The Next Generation of Automated Reasoning Methods

Bart Selman
Cornell University

Joint work with Carla Gomes.

The Quest for Machine Reasoning

Objective:

Develop foundations, technology, and tools to enable effective practical machine reasoning.

Machine Reasoning (1960-90s) ⟹ **Current reasoning technology**

Computational complexity of reasoning *appears* to severly limit real-world applications.

Revisiting the challenge: Significant progress with new ideas / tools for dealing with complexity (scale-up), uncertainty, and multi-agent reasoning.

Fundamental challenge: Combinatorial Search Spaces

Significant progress in the last decade.

How much?

For propositional reasoning:

-- We went from 100 variables, 200 clauses (early 90's)
to 1,000,000 vars. and 5,000,000 constraints in
10 years. Search space: from 10^30 to 10^300,000.

-- Applications: Hardware and Software Verification,
Test pattern generation, Planning, Protocol Design,
Routers, Timetabling, E-Commerce (combinatorial
auctions), etc.

How can deal with such large combinatorial spaces and still do a decent job?

I'll discuss recent formal insights into combinatorial search spaces and their practical implications that makes searching such ultra-large spaces possible.

Brings together ideas from physics of disordered systems (spin glasses), combinatorics of random structures, and algorithms.

But first, what is BIG?

What is BIG?

Consider a real-world Boolean Satisfiability (SAT) problem

The instance `bmc-ibm-6.cnf`, IBM LSU 1997:

```
p cnf
−1 7 0
−1 6 0
−1 5 0
−1 −4 0
−1 3 0
−1 2 0
−1 −8 0
−9 15 0
−9 14 0
−9 13 0
−9 −12 0
−9 11 0
−9 10 0
−9 −16 0
−17 23 0
−17 22 0
```

I.e., ((not x_1) or x_7)
((not x_1) or x_6)
etc.

x_1, x_2, x_3, etc. our Boolean variables (set to True or False)

Set x_1 to False ??

10 pages later:

185 −9 0
185 −1 0
177 169 161 153 145 137 129 121 113 105 97
89 81 73 65 57 49 41
33 25 17 9 1 −185 0
186 −187 0
186 −188 0

…

I.e., (x_177 or x_169 or x_161 or x_153 …
x_33 or x_25 or x_17 or x_9 or x_1 or (not x_185))

clauses / constraints are getting more interesting…

Note x_1 …

4000 pages later:

10236 −10050 0
10236 −10051 0
10236 −10235 0
10008 10009 10010 10011 10012 10013 10014
 10015 10016 10017 10018 10019 10020 10021
 10022 10023 10024 10025 10026 10027 10028
 10029 10030 10031 10032 10033 10034 10035
 10036 10037 10086 10087 10088 10089 10090
 10091 10092 10093 10094 10095 10096 10097
 10098 10099 10100 10101 10102 10103 10104
 10105 10106 10107 10108 −55 −54 53 −52 −51 50
 10047 10048 10049 10050 10051 10235 −10236 0
10237 −10008 0
10237 −10009 0
10237 −10010 0

…

Finally, 15,000 pages later:

$-7\ 260\ 0$
$7\ -260\ 0$
$1072\ 1070\ 0$
$-15\ -14\ -13\ -12\ -11\ -10\ 0$
$-15\ -14\ -13\ -12\ -11\ 10\ 0$
$-15\ -14\ -13\ -12\ 11\ -10\ 0$
$-15\ -14\ -13\ -12\ 11\ 10\ 0$
$-7\ -6\ -5\ -4\ -3\ -2\ 0$
$-7\ -6\ -5\ -4\ -3\ 2\ 0$
$-7\ -6\ -5\ -4\ 3\ -2\ 0$
$-7\ -6\ -5\ -4\ 3\ 2\ 0$
$185\ 0$

Combinatorial search space of truth assignments:

HOW?

$$2^{50000} \approx 3.160699437 \cdot 10^{15051}$$

Current SAT solvers solve this instance in approx. 1 minute!

Progress SAT Solvers

Instance	Posit' 94	Grasp' 96	Sato' 98	Chaff' 01
ssa2670-136	40,66s	1,2s	0,95s	0,02s
bf1355-638	1805,21s	0,11s	0,04s	0,01s
pret150_25	>3000s	0,21s	0,09s	0,01s
dubois100	>3000s	11,85s	0,08s	0,01s
aim200-2_0-no-1	>3000s	0,01s	0s	0s
2dlx_..._bug005	>3000s	>3000s	>3000s	2,9s
c6288	>3000s	>3000s	>3000s	>3000s

Source: Marques Silva 2002

From academically interesting to practically relevant.

We now have regular SAT solver competitions.

Germany ’89, Dimacs ’93, China ’96, SAT-02, SAT-03, SAT-04, SAT05.

E.g. at SAT-2004 (Vancouver, May 04):

--- 35+ solvers submitted

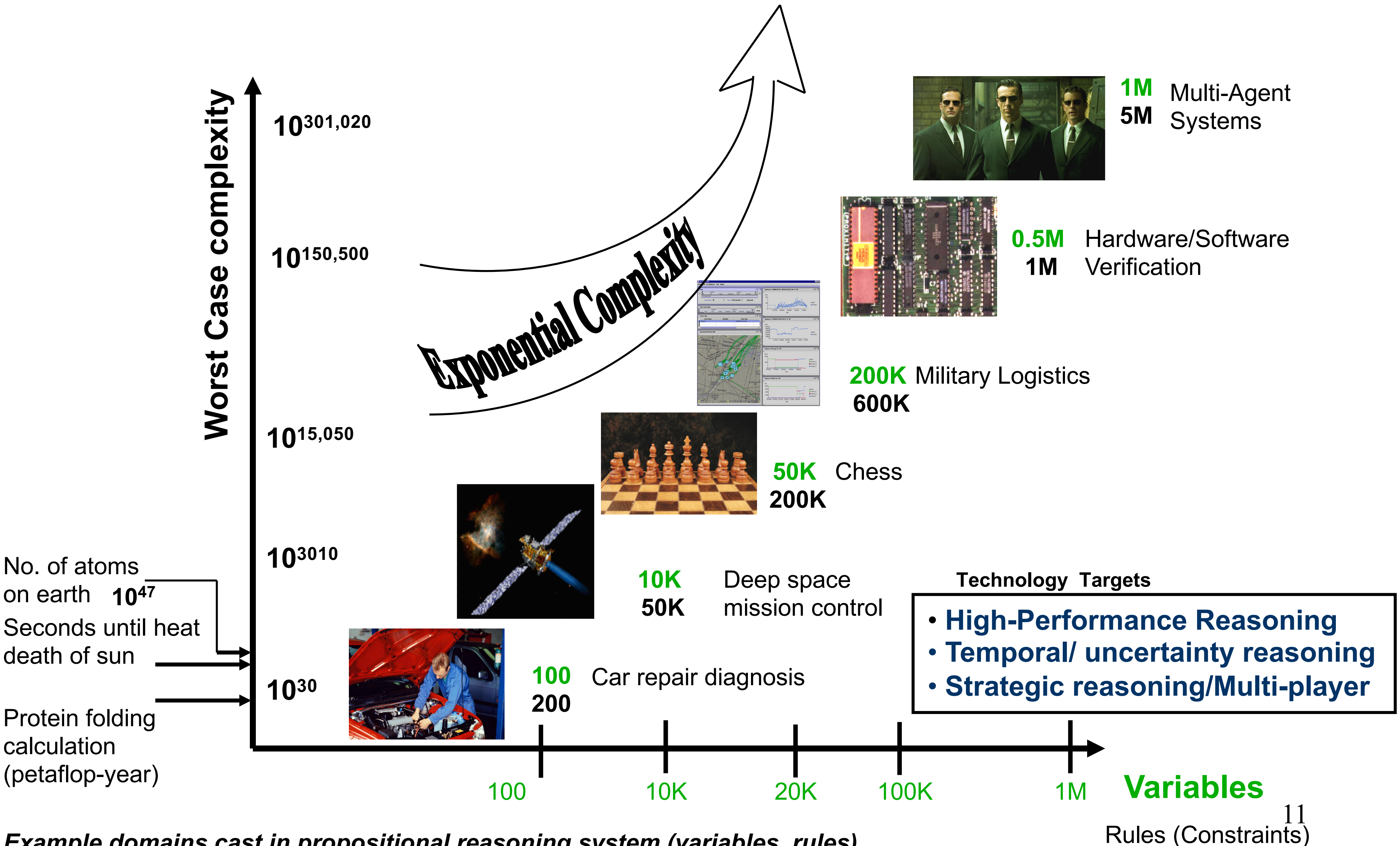
--- 500+ industrial benchmarks

--- 50,000+ instances available on the WWW.

Real-World Reasoning

Tackling inherent computational complexity

DARPA Research Program



Example domains cast in propositional reasoning system (variables, rules).

A Journey from Random to Structured Instances

I --- Random Instances

--- phase transitions and algorithms

--- from physics to computer science

II --- Capturing Problem Structure

--- problem mixtures (tractable / intractable)

--- backdoor variables, restarts, and heavy tails

III --- Beyond Satisfaction

--- sampling, counting, and probabilities

--- quantification

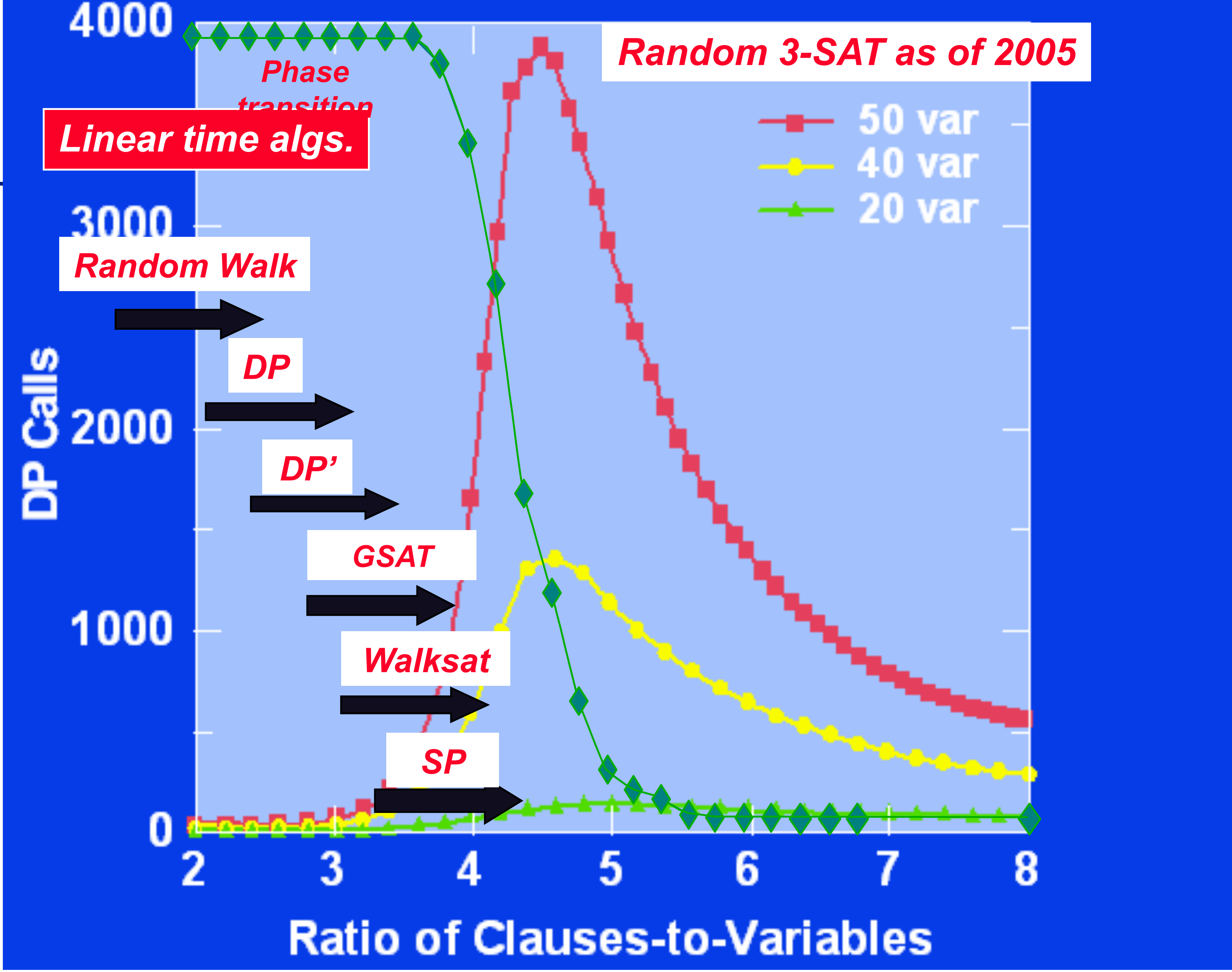
Part I) ---- Random Instances

Easy-Hard-Easy patterns (computational) and
SAT/UNSAT phase transitions ("structural").

Their study provides an interplay of work from statistical physics, computer science, and combinatorics.

We'll briefly consider "The State of Random 3-SAT".

("phase transitions in computational problems";
also CAM '04 talk by Jennifer Chayes)



Mitchell, Selman, and Levesque '92

Linear time results --- Random 3-SAT

Random walk up to ratio 1.36 (Alekhnovich and Ben Sasson 03).
empirically up to 2.5
Davis Putnam (DP) up to 3.42 (Kaporis et al. '02) empirically up to 3.6
exponential, ratio 4.0 and up (Achlioptas and Beame '02)
approx. 400 vars at phase transition
GSAT up till ratio 3.92 (Selman et al. '92, Zecchina et al. '02)
approx. 1,000 vars at phase transition
Walksat up till ratio 4.1 (empirical, Selman et al. '93)
approx. 100,000 vars at phase transition
Survey propagation (SP) up till 4.2
(empirical, Mezard, Parisi, Zecchina '02)
approx. 1,000,000 vars near phase transition

Unsat phase: little algorithmic progress.
Exponential resolution lower-bound (Chvatal and Szemeredi 1988)

Linear time results --- Random 3-SAT

Random walk up to ratio 1.36 (Alekhnovich and Ben Sasson 03).
empirically up to 2.5
Davis Putnam (DP) up to 3.42 (Kaporis et al. '02) empirically up to 3.6
exponential, ratio 4.0 and up (Achlioptas and Beame '02)
approx. 400 vars at phase transition
GSAT up till ratio 3.92 (Selman et al. '92, Zecchina et al. '02)
approx. 1,000 vars at phase transition
Walksat up till ratio 4.1 (empirical, Selman et al. '93)
approx. 100,000 vars at phase transition
Survey propagation (SP) up till 4.2
(empirical, Mezard, Parisi, Zecchina '02)
approx. 1,000,000 vars near phase transition

Unsat phase: little algorithmic progress.
Exponential resolution lower-bound (Chvatal and Szemeredi 1988)

Linear time results --- Random 3-SAT

Random walk up to ratio 1.36 (Alekhnovich and Ben Sasson 03).
empirically up to 2.5
Davis Putnam (DP) up to 3.42 (Kaporis et al. ’02) empirically up to 3.6
exponential, ratio 4.0 and up (Achlioptas and Beame ’02)
approx. 400 vars at phase transition
GSAT up till ratio 3.92 (Selman et al. ’92, Zecchina et al. ‘02)
approx. 1,000 vars at phase transition
Walksat up till ratio 4.1 (empirical, Selman et al. ’93)
approx. 100,000 vars at phase transition
Survey propagation (SP) up till 4.2
(empirical, Mezard, Parisi, Zecchina ’02)
approx. 1,000,000 vars near phase transition

Unsat phase: little algorithmic progress.
Exponential resolution lower-bound (Chvatal and Szemeredi 1988)

Linear time results --- Random 3-SAT

Random walk up to ratio 1.36 (Alekhnovich and Ben Sasson 03).
empirically up to 2.5

