Rigorous analysis of heuristics for NP-hard problems

Uriel Feige

Weizmann Institute

Microsoft Research

Computational problems

We would love to have algorithms that:

- Produce optimal results.
- Are efficient (polynomial time).
- Work on every input instance.

NP-hardness

For many combinatorial problems, the goal of achieving all three properties simultaneously is too ambitious (NP-hard).

We should set goals that are more modest.

Relaxing the desired properties

Optimality: approximation algorithms.

Efficiency: sub-exponential algorithms, fixed parameter tractability.

Firm theoretical foundations. Both positive and negative results.

Heuristics

Relax the universality property: need not work on every input.

In this talk: heuristics are required to produce optimal results in polynomial time, on typical inputs.

Conceptual problem: the notion typical is not well defined.

Some questions

Explain apparent success of known heuristics.

Come up with good heuristic ideas.

Match heuristics to problems.

Investigate fundamental limitations.

Prove that a certain heuristic is good.

Prove that a certain heuristic is bad.

In this talk

Some theoretical frameworks for studying heuristics.

Some algorithmic ideas that are often used.

Heuristics is a huge subject. This talk presents only a narrow view, and excludes many important and relevant work.

The importance of modeling

For a rigorous treatment of heuristics, need a rigorous definition for typical inputs.

Given a rigorous definition for typical inputs (for example, planar graphs), one is no longer dealing with a fuzzy notion of heuristics, but rather with the familiar notion of worst case analysis.

Probabilistic models

A typical input can be modeled as a random input chosen from some well defined distribution on inputs.

Again, design of heuristics often boils down to worst case analysis:

- Most random inputs have property P.
- Algorithm works on all inputs with property P.

Rigorous analysis

In this talk, limit ourselves to discussion of heuristics in well defined models. In these models, prove theorems.

To early to assess the relevance and success of the methodology.

Some theoretical frameworks

Random inputs.

Planted solution models.

Semi-random models, monotone adversary.

Smoothed analysis.

Stable inputs.

Random inputs

Typical example: random graphs, n vertices, m edges.

An algorithm for finding Hamiltonian cycles in random graphs, even when the minimum degree is 2 [Bollobas,Fenner,Frieze].

No algorithm known for max clique in random graphs.

Planted solution models

Useful when random model seems too difficult.

Example: plant in a uniform random graph a clique of large size k. Can a polynomial time algorithm find the k-clique?

- Yes, when $k = \Omega(\sqrt{n})$ [Alon,Krivelevich,Sudakov].
- Unknown when $k = o(\sqrt{n})$.

Semi random model [Blum-Spencer]

Useful in order to overcome over-fitting of algorithms to the random model. Adds robustness to algorithms.

Example, when $k >> \sqrt{n \log n}$, vertices of planted k-clique have highest degree.

Algorithm may select the k highest degree vertices and check if they form a clique.

Monotone adversary [Feige-Kilian]

Adversary may change the random input, but only in one direction.

Planted clique: adversary may remove arbitrarily many non-clique edges.

Degree based algorithm no longer works.

Semidefinite programming does work, when $k = \Omega(\sqrt{n})$ [Feige-Krauthgamer].

Smoothed analysis [Spielman-Teng]

Arbitrary input, random perturbation.

Typical input – low order bits are random.

Explain success of simplex algorithm [ST].

FPTAS implies easy smoothed instances [Beier-Voecking].

Smoothed versus semirandom

Smoothed analysis:

- arbitrary instance – defines an arbitrary region.
- random input is chosen in this region.
- stronger when region is small.

Monotone adversary:

- random instance – defines a random region.
- arbitrary input is chosen in region.
- stronger when region is large.

Stable inputs [Bilu-Linial]

In some applications (clustering), the interesting inputs are those that are stable in the sense that a small perturbation in the input does not change the combinatorial solution.

An algorithm for (highly) stable instances of cut problems [BL].

Stable versus smooth

Consider regions induced by combinatorial solution.

In both cases, must solve all instances that are far from the boundary of their region.

