Testing Monotonicity

Alexander Belov
CWI

Eric Blais
University of Waterloo

15-Apr-2016, Cambridge-UK

Introduction

Birds can fly

Birds can fly

Birds can fly

Birds can fly



Birds can fly

Birds can fly





Birds can fly





Birds can fly






Birds can fly





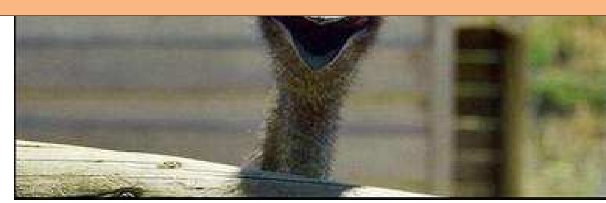


Birds can fly





- Accept hypothesis that are true;
- Reject hypothesis that are too wrong:
  Fail on too many occasions.


Formal Definition

- Population: Boolean function $f\colon \{0,1\}^n \to \{0,1\}$.

Inputs	$\longleftrightarrow$	Objects
Variables	$\longleftrightarrow$	Parameters
Value of the function	$\longleftrightarrow$	Property under investigation

- Hypothesis: The function f possesses some property $\mathcal{P}$.

- Population: Boolean function $f\colon \{0,1\}^n \to \{0,1\}$.

Inputs	$\longleftrightarrow$	Objects
Variables	$\longleftrightarrow$	Parameters
Value of the function	$\longleftrightarrow$	Property under investigation

- Hypothesis: The function f possesses some property $\mathcal{P}$.

Given $\mathcal{P}$, construct an algorithm that

- Accepts if $f \in \mathcal{P}$.
- Rejects if f is far from $\mathcal{P}$:

  for any $g \in \mathcal{P}$, relative Hamming distance $h(f,g) \geq \varepsilon$:

  f and g differ on $\geq \varepsilon 2^n$ inputs.

- Population: Boolean function $f\colon \{0,1\}^n \to \{0,1\}$.

Inputs	$\longleftrightarrow$	Objects
Variables	$\longleftrightarrow$	Parameters
Value of the function	$\longleftrightarrow$	Property under investigation

- Hypothesis: The function f possesses some property $\mathcal{P}$.

Given $\mathcal{P}$, construct an algorithm that

- Accepts if $f \in \mathcal{P}$.
- Rejects if f is far from $\mathcal{P}$:

  for any $g \in \mathcal{P}$, relative Hamming distance $h(f,g) \geq \varepsilon$:

  f and g differ on $\geq \varepsilon 2^n$ inputs.

Popular $\mathcal{P}$s:

- Juntas: the function depends on few variables.
- Monotonicity: improving a parameter improves the property.

- Population: Boolean function $f\colon \{0,1\}^n \to \{0,1\}$.

Inputs	$\longleftrightarrow$	Objects
Variables	$\longleftrightarrow$	Parameters
Value of the function	$\longleftrightarrow$	Property under investigation

- Hypothesis: The function f possesses some property $\mathcal{P}$.

Given $\mathcal{P}$, construct an algorithm that

- Accepts if $f \in \mathcal{P}$.
- Rejects if f is far from $\mathcal{P}$:

  for any $g \in \mathcal{P}$, relative Hamming distance $h(f,g) \geq \varepsilon$:

  f and g differ on $\geq \varepsilon 2^n$ inputs.

Popular $\mathcal{P}$s:

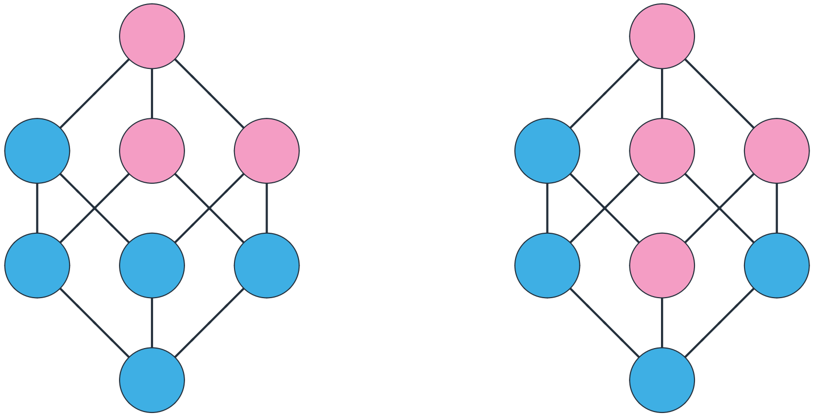
- Juntas: the function depends on few variables.
- Monotonicity: improving a parameter improves the property.

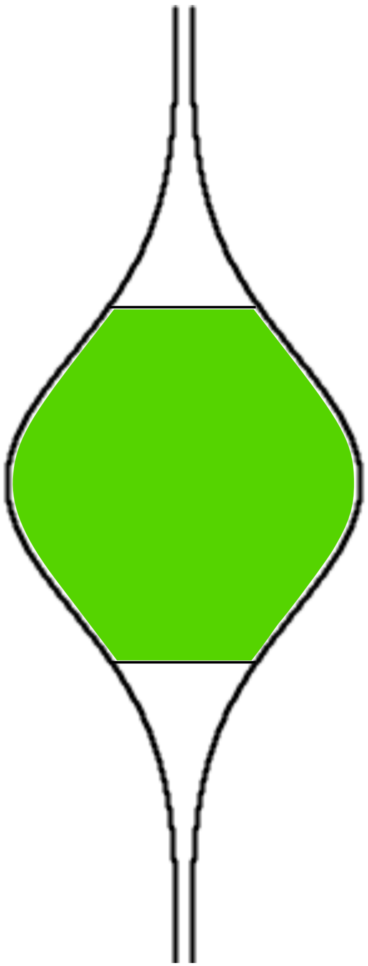
Partial order on Boolean strings:

$$\begin{array}{rcl} x & \preceq & y: \qquad \forall i \in [n]: x_i = 1 \implies y_i = 1. \\ 0010 & \preceq & 0111 \end{array}$$

A function $f\colon \{0,1\}^n \to \{0,1\}$ is monotone iff

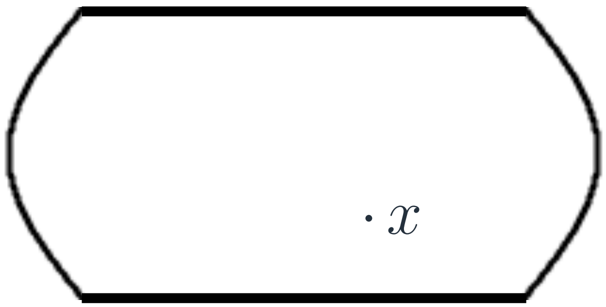
$$\forall x, y \in \{0,1\}^n: \quad x \preceq y \implies f(x) \leq f(y).$$



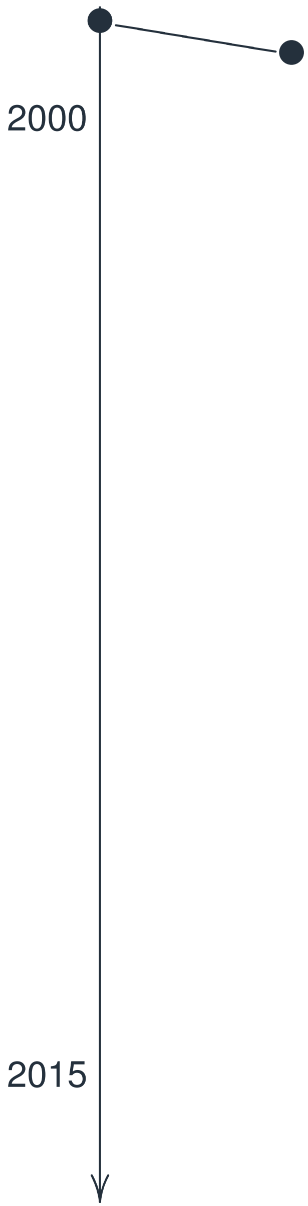


- Monotone $=$ Monotonely non-decreasing
- Interested in query complexity
  - as a function of n and ε;
  - dependence on n more important;
  - the size of the input $N = 2^n$.
- Restrict to the inputs in the middle of the cube:

$$|x| = \frac{n}{2} \pm O_\varepsilon(\sqrt{n})$$



Time Line



[Goldreich, Goldwasser, Lehman, Ron ’1998]

Defined the problem, proposed a simple edge tester.

