The complexity of inversion of explicit Goldreich's function by DPLL algorithms

Dmitry Itsykson, Dmitry Sokolov

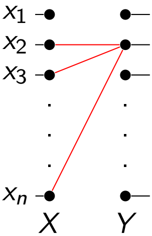
Steklov Institute of Mathematics at St. Petersburg,
Academic University

CSR 2011, Saint-Petersburg
June 15

Goldreich's function

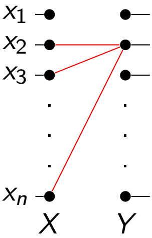
$f : \{0,1\}^n \to \{0,1\}^n$

$f : \{0,1\}^n \to \{0,1\}^n$



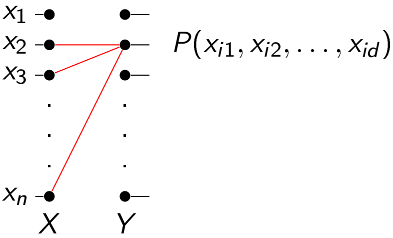
Goldreich's function

$f : \{0,1\}^n \to \{0,1\}^n$



$P(x_{i1}, x_{i2}, \dots, x_{id})$

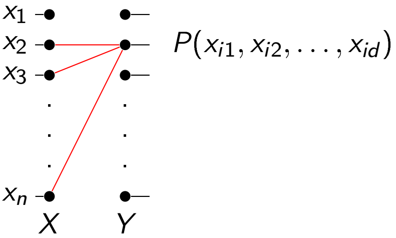
$f : \{0,1\}^n \to \{0,1\}^n$



- $G(X, Y, E)$ is a bipartite graph;

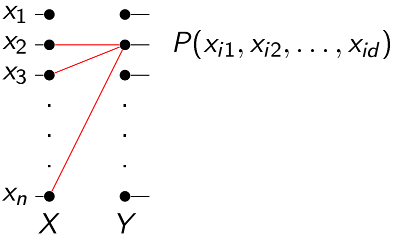
Goldreich's function

$f : \{0,1\}^n \to \{0,1\}^n$



- $G(X, Y, E)$ is a bipartite graph;
- $\forall y \in Y \quad deg(y) = d$

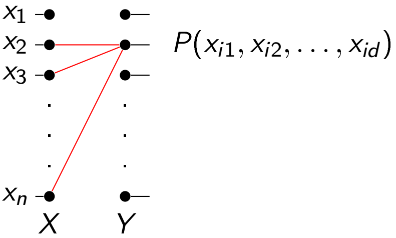
$f : \{0,1\}^n \to \{0,1\}^n$



- $G(X, Y, E)$ is a bipartite graph;
- $\forall y \in Y \quad deg(y) = d$
- d is a constant.

Goldreich's function

$f : \{0,1\}^n \to \{0,1\}^n$



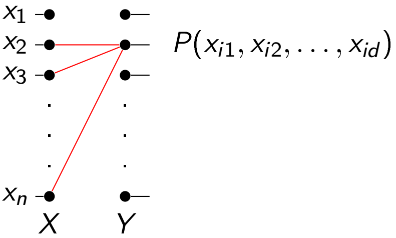
- $G(X, Y, E)$ is a bipartite graph;
- $\forall y \in Y \quad deg(y) = d$
- d is a constant.

Goldreich's conjecture:

- P is a random predicate;
- G is an expander;

then function f is a one-way.

$f : \{0,1\}^n \to \{0,1\}^n$



- $G(X, Y, E)$ is a bipartite graph;
- $\forall y \in Y \quad deg(y) = d$
- d is a constant.

Goldreich's conjecture:

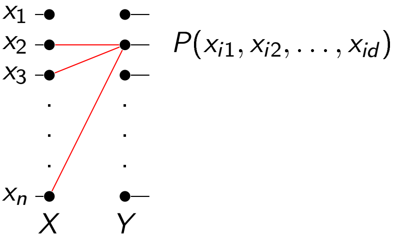
- P is a random predicate;
- G is an expander;

then function f is a one-way.