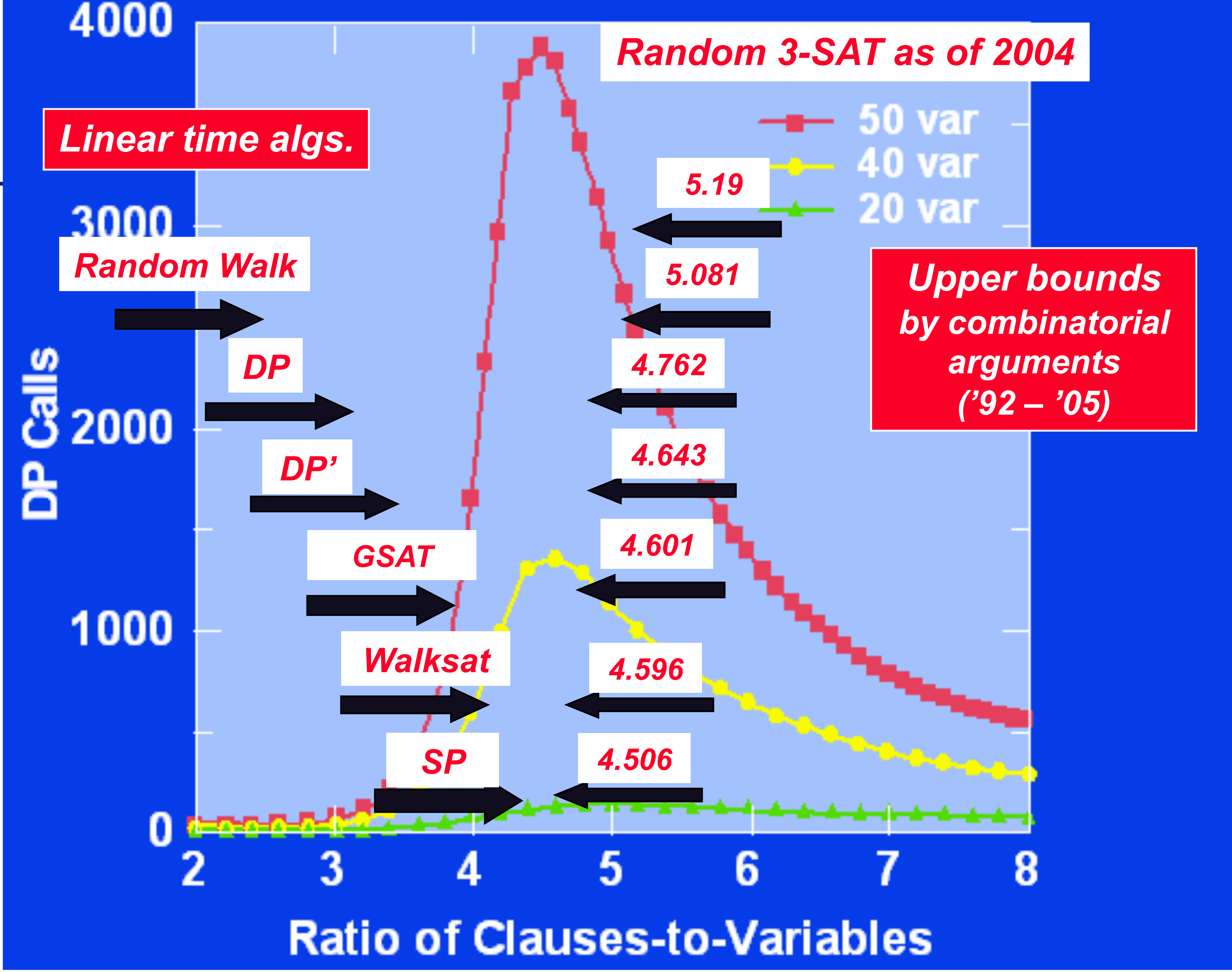
Davis Putnam (DP) up to 3.42 (Kaporis et al. ’02) empirically up to 3.6
exponential, ratio 4.0 and up (Achlioptas and Beame ’02)
approx. 400 vars at phase transition

GSAT up till ratio 3.92 (Selman et al. ’92, Zecchina et al. ‘02)
approx. 1,000 vars at phase transition

Walksat up till ratio 4.1 (empirical, Selman et al. ’93)
approx. 100,000 vars at phase transition

Survey propagation (SP) up till 4.2
(empirical, Mezard, Parisi, Zecchina ’02)
approx. 1,000,000 vars near phase transition

Unsat phase: little algorithmic progress.
Exponential resolution lower-bound (Chvatal and Szemeredi 1988)


Random 3-SAT as of 2004
Linear time algs.
Random Walk
DP
DP'
GSAT
Walksat
SP
5.19
5.081
4.762
4.643
4.601
4.596
4.506
Upper bounds by combinatorial arguments ('92 – '05)
50 var
40 var
20 var
DP Calls
4000
3000
2000
1000
0
2
3
4
5
6
7
8
Ratio of Clauses-to-Variables

More details on how to derive 5.19 point

Calculate the expected number of satisfying assignments given a set of M random clauses on N vars.

Total num possible assignments is 2^N.

Let S_i be an indicator variable for truth assignment i. That is, a random 0/1 variable, s.t., $S_i = 1$ iff i^{th} assignment satisfies the M random clauses (= 0 otherwise).

A random assignment satisfies 1 clause with with probability $7/8$. A random assignment satisfies M random clauses (independently generated) with probability $(7/8)^M$.

We need to calculate

E[#satisfying_assignments]
= E[S_1 + S_2 + S_3 + … + S_2^N]
= E[\sum_all_assigns S_i]
= \sum_all_assigns E[S_i])   % by linearity of expectation
= \sum_all_assigns (⅞)^M   % ⅞^M is prob. S_i = 1 for i^th assign after M clauses
= 2^N . (⅞)^M   % sum over 2^N assignments

Note: “Competition” between 2^N   and (⅞)^M.

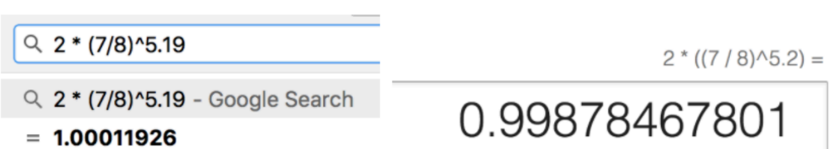
Now, we want to know, when N → \inf, for what ratio of M/N does
E[#satisfying assignments] go to 0?

Now, we want to know, when N → \inf, for what ratio of M/N does E[#satisfying assignments] go to 0?

Consider what 2^N (⅞)^M looks like with M/N = 5.19 and N ---> \inf.
Let M = 5.19 N. We get 2^N (⅞)^(5.19 N) = (2 . (⅞)^5.19)^N \approx **1.00011926^N**.

Also, M = 5.0 N. We get **0.99878^N**. Goes to 0.0! No solutions in expectation!!!!

So, if we take slightly more clauses over 5.19 ratio, base tips below 1.0 and with N to \infinity, total goes to 0. I.e., Prob(#assign = 0) → 1.0. Formulas become unsat at transition point when



Aside: Why is real threshold point lower? “Correlations.” But better explanation is “solution clustering.” Discuss details.

Aside: **linearity of expectation** **(the key tool in most CS probabilistic arguments!)** is surprisingly powerful but also remains quite counterintuitive. It applies to any sum of r.v.'s even is they are fully correlated. E.g., two dice that are connected by an invisible wire that forces them to always show both the same number. Of course, watching just one die, you will see ⅙ chance of each number but you will only see 1-1, 2-2, 3-3, 4-4, 5-5, 6-6 and never 1-2 etc. The expected value of the sum is still E[X + Y] = E[X] + E[Y] = 2 E[X] = 2 Expected_value_of_die_roll = ⅙ (1 + 2 + 3 +4 + 5 + 6) = ⅙ * 21 = 3.5. So, E[X + Y] = 7, **no matter how the die rolls are correlated to each other! (Imagine both dice come up with same value always… 1,1 or 2,2, … or 6,6. In expectation we still get 3.5. Still works!)**

Exact Location of Threshold

Surprisingly challenging problem ...

Current rigorously proved results:

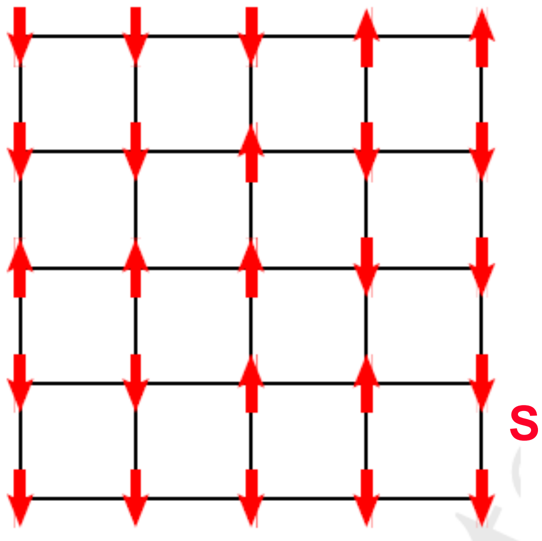
3SAT threshold lies between 3.42 and 4.506.

Motwani et al. 1994; Broder et al. 1992; Frieze and Suen 1996; Dubois 1990, 1997; Kirousis et al. 1995; *Friedgut 1997*; Archlioptas et al. 1999; Beame, Karp, Pitassi, and Saks 1998; Impagliazzo and Paturi 1999; Bollobas, Borgs, Chayes, Han Kim, and Wilson1999; Achlioptas, Beame and Molloy 2001; Frieze 2001; Zecchina et al. 2002; Kirousis et al. 2004; Gomes and Selman, *Nature* '05; Achlioptas et al. *Nature* '05; and ongoing…

Empirical: 4.25 --- Mitchell, Selman, and Levesque '92, Crawford '93.

From Physics to Computer Science

Exploits correspondence between SAT and physical systems with many interacting particles.



$$\text{e.g. spin } x_i \text{ and } x_j \text{ want to align:}$$

$$\left(x_i \vee \neg x_j \right) \wedge \left(\neg x_i \vee x_j \right)$$

Satisfied iff [(x_i = 1 and x_j =1) OR (x_i =0 and x_j=0)]

Basic model for magnetism: The Ising model (Ising '24). Spins are "trying to align themselves". But system can be "frustrated" some pairs want to align; some want to point in the opposite direction of each other.

We can now assign a probability distribution over the assignments/ states --- the Boltzmann distribution:

$$\text{Prob}(S) = 1/Z \; * \; \exp(-\, E(S) \,/\, kT)$$

where,

E is the “energy” = # unsatisfied constraints,
T is the “temperature” a control parameter,
k is the Boltzmann constant, and
Z is the “partition function” (normalizes).

Distribution has a physical interpretation (captures thermodynamic equilibrium) but, for us, key property:

With T → 0, only minimum energy states have non-zero probability. *So, by taking T → 0, we can find properties of the satisfying assignments of the SAT problem.*

In fact, partition function Z, contains all necessary information.

$$Z = \sum \exp(-E(S)/kT)$$

sum is over all 2^N possible states / (truth) assignments.

Are we really making progress here??
Sum over an exponential number of terms, 2^N... in physics, $N \sim 10^{23}$.

Fortunately, physicists have been studying "Z" for 100+ years.
(Feynman Lectures: "Statistical physics = study of Z".)

They have developed a powerful set of analytical tools to calculate / approximate Z : e.g. mean field approximations, Monte Carlo methods, matrix transfer methods, renormalization techniques, replica methods and cavity methods.

Physics contributing to computation

80's --- Simulated annealing

General combinatorial search technique, inspired by physics.

(Kirkpatrick '83)

90's --- Phase transitions in computational systems

Discovery of physical laws and phenomena (e.g. 1^{st} and 2^{nd} order transitions) in computational systems.

Cheeseman et al. '91; Mitchell et al. '92;

Explicit connection to physics:

Kirkpatrick and Selman '94 (finite-size scaling);

Monasson et al.'99. (order phase transition))

'02 --- Survey Propagation

Analytical tool from statistical physics leads to powerful algorithmic method. (Mezard *et al.* '02).

More expected!

Physics contributing to computation

80's --- Simulated annealing

General combinatorial search technique, inspired by physics.

(Kirkpatrick *et al*., *Science* '83)

90's --- Phase transitions in computational systems

Discovery of physical laws and phenomena (e.g. 1st and 2nd order transitions) in computational systems.

(Cheeseman *et al*. '91; Selman *et al*. '92;

Explicit connection to physics:

Kirkpatrick and Selman, *Science* '94 (finite-size scaling);

Monasson *et al*., *Nature* '99. (order of phase transition))

'02 --- Survey Propagation

Analytical tool from statistical physics leads to powerful algorithmic method. (Mezard *et al., Science* '02).

More expected!

A Journey from Random to Structured Instances

I --- Random Instances ✓

--- phase transitions and algorithms

--- from physics to computer science

II --- Capturing Problem Structure

--- problem mixtures (tractable / intractable)

--- backdoor variables, restarts, and heavy-tails

III --- Beyond Satisfaction

--- sampling, counting, and probabilities

--- quantification

Part II) --- Capturing Problem Structure

Results and algorithms for hard random k-SAT problems have had significant impact on development of practical SAT solvers. However…

Next challenge: Dealing with SAT problems with more inherent structure.

Topics (with lots of room for further analysis):

A) Mixtures of tractable/intractable stucture

B) Backdoor variables and heavy tails

II A) Mixtures: The 2+p-SAT problem

Motivation: **Most real-world computational problems involve some mix of tractable and intractable sub-problems.**

Study: mixture of binary and ternary clauses

p = fraction ternary

p = 0.0 --- 2-SAT / p = 1.0 --- 3-SAT

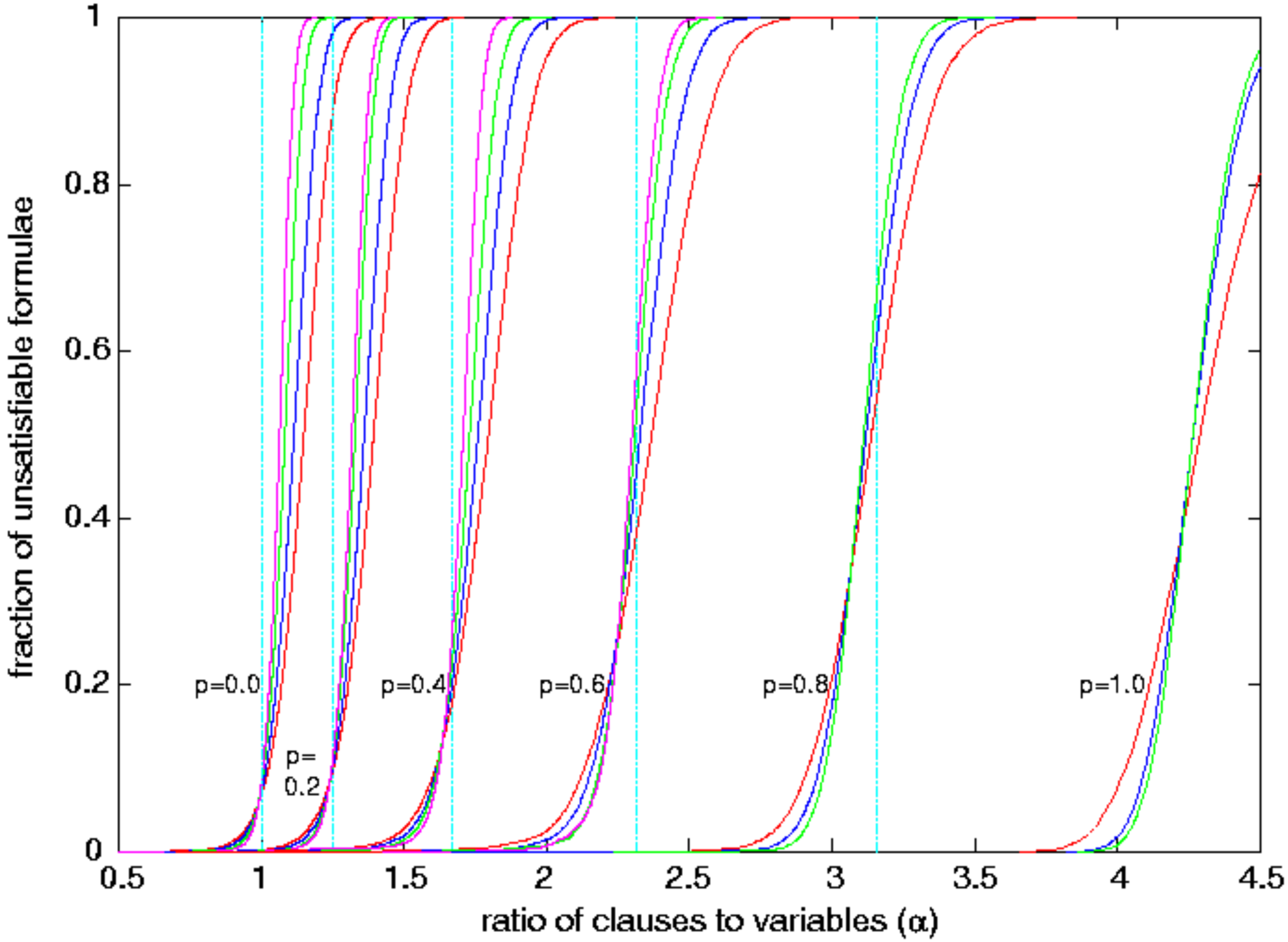
What happens in between?

