Lower Bounds for DNF-Refutations of a Relativized Weak PHP

Albert Atserias
Universitat Politècnica de Catalunya
Barcelona, Spain

Joint work with
Sergi Oliva (Barcelona)
Mortiz Müller (Vienna)

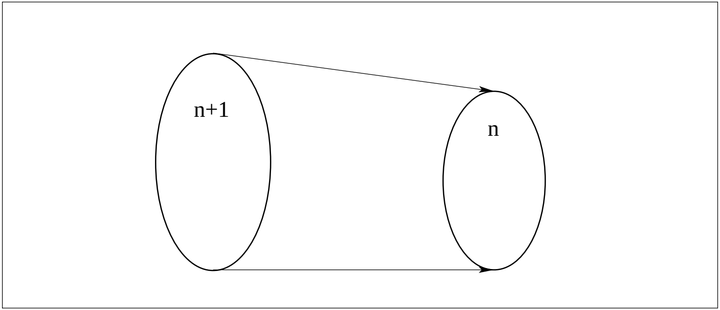
Part I

BACKGROUND

Pigeonhole Principle

PHP$_n^{n+1}$:

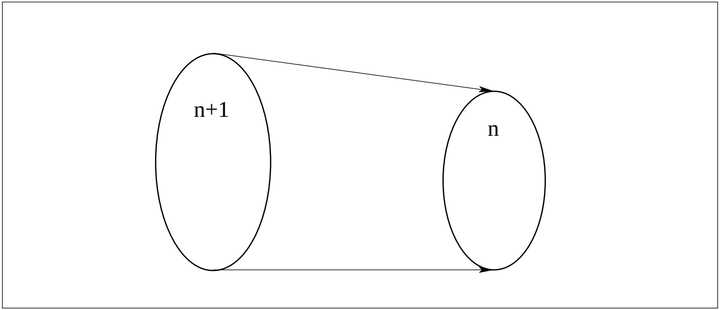
If $n+1$ pigeons fly to n holes, then some hole gets overbooked.



Pigeonhole Principle

$\mathbf{PHP}_n^{n+1}$:

It can't be that $n+1$ pigeons fly but only n make it.



PHP vs. Counting

Counting should be all there is to it:

- If you can count, you can prove it.
- If you can't count, your can't prove it.

PHP vs. Counting

Counting should be all there is to it:

- If you can count, you can prove it.
- If you can't count, your can't prove it.

History:

PHP vs. Counting

Counting should be all there is to it:

- If you can count, you can prove it.
- If you can't count, your can't prove it.

History:

- PHP_n^{n+1} has poly-size P/poly-proofs [CR79].

PHP vs. Counting

Counting should be all there is to it:

- If you can count, you can prove it.
- If you can't count, your can't prove it.

History:

- PHP_n^{n+1} has poly-size P/poly-proofs [CR79].
- PHP_n^{n+1} has poly-size NC^1-proofs [B87].

PHP vs. Counting

Counting should be all there is to it:

- If you can count, you can prove it.
- If you can't count, your can't prove it.

History:

- PHP_n^{n+1} has poly-size P/poly-proofs [CR79].
- PHP_n^{n+1} has poly-size NC^1-proofs [B87].
- PHP_n^{n+1} has poly-size TC^0-proofs [CCT87].

PHP vs. Counting

Counting should be all there is to it:

- If you can count, you can prove it.
- If you can't count, your can't prove it.

History:

- PHP_n^{n+1} has poly-size P/poly-proofs [CR79].
- PHP_n^{n+1} has poly-size NC^1-proofs [B87].
- PHP_n^{n+1} has poly-size TC^0-proofs [CCT87].
- PHP_n^{n+1} doesn't have subexp-size R-proofs [H85].

PHP vs. Counting

Counting should be all there is to it:

- If you can count, you can prove it.
- If you can't count, your can't prove it.

History:

- PHP_n^{n+1} has poly-size P/poly-proofs [CR79].
- PHP_n^{n+1} has poly-size NC^1-proofs [B87].
- PHP_n^{n+1} has poly-size TC^0-proofs [CCT87].
- PHP_n^{n+1} doesn't have subexp-size R-proofs [H85].
- PHP_n^{n+1} doesn't have poly-size AC^0-proofs [A88].

