Random Constraint Optimization [a]

The case of **Max-Cut**

L.M. Kirousis

Department of Computer Engineering and Informatics, University of Patras

&

RA Computer Technology Institute

Joint work with A.C. Kaporis and E.C. Stavropoulos

[a] Research partially supported by European Social Fund (ESF), Operational Program for Educational and Vacational Training II *(EPEAEK II)*; by the Research and Academic Computer Technology Institute (RACTI) and by the Future and Emerging Technologies programme of the EU under contract 001907 *“Dynamically Evolving, Large-Scale Information Systems (DELIS).”*

Circumventing NP-hardness

Circumventing NP-hardness

- Reminder: The introduction of randomness in the design of an algorithm cannot avoid the worst cases of an NP-hard problem.

Circumventing NP-hardness

- Reminder: The introduction of randomness in the design of an algorithm cannot avoid the worst cases of an NP-hard problem.
- Randomized algorithms are good, e.g. when the input is designed by a limited capacity adversary.
- In such a case, randomness in the choice of steps of an algorithm further restricts the adversary's knowledge.

Circumventing NP-hardness

- Reminder: The introduction of randomness in the design of an algorithm cannot avoid the worst cases of an NP-hard problem.
- Randomized algorithms are good, e.g. when the input is designed by a limited capacity adversary.
- In such a case, randomness in the choice of steps of an algorithm further restricts the adversary's knowledge.
- But has no effect if the adversary is omniscient.
- Formally: If $\mathrm{NP} \subset \mathrm{BPP}$ then the polynomial hierarchy collapses at the second level and also RP = NP.

First approach to NP-Hardness

Introduce randomness in the choice of the inputs.

That is, assume inputs are drawn according to a distribution.

- Then *perhaps* a realistic distribution might be found, for which worst case inputs are rare.
- However, an unwisely chosen distribution of inputs might trivialize an interesting problem.

An "easy" SAT distribution

n variables: $\{x_1, \ldots, x_n\}$. m clauses constructed independently as follows: a literal l (positive or negative) is independently placed in each clause with probability $p(n, m)$. Length of a clause is random with expectation $k = 2pn$ (average k-SAT).

An "easy" SAT distribution

n variables: $\{x_1, \ldots, x_n\}$. m clauses constructed independently as follows: a literal l (positive or negative) is independently placed in each clause with probability $p(n, m)$. Length of a clause is random with expectation $k = 2pn$ (average k-SAT).

- [Franco 1986]: depending on the value of p,
  - asymptotically almost all (a.a.a.) instances of average k-SAT are satisfied or
  - a.a.a. instances are not satisfied.
- Also: there are very easy algorithms that return a satisfying assignment or a proof of contradiction, respectively.

Moral: Watch out for trivializing distributions of the input.

History repeats

Consider a CSP with n variables, fixed arity k for the constraints and fixed domain size D for the variables.

Assume that random input instances are generated as follows:

- First randomly choose the hyperedges of the constraint hypergraph (a hyperedge comprises of the k variables that a constraint entails),
- secondly, for each hyperedge C choose the k-tuples of values that are to be non-admissible by the constraint corresponding to C.

History repeats

Consider a CSP with n variables, fixed arity k for the constraints and fixed domain size D for the variables.

Assume that random input instances are generated as follows:

- First randomly choose the hyperedges of the constraint hypergraph (a hyperedge comprises of the k variables that a constraint entails),
- secondly, for each hyperedge C choose the k-tuples of values that are to be non-admissible by the constraint corresponding to C.

[Achlioptas et al. 1997]: such random instances of CSP are a.a.a. not satisfiable (in most interesting cases).

In this presentation we focus on Boolean CSP's about graphs or Boolean formulas. Random model $G_{n,m}$.

In this presentation we focus on Boolean CSP's about graphs or Boolean formulas. Random model $G_{n,m}$.

- n: number of vertices or variables,
- m: the number of edges or fixed-length clauses selected uniformly at random from the sample space.

In this presentation we focus on Boolean CSP's about graphs or Boolean formulas. Random model $G_{n,m}$.

- n: number of vertices or variables,
- m: the number of edges or fixed-length clauses selected uniformly at random from the sample space.

Closely related: the random model $G_{n,p}$.

- p: the probability for an edge or clause to be included (independently) in the instance generated.

In $G_{n,m}$ ($G_{n,p}$, respectively) the number (expected number, respectively) of edges or clauses is assumed to be $\Theta(n)$ (sparse graphs/formulas).

Algorithmics on random graphs

[Frieze and McDiarmid 1996]:

- "...average case analysis ...banishes the pessimism of worst-case analysis."
- " ...one can criticise the models as being unrealistic but they are probably no more so than the pathological examples used in the proofs of NP-Completeness."

