Towards Industrial-like Random SAT Instances

Carlos Ansótegui
DIEI, UdL, Lleida, Spain

María Luisa Bonet
LSI, UPC, Barcelona, Spain

Jordi Levy
IIIA, CSIC, Barcelona, Spain

IJCAI'09, Pasadena

SAT

- SAT is a central problem in computer science and AI
- The problem is NP-complete
- State-of-the-art solvers (heuristics, backjumping, learning, restarts, . . .) are of practical use with real-world SAT instances
- **Objective:** Design solvers that perform well on real-world SAT instances

SAT

- SAT is a central problem in computer science and AI
- The problem is NP-complete
- State-of-the-art solvers (heuristics, backjumping, learning, restarts, ...) are of practical use with real-world SAT instances
- **Objective:** Design solvers that perform well on real-world SAT instances

What is a "real-world" SAT instance?

SAT Competitions

- SAT competitions evaluate ideas, techniques and solvers
- Competitions use benchmarks:
  - Randomly generated
    - ⇒ Unlimited in number
    - ⇒ Families of instances: one for every number of vars
    - ⇒ Generated on demand: fair in competitions
    - ⇒ Parameterized degree of difficulty
  - Industrial
    - ⇒ Limited in number
    - ⇒ Specially valuable
  - Crafted

SAT Competitions

- SAT competitions evaluate ideas, techniques and solvers
- Competitions use benchmarks:
  - Randomly generated
    - ⇒ Unlimited in number
    - ⇒ Families of instances: one for every number of vars
    - ⇒ Generated on demand: fair in competitions
    - ⇒ Parameterized degree of difficulty
  - Industrial
    - ⇒ Limited in number
    - ⇒ Specially valuable
  - Crafted

SAT Competitions

- SAT competitions evaluate ideas, techniques and solvers
- Competitions use benchmarks:
  - Randomly generated
    - ⇒ Unlimited in number
    - ⇒ Families of instances: one for every number of vars
    - ⇒ Generated on demand: fair in competitions
    - ⇒ Parameterized degree of difficulty
  - Industrial
    - ⇒ Limited in number
    - ⇒ Specially valuable
  - Crafted

General Objective

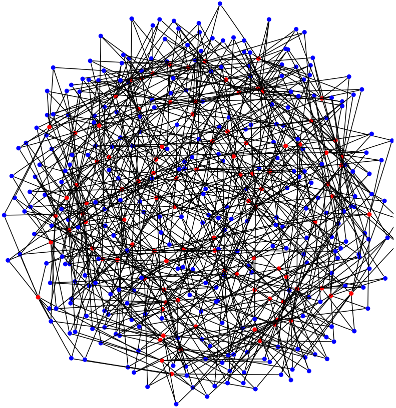
Create generators of random instances with properties similar to industrial ones to test solvers

Stated as 10th challenge by Kautz&Sellman in "Ten Challenges in Propositional Reasoning and Search":

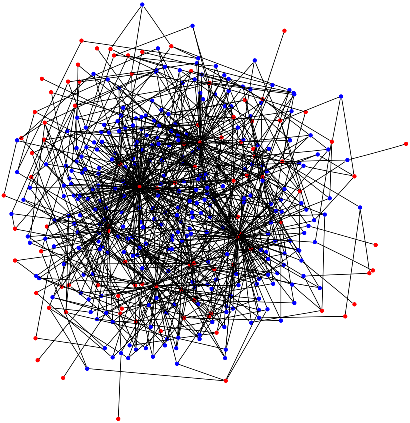
> *Develop a generator for problem instances that have computational properties that are more similar to real-world instances[...] While hundreds of specific [industrial] problems are available, it would be useful to be able to randomly generate similar problems by the thousands for testing purposes*

Also Rina Dechter in her book proposes the same objective

Random 3-CNF



Real-World Instance



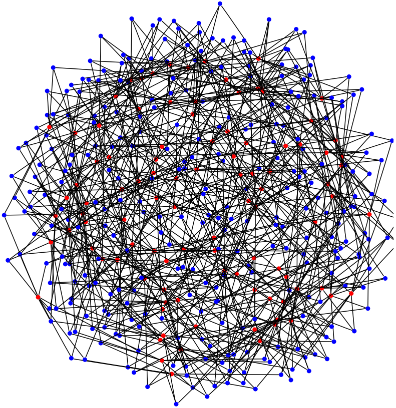
Bi-partite graph: Nodes: are variables v and clauses c