PHP vs. Counting

Counting should be all there is to it:

- If you can count, you can prove it.
- If you can't count, your can't prove it.

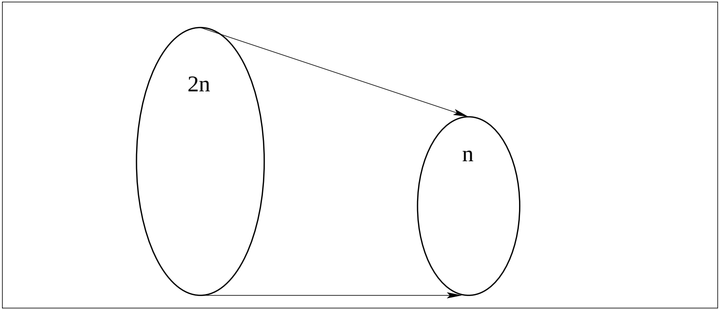
History:

- PHP_n^{n+1} has poly-size P/poly-proofs [CR79].
- PHP_n^{n+1} has poly-size NC^1-proofs [B87].
- PHP_n^{n+1} has poly-size TC^0-proofs [CCT87].
- PHP_n^{n+1} doesn't have subexp-size R-proofs [H85].
- PHP_n^{n+1} doesn't have poly-size AC^0-proofs [A88].
- PHP_n^{n+1} doesn't have subexp-size AC^0-proofs [BIP+KPW92].

Weak Pigeonhole Principle

$\mathbf{PHP}^{2n}_{n}$:

It can't be that $2n$ pigeons fly but only n make it.



WPHP vs. Approximate counting

Approximate counting should be enough:

WPHP vs. Approximate counting

Approximate counting should be enough:

1. Assume the opposite of what PHP_n^{2n} says.

WPHP vs. Approximate counting

Approximate counting should be enough:

1. Assume the opposite of what PHP_n^{2n} says.
2. Count flights approximately: $\geq (2-\epsilon)n$.

WPHP vs. Approximate counting

Approximate counting should be enough:

1. Assume the opposite of what PHP_n^{2n} says.
2. Count flights approximately: $\geq (2-\epsilon)n$.
3. Count landings approximately: $\leq (1+\epsilon)n$.

WPHP vs. Approximate counting

Approximate counting should be enough:

1. Assume the opposite of what PHP^{2n}_{n} says.
2. Count flights approximately: $\geq (2 - \epsilon)n$.
3. Count landings approximately: $\leq (1 + \epsilon)n$.
4. Contradiction.

WPHP vs. Approximate counting

Approximate counting should be enough:

1. Assume the opposite of what PHP_n^{2n} says.
2. Count flights approximately: $\geq (2-\epsilon)n$.
3. Count landings approximately: $\leq (1+\epsilon)n$.
4. Contradiction.

Note:

Poly-size AC^0 circuits can approximately count [AB84, A93].

WPHP vs. Approximate counting

Approximate counting should be enough:

1. Assume the opposite of what PHP_n^{2n} says.
2. Count flights approximately: $\geq (2-\epsilon)n$.
3. Count landings approximately: $\leq (1+\epsilon)n$.
4. Contradiction.

Note:

Poly-size AC^0 circuits can approximately count [AB84, A93].

However:

Does PHP_n^{2n} have poly-size AC^0-refutations? OPEN

What's Known?

Upper bounds for $\mathbf{PHP}_n^{2n}$:

Lower bounds for $\mathbf{PHP}_n^{2n}$:

What's Known?

Upper bounds for $\mathbf{PHP}_n^{2n}$:

1. PHP_n^{2n} has quasipoly-size AC^0-proofs [PWW88].

Lower bounds for $\mathbf{PHP}_n^{2n}$:

What's Known?

Upper bounds for PHP_n^{2n}:

1. PHP_n^{2n} has quasipoly-size AC^0-proofs [PWW88].
2. PHP_n^{2n} has quasipoly-size $\Sigma_{2,\mathrm{polylog}}$-proofs [MPW00].

Lower bounds for PHP_n^{2n}:

What's Known?

