Beyond Worst-Case Analysis

a tour d'horizon

Tim Roughgarden (Stanford University)

see also lecture notes and YouTube videos for Stanford's CS264 course (on my Web page)

General Formalism

Performance measure: *cost(A,z)*

- A = algorithm, z = input

Examples:

- running time (or space, I/O operations, etc.)
- solution quality (or approximation ratio)
- correctness (1 or 0)

Issue: how to compare incomparable algorithms?

- rare exception: *instance optimality* [Fagin/Loten/Naor 03], [Afshani/Barbay/Chan 09], ...

Worst-Case Analysis

One approach: summarize performance profile $\{cost(A,z)\}_z$ with a single number cost(A)

- rare exception: bijective analysis [Angelopoulos/Dorrigiv/López-Ortiz 07], [Angelopoulos/Schweitzer 09]

Worst-case analysis: $cost(A) := \sup_z cost(A,z)$

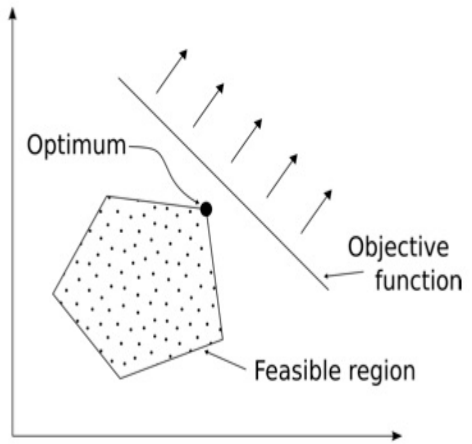
- often parameterized, e.g. by input size |z|

Pros of WCA: universal applicability (no data model)

- relatively analytically tractable
- countless killer applications

WCA Failure Modes: Simplex

Linear programming: optimize linear objective s.t. linear constraints.



Simplex method: [Dantzig 1940s] very fast in practice (# of iterations≈linear)

[Klee/Minty 72] there exist instances where simplex requires exponential number of iterations.

Irony: many worst-case polynomial-time LP algorithms unusable in practice (e.g., ellipsoid).

WCA Failure Modes: Clustering

Clustering: group data points "coherently."

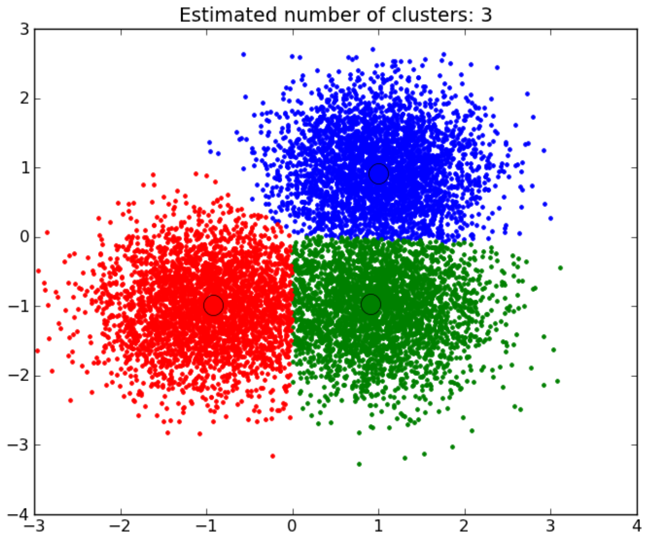
Formalization?: optimization => NP-hard

- k-means, k-median, k-sum, correlation clustering, etc.

In practice: simple algorithms (e.g., k-means++) routinely find meaningful clusters.

- "clustering is hard only when it doesn't matter"

[Daniely/Linial/Saks 12]



WCA Failure Modes: Paging

Online paging: manage cache of size k to minimize # of page faults with online requests.

Gold standard in practice: LRU.

- better than e.g. FIFO due to "locality of reference"

Worst-case analysis: [Sleator/Tarjan 85] every deterministic algorithm is equally terrible!

- page fault rate = 100%, best in hindsight (FIF) ≤ (1/k)%
- how to incorporate locality of reference in the model?

Refinements of WCA

Theorem: [Albers/Favrholdt/Giel 05] suppose $\leq f(w)$ distinct pages requested in windows of size w:

1. worst-case fault rate always $\geq \alpha_f(k)$
   - $\alpha_f(k) \approx 1/\sqrt{k}$ if $f(w) = \sqrt{w}$,); $\alpha_f(k) \approx k/2^k$ if $f(w) = \log w$
2. for LRU, worst-case fault rate always $\leq \alpha_f(k)$
3. for FIFO, exist f,k s.t. fault rate can be $> \alpha_f(k)$

Broader point: fine-grained input parameterizations can be key to meaningful WCA results.

WCA Report Card

1. *Performance prediction:* generally poor unless little variation across inputs

2. *Identify optimal algorithms:* works for some problems (sorting, graph search, etc.) but not others (linear programming, paging, etc.)