For instances near the boundary:

- Smoothed analysis: solve a perturbed input.
- Stable inputs: do nothing.

Running example: 3SAT

$$(x_1 \vee x_2 \vee \bar{x}_3) \wedge (x_2 \vee \bar{x}_4 \vee x_5) \wedge \cdots$$

n variables, m clauses, 3 literals per clause.

Clauses chosen independently at random.

Random formula f with m >> n.

Probabilistic estimates

The expected number of satisfying assignments for f is:

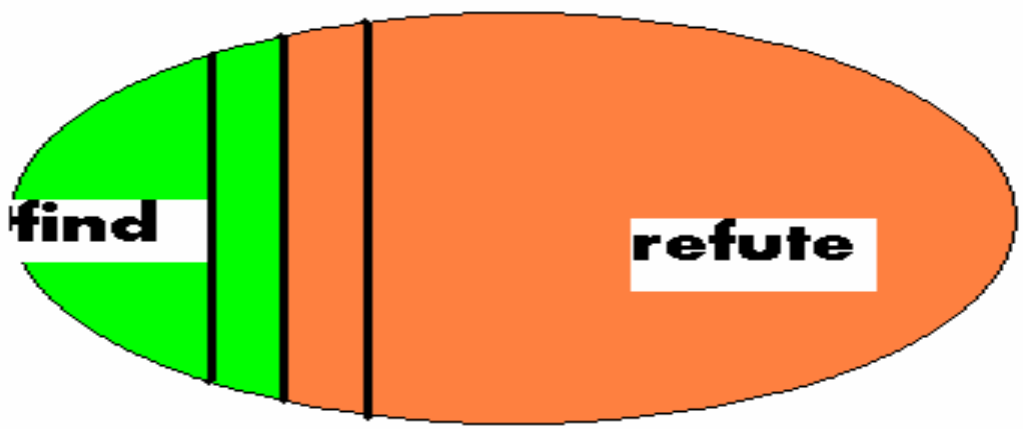
$$(1 - 1 / 2^3)^m * 2^n$$

When m >> n, the formula f is unlikely to be satisfiable.

Two tasks

Search: if the formula is satisfiable, then find a satisfying assignment.

Refutation: if formula is not satisfiable, then find a certificate for nonsatisfiability.



Simple case

When $m >> n \log n$, then if formula is satisfiable, the satisfying assignment is likely to be unique.

Then distribution on random satisfiable formulas can be approximated by planted solution distribution.

Planted solution model

First pick at random an assignment a to the variables.

Then choose at random clauses, discarding clauses not satisfied by a, until m clauses are reached.

When $m >> n \log n$, a is likely to be a unique satisfying assignment.

Statistical properties

For every variable x, in every clause C that contained x and was discarded, the polarity of x in C disagreed with its polarity in a.

Set x according to the polarity that agrees with the majority of its occurrences in f.

When $m >> n \log n$, it is likely that this algorithm exactly recovers a.

Sparser formulas

$m = d*n$ for some large constant d.

Distribution generated by planted model no longer known to be statistically close to that of random satisfiable formulas. Favors formulas with many satisfying assignments.

We present algorithm only for planted model.

Majority vote

Majority vote assignment $a(0)$.

For most variables, $a(0) = a$, and $a(0)$ satisfies most clauses.

Still, linear fraction of variables disagree with a, and a linear fraction of clauses are not satisfied.

This fraction is exponentially small in d.

Hill climbing

Moving towards satisfying assignment.

Alon-Kahale (for 3-coloring).

Flaxman (for planted 3SAT).

Feige-Vilenchik (for semirandom 3SAT).

Semirandom model: monotone adversary can add arbitrary clauses in which all three literals are set in agreement with a.

Conservative local search

a(j) is the assignment at iteration j, T(j) is the set of clauses already satisfied.

a(0) is the majority vote.

Pick an arbitrary clause C not in T(j).

Find the assignment closest (in Hamming distance) to a(j) that satisfies T(j) + C.