- f is computed by constant depth circuit;

Goldreich's function

$f : \{0,1\}^n \to \{0,1\}^n$



- $G(X, Y, E)$ is a bipartite graph;
- $\forall y \in Y \quad deg(y) = d$
- d is a constant.

Goldreich's conjecture:

- P is a random predicate;
- G is an expander;

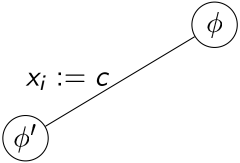
then function f is a one-way.

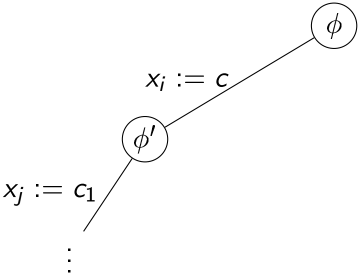
- f is computed by constant depth circuit;
- [Applebaum, Ishai, Kushilevitz 2006] If one-way functions exist then there is a one-way function that can be computed by constant depth circuit.

DPLL algorithms

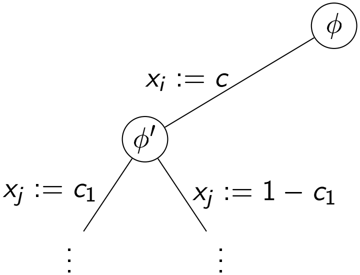
ϕ

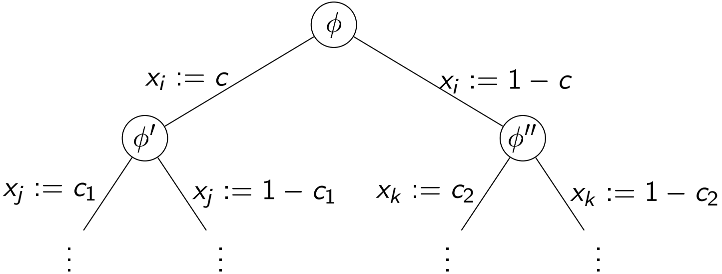
DPLL algorithms



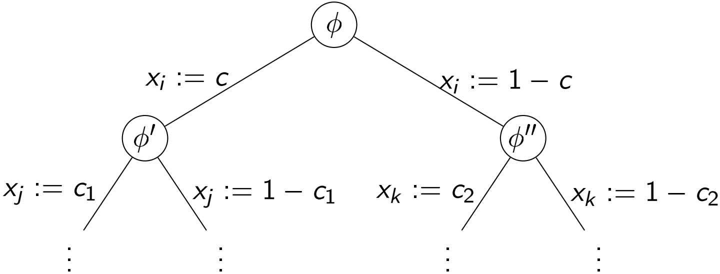

ϕ
$x_i := c$
ϕ'
$x_j := c_1$
$\vdots$

DPLL algorithms



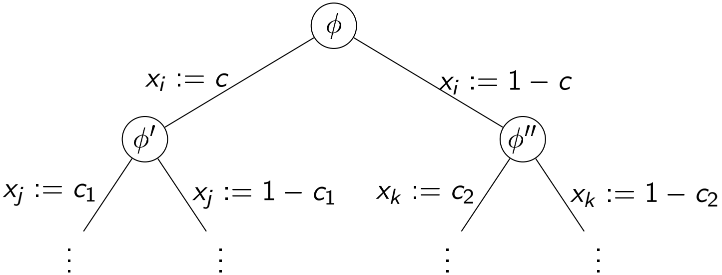

ϕ
$x_i := c$
$x_i := 1 - c$
ϕ'
ϕ''
$x_j := c_1$
$x_j := 1 - c_1$
$x_k := c_2$
$x_k := 1 - c_2$

DPLL algorithms



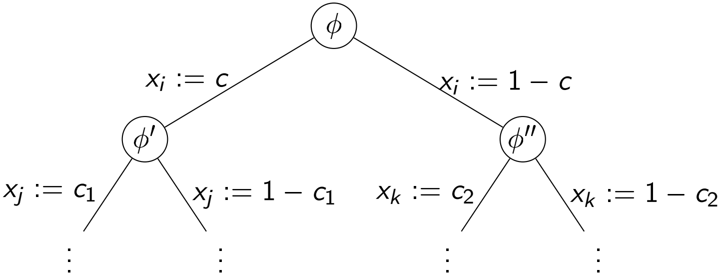
- Heuristic **A** chooses a variable for splitting.