Phase transitions **(as expected…)**

Computational properties **(surprise…)**

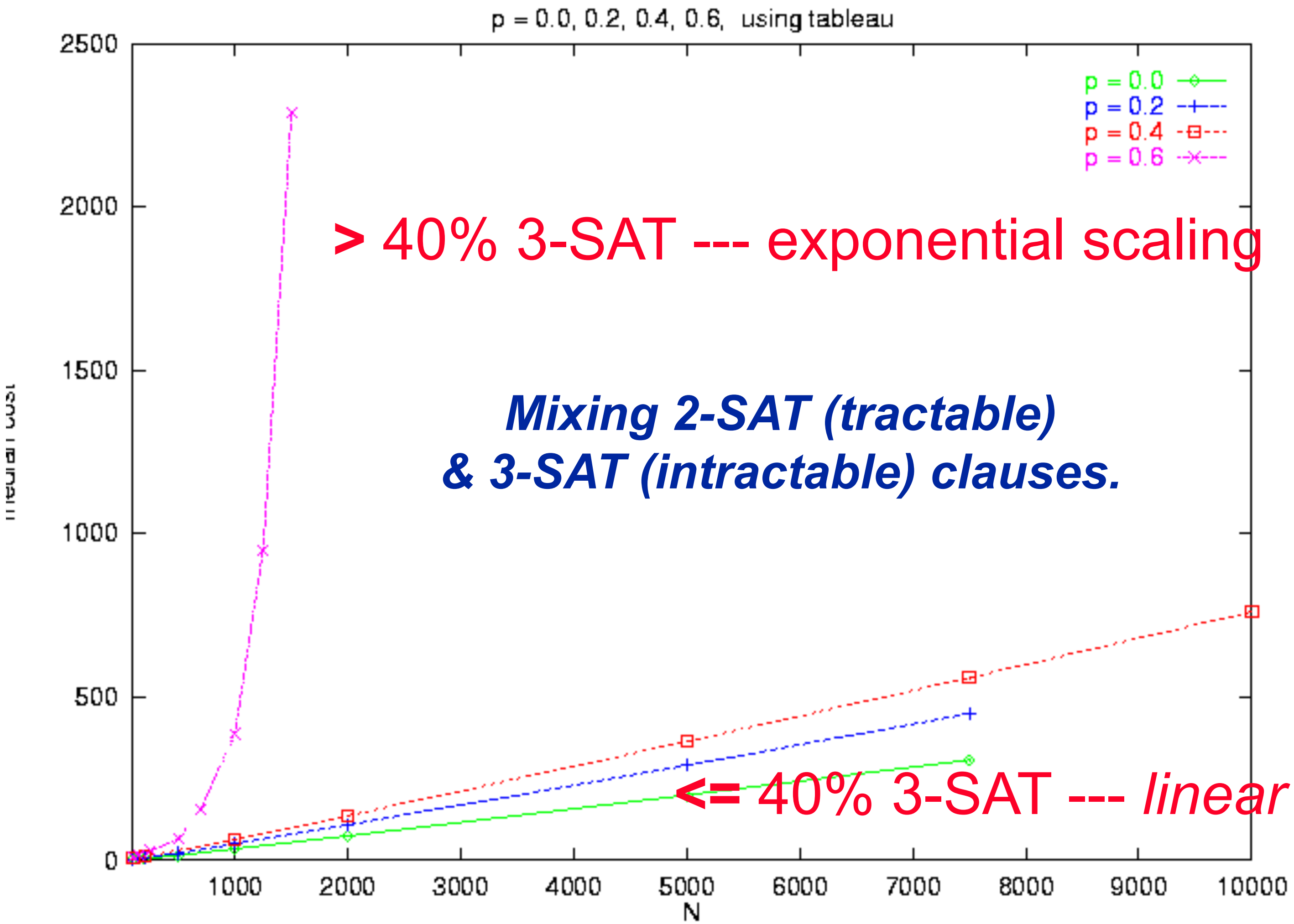
(Monasson, Zecchina, Kirkpatrick, Selman, Troyansky *1999*.)

Phase Transition for 2+p-SAT



We have good approximations for location of thresholds.

Computational Cost: 2+p-SAT
Tractable substructure can dominate!



Medium cost

> 40% 3-SAT --- exponential scaling

Mixing 2-SAT (tractable)
& 3-SAT (intractable) clauses.

<= 40% 3-SAT --- *linear scaling*

Num variables

(Monasson et al. 99; Achlioptas '00)

Results for 2+p-SAT

$p <= 0.4$ --- model behaves as 2-SAT

search proc. "sees" only binary constraints

smooth, continuous phase transition (2^{nd} order)

$p > 0.4$ --- behaves as 3-SAT (exponential scaling)

abrupt, discontinuous transition (1^{st} order)

Note: problem is NP-complete for any $p > 0$.

Lesson learned

In a worst-case intractable problem --- such as 2+p-SAT --- *having a sufficient amount of tractable problem substructure (possibly hidden)* can lead to provably *poly-time* average case behavior.

Next:

Capturing hidden problem structure.

(Gomes et al. 03, 04)

II B) --- Backdoors to the real-world

Observation: Complete backtrack style search SAT solvers (e.g. DPLL) display a remarkably wide range of run times.

Even when repeatedly solving the same problem instance; variable branching is choice randomized.

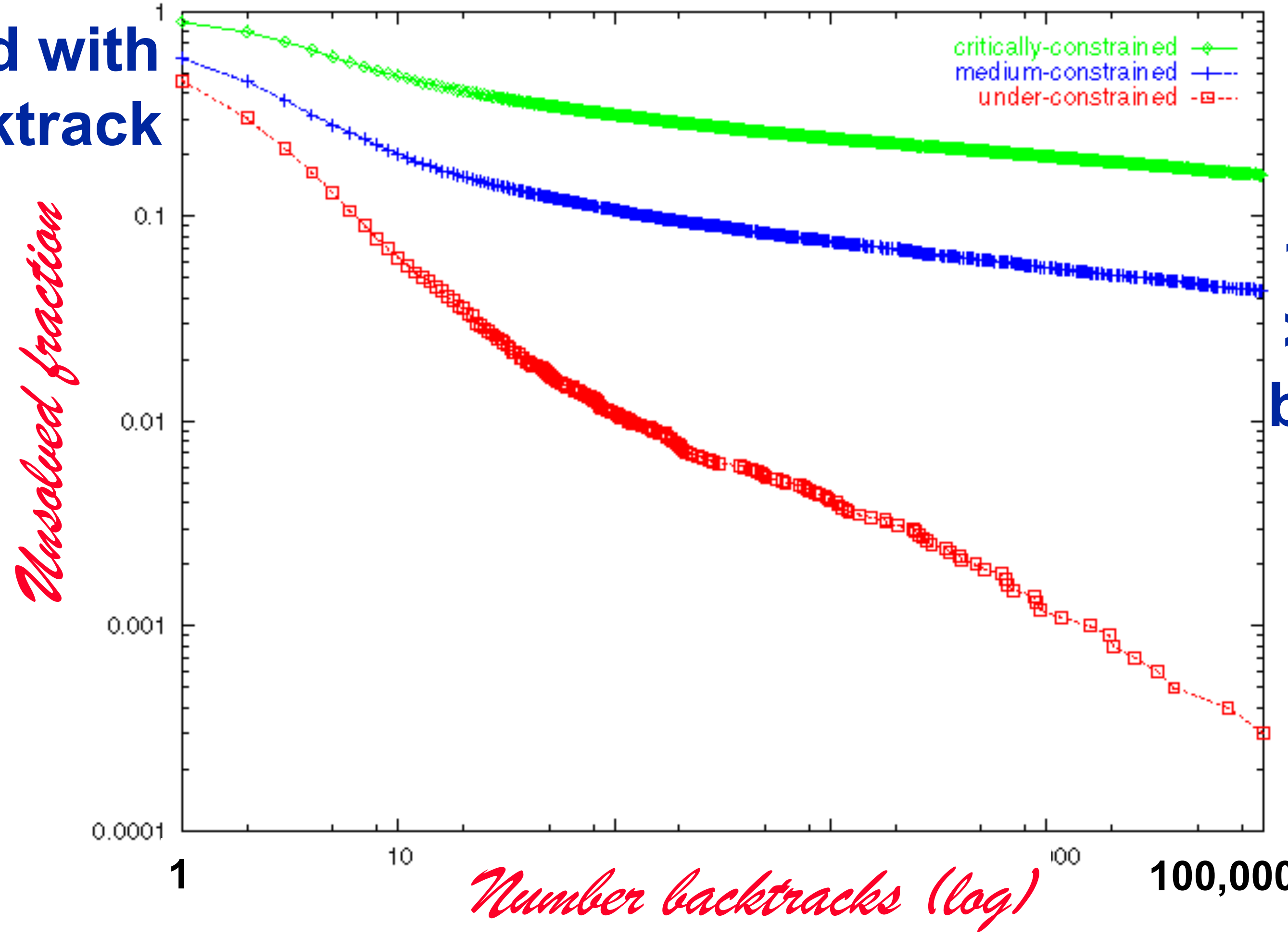
Run time distributions are often "heavy-tailed".

Orders of magnitude difference in run time on different runs.

(Gomes et al. 1998; 2000)

Heavy-tails on structured problems

50% runs: solved with 1 backtrack



10% runs: > 100,000 backtracks

Randomized Restarts

Solution: randomize the backtrack strategy

- Add noise to the heuristic branching (variable choice) function
- Cutoff and restart search after a fixed number of backtracks

Provably Eliminates heavy tails

In practice: rapid restarts with low cutoff can dramatically improve performance

(Gomes et al. 1998, 1999)

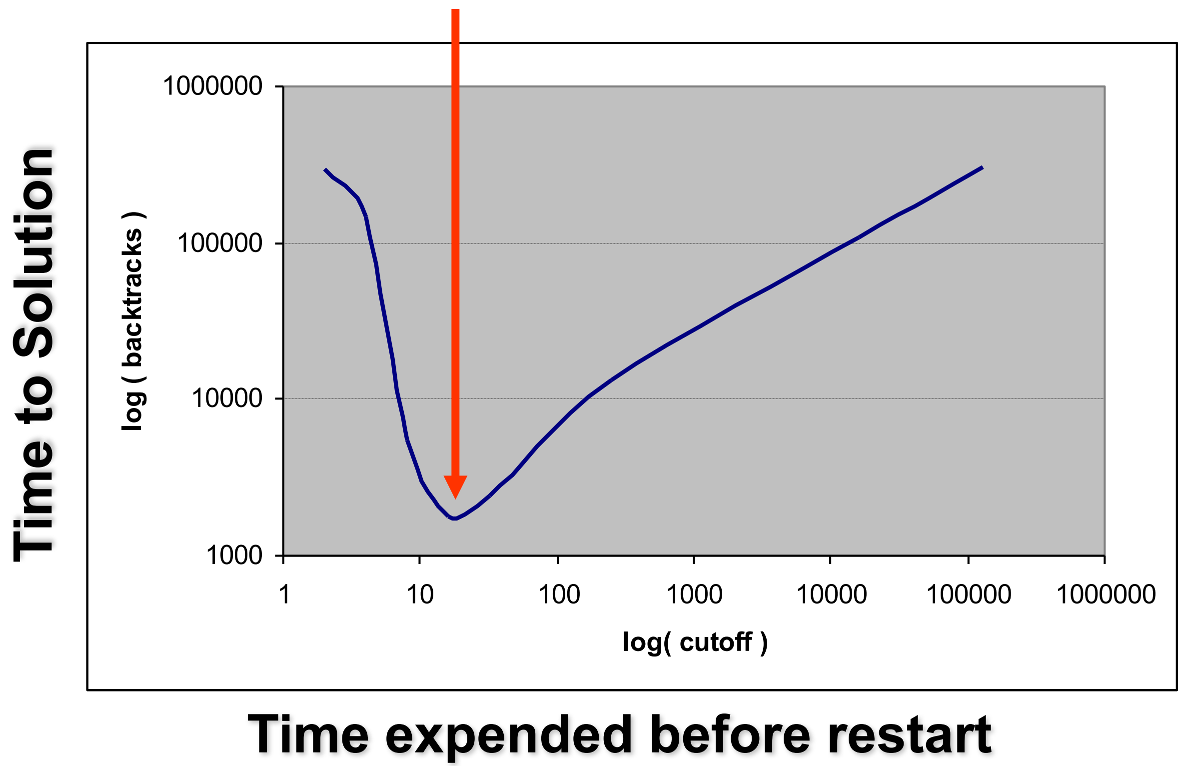
Exploited in many current SAT solvers combined with clause learning and non-chronological backtracking. (Chaff etc.)

Restarts on Planning Problem

Consider simple fixed policy:

> ***Restart search if run-time is greater than x***

Order magnitude speedup.



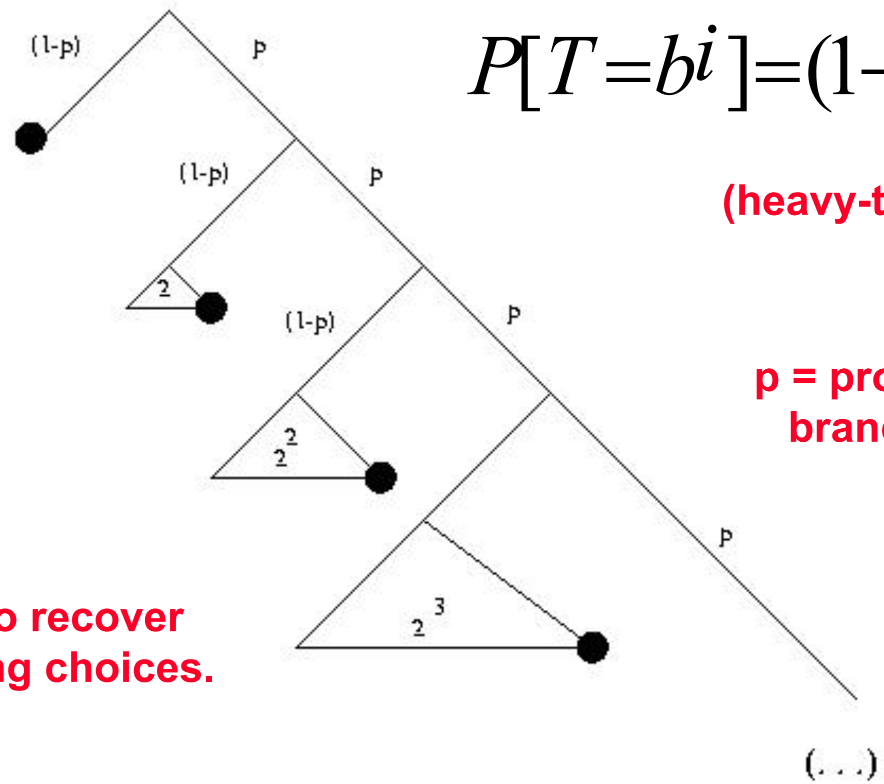
Sample Results Random Restarts

	Deterministic	R^3
Logistics Planning	108 mins.	95 sec.
Scheduling 14	411 sec	250 sec
Scheduling 16	---(*)	1.4 hours
Scheduling 18	---(*)	~18 hrs
Circuit Synthesis 1	---(*)	165sec.
Circuit Synthesis 2	---(*)	17min.

(*) not found after 2 days

Formal Model Yielding Heavy-Tailed Behavior

T - the number of leaf nodes visited up to and including the successful node; b - branching factor



$$P[T = b^i] = (1-p)\,p^i \quad i \geq 0$$

(heavy-tailed distribution)

p = probability wrong branching choice.

2^k time to recover from k wrong choices.

b = 2

● successful leaf

(Chen, Gomes, and Selman '01; Williams, Gomes, and Selman'03)

Expected Run Time

$$p \geq 1/b \quad E[T] \rightarrow \infty$$

(infinite expected time)

Variance

$$p > 1/b^2 \quad V[T] \rightarrow \infty$$

(infinite variance)

Tail

$$p \geq 1/b^2 \quad P[T > L] > C L^{\alpha} \quad \alpha < 2$$

(heavy-tailed)

Balancing exponential decay in making wrong branching decisions with exponential growth in cost of mistakes.

(related to sequential de-coding, Berlekamp et al. 1972)

Intuitively: Exponential penalties hidden in backtrack search, consisting of large inconsistent subtrees in the search space.

But, for restarts to be effective, you also need short runs.

Where do short runs come from?

Explaining short runs: Backdoors to tractability

Informally:

A backdoor to a given problem is a subset of the variables such that once they are assigned values, the polynomial propagation mechanism of the SAT solver solves the remaining formula.

Formal definition includes the notion of a “subsolver”:

a polynomial simplification procedure with certain general characteristics found in current DPLL SAT solvers.

Backdoors correspond to “clever reasoning shortcuts” in the search space.

Backdoors (wrt subsolver A; SAT case):

Definition 2. [backdoor] *A nonempty subset S of the variables is a* backdoor *for F w.r.t. A if for some $a_S : S \rightarrow \{False, True\}$, A returns a satisfying assignment of $F[a_S]$.*

Strong backdoors (wrt subsolver A; UNSAT case):

Definition 3. [strong backdoor] *A nonempty subset S of the variables is a* strong backdoor *for C w.r.t. A if for all $a_S : S \rightarrow \{False, True\}$, A returns a satisfying assignment or concludes unsatisfiability of $F[a_S]$.*

Note: Notion of backdoor is related to but different from constraint-graph based notions such as cutsets. (Dechter 1990; 2000)

Explaining short runs: Backdoors to tractability

Informally:

A backdoor to a given problem is a subset of the variables such that once they are assigned values, the polynomial propagation mechanism of the SAT solver solves the remaining formula.

Formal definition includes the notion of a “subsolver”:

a polynomial simplification procedure with certain general characteristics found in current DPLL SAT solvers.

Backdoors correspond to “clever reasoning shorcuts” in the search space.

Backdoors can be surprisingly small:

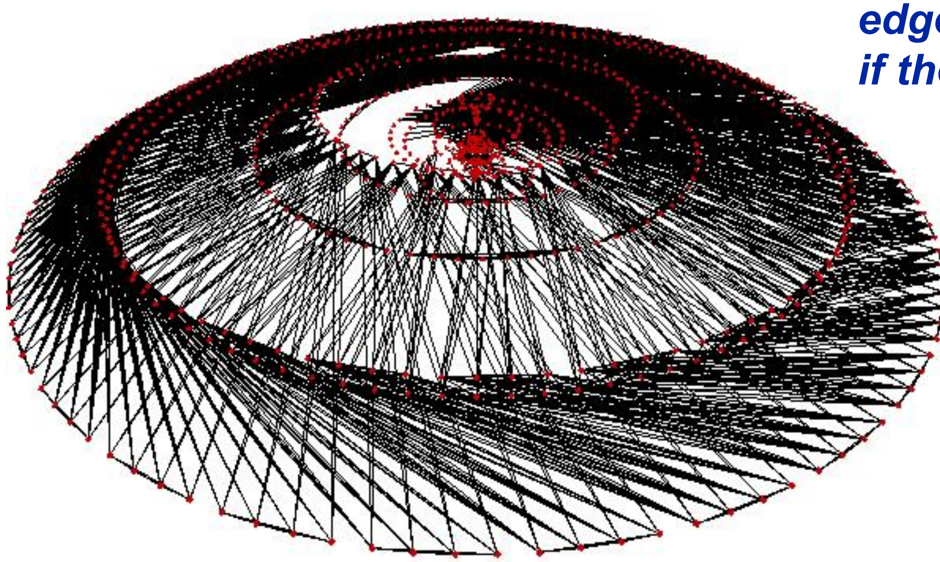
instance	# vars	# clauses	backdoor	fract.
logistics.d	6783	437431	12	0.0018
3bitadd_32	8704	32316	53	0.0061
pipe_01	7736	26087	23	0.0030
qg_30_1	1235	8523	14	0.0113
qg_35_1	1597	10658	15	0.0094

Most recent: Other combinatorial domains. E.g. graphplan planning, near constant size backdoors (2 or 3 variables) and log(n) size in certain domains. ***(Hoffmann, Gomes, Selman '04)***

Backdoors capture critical problem resources (bottlenecks).

