The PCP Theorem

Alek, Andrei, Nathan
Mentor: Jonathan

MIT DRP

2023

OUTLINE

- Hardness of approximation
- Statement of theorem
- Constraint satisfaction problems
- PCP proof:
  - Preprocessing
  - Gap Amplification
  - Alphabet reduction
- Proof-checking interpretation of PCP theorem

Approximating 3SAT

Unsatisfiable 3SAT formula:

$$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \overline{x}_3) \wedge (x_1 \vee \overline{x}_2 \vee x_3) \wedge (\overline{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x}_2 \vee \overline{x}_3) \wedge$$

$$(\overline{x}_1 \vee x_2 \vee \overline{x}_3) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee x_3) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee \overline{x}_3) \wedge (x_1 \vee x_2 \vee x_4) \wedge (x_2 \vee \overline{x}_3 \vee \overline{x}_4)$$

Approximating 3SAT

Unsatisfiable 3SAT formula:

$$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \overline{x}_3) \wedge (x_1 \vee \overline{x}_2 \vee x_3) \wedge (\overline{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x}_2 \vee \overline{x}_3) \wedge$$

$$(\overline{x}_1 \vee x_2 \vee \overline{x}_3) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee x_3) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee \overline{x}_3) \wedge (x_1 \vee x_2 \vee x_4) \wedge (x_2 \vee \overline{x}_3 \vee \overline{x}_4)$$

Satisfying assignment for 9/10 clauses:

$$x_1 = \texttt{FALSE}$$

$$x_2 = \texttt{TRUE}$$

$$x_3 = \texttt{TRUE}$$

$$x_4 = \texttt{FALSE}$$

Approximating 3SAT

Another unsatisfiable 3SAT formula:

$$(x_1 \vee x_1 \vee x_1) \wedge (x_2 \vee x_2 \vee x_2) \wedge (x_3 \vee x_3 \vee x_3) \wedge (x_4 \vee x_4 \vee x_4) \wedge (\overline{x}_1 \vee \overline{x}_1 \vee \overline{x}_1) \wedge$$

$$(\overline{x}_2 \vee \overline{x}_2 \vee \overline{x}_2) \wedge (\overline{x}_3 \vee \overline{x}_3 \vee \overline{x}_3) \wedge (\overline{x}_4 \vee \overline{x}_4 \vee \overline{x}_4) \wedge (x_1 \vee x_1 \vee x_1) \wedge (\overline{x}_1 \vee \overline{x}_1 \vee \overline{x}_1)$$

Satisfying assignment for 5/10 clauses:

$$x_1 = \texttt{FALSE}$$

$$x_2 = \texttt{FALSE}$$

$$x_3 = \texttt{TRUE}$$

$$x_4 = \texttt{TRUE}$$

Approximating 3SAT

A 3SAT instance has ***gap*** ε if any assignment violates an ε fraction of constraints.

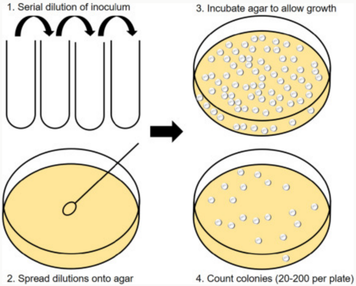
Goal: ***ε-approximate*** 3SAT
i.e. want an algorithm that is

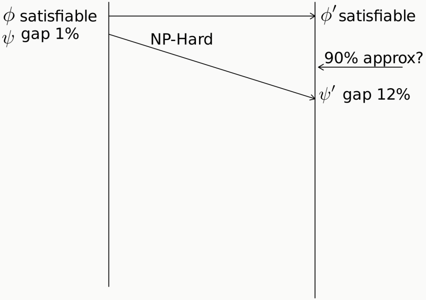
Complete:
`ACCEPT`s satisfiable formulas
Sound:
`REJECT`s formulas with gap $\geq \varepsilon$.

PCP THEOREM

Theorem

It is NP-hard to 90%-approximate 3SAT*, because we can efficiently transform* 3SAT *instances to* 3SAT *instances with gap 12%.*





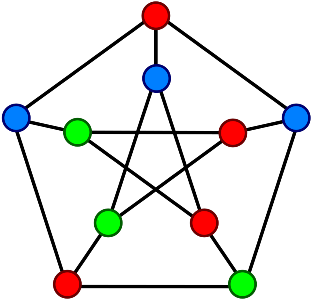
CONSTRAINT SATISFACTION PROBLEMS (qCSP_W)

Definition (qCSP_W)

q-local constraint system over alphabet of size W

Example:

- 3COLOR: 2-local (constraint graph), alphabet $\{R, G, B\}$.
- 3SAT: 3-local, alphabet $\{0, 1\}$.



Proof Outline

small gap → big gap

Lemma (Constraint Expander)

$\mathsf{qCSP}_2 \rightarrow \mathsf{2CSP}_2$ *with constraint graph forming an expander.*
Minor decay of gap and increase in number of constraints.