3. *Design new algorithms:* wildly successful (1000s of algorithms, many of them practical)
   – performance measure as "brainstorm organizer"

Beyond Worst-Case Analysis

Cons of worst-case analysis:

- often overly pessimistic
- can rank algorithms inaccurately (LP, paging)
- no data model (or rather: "Murphy's Law" model)

To go beyond: need to articulate a model of "relevant inputs."

  – in algorithm analysis, like in algorithm design, no "silver bullet" – most illuminating model will depend on the type of problem

Outline (Part 1)

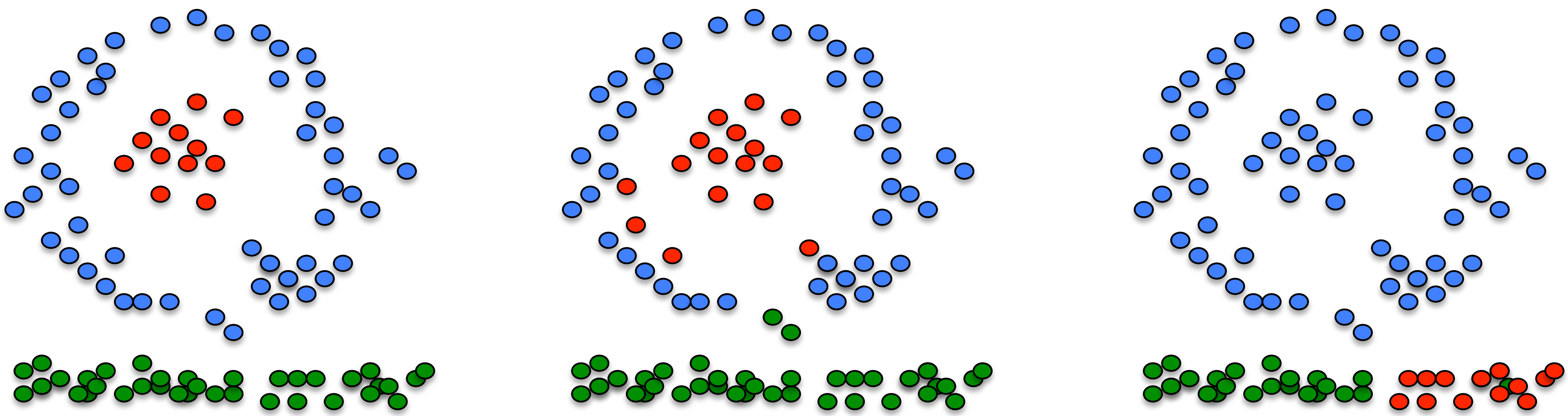
1. What is worst-case analysis?
2. Worst-case analysis failure modes
3. Clustering is hard only when it doesn't matter
4. Sparse recovery

Coming in Part 2: planted and semi-random models, smoothed analysis and other hybrid analysis frameworks

Approximation Stability

Approximation Stability: [Balcan/Blum/Gupta 09] an instance is *α-approximation stable* if all α-approximate solutions cluster almost as in OPT.

target/OPT   α-approximation   α-approximation



allowed   **not allowed!**

Stable k-Median Instances

Thesis: "clustering is hard only when it doesn't matter."

Recall: k-median/min-sum clustering.

– NP-hard to approximate better than ≈ 1.73 [Jain/Madian/Saberi 02]

Main Theorem: [Balcan/Blum/Gupta 09]

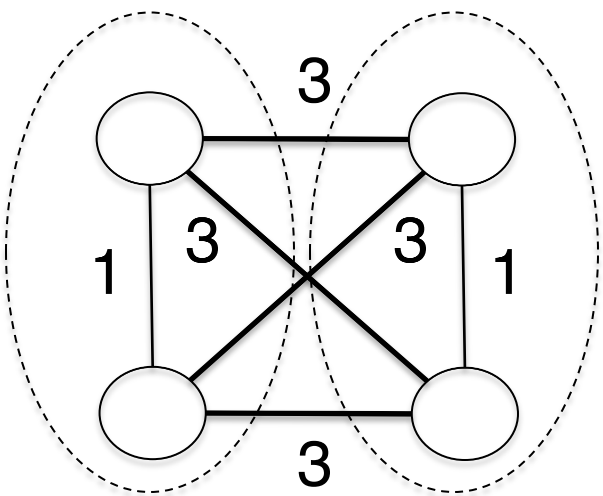
for metric k-median, α-approximation stable instances are easy, even when close to 1.