Upper bounds for PHP^{2n}_n:

1. PHP^{2n}_n has quasipoly-size AC^0-proofs [PWW88].
2. PHP^{2n}_n has quasipoly-size $\Sigma_{2,\mathrm{polylog}}$-proofs [MPW00].
3. PHP^{2n}_n has $n^{O(d(\log n)^{1/d})}$-size Σ_d-proofs [A01].

Lower bounds for PHP^{2n}_n:

What's Known?

Upper bounds for $\mathbf{PHP}_n^{2n}$:

1. PHP_n^{2n} has quasipoly-size AC^0-proofs [PWW88].
2. PHP_n^{2n} has quasipoly-size $\Sigma_{2,\mathrm{polylog}}$-proofs [MPW00].
3. PHP_n^{2n} has $n^{O(d(\log n)^{1/d})}$-size Σ_d-proofs [A01].

Lower bounds for $\mathbf{PHP}_n^{2n}$:

1. PHP_n^{2n} doesn't have subexp-size R-proofs [BT88].

What's Known?

Upper bounds for PHP_n^{2n}:

1. PHP_n^{2n} has quasipoly-size AC^0-proofs [PWW88].
2. PHP_n^{2n} has quasipoly-size $\Sigma_{2,\mathrm{polylog}}$-proofs [MPW00].
3. PHP_n^{2n} has $n^{O(d(\log n)^{1/d})}$-size Σ_d-proofs [A01].

Lower bounds for PHP_n^{2n}:

1. PHP_n^{2n} doesn't have subexp-size R-proofs [BT88].
2. PHP_n^{2n} doesn't have subexp-size $\Sigma_{2,2}$-proofs [ABE01].

What's Known?

Upper bounds for PHP_n^{2n}:

1. PHP_n^{2n} has quasipoly-size AC^0-proofs [PWW88].
2. PHP_n^{2n} has quasipoly-size $\Sigma_{2,\mathrm{polylog}}$-proofs [MPW00].
3. PHP_n^{2n} has $n^{O(d(\log n)^{1/d})}$-size Σ_d-proofs [A01].

Lower bounds for PHP_n^{2n}:

1. PHP_n^{2n} doesn't have subexp-size R-proofs [BT88].
2. PHP_n^{2n} doesn't have subexp-size $\Sigma_{2,2}$-proofs [ABE01].
3. PHP_n^{2n} doesn't have subexp-size $\Sigma_{2,(\log n)^{1/2-\epsilon}}$-proofs [SBI02].

What's Known?

Upper bounds for PHP_n^{2n}:

1. PHP_n^{2n} has quasipoly-size AC^0-proofs [PWW88].
2. PHP_n^{2n} has quasipoly-size $\Sigma_{2,\mathrm{polylog}}$-proofs [MPW00].
3. PHP_n^{2n} has $n^{O(d(\log n)^{1/d})}$-size Σ_d-proofs [A01].

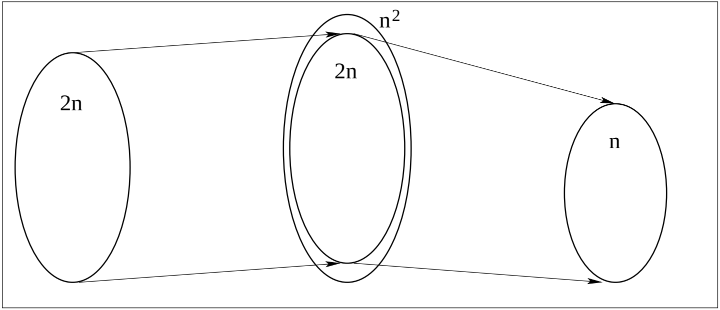
Lower bounds for PHP_n^{2n}:

1. PHP_n^{2n} doesn't have subexp-size R-proofs [BT88].
2. PHP_n^{2n} doesn't have subexp-size $\Sigma_{2,2}$-proofs [ABE01].
3. PHP_n^{2n} doesn't have subexp-size $\Sigma_{2,(\log n)^{1/2-\epsilon}}$-proofs [SBI02].
4. PHP_n^{2n} doesn't have subexp-size $\Sigma_{2,(\log n)^{1-\epsilon}}$-proofs [R02].