DPLL algorithms



- Heuristic **A** chooses a variable for splitting.
- Heuristic **B** chooses first value.

DPLL algorithms



- Heuristic **A** chooses a variable for splitting.
- Heuristic **B** chooses first value.
- Simplification rules:
  - unit clause elimination;
  - pure literal rule.

Lower bounds for DPLL algorithms

Lower bounds for DPLL algorithms

- Unsatisfiable formulas
  - Exponential lower bounds for resolution refutations of unsatisfiable formulas translate to backtracking algorithms.
  - [Tseitin, 1968] ... [Pudlak, Implagliazzo, 2000].

Lower bounds for DPLL algorithms

- Unsatisfiable formulas
  - Exponential lower bounds for resolution refutations of unsatisfiable formulas translate to backtracking algorithms.
  - [Tseitin, 1968] ... [Pudlak, Implagliazzo, 2000].
- Satisfiable formulas
  - If $\mathbf{P} = \mathbf{NP}$ then there are no superpolynomial lower bounds for backtracking algorithms since heuristic **B** may choose correct value.

- Unsatisfiable formulas
  - Exponential lower bounds for resolution refutations of unsatisfiable formulas translate to backtracking algorithms.
  - [Tseitin, 1968] ... [Pudlak, Implagliazzo, 2000].
- Satisfiable formulas
  - If $\mathbf{P} = \mathbf{NP}$ then there are no superpolynomial lower bounds for backtracking algorithms since heuristic **B** may choose correct value.
  - Inverting of functions corresponds to satisfiable formulas.

Lower bounds for DPLL algorithms

- Unsatisfiable formulas
  - Exponential lower bounds for resolution refutations of unsatisfiable formulas translate to backtracking algorithms.
  - [Tseitin, 1968] ... [Pudlak, Implagliazzo, 2000].
- Satisfiable formulas
  - If $\mathbf{P} = \mathbf{NP}$ then there are no superpolynomial lower bounds for backtracking algorithms since heuristic **B** may choose correct value.
  - Inverting of functions corresponds to satisfiable formulas.
  - [Nikolenko 2002], [Achilioptas, Beame, Molloy 2003-2004] exponential lower bound for specific backtracking algorithms.
  - [Alekhnovich, Hirsch, Itsykson 2005] Exponential lower bound for myopic and drunken algorithms.

- Unsatisfiable formulas
  - Exponential lower bounds for resolution refutations of unsatisfiable formulas translate to backtracking algorithms.
  - [Tseitin, 1968] ... [Pudlak, Implagliazzo, 2000].
- Satisfiable formulas
  - If $\mathbf{P} = \mathbf{NP}$ then there are no superpolynomial lower bounds for backtracking algorithms since heuristic **B** may choose correct value.
  - Inverting of functions corresponds to satisfiable formulas.
  - [Nikolenko 2002], [Achilioptas, Beame, Molloy 2003-2004] exponential lower bound for specific backtracking algorithms.
  - [Alekhnovich, Hirsch, Itsykson 2005] Exponential lower bound for myopic and drunken algorithms.
  - Exponential lower bound for inversion of Goldreich's function by myopic [J. Cook et al. 2009] and drunken [Itsykson 2010] algorithms.

Drunken and myopic algorithms

Drunken and myopic algorithms

Definition

Drunken algorithms:

- **A**: any;
- **B**: random $50:50$.

Drunken and myopic algorithms

Definition

Drunken algorithms:

- **A**: any;
- **B**: random $50 : 50$.

Definition

Myopic heuristic:

Drunken and myopic algorithms

Definition

Drunken algorithms:

- **A**: any;
- **B**: random $50 : 50$.