Edges: v—c if clause **c** contains variable **v**

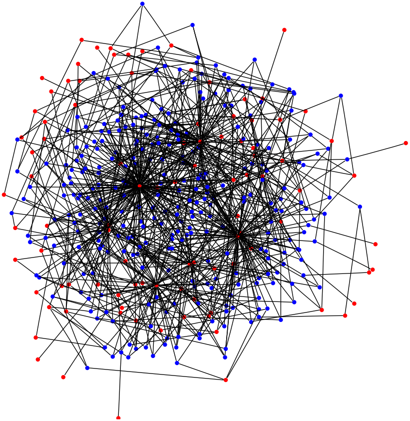
Arity of nodes: number of variable occurrences $\approx 4.25 * 3 = 12.75$

and clause size ≈ 3

Random 3-CNF



Real-World Instance



Real-world instances contain hubs, are scale-free graphs !!!

⇒ See our paper at CP'09

Main Idea

Choose variables (and clauses) randomly following a non-uniform probability distribution

$$P[X = i] = \phi(i; n)$$

The distribution depends on the number of variables

Example: Take $\phi(i; n) = \frac{1-b}{1-b^n} b^i$ with $0 < b \leq 1$

Construct clauses of size 3 randomly selecting 3 distinct variables following this probability distribution

Main Idea

Choose variables (and clauses) randomly following a non-uniform probability distribution

$$P[X = i] = \phi(i; n)$$

The distribution depends on the number of variables

Example: Take $\phi(i; n) = \frac{1-b}{1-b^n} b^i$ with $0 < b \leq 1$

Construct clauses of size 3 randomly selecting 3 distinct variables following this probability distribution

Main Idea

Choose variables (and clauses) randomly following a non-uniform probability distribution

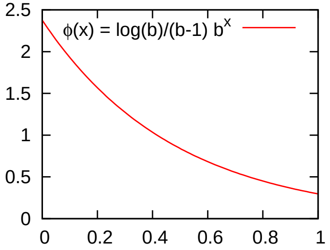
$$P[X = i] = \phi(i; n)$$

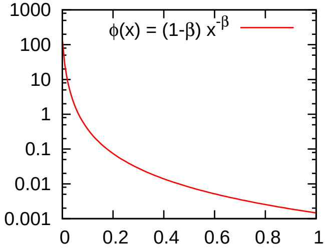
The distribution depends on the number of variables

Example: Take $\phi(i; n) = \frac{1-b}{1-b^n} b^i$ with $0 < b \leq 1$

Construct clauses of size 3 randomly selecting 3 distinct variables following this probability distribution

Problem: the phase-transition point depends on n !!!





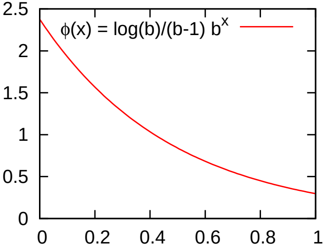
Given a continuous prob. distribution function in [0 1]

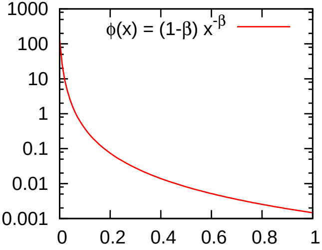
$$\phi^{geo}(x;b) = \frac{\ln(b)}{b-1}\, b^x \qquad \phi^{pow}(x;\beta) = (1-\beta)\, x^{-\beta}$$

Define $P(X = i; n) \propto \phi(i/n)$ taking n equidistant points:

$$P(i;b,n) = \frac{1-b^{1/n}}{1-b}\, b^{i/n} \qquad P(i;\beta,n) = \frac{i^{-\beta}}{\sum_{j=1}^{n} j^{-\beta}}$$