A Relativized PHP

$\mathbf{PHP}^{\ell,m}_n$ with $\ell > m > n$:

If can't be that m out of ℓ pigeons fly but only n make it.



RWPHP vs. Approximate counting

RWPHP vs. Approximate counting

Approximate counting should still be enough for $\mathbf{PHP}_n^{n^2,2n}$:

Same argument.

RWPHP vs. Approximate counting

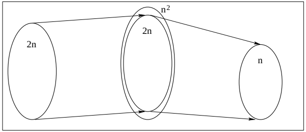
Approximate counting should still be enough for $\mathbf{PHP}_n^{n^2,2n}$:

Same argument.

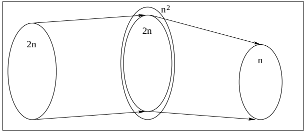
Upper bound and equivalence:

F1: $\mathrm{PHP}_n^{n^2,2n}$ has quasipoly-size proofs in $\Sigma_{2,\mathrm{polylog}}$.
F2: $\mathrm{PHP}_n^{n^2,2n} \equiv \mathrm{PHP}_n^{2n}$ mod poly-size AC^0-proofs.

The Formula: $\mathrm{PHP}_n^{n^2,2n}$



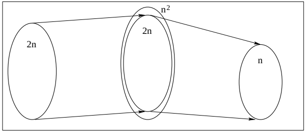
The Formula: $\mathrm{PHP}_n^{n^2,2n}$



Variables:

- $x_{i,j}$: the i-th flying pigeon is j.
- y_j: pigeon j flies.
- $z_{j,k}$: pigeon j flies to hole k.

The Formula: $\mathrm{PHP}_n^{n^2,2n}$



Variables:

- $x_{i,j}$: the i-th flying pigeon is j.
- y_j: pigeon j flies.
- $z_{j,k}$: pigeon j flies to hole k.

Clauses:

- some pigeon is pigeon number i,
- pigeon j cannot be both the i-th and the i'-th, for $i \neq i'$,
- if pigeon j is the i-th, then j flies and it reaches some hole,
- if j and j' fly, then they do not both reach hole k, for $j \neq j'$.

Part II

MAIN RESULT

Main Result

Theorem:

Every Σ_2-proof of $PHP_n^{n^2,2n}$ has size $n^{(\log n)^{1/2-o(1)}}$.

Observations/Comments

Observations/Comments

Quasi-polynomial upper and lower bounds at depth-2:

$$n^{(\log n)^{1/2-o(1)}} \leq \text{OPT} \leq n^{(\log n)^{O(1)}}$$

Observations/Comments

Quasi-polynomial upper and lower bounds at depth-2:

$$n^{(\log n)^{1/2-o(1)}} \leq \text{OPT} \leq n^{(\log n)^{O(1)}}$$

Lifts the right amount of hardness from depth-1 to depth-2:

could be useful for the depth-hierarchy problem for AC^0-proofs

Observations/Comments

Quasi-polynomial upper and lower bounds at depth-2:

$$n^{(\log n)^{1/2-o(1)}} \leq \mathrm{OPT} \leq n^{(\log n)^{O(1)}}$$

Lifts the right amount of hardness from depth-1 to depth-2:

could be useful for the depth-hierarchy problem for AC^0-proofs

Reaching depth-2 without hiding Ajtai's method in it:

first super-polynomial lower bound for $\Sigma_{>1}$-proofs that does not follow from Ajtai's forcing method.

Part III

PROOF SKETCH

Random Restriction Method in Proof Complexity

Illustrated for PHP_n^{2n} [ABE, SBI, R]:

Illustrated for PHP_n^{2n} [ABE, SBI, R]:

1. assume PHP_n^{2n} has a small $\Sigma_{2,k}$-refutation,

Random Restriction Method in Proof Complexity

Illustrated for PHP_n^{2n} [ABE, SBI, R]:

1. assume PHP_n^{2n} has a small $\Sigma_{2,k}$-refutation,
2. restrict it to a low-degree bipartite resilient expander G,

Random Restriction Method in Proof Complexity

Illustrated for PHP_n^{2n} [ABE, SBI, R]:

1. assume PHP_n^{2n} has a small $\Sigma_{2,k}$-refutation,
2. restrict it to a low-degree bipartite resilient expander G,
3. assign a random $\sim n/2$-matching $M \subseteq G$ of pigeons to holes,

Random Restriction Method in Proof Complexity

Illustrated for PHP_n^{2n} [ABE, SBI, R]:

1. assume PHP_n^{2n} has a small $\Sigma_{2,k}$-refutation,
2. restrict it to a low-degree bipartite resilient expander G,
3. assign a random $\sim n/2$-matching $M \subseteq G$ of pigeons to holes,
4. show each $\Sigma_{2,k}$-formula simplifies to a low-depth decision tree,

Random Restriction Method in Proof Complexity

Illustrated for PHP_n^{2n} [ABE, SBI, R]:

1. assume PHP_n^{2n} has a small $\Sigma_{2,k}$-refutation,
2. restrict it to a low-degree bipartite resilient expander G,
3. assign a random $\sim n/2$-matching $M \subseteq G$ of pigeons to holes,
4. show each $\Sigma_{2,k}$-formula simplifies to a low-depth decision tree,
5. use expansion of $G \setminus V(M)$ to argue directly that restricted formula does not have such refutations.

Random Restriction Method in Proof Complexity

Illustrated for PHP_n^{2n} [ABE, SBI, R]:

1. assume PHP_n^{2n} has a small $\Sigma_{2,k}$-refutation,
2. restrict it to a low-degree bipartite resilient expander G,
3. assign a random $\sim n/2$-matching $M \subseteq G$ of pigeons to holes,
4. show each $\Sigma_{2,k}$-formula simplifies to a low-depth decision tree,
5. use expansion of $G \setminus V(M)$ to argue directly that restricted formula does not have such refutations.

Main obstacle:

step 4 is fundamentally limited to $k \leq (\log n)^{1-o(1)}$

Random Restriction Method in Proof Complexity

Illustrated for PHP^{2n}_n [ABE, SBI, R]:

1. assume PHP^{2n}_n has a small $\Sigma_{2,k}$-refutation,
2. restrict it to a low-degree bipartite resilient expander G,
3. assign a random $\sim n/2$-matching $M \subseteq G$ of pigeons to holes,
4. show each $\Sigma_{2,k}$-formula simplifies to a low-depth decision tree,
5. use expansion of $G \setminus V(M)$ to argue directly that restricted formula does not have such refutations.

Main obstacle:

step 4 is fundamentally limited to $k \leq (\log n)^{1-o(1)}$

Discouraging obstacle:

there is little hope this technique would yield superpolynomial, but not superquasipolynomial, lower bounds

Enters Relativization: first attempt

Steps:

Enters Relativization: first attempt

Steps:

1. assume $\mathrm{PHP}_n^{n^2,2n}$ has a small Σ_2-refutation,

Enters Relativization: first attempt

Steps:

1. assume $\mathrm{PHP}_n^{n^2,2n}$ has a small Σ_2-refutation,
2. choose $2n$ out of n^2 flying pigeons at random,

Enters Relativization: first attempt

Steps:

1. assume $\mathrm{PHP}_n^{n^2,2n}$ has a small Σ_2-refutation,
2. choose $2n$ out of n^2 flying pigeons at random,
3. set all the x-variables and the y-variables accordingly,

Enters Relativization: first attempt

Steps:

1. assume $\mathrm{PHP}_n^{n^2,2n}$ has a small Σ_2-refutation,
2. choose $2n$ out of n^2 flying pigeons at random,
3. set all the x-variables and the y-variables accordingly,
4. set all z-variables about other pigeons to $1/0$ with prob. $1/2$,

Enters Relativization: first attempt

Steps:

1. assume $\mathrm{PHP}_n^{n^2,2n}$ has a small Σ_2-refutation,
2. choose $2n$ out of n^2 flying pigeons at random,
3. set all the x-variables and the y-variables accordingly,
4. set all z-variables about other pigeons to $1/0$ with prob. $1/2$,
5. argue each remaining term has length $\leq (\log n)^{1/2-o(1)}$,

Enters Relativization: first attempt

Steps:

1. assume $\mathrm{PHP}_n^{n^2,2n}$ has a small Σ_2-refutation,
2. choose $2n$ out of n^2 flying pigeons at random,
3. set all the x-variables and the y-variables accordingly,
4. set all z-variables about other pigeons to $1/0$ with prob. $1/2$,
5. argue each remaining term has length $\leq (\log n)^{1/2-o(1)}$,
6. hope we reduced to the case PHP_n^{2n}.

