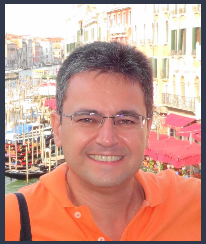
Sub-exponential Approximation Schemes: From Dense to Almost-Sparse

Dimitris Fotakis Michael Lampis Vangelis Paschos



NTU Athens Université Paris Dauphine

Nov 5, 2015

Overview

Things you will hear in this talk:

- CSPs, Max-Cut, Max-3-SAT,...
  Topic: Approximating Max-k-CSP (k fixed)

- Satisfiability
- Lower Bounds
- Tight Results
- Exponential-time Algorithms

Overview

Things you will hear in this talk:

- CSPs, Max-Cut, Max-3-SAT,. . .
  Topic: Approximating Max-k-CSP (k fixed)

- Satisfiability
- Lower Bounds
- Tight Results
- Exponential-time Algorithms

Overview

Things you will hear in this talk:

- CSPs, Max-Cut, Max-3-SAT,...
  Topic: Approximating Max-k-CSP (k fixed)
- This is hard in polynomial time.
  Solution: **sub**-exponential time.

- Satisfiability
- Lower Bounds
- Tight Results
- Exponential-time Algorithms

Overview

Things you will hear in this talk:

- CSPs, Max-Cut, Max-3-SAT,. . .
  Topic: Approximating Max-k-CSP (k fixed)
- This is hard in polynomial time.
  Solution: **sub**-exponential time.

- Satisfiability
- Lower Bounds
- Tight Results
- Exponential-time Algorithms

Overview

Things you will hear in this talk:

- CSPs, Max-Cut, Max-3-SAT,...
  Topic: Approximating Max-k-CSP (k fixed)
- This is hard in polynomial time.
  Solution: **sub**-exponential time.
- Running time 2^{n^c}.
  Question: Is c optimal?
  Use ETH to prove it.

- Satisfiability ✓
- Lower Bounds
- Tight Results
- Exponential-time Algorithms ✓

Overview

Things you will hear in this talk:

- CSPs, Max-Cut, Max-3-SAT,...
  Topic: Approximating Max-k-CSP (k fixed)
- This is hard in polynomial time.
  Solution: **sub**-exponential time.
- Running time 2^{n^c}.
  Question: Is c optimal?
  Use ETH to prove it.

- Satisfiability ✓
- Lower Bounds ✓
- Tight Results
- Exponential-time Algorithms ✓

Overview

Things you will hear in this talk:

- CSPs, Max-Cut, Max-3-SAT,. . .
  Topic: Approximating Max-k-CSP (k fixed)
- This is hard in polynomial time.
  Solution: **sub**-exponential time.
- Running time 2^{n^c}.
  Question: Is c optimal?
  Use ETH to prove it.

- Satisfiability ✓
- Lower Bounds ✓
- Tight Results ✓
- Exponential-time Algorithms ✓

Overview

Things you will hear in this talk:

- CSPs, Max-Cut, Max-3-SAT,...
  Topic: Approximating Max-k-CSP (k fixed)
- This is hard in polynomial time.
  Solution: **sub**-exponential time.
- Running time 2^{n^c}.
  Question: Is c optimal?
  Use ETH to prove it.

- Satisfiability ✓
- Lower Bounds ✓
- Tight Results ✓
- Exponential-time Algorithms ✓

Motivation – Sub-Exponential Approximation

(50 years in a slide)

- Efficient = Poly-time ('60s)

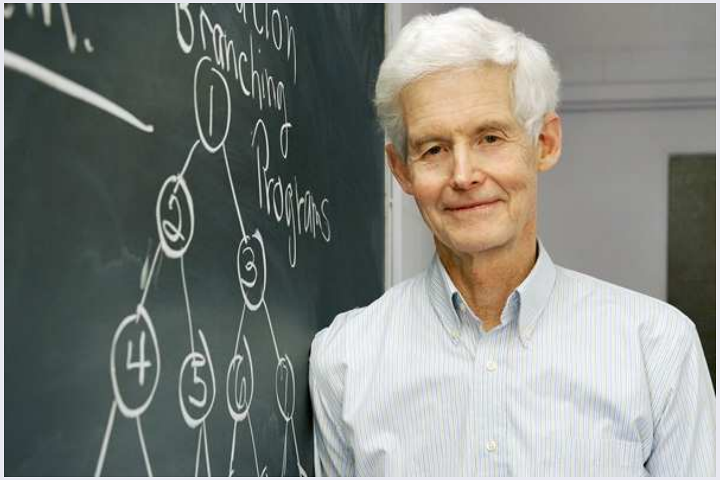
Jack Edmonds

Juris Hartmanis

Richard Stearns

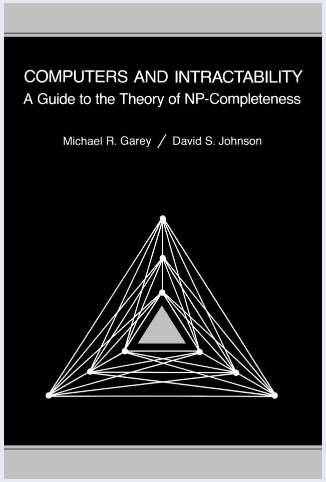
(50 years in a slide)

- Efficient = Poly-time ('60s)
- Everything is NP-hard!($*$) ('70s)



Stephen Cook

Richard Karp

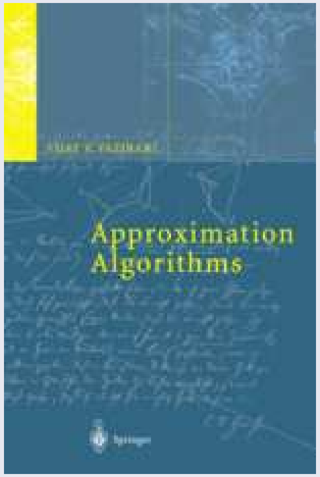


Garey& Johnson

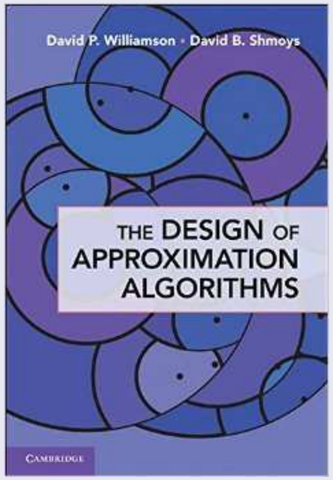
(50 years in a slide)

- Efficient = Poly-time ('60s)
- Everything is NP-hard!(*) ('70s)
- So, we should approximate ('80s)

David Johnson



V. Vazirani

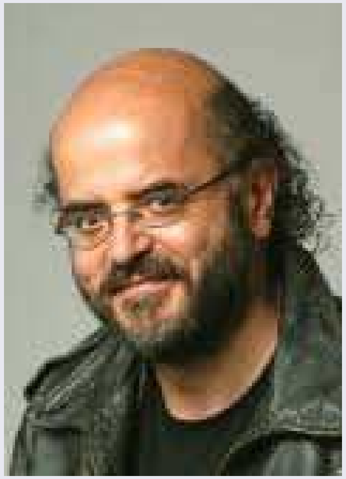


Williamson&Shmoys

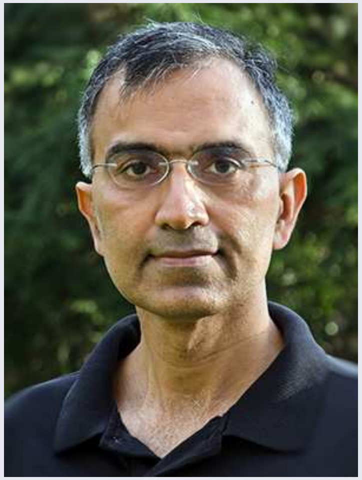
Motivation – Sub-Exponential Approximation

(50 years in a slide)

- Efficient = Poly-time ('60s)
- Everything is NP-hard!(*) ('70s)
- So, we should approximate ('80s)
- Everything is APX-hard!(*) ('90s)



Christos Papadimitriou



Sanjeev Arora

Johan Håstad

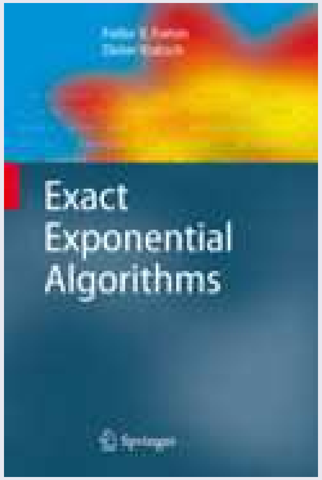
Motivation – Sub-Exponential Approximation

(50 years in a slide)

- Efficient = Poly-time ('60s)
- Everything is NP-hard!(*) ('70s)
- So, we should approximate ('80s)
- Everything is APX-hard!(*) ('90s)
- More than poly-time? Everything ETH-hard ('00s)

Mike Fellows

Russell Impagliazzo



Fomin&Kratsch

Motivation – Sub-Exponential Approximation

(50 years in a slide)

- Efficient = Poly-time ('60s)
- Everything is NP-hard!(*) ('70s)
- So, we should approximate ('80s)
- Everything is APX-hard!(*) ('90s)
- More than poly-time? Everything ETH-hard ('00s)

Bottom line:

- Most problems hard to solve exactly in $2^{o(n)}$ time
- Most problems hard to approximate in $n^{O(1)}$ time
- $\rightarrow$ Perhaps we can approximate in $2^{o(n)}$ time?

