Analysis and Generation of Pseudo-Industrial MaxSAT Instances

Carlos Ansótegui
DIEI, UdL,
Lleida, Spain

María Luisa Bonet
LSI, UPC,
Barcelona, Spain

Jordi Levy
IIIA, CSIC,
Barcelona, Spain

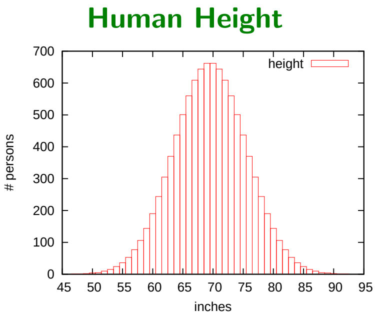
Chu Min Li
Université de Picardie Jules Verne,
Paris, France

CCIA'12, Alacant, Spain

Modeling “Real” Problems

- SAT and MaxSAT are central problems in AI
- They are NP-complete... but this is in the **worst** case
- State-of-the-art SAT and MaxSAT solvers (heuristics, backjumping, learning, restarts, SAT-base MaxSAT solvers ...) are of practical use with **real-world** SAT and MaxSAT instances... **Why???**
- Generating **pseudo-real** instances can help us to solve **real** instances

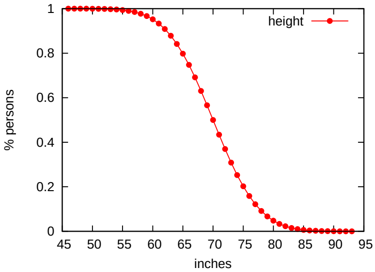
Exponential vs Powerlaw Distributions



- Human heights follows a normal distribution with mean 70 inches

Exponential vs Powerlaw Distributions

Human Height

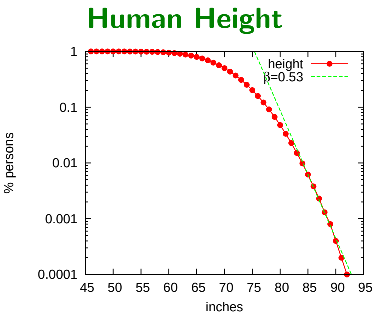


- **Cumulative Distribution:** $F(x) =$% persons with height $\geq x$

$$F(x) = \int_x^\infty f(x)dx$$

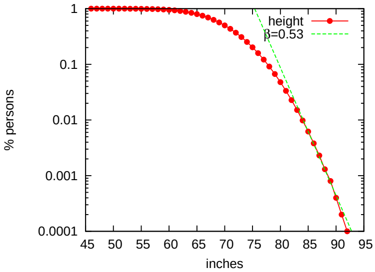
- How decays the **tail**:
  Exponentially: $f^{exp}(x) \sim e^{-\beta x}$ $F^{exp}(x) \sim e^{-\beta x}$
  Powerlaw: $f^{pow}(x) \sim x^{-\alpha}$ $F^{pow}(x) \sim x^{-\alpha+1}$

Exponential vs Powerlaw Distributions



- How decays the **tail**:
  Exponentially: $f^{exp}(x) \sim e^{-\beta\, x}$ $F^{exp}(x) \sim e^{-\beta\, x}$
  Powerlaw: $f^{pow}(x) \sim x^{-\alpha}$ $F^{pow}(x) \sim x^{-\alpha+1}$
- Modify the scale:
  - **Exponential:** we get a line in the **logarithmic** scale
  - **Powerlaw:** we get a line in the **double-logarithmic** scale