Increment j and repeat.

Time complexity

The algorithm obviously finds a satisfying assignment. The only question is how fast.

The number of iterations is at most m (the number of satisfied clauses increases in every iteration).

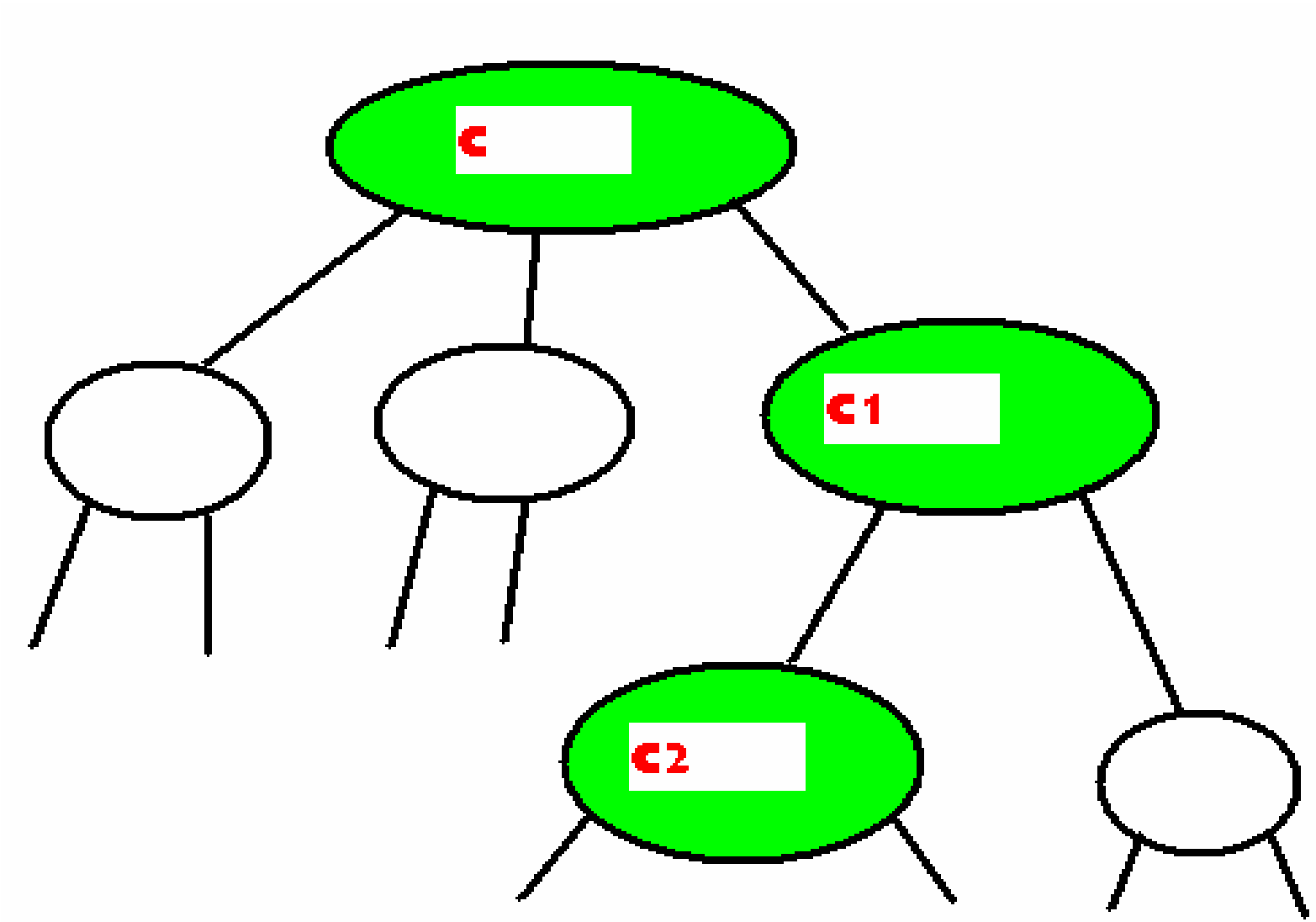
Complexity per iteration

Let h be Hamming distance between a(j) and a(j+1).