Dead on Arrival?

- Better than $3/2$ for TSP, $4/3$ for Max-3-DM, $7/8$ for Max-3-SAT,. . . , in **sub-exponential time**?

Dead on Arrival?

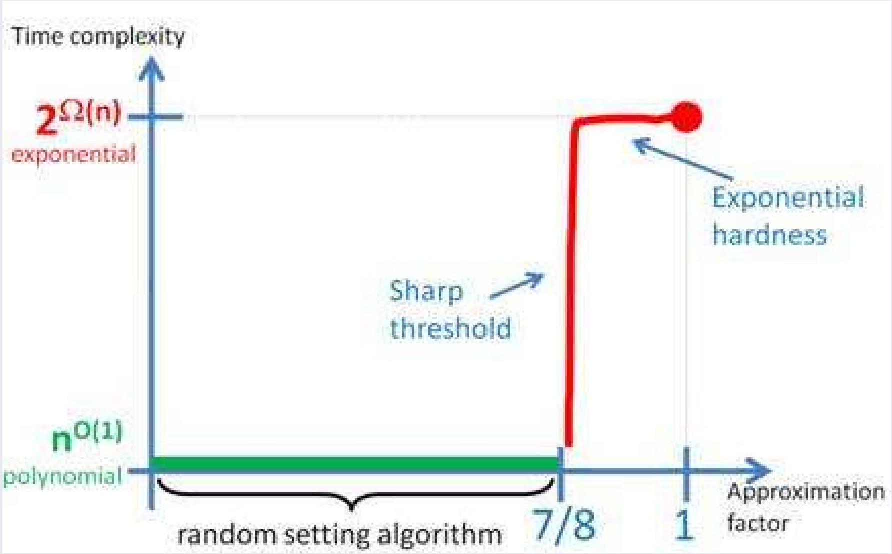
- Better than $3/2$ for TSP, $4/3$ for Max-3-DM, $7/8$ for Max-3-SAT,. . . , in **sub-exponential time**?

Probably won't work

(at least for Max-3-SAT)

Dead on Arrival?

- Better than $3/2$ for TSP, $4/3$ for Max-3-DM, $7/8$ for Max-3-SAT,. . . , in **sub-exponential time**?



Almost-linear PCPs (Moshkovitz& Raz) and P-time hardness (Håstad) give tight inapproximability for Max-3-SAT even for $2^{n^{1-\epsilon}}$ time.

(Credit: Dana Moshkovitz)

Dead on Arrival?

- Better than $3/2$ for TSP, $4/3$ for Max-3-DM, $7/8$ for Max-3-SAT,..., in **sub-exponential time**?

  If this is the “normal” behavior of APX problems, what’s the point of sub-exponential approximation?

  - Is this the “normal” behavior?
  - What about problems outside APX?
    - E.g., r-approximation in time $2^{n/r}$ for Ind. Set ([Chalermsook, Laekhanukit, Nanongkai ’13])
    - r-approximation in time $2^{n/r^2}$ for Max Minimal VC
      $\log(n/r)$-approximation in time $2^{n/r}$ for ATSP ([Bonnet,L.,Paschos arxiv ’15]).
  - **What else?**

Strategy

- We **cannot** get better than 7/8 for Max-3-SAT in sub-exp time (under ETH).
- We will therefore try to get something else:

Strategy

- We **cannot** get better than 7/8 for Max-3-SAT in sub-exp time (under ETH).
- We will therefore try to get something else:

An island of tractability:

- Max-k-CSP admits a PTAS (a $(1-\epsilon)$-approximation for all $\epsilon > 0$) for **dense** instances
- (Arora, Karger, Karpinski '99), (de la Vega '96)

Strategy

- We **cannot** get better than 7/8 for Max-3-SAT in sub-exp time (under ETH).
- We will therefore try to get something else:

An island of tractability:

- Max-k-CSP admits a PTAS (a $(1-\epsilon)$-approximation for all $\epsilon > 0$) for **dense** instances
- (Arora, Karger, Karpinski ’99), (de la Vega ’96)

Extending the island:

- We give a version of the AKK scheme which **can handle sparser instances**, at the expense of needing **sub-exponential time**.
- Our scheme provides a smooth trade-off
  - For dense instances we get a PTAS
  - As instances gradually get more sparse, we need more time…
  - …until our scheme does not work any more

Summary of results

For any $\epsilon > 0$, $\delta \in [0, 1]$ and fixed $k \geq 2$ we have the following:

- Given a Max-k-CSP instance with $n^{k-1+\delta}$ constraints
- We can produce a $(1 - \epsilon)$-approximate solution
- In time $2^{O(n^{1-\delta} \ln n/\epsilon^3)}$

Summary of results

For any $\epsilon > 0$, $\delta \in [0, 1]$ and fixed $k \geq 2$ we have the following:

- Given a Max-k-CSP instance with $n^{k-1+\delta}$ constraints
- We can produce a $(1 - \epsilon)$-approximate solution
- In time $2^{O(n^{1-\delta} \ln n / \epsilon^3)}$

- Note: This includes the AKK PTAS as a special case ($\delta = 1$)
- Advantage: we provide a smooth trade-off from the "easy case" (dense instances) to more general cases

Summary of results

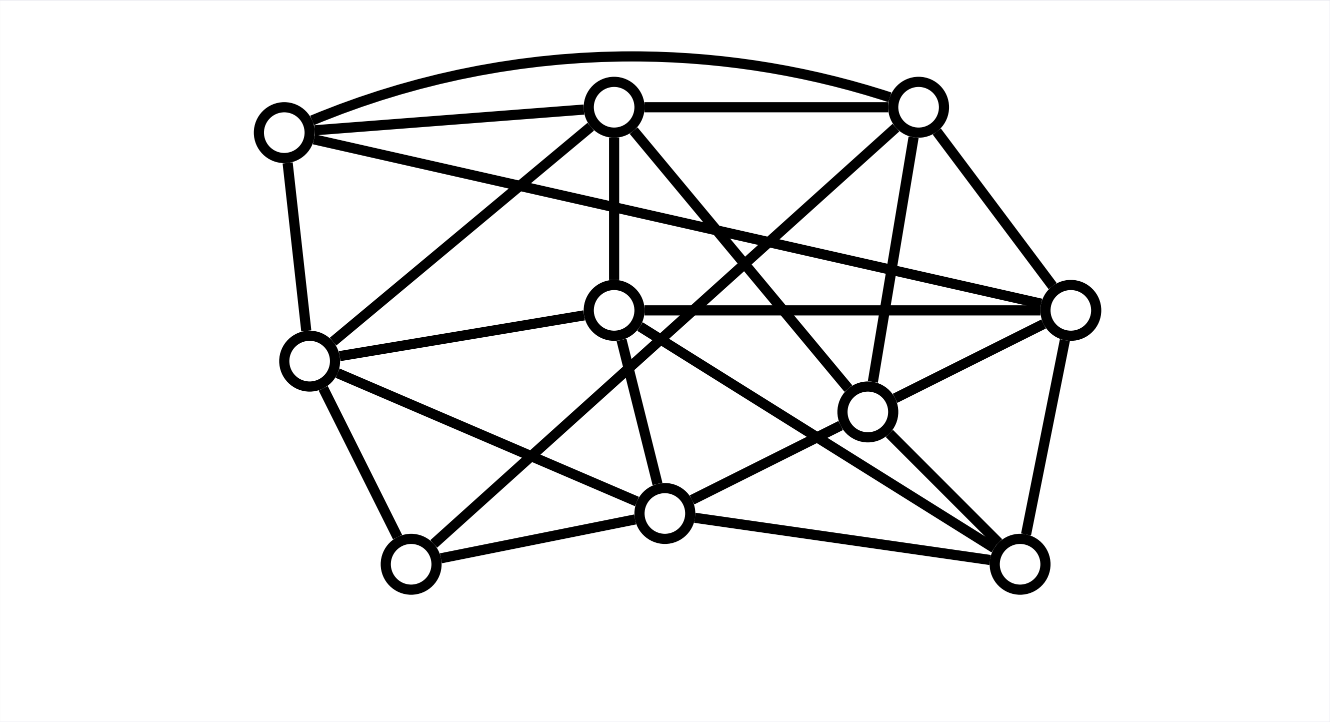
For any $\epsilon > 0$, $\delta \in [0,1]$ and fixed $k \geq 2$ we have the following:

- Given a Max-k-CSP instance with $n^{k-1+\delta}$ constraints
- We can produce a $(1-\epsilon)$-approximate solution
- In time $2^{O(n^{1-\delta} \ln n/\epsilon^3)}$

- Note: This includes the AKK PTAS as a special case ($\delta = 1$)
- Advantage: we provide a smooth trade-off from the “easy case” (dense instances) to more general cases

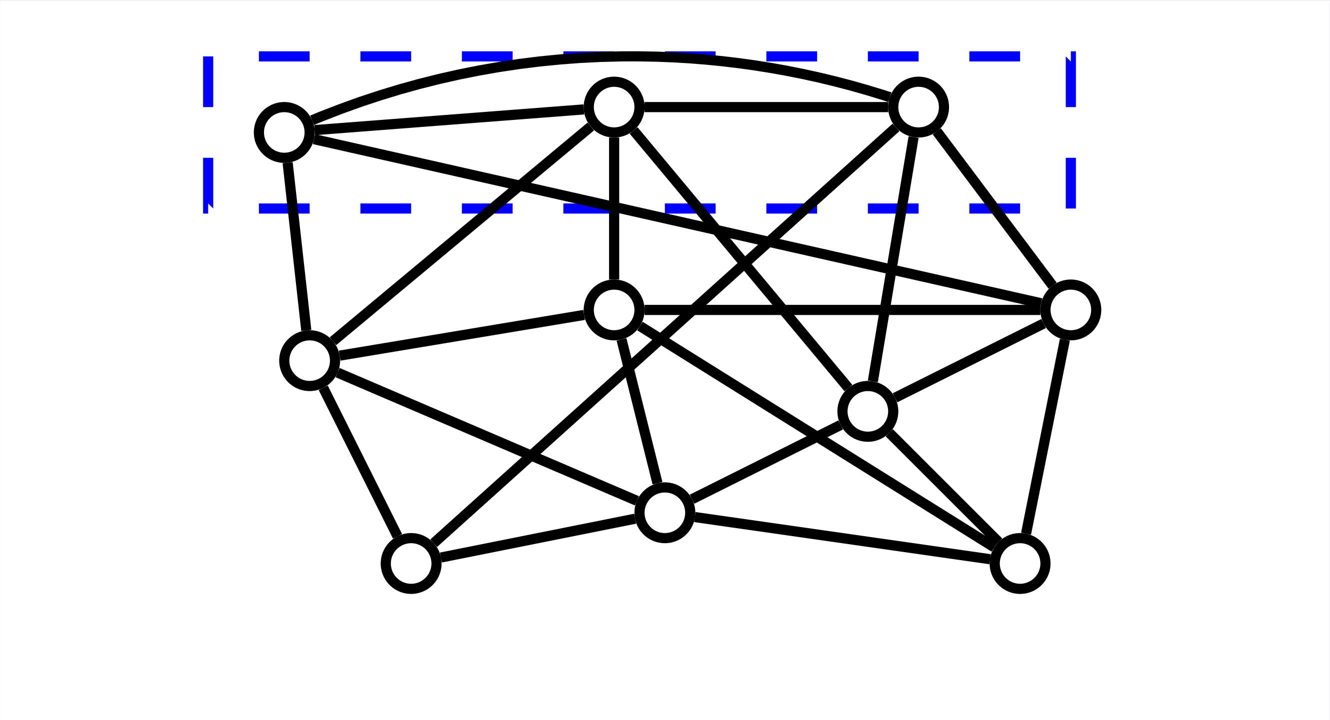
- We will also give some “tight” bounds, ruling out natural possible improvements.

Basic scheme (Max Cut)



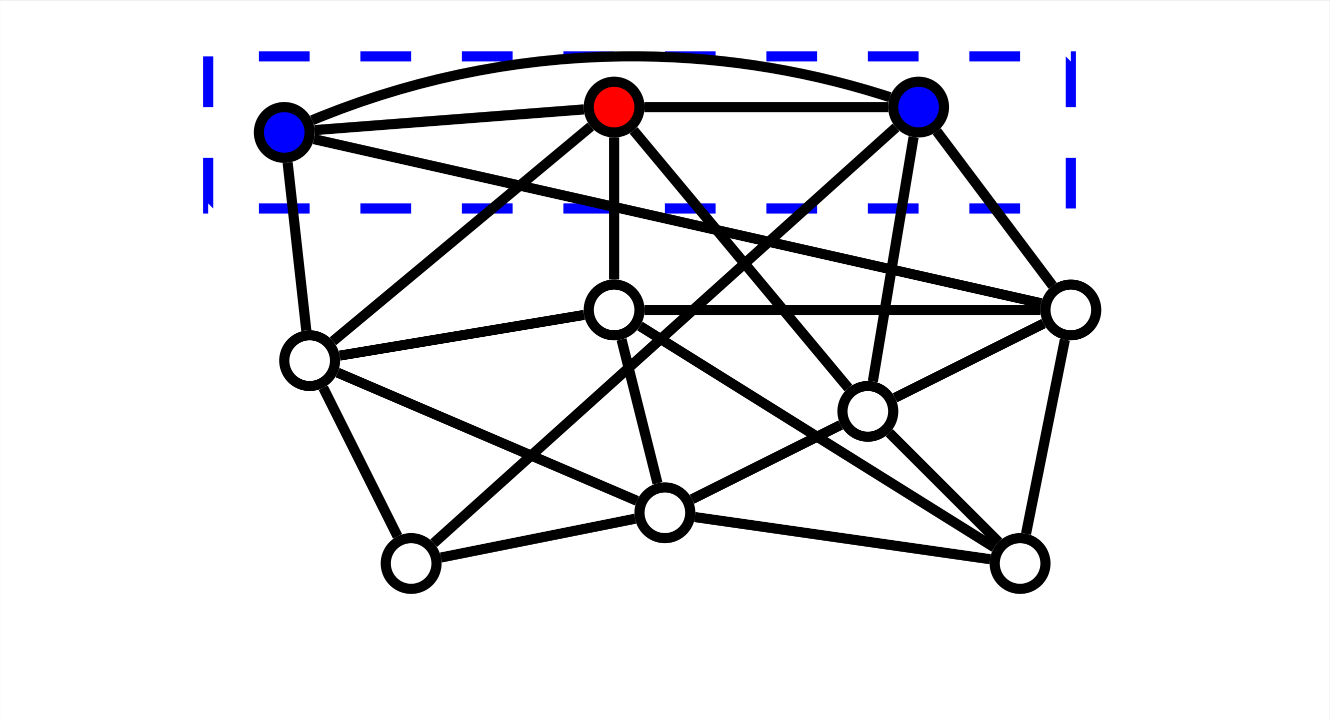
We are given a dense graph for which we want to find a large cut

Basic scheme (Max Cut)