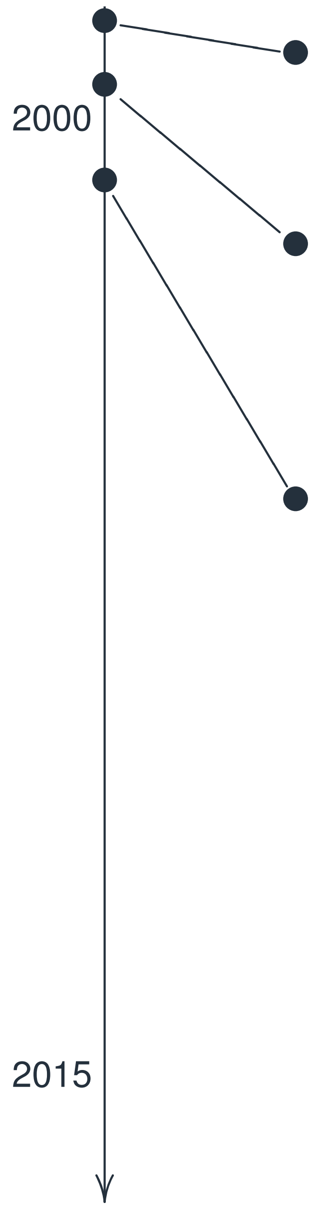
Time Line

2000

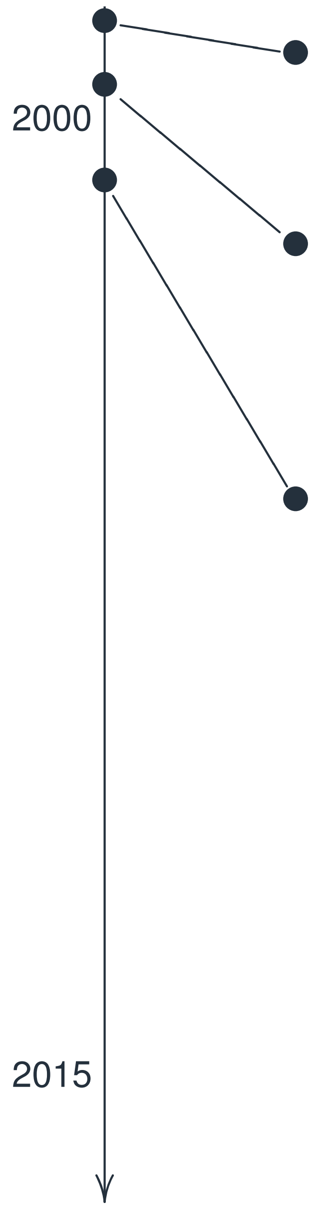
- [Goldreich, Goldwasser, Lehman, Ron '1998]
  Defined the problem, proposed a simple edge tester.
- [Dodis, Goldreich, Lehman, Raskhodnikova, Ron, Samorodnitsky '1999]
  Analysed the edge tester. $O(n/\varepsilon)$ upper bound.

2015

Time Line



- [Goldreich, Goldwasser, Lehman, Ron '1998]
  Defined the problem, proposed a simple edge tester.
- [Dodis, Goldreich, Lehman, Raskhodnikova, Ron, Samorodnitsky '1999]
  Analysed the edge tester. $O(n/\varepsilon)$ upper bound.
- [Fischer, Lehman, Newman, Raskhodnikova, Rubinfeld, Samorodnitsky '2002]
  $\Omega(\log n)$ lower bound for non-adaptive algorithms;



- [Goldreich, Goldwasser, Lehman, Ron '1998]
  Defined the problem, proposed a simple edge tester.
- [Dodis, Goldreich, Lehman, Raskhodnikova, Ron, Samorodnitsky '1999]
  Analysed the edge tester. $O(n/\varepsilon)$ upper bound.
- [Fischer, Lehman, Newman, Raskhodnikova, Rubinfeld, Samorodnitsky '2002]
  $\Omega(\log n)$ lower bound for non-adaptive algorithms;
  $\Omega(\log \log n)$ general lower bound;

Time Line

- [Goldreich, Goldwasser, Lehman, Ron '1998]
  Defined the problem, proposed a simple edge tester.
- [Dodis, Goldreich, Lehman, Raskhodnikova, Ron, Samorodnitsky '1999]
  Analysed the edge tester. $O(n/\varepsilon)$ upper bound.
- [Fischer, Lehman, Newman, Raskhodnikova, Rubinfeld, Samorodnitsky '2002]
  $\Omega(\log n)$ lower bound for non-adaptive algorithms;
  $\Omega(\log \log n)$ general lower bound;
  $\Omega(\log \log \log N)$ lower bound, N: size of the input.

Time Line

- [Goldreich, Goldwasser, Lehman, Ron '1998]
  Defined the problem, proposed a simple edge tester.
- [Dodis, Goldreich, Lehman, Raskhodnikova, Ron, Samorodnitsky '1999]
  Analysed the edge tester. $O(n/\varepsilon)$ upper bound.
- [Fischer, Lehman, Newman, Raskhodnikova, Rubinfeld, Samorodnitsky '2002]
  $\Omega(\log n)$ lower bound for non-adaptive algorithms;
  $\Omega(\log \log n)$ general lower bound;
  $\Omega(\log \log \log N)$ lower bound, N: size of the input.

New algorithms
and lower bounds
in expectation...

2000

- [Goldreich, Goldwasser, Lehman, Ron '1998]
  Defined the problem, proposed a simple edge tester.
- [Dodis, Goldreich, Lehman, Raskhodnikova, Ron, Samorodnitsky '1999]
  Analysed the edge tester. $O(n/\varepsilon)$ upper bound.
- [Fischer, Lehman, Newman, Raskhodnikova, Rubinfeld, Samorodnitsky '2002]
  $\Omega(\log n)$ lower bound for non-adaptive algorithms;
  $\Omega(\log \log n)$ general lower bound;
  $\Omega(\log \log \log N)$ lower bound, N: size of the input.

New algorithms
and lower bounds
in expectation...

2015

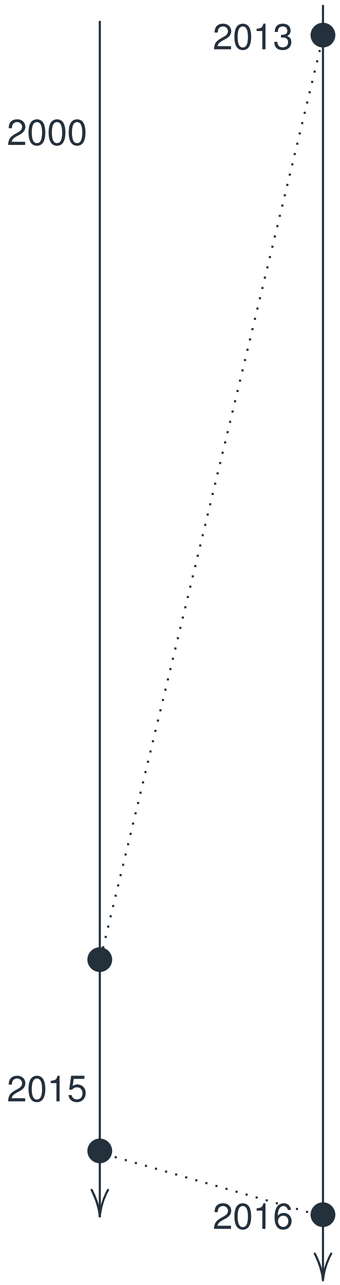
2000

- [Goldreich, Goldwasser, Lehman, Ron '1998]
  Defined the problem, proposed a simple edge tester.
- [Dodis, Goldreich, Lehman, Raskhodnikova, Ron, Samorodnitsky '1999]
  Analysed the edge tester. $O(n/\varepsilon)$ upper bound.
- [Fischer, Lehman, Newman, Raskhodnikova, Rubinfeld, Samorodnitsky '2002]
  $\Omega(\log n)$ lower bound for non-adaptive algorithms;
  $\Omega(\log \log n)$ general lower bound;
  $\Omega(\log \log \log N)$ lower bound, N: size of the input.