Human Height

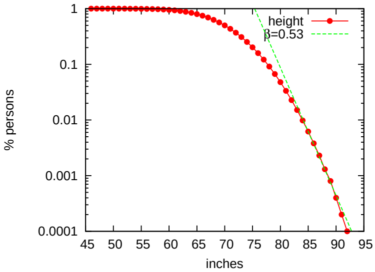


- logarithmic scale
- Decay as

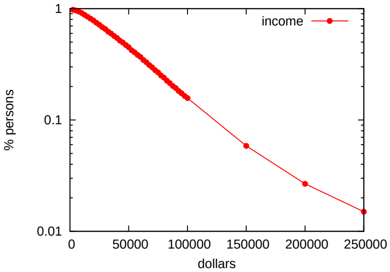
$$F^{exp}(x) \sim e^{-0.53\,x}$$

Giants are very rare

Human Height



Personal Income



- logarithmic scale
- Decay as

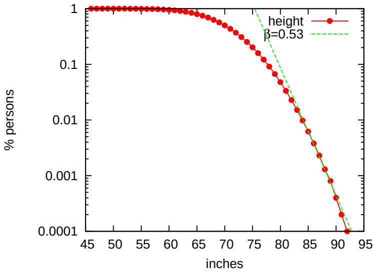
$$F^{exp}(x) \sim e^{-0.53\,x}$$

Giants are very rare

- logarithmic scale

Exponential vs Powerlaw Distributions

Human Height

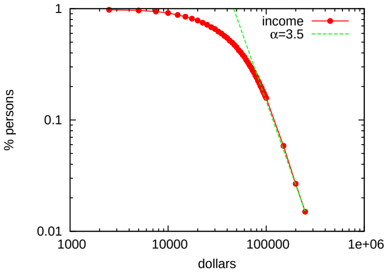


- logarithmic scale
- Decay as

$$F^{exp}(x) \sim e^{-0.53\,x}$$

Giants are very rare

Personal Income



- double-logarithmic scale
- Decay as

$$F^{pow}(x) \sim x^{-2.5}$$

Riches are not so rare

Estimation of the Parameters α, β, Δ and k_{min}

Empirical data $k_1, \dots, k_n$,

- **Estimate α (or β)** maximizing $P(k_1, \dots, k_n \mid \alpha)$

Estimation of the Parameters α, β, Δ and k_{min}

Empirical data $k_1, \ldots, k_n$,

- **Estimate α (or β)** maximizing $P(k_1, \ldots, k_n \mid \alpha)$

$$P^{pow}(k_1, \ldots, k_n \mid \alpha) = \prod_{i=1}^{n} \frac{\alpha-1}{k_{min}} \left(\frac{k_i}{k_{min}}\right)^{-\alpha}$$

$$\frac{\partial}{\partial \alpha} \log P^{pow}(x_1, \ldots, x_n \mid \alpha) =$$
$$= \frac{n}{\alpha-1} - \sum_{i=1}^{n} \log \frac{k_i}{k_{min}} = 0$$

We get

$$\hat{\alpha} = 1 + \frac{n}{\sum_{i=1}^{n} \log(k_i / k_{min})}$$

For the discrete case, a good approx. for larges k_{min} is

$$\hat{\alpha} = 1 + \frac{n}{\sum_{i=1}^{n} \log \frac{k_i}{k_{min}-1/2}}$$

Estimation of the Parameters α, β, Δ and k_{min}

Empirical data $k_1, \dots, k_n$,

- **Estimate α (or β)** maximizing $P(k_1, \dots, k_n \mid \alpha)$

$$P^{pow}(k_1, \dots, k_n \mid \alpha) = \prod_{i=1}^{n} \frac{\alpha-1}{k_{min}} \left(\frac{k_i}{k_{min}}\right)^{-\alpha}$$

$$P^{exp}(k_1, \dots, k_n \mid \beta) = \prod_{i=1}^{n} (1 - e^{-\beta})\, e^{-\beta(k_i - k_{min})}$$

$$\frac{\partial}{\partial\alpha} \log P^{pow}(x_1, \dots, x_n \mid \alpha) = \frac{n}{\alpha-1} - \sum_{i=1}^{n} \log \frac{k_i}{k_{min}} = 0$$

We get

$$\hat{\alpha} = 1 + \frac{n}{\sum_{i=1}^{n} \log(k_i / k_{min})}$$

$$\frac{\partial}{\partial\beta} \log P^{exp}(k_1, \dots, k_n \mid \beta) = \frac{n\, e^{-\beta}}{1-e^{-\beta}} - \sum_{i=1}^{n} (k_i - k_{min}) = 0$$