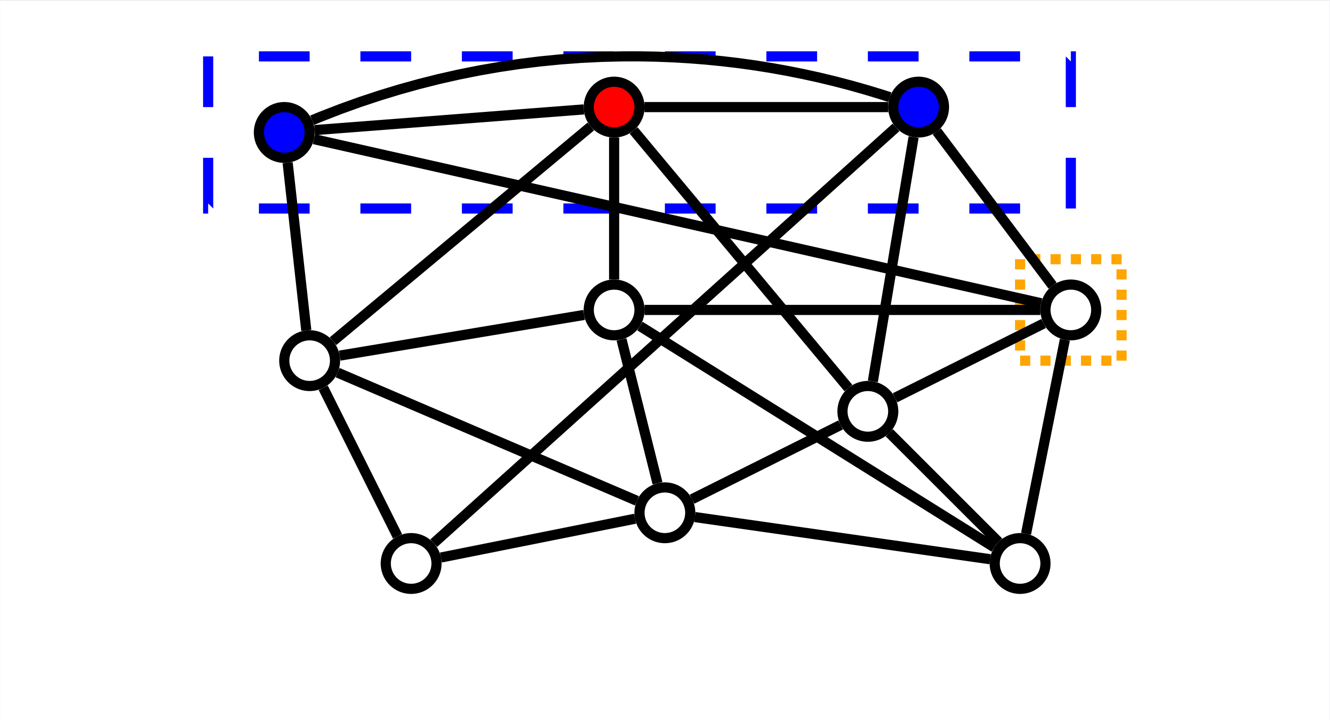
Randomly select a “sample” of its vertices

Basic scheme (Max Cut)



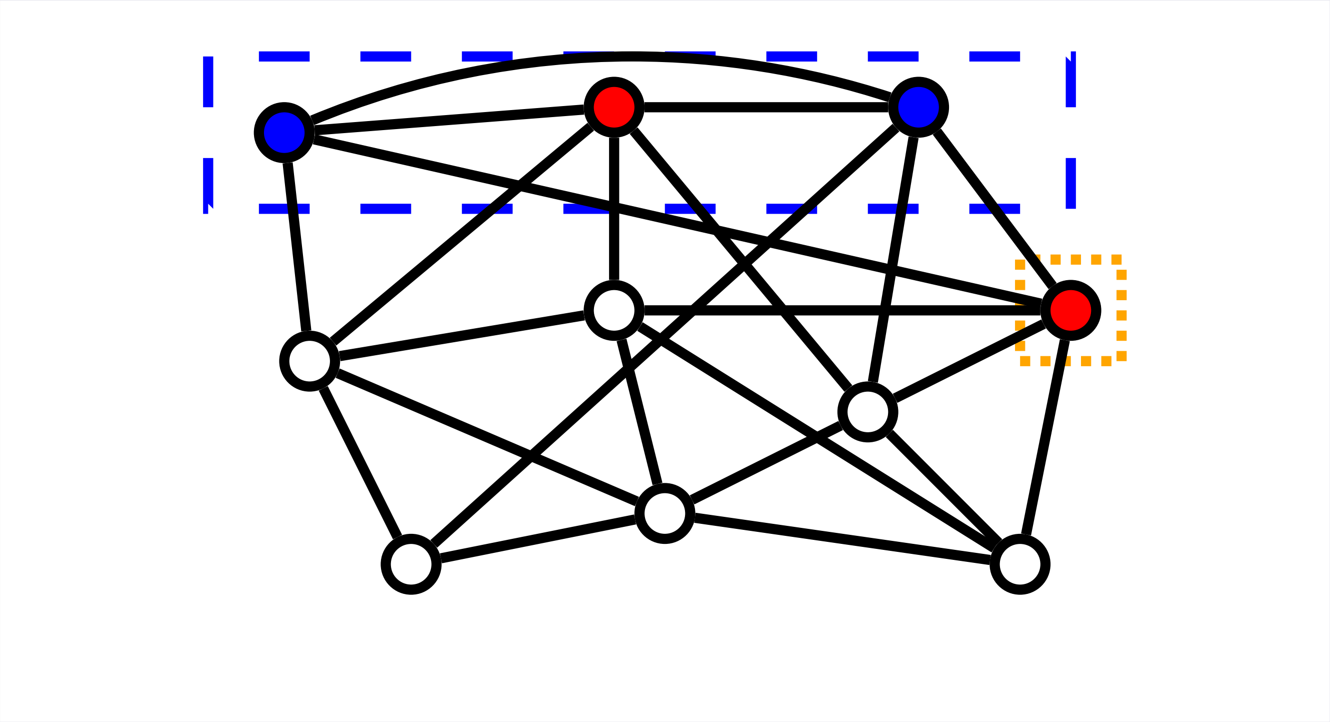
Guess their correct partition

Basic scheme (Max Cut)



For every vertex outside the sample, examine its neighbors in the sample

Basic scheme (Max Cut)



Greedily set its value depending on this neighborhood

Basic scheme (Max Cut)

- The sample we select has size $O(\log n)$ (hidden constants depend on degree and ϵ)
- $\rightarrow$ running time $n^{O(1)}$ (will try all partitions of sample)

Basic scheme (Max Cut)

- The sample we select has size $O(\log n)$ (hidden constants depend on degree and ϵ)
- $\rightarrow$ running time $n^{O(1)}$ (will try all partitions of sample)

Why this works (intuitively):

- Because graph is dense $\rightarrow$ every vertex outside sample S has many neighbors in S
- $\rightarrow$ examining $N(u) \cap S$ is (whp) a good representation of $N(u)$ in the optimal solution
- If a vertex in $V \setminus S$ has $>> 50\%$ of its neighbors on one side in the optimal solution, it will (whp) have $>> 50\%$ of its neighbors on that side in S

(de la Vega '96)

General scheme (Max-k-CSP)

Max Cut:

$$\max \sum_{(i,j)\in E} x_i(1-x_j) + x_j(1-x_i)$$

General scheme (Max-k-CSP)

Max-2-SAT:

$$\max \sum_{(i,j)\in C} x_i(1-x_j) + x_j(1-x_i) + x_i x_j$$

General scheme (Max-k-CSP)

Max-3-SAT:

$$\max \sum_{(i,j,k)\in C} x_i(1-x_j)(1-x_k) + (1-x_i)x_j(1-x_k) + \ldots + x_i x_j x_k$$

General scheme (Max-k-CSP)

Max-k-CSP:

$$\max p(\vec{x})$$

where $p()$ is a degree k polynomial.

The AKK scheme offers a PTAS that finds an assignment almost maximizing p when the polynomial has at least $\Omega(n^k)$ terms.

General scheme (Max-k-CSP)

Max Cut:

$$\max \sum_{(i,j)\in E} x_i(1-x_j) + x_j(1-x_i)$$

General scheme (Max-k-CSP)

Max Cut:

$$\max \sum_{(i,j)} c_{ij} x_i x_j + \sum_i c_i x_i + C$$

General scheme (Max-k-CSP)

Max Cut:

$$\max \sum_i x_i r_i$$

where $r_i(\vec{x} - x_i)$ is the (linear) polynomial of the remaining variables I obtain if I factor out x_i.

General scheme (Max-k-CSP)

Max Cut:

$$\max \sum_i x_i r_i$$

where $r_i(\vec{x} - x_i)$ is the (linear) polynomial of the remaining variables I obtain if I factor out x_i.

Main idea: Estimate the values of the r_i's using brute force on a small sample.

General scheme (Max-k-CSP)

Max Cut:

$$\max \sum_i x_i r_i$$

s.t.

$$\hat{r}_i - \epsilon n \leq \quad \sum_{j \in N(i)} c_{ij} x_j \quad \leq \hat{r}_i + \epsilon n$$

where $\hat{r}_i$ is the estimate I have for r_i.

This is now a **linear** program.

General scheme (Max-k-CSP)

Max Cut:

$$\max \sum_i x_i r_i$$

Summary of algorithm:

- Estimate the r_i values using a sample
  - Need large enough sample to guarantee $\hat{r}_i \approx r_i$
  - This turns QIP $\rightarrow$ ILP
- Solve fractional relaxation of ILP
- Round solution

Sub-exponential Extension (Max Cut)

Summary of algorithm:

- Estimate the r_i values using a sample
  - Need large enough sample to guarantee $\hat{r}_i \approx r_i$
  - This turns QIP $\to$ ILP
- Solve fractional relaxation of ILP
- Round solution

Sub-exponential Extension (Max Cut)

Summary of algorithm:

- Estimate the r_i values using a sample
  - Need large enough sample to guarantee $\hat{r}_i \approx r_i$
  - This turns QIP $\rightarrow$ ILP
- Solve fractional relaxation of ILP
- Round solution

Main idea: **Use larger sample**

Sub-exponential Extension (Max Cut)

Summary of algorithm:

- Estimate the r_i values using a sample
  - Need large enough sample to guarantee $\hat{r}_i \approx r_i$
  - This turns QIP $\rightarrow$ ILP
- Solve fractional relaxation of ILP
- Round solution

Main idea: **Use larger sample**

- Suppose graph has average degree $\Delta = n^\delta$
- We sample $\frac{n \log n}{\Delta} = n^{1-\delta} \log n$ vertices
- $\rightarrow$ whp $\hat{r}_i \approx r_i$.