Enters Relativization: first attempt

Steps:

1. assume $\mathrm{PHP}_n^{n^2,2n}$ has a small Σ_2-refutation,
2. choose $2n$ out of n^2 flying pigeons at random,
3. set all the x-variables and the y-variables accordingly,
4. set all z-variables about other pigeons to $1/0$ with prob. $1/2$,
5. argue each remaining term has length $\leq (\log n)^{1/2-o(1)}$,
6. hope we reduced to the case PHP_n^{2n}.

An obstacle:

the length in step 5 is "number of mentioned pigeons"
instead of number of variables as we would need

Analysis of step 5

2. choose $2n$ out of n^2 flying pigeons at random,
3. set x- and y-variables accordingly,
4. set z-variables randomly for other pigeons,
5. argue each remaining term has "length" $\leq (\log n)^{1/2-o(1)}$

Analysis of step 5

2. choose $2n$ out of n^2 flying pigeons at random,
3. set x- and y-variables accordingly,
4. set z-variables randomly for other pigeons,
5. argue each remaining term has "length" $\leq (\log n)^{1/2-o(1)}$

Furst-Saxe-Sipser kind of argument:

Analysis of step 5

2. choose $2n$ out of n^2 flying pigeons at random,
3. set x- and y-variables accordingly,
4. set z-variables randomly for other pigeons,
5. argue each remaining term has "length" $\leq (\log n)^{1/2-o(1)}$

Furst-Saxe-Sipser kind of argument:

L1: If t mentions at least s pigeons, then

$$\Pr[\ t \text{ survives }] \leq \left(\frac{2n}{n^2 - s} + \frac{1}{2} \right)^s$$

Analysis of step 5

2. choose $2n$ out of n^2 flying pigeons at random,
3. set x- and y-variables accordingly,
4. set z-variables randomly for other pigeons,
5. argue each remaining term has "length" $\leq (\log n)^{1/2-o(1)}$

Furst-Saxe-Sipser kind of argument:

L1: If t mentions at least s pigeons, then

$$\Pr[\ t \text{ survives }] \leq \left(\frac{2n}{n^2 - s} + \frac{1}{2}\right)^s$$

L2: If t mentions less than s pigeons and $k < s$, then

$$\Pr[\ t \text{ mentions } \geq k \text{ pigeons }] \leq \binom{s}{k}\left(\frac{2n}{n^2}\right)^k$$

Bypassing the obstacle: first attempt

Recall the obstacle:

length is "number of mentioned pigeons" k

Bypassing the obstacle: first attempt

Recall the obstacle:

length is “number of mentioned pigeons” k

Obvious (but failing) idea:

6. restrict to a low-degree resilient expander after step 5

Bypassing the obstacle: first attempt

Recall the obstacle:

length is "number of mentioned pigeons" k

Obvious (but failing) idea:

6. restrict to a low-degree resilient expander after step 5

Obstacle:

resilient expanders require degree $\geq \Omega(\log n / \log \log n)$
so we cannot afford more than $k = o(1)$.

Bypassing the obstacle: second attempt

Idea:

restrict to resilient expander but encode flights in binary!

$$x_{j,k} := \bigwedge_{b=1}^{\log\log n} x'_{j;b}$$

Bypassing the obstacle: second attempt

Idea:

restrict to resilient expander but encode flights in binary!

$$x_{j,k} := \bigwedge_{b=1}^{\log\log n} x'_{j;b}$$

Effect:

moves us from k pigeons/term to $k \log\log n$ vars/term
so we can afford $k = (\log n)^{1-o(1)}$.

Bypassing the obstacle: second attempt

Idea:

restrict to resilient expander but encode flights in binary!

$$x_{j,k} := \bigwedge_{b=1}^{\log \log n} x'_{j;b}$$

Effect:

moves us from k pigeons/term to $k \log \log n$ vars/term
so we can afford $k = (\log n)^{1-o(1)}$.