2015

- New algorithms and lower bounds in expectation...

Time Line: Zoom In



Time Line: Zoom In

STOC'13

- [Chakrabarty, Seshadhri]
  Proposed a path tester. Complexity: $\widetilde{O}(n^{7/8}\varepsilon^{-3/2})$.

FOCS'14

- [Chen, Servedio, Tan]
  $\widetilde{\Omega}(n^{1/5})$ non-adaptive lower bound,
  improved the upper bound to $\widetilde{O}(n^{5/6}\varepsilon^{-4})$.

STOC'15

- [Chen, De, Servedio, Tan]
  Improved the non-adaptive lower bound to $n^{1/2-o(1)}$.

FOCS'15

- [Khot, Minzer, Safra]
  Brought complexity of path tester to $\widetilde{O}\left(\frac{\sqrt{n}}{\varepsilon^2}\right)$.

Time Line: Zoom In

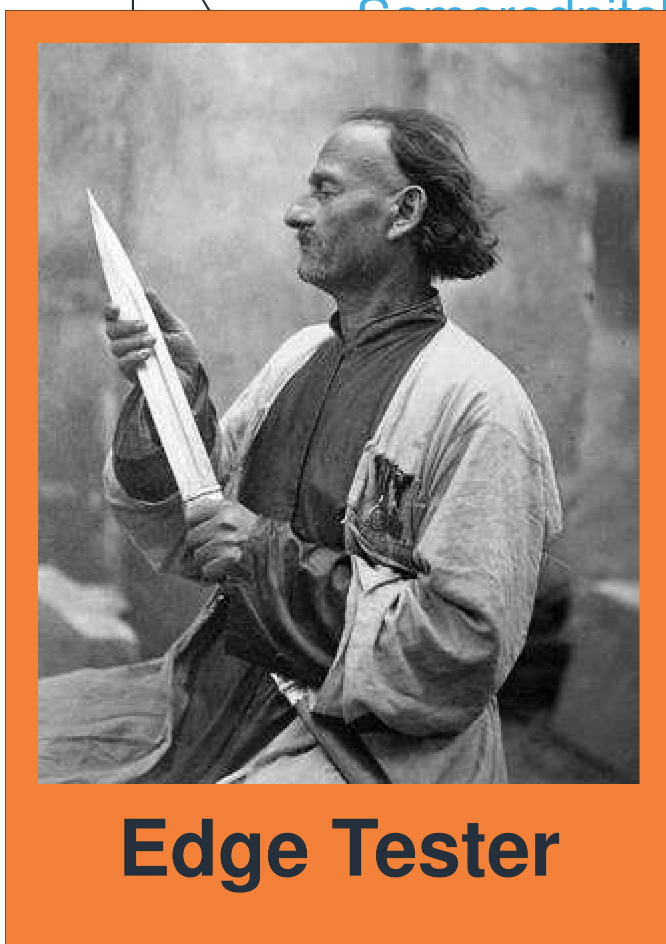
- STOC'13 — [Chakrabarty, Seshadhri]
  Proposed a path tester. Complexity: $\widetilde{O}(n^{7/8}\varepsilon^{-3/2})$.
- FOCS'14 — [Chen, Servedio, Tan]
  $\widetilde{\Omega}(n^{1/5})$ non-adaptive lower bound,
  improved the upper bound to $\widetilde{O}(n^{5/6}\varepsilon^{-4})$.
- STOC'15 — [Chen, De, Servedio, Tan]
  Improved the non-adaptive lower bound to $n^{1/2-o(1)}$.
- FOCS'15 — [Khot, Minzer, Safra]
  Brought complexity of path tester to $\widetilde{O}\left(\frac{\sqrt{n}}{\varepsilon^2}\right)$.
- STOC'16 — [Belovs, Blais]
  $\widetilde{\Omega}(n^{1/4})$ adaptive lower bound.

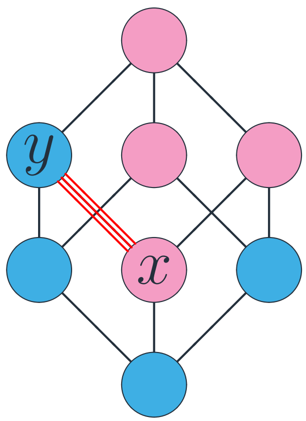
Non-Adaptive Algorithms

Edge Tester

2000

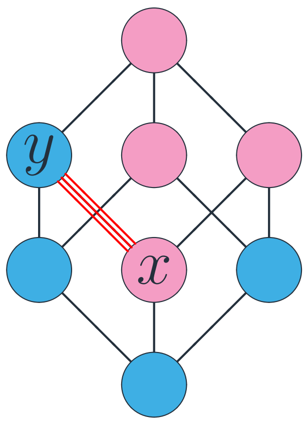
- [Goldreich, Goldwasser, Lehman, Ron '1998]
  Defined the problem, proposed a simple edge tester.
- [Dodis, Goldreich, Lehman, Raskhodnikova, Ron, Samorodnitsky '1999]
  edge tester. $O(n/\varepsilon)$ upper bound.
- man, Newman, Raskhodnikova,
  morodnitsky '2002]
  er bound for non-adaptive algorithms;
  general lower bound;
  $N)$ lower bound, N: size of the input.



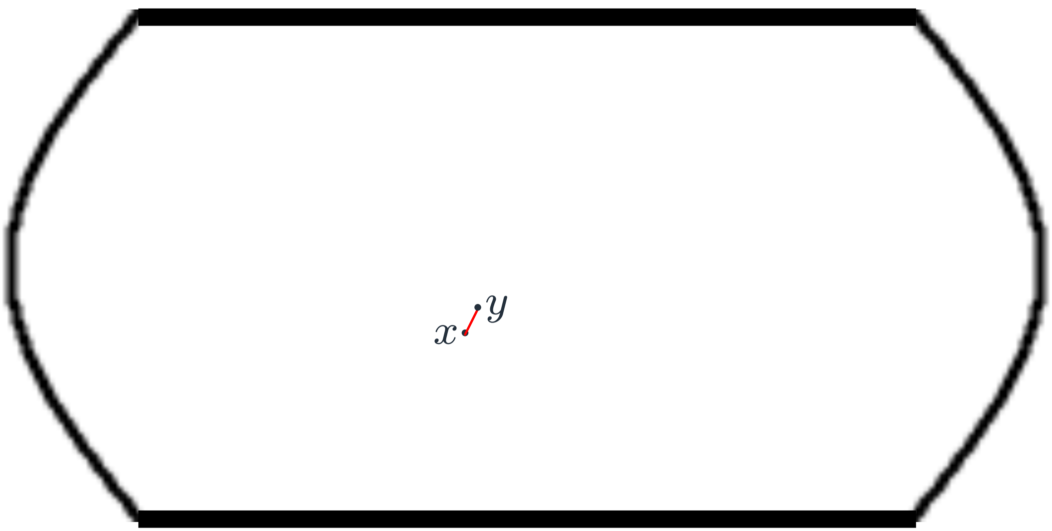


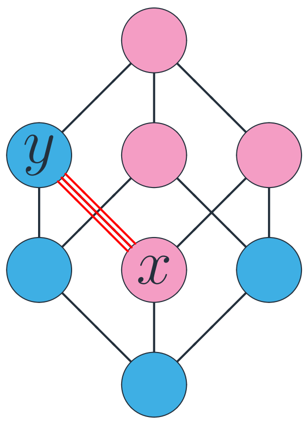
- Sample an edge xy of the hypercube $\{0,1\}^n$ uniformly at random ($x \prec y$, at distance 1).
- **Accept** if the edge is monotone, and **reject** otherwise.