Sub-exponential Extension (Max Cut)

Summary of algorithm:

- Estimate the r_i values using a sample
  - Need large enough sample to guarantee $\hat{r}_i \approx r_i$
  - This turns QIP $\rightarrow$ ILP
- Solve fractional relaxation of ILP
- Round solution

Main idea: **Use larger sample**
We are almost done!

- Must prove sample size enough for $\hat{r}_i$
  - Pitfall: Additive error ϵn no longer negligible!
- Must prove rounding step still works

Don't worry, it all works!

General scheme $k \geq 3$

Summary so far $k = 2$:

- AKK: Average degree $\Omega(n)$, sample of $O(\log n)$ vertices
- Extension: Average degree n^{δ}, sample of $n^{1-\delta} \log n$
- $\rightarrow$ in time $2^{\sqrt{n}}$ can "solve" Max-Cut for $|E| \geq n^{1.5}$

General scheme $k \geq 3$

Summary so far $k = 2$:

- AKK: Average degree $\Omega(n)$, sample of $O(\log n)$ vertices
- Extension: Average degree n^{δ}, sample of $n^{1-\delta} \log n$
- $\rightarrow$ in time $2^{\sqrt{n}}$ can "solve" Max-Cut for $|E| \geq n^{1.5}$

How about Max-3-SAT?

- In poly time can solve instances with n^3 clauses
- In $2^{\sqrt{n}}$ time can solve instances with … clauses?

General scheme $k \geq 3$

Summary so far $k = 2$:

- AKK: Average degree $\Omega(n)$, sample of $O(\log n)$ vertices
- Extension: Average degree n^{δ}, sample of $n^{1-\delta} \log n$
- $\rightarrow$ in time $2^{\sqrt{n}}$ can "solve" Max-Cut for $|E| \geq n^{1.5}$

How about Max-3-SAT?

- In poly time can solve instances with n^3 clauses
- In $2^{\sqrt{n}}$ time can solve instances with $n^{2.5}$ clauses

General scheme $k \geq 3$

AKK scheme for $k \geq 3$

- Write $p(\vec{x})$ as $\sum_i x_i r_i$
- Each r_i has degree $k-1$
- Write $r_i = \sum_j x_j r_{ij}$
- Each r_{ij} has degree $k-2$
- ...
- Until we get to linear $\rightarrow$ write ILP

General scheme $k \geq 3$

AKK scheme for $k \geq 3$

- Write $p(\vec{x})$ as $\sum_i x_i r_i$
- Each r_i has degree $k-1$
- Write $r_i = \sum_j x_j r_{ij}$
- Each r_{ij} has degree $k-2$
- $\ldots$
- Until we get to linear $\rightarrow$ write ILP

Note: In order for this to work, all $r_{ij\ldots}$ polynomials must be **dense**

- This is true if original polynomial was dense.

General scheme $k \geq 3$

AKK scheme for $k \geq 3$

- Write $p(\vec{x})$ as $\sum_i x_i r_i$
- Each r_i has degree $k-1$
- Write $r_i = \sum_j x_j r_{ij}$
- Each r_{ij} has degree $k-2$
- $\ldots$
- Until we get to linear $\rightarrow$ write ILP

- In our scheme, if p has $n^{k-1+\delta}$ terms
- r_i has $n^{k-2+\delta}$ terms
- r_{ij} has $n^{k-3+\delta}$ terms
- $\ldots$

It seems that the “right” density to require is $n^{k-1+\delta}$?

General scheme – summary

- Input: Max-k-CSP isntance with $n^{k-1+\delta}$ constraints
- Algorithm:
  - Sample $2^{n^{1-\delta}\log n/\epsilon^3}$ variables, guess their value
  - Write CSP as a polynomial optimization problem
  - Estimate non-linear coefficients using sample
  - Solve fractional LP
  - Round solution

General scheme – summary

- Input: Max-k-CSP isntance with $n^{k-1+\delta}$ constraints
- Algorithm:
  - Sample $2^{n^{1-\delta}\log n/\epsilon^3}$ variables, guess their value
  - Write CSP as a polynomial optimization problem
  - Estimate non-linear coefficients using sample
  - Solve fractional LP
  - Round solution

- Works for any CSP (for fixed k)
- Covers "all instances" for $k = 2$

General scheme – summary

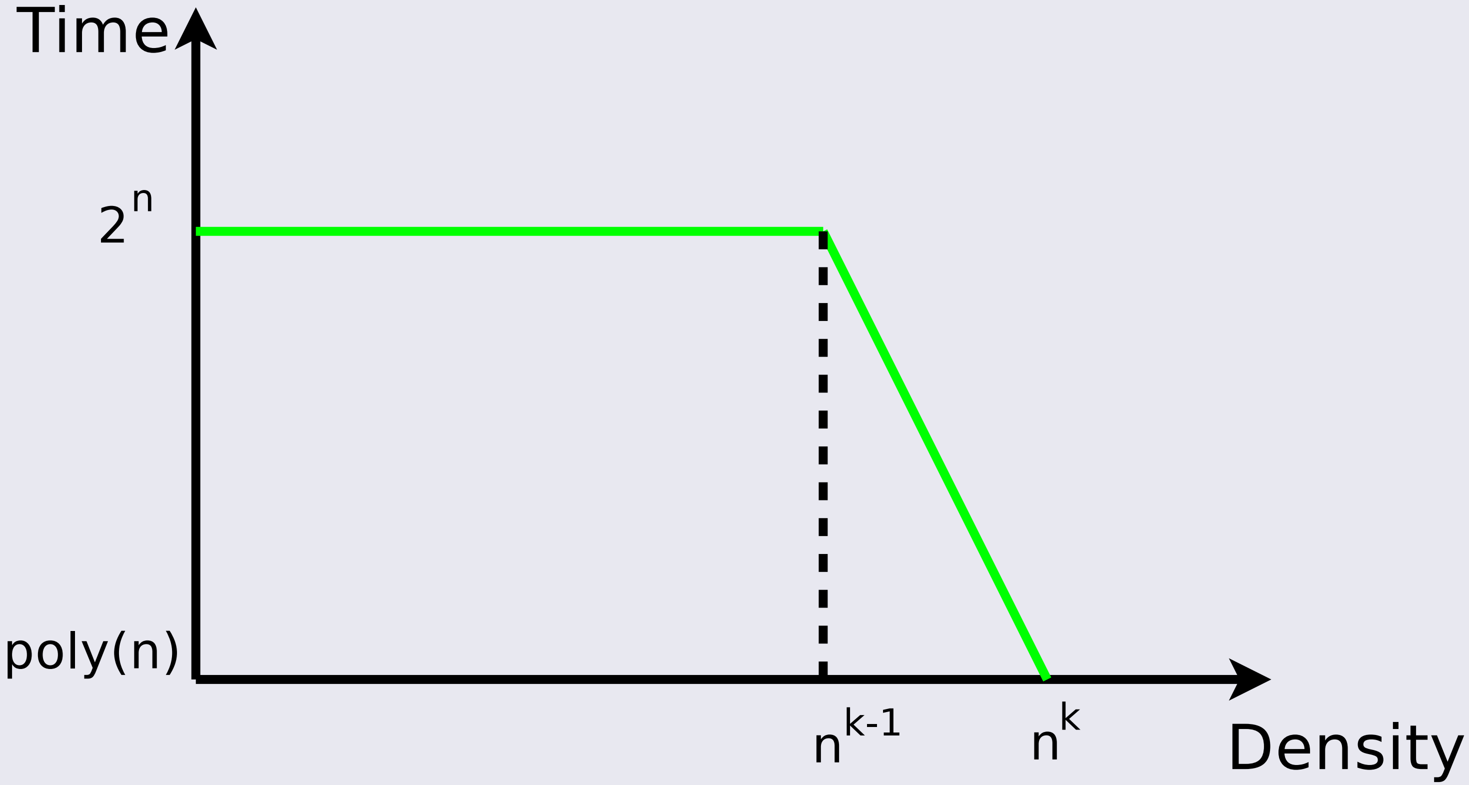
- Input: Max-k-CSP isntance with $n^{k-1+\delta}$ constraints
- Algorithm:
  - Sample $2^{n^{1-\delta}\log n/\epsilon^3}$ variables, guess their value
  - Write CSP as a polynomial optimization problem
  - Estimate non-linear coefficients using sample
  - Solve fractional LP
  - Round solution

