CS 172: Computability and Complexity

SAT: Typical-Case Complexity

Sanjit A. Seshia
EECS, UC Berkeley
k-SAT

• A SAT problem with input in CNF with at most k literals in each clause

• Complexity for non-trivial values of k:
  – 2-SAT: in P
  – 3-SAT: NP-complete
  – > 3-SAT: ?
3-SAT: Complexity Bounds
(circa 2006/07)

• Obvious upper bound on run-time?
  – In terms of $n$, the number of Boolean variables
• Best known deterministic upper bound
  $1.473^n$
• Best known randomized upper bound
  $1.324^n$
• Best known lower bound
  $n^{1.801}$
  – If limited to sub poly space
Worst-Case Complexity
Beyond Worst-Case Complexity

• What we really care about is “typical-case” complexity
• But how can one measure “typical-case”?
• Two approaches:
  – Is your problem a restricted form of 3-SAT? That might be polynomial-time solvable
  – Experiment with (random) SAT instances and see how the solver run-time varies with formula parameters (#vars, #clauses, … )
Special Cases of 3-SAT

• You already know two poly-time solvable cases:
  – 2-SAT
  – Horn-SAT
Phase Transitions in k-SAT

• Consider a fixed-length clause model
  – k-SAT means that each clause contains exactly k literals

• Let SAT problem comprise $m$ clauses and $n$ variables
  – Randomly generate the problem for fixed $k$ and varying $m$ and $n$

• Question: How does the problem hardness vary with $m/n$?
  – Why is $m/n$ interesting?
3-SAT Hardness

As $n$ increases, the hardness transition grows sharper.
Transition from SAT to UNSAT at $m/n \approx 4.3$
Threshold Conjecture

• For every $k$, there exists a $c^*$ such that
  – For $m/n < c^*$, as $n \to \infty$, problem is satisfiable with probability 1
  – For $m/n > c^*$, as $n \to \infty$, problem is unsatisfiable with probability 1

• Conjecture proved true for $k=2$ and $c^*=1$

• For $k=3$, current status is that $c^*$ is in the range $3.42 - 4.51$
The (2+p)-SAT Model

• We know:
  – 2-SAT is in P
  – 3-SAT is NP-complete

• Problems are (typically) a mix of binary and ternary clauses
  – Let $p \in \{0,1\}$
  – Let problem comprise $(1-p)$ fraction of binary clauses and $p$ of ternary
  – So-called (2+p)-SAT problem
Experimentation with random (2+p)-SAT

- When $p < \sim 0.41$
  - Problem behaves like 2-SAT
  - Linear scaling
- When $p > \sim 0.42$
  - Problem behaves like 3-SAT
  - Exponential scaling
Backbones and Backdoors

• **Backbone** [Parkes; Monasson et al.]
  – Subset of literals that must be true in every satisfying assignment (if one exists)
  – Empirically related to hardness of problems

• **Backdoor** [Williams, Gomes, Selman]
  – Subset of variables such that once you’ve given those a suitable assignment (if one exists), the rest of the problem is poly-time solvable
  – Also empirically related to hardness

• But no easy way to find such backbones / backdoors! 😞
A Classification of SAT Algorithms

- **Davis-Putnam (DP)**
  - Based on resolution
- **Davis-Logemann-Loveland (DLL/DPLL)**
  - Search-based
  - Basis for current most successful solvers
- **Stalmarck's algorithm**
  - "Different" kind of search, proprietary algorithm
- **Stochastic search**
  - Local search, hill climbing, etc.
  - Unable to prove unsatisfiability (incomplete)
Resolution

• Two CNF clauses that contain a variable x in opposite phases (polarities) imply a new CNF clause that contains all literals except x and x’

• \((a + b) (a’ + c) = (a + b)(a’ + c)(b + c)\)

• Why is this true?
The Davis-Putnam Algorithm

• Iteratively select a variable $x$ to perform resolution on
• Retain only the newly added clauses and the ones not containing $x$
• Termination: You either
  – Derive the empty clause (conclude UNSAT)
  – Or all variables have been selected
Resolution Example

How many clauses can you end up with? (at any iteration)
Further Reading

• Article on Phase Transitions in SAT (linked on website)