- can recover a clustering structurally close to target/OPT in poly-time

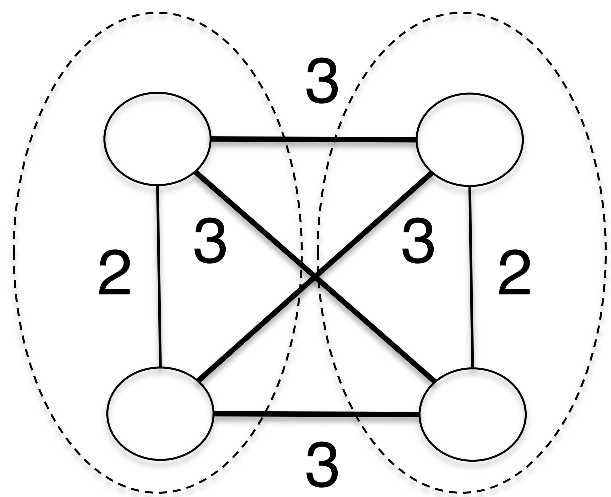
Perturbation Stability

Perturbation Stability: [Bilu/Linial 10] an instance is *γ-perturbation stable* if OPT is invariant under all perturbations of distances by factors in [1, γ]

- motivation: distances often heuristic, anyways



the max cut

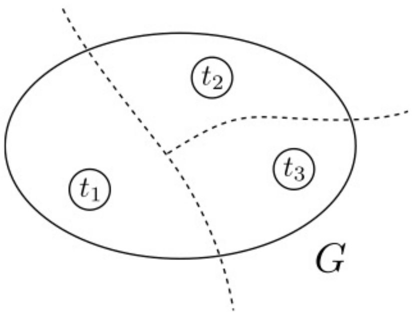


still the max cut

Minimum Multiway Cut

Case Study: [Makarychev/Makarychev/Vijayaraghavan 14] the min multiway cut problem.

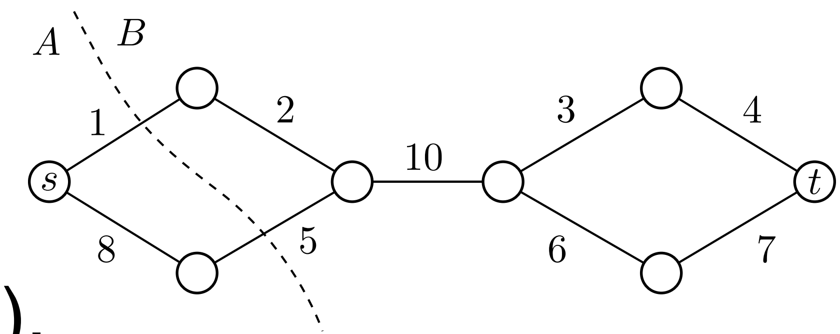
- undirected graph G=(V,E)
- costs c_e for each edge e
- terminals $t_1,\ldots,t_k$



Theorem: [Makarychev/Makarychev/Vijayaraghavan 14] a suitable LP relaxation is exact for all 4-perturbation stable multiway cut instances.

Warm-Up: Minimum s-t Cut

Folklore: LP relaxation of the min s-t cut problem is exact (opt soln = integral).



$$\min \sum_{e \in E} c_e x_e.$$

subject to:

$$
\begin{aligned}
d_s &= 0 \\
d_t &= 1 \\
x_e &\geq d_u - d_v && \text{for every edge } e = (u,v) \\
x_e &\geq d_v - d_u && \text{for every edge } e = (u,v) \\
d_v, x_e &\geq 0 && \text{for every edge } e \in E \text{ and } v \in V
\end{aligned}
$$

Proof idea: randomized rounding yields optimal cut.

- cut ball of random radius r in (0,1) around s
- expected cost ≤ LP OPT
- must produce optimal cut with probability 1

Min Multiway Cut (Relaxation)

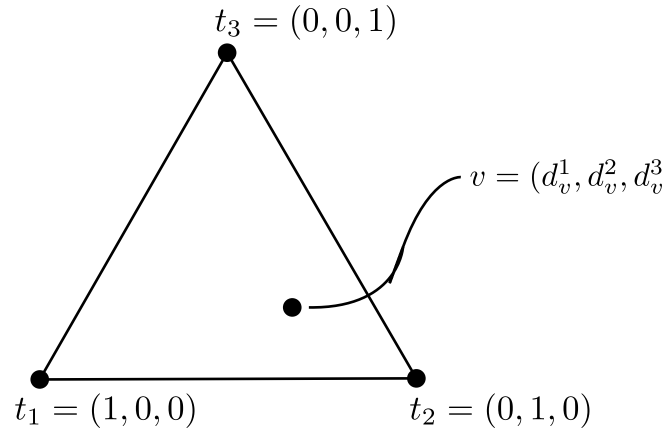
Theorem: [Makarychev/Makarychev/Vijayaraghavan 14]
LP relaxation exact for all 4-perturbation stable instances.

LP Relaxation: [Călinescu/Karloff/Rabani 00]

$$\min \sum_{e \in E} c_e x_e.$$

subject to:

$$\begin{aligned}
\sum_{i=1}^{k} d_v^i &= 1 && \text{for } v \in V \\
d_{t_i}^i &= 1 && \text{for } i = 1, 2, \ldots, k \\
y_e^i &\geq d_u^i - d_v^i && \text{for } e \in E \text{ and } i = 1, 2, \ldots, k \\
y_e^i &\geq d_v^i - d_u^i && \text{for } e \in E \text{ and } i = 1, 2, \ldots, k \\
x_e &= \frac{1}{2} \sum_{i=1}^{k} y_e^i && \text{for } e \in E \\
d_v^i, y_e^i, x_e &\geq 0 && \text{for } e \in E,\ v \in V, \text{ and } i = 1, 2, \ldots, k
\end{aligned}$$



Min Multiway Cut (Recovery)

Lemma: [Kleinberg/Tardos 00] there is a randomized rounding algorithm such that:

- Pr[edge e cut] ≤ $2x_e$
- Pr[edge e not cut] ≥ $(1-x_e)/2$

Proof idea (of Theorem): copy min s-t cut proof.