Definition

Myopic heuristic:

- sees structure of the formula;

Drunken and myopic algorithms

Definition

Drunken algorithms:

- **A**: any;
- **B**: random $50 : 50$.

Definition

Myopic heuristic:

- sees structure of the formula;
- doesn't see negation signs;

Drunken and myopic algorithms

Definition

Drunken algorithms:

- **A**: any;
- **B**: random $50 : 50$.

Definition

Myopic heuristic:

- sees structure of the formula;
- doesn't see negation signs;
- requests negations in $K = n^{1-\epsilon}$ clause.

$(x_1 \vee x_3 \vee x_5)$
$(x_2 \vee x_3)$
$(x_2 \vee x_4 \vee x_5)$
$(x_1 \vee x_4 \vee x_6)$

Definition

Drunken algorithms:

- **A**: any;
- **B**: random 50 : 50.

Definition

Myopic heuristic:

- sees structure of the formula;
- doesn't see negation signs;
- requests negations in $K = n^{1-\epsilon}$ clause.

$(x_1 \vee x_3 \vee x_5)$
$(x_2 \vee x_3)$
$(x_2 \vee x_4 \vee x_5)$
$(x_1 \vee x_4 \vee x_6)$

Drunken and myopic algorithms

Definition

Drunken algorithms:

- **A**: any;
- **B**: random 50 : 50.

Definition

Myopic heuristic:

- sees structure of the formula;
- doesn't see negation signs;
- requests negations in $K = n^{1-\epsilon}$ clause.

$(x_1 \vee x_3 \vee x_5)$
$(x_2 \vee x_3)$
$(x_2 \vee x_4 \vee x_5)$
$(x_1 \vee x_4 \vee x_6)$

Definition

Drunken algorithms:

- **A**: any;
- **B**: random 50 : 50.

Definition

Myopic heuristic:

- sees structure of the formula;
- doesn't see negation signs;
- requests negations in $K = n^{1-\epsilon}$ clause.

$$\begin{array}{lcl} (x_1 \vee x_3 \vee x_5) & & (x_1 \vee x_3 \vee x_5) \\ \color{red}{(x_2 \vee x_3)} & & (x_2 \vee \color{red}{\neg} x_3) \\ (x_2 \vee x_4 \vee x_5) & \Rightarrow & (x_2 \vee x_4 \vee x_5) \\ \color{red}{(x_1 \vee x_4 \vee x_6)} & & (x_1 \vee \color{red}{\neg} x_4 \vee x_6) \end{array}$$

Myopic algorithms

Myopic algorithms

Definition

Myopic algorithm:

- $\mathbf{A}, \mathbf{B}$ are myopic heuristics.

Myopic algorithms

Definition

Myopic algorithm:

- $\mathbf{A}, \mathbf{B}$ are myopic heuristics.

- [Alekhnovich, Hirsch, Itsykson 2005]
  - P is a linear predicate.
  - G is a random graph.

Myopic algorithms

Definition

Myopic algorithm:

- $\mathbf{A}, \mathbf{B}$ are myopic heuristics.

- [Alekhnovich, Hirsch, Itsykson 2005]
  - P is a linear predicate.
  - G is a random graph.
- [J. Cook, Etesami, Miller, Trevisan 2009]

Myopic algorithms

Definition

Myopic algorithm:

- $\mathbf{A}, \mathbf{B}$ are myopic heuristics.

- [Alekhnovich, Hirsch, Itsykson 2005]
  - P is a linear predicate.
  - G is a random graph.
- [J. Cook, Etesami, Miller, Trevisan 2009]
  - $P = x_1 + x_2 + \cdots + x_{d-2} + x_{d-1}x_d$.
  - In fact: $P = x_1 + x_2 + \cdots + x_{d-k} + Q(x_{d-k+1}, \ldots, x_d)$.

Myopic algorithms

Definition

Myopic algorithm:

- **A**, **B** are myopic heuristics.