- Works for any CSP (for fixed k)
- Covers “all instances” for $k = 2$

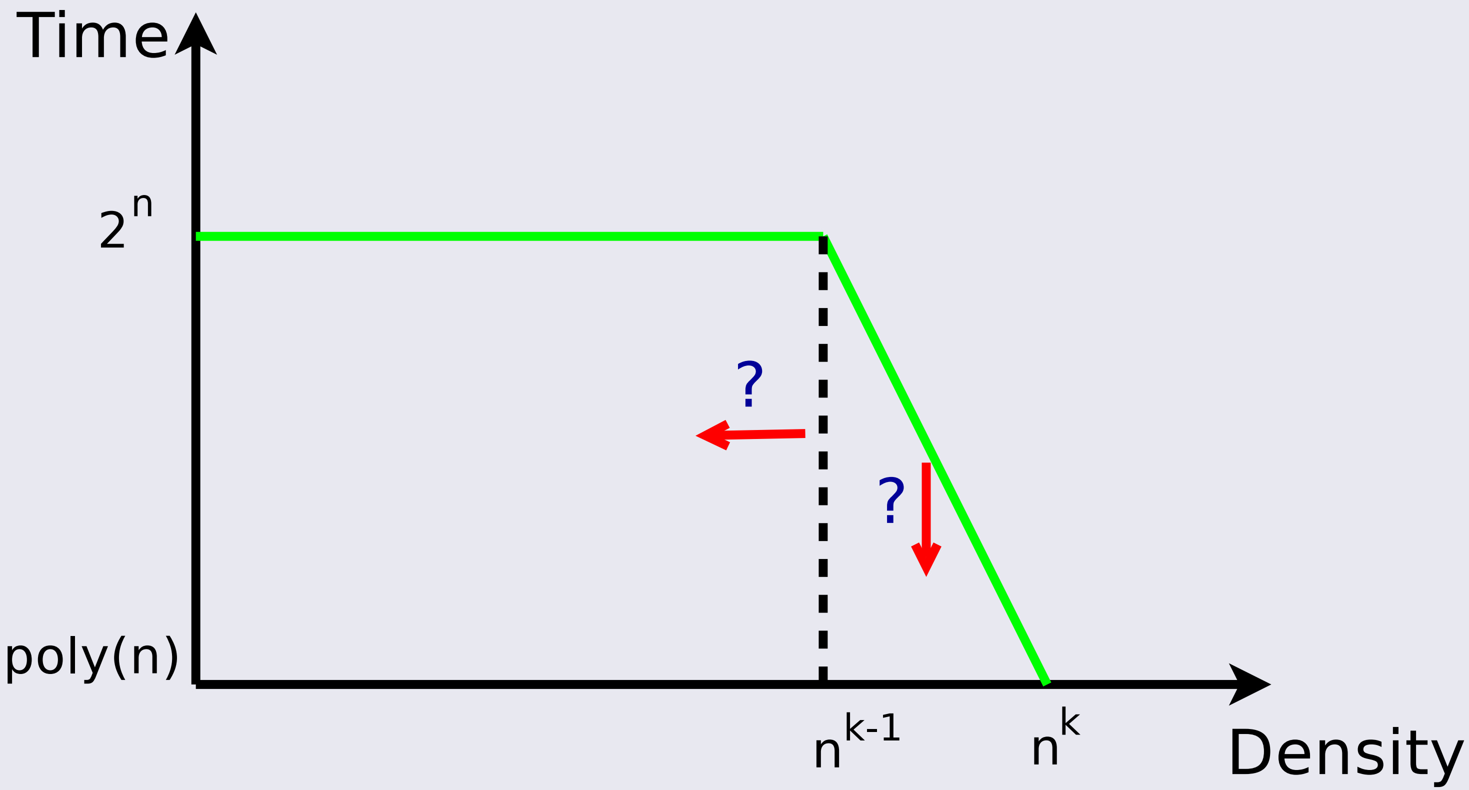
Can we do better?

- Smaller sample/faster running time?
- Handle $k \geq 3$ better?

Summary – with a picture



Complexity/Density trade-off for Max-k-CSP



Possible Improvements? Faster? More general?

Density lower bound

Theorem: Assuming ETH, for all $k \geq 3$, $\exists r < 1$ s.t. $\forall \epsilon > 0$ no algorithm can r-approximate Max-k-SAT with $m \leq n^{k-1}$ in time $2^{n^{1-\epsilon}}$

Density lower bound

Theorem: Assuming ETH, for all $k \geq 3$, $\exists r < 1$ s.t. $\forall \epsilon > 0$ no algorithm can r-approximate Max-k-SAT with $m \leq n^{k-1}$ in time $2^{n^{1-\epsilon}}$

In English: For density less than n^{k-1} we need exponential time to get $(1-\epsilon)$-approximation.

Density lower bound

Theorem: Assuming ETH, for all $k \geq 3$, $\exists r < 1$ s.t. $\forall \epsilon > 0$ no algorithm can r-approximate Max-k-SAT with $m \leq n^{k-1}$ in time $2^{n^{1-\epsilon}}$

Starting Point: Max-2-SAT is "APX-ETH"-hard on instances with $|V| = n$ and $m = O(|V|)$.

Density lower bound

Theorem: Assuming ETH, for all $k \geq 3$, $\exists r < 1$ s.t. $\forall \epsilon > 0$ no algorithm can r-approximate Max-k-SAT with $m \leq n^{k-1}$ in time $2^{n^{1-\epsilon}}$

Starting Point: Max-2-SAT is "APX-ETH"-hard on instances with $|V| = n$ and $m = O(|V|)$.
Proof: ($k = 3$)

- Add n new variables $y_1, \ldots, y_n$
- For each clause $(x_i \vee x_j)$, for each $k \in \{1, \ldots, n\}$ we construct the clauses $(x_i \vee x_j \vee y_k)$ and $(x_i \vee x_j \vee \neg y_k)$
- Gap remains!
- Number of clauses $\approx n^2$

Density lower bound

Theorem: Assuming ETH, for all $k \geq 3$, $\exists r < 1$ s.t. $\forall \epsilon > 0$ no algorithm can r-approximate Max-k-SAT with $m \leq n^{k-1}$ in time $2^{n^{1-\epsilon}}$

Starting Point: Max-2-SAT is "APX-ETH"-hard on instances with $|V| = n$ and $m = O(|V|)$.
Proof: ($k = 3$)

- Add n new variables $y_1, \ldots, y_n$
- For each clause $(x_i \vee x_j)$, for each $k \in \{1, \ldots, n\}$ we construct the clauses $(x_i \vee x_j \vee y_k)$ and $(x_i \vee x_j \vee \neg y_k)$
- Gap remains!
- Number of clauses $\approx n^2$

Reduction similar for $k > 3$

Running time lower bound

Theorem: Assuming ETH, for all $\delta \in (0,1)$, $\exists r < 1$ s.t. $\forall \epsilon > 0$ no algorithm can r-approximate Max Cut with $|E| = n^{1+\delta}$ in time $2^{n^{1-\delta-\epsilon}}$

Running time lower bound

Theorem: Assuming ETH, for all $\delta \in (0,1)$, $\exists r < 1$ s.t. $\forall \epsilon > 0$ no algorithm can r-approximate Max Cut with $|E| = n^{1+\delta}$ in time $2^{n^{1-\delta-\epsilon}}$

In English: Our sample size is optimal. For density n^{δ} we need time $2^{n^{1-\delta}}$.

Running time lower bound

Theorem: Assuming ETH, for all $\delta \in (0,1)$, $\exists r < 1$ s.t. $\forall \epsilon > 0$ no algorithm can r-approximate Max Cut with $|E| = n^{1+\delta}$ in time $2^{n^{1-\delta-\epsilon}}$

Starting Point: Max Cut is "APX-ETH"-hard on instances with $|V| = n$ and $|E| = O(|V|)$.