- lose 2 factors of 2 from lemma
- absorbed by 4-stability assumption
- LP relaxation must solve to integers

Open Questions

1. Improve over the factor of 4.
2. Prove NP-hardness for γ-perturbation stable instances for as large a γ as you can.
3. Connections between poly-time approximation and poly-time recovery in stable instances?
   - [Makarychev/Makarychev/Vijayaraghavan 14] tight connection between exact recovery in stable max cut instances and approximability of sparsest cut/ low-distortion $l_2^2 \to l_1$ embeddings
   - [Balcan/Haghtalab/White 16] k-center

Outline (Part 1)

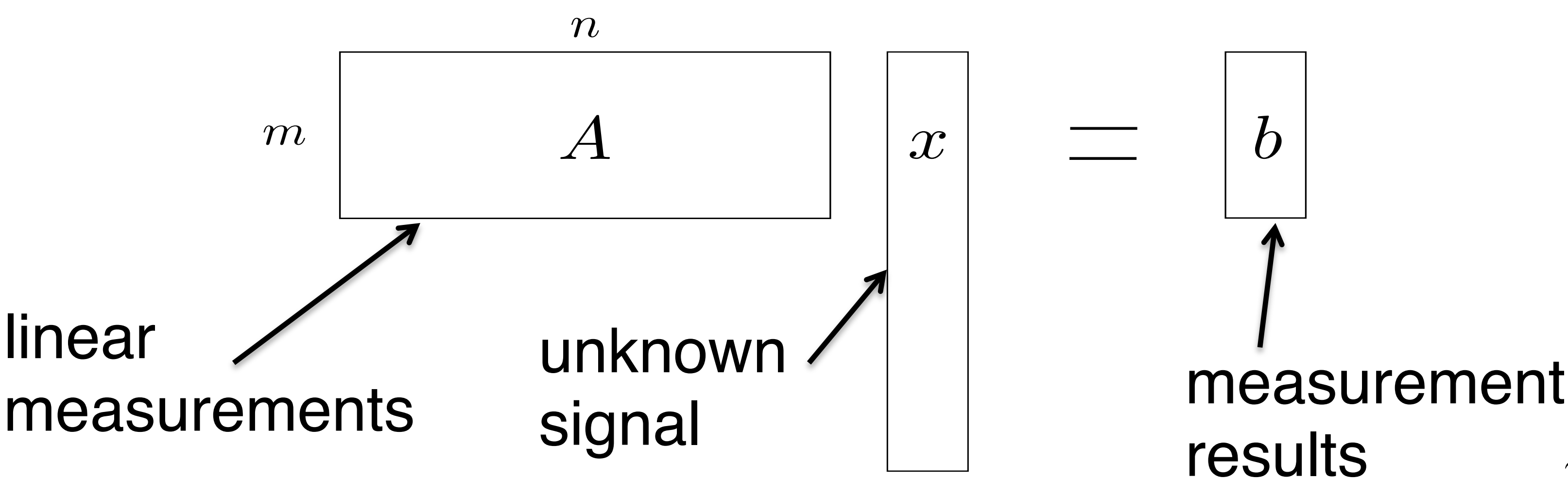
1. What is worst-case analysis?
2. Worst-case analysis failure modes
3. Clustering is hard only when it doesn't matter
4. Sparse recovery

Coming in Part 2: planted and semi-random models, smoothed analysis and other hybrid analysis frameworks

Compressive Sensing

Sparse recovery: recover unknown (but "simple") object from a few "clues." (ideally, in poly time)

Case study: compressive sensing [Donoho 06], [Candes/Romberg/Tao 06]

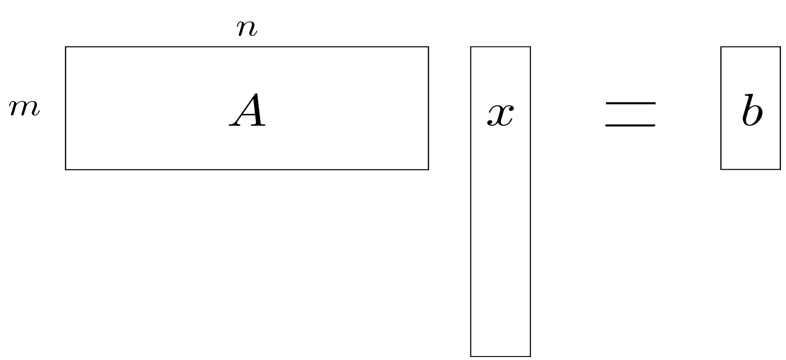


L_1-Minimization

Key assumption: unknown signal x is (approximately) *k-sparse* (only k non-zeros).