We get

$$\hat{\beta} = \log\left(\frac{n}{\sum_{i=1}^{n} (k_i - k_{min})} + 1\right)$$

For the discrete case, a good approx. for larges k_{min} is

$$\hat{\alpha} = 1 + \frac{n}{\sum_{i=1}^{n} \log \frac{k_i}{k_{min}-1/2}}$$

Estimation of the Parameters α, β, Δ and k_{min}

Empirical data $k_1, \ldots, k_n$,

- **Estimate α (or β)** maximizing $P(k_1, \ldots, k_n \mid \alpha)$

$$\hat{\alpha} = 1 + \frac{n}{\sum_{i=1}^{n} \log \frac{k_i}{k_{min} - 1/2}} \qquad \hat{\beta} = \log\left(\frac{n}{\sum_{i=1}^{n}(k_i - k_{min})} + 1\right)$$

- Compute the error **for every possible k_{min}** as

$$\Delta^{pow}(k_{min}) = \max_{k \geq k_{min}} \{|F^{emp}(k) - F^{pow}(k; \hat{\alpha}, k_{min})|\}$$

Estimation of the Parameters α, β, Δ and k_{min}

Empirical data $k_1, \ldots, k_n$,

- **Estimate α (or β)** maximizing $P(k_1, \ldots, k_n \mid \alpha)$

$$\hat{\alpha} = 1 + \frac{n}{\sum_{i=1}^{n} \log \frac{k_i}{k_{min}-1/2}} \qquad \hat{\beta} = \log\left(\frac{n}{\sum_{i=1}^{n}(k_i - k_{min})} + 1\right)$$

- Compute the error **for every possible k_{min}** as

$$\Delta^{pow}(k_{min}) = \max_{k \geq k_{min}} \{|F^{emp}(k) - F^{pow}(k; \hat{\alpha}, k_{min})|\}$$

- **Estimate k_{min}** as the value resulting into the smallest error

$$\Delta^{pow} = \min_{k_{min} \geq 1} \Delta(k_{min})$$

- If $\Delta^{pow} > \Delta^{exp}$ the distribution is powerlaw

Two Models of Graphs

Erdös-Rényi Graphs

- For every pair $< v_i, v_j >$, put an edge with probability p
- Arity of nodes follows a binomial distribution

$$P(k) = \binom{n-1}{k} p^k \, (1-p)^{n-1-k}$$

Scalefree Graphs

- There is not a formal definition
- Arity of nodes follows a powerlaw distribution

$$P(k) = C \, k^{-\alpha}$$

- They are also self-similar

SAT Formulas as Bi-partite Graphs

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

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

SAT Formulas as Bi-partite Graphs

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

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

Arity of nodes: **v** number of occurrences

c size

SAT Formulas as Bi-partite Graphs

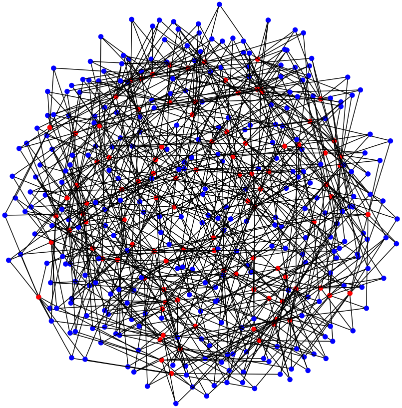
Bi-partite graph: Nodes: are variables **v** and clauses **c**

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

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

c size $= 3$

Random 3-CNF



Number of var occurrences very close to the mean

SAT Formulas as Bi-partite Graphs

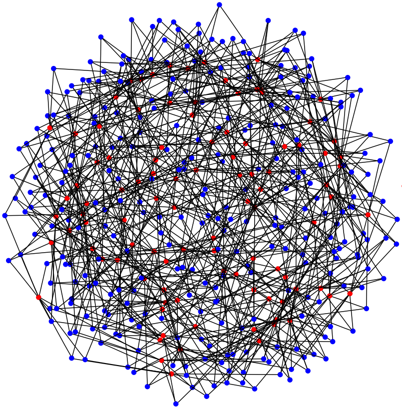
Bi-partite graph: Nodes: are variables v and clauses c

