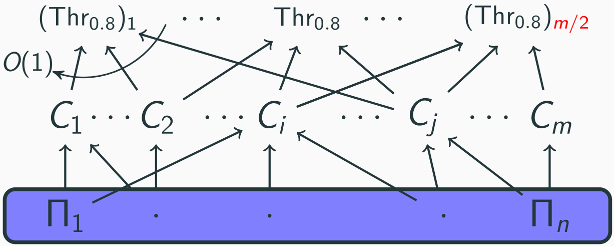
Maintaining fraction of 1's

$\Pi_1 \quad \cdot \quad \cdot \quad \cdot \quad \Pi_n$

Maintaining fraction of 1's

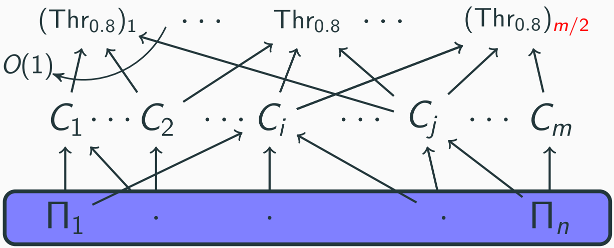


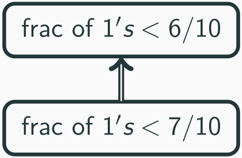
Maintaining fraction of 1's



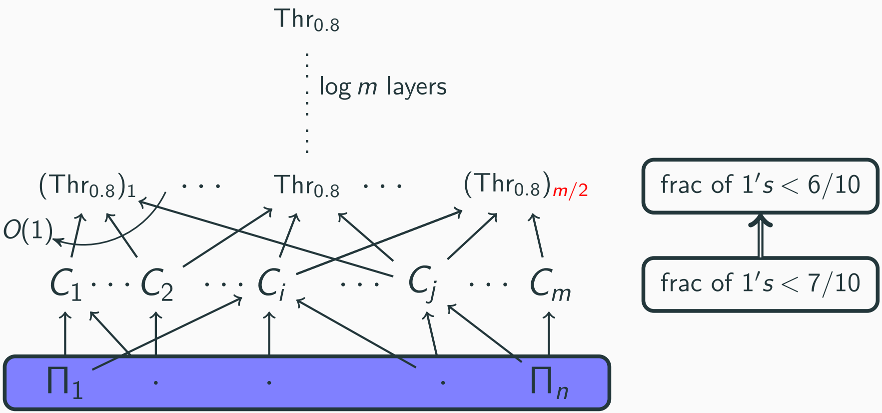
frac of $1's < 7/10$

Maintaining fraction of 1's



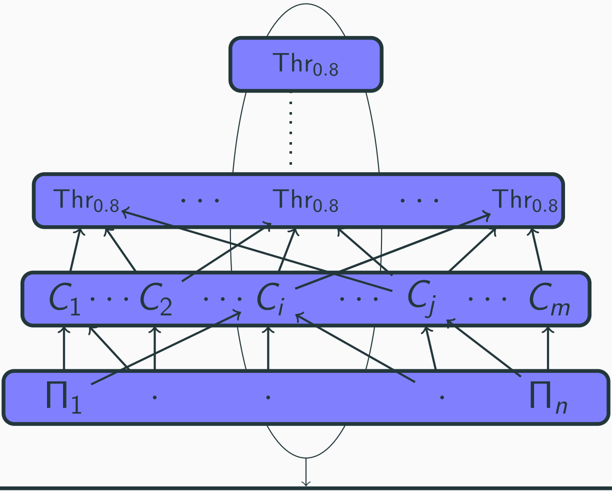


Maintaining fraction of 1's



Final PCP

Final PCP



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

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

Parameters of the Reduction

This gives us a PCP that has the following properties:

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

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: $O(m)$

Main theorem

Main theorem

Theorem

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

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

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

Comparison to Best-Known PCPs

We get the following result for $NTIME[O(n)]$:

Corollary

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

Comparison to Best-Known PCPs

We get the following result for $NTIME[O(n)]$:

Corollary

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

While the current best known linear-sized PCP is:

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

Conclusion

Conclusion

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

Conclusion

- Our results imply that building linear-sized PCPs with minimal queries for NTIME[$O(n)$] and perfect completeness should be nearly as hard (or easy!) as linear-sized PCPs with minimal queries for 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

Open Problems

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

Corollary

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

Open Problems

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

Corollary

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

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

Open Problems

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

Corollary

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

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

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

  This is what one gets using the current PCPs for $\mathrm{NTIME}[O(n)]$.
  Can one prove that,
  $\mathrm{PCP}_{c,s}[\log n + O(1), O(1)] \subseteq \mathrm{PCP}_{1,s'}[\log n + o(\log\log n), O(1)]$?
- 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$(1-\epsilon, 2k/2^k)$ (which is tight up to constant factors).

Thanks! Questions?

References i

M. Bellare, O. Goldreich, and M. Sudan.
Free bits, pcps, and nonapproximability-towards tight results.
SIAM J. Comput., 27(3):804–915, 1998.