Fact: minimizing sparsity s.t. linear constraints ("l_0-minimization") is NP-hard in general. [Khachiyan 95]

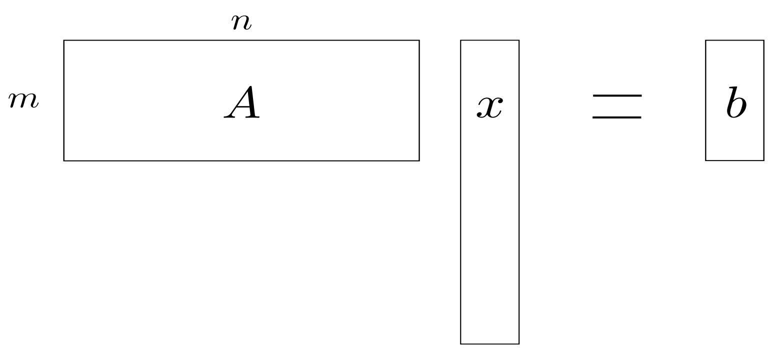
Heuristic: *l_1-minimization*: minimizing the l_1 norm over solutions to Az=b (in z) (a linear program).



Question: when does it work?

Recovery Under RIP

Theorem: if A satisfies the "restricted isometry property (RIP)" then l_1-minimization recovers x (approximately).



Example: random matrix (Gaussian entries) satisfies RIP w.h.p. if $m=\Omega(k \log (n/k))$.

- cf., Johnson-Lindenstrauss transform

Largely open: port sparse recovery techniques over to more combinatorial problems.

Part 1 Summary

- algorithm analysis is hard, worst-case analysis can fail
  - almost all algorithms are incomparable
- going beyond worst-case analysis requires a model of "relevant inputs"
- *approximation stability:* all near-optimal solutions are "structurally close" to target solution
- *perturbation stability:* optimal solution invariant under perturbations of objective function
- *exact recovery:* characterize the inputs for which a given algorithm (like LP) computes the optimal solution
  - examples: min multiway cut, compressive sensing

Intermission

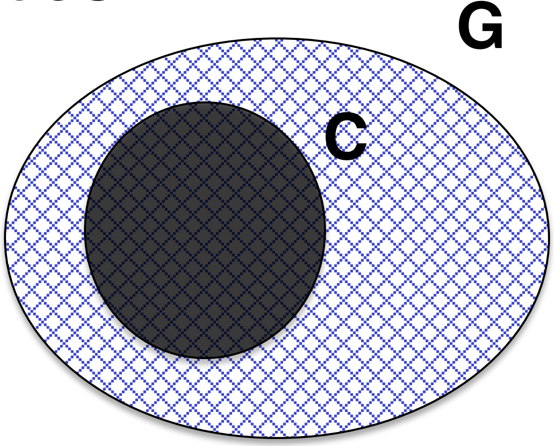
Outline (Part 2)

1. Planted and semi-random models.
   - planted clique
   - semi-random models
   - planted bisection
   - recovery from noisy parities
2. Smoothed analysis.
3. More hybrid models.
4. Distribution-free benchmarks/instance classes.

Planted Clique

Setup: [Jerrum 92]

- let H = Erdös-Renyi random graph, from G(n,½)
- let C = random subset of k vertices
- final graph G = H + clique on C



Goal: recover C in poly time.

- – easier for bigger k
- – cf., “meaningful clusterings”

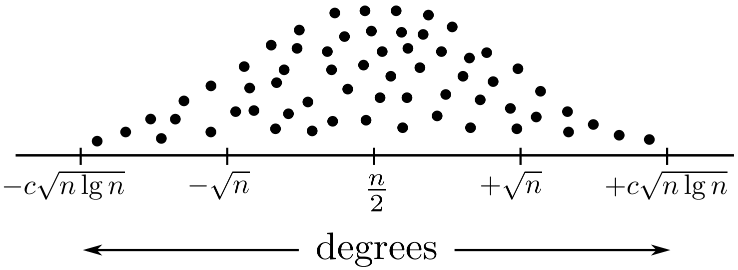
State-of-the-art: [Alon/Krivelevich/Sudakov 98]

poly-time recovery when $k = \Omega(\sqrt{n})$.

An Easy Positive Result

Observation: [Kucera 95] poly-time recovery when k = Ω(√(n log n)).

Reason: in random graph H, all degrees in [n/2-c√(n log n), n/2+c(√n log n)] w.h.p.



So: if k = Ω(√(n log n)), C = the k vertices with the largest degrees.

Problem: algorithm tailored to input distribution.

– how to encourage "robust" algorithms?

On Average-Case Analysis

Average-case analysis: $cost(A) := E_z[cost(A,z)]$

– for some distribution over inputs z

- well motivated if:
  – (i) detailed and stable understanding of distribution;
  – and (ii) don't need a general-purpose solution

Concern: advocates brittle solutions overly tailored to input distribution.