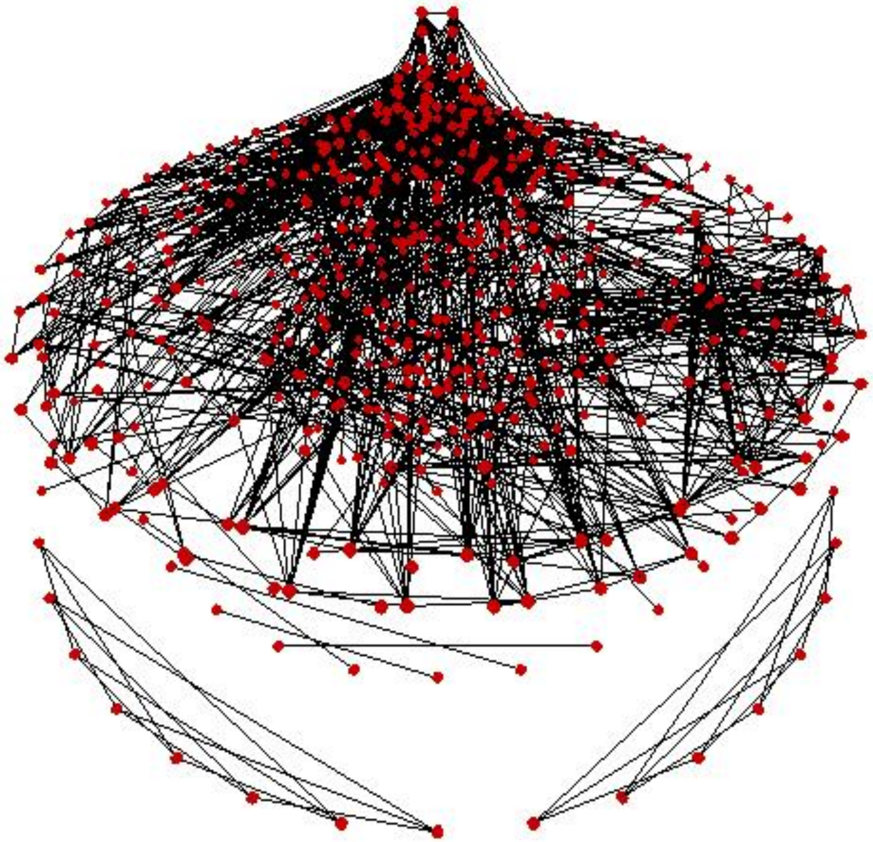
Backdoors --- “seeing is believing”

Constraint graph of reasoning problem.
One node per variable:
edge between two variables if they share a constraint.

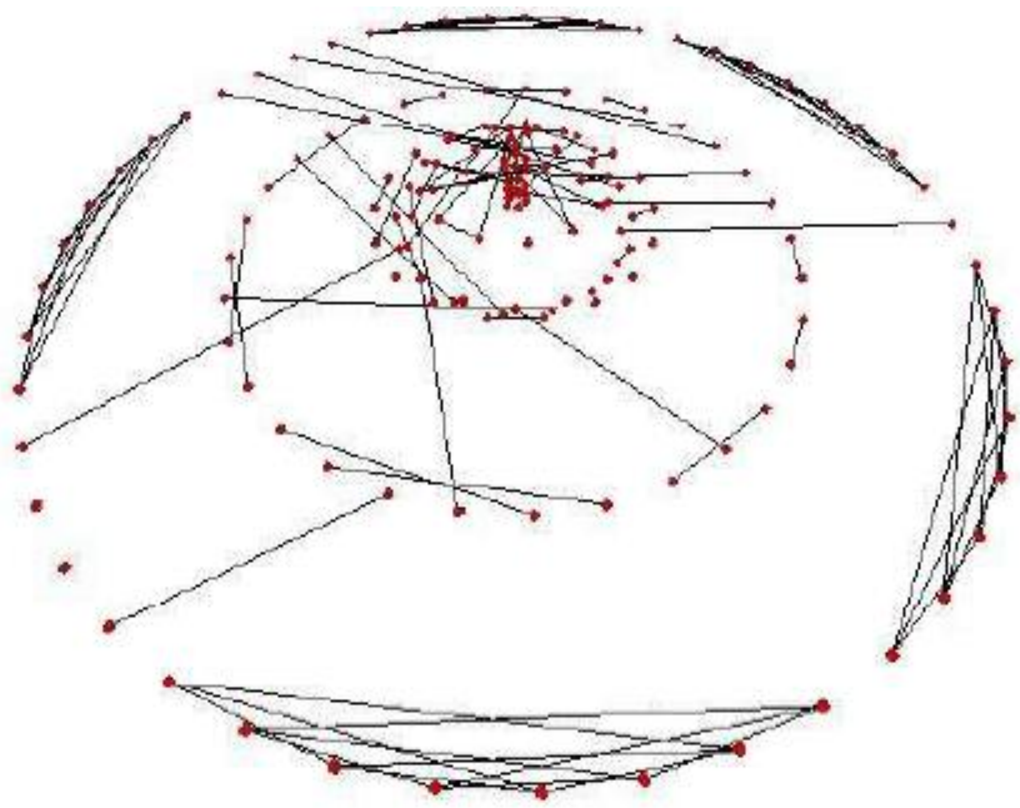


Logistics_b.cnf planning formula.
843 vars, 7,301 clauses, approx min backdoor 16
(backdoor set = reasoning shortcut)

Visualization by Anand Kapur.

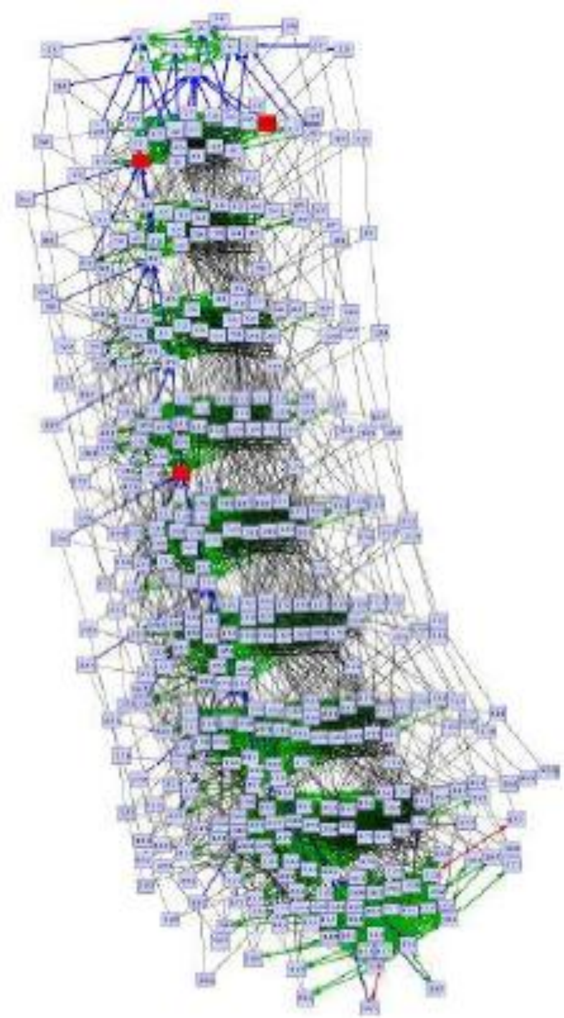


Logistics.b.cnf after setting 5 backdoor vars.

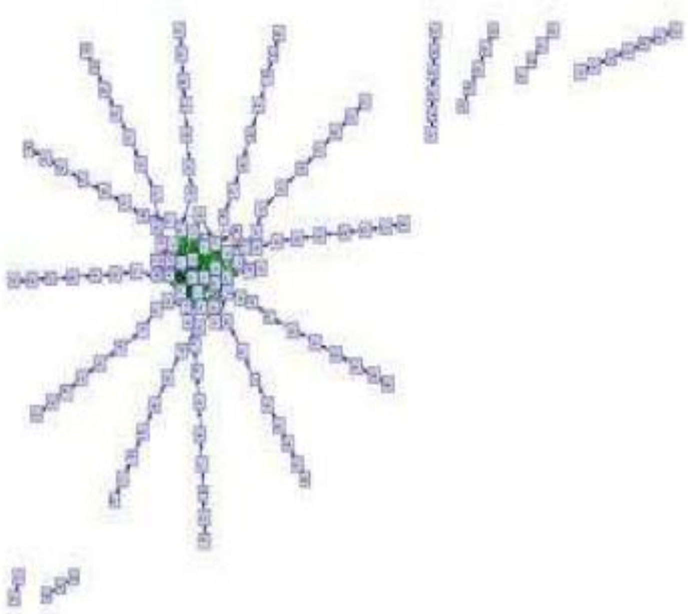


After setting just 12 (out of 800+) backdoor vars – problem almost solved.

Another example

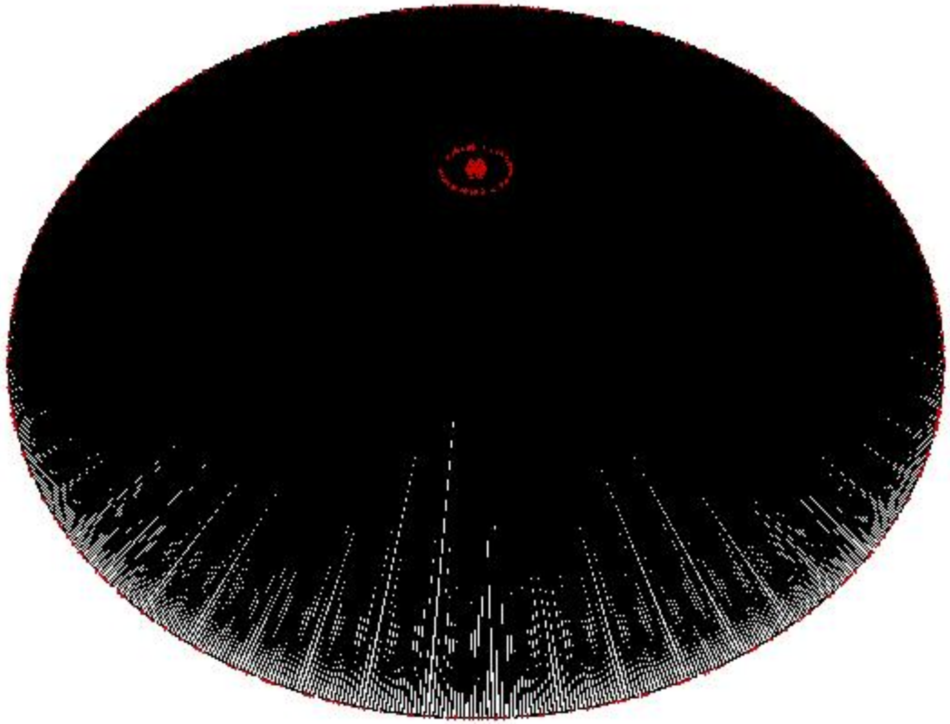


MAP-6-7.cnf infeasible planning instances. Strong backdoor of size 3.
392 vars, 2,578 clauses.

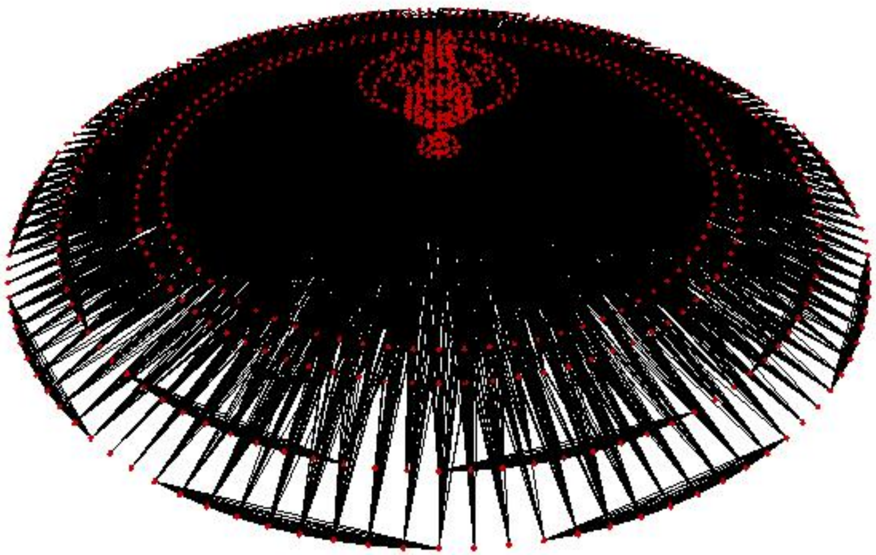


After setting 2 (out of 392) backdoor vars. ---
reducing problem complexity in just a few steps!

Last example.

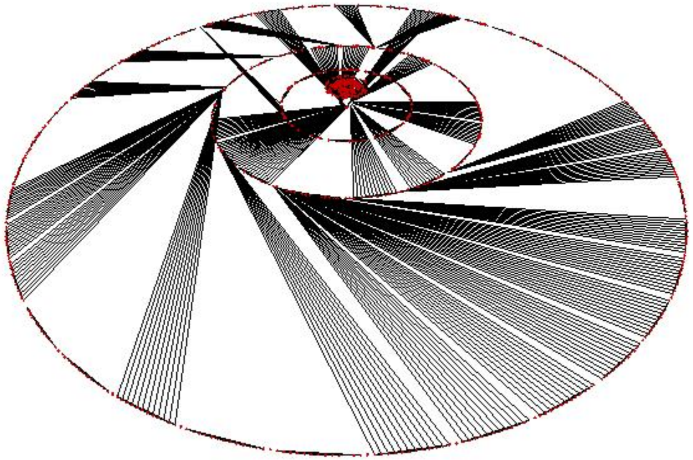


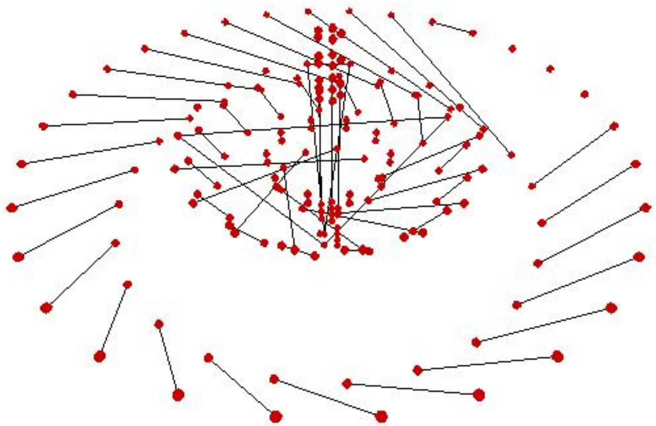
Inductive inference problem --- ii16a1.cnf. 1650 vars, 19,368 clauses.
Backdoor size 40.



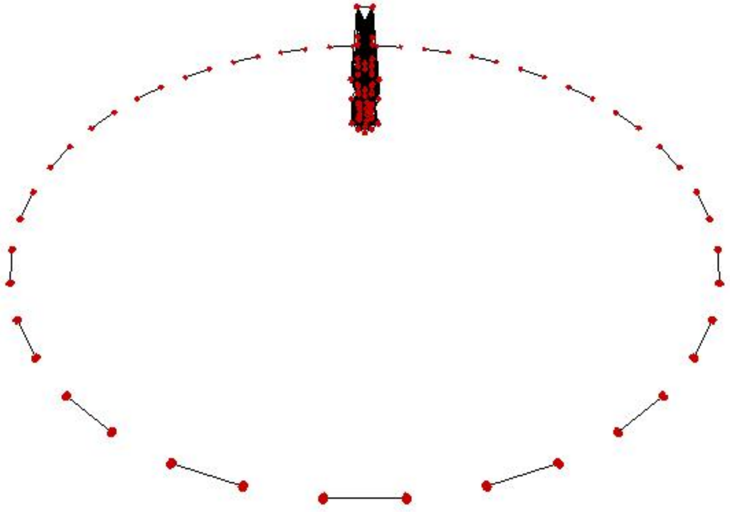
After setting 6 backdoor vars.

Some other intermediate stages:





After setting 38 (out of 1600+) backdoor vars:



So: Real-world structure *hidden* in the network. Can be exploited by automated reasoning engines.

But… we also need to take into account the cost of finding the backdoor!