Edge Tester: Algorithm



- Sample an edge xy of the hypercube $\{0,1\}^n$ uniformly at random ($x \prec y$, at distance 1).
- **Accept** if the edge is monotone, and **reject** otherwise.





- Sample an edge xy of the hypercube $\{0,1\}^n$ uniformly at random ($x \prec y$, at distance 1).
- **Accept** if the edge is monotone, and **reject** otherwise.

Theorem. *The edge tester*

(a) always accepts a monotone function;

(b) rejects a non-monotone function with probability $\Omega(\varepsilon/n)$.

- requires $O(n/\varepsilon)$ queries to test for monotonicity with $\Omega(1)$ success probability.
- is a non-adaptive tester with one-sided error.

Theorem. *The edge tester*

(a) always accepts a monotone function;

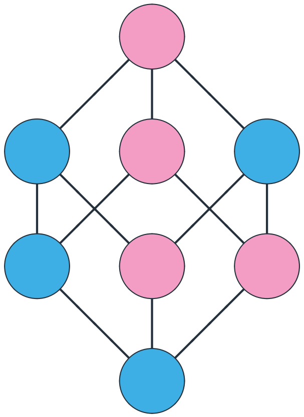
(b) rejects a non-monotone function with probability $\Omega(\varepsilon/n)$.

follows from

Observation. *There exists a monotone function at distance* $\leq 2K$ *from* $f\colon \{0,1\}^n \to \{0,1\}$, *where* K *is the number of non-monotone edges of* f.

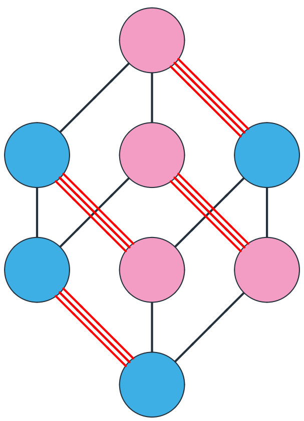
Indeed, $n2^{n-1}$ is the total number of edges,
and $\varepsilon 2^{n-1}$ is the number of non-monotone ones.

Observation. *There exists a monotone function at distance* $\leq 2K$ *from* $f\colon \{0,1\}^n \to \{0,1\}$*, where* K *is the number of non-monotone edges of* f*.*



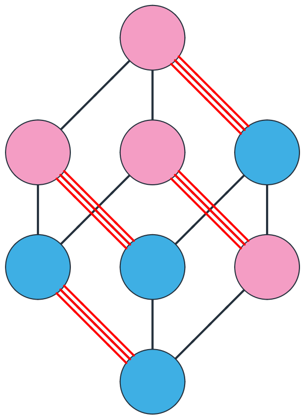
- For $i = 1, \ldots, n$:
  Sort the edges along the i-th direction
  (Replace 10-edges by 01-edges.)

Observation. *There exists a monotone function at distance* $\leq 2K$ *from* $f\colon \{0,1\}^n \to \{0,1\}$*, where* K *is the number of non-monotone edges of* f*.*



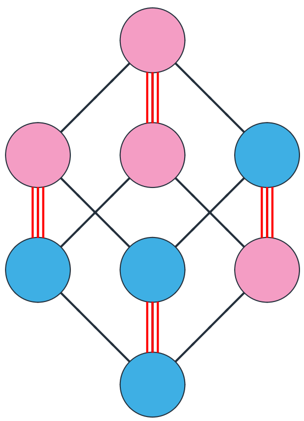
- For $i = 1, \ldots, n$:
  Sort the edges along the i-th direction
  (Replace 10-edges by 01-edges.)

Observation. *There exists a monotone function at distance* $\leq 2K$ *from* $f\colon \{0,1\}^n \to \{0,1\}$, *where* K *is the number of non-monotone edges of* f.



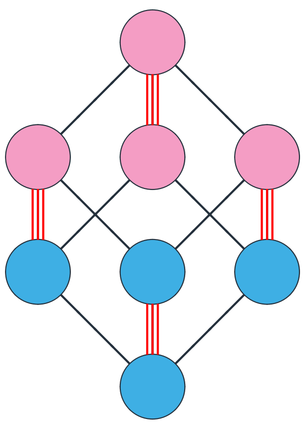
- For $i = 1, \ldots, n$:
  Sort the edges along the i-th direction
  (Replace 10-edges by 01-edges.)

Observation. *There exists a monotone function at distance* $\leq 2K$ *from* $f\colon \{0,1\}^n \to \{0,1\}$, *where* K *is the number of non-monotone edges of* f.



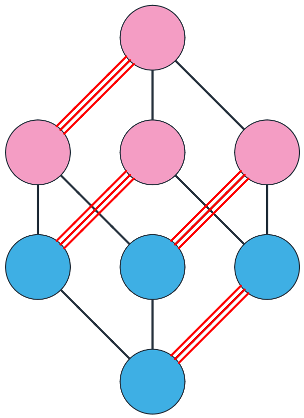
- For $i = 1, \ldots, n$:
  Sort the edges along the i-th direction
  (Replace 10-edges by 01-edges.)

Observation. *There exists a monotone function at distance* $\leq 2K$ *from* $f\colon \{0,1\}^n \to \{0,1\}$*, where* K *is the number of non-monotone edges of* f*.*



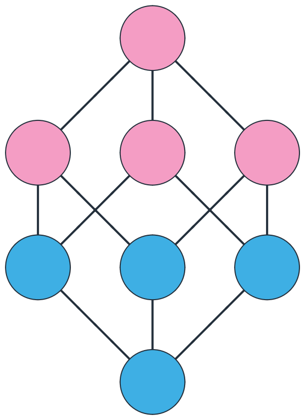
- For $i = 1, \ldots, n$:
  Sort the edges along the i-th direction
  (Replace 10-edges by 01-edges.)

Observation. *There exists a monotone function at distance* $\leq 2K$ *from* $f\colon \{0,1\}^n \to \{0,1\}$*, where* K *is the number of non-monotone edges of* f*.*



- For $i = 1, \ldots, n$:
  Sort the edges along the i-th direction
  (Replace 10-edges by 01-edges.)

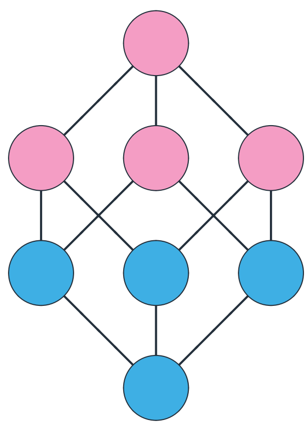
Observation. *There exists a monotone function at distance* $\leq 2K$ *from* $f\colon \{0,1\}^n \to \{0,1\}$, *where* K *is the number of non-monotone edges of* f.



- For $i = 1, \ldots, n$:
  Sort the edges along the i-th direction
  (Replace 10-edges by 01-edges.)

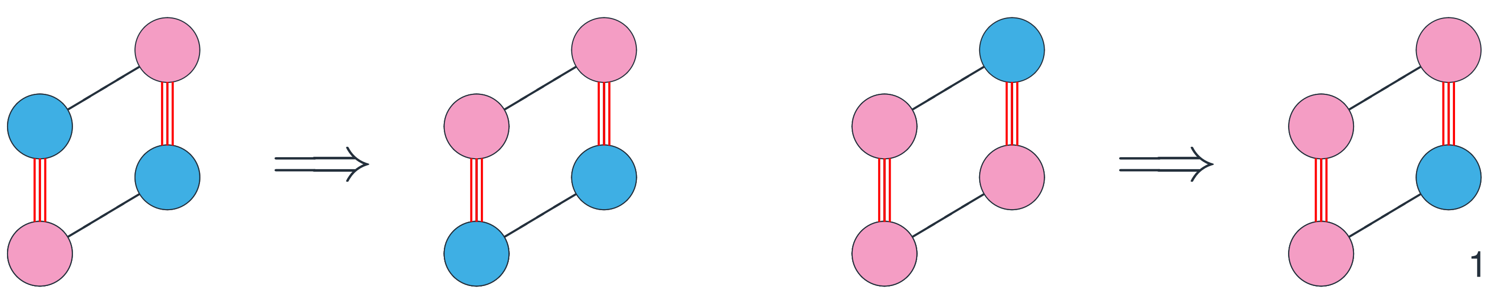
Edge Tester: Shifting

Observation. *There exists a monotone function at distance* $\leq 2K$ *from* $f\colon \{0,1\}^n \to \{0,1\}$, *where* K *is the number of non-monotone edges of* f.