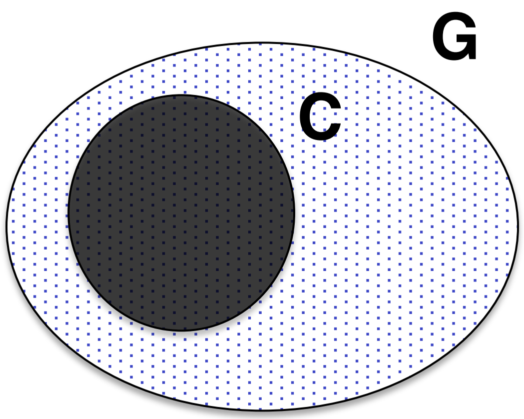
– which might be wrong, change over time, or be different in different applications

Semi-Random Models

Idea: [Blum/Spencer 95] nature and an adversary collaborate to produce a (random) input.

Semi-random planted clique: [Feige/Killian 01]

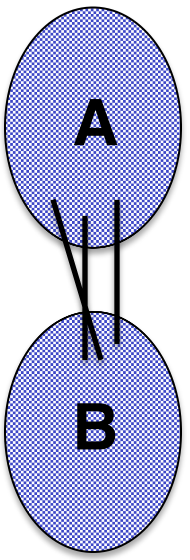
- adversary allowed to delete non-clique edges



Note: "top degrees" algorithm no longer works!

Theorem: [Feige/Krauthgamer 00] poly-time recovery when $k = \Omega(\sqrt{n})$. [using SDP/Lovasz theta function]

Planted Bisection



Setup: [Bui/Chaudhuri/Leighton/Sipser 92]

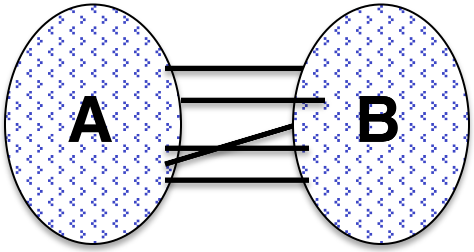
- let A, B = n/2 vertices each
- p = edge density inside A, B
- q = edge density between A, B (q < p)

Known: characterization of p and q such that exact recovery of A,B possible (w.h.p.).

  – [Feige/Killian 01], [McSherry 01], [Abbe/Bandeira/Hall 15], ...

- positive results generally extend to semi-random model
  – adversary can add edges inside A,B
    or delete edge between A, B

Planted Bisection



Sparse regime: $p = a/n$, $q = b/n$.

- only partial recovery possible (due to isolated nodes)

Theorem: [Mossel/Neeman/Sly 13,14], [Massoulié 14] partial recovery possible iff $(a-b)^2 > 2(a+b)$.

Theorem: [Moitra/Perry/Wein 16] there is a range of a,b with $(a-b)^2 > 2(a+b)$ such that partial recovery is *not* possible in the semi-random model.

- semi-random models strictly harder than random models

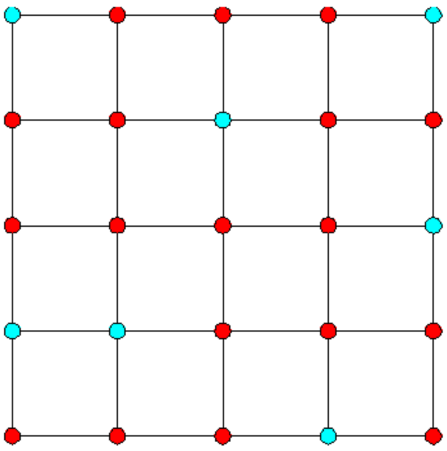
Open Questions

1. Are SDP relaxations always optimal in semi-random models?
   – see [Moitra/Perry/Wein 16] for partial results
2. Positive results for stronger adversaries.
   – see [Makarychev/Makarychev/Vijayaraghavan 12,14]
3. Computational separation between random and semi-random models?
4. Replace planted clique hardness assumption with (weaker) semi-random clique hardness?

Recovery From Noisy Parities

Setup: [Globerson/Roughgarden/Sontag/Yildirim 15]

- known graph G=(V,E)
- unknown labeling X:V -> {0,1}
- given noisy parity of each edge



Goal: (approximately) recover X.

Results: can achieve error -> 0 as noise -> 0 if G is a bounded-face planar graph or an expander.

Not possible if G is a path.

More Open Questions

1. Characterize graphs where good approximate recovery is possible (as noise -> 0).
   – some kind of “weak expansion” condition?
2. Computationally efficient recovery for expanders. (or hardness results)
3. Take advantage of noisy node labels.
4. More than two labels.

Outline (Part 2)

1. Planted and semi-random models.
2. Smoothed analysis.
   - the simplex method
   - binary optimization problems
   - local search
3. More hybrid models.
4. Distribution-free benchmarks/instance classes.

Smoothed Analysis

Idea: [Spielman/Teng 01] semi-random model:

- start with arbitrary input
- nature applies a small random perturbation

Theorem: [Spielman/Teng 01] the simplex method (with the "shadow pivot rule") has polynomial smoothed complexity.