- [Alekhnovich, Hirsch, Itsykson 2005]
  - P is a linear predicate.
  - G is a random graph.
- [J. Cook, Etesami, Miller, Trevisan 2009]
  - $P = x_1 + x_2 + \cdots + x_{d-2} + x_{d-1}x_d$.
  - In fact: $P = x_1 + x_2 + \cdots + x_{d-k} + Q(x_{d-k+1}, \ldots, x_d)$.

Disadvantages:

Myopic algorithms

Definition

Myopic algorithm:

- **A**, **B** are myopic heuristics.

- [Alekhnovich, Hirsch, Itsykson 2005]
  - P is a linear predicate.
  - G is a random graph.
- [J. Cook, Etesami, Miller, Trevisan 2009]
  - $P = x_1 + x_2 + \cdots + x_{d-2} + x_{d-1}x_d$.
  - In fact: $P = x_1 + x_2 + \cdots + x_{d-k} + Q(x_{d-k+1}, \ldots, x_d)$.

Disadvantages:

- G is a random graph.

Myopic algorithms

Definition

Myopic algorithm:

- **A**, **B** are myopic heuristics.

- [Alekhnovich, Hirsch, Itsykson 2005]
  - P is a linear predicate.
  - G is a random graph.
- [J. Cook, Etesami, Miller, Trevisan 2009]
  - $P = x_1 + x_2 + \cdots + x_{d-2} + x_{d-1}x_d$.
  - In fact: $P = x_1 + x_2 + \cdots + x_{d-k} + Q(x_{d-k+1}, \ldots, x_d)$.

Disadvantages:

- G is a random graph.
- K is a constant.

Myopic algorithms

Definition

Myopic algorithm:

- $\mathbf{A}, \mathbf{B}$ are myopic heuristics.

- [Alekhnovich, Hirsch, Itsykson 2005]
  - P is a linear predicate.
  - G is a random graph.
- [J. Cook, Etesami, Miller, Trevisan 2009]
  - $P = x_1 + x_2 + \cdots + x_{d-2} + x_{d-1}x_d$.
  - In fact: $P = x_1 + x_2 + \cdots + x_{d-k} + Q(x_{d-k+1}, \ldots, x_d)$.

Disadvantages:

- G is a random graph.
- K is a constant.
- Too complicated proof.

In our work:

Definition

Myopic algorithm:

- **A**, **B** are myopic heuristics.

- [Alekhnovich, Hirsch, Itsykson 2005]
  - P is a linear predicate.
  - G is a random graph.
- [J. Cook, Etesami, Miller, Trevisan 2009]
  - $P = x_1 + x_2 + \dots + x_{d-2} + x_{d-1}x_d$.
  - In fact: $P = x_1 + x_2 + \dots + x_{d-k} + Q(x_{d-k+1}, \dots, x_d)$.

Disadvantages:

- G is a random graph.
- K is a constant.
- Too complicated proof.

In our work:

- G is based on expander.

Definition

Myopic algorithm:

- $\mathbf{A}, \mathbf{B}$ are myopic heuristics.

- [Alekhnovich, Hirsch, Itsykson 2005]
  - P is a linear predicate.
  - G is a random graph.
- [J. Cook, Etesami, Miller, Trevisan 2009]
  - $P = x_1 + x_2 + \cdots + x_{d-2} + x_{d-1}x_d$.
  - In fact: $P = x_1 + x_2 + \cdots + x_{d-k} + Q(x_{d-k+1}, \ldots, x_d)$.

Disadvantages:

- G is a random graph.
- K is a constant.
- Too complicated proof.

In our work:

- G is based on expander.
- $K = n^{1-\epsilon}$.

Definition

Myopic algorithm:

- $\mathbf{A}, \mathbf{B}$ are myopic heuristics.

- [Alekhnovich, Hirsch, Itsykson 2005]
  - P is a linear predicate.
  - G is a random graph.
- [J. Cook, Etesami, Miller, Trevisan 2009]
  - $P = x_1 + x_2 + \cdots + x_{d-2} + x_{d-1}x_d$.
  - In fact: $P = x_1 + x_2 + \cdots + x_{d-k} + Q(x_{d-k+1}, \ldots, x_d)$.