- For $i = 1, \ldots, n$:
  Sort the edges along the i-th direction
  (Replace 10-edges by 01-edges.)

Lemma. *Shifting in the* i*-th direction does not increase the number of non-monotone edges in the* j*-th direction.*



STOC'13
- [Chakrabarty, Seshadhri]
  Proposed a path tester. Complexity: $\widetilde{O}(n^{7/8}\varepsilon^{-3/2})$.

FOCS'14
- [Chen, Se
  $\widetilde{\Omega}(n^{1/5})$
  improved

STOC'15
- [Chen, De
  Improved $n^{1/2-o(1)}$.

FOCS'15
- [Khot, Mi
  Brought complexity of path tester to $\widetilde{O}\left(\frac{\sqrt{n}}{\varepsilon^2}\right)$.

STOC'16
- [Belovs, Blais]
  $\widetilde{\Omega}(n^{1/4})$ adaptive lower bound.

Path Tester

Path Tester: Motivation

The edge tester performs badly on the anti-dictator function:

$$f(x) = \neg x_i.$$

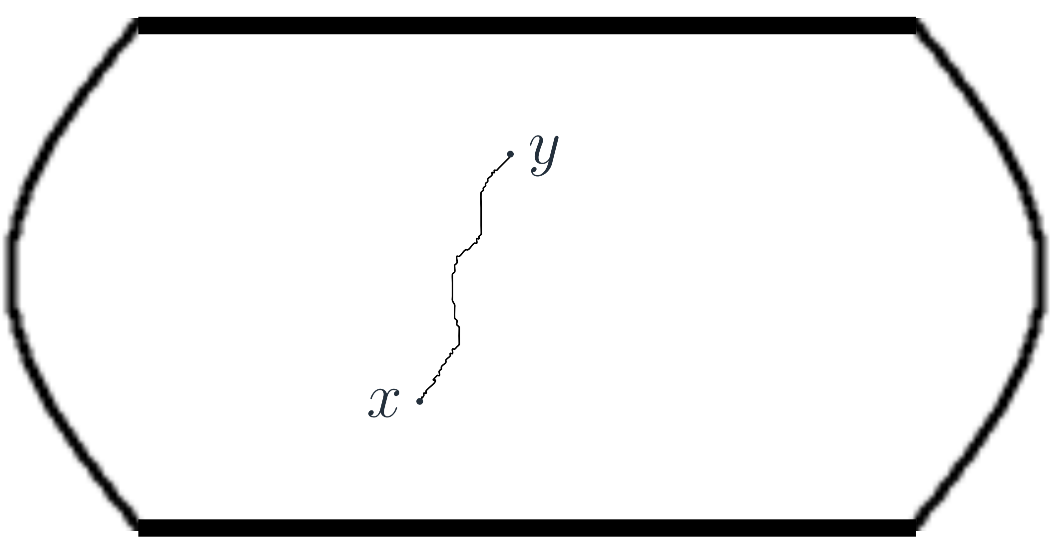
- At distance $1/2$ to monotone.
- Only 2^{n-1} non-monotone edges: Probability $1/n$ to succeed.

The edge tester performs badly on the anti-dictator function:

$$f(x) = \neg x_i.$$

- At distance $1/2$ to monotone.
- Only 2^{n-1} non-monotone edges: Probability $1/n$ to succeed.

Idea: Test multiple coordinates with one query.
Query $x \prec y$ at larger distances.



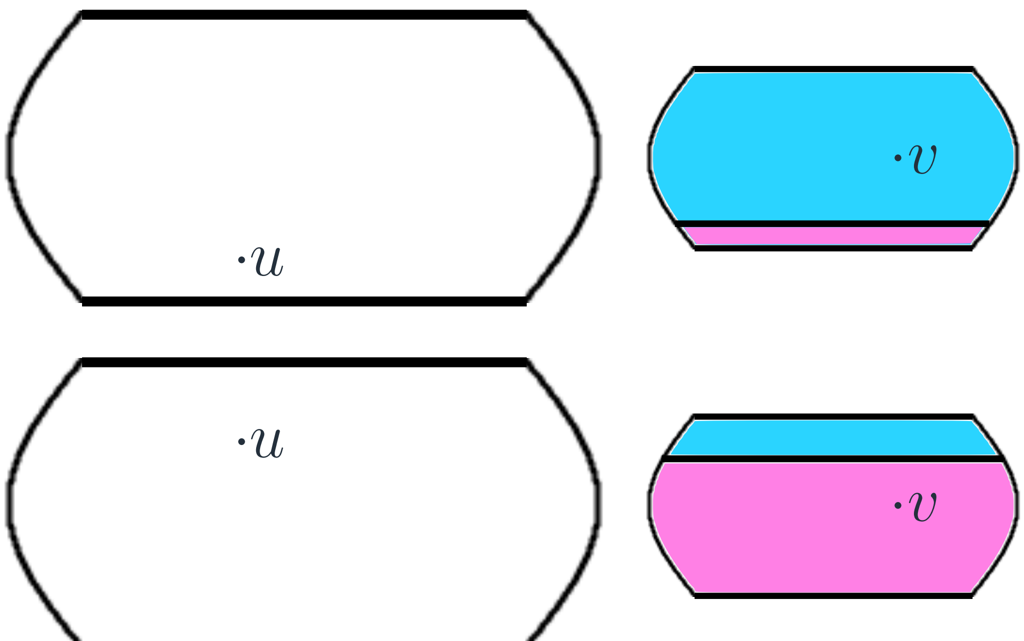
Path Tester: Hard functions

A Linear Threshold Function (LTF)

$f\colon \{-1,1\}^{n+m} \to \{-1,1\}$, with $m \ll n$.

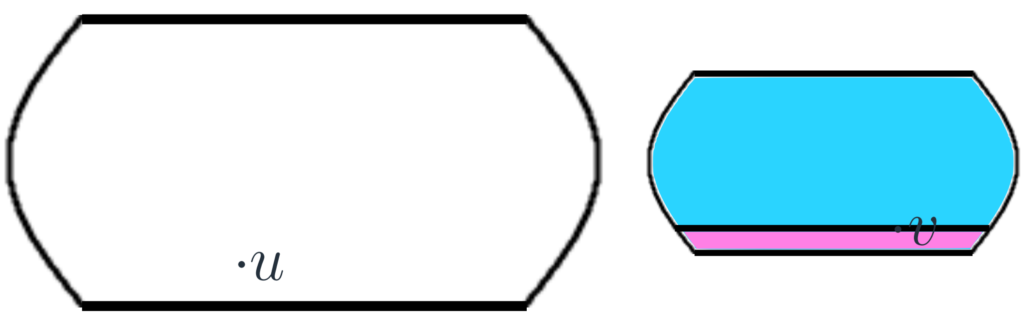
Write $f(u,v)$ with $u \in \{-1,1\}^n$, and $v \in \{-1,1\}^m$.

$$f(u,v) = 1 \quad \text{iff} \quad \frac{1}{\sqrt{n}} \sum_i u_i - \frac{1}{\sqrt{m}} \sum_j v_j \geq 0$$



$$h(u,v) = \frac{1}{\sqrt{n}}\sum_i u_i - \frac{1}{\sqrt{m}}\sum_j v_j \geq 0$$

Suppose $(u,v) \prec (u',v')$ are at distance k.
We want $f(u,v) = 1$ and $f(u',v') = 0$.