Families of Probability Distributions





Given a continuous prob. distribution function in [0 1]

$$\phi^{geo}(x; b) = \frac{\ln(b)}{b-1}\, b^x \qquad \phi^{pow}(x; \beta, \epsilon) = \frac{1-\beta}{(1+\epsilon)^{1-\beta} - \epsilon^{1-\beta}}\, (x+\epsilon)^{-\beta}$$

Define $P(X = i; n) \propto \phi(i/n)$ taking n equidistant points:

$$P(i; b, n) = \frac{1 - b^{1/n}}{1 - b}\, b^{i/n} \qquad P(i; \beta, \epsilon, n) = \frac{(i + \epsilon \cdot n)^{-\beta}}{\sum_{j=1}^{n} (j + \epsilon \cdot n)^{-\beta}}$$

Input: n, m, k, b
Output: a k-SAT instance with n variables and m clauses

$F = \emptyset$
for $i = 1$ **to** m **do**
  repeat
    $C_i = \square$
    for $j = 1$ **to** k **do**
      $p = rand([0\ 1))$
      $v = 0$
      while $p > Pr(v; b, n)$ **do**
        $v = v + 1$
        $p = p - P(v; b, n)$

      $C_i = C_i \vee (-1)^{rand(\{0,1\})} \cdot v$

  until C_i is not a tautology or simplifiable
  $F = F \cup \{C_i\}$

$rand([0\ 1)) \rightarrow$

0	$P(0; b, n)$
	$P(1; b, n)$
	$P(2; b, n)$
	$\cdots$
1	$P(n-1; b, n)$

Input: n, m, k, b
Output: a k-SAT instance with n variables and m clauses

$bag = \emptyset$
for $v = 1$ **to** n **do**
    $bag = bag \cup \{\lfloor P(v; B, n)\frac{k\,m}{2} \rfloor$ copies of $v\}$
    $bag = bag \cup \{\lfloor P(v; B, n)\frac{k\,m}{2} \rfloor$ copies of $\overline{v}\}$
endfor
$S =$ subset of $k\,m - |bag|$ literals from $\{1, \ldots, n, \overline{1}, \ldots, \overline{n}\}$
    maximizing $Pr(v; b, n)\frac{k\,m}{2} - \lfloor Pr(v; b, n)\frac{k\,m}{2} \rfloor$
$bag = bag \cup S$
repeat
    $F = \emptyset$
    for $i = 1$ **to** m **do**
        $C_i =$ random multiset of k literals from bag
        $bag = bag \setminus C_i$
        $F = F \cup \{C_i\}$
until F does not contain tautologies or simplifiable clauses

Generator of Clauses of Variable Size

Input: $n, m, k, \beta_v, \beta_c$
Output: a SAT instance with n variables, m clauses

```
for i = 1 to m do
        C_i := □;
for i = 1 to k * m do
        repeat
                p := rand(); v := 1;
                while p > P(v; β_v, n) do
                        p := p − P(v; b, n); v := v + 1;
                endwhile
                p := rand(); c := 1;
                while p > P(c; β_c, m) do
                        p := p − P(v; b, n); c := c + 1;
                endwhile
        while v ∈ C_c
        C_c := C_c ∨ (−1)^{rand(2)} · v;
endfor
```

What Distribution Fits Better?

- Analyzed 100 instances of the SAT Race 2008
- $n = 25.693.792$ variables
- $\sum_{i=1}^{n} N(i) = 349.760.681$ occurrences
- $E[N(i)] = \sum_{i=1}^{n} N(i)/n = 13.6$ average number of occurrences

What Distribution Fits Better?

- Analyzed 100 instances of the SAT Race 2008
- $n = 25.693.792$ variables
- $\sum_{i=1}^{n} N(i) = 349.760.681$ occurrences
- $E[N(i)] = \sum_{i=1}^{n} N(i)/n = 13.6$ average number of occurrences
- 90% of variables have less than this number of occurrences
  60% have 6 or less occurrences

What Distribution Fits Better?