- for every initial LP, expected (over perturbation) running time is polynomial in input size and $1/\Phi$
- improved and simplified in [Deshpande/Spielman 05], [Vershynin 06]

Binary Optimization Problems

Setup: [Beier/Vöcking 06] n 0-1 decision variables (x_i)

- objective: max $\Sigma_i v_i x_i$ (v_i's randomly perturbed)
- abstract constraints (feasible sets=subset of $2^{[n]}$)
  - examples: max spanning tree, knapsack, max-weight independent set, etc.

Theorem: [Beier/Vöcking 06] a binary optimization problem is solvable in smoothed polynomial time *if and only if* it is solvable in pseudo-polynomial time.

- weakly NP-hard -> in "smoothed P"
- strongly NP-hard -> not in "smoothed P"

Proof Idea: The Isolation Lemma

Theorem: a binary optimization problem is solvable in smoothed polynomial time if and only if it is solvable in pseudo-polynomial time.

Proof of "if" direction: ("only if" is easy)

- each v_i drawn from distribution with density ≤ $1/\Phi$
- Isolation Lemma: [Mulmuley/Vazirani/Vazirani 87] with high probability, gap between 1st- and 2nd-best feasible solutions is at least $\Phi/\text{poly}(n)$
- lazy approach: only read as many bits as needed to certify optimality (log # of bits => poly-time)

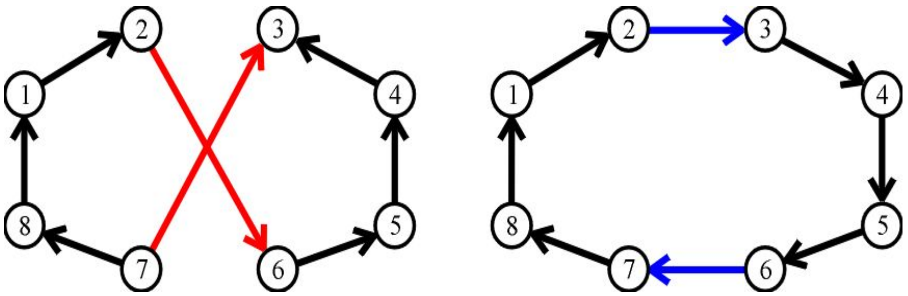
Smoothed Analysis of Local Search

Local search: often huge gap between worst-case and empirical running times.

- smoothed analysis killer app: k-means [Arthur/ Vassilvitskii 06], [Arthur/Manthey/Röglin 11]

Example: [Englert/Röglin/Vöcking 07] 2-OPT (for TSP).

Proof idea:



- only $O(n^4)$ moves
- Isolation Lemma + Union Bound => w.h.p., every local move makes $\geq \Phi/\text{poly}(n)$ progress

Local Search for Max Cut

Max cut: [Elsässer/Tscheuschner 11] same idea works for max cut (with flip neighborhood) if max degree $\Delta = O(\log n)$.

- only poly # of distinct local moves

Improvement: [Etscheid/Röglin 14] in general, smoothed complexity at most quasi-polynomial.

Open: but is it polynomial?

Open Questions

1. Does *every* local search problem for a binary optimization problem (with poly "diameter") have poly smoothed complexity?
   - max cut with flip neighborhood a special case
   - "avoiding the union bound"

2. Better smoothed analysis of simplex
   - better running time bounds (linear?), non-Gaussian perturbations, other pivot rules, sparsity-preserving perturbations

Outline (Part 2)

1. Planted and semi-random models.
2. Smoothed analysis.
3. More hybrid models.
   - examples
   - data-driven algorithm design
4. Distribution-free benchmarks/instance classes.

Hybrid Models

Thesis: for many problems there is a “sweet spot” between worst- and average-case analysis.