We considered:

- **Generalized Iterative Deepening**
- **Randomized Generalized Iterative Deepening**
- **Variable and value selection heuristics**
  - **(as in current solvers)**

Size backdoor

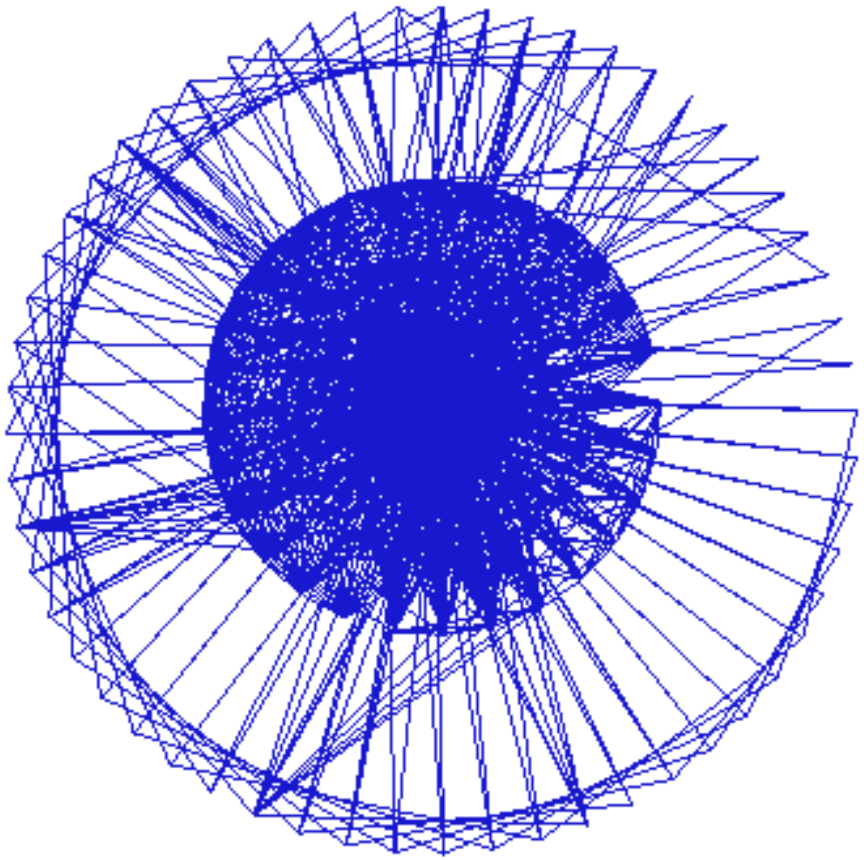
$B(n)$	deterministic	randomized	heuristic
n/k	small $exp(n)$	smaller $exp(n)$	tiny $exp(n)$
$O(\log n)$	$\left(\frac{n}{\sqrt{\log n}}\right)^{O(\log n)}$	$\left(\frac{n}{\log n}\right)^{O(\log n)}$	$poly(n)$
$O(1)$	$poly(n)$	$poly(n)$	$poly(n)$

n = num. vars.
k is a constant

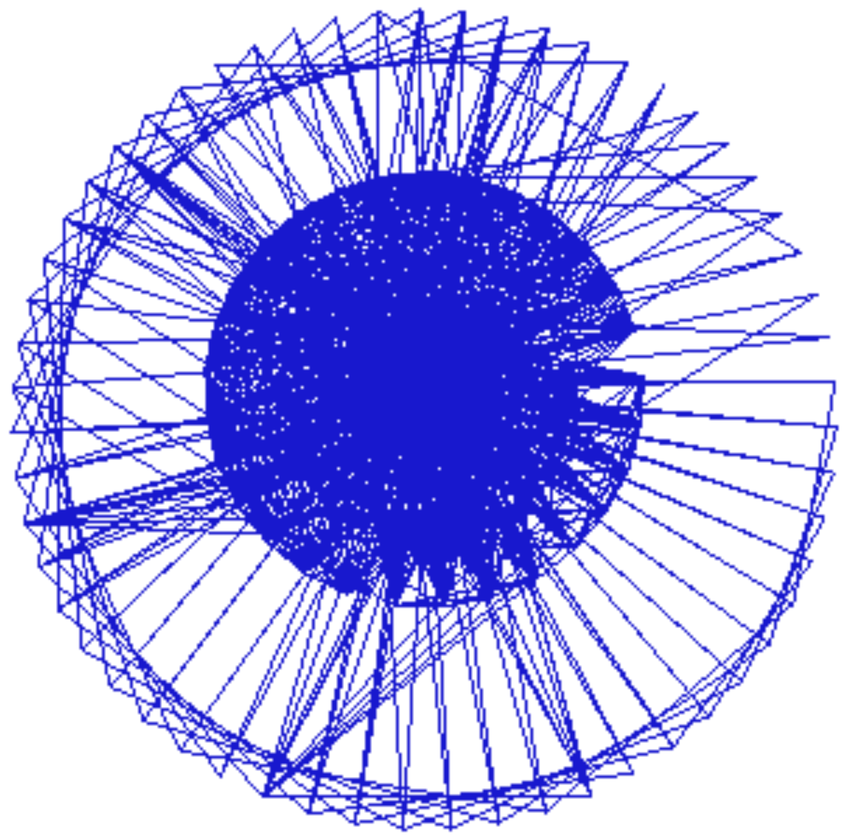
Current solvers

(Williams, Gomes, and Selman ’04)

Dynamic view: Running SAT solver
(no backdoor detection)



SAT solver detects backdoor set



A Journey from Random to Structured Instances

I --- Random Instances ✓
--- phase transitions and algorithms

II --- Capturing Problem Structure ✓
--- problem mixtures (tractable / intractable)
--- backdoor variables and heavy tails

III --- Beyond Satisfaction
--- sampling, counting, and probabilities
--- quantifiers

Part III) --- Beyond Satisfaction

Can we extend SAT/CSP techniques to solve harder counting/sampling problems?

Such an extension would lead us to a wide range of new applications.

SAT testing → counting/sampling

SAT testing	counting/sampling
↓	↓
logic inference	**probabilistic reasoning**
NP / co-NP-complete	*#P-complete*

Note: counting solutions and sampling solutions are computationally near equivalent.

Related work: Kautz et al. '04; Bacchus et al. '03; Darwich '04 & '05; Littman '03.

Standard Methods for Sampling: Markov Chain Monte Carlo (MCMC)

Based on setting up a Markov chain with a predefined stationary distribution.

E.g. simulated annealing.

Draw samples from the stationary distribution by running the Markov chain for a sufficiently long time.

Problem: for interesting problems, Markov chain takes exponential time to converge to its stationary distribution.

Bottom line: standard MCMC (e.g. SA) too slow.

First attempt

Use local search style algorithm:

Biased random walk = a random walk with greedy bias.

Example: WalkSat `(Selman *et al*, 1993)`**, effective on SAT.**

Can we use it to sample from solution space?

- Does WalkSat reach all solutions?
- How uniform/non-uniform is the sampling?

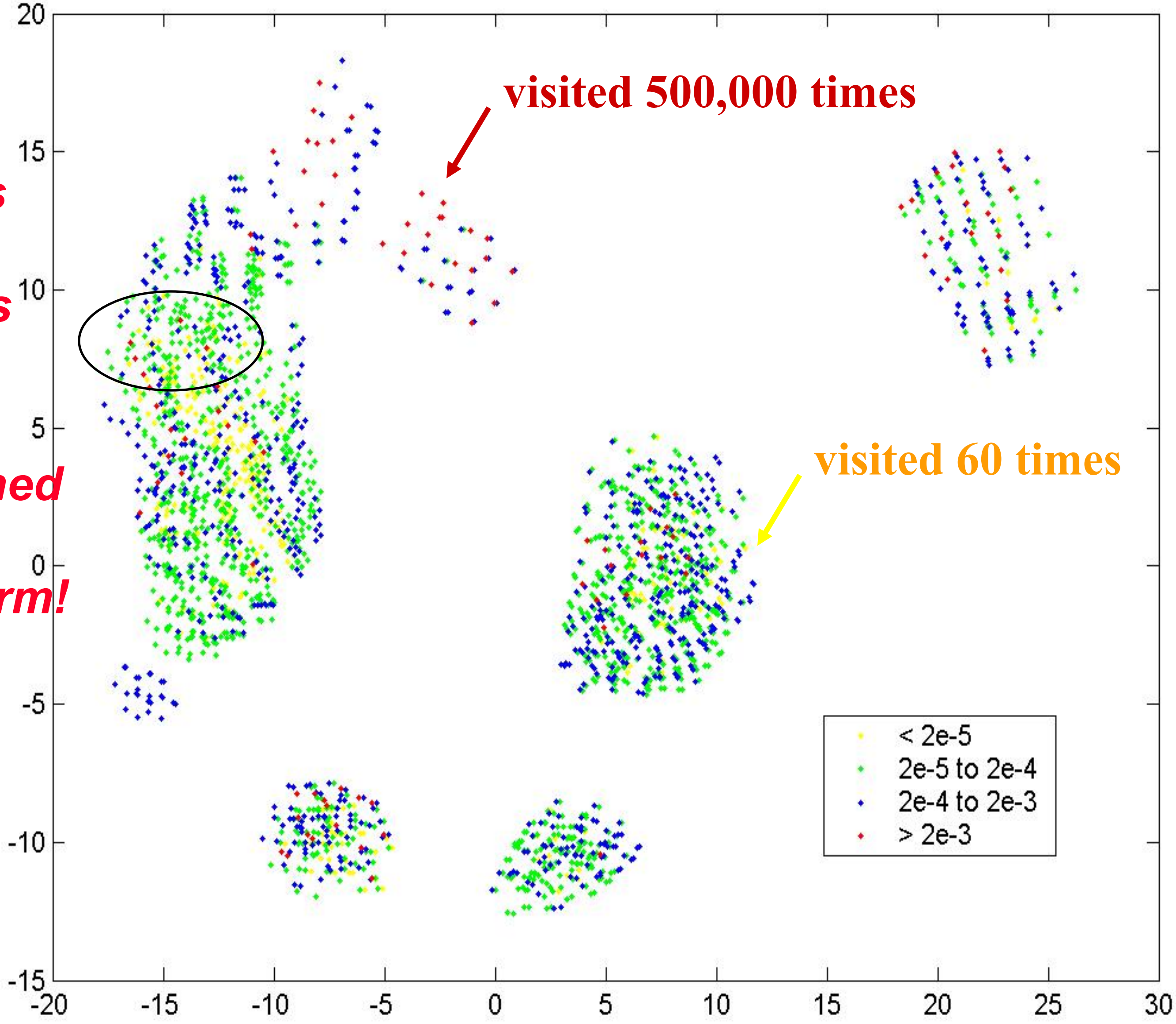
(Wei Wei and Selman ’04)

WalkSat

2,500 solutions

50,000,000 runs

All solns reached but quite nonuniform!



Hamming distance

Probability Ranges for Different Domains

Instance	Runs	Hits Rarest	Hits Common	Common-to - Rare Ratio
Random	50×10^6	53	9×10^5	1.7×10^4
Logistics	1×10^6	84	4×10^3	50
Verif.	1×10^6	45	318	7

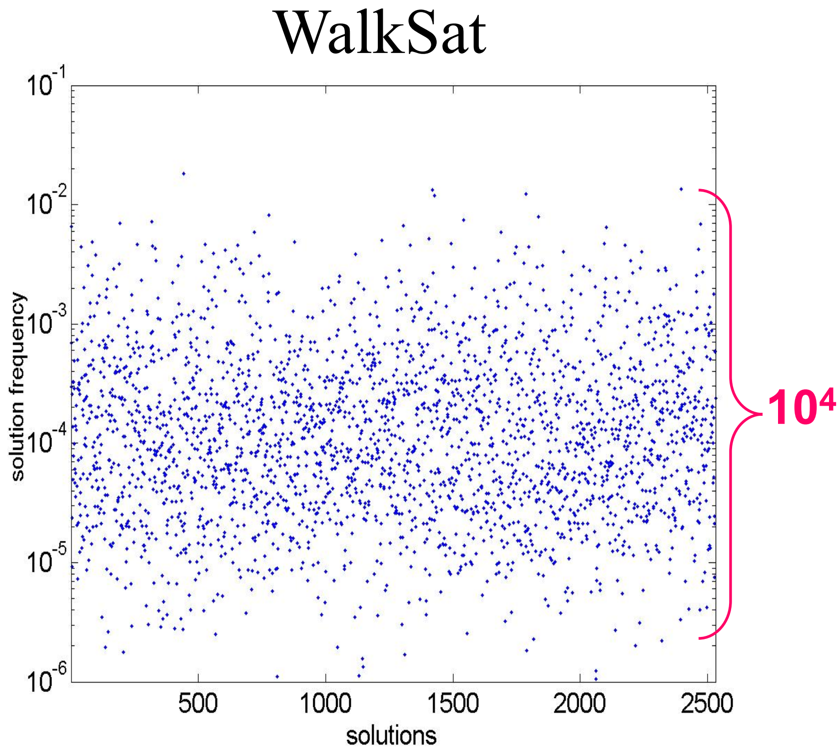
Improving the Uniformity of Sampling

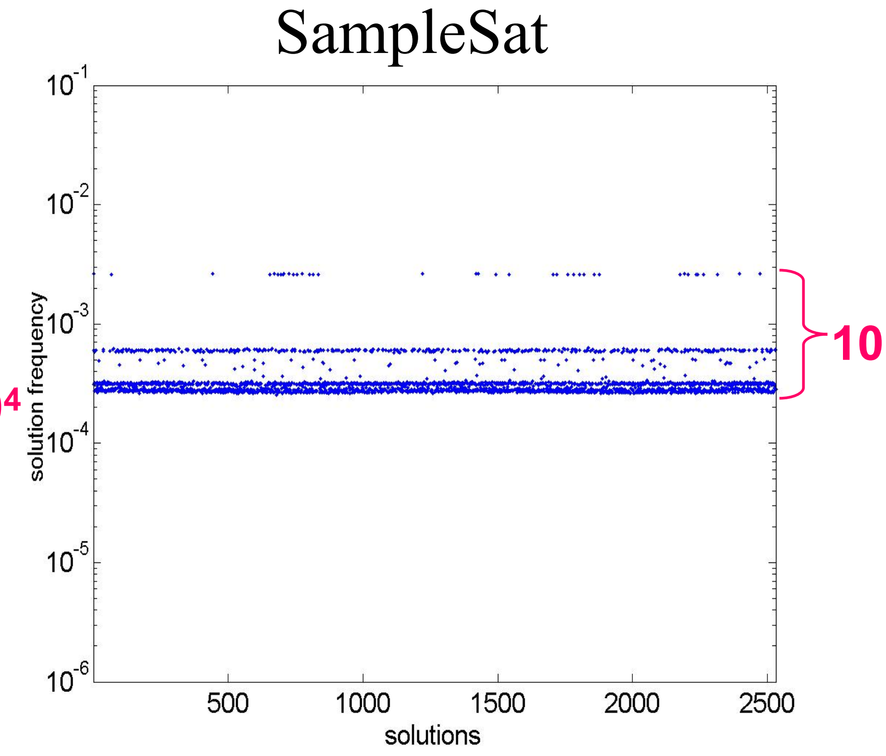
Nonergodic	Ergodic	Ergodic
Quickly reach sinks	Slow convergence	But does not satisfy DBC
↓	↓	↓
WalkSat	+ SA	= SampleSat

SampleSat:

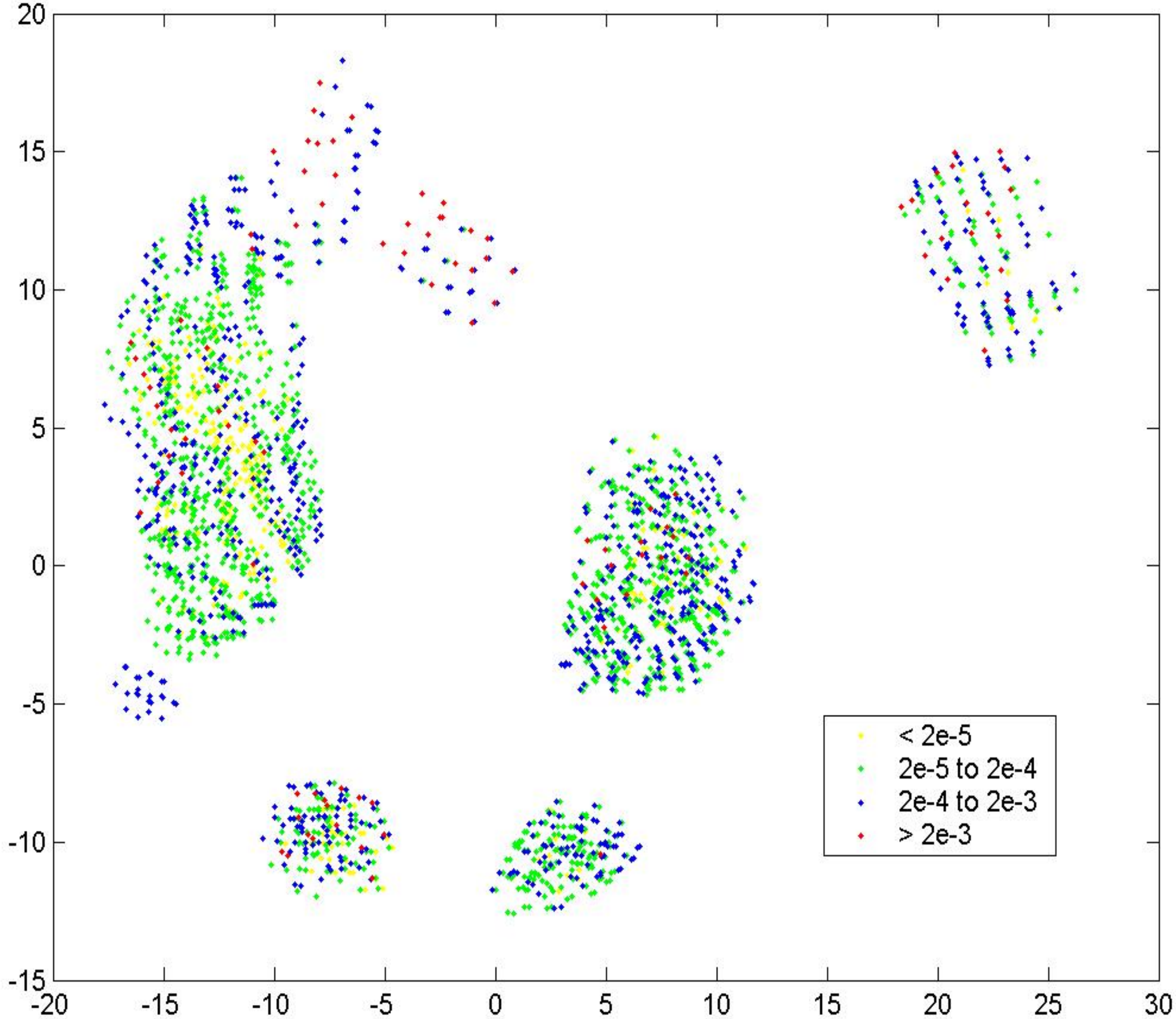
- **With probability *p*, the algorithm makes a biased random walk move**
- **With probability *1-p*, the algorithm makes a SA (simulated annealing) move**