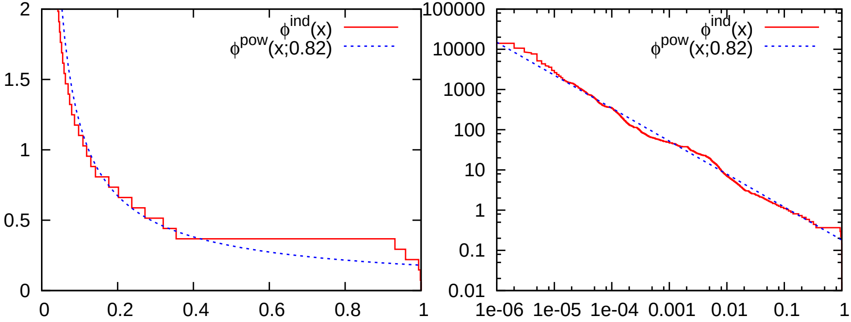
- 90% of variables have less than this number of occurrences
  60% have 6 or less occurrences
- Let $N(i)$ = number of occurrences of the ith most frequent variable $(N(i) \geq N(i+1))$
- Estimate

$$\phi^{ind}(i/n) = \frac{n}{\sum_{j=1}^{n} N(j)} N(i)$$

What Distribution Fits Better?

- Let $N(i)$ = number of occurrences of the ith most frequent variable ($N(i) \geq N(i+1)$)
- Estimate

$$\phi^{ind}(i/n) = \frac{n}{\sum_{j=1}^{n} N(j)}\, N(i)$$



Probabilities of Variables/Occurrences

Do not confuse:

$P(X=i) \sim \phi(i/n)$	Probability that variable i is chosen to be included in a clause
$P(O=k)$	Probability that a randomly chosen variable has k occurrences in the generated formula

Theorem

In the powerlaw model, with $\phi^{pow}(x;\beta) = (1-\beta)\,x^{-\beta}$, when n tends to ∞, the probability that a variable has k occurrences follows a powerlaw distribution $P(k) \sim k^{-\alpha}$, where $\alpha = 1/\beta + 1$.

Probabilities of Variables/Occurrences

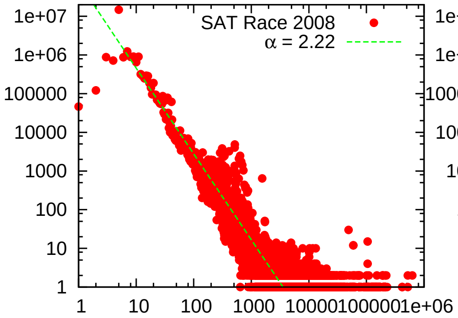
Do not confuse:

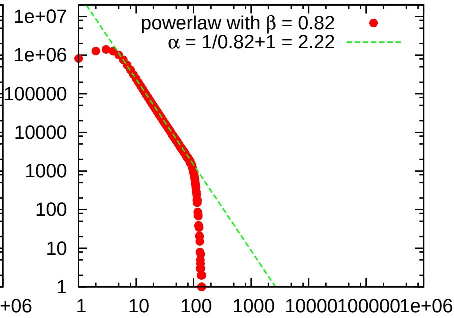
$P(X = i) \sim i^{-\beta}$	Probability that variable i is chosen to be included in a clause
$P(O = k) \sim k^{-\alpha}$	Probability that a randomly chosen variable has k occurrences in the generated formula

Theorem

In the powerlaw model, with $\phi^{pow}(x;\beta) = (1-\beta)\,x^{-\beta}$, when n tends to ∞, the probability that a variable has k occurrences follows a powerlaw distribution $P(k) \sim k^{-\alpha}$, where $\alpha = 1/\beta + 1$.