Running time lower bound

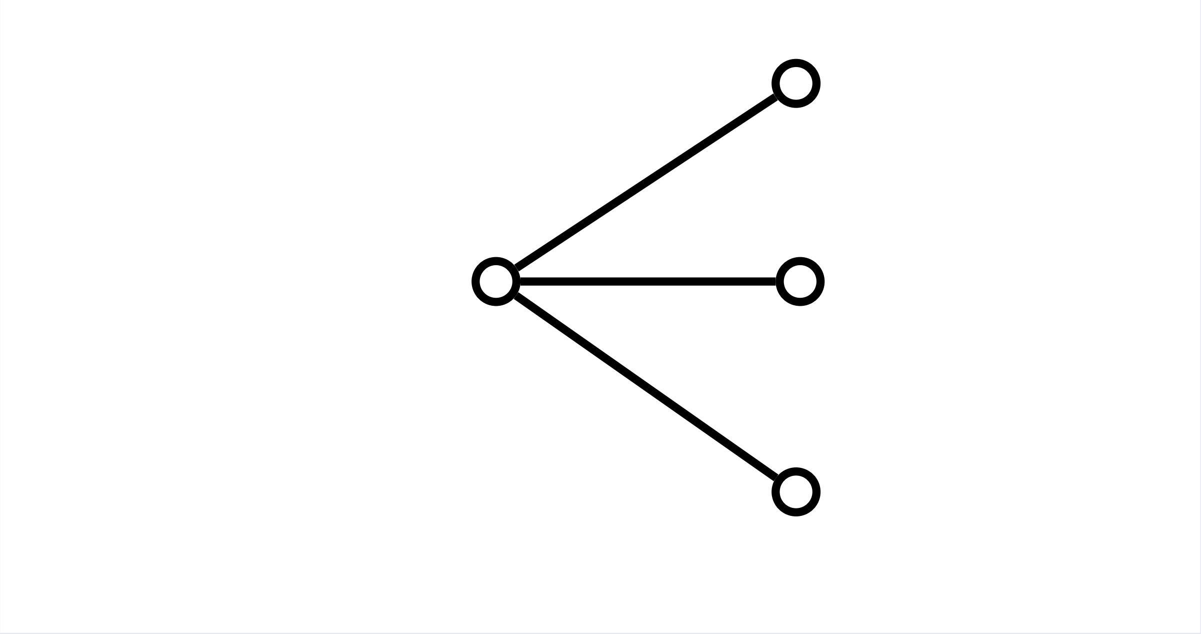
Theorem: Assuming ETH, for all $\delta \in (0,1)$, $\exists r < 1$ s.t. $\forall \epsilon > 0$ no algorithm can r-approximate Max Cut with $|E| = n^{1+\delta}$ in time $2^{n^{1-\delta-\epsilon}}$

Starting Point: Max Cut is "APX-ETH"-hard on 5-regular instances with $|V| = n$.

Running time lower bound

Theorem: Assuming ETH, for all $\delta \in (0,1)$, $\exists r < 1$ s.t. $\forall \epsilon > 0$ no algorithm can r-approximate Max Cut with $|E| = n^{1+\delta}$ in time $2^{n^{1-\delta-\epsilon}}$

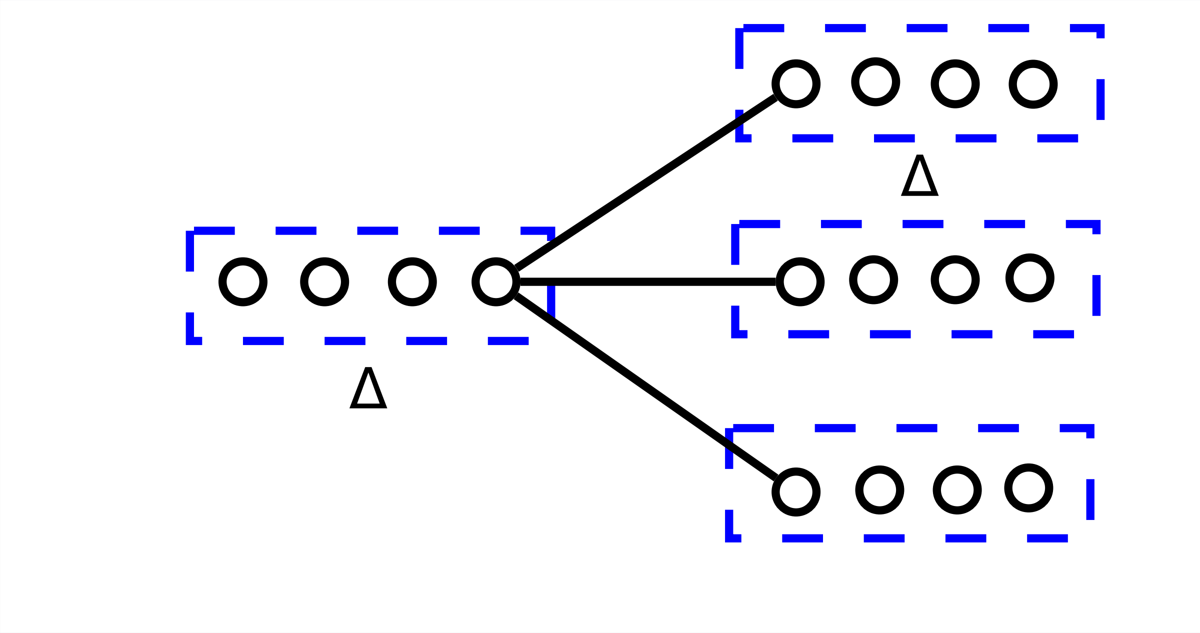
Starting Point: Max Cut is "APX-ETH"-hard on 5-regular instances with $|V| = n$.



Running time lower bound

Theorem: Assuming ETH, for all $\delta \in (0,1)$, $\exists r < 1$ s.t. $\forall \epsilon > 0$ no algorithm can r-approximate Max Cut with $|E| = n^{1+\delta}$ in time $2^{n^{1-\delta-\epsilon}}$

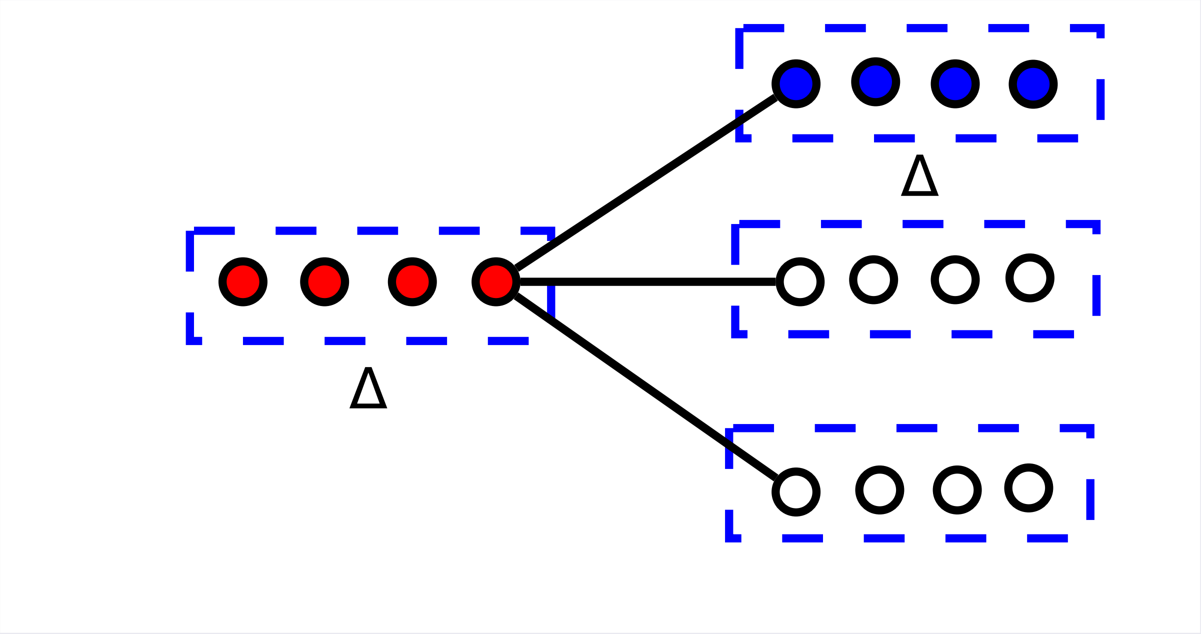
Starting Point: Max Cut is "APX-ETH"-hard on 5-regular instances with $|V| = n$.