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

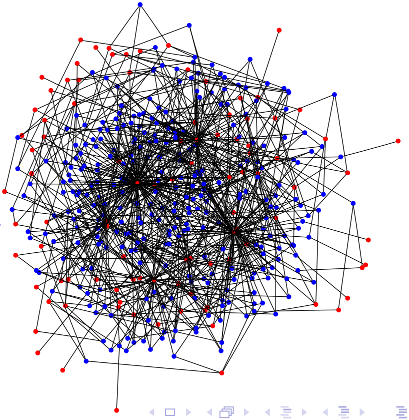
Hubs: v very frequent variables

c very large clauses

Random 3-CNF



Industrial Instance



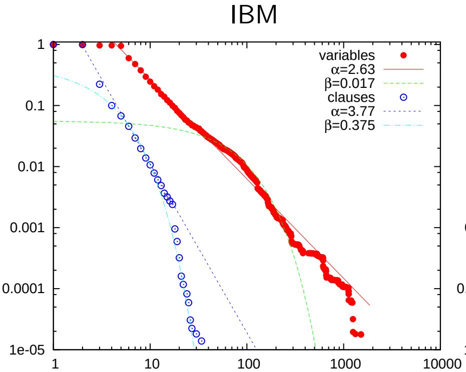
Study of some Families (Variables)

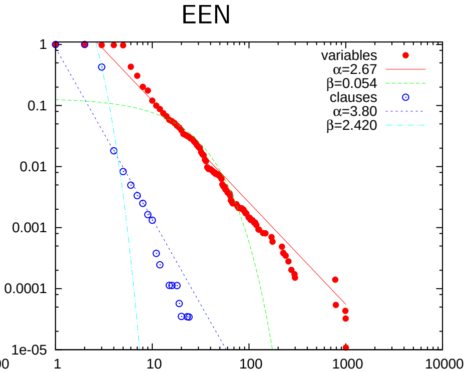
Family	inst	n	$E[V]$	$Var[V]$	Powerlaw α	k_{min}^{pow}	Δ^{pow}	Exponential β	k_{min}^{exp}	Δ^{exp}
cmu	3	16678	7.95	12.11	3.49	5	**0.072**	0.224	4	0.176
een	12	739744	7.60	13.26	2.67	10	**0.043**	0.054	15	0.136
fuhs	2	73486	9.05	14.56	2.79	62	**0.075**	0.181	4	0.158
goldb	11	114038	21.02	88.71	2.05	21	**0.042**	0.003	100	0.204
grieu	9	6914	364.21	42.23	1.77	100	0.577	0.004	100	0.538
ibm	38	4985723	10.75	23.97	2.63	7	**0.027**	0.017	45	0.083
manol	59	7827736	6.93	16.24	2.95	57	0.059	0.017	76	**0.033**
mizh	13	725644	12.49	148.12	4.09	15	0.172	0.034	22	0.247
narai	6	9642548	9.72	16.38	3.85	5	0.152	0.109	1	0.347
palac	2	298266	10.82	60.55	1.84	20	0.087	0.003	100	0.074
post	10	12906872	7.90	44.15	2.57	12	0.132	0.135	1	0.334
schup	7	2196731	8.09	12.40	2.59	41	0.120	0.063	9	0.182
simon	12	798804	7.78	11.96	2.53	14	**0.028**	0.022	50	0.065
uts	10	1420464	13.01	74.70	1.76	69	0.111	0.003	75	**0.088**
velev	60	8442829	88.31	379.04	1.82	13	**0.030**	0.003	87	0.287
random	40	400000	12.75	3.57	18.65	24	0.019	0.777	25	**0.008**
SAT'08	100	27964721	13.30	113.48	2.29	12	**0.051**	0.003	73	0.254

Study of some Families (Clauses)