Lemma (Gap Amplification)

ε*-gap* $\mathsf{2CSP}_2 \rightarrow 6\varepsilon$*-gap* $\mathsf{2CSP}_W$
Increase in alphabet size and increase in number of constraints.

Lemma (Alphabet Reduction)

$\mathsf{2CSP}_W \rightarrow \mathsf{qCSP}_2$
Minor decay of gap and increase in number of constraints.

Constraint Expander

- If a variable occurs in too many constraints we make copies of the variable and add constraints dictating that the copies agree.
- Next, we make the graph d-regular
- Next we add trivial constraints corresponding to self loops and edges of an expander so that the constraint graph becomes an expander

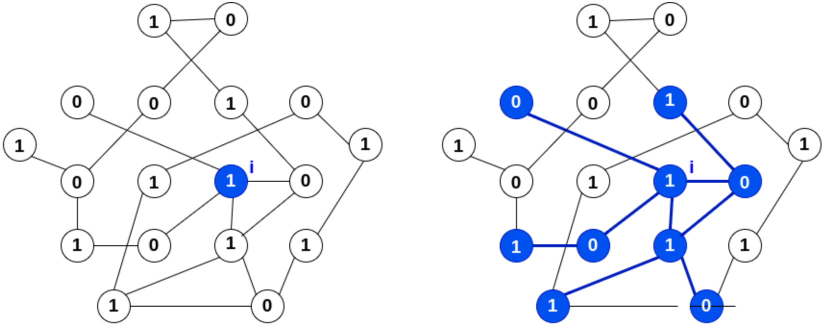
GAP AMPLIFICATION

Ideas:

- Encode many old variables in a single new variable
- Encode many old constraints in a single new constraint
- Ensure that many violated constraints in the old variables correspond to even more violated constraints in the new ones

GAP AMPLIFICATION

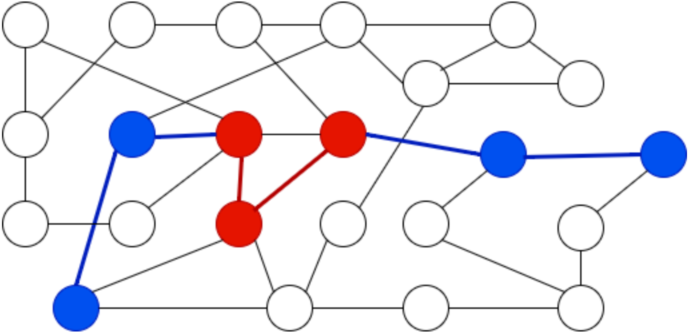
Variables y_i in the new problem encode values for all variables reachable within distance $t + \sqrt{t}$ from i in the original graph.



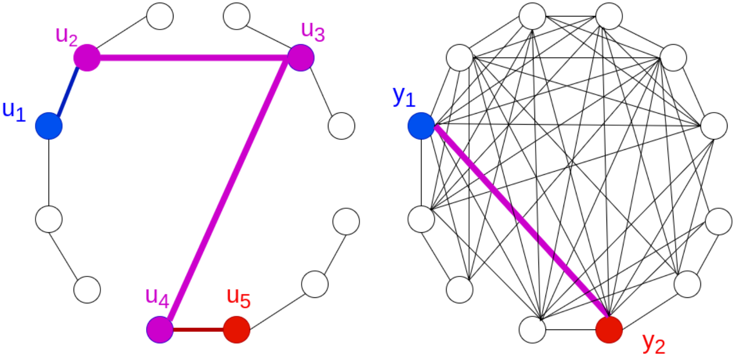
$u_i = 1$ $y_i = 100110101$

Gap Amplification

For every path of length $2t + 2$ we have a constraint in G' between the two endpoints ensuring that all constraints in the overlap are met.



Gap Amplification



u_1 OR u_2

u_2 NAND u_3

u_3 XOR u_4

u_4 NOR u_5

...

$(y_1[2]$ NAND $y_2[3])$ AND $(y_1[4]$ XOR $y_2[1])$

...

GAP AMPLIFICATION

Soundness:
Satisfying assignment in G can be directly translated to satisfying assignment in G'.

Completeness:

- At least ϵ-fraction of the constraints are violated in the original problem
- Want to show 6ϵ-fraction of paths in the new problem contain violated constraints
- Issue: variables in the new problem may not give consistent assignments to the original variables

Gap Amplification

Majority assignments:

- For each old variable, consider the value assigned to it by the majority of the new variables at the end of length-t walks
- Majority assignment violates at least an ϵ fraction of the old constraints
- Denote by S the set of old constraints violated by the majority assignments

Gap Amplification

Bounding expected number of violated constraints:

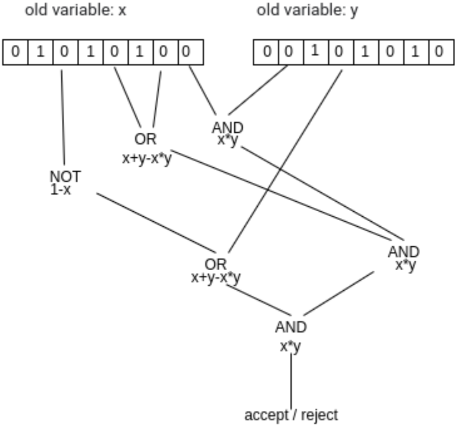
- Consider the $\frac{\sqrt{t}}{100}$ interval in the middle of a random $(2t+2)$-path
- $\left(t + \frac{\sqrt{t}}{100}\right)$-length paths are distributed very similarly to t-length paths
- $\implies$ randomly chosen $(2t+2)$ path contains $\Omega(\epsilon\sqrt{t})$ elements of S in expectation

Gap Amplification

Bounding probability of violated constraint:

- A bound on the probability of a randomly chosen $(2t+2)$-path containing violated old constraints can be obtained from lower bounds on expectation and upper bounds on variance
- We just proved $\Omega(\epsilon\sqrt{t})$ lower bound on expectation
- $O(\epsilon\sqrt{t})$ upper bound on variance comes from expander properties
- $\implies$ randomly chosen new constraint has $\Omega(\epsilon\sqrt{t})$ chance of being violated; choosing large constant t makes this always at least 6ϵ

Alphabet Reduction



Constraints:

$$\sum_{i,j} \alpha_{i,j,k} x_i y_j = b_k$$

$$(x \otimes y)_{i,j} = x_i \cdot y_j$$

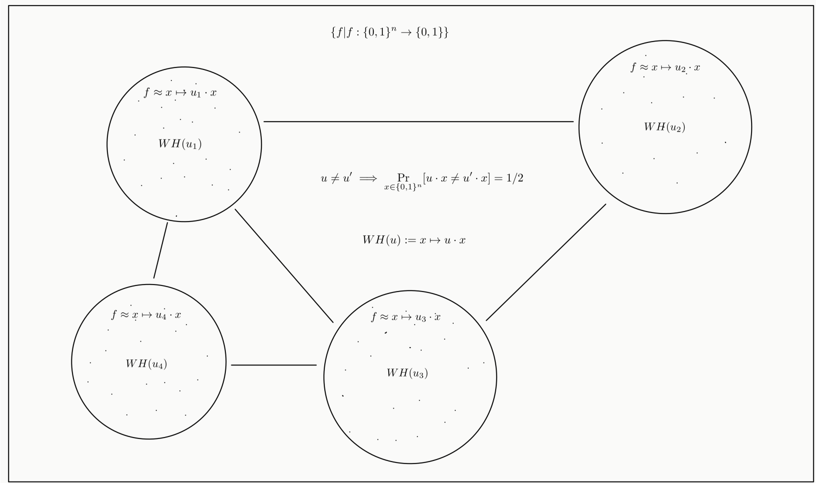
$$A(x \otimes y) = b$$

Alphabet Reduction

- Try 1: make variable for each bit in old variables
  - binary alphabet!
  - not very locally checkable
- Try 2: ***"Walsh Hadamard Code"***
  - $WH(u) = x \mapsto x \cdot u$; write down truth table
  - $|u| = n \implies |WH(u)| = n2^n$
  - $u \neq u'$ not locally checkable: u, u' may only differ on one bit
  - $WH(u) \neq WH(u')$ locally checkable: $WH(u), WH(u')$ differ on $1/2$ of their bits
  - But, we can't efficiently check if a string is a WH-code
- Try 3: Approximately a WH-code
  - easy to check!

Alphabet Reduction

Error correction: if a state is “nearly linear”, it is close to a unique *WH* code, which we can determine easily [BLR]



ALPHABET REDUCTION: PUTTING IT ALL TOGETHER

New Variables: Variable for each bit of

$$WH(u_1), WH(u_2), WH(u_1 \circ u_2), WH((u_1 \circ u_2) \otimes (u_1 \circ u_2))$$

for each old variable u_1, u_2 and each constraint on u_1, u_2.

Soundness: encode old satisfying assignment
Completeness:

1. Check that terms are valid WH-codes (i.e. nearly linear)
2. Check that terms are appropriate concatenations / tensors
3. Check that solution solves the quadratic equations

proof idea: check random subsets

Proof system interpretation of PCP theorem

- ***Proof system***: prover and verifier
- ***Soundness***: there is an honest prover that convinces verifier
- ***Completeness***: no crooked prover can trick verifier

- Probabilistically checkable proof:
- $PCP(r, q) : O(r)$ random bits, access to $O(q)$ bits of proof

$$\mathsf{NP} = \mathsf{PCP}(\log n, 1)$$

Acknowledgements

- Thanks to Irit Dinur for developing the proof we follow here, and for elucidating it in lecture notes
- Thanks to Arora and Barak for clear coverage in their textbook
- Thanks to Jonathan for fantastic mentorship