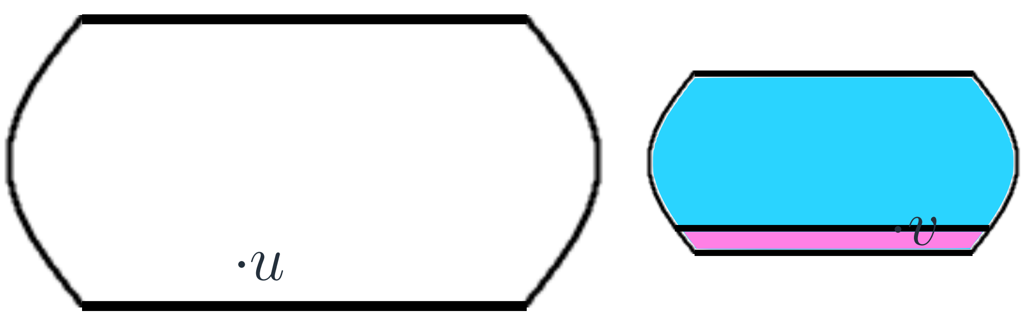
Path Tester: Hard functions

$$h(u,v) = \frac{1}{\sqrt{n}} \sum_i u_i - \frac{1}{\sqrt{m}} \sum_j v_j \geq 0$$

Suppose $(u,v) \prec (u',v')$ are at distance k.
We want $f(u,v) = 1$ and $f(u',v') = 0$.

- If k is large, this almost never happens, since, almost surely,

$$h(u',v') - h(u,v) \approx \frac{1}{\sqrt{n}} \cdot k - \frac{1}{\sqrt{m}} \cdot k \frac{m}{n} = k\left(\frac{1}{\sqrt{n}} - \frac{\sqrt{m}}{n}\right) > 0.$$



Path Tester: Hard functions

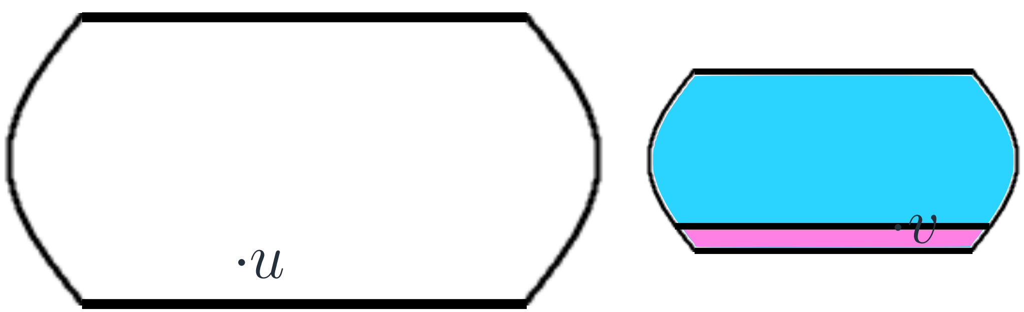
$$h(u,v) = \frac{1}{\sqrt{n}}\sum_i u_i - \frac{1}{\sqrt{m}}\sum_j v_j \geq 0$$

Suppose $(u,v) \prec (u',v')$ are at distance k.
We want $f(u,v) = 1$ and $f(u',v') = 0$.

- If $k = \frac{1}{10}\sqrt{\frac{n}{m}}$, then, with probability $\Omega(1)$, $f(u,\cdot) = f(u',\cdot)$.

  Success Probability $= \frac{1}{\sqrt{m}} \cdot k\frac{m}{n} = \frac{1}{\sqrt{m}} \cdot \sqrt{\frac{n}{m}\frac{m}{n}} = \frac{1}{\sqrt{n}}$.

- If k is even smaller, the probability decreases.



- Pick a parameter $k = 1, 2, 4, 8, \ldots, \sqrt{n}$ uniformly at random.
- Sample, uniformly at random, $x \prec y$ at distance k.
- **Accept** if the edge is monotone, and **reject** otherwise.

Theorem[Knot, Minzer, Safra ' 2015]. *The pair tester rejects a non-monotone function with probability* $\widetilde{\Omega}(\varepsilon^2/\sqrt{n})$.

	Edge Tester	Path Tester
Classical	$O(n/\varepsilon)$	$\widetilde{O}(\sqrt{n}/\varepsilon^2)$
Quantum	$O\left(\sqrt{n/\varepsilon}\right)$	$\widetilde{O}\left(n^{1/4}/\varepsilon\right)$

$$\widetilde{O}\left(n^{1/4}/\sqrt{\varepsilon}\right)$$

Non-Adaptive Lower Bounds

Random LTFs

Consider a random LTF $f\colon \{-1,1\}^n \to \{-1,1\}$:

$$f(x) = \operatorname{sgn}(\nu_1 x_1 + \nu_2 x_2 + \cdots + \nu_n x_n),$$

where

Yes case

$$\nu_i = \begin{cases} 1, & \text{w/ prob. } 1/2 \\ 3, & \text{w/ prob. } 1/2 \end{cases}$$

No case

$$\nu_i = \begin{cases} -1, & \text{w/ prob. } 1/10 \\ 7/3, & \text{w/ prob. } 9/10 \end{cases}$$

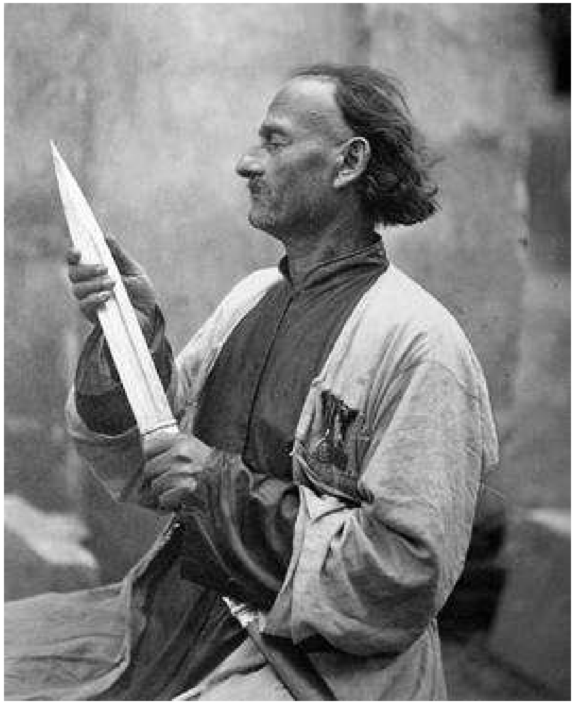
Theorem[Chen, Servedio, Tan]. *For any nearly-balanced* $x_1, \dots, x_q \in \{-1,1\}^n$,

$$d_{\mathrm{TVD}}\left(\Big(f(x_1), \dots, f(x_q)\Big)_{f\sim\text{Yes}}, \Big(g(x_1), \dots, g(x_q)\Big)_{g\sim\text{No}}\right) = \widetilde{O}\left(\frac{q^{5/4}}{n^{1/4}}\right).$$

- Gives $\widetilde{\Omega}(n^{1/5})$ lower bound. $n^{1/2-o(1)}$ bound is similar.

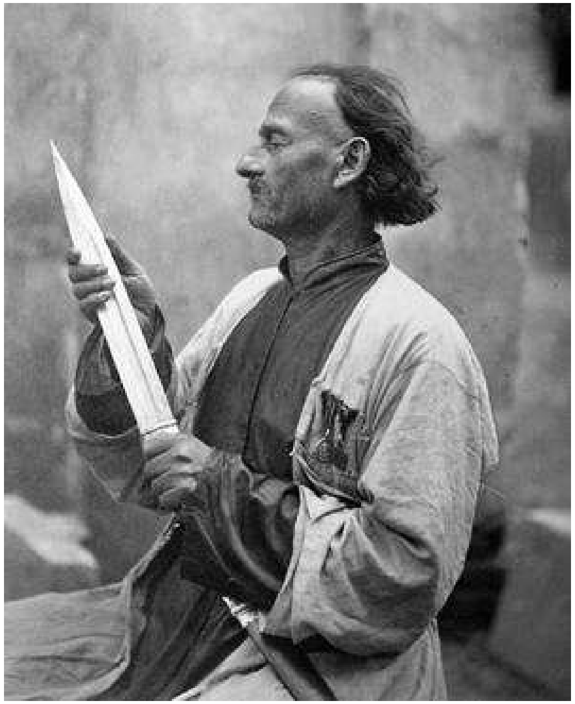
Adaptivity

- Remember the edge tester.