Family	inst	m	$E[C]$	$Var[C]$	Powerlaw α	k_{min}^{pow}	Δ^{pow}	Exponential β	k_{min}^{exp}	Δ^{exp}
cmu	3	53769	2.46	1.21	5.35	3	0.126	1.778	3	**0.048**
een	12	2278059	2.47	0.69	3.80	4	**0.044**	2.420	3	0.046
fuhs	2	256742	2.59	0.82	4.89	5	0.041	2.182	3	**0.020**
goldb	11	710559	3.37	1.46	10.48	5	0.158	4.803	5	**0.008**
grieu	9	961030	2.62	0.76	8.54	26	0.108	3.878	3	**0.020**
ibm	38	21084555	2.54	1.57	3.77	6	**0.023**	0.375	4	0.032
manol	59	23244626	2.33	0.47						
mizh	13	3036234	2.98	0.91	1.58	1	0.328	0.408	1	0.334
narai	6	37639556	2.49	2.05	3.33	2	**0.088**	1.113	2	0.090
palac	2	1274356	2.53	9.33	1.71	4	0.116	1.055	2	0.116
post	10	42441234	2.40	1.39	3.33	2	0.143	2.884	33	**0.053**
schup	7	6947242	2.56	1.36	4.30	4	0.093	2.585	3	**0.046**
simon	12	2675233	2.32	0.90	3.76	4	0.033	0.498	5	**0.026**
uts	10	7101806	2.60	11.56	3.63	2	0.114	0.004	35	0.116
velev	60	253221473	2.94	9.01	3.35	72	0.042	0.021	28	**0.040**
random	40	1700000	3.00	0.00						
SAT'08	100	140942860	2.64	5.68	3.03	17	**0.054**	0.074	10	0.068

Cumulative Distributions for some Families (1)

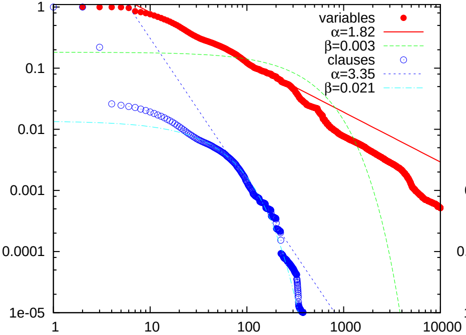




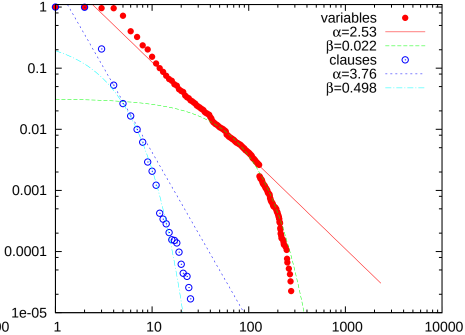
- **Vars** follows a **powerlaw** with $\alpha \approx 2.6$
- **Clauses** follows a **powerlaw** with $\alpha \approx 3.8$

Cumulative Distributions for some Families (2)