Disadvantages:

- G is a random graph.
- K is a constant.
- Too complicated proof.

In our work:

- G is based on expander.
- $K = n^{1-\epsilon}$.
- "Simple" proof.

Our results

$P(x_1, \ldots, x_d) = x_1 \oplus x_2 \oplus \ldots \oplus x_{d-k} \oplus Q(x_{d-k+1}, \ldots, x_d)$, Q is an arbitrary, $k < d/4$.

Our results

$P(x_1, \ldots, x_d) = x_1 \oplus x_2 \oplus \ldots \oplus x_{d-k} \oplus Q(x_{d-k+1}, \ldots, x_d)$, Q is an arbitrary, $k < d/4$.

Theorem

There exists an explicit graph G such that every myopic or drunken DPLL algorithm makes at least $2^{n^{\Omega(1)}}$ steps on "$f(x) = f(a)$" for almost all $a \in \{0, 1\}^n$.

Our results

$P(x_1, \ldots, x_d) = x_1 \oplus x_2 \oplus \ldots \oplus x_{d-k} \oplus Q(x_{d-k+1}, \ldots, x_d)$, Q is an arbitrary, $k < d/4$.

Theorem

There exists an explicit graph G such that every myopic or drunken DPLL algorithm makes at least $2^{n^{\Omega(1)}}$ steps on "$f(x) = f(a)$" for almost all $a \in \{0,1\}^n$.

- G is based on expander instead of random graph.

Our results

$P(x_1, \ldots, x_d) = x_1 \oplus x_2 \oplus \ldots \oplus x_{d-k} \oplus Q(x_{d-k+1}, \ldots, x_d)$, Q is an arbitrary, $k < d/4$.

Theorem

There exists an explicit graph G such that every myopic or drunken DPLL algorithm makes at least $2^{n^{\Omega(1)}}$ steps on "$f(x) = f(a)$" for almost all $a \in \{0,1\}^n$.

- G is based on expander instead of random graph.
- For drunken algorithms the proof follows [Itsykson, CSR-2010]

Our results

$P(x_1, \ldots, x_d) = x_1 \oplus x_2 \oplus \ldots \oplus x_{d-k} \oplus Q(x_{d-k+1}, \ldots, x_d)$, Q is an arbitrary, $k < d/4$.

Theorem

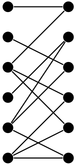
There exists an explicit graph G such that every myopic or drunken DPLL algorithm makes at least $2^{n^{\Omega(1)}}$ steps on "$f(x) = f(a)$" for almost all $a \in \{0,1\}^n$.

- G is based on expander instead of random graph.
- For drunken algorithms the proof follows [Itsykson, CSR-2010]
- For myopic
  - we simplify previous proof and
  - $K = n^{1-\epsilon}$

Graph construction

$$P(x_1, \ldots, x_d) = x_1 \oplus x_2 \oplus \ldots \oplus x_{d-k} \oplus Q(x_{d-k+1}, \ldots, x_d)$$

$$P(x_1, \ldots, x_d) = x_1 \oplus x_2 \oplus \ldots \oplus x_{d-k} \oplus Q(x_{d-k+1}, \ldots, x_d)$$

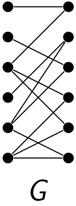


G

- G is an expander;

Graph construction

$$P(x_1, \ldots, x_d) = x_1 \oplus x_2 \oplus \ldots \oplus x_{d-k} \oplus Q(x_{d-k+1}, \ldots, x_d)$$

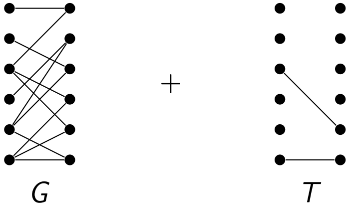


$+$

- G is an expander;