Solvable technicalities:

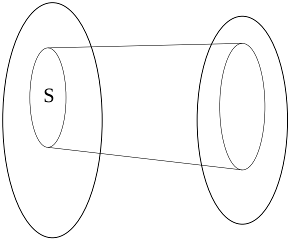
1. depth increases by 1 (solution: not really),
2. formula changes (solution: reprove height lower bound for it).

Part IV

RESILIENT EXPANDERS

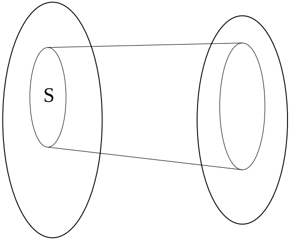
Bipartite expanders

(s, e)-**expander**: $|N_G(S)| \geq (1 + e)|S|$

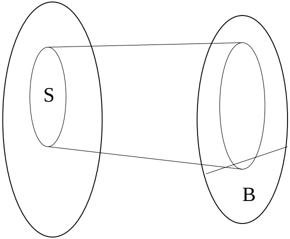


Bipartite expanders

(s, e)-**expander**: $|N_G(S)| \geq (1 + e)|S|$



(s, e, b)-**resilient expander**: $\Pr_B \left[\, |N_{G \setminus B}(S)| \geq (1 + e)|S| \,\right] \geq 1/2$



Parameters

Fix $|L| = 2n$, $|R| = n$, $s = n^{1-\Omega(1)}$, $e = 1$, and $b = \Theta(n/\log n)$.

choose a random left-degree-d_L bipartite random graph G

Parameters

Fix $|L| = 2n$, $|R| = n$, $s = n^{1-\Omega(1)}$, $e = 1$, and $b = \Theta(n/\log n)$.

choose a random left-degree-d_L bipartite random graph G

Question:

for what d_L's is G an (s, e, b)-resilient expander?

Parameters

Fix $|L| = 2n$, $|R| = n$, $s = n^{1-\Omega(1)}$, $e = 1$, and $b = \Theta(n/\log n)$.

choose a random left-degree-d_L bipartite random graph G

Question:

for what d_L's is G an (s, e, b)-resilient expander?

Lemma:

$$\Omega\left(\frac{\log n}{\log\log n}\right) \leq d_L \leq O(\log n)$$

Interesting part of lemma

Lemma:

$$\Omega\left(\frac{\log n}{\log\log n}\right) \stackrel{(1)}{\leq} d_L \stackrel{(2)}{\leq} O(\log n)$$

Proof of (1): Assume $d_L = o(\log n / \log\log n)$.

Interesting part of lemma

Lemma:

$$\Omega\left(\frac{\log n}{\log\log n}\right) \overset{(1)}{\leq} d_L \overset{(2)}{\leq} O(\log n)$$

Proof of (1): Assume $d_L = o(\log n / \log\log n)$.

$\Rightarrow$ also $d_R = O(\log n)$ whp,

Interesting part of lemma

Lemma:

$$\Omega\left(\frac{\log n}{\log\log n}\right) \overset{(1)}{\leq} d_L \overset{(2)}{\leq} O(\log n)$$

Proof of (1): Assume $d_L = o(\log n / \log\log n)$.

$\Rightarrow$ also $d_R = O(\log n)$ whp,

$\Rightarrow$ there are $\Omega(n/(\log n)^2)$ vertices with disjoint neighborhoods,

Interesting part of lemma

Lemma:

$$\Omega\left(\frac{\log n}{\log\log n}\right) \overset{(1)}{\leq} d_L \overset{(2)}{\leq} O(\log n)$$

Proof of (1): Assume $d_L = o(\log n/\log\log n)$.

$\Rightarrow$ also $d_R = O(\log n)$ whp,

$\Rightarrow$ there are $\Omega(n/(\log n)^2)$ vertices with disjoint neighborhoods,

$\Rightarrow$ probability that every L-vertex has some neighbor outside B is

$$\leq \left(1 - \left(\frac{1}{\Theta(\log n)}\right)^{o(\log n/\log\log n)}\right)^{n/(\log n)^2} = o(1).$$

Q.E.D.

(thanks)