Comparison Between WalkSat and SampleSat





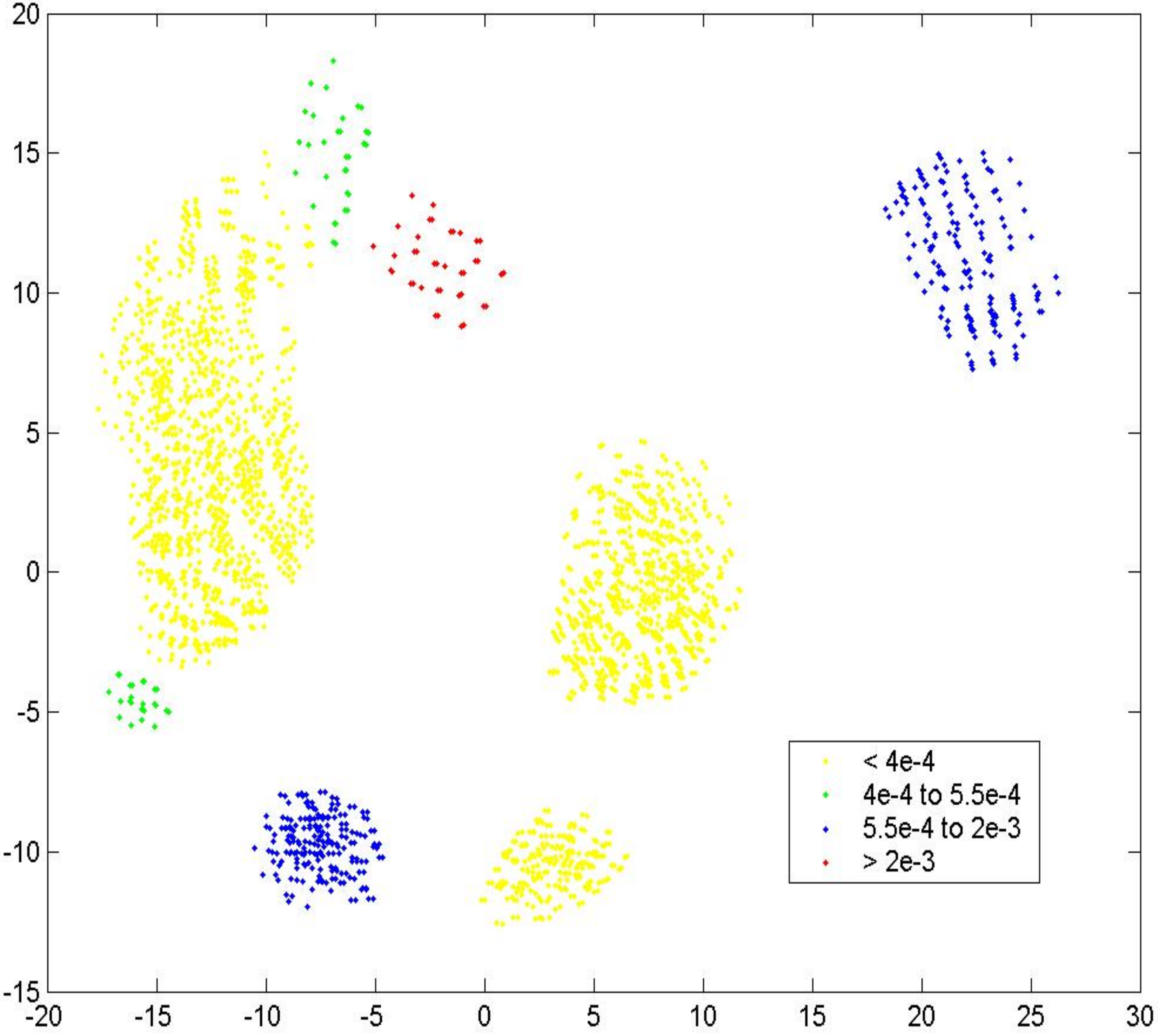
WalkSat



Hamming distance

SampleSAT

Note:
Uniform sampling within clusters.



Instance	Runs	Hits Rarest	Hits Common	Common-to - Rare Ratio WalkSat	Ratio SampleSat
Random	50×10^6	53	9×10^5	1.7×10^4	10
Logistics	1×10^6	84	4×10^3	50	17
Verif.	1×10^6	45	318	7	4

Formal results, see Wei Wei and Selman ('04).

Verification on Larger formulas - ApproxCount

Small formulas → Use solution frequencies.

How to verify on large formulas *(e.g. 10^25 solns)?*

A solution sampling procedure can be used to (approximately) count the number of satisfying assignments. (Jerrum and Valiant '86)

Comparison to exact counting (DPLL-style).

instance	#variables	Exact count	ApproxCount	Average Error / var
prob004-log-a	1790	2.6×10^{16}	1.4×10^{16}	0.03%
wff.3.200.810	200	3.6×10^{12}	3.0×10^{12}	0.09%
dp02s02.shuffled	319	1.5×10^{25}	1.2×10^{25}	0.07%

Beyond exact model counters

instance	#variables	#solutions	ApproxCount	Average Error / var
P(30,20)	600	7×10^{25}	7×10^{24}	0.4%
P(20,10)	200	7×10^{11}	2×10^{11}	0.6%

Summary: Counting & Sampling

Results show potential for modified SAT (CSP?) solvers (local search) for counting / sampling solutions.

Can handle solution spaces with 10^25 and more solutions.

Range of potential applications: e.g. many forms of probabilistic (Bayesian) reasoning.

Part III b) Quantified Reasoning

Quantified Boolean Formulas (QBF) extend Boolean logic by allowing quantification over variables (exists and forall)

Quantifiers prefix the clauses

$$\exists b_0 \ldots \forall b_{j+1} \ldots \forall b_h \exists b_{h+1} \ldots [(b_0 \vee \neg b_h) \wedge (\neg b_0 \vee \neg b_h)]$$

QBF is satisfiable iff

there exists a way of setting the existential vars such that for every possible assigment to the universal vars the clauses are satisfied.

Literally a "game played on the clauses":

Existential player tries hard to satisfy all clauses in the matrix.

Universal player tries hard to "spoil" it for the existential player: i.e., break ("unsatisfy") one or more clauses.

Formally: Problem is PSPACE- complete.

Range of new applications: Multi-agent reasoning, unbounded planning, unbounded model-checking (verification), and certain forms probabilistic reasoning and contingency planning.

Can we transfer successful SAT techniques to QBF?

Cautiously optimistic. But very sensitive to problem encodings.
(Antsotegui, Gomes, and Selman ’05)

Related work: Walsh ’03; Gent, Nightingale, and Stergiou ’05; Pan & Vardi 04; Giunchiglia *et al.* 04; Malik 04; and Williams ’05.

The Achilles' Heel of QBF

QBF is much more sensitive to problem encoding.

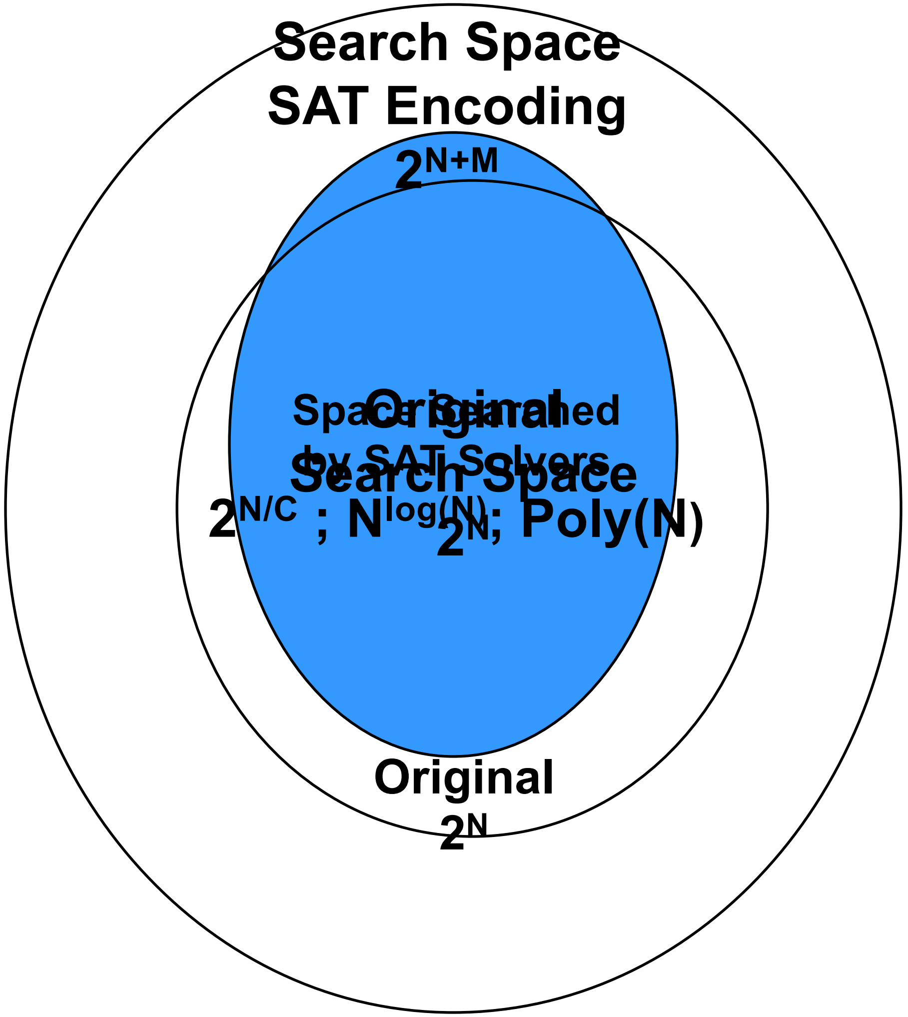
SAT/QBF encodings require auxiliary variables.
These variables significantly increase the raw combinatorial search space.

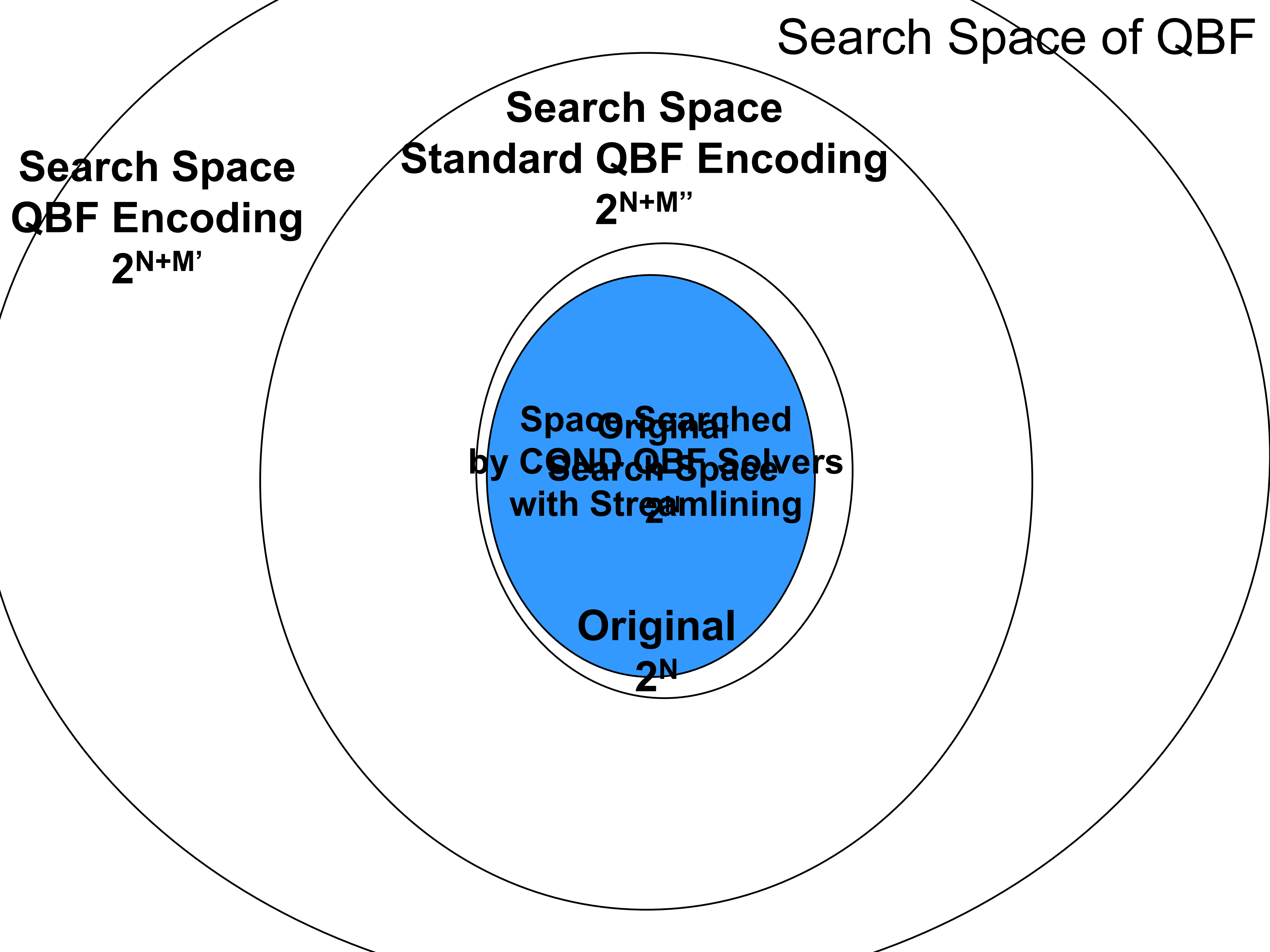
Not an issue for SAT: Propagation forces search to stay within combinatorial space of original task.

Not so for QBF! Universal player pushes to violate domain constraints (trying to violate one or more clauses). Search leads quickly outside of search space of original problems.

Unless, encodings are carefully engineered.

Search Space for SAT Approaches




Search Space of QBF
Search Space
QBF Encoding
2^{N+M'}
Search Space
Standard QBF Encoding
2^{N+M''}
Space Searched
by COND QBF Solvers
with Streamlining
Original
Search Space
2^N
Original
2^N

Summary

We journeyed from random to structured combinatorial reasoning problems.

Path from 100 var instances (early 90's) to 1,000,000 var instances (current).

Still moving forward!

Random instances:

--- linear time algs. approaching phase transition.

--- physics methods for computer science

Structure: --- mixture tractable / intractable (2+P-SAT)

--- backdoor sets, randomization, and restarts.

Beyond satisfaction: Potential for sampling, counting, and quantification.

Overall, significant progress in reasoning technology in last decade.

Research provides an active interplay between algorithm design, analysis, and experimenation; between computer scientists, physicists, and mathematicians.

Emerging Application Strategy: Automated reasoning tools as a true "cognitive assistant".

E.g., In hardware design (IBM), portfolio of reasoning engines running in parallel providing real-time feedback to hardware designers and testers

***The human design creativity is complemented with automated validation and feedback,* which enables the analysis of subtle interactions in large-scale complex artifacts.**

Realizing the dream of "automated reasoning".

Thanks to Carla!

The end. ☹

D) Lessons Learned

General theme:

2+p-SAT results & rapid restart strategies suggest that hidden tractable sub-structure in formulas can dramatically reduce overall complexity.

Current SAT solvers:

Carefully balance cost of search for hidden special structure (e.g. 2-SAT, Horn etc.) and cost of unrestricted search.

Strategies:

1) Discovering structure / Clause learning ---
   clauses are implied ("lemmas")
   challenge: find the right ones.
   a) Chaff solver: store lots of them (millions) and use clever indexing.
   b) Random walk procedures: store only long range dependencies.