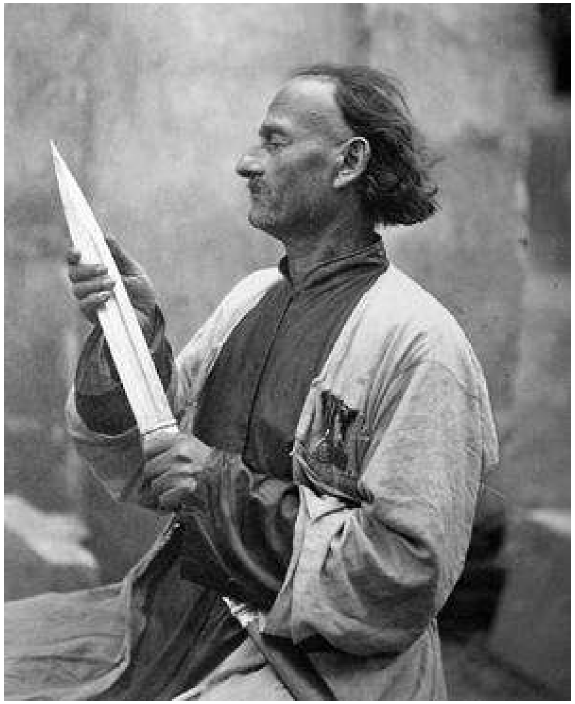


Intro

- Remember the edge tester.
- For any monotone
$$f\colon \{0,1\}^n \to \{0,1\},$$
at most $O(\frac{1}{\sqrt{n}})$ fraction of the edges are "interesting" (non-constant).



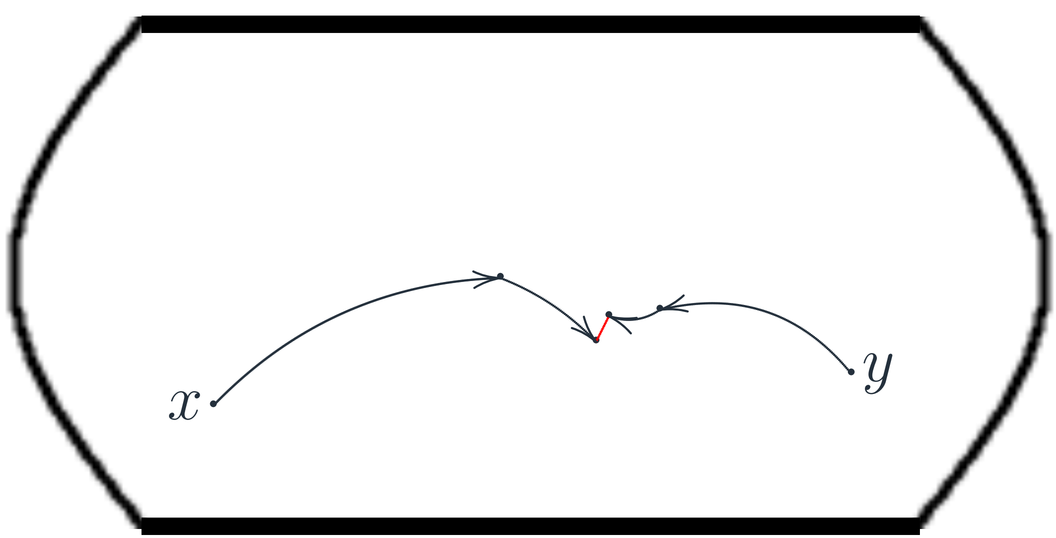
- Remember the edge tester.
- For any monotone
  $$f : \{0,1\}^n \to \{0,1\},$$
  at most $O(\frac{1}{\sqrt{n}})$ fraction of the edges are "interesting" (non-constant).
- Can we get an algorithm that always query "interesting" edges?



Bisection Algorithm

The Bisection Algorithm

- Sample $x \in f^{-1}(0)$ and $y \in f^{-1}(1)$ uniformly at random.
- **While** x and y differ in more than 1 variable:
  - Generate a uniformly random z between x and y
  - **If** $f(z) = 0$, let $x \leftarrow z$; otherwise, $y \leftarrow z$.
- **Accept** if xy is a monotone edge, and **reject** otherwise.



- Sample $x \in f^{-1}(0)$ and $y \in f^{-1}(1)$ uniformly at random.
- **While** x and y differ in more than 1 variable:
  - Generate a uniformly random z between x and y
  - **If** $f(z) = 0$, let $x \leftarrow z$; otherwise, $y \leftarrow z$.
- **Accept** if xy is a monotone edge, and **reject** otherwise.

+ The algorithm only tests non-constant edges.

- Sample $x \in f^{-1}(0)$ and $y \in f^{-1}(1)$ uniformly at random.
- **While** x and y differ in more than 1 variable:
  - Generate a uniformly random z between x and y
  - **If** $f(z) = 0$, let $x \leftarrow z$; otherwise, $y \leftarrow z$.
- **Accept** if xy is a monotone edge, and **reject** otherwise.

+ The algorithm only tests non-constant edges.

− Who knows as to which probability distribution it does it.

- Sample $x \in f^{-1}(0)$ and $y \in f^{-1}(1)$ uniformly at random.
- **While** x and y differ in more than 1 variable:
  - Generate a uniformly random z between x and y
  - **If** $f(z) = 0$, let $x \leftarrow z$; otherwise, $y \leftarrow z$.
- **Accept** if xy is a monotone edge, and **reject** otherwise.

\+ The algorithm only tests non-constant edges.

− Who knows as to which probability distribution it does it.

\+ Tests “nice” LTFs in $O(\log n)$ queries.

Fooling Bisection Algorithm

Let f be a monotone Boolean function.

$$f(\ x_1,\ x_2,\ x_3,\ x_4,\ x_5,\ x_6,\ x_7,\ x_8,\ x_9,\ x_{10}\ )$$

Fooling Bisection Algorithm

Let f be a monotone Boolean function.

Bisection algorithm generates probability distribution on variables.

$$f(\ x_1,\ x_2,\ x_3,\ x_4,\ x_5,\ x_6,\ x_7,\ x_8,\ x_9,\ x_{10}\ )$$

$$p_1 \quad p_2 \quad p_3 \quad p_4 \quad p_5 \quad p_6 \quad p_7 \quad p_8 \quad p_9 \quad p_{10}$$

Let f be a monotone Boolean function.
Bisection algorithm generates probability distribution on variables.

$$f(\; x_1,\; x_2,\; \neg x_3,\; x_4,\; \neg x_5,\; x_6,\; x_7,\; \neg x_8,\; x_9,\; x_{10} \;)$$

$$p_1 \quad p_2 \quad p_3 \quad p_4 \quad p_5 \quad p_6 \quad p_7 \quad p_8 \quad p_9 \quad p_{10}$$

Negating some variables, we get a non-monotone function

$$x \mapsto f(x^S).$$

Let f be a monotone Boolean function.
Bisection algorithm generates probability distribution on variables.

$$f(\; x_1, \; x_2, \; \neg x_3, \; x_4, \; \neg x_5, \; x_6, \; x_7, \; \neg x_8, \; x_9, \; x_{10} \;)$$

$$p_1 \quad p_2 \quad p_3 \quad p_4 \quad p_5 \quad p_6 \quad p_7 \quad p_8 \quad p_9 \quad p_{10}$$

Negating some variables, we get a non-monotone function

$$x \mapsto f(x^S).$$

1. Probability of the Bisection algorithm rejecting it — ?

Let f be a monotone Boolean function.
Bisection algorithm generates probability distribution on variables.

$$f(\; x_1, \; x_2, \neg x_3, \; x_4, \neg x_5, \; x_6, \; x_7, \neg x_8, \; x_9, \; x_{10} \;)$$

$$p_1 \quad p_2 \quad p_3 \quad p_4 \quad p_5 \quad p_6 \quad p_7 \quad p_8 \quad p_9 \quad p_{10}$$

Negating some variables, we get a non-monotone function

$$x \mapsto f(x^S).$$

1. Probability of the Bisection algorithm rejecting it — ?

$$\sum_{i \in S} p_i$$

2. Distance to monotonicity of

$$f(x_1, x_2, \neg x_3, x_4, \neg x_5, x_6, x_7, \neg x_8, x_9, x_{10})$$

Fooling Bisection Algorithm: Distance

2. Distance to monotonicity of

$$f(\;x_1,\;x_2,\neg x_3,\;x_4,\neg x_5,\;x_6,\;x_7,\neg x_8,\;x_9,\;x_{10}\;)$$

Noise sensitivity of a function f is defined as

$$\mathrm{NS}_\delta(f) = \Pr_{x,\,S}\Big[f(x) \neq f(x^S)\Big],$$

where $x \sim \{0,1\}^n$ and $S \subseteq [n]$, each element with probability δ.