Algorithmics on random graphs

[Frieze and McDiarmid 1996]:

- "...average case analysis ...banishes the pessimism of worst-case analysis."
- " ...one can criticise the models as being unrealistic but they are probably no more so than the pathological examples used in the proofs of NP-Completeness."

For Example: Hamilton cycles can be found efficiently in $G_{n,m}$, if m large enough to guarantee, in probability, that the min-degree is at least 2.

Second approach to NP-Hardness

In case optimization is the goal, opt for approximation algorithms.

Approximation Factor of an Algorithm: Ratio of the size of the solution returned by the algorithm to the size of the optimal solution (for maximization <1).

Second approach to NP-Hardness

In case optimization is the goal, opt for approximation algorithms.

Approximation Factor of an Algorithm: Ratio of the size of the solution returned by the algorithm to the size of the optimal solution (for maximization <1).

Extended literature both on:

- Positive approach: Find efficient approximation algorithms with as large approximation factor as possible.
- Inapproximability approach: Under a putative hypothesis (e.g. NP $\neq$ P) prove that no efficient algorithm exists with approximation factor larger than a certain value.

Combining both approaches

Both randomness of the input and approximation: Go beyond an inapproximability result by assuming the input is drawn according to a distribution.

Basic strategy:

- Find an upper bound $\mathbf{ub}$ such that the size of solutions is a.a.a. at most $\mathbf{ub}$.
- Design an algorithm that a.a.a. returns a solution with size at least $\mathbf{lb}$ (so $\mathbf{lb}$ can be considered as a *typical* lower bound) .
- Try to show that the ratio $\mathbf{lb}/\mathbf{ub}$ exceeds an approximation factor known to be unattainable (under a putative hypothesis) for all inputs.

Previous results

[Håstad 2001] MAX-3-SAT cannot be approximated by a factor $> 7/8$.

- [Fernandez de la Vega & Karpinski 2002] Approximation factor $8/9$ for a.a.a. instances.
- Improvement [Interian 2004] to $19/20$.

Previous results

[Håstad 2001] MAX-3-SAT cannot be approximated by a factor $> 7/8$.

- [Fernandez de la Vega & Karpinski 2002] Approximation factor $8/9$ for a.a.a. instances.
- Improvement [Interian 2004] to $19/20$.

Above results do not make use of the degree structure of literals in the random input (degree of a literal = number of its occurrences in the formula).

So cannot be directly extended to problems where “degree” is more important than in MAX-SAT, like MAX-CUT.

MAX-CUT: Color the vertices of a graph either red or blue, so that the number of bichromatic edges is maximized.

Upper bounds

To show that $\mathbf{ub}$ is an upper bound (First Moment Method):

- Start with Markov inequality:

$$\Pr[\exists s : |s| > \mathbf{ub}] \leq \mathrm{Ex}(|\{s : |s| > \mathbf{ub}\}|).$$

- Show that the rhs expectation above is asymptotically zero.

Lottery phenomenon: The probability of winning might be almost zero but the expectation of earnings “measurably positive.”

- FMM has been used to get upper bounds for random versions of MAX-3-SAT, MAX-k-SAT and MAX-CUT.
- However, results obtained by the “naive” FMM are mostly far from optimal.
- Exception: MAX-k-SAT, where asymptotically with k the “naive” FMM gives good upper bounds [Achlioptas, Naor & Peres 2003].

Fine FMM

- To make the rhs of Markov inequality

$$\Pr[\exists s : |s| > \mathbf{ub}] \leq \mathrm{Ex}(|\{s : |s| > \mathbf{ub}\}|)$$

  closer to its lhs, exclude large solutions that may only occur in small-probability instances.
- NB: Care must be taken so that an instance with a non-empty set of solutions, still retains at least one solution after the exclusion of its large solutions.

Application: MAX-CUT [Kaporis et al. 2006]

Random MAX-CUT

Results for deterministic MAX-CUT:

- [Goemans & Williamson] 1994 Approximation factor $\alpha_{GW} \approx 0.87856$
- [Håstad 2001] No approximation with factor $> 16/17 \approx 0.94118$ (unless P$=$NP).
- [Khot et al. 2004] Even α_{GW} is optimal assuming the Unique Games and Majority is Stablest conjectures. NB: some slightly stronger conjectures recently falsified [Charikar et al. 2006]

Goal: Go beyond Håstad's threshold for Random MAX-CUT.

Majority cuts

Apply FMM to majority cuts:

- At least half of the edges incident on any vertex are bichromatic.
- Any vertex of even degree whose exactly half edges are bichromatic is red.

NB: After recoloring a vertex violating any of the above conditions either

- the cut size increases,
- or the cut size remains the same but the number of red vertices increases.