Graph construction

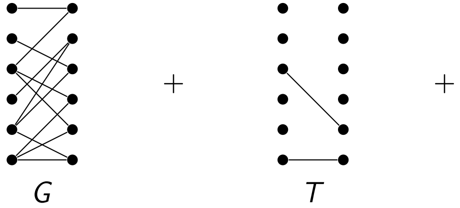
$$P(x_1, \ldots, x_d) = x_1 \oplus x_2 \oplus \ldots \oplus x_{d-k} \oplus Q(x_{d-k+1}, \ldots, x_d)$$



- G is an expander;
- $G + T$ has full rank. $\forall y \in Y \subset T,\ deg(y) = 1$;

Graph construction

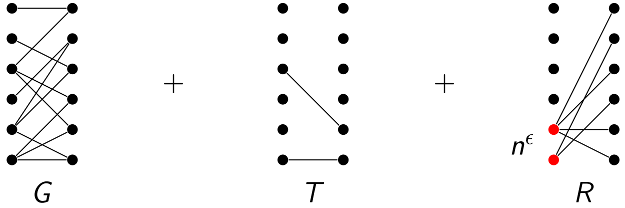
$$P(x_1, \dots, x_d) = {\color{red} x_1 \oplus x_2 \oplus \dots \oplus x_{d-k}} \oplus Q(x_{d-k+1}, \dots, x_d)$$



- G is an expander;
- $G + T$ has full rank. $\forall y \in Y \subset T,\ deg(y) = 1$;
  - $G + T$ is an expander.

Graph construction

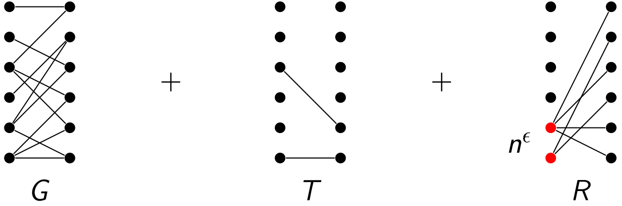
$$P(x_1, \ldots, x_d) = x_1 \oplus x_2 \oplus \ldots \oplus x_{d-k} \oplus {\color{red} Q(x_{d-k+1}, \ldots, x_d)}$$



- G is an expander;
- $G + T$ has full rank. $\forall y \in Y \subset T,\ deg(y) = 1$;
  - $G + T$ is an expander.
- R contains nonlinear edges. $|\{x \mid x \in X,\ deg(x) \neq 0\}| \leq n^\epsilon$.
  - $G + T + R$ is an expander.

Graph construction

$$P(x_1, \dots, x_d) = x_1 \oplus x_2 \oplus \dots \oplus x_{d-k} \oplus \color{red}{Q(x_{d-k+1}, \dots, x_d)}$$



- G is an expander;
- $G + T$ has full rank. $\forall y \in Y \subset T,\ deg(y) = 1$;
  - $G + T$ is an expander.
- R contains nonlinear edges. $|\{x \mid x \in X,\ deg(x) \neq 0\}| \leq n^\epsilon$.
  - $G + T + R$ is an expander.
  - Size of preimages no more than 2^{n^ϵ}.

We can invert $f_{G+T+R,P}$ in time $poly(n)2^{n^\epsilon}$, but this is still much!

Plan of the proof

Plan of the proof

- Lower bounds for unsatisfiable formulas.

Plan of the proof

- Lower bounds for unsatisfiable formulas.
  - G is an expander.
  - P is almost linear.
  - Lower bounds for resolution proofs

Plan of the proof

- Lower bounds for unsatisfiable formulas.
  - G is an expander.
  - P is almost linear.
  - Lower bounds for resolution proofs
- With probability $1 - 2^{-\Omega(n)}$ after several steps current formula becomes unsatisfiable.

Plan of the proof

- Lower bounds for unsatisfiable formulas.
  - G is an expander.
  - P is almost linear.
  - Lower bounds for resolution proofs
- With probability $1 - 2^{-\Omega(n)}$ after several steps current formula becomes unsatisfiable.
  - G is an expander.
  - f is almost bijection.
  - Myopic algorithm can't recognize different absolute terms.