2. Distance to monotonicity of

$$f(\ x_1,\ x_2, \neg x_3,\ x_4, \neg x_5,\ x_6,\ x_7, \neg x_8,\ x_9,\ x_{10})$$

Noise sensitivity of a function f is defined as

$$\mathrm{NS}_\delta(f) = \Pr_{x,\,S}\Big[f(x) \neq f(x^S)\Big],$$

where $x \sim \{0,1\}^n$ and $S \subseteq [n]$, each element with probability δ.

The distance is at least

$$\frac{1}{2} \Pr_{x \sim \{0,1\}^n}\Big[f(x) \neq f(x^S)\Big].$$

Fooling Bisection Algorithm: Distance

The distance of $x \mapsto f(x^S)$ to monotonicity is at least

$$\frac{1}{2} \Pr_{x \sim \{0,1\}^n}\Big[f(x) \neq f(x^S)\Big].$$

Proof. Write $x = (u, v)$ for $u \in \{0,1\}^{[n] \setminus S}$, $v \in \{0,1\}^S$.
Let μ and $\varkappa$ be the distance to a monotone and a constant function.

$$\begin{aligned}\mu(f) &\geq \mathbb{E}_u \mu\big(f(u,\cdot)\big) = \mathbb{E}_u \varkappa\big(f(u,\cdot)\big) \\ &\geq \frac{1}{2}\mathbb{E}_u \Pr_v\Big[f(u,v) \neq f(u,v^S)\Big] = \frac{1}{2}\Pr_x\Big[f(x) \neq f(x^S)\Big].\end{aligned}$$

$$f(\;x_1,\; x_2,\; \neg x_3,\; x_4,\; \neg x_5,\; x_6,\; x_7,\; \neg x_8,\; x_9,\; x_{10}\;)$$

$$p_1 \quad p_2 \quad p_3 \quad p_4 \quad p_5 \quad p_6 \quad p_7 \quad p_8 \quad p_9 \quad p_{10}$$

$$\sum_{i \in S} p_i \qquad \text{vs.} \qquad \frac{1}{2} \Pr_{x \sim \{0,1\}^n}\left[f(x) \neq f(x^S)\right]$$

$$\exists f: \quad \mathrm{NS}_{\frac{1}{\sqrt{n}}}(f) = \Pr_{x,\,S}\left[f(x) \neq f(x^S)\right] = \Omega(1).$$

Fooling Bisection Algorithm: Fooled!

$$f\left(\begin{matrix} x_1, & x_2, & \neg x_3, & x_4, & \neg x_5, & x_6, & x_7, & \neg x_8, & x_9, & x_{10} \\ p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 & p_9 & p_{10} \end{matrix} \right)$$

$$\sum_{i \in S} p_i \quad \text{vs.} \quad \frac{1}{2} \Pr_{x \sim \{0,1\}^n}\left[f(x) \neq f(x^S) \right]$$

$$\exists f: \quad \mathrm{NS}_{\frac{1}{\sqrt{n}}}(f) = \Pr_{x,\, S}\left[f(x) \neq f(x^S) \right] = \Omega(1).$$

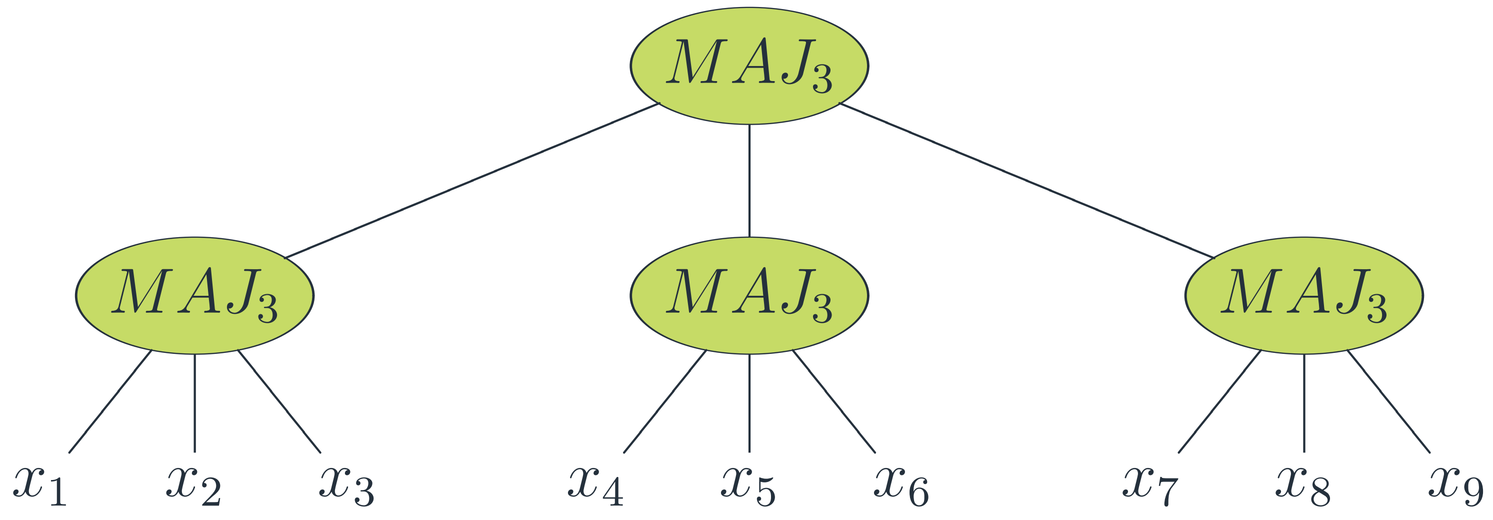
Exists S such that $x \mapsto f(x^S)$

(a) is $\Omega(1)$ far from monotone;

(b) is rejected by the Bisection algorithm with probability $O\left(\frac{1}{\sqrt{n}}\right)$.

What are the noise-sensitive monotone functions?

(a) Iterated Majority



What are the noise-sensitive monotone functions?

(b) Talagrand's Random DNF

A disjunction of $2^{\sqrt{n}}$ independent random clauses of size $\sqrt{n}$.

$$f_C(x) = \bigwedge_{a \in [\sqrt{n}]} x_{C(a)} \quad \text{and} \quad f(x) = \bigvee_{j \in [2^{\sqrt{n}}]} f_{C_j}(x).$$

Let Tal be Talagrand's Random DNF, and

$$\mathsf{Tal}^{\pm} = \{x \mapsto f(x^S) \mid f \sim \mathsf{Tal},\ S\}.$$

Theorem. *For all $q = O(n^{1/4} \log^{-2} n)$, nearly-balanced $x_1, \ldots, x_q \in \{0,1\}^n$ and $b_1, \ldots, b_q \in \{0,1\}$, we have*

$$\Pr_{f \sim \mathsf{Tal}}\Big[\forall i\colon f(x_i) = b_i\Big] \leq (1 + o(1)) \Pr_{g \sim \mathsf{Tal}^{\pm}}\Big[\forall i\colon g(x_i) = b_i\Big] + o(2^{-q}).$$

Open Problems

- Close the gap between $\widetilde{\Omega}(n^{1/4})$ and $\widetilde{O}(\sqrt{n})$.
- Can we get more from the bisection algorithm? When is it effective?

- Prove quantum lower bounds.
  - Monotonicity on the line — ? $f : [n] \to [m]$.

Thank you!