At least one of three variables in C needs to be flipped.

In a clause that becomes not satisfied in T(j), at least one of two variables needs to be flipped.

Time proportional to $3 * 2^{h-1}$


c
c1
c2

Main technical lemma

Lemma: With high probability over the choice of f, in all iterations $h < O(\log n)$.

Hence algorithm runs in polynomial time.

(True also for the semirandom model.)

Sketch of proof – the core

A variable x for which a(0) = a is a core variable if flipping x ruins T(0), and T(0) can then be satisfied only by flipping a linear number of other variables.

The set of clauses not satisfied by the core decomposes into sub-formulas of size O(log n) not sharing non-core variables.

Main invariant

An iteration can be completed in O(log n) flips, of non-core variables.

As long as h = O(log n), no core variable will accidentally be flipped, and the invariant is maintained.

The algorithm need not know the core.

Worst case analysis

Algorithm works on every input formula f with property P (defined in terms of core).

Probabilistic analysis (much too complicated to be shown here) shows that in the planted model, input formula f is likely to have property P.

Open problems

Does the algorithm run in polynomial time on random satisfiable formulas?

When $m >> n$? For arbitrary m?

Does the cavity method (survey propagation [Braunstein, Mezard, Zecchina]) provably work on random formulas?

Alternative algorithms?

More challenging models?

Refutation algorithms

If the formula is not satisfiable, the algorithm presented takes exponential time to detect this.

Heuristics for finding solutions are not the same as heuristics for refutation (unlike worst case algorithms).

Common refutation algorithms (resolution) take exponential time on random formulas.

Refutation by approximation

When $m >> n$, every assignment satisfies roughly $7m/8$ clauses of a random formula.

An algorithm for approximating max 3sat within a ratio strictly better than 7/8 would refute most dense 3SAT formulas.

Unfortunately, approximating max 3sat (in the worst case) beyond 7/8 is NP-hard [Hastad].

Turning the argument around

What if refuting random 3sat is hard?

Would imply hardness of approximation:

- Max 3sat beyond 7/8 (PCP + Fourier).
- Min bisection, dense k-subgraph, bipartite clique, 2-catalog segmentation, treewidth, etc.

A good rule of thumb. Most of its predictions (with weaker constants) can be proved assuming NP not in subexponential time [Khot].

A simple refutation algorithm

Assume $m > n^2$.

There are 3n clauses that contain x1.

Suffices to refute this subformula f1.

Substitute x1 = 0. Simplify to a 2CNF formula.

Random 2CNF formula with 3n/2 clauses.

Unlikely to be satisfiable.

2SAT can be refuted in polynomial time.

Repeat with x1 = 1.

Best current bounds

Can refute random formulas with $m > cn^{3/2}$ [Feige-Ofek].

Based on pair-wise statistical irregularities, and eigenvalue computations.

Can be run in practice on formulas with n=50000, $m = 2.5n^{3/2}$, if one trusts standard software packages for the eigenvalue computations.

The basic idea [Goerdt-Krivelevich]

Will be shown for random 4SAT formula f with $m > cn^2$

In a satisfying assignment a, at least half the variables are negative (w.l.o.g.).

Let S be the set of variables negative in a.