Degree considerations

To compute the expected number of majority cuts exceeding a threshold $\mathbf{ub}$ (as a function of the density $d =$ the edges-to-variables ratio m/n), assume that:

G is random conditional that the number of vertices of degree k is fixed and equal to the expectation *as if*

- the degree of each vertex were Poisson distributed with mean $2d$.

In other words, we assume that the *degree sequence* of the random graph is typical.

More about the degrees

Observation: To compute the max cut of a graph, ignore the edges that are incident to a vertex of degree 1 (those *must* be included in a maximum cut). Therefore:

- Start with a random graph G with typical degree sequence.
- Then recursively delete edges incident on vertices of degree 1, to get a new graph G' known as the 2-core of G.

  Reminder: the k-core of a graph is the maximal subgraph of G obtained

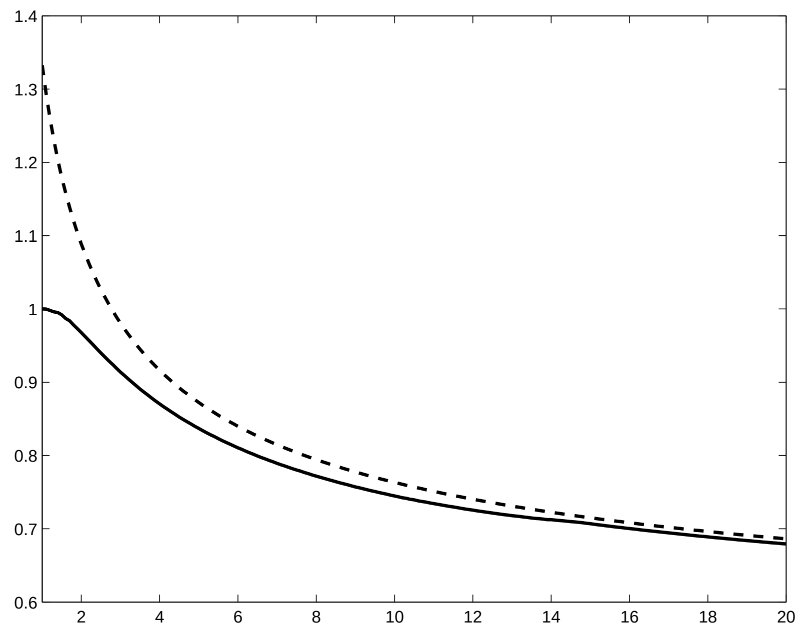
  - by edge deletion only,
  - and such that all vertices are either isolated or have degree $\geq k$.

More about the degrees—cont/ed

- Compute (analytically) the degree sequence of G' by solving the system of differential equations that describes the mean path of the edge deletion process (Wormald's technique).
- Then compute the expected number of *majority* cuts on G' (a non-trivial analytic computation).
  NB: G' is random conditional its degree sequence.
- Apply FMM.

Thus: for each given value $d > 0$, numerically obtain an upper bound of the max cut $\mathbf{ub}(d)$ that holds for a.a.a. graphs with density d.

Comparisons



The bound $\mathbf{ub}(d)$ computed as previously juxtaposed with the bound obtained by direct application of FMM by e.g. [Bertoni et al. 1997] (dashed line).

The lower bound

General strategy:

- Design an algorithm that finds a cut.
- Then for each given density d compute a $\mathbf{lb}(d)$ such that the algorithm a.a.a. returns a cut of size at least $\mathbf{lb}(d)$.
- Method of differential equations (Wormald) for the probabilistic analysis of the algorithm.
- Previously analyzed algorithms did not take advantage of the typical degree sequence, nor of the fact that we can start with the 2-core.

Bird's eye view of the algorithm

[Kaporis et al. 2006]

- Start with a random graph G, with a typical degree sequence corresponding to density d.
- Find the 2-core G' of G. The edges deleted to get G' are part of the cut.
- Successively color one selected vertex of G' either red or blue.
- Discrepancy of a vertex v after a coloring step t:

$$D(v) = |d_b(v) - d_r(v)|, \text{ where}$$

$$d_b(v) = \text{ number of currently blue neighbors of } v$$

$$d_r(v) = \text{ number of currently red neighbors of} v$$

Algorithm—cont/ed

- Always select a vertex with max discrepancy.
- Color it greedily (to guarantee that the number of bichromatic edges to be generated at the current step is maximized).
- Break ties (same max discrepancy) by choosing a vertex where the number of yet uncolored neighbors is minimized (to minimize the impact of the selection on future selections).