- where unknown distribution D lies in some known set

worst-case

$\sup_z \text{cost}(A,z)$

average-case

$E_z[\text{cost}(A,z)]$

hybrid models

$\sup_D E_{z\sim D}[\text{cost}(A,z)]$

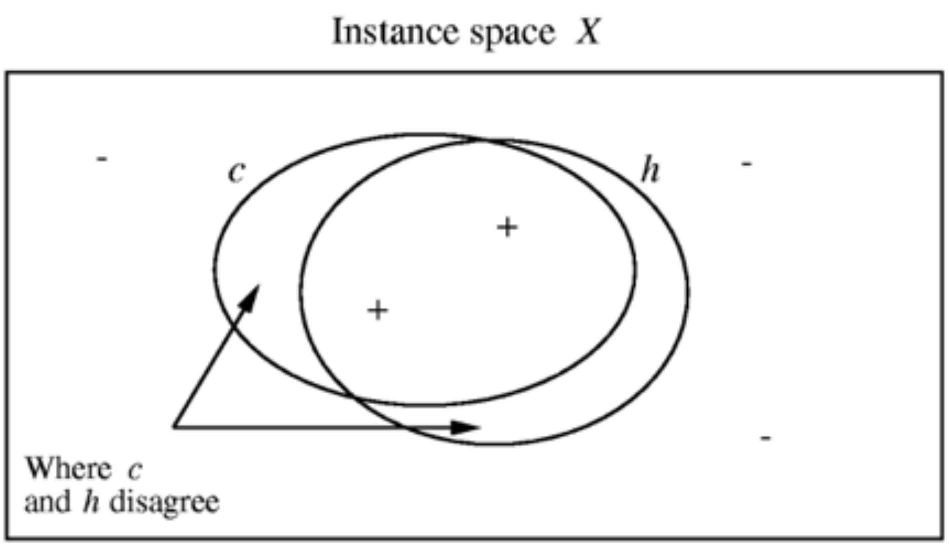
Hybrid Models: Examples

1. Semi-random models. (adversary => distribution)
2. Smoothed analysis. (initial input => distribution)
3. Random order models. (secretary problems)
4. Competitive guarantees for M/G/1 queues.
5. Prior-independent auctions. (see Anna's talk)
6. Diffuse and statistical adversaries. (paging) [Raghavan 91], [Koutsoupias/Papadimitriou 00]
   - adversary = input distribution with large min-entropy or other statistical properties

PAC Learning

Setup: [Valiant 84] receive i.i.d. labeled samples from *unknown* distribution, want to learn (approximately) the target concept (w.h.p.).

– single learning algorithm works for all distributions



Data-Driven Algorithm Design

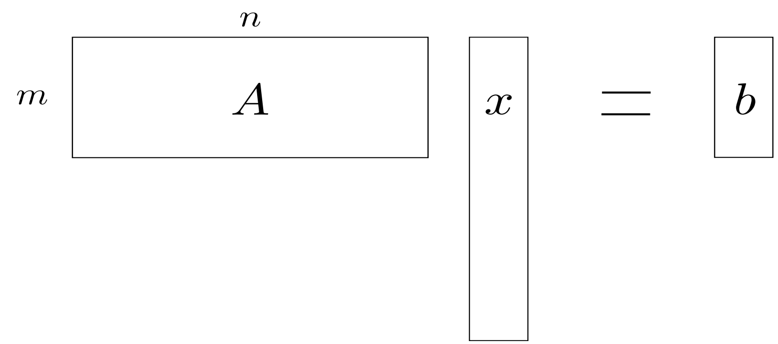
- self-improving algorithms for sorting [Ailon/Chazelle/Liu/Seshadhri 06] Delaunay triangulations [Clarkson/Seshadhri 08], convex hulls [Clarkson/Mulzer/Seshadhri 10]
  - assume elements or points are independent, want to run as fast as information-theoretic optimal
- revenue-maximizing auctions (see Anna's talk)
  - [Elkind 07], [Cole/Roughgarden 14], [Morgenstern/Roughgarden 15,16], [Devanur/Huang/Psamos 16], ...
  - learn a near-optimal auction from samples
- application-specific algorithm selection
  - see my Open Lecture (10/24) [Gupta/Roughgarden 16]
  - inspired by [Leyton-Brown et al.]

Outline (Part 2)

1. Planted and semi-random models.
2. Smoothed analysis.
3. More hybrid models.
4. Distribution-free benchmarks/instance classes.
   - compressed sensing revisited
   - no-regret algorithms re-interpreted
   - further examples

Recall: Recovery Under RIP

Theorem: if A satisfies the "restricted isometry property (RIP)" then l_1-minimization recovers k-sparse x.



Example: random matrix (Gaussian entries) satisfies RIP w.h.p. if $m=\Omega(k \log (n/k))$.

Question: other applications of such "average-case thought experiments"?

No-Regret Online Learning

Setup: action set A. Each day t=1,2,...,T:

- algorithm picks a distribution over actions
- adversary picks a reward vector $\{ r^t(a) \}_{a \text{ in } A}$

Well-Known Results:

- can't compete with best sequence in hindsight.
- *can* compete with best *fixed action* in hindsight
  – need the right benchmark to discover the right algorithms!

A Re-Interpretation (Folklore)

Average-case thought experiment: suppose every reward vector drawn i.i.d. from a distribution D.

- optimal strategy: always play action with highest expected reward (i.i.d.=>time-invariant)

Upshot: a no-regret algorithm does (almost) as well as OPT for *every* unknown distribution D

- another folklore example: static optimality of data structures (compete with OPT for all i.i.d. sequences of accesses)

More Examples

Distribution-free benchmarks:

- prior-free auction design (see [Goldberg/Hartline/Karlin/Saks/Wright 06]) as a deterministic proxy for i.i.d. bidders [Hartline/Roughgarden 08]

Distribution-free instance classes:

- social networks (see my talk in Sept. workshop)
  - graphs that are deterministic proxies for generative models [Gupta/Roughgarden/Seshadhri 14]
  - in same spirit: [Brach/Cygan/Lacki/Sankowski 16] [Borassi/Crescenzi/Trevisan 16]

Part 2 Summary

- distributions useful to define "relevant inputs"
  - but average-case analysis encourages algorithms tailored to distributional assumptions
- semi-random/hybrid models: a "sweet spot" between worst- and average-case analysis that encourages more robust solutions
  - clique, bisection, smoothed analysis, learning, etc.
- "average-case thought experiment:" define benchmarks/instance classes as deterministic proxies for an unknown distribution