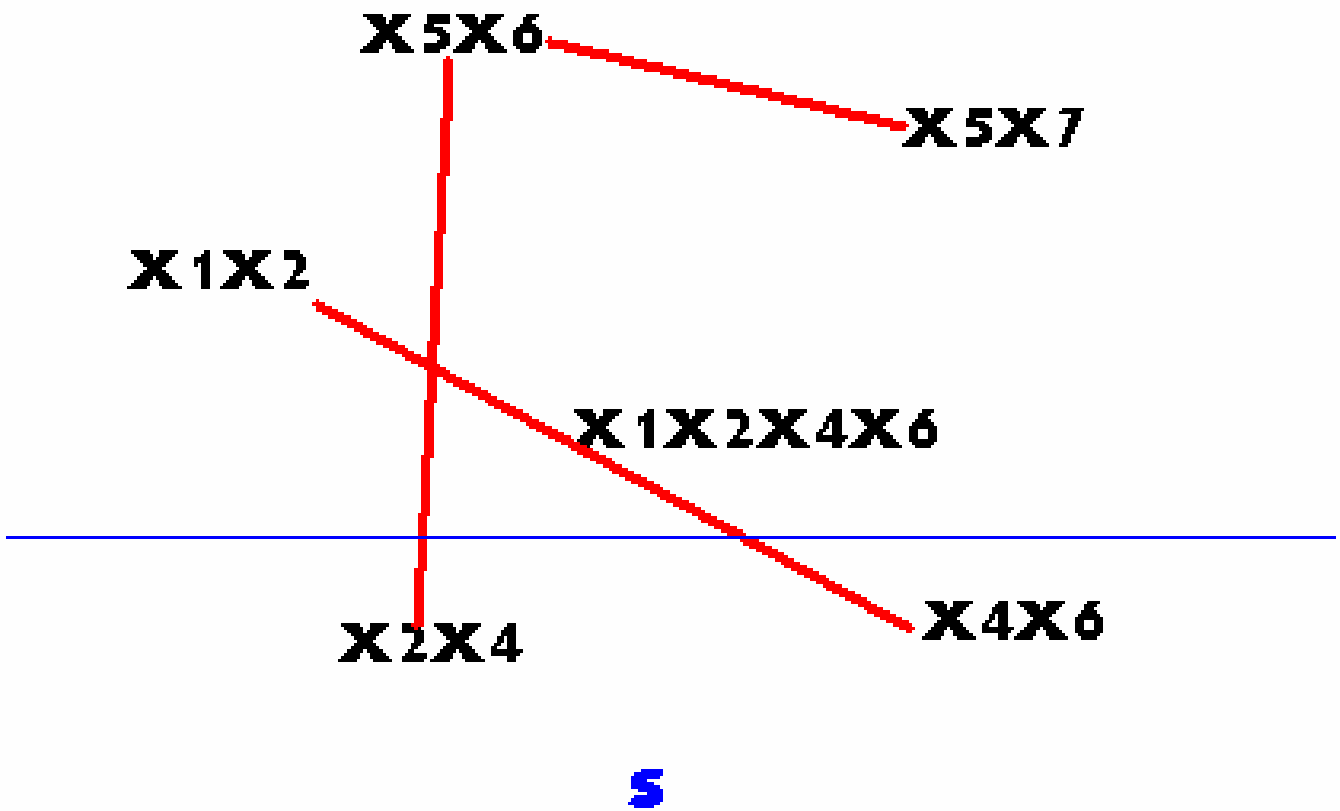
Then there is no positive clause in f whose four variables are in S.

Reduction to graph problem

Every pair of variables [xi xj] – a vertex.

Every positive clause (xi xj xk xl) – an edge ([xi xj], [xk xl]).

S forms an independent set of size N/4.


X5X6
X5X7
X1X2
X1X2X4X6
X4X6
X2X4
S

Random non-satisfiable f

Random graph with N vertices and much more than N edges.

Unlikely to have an independent set of size N/4.

Moreover, this can be certified efficiently, by eigenvalue techniques (or by SDP, computing the theta function of Lovasz).

Refutes random 4SAT with $m > cn^2$

Extension to 3SAT

Trivially extends when $m > cn^2$

With additional ideas, get down to $m > cn^{3/2}$

A certain natural SDP cannot get below $m < cn^{3/2}$ [Feige-Ofek].

Neither can resolution [Ben-Sasson and Widgerson].

Goal: refute random 3SAT with m = O(n).

Summary

Several rigorous models in which to study heuristics.

Rigorous results in these models, including hardness results (not discussed in this talk).

The heuristics may be quite sophisticated.

Wide open research area.