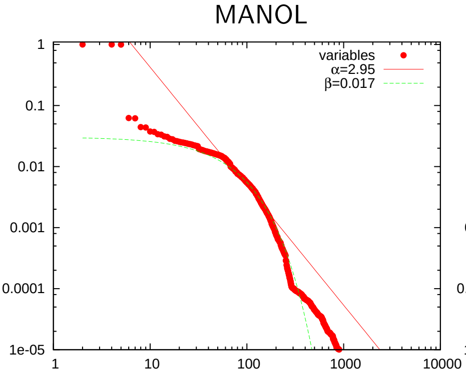
VELEV

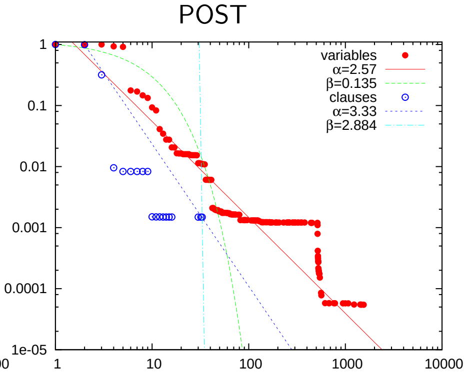


SIMON

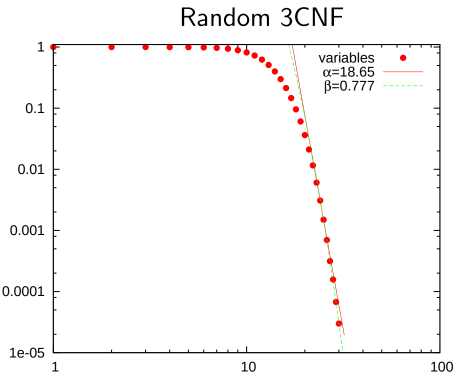


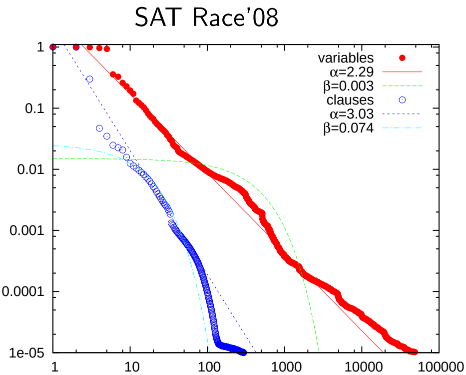
- **Vars** follows a **powerlaw** with $1.8 \leq \alpha \leq 3.5$
- **Clauses** follows an **exponential**





- **Vars** does not seem to follow any "simple" distribution
- neither **Clauses**, or there are very few values

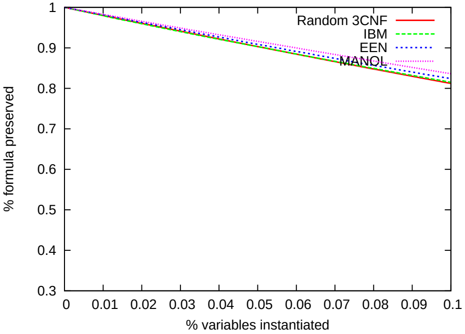




In general:

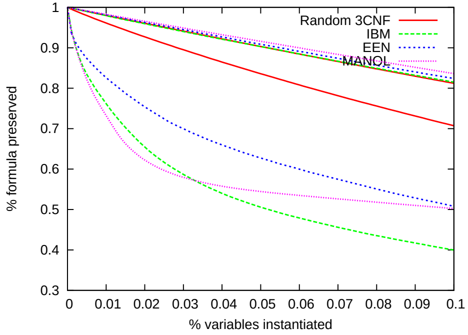
- **Vars** follows a **powerlaw** distribution in more cases than **clauses**
- α is bigger for **clauses** than for **vars**

Instantiating Variables: How Formula Decreases



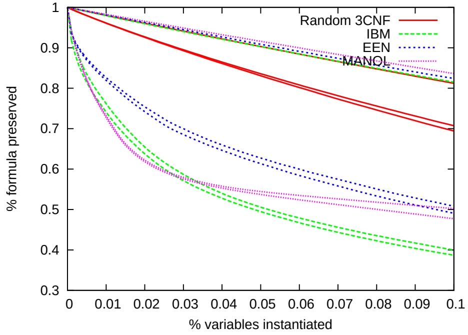
- 3 heuristics: **Random Heuristics (RH)** , **Jeroslow-Wang (JW)** and **Most-Frequent (MF)**
- **RH** has the same effect in all families
- **JW** is better than **RH**
- **JW** is not so good in **random 3CNF** as in **industrial instances**, but still better than **RH**
- **JW** and **MF** have the same effect

Instantiating Variables: How Formula Decreases



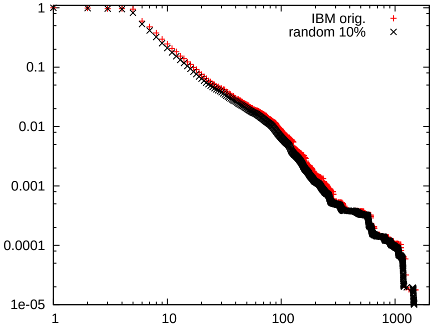
- 3 heuristics: **Random Heuristics (RH)** , **Jeroslow-Wang (JW)** and **Most-Frequent (MF)**
- **RH** has the same effect in all families
- **JW** is better than **RH**
- **JW** is not so good in **random 3CNF** as in **industrial instances**, but still better than **RH**
- **JW** and **MF** have the same effect