Prob. Dist. of Num. of Occurrences

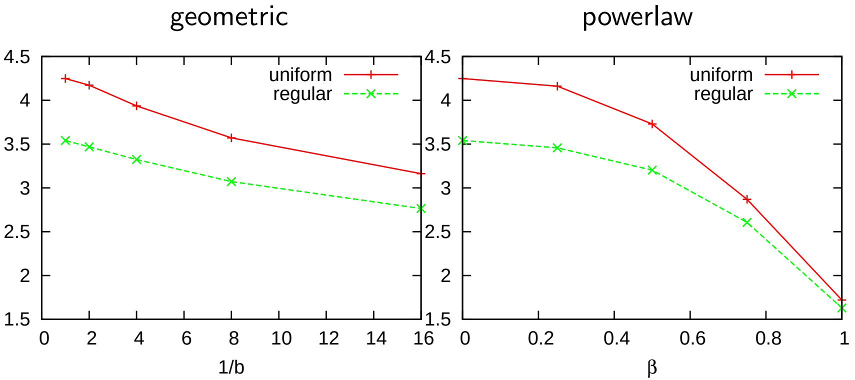




- SAT Race'08 var. occurrences follow a powerlaw distribution with $\alpha = 2.22$
- Random instances generated with $\phi^{pow}(x;\beta)$, where $\beta = 0.82$, too

$$\alpha = 2.22 = \frac{1}{0.82} + 1 = \frac{1}{\beta} + 1$$

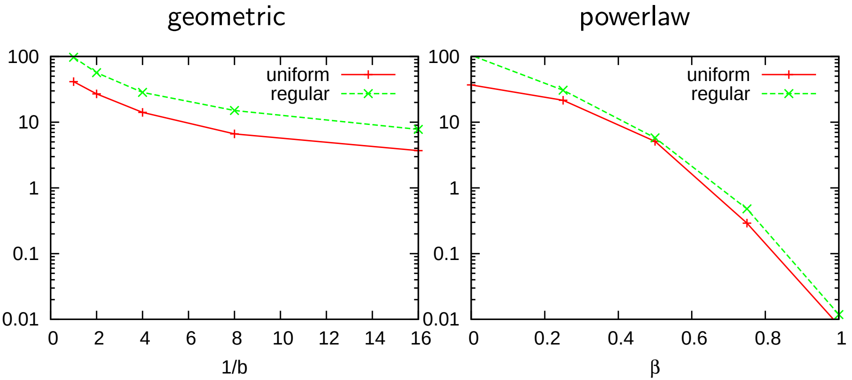
Phase Transition Point



Remember:

geometric model with $b = 1$	=	powerlaw model with $\beta = 0$	=	classical random 3-CNF model

How Industrial Instances Look Like?



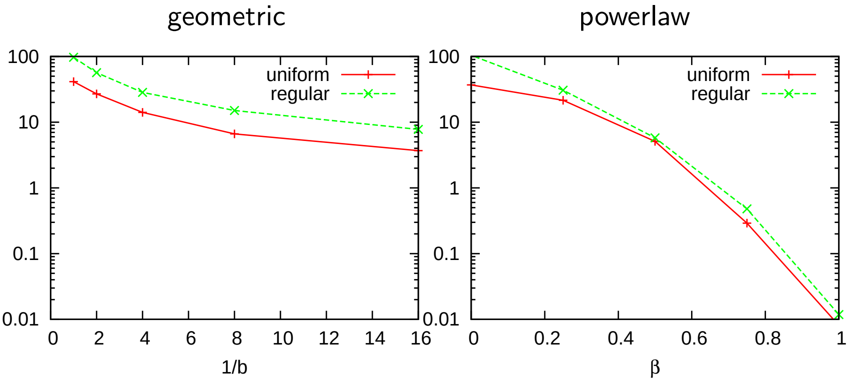
Graphics show:

$$\frac{\text{minisat CPU time}}{\text{kcnf CPU time}}$$

Specialized in industrial inst. (minisat CPU time)

Specialized in random inst. (kcnf CPU time)

How Industrial Instances Look Like?



Graphics show:

$$\frac{\text{minisat CPU time}}{\text{kcnf CPU time}}$$

Minisat beat kcnf in the powerlaw model,
but not in the geometric

Conclusions

- We have capture some properties of some industrial instances like powerlaw distribution of the number of occurrences and of the clause size
- Behavior of solvers show that we are on the right path to understand the nature of real-world instances
- In CP'09, we'll show a more complete study of families of industrial instances and the effect of solvers (learning, instantiation) on the structure of formulas
- Future work: Study other structural properties of industrial instances (symmetries, self-similarity,...)