2) Use randomization, restarts and heuristics to find "backdoor variables"

3) Learn new concepts (new variables) --- very challenging in general domains. (for SATPlan, Huang, Kautz, and Selman 1999)

1.) Example of adding derived dependencies

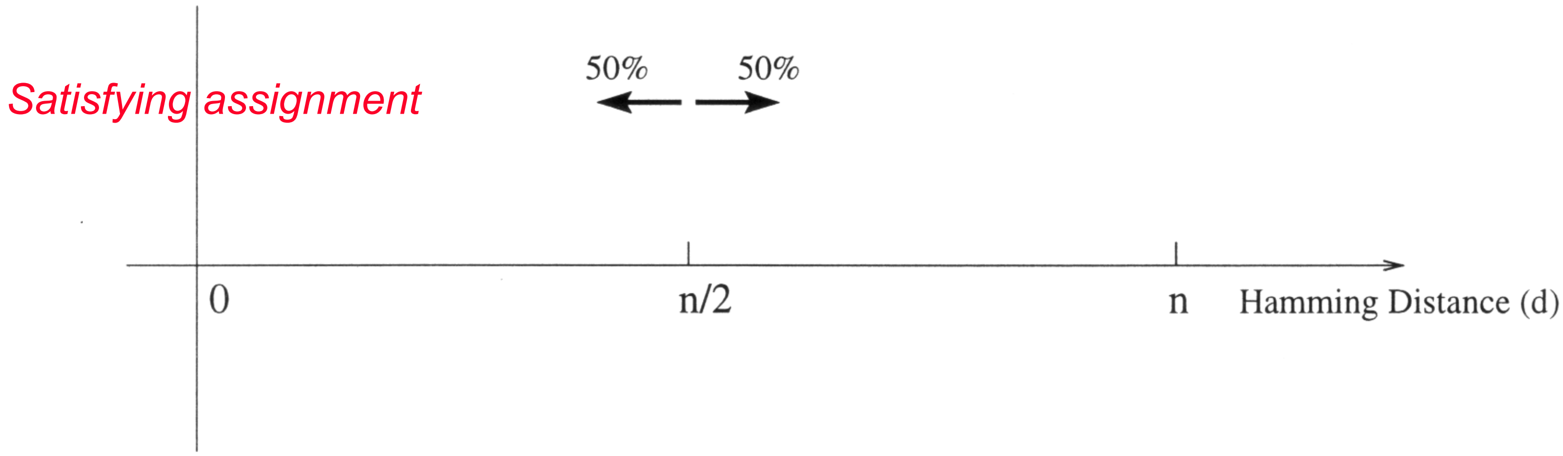
Random walk (RW) procedure:

1) Pick random truth assignment.

2) Repeat until all clauses are satisfied:
Flip random variable from unsatisfied clause.

Solves 2SAT in O(n^2) flips. (Papadimitriou 1992)

Why? Very elegant argument.



We have an unbiased random walk with a reflecting (max Hamming distance) and an absorbing barrier (satisfying assignment) at distance 0.

We start at a Hamming distance of approx. ½ N.

Property of unbiased random walks: after n^2 flips, with high probability, we will hit the origin (the satisfying assignment).

So, O(n^2) randomized algorithm (worst-case!) for 2-SAT.

Unfortunately, does not work for k-SAT with k>= 3. ☹

Still, Schoening (1999) shows that for 3-SAT, *Rapid Random Restarts* of RW of 3N steps, gives an improved exponential time algorithm. O(1.334^N) vs. O(2^n) for the obvious search strategy. Best known worst-case bound for 3-SAT (some recent improvements).

Can we make RW practical for SAT?

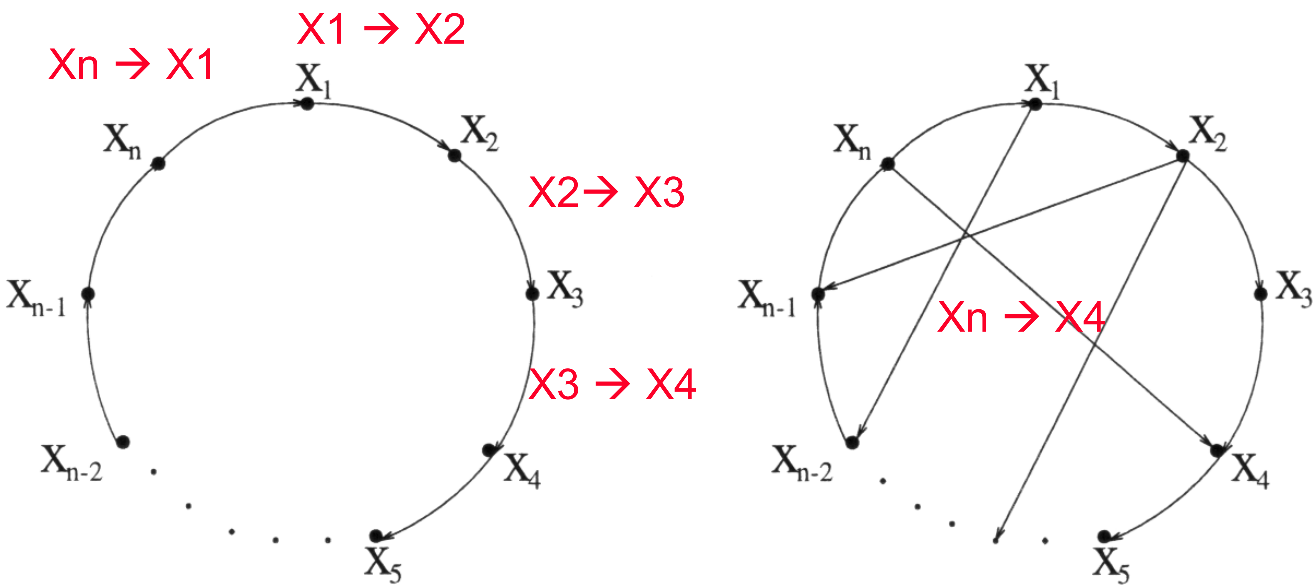
Yes. Use a *biased* random walk procedure

WalkSat Procedure (Selman, Kautz, and Cohen 1993)

Repeatedly select an unsatisfied clause:

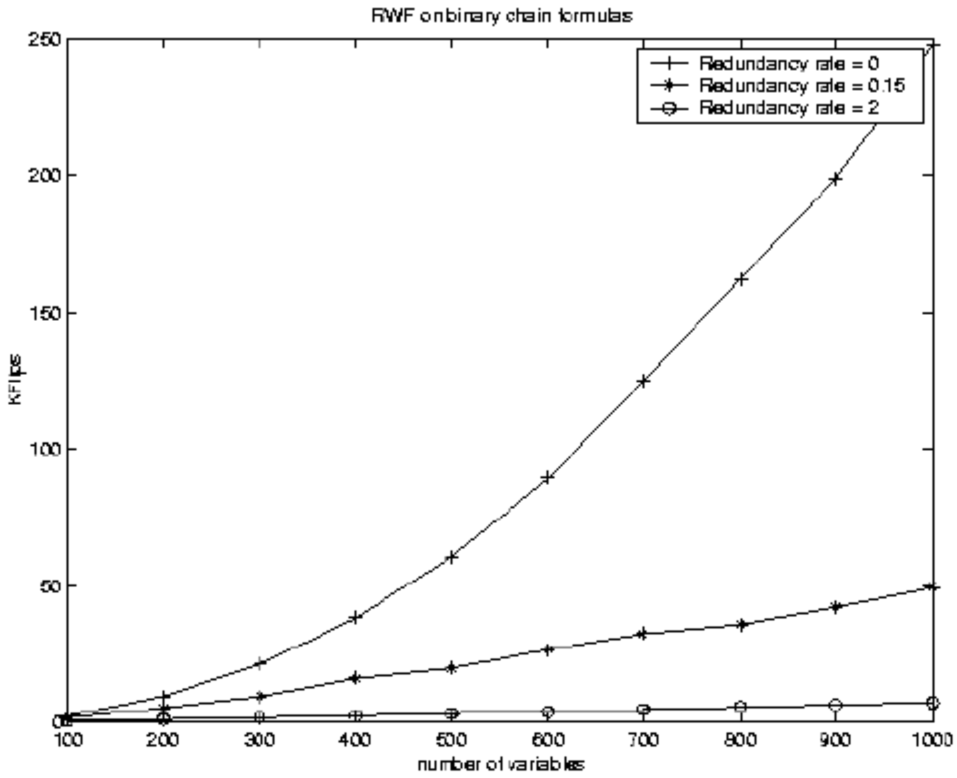
1) With probability p, flip a randomly selected variable in clause (i.e. the "usual").
2) With probability 1-p, flip greedily, i.e. flip variable in
   clause that yields greatest number
   of satisfied clauses (I.e., we introduce greedy bias).

First, bringing out the worst in random walks…
(2-SAT)



Theorem 1. *The RW procedure takes $\Theta(N^2)$ to find a satisfying assignment of F_{2chain}.*

Note: Only 2 satisfying assignments, all False and all True.



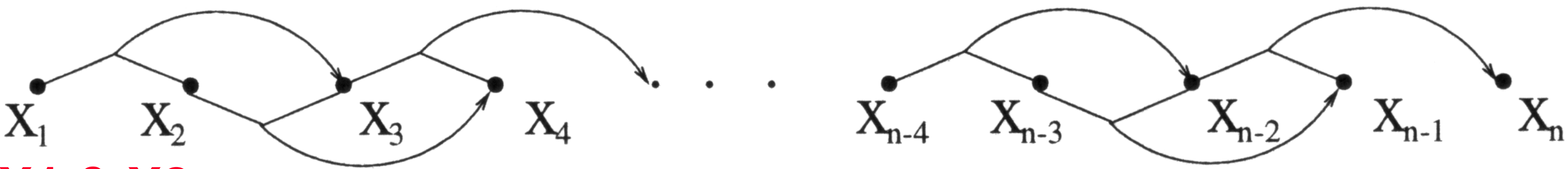
Adding redundant clauses / constraints, reduces run time of Walksat from N^2 down to N^1.1 (empirical).

Derived clauses capture long range dependencies.

What about 3-SAT? Again, consider "chain" formulas.

X1 & X2 → X3

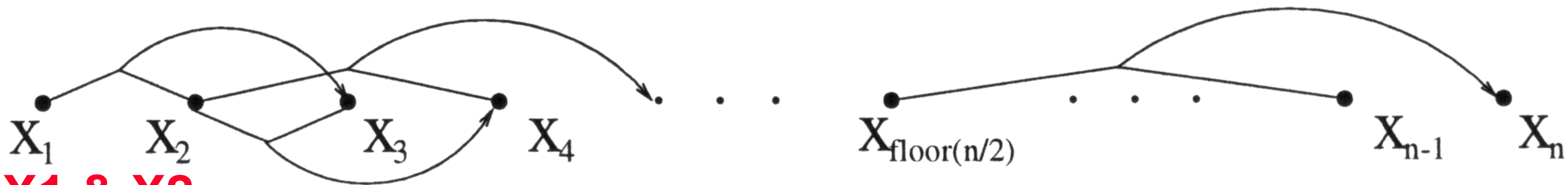
X_(n-2) & X_(n-1) → X_n



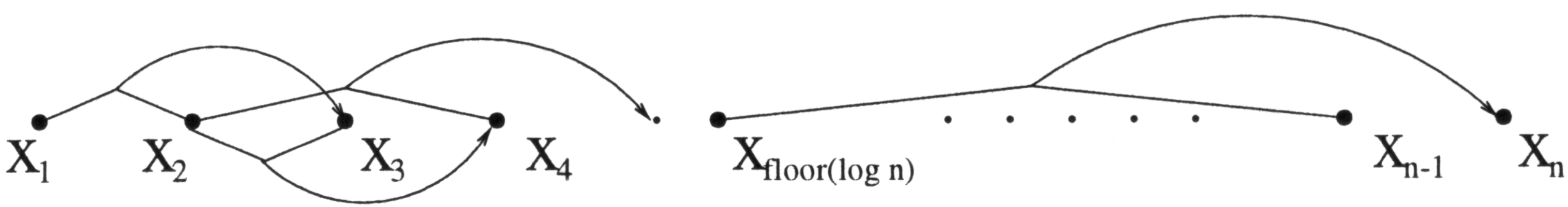
X1 & X2

X2 & X3 → X4

X_floor(n/2) & X_(n-1) → X_n



X1 & X2



Theorem 2. *Given a ternary chain formula $F_{3chain,i-2}$, starting at a random assignment s, the expected value of $f(s)$, the number of flips to reach a satisfying truth assignment, scales exponentially in n.*

Theorem 3. *Given a ternary chain formula $F_{3chain,low(i)}$ and let s be an arbitrary truth assignment, we have for the expected number of flips for the RW procedure:*

a) $\mathbf{E}(f(s)) = O(n \cdot n^{\log n})$, *for* $low(i) = \lfloor \frac{i}{2} \rfloor$

b) $\mathbf{E}(f(s)) = O(n^2 \cdot (\log n)^2)$, *for* $low(i) = \lfloor \log i \rfloor$.

Note: Thm. 2. Exponential behavior of RW. Not surprising perhaps.

But, Thm. 3, with longer range dependencies in formula, we get poly and quasi-poly behavior!

First, tractable class of 3-SAT problems for Random Walk.

Results suggest: Adding implied long range dependencies can significantly speed up RW and Walksat style procedures.

Formulas	< 40 sec	< 400 sec	< 4000 sec
$\alpha = 0.0$	15	26	42
$\alpha = 0.2$	**85**	**98**	**100**
$\alpha = 1.0$	13	33	64

Formulas from hardware verification benchmark

(Bryant and Velev 2001)

\alpha is redundancy rate

(Wei Wei and Selman '02)

Summary

During the past few years, we have obtained a much better understanding of the nature of computationally hard problems ("phase transitions")

Rich interactions between statistical physics, computer science and mathematics, and between theory, experiments, and applications.

Summary, cont.

Clear algorithmic progress (SAT solvers) ---

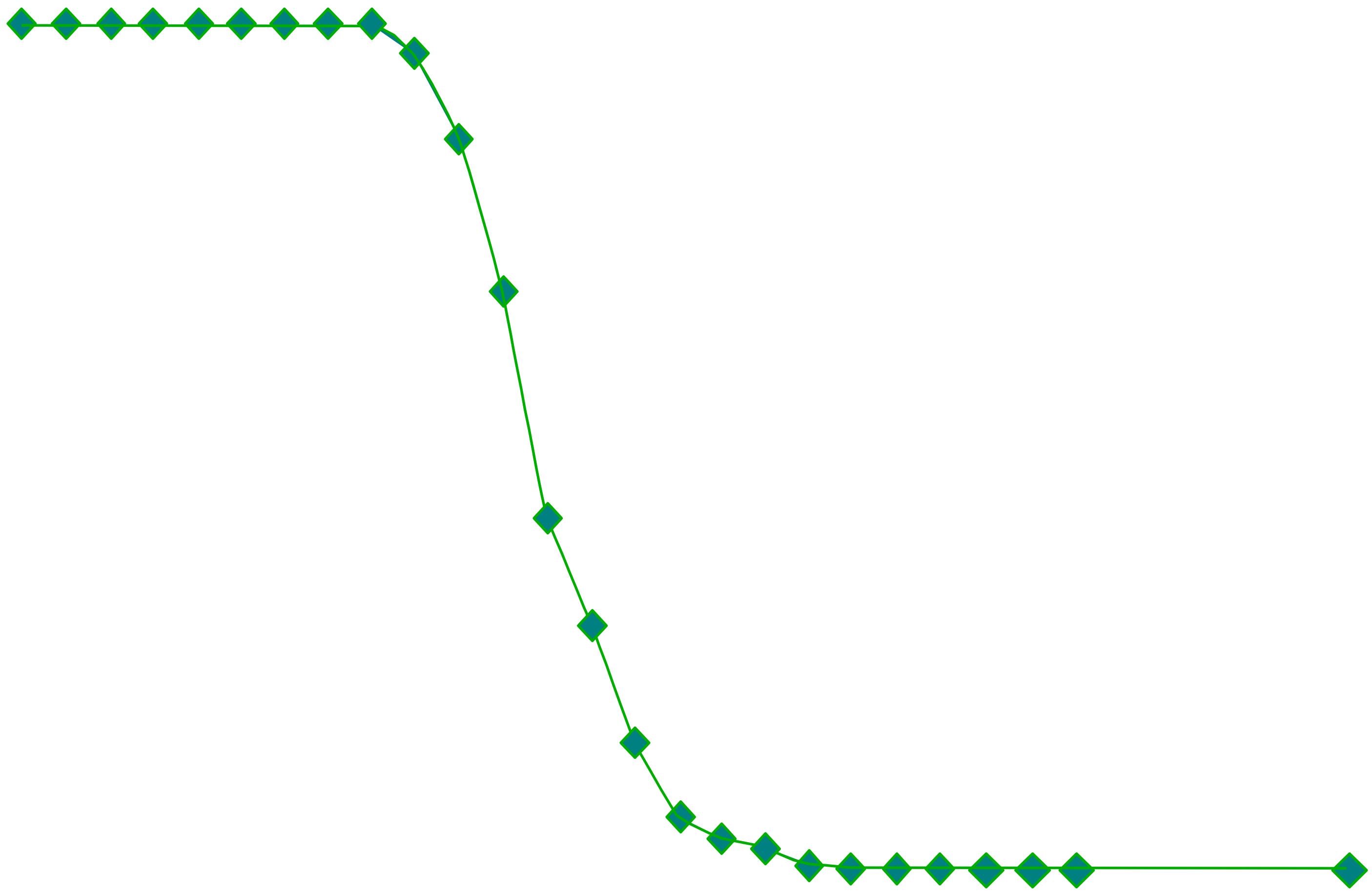
- 1 million vars & 5 million clauses
- Still discovering new applications!