Instantiating Variables: How Formula Decreases



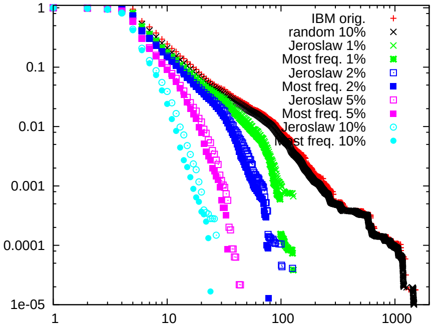
- 3 heuristics: **Random Heuristics (RH)** , **Jeroslow-Wang (JW)** and **Most-Frequent (MF)**
- **RH** has the same effect in all families
- **JW** is better than **RH**
- **JW** is not so good in **random 3CNF** as in **industrial instances**, but still better than **RH**
- **JW** and **MF** have the same effect

Instantiating Variables: How Formula is Modified



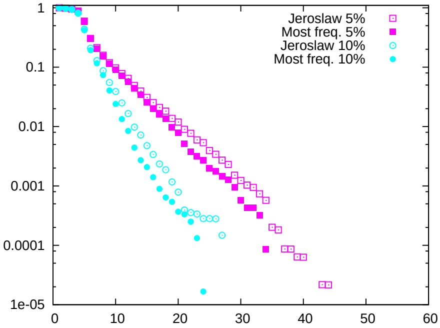
- Experiments with the IBM family
- RH preserve the scalefree structure

Instantiating Variables: How Formula is Modified



- Experiments with the IBM family
- RH preserve the scalefree structure
- JW and MF instantiate frequent variables and could destroy the scalefree structure

Instantiating Variables: How Formula is Modified



- Experiments with the IBM family
- RH preserve the scalefree structure
- JW and MF instantiate frequent variables and could destroy the scalefree structure

Instantiating Variables: How Formula is Modified

	random				Jeroslow-Wang				Most freq.			
	Powerlaw		Exponential		Powerlaw		Exponential		Powerlaw		Exponential	
	α	Δ^{pow}	β	Δ^{exp}	α	Δ^{pow}	β	Δ^{exp}	α	Δ^{pow}	β	Δ^{exp}
0%	**2.63**	**0.027**	0.017	0.083	**2.63**	**0.027**	0.017	0.083	**2.63**	**0.027**	0.017	0.083
1%	**2.56**	**0.027**	0.017	0.078	**2.72**	**0.017**	0.046	0.036	**2.79**	**0.020**	0.052	0.034
2%	**2.57**	**0.025**	0.017	0.076	**2.82**	**0.015**	0.083	0.026	**2.89**	**0.012**	0.093	0.030
5%	**2.59**	**0.020**	0.018	0.075	3.27	0.029	0.218	**0.019**	3.39	0.029	0.250	**0.014**
10%	**2.62**	**0.021**	0.019	0.077	5.79	0.023	0.407	**0.014**	5.90	0.023	0.510	**0.021**

- Experiments with the IBM family
- RH preserve the scalefree structure
  α **is preserved**
- JW and MF destroy the structure after instantiating 5% of the variables
  α **is increased**

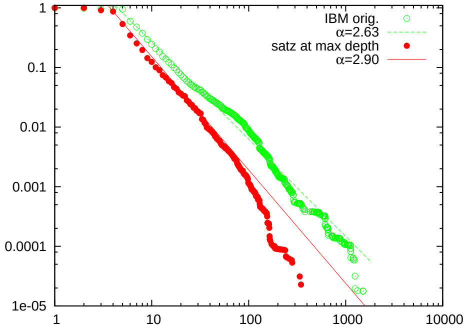
Instantiating Variables: How Formula is Modified