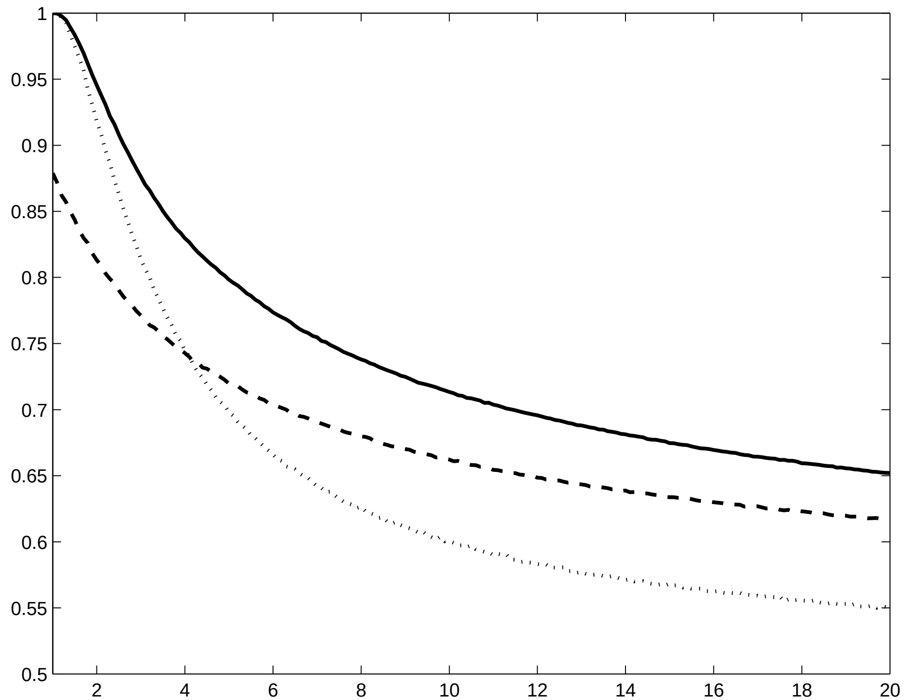
Analysis of the algorithm

Method of differential equations:

- Compute the degree sequence of the 2-core G' of G.
- Model with differential equations the evolution—at each step of the algorithm—of the following parameters of G':
  - The number of vertices v with a given pair of values for $d_b(v)$ and $d_r(v)$,
  - the number c of the currently bichromatic edges (current cut size).

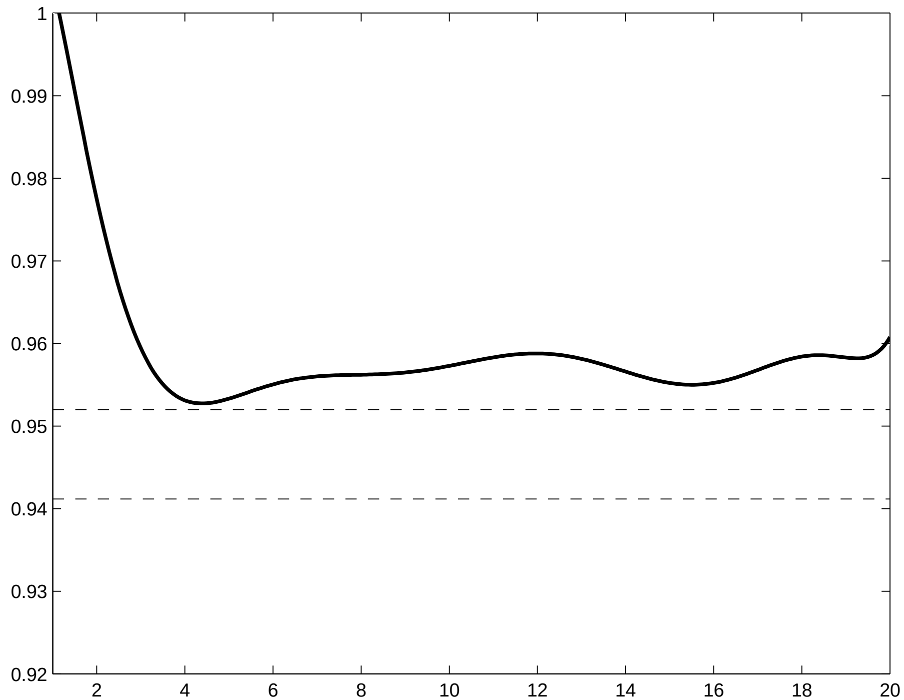
Previously analyzed algorithms that compute typical-case lower bounds did not make use of the degree sequence.

Comparisons



The bound $lb(d)$ computed as previously juxtaposed with the values of algorithms by [Coja-Oghlan et al. 2003] (dashed) and [Coppersmith et al. 2003] (dotted).

Putting everything together



The approximation ratio $\mathrm{lb}(d)/\mathrm{ub}(d)$. The lower dashed line corresponds to Håstad inapproximability threshold $16/17$, while the upper dashed line to the approximation ratio $0.952 > 16/17$ [Kaporis et al. 2006].