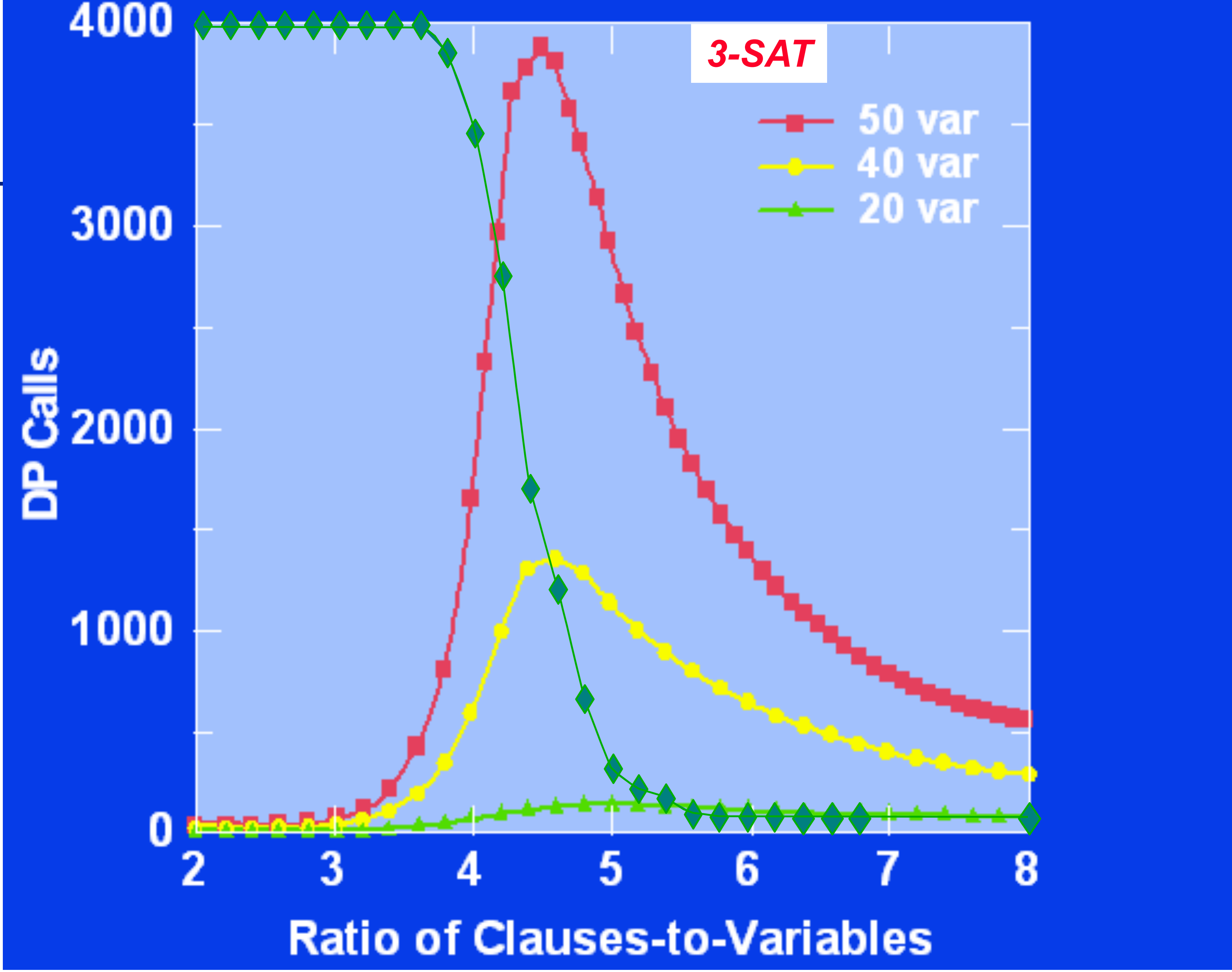
Strategies for exploiting hidden structure:

- restarts ("hunting for backdoors")
- adding long range dependencies (speeds up random walks)
- clause learning
- add new variables / concepts --- an open challenge

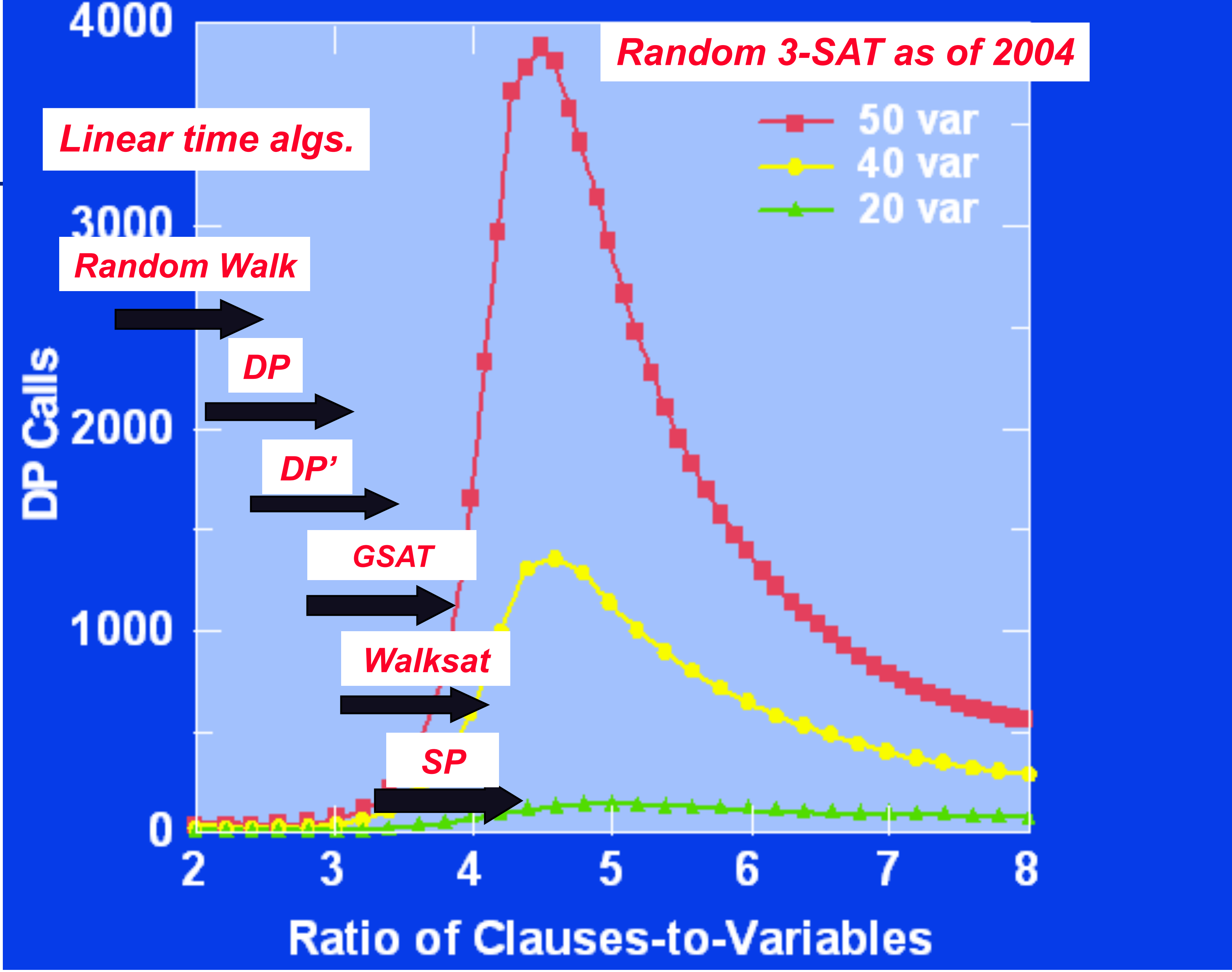
Other directions:

- model counting / sampling
- QBF
- Survey propagation / Belief propagation methods

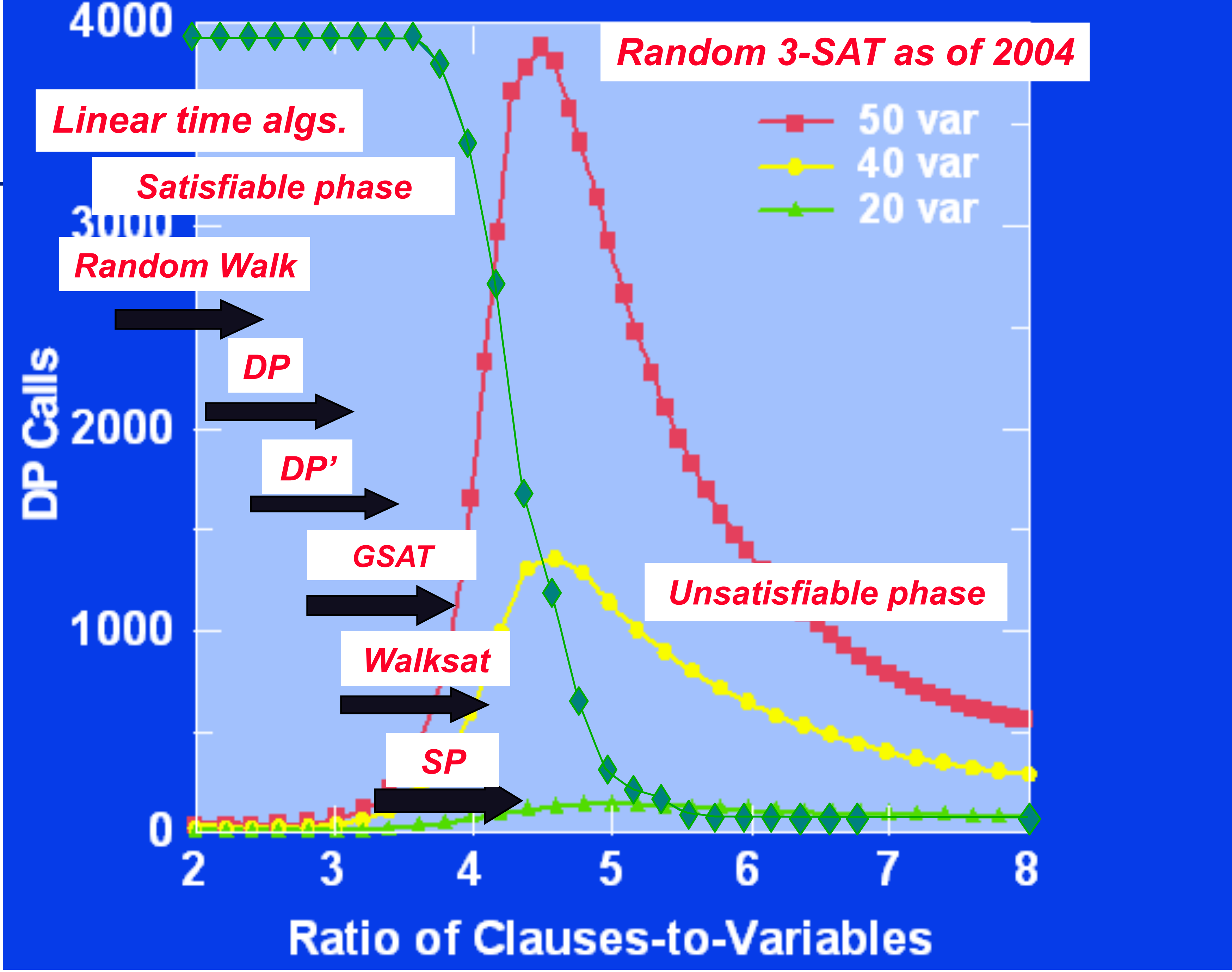




3-SAT
50 var
40 var
20 var
DP Calls
4000
3000
2000
1000
0
2
3
4
5
6
7
8
Ratio of Clauses-to-Variables



Random 3-SAT as of 2004
Linear time algs.
Random Walk
DP
DP'
GSAT
Walksat
SP
50 var
40 var
20 var
DP Calls
4000
3000
2000
1000
0
2
3
4
5
6
7
8
Ratio of Clauses-to-Variables


Random 3-SAT as of 2004
Linear time algs.
Satisfiable phase
Random Walk
DP
DP'
GSAT
Walksat
SP
Unsatisfiable phase
50 var
40 var
20 var
DP Calls
0
1000
2000
3000
4000
2
3
4
5
6
7
8
Ratio of Clauses-to-Variables

Refines (completes?) our understanding of combinatorial search space in random k-SAT.

In particular, p<= 0.4, solutions are clustered together at the bottom of the bowl-shaped energy landscape.

GSAT / zero temperature annealing can reach solutions easily (poly time).

Above p>0.4, critical (backbone) variables emerge. Solution space breaks up into small clusters (exponentially many) with diameter small compared to inter-cluster diameter.

Solution: use cavity field method from statistical physics. (Mezard et al. Science, 2002. Achlioptas et. al.; Gomes & Selman, Nature '05.)

Physics contributing to computation

80's --- Simulated annealing

General combinatorial search technique, inspired by physics.

(Kirkpatric, *Science* '83)

90's --- Phase transitions in computational systems

Discovery of physical laws and phenomena (e.g. 1st and 2nd order transitions) in computational systems.

(Kirkpatrick and Selman, *Science* '94; Monasson et al. *Nature* '99.)

'02 --- Survey Propagation

Analytical tool from statistical physics leads to powerful algorithmic method. (Mezard *et al.*, *Science* '02).

More expected!

Explaining short runs: Backdoors to tractability

Informally:

A backdoor to a given problem is a subset of the variables such that once they are assigned values, the polynomial propagation mechanism of the SAT solver solves the remaining formula.

Formal definition includes the notion of a “subsolver”:

a polynomial simplification procedure with certain general characteristics found in current DPLL SAT solvers.

Backdoors correspond to “clever reasoning shorcuts” in the search space.

Note: Notion of backdoor is related to but different from constraint-graph based notions such as cutsets.
(Dechter 1990; 2000)

Decay of Distributions

Standard --- Exponential Decay

e.g. Normal:

$$\Pr[X > x] \approx Ce^{-x^2},\ \ for\ some\ C > 0,\ x > 1$$

Heavy-Tailed --- Power Law Decay

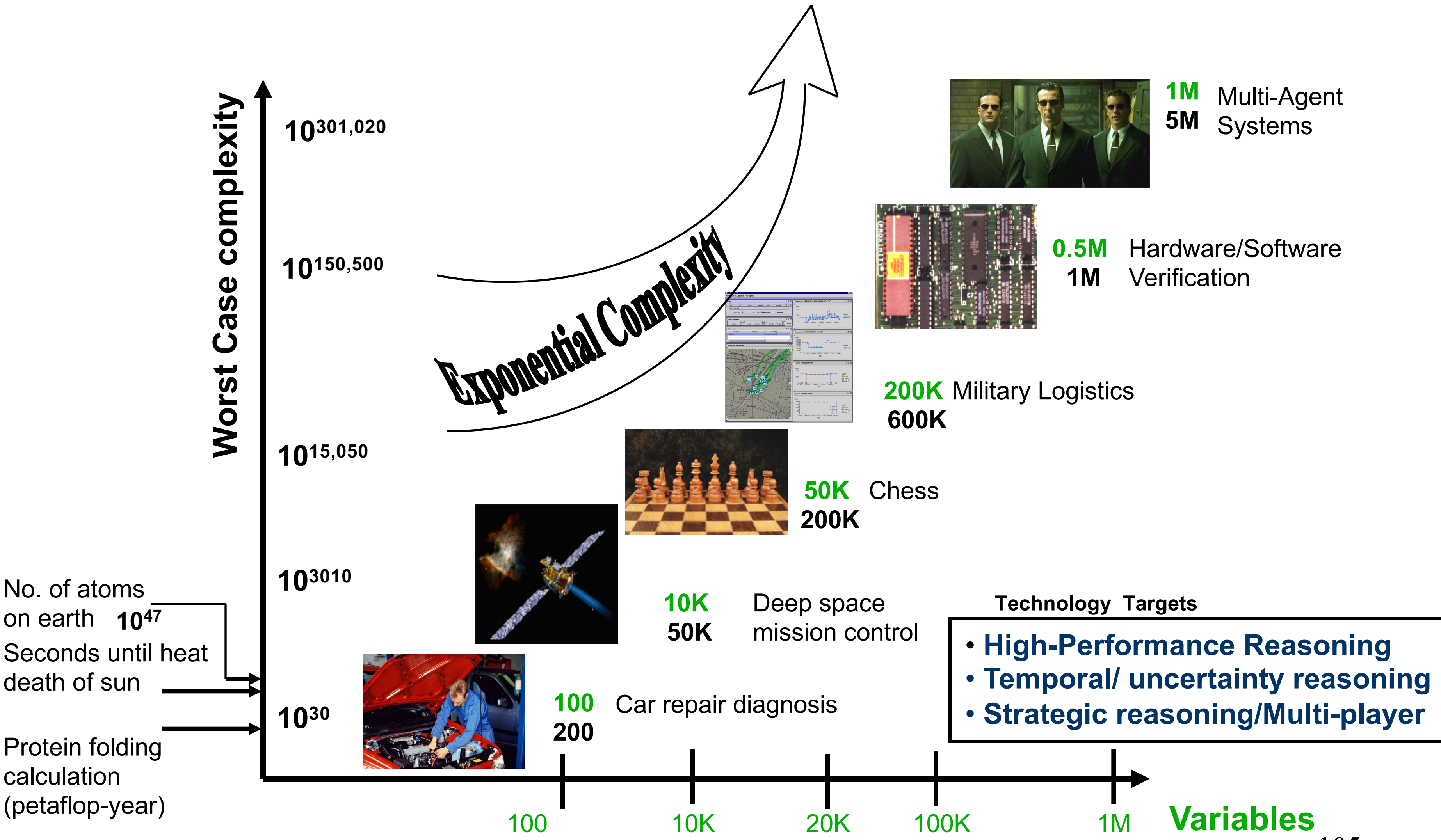
e.g. Pareto-Levy:

$$\Pr[X > x] = Cx^{-\alpha},\ \ x > 0$$

Real-World Reasoning

Tackling inherent computational complexity

DARPA Research Program



Example domains cast in propositional reasoning system (variables, rules).