	random				Jeroslow-Wang				Most freq.			
	Powerlaw		Exponential		Powerlaw		Exponential		Powerlaw		Exponential	
	α	Δ^{pow}	β	Δ^{exp}	α	Δ^{pow}	β	Δ^{exp}	α	Δ^{pow}	β	Δ^{exp}
0%	**2.63**	**0.027**	0.017	0.083	**2.63**	**0.027**	0.017	0.083	**2.63**	**0.027**	0.017	0.083
1%	**2.56**	**0.027**	0.017	0.078	**2.72**	**0.017**	0.046	0.036	**2.79**	**0.020**	0.052	0.034
2%	**2.57**	**0.025**	0.017	0.076	**2.82**	**0.015**	0.083	0.026	**2.89**	**0.012**	0.093	0.030
5%	**2.59**	**0.020**	0.018	0.075	3.27	0.029	0.218	**0.019**	3.39	0.029	0.250	**0.014**
10%	**2.62**	**0.021**	0.019	0.077	5.79	0.023	0.407	**0.014**	5.90	0.023	0.510	**0.021**

- Experiments with the IBM family
- RH preserve the scalefree structure
- JW and MF destroy the structure after instantiating 5% of the variables

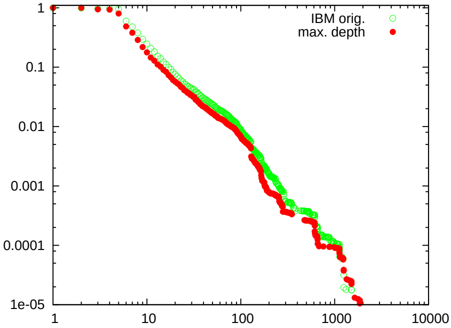
After instantiating 5% vars the assignment is inconsistent

Experiments with Real SAT Solvers



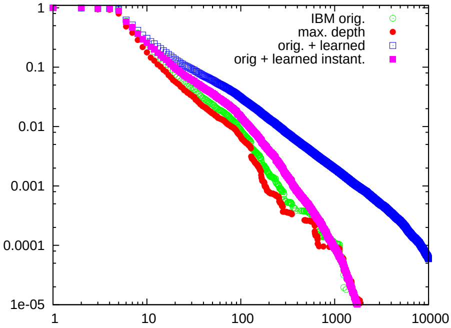
- Solver: Satz Family: IBM
- Formula under the longest assignment found after 1h of search: α is increased but the formula is still scalefree

Experiments with Real SAT Solvers



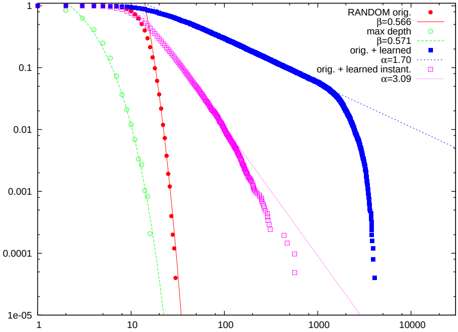
- Solver: Minisat Family: IBM
- Formula under the longest assignment found after 1000s of search: α is increased but the formula is still scalefree

Experiments with Real SAT Solvers



- Solver: Minisat Family: IBM
- Formula under the longest assignment found after 1000s of search: α is increased but the formula is still scalefree
- Original formula plus clauses learned after 200000 decisions: α is decreased substantially
- Instantiated or working formula (plus learned clauses) after 200000 decisions: α is approximately as in the original formula

Experiments with Random 3CNF Formulas



- Solver: Minisat Family: Random 3CNF
- **Learned clauses makes the formula scalefree**
- For the working formula $\alpha \approx 3$

Conclusions

- **Most industrial** SAT formulas seems to be **scalefree**
- **Contrarily to what one could expect** formulas does **not loose scalefree** structure after instantiating **most frequent** variables
- **Learning seems** to help to **preserve scalefree** structure
- Even more, **learning seems** to make **random 3CNF formulas scalefree**
- Solvers specialized in industrial instances **seem** to take profit of their scalefree structure

Open Question:

- **Why** many industrial SAT instances are scalefree? Preferential attachment?
- **Why** the scalefree structure helps? Is it true?
- **How** we can take more advantage of the structure?