Running time lower bound

Theorem: Assuming ETH, for all $\delta \in (0,1)$, $\exists r < 1$ s.t. $\forall \epsilon > 0$ no algorithm can r-approximate Max Cut with $|E| = n^{1+\delta}$ in time $2^{n^{1-\delta-\epsilon}}$

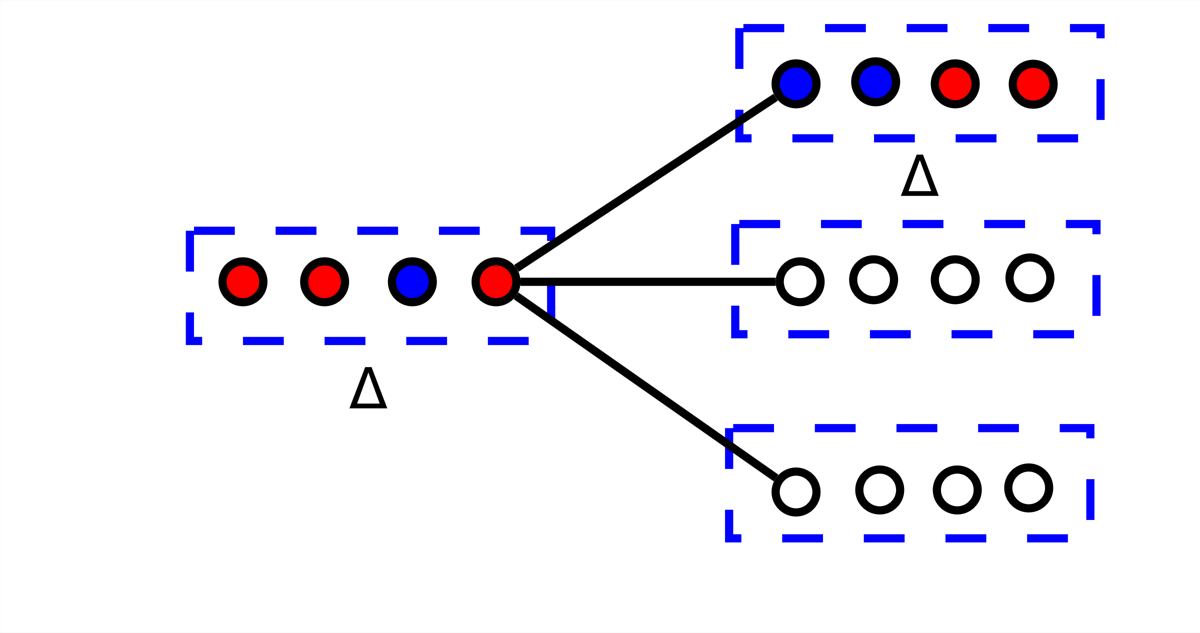
Starting Point: Max Cut is "APX-ETH"-hard on 5-regular instances with $|V| = n$.



Running time lower bound

Theorem: Assuming ETH, for all $\delta \in (0,1)$, $\exists r < 1$ s.t. $\forall \epsilon > 0$ no algorithm can r-approximate Max Cut with $|E| = n^{1+\delta}$ in time $2^{n^{1-\delta-\epsilon}}$

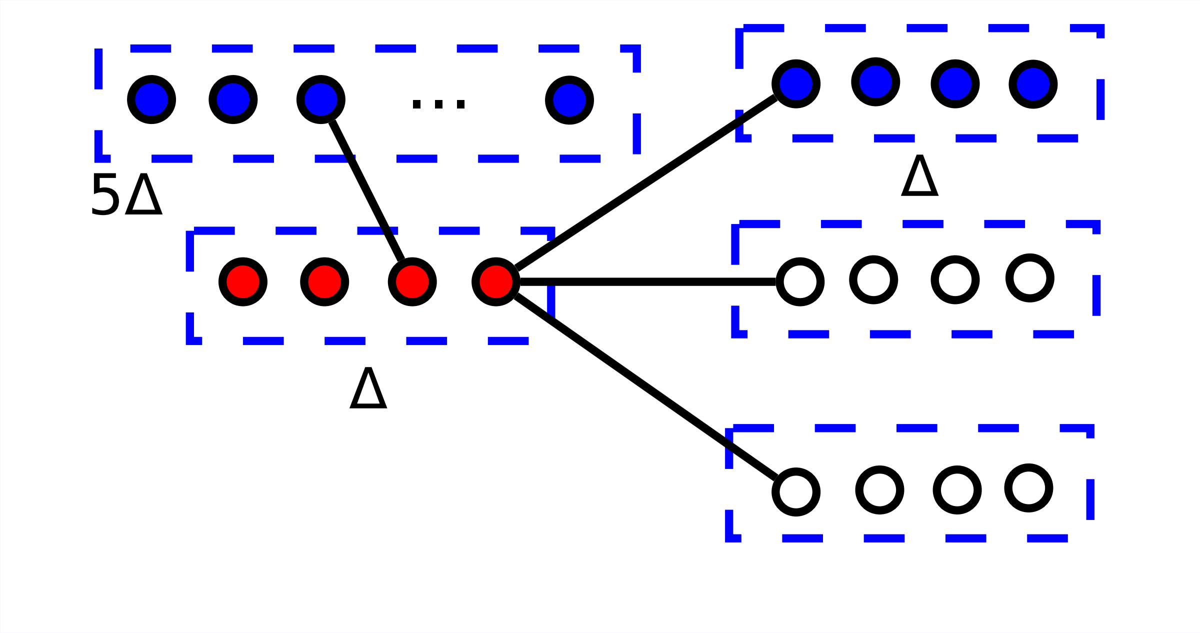
Starting Point: Max Cut is "APX-ETH"-hard on 5-regular instances with $|V| = n$.



Running time lower bound

Theorem: Assuming ETH, for all $\delta \in (0,1)$, $\exists r < 1$ s.t. $\forall \epsilon > 0$ no algorithm can r-approximate Max Cut with $|E| = n^{1+\delta}$ in time $2^{n^{1-\delta-\epsilon}}$

Starting Point: Max Cut is "APX-ETH"-hard on 5-regular instances with $|V| = n$.



Running time lower bound

Theorem: Assuming ETH, for all $\delta \in (0,1)$, $\exists r < 1$ s.t. $\forall \epsilon > 0$ no algorithm can r-approximate Max Cut with $|E| = n^{1+\delta}$ in time $2^{n^{1-\delta-\epsilon}}$

Starting Point: Max Cut is "APX-ETH"-hard on 5-regular instances with $|V| = n$.

- A constant gap remains for any Δ
- $|V'| = n\Delta$, $|E| = n\Delta^2$, Avg. degree $= \Delta$
- If we cound do better than $2^{|V'|/\Delta}$ then $\neg$ETH

Running time lower bound

Theorem: Assuming ETH, for all $\delta \in (0,1)$, $\exists r < 1$ s.t. $\forall \epsilon > 0$ no algorithm can r-approximate Max Cut with $|E| = n^{1+\delta}$ in time $2^{n^{1-\delta-\epsilon}}$

Starting Point: Max Cut is "APX-ETH"-hard on 5-regular instances with $|V| = n$.

- A constant gap remains for any Δ
- $|V'| = n\Delta$, $|E| = n\Delta^2$, Avg. degree $= \Delta$
- If we cound do better than $2^{|V'|/\Delta}$ then $\neg$ETH

- **Bonus:** The two reductions compose! Optimal running times everywhere!

Extensions: k-Densest

- Find k vertices that induce max edges
- AKK scheme works for $k = \Theta(n)$
  - Reason: Then $\text{OPT} = \Theta(n^2)$

Extensions: k-Densest

- Find k vertices that induce max edges
- AKK scheme works for $k = \Theta(n)$
  - Reason: Then $\mathrm{OPT} = \Theta(n^2)$
- How to get something for any k?
- Simple win/win algorithm
  - If k “large”, we can run our algorithm
  - If k “small”, brute force $\binom{n}{k}$
- Balancing gives $2^{n^{1-\delta/3} \log n/\epsilon^3}$ time for $|E| = n^{1+\delta}$

Extensions: k-Densest

- Find k vertices that induce max edges
- AKK scheme works for $k = \Theta(n)$
  - Reason: Then $\mathrm{OPT} = \Theta(n^2)$
- How to get something for any k?
- Simple win/win algorithm
  - If k "large", we can run our algorithm
  - If k "small", brute force $\binom{n}{k}$
- Balancing gives $2^{n^{1-\delta/3}\log n/\epsilon^3}$ time for $|E| = n^{1+\delta}$
- Can we do better?

Extensions: 3-Coloring

- 3-Coloring is in P for graphs with large minimum degree
  - Dense graphs contain a dominating set of size $O(\log n)$
  - Guess its coloring
  - Coloring remaining graph is 2-List Coloring (in P)

Extensions: 3-Coloring

- 3-Coloring is in P for graphs with large minimum degree
  - Dense graphs contain a dominating set of size $O(\log n)$
  - Guess its coloring
  - Coloring remaining graph is 2-List Coloring (in P)
- Can extend this to graphs with minimum degree Δ
  - Exists dominating set with size $\frac{n}{\Delta} \log n$
  - Guess its coloring
  - $\ldots$

Conclusions

- Density is a crucial parameter for approximating Max-k-CSP
  - Especially useful in sub-exponential setting
  - Smooth trade-off between performance and generality
  - “Tight” bounds
- Questions:
  - Other applications?
  - Other interesting sub-exponential approximations?

Thank you!