Beyond Satisfiability: Model Counting, Quantification, and Randomization

Carla P. Gomes
Cornell University

Connections II
Caltech
2006

Satisfiability and Beyond

- SAT
- MAXSAT, SMT
- #SAT
- QBF

Motivation: Significant progress in SAT

From 100 variables, 200 constraints (early 90's)
to 1,000,000 vars. and 5,000,000 clauses in 15 years.

Applications:

Hardware and Software Verification, Planning,
Scheduling, Optimal Control, Protocol Design,
Routing, Multi-agent systems, E-Commerce (E-auctions and electronic trading agents), etc.

Motivation: Defying NP-Completeness

Current state of the art complete or exact solvers (SAT/CSP/MIP) can handle very large problem instances of real-world combinatorial:

→We are dealing with formidable search spaces of exponential size --- to prove unsatisifability or optimality we have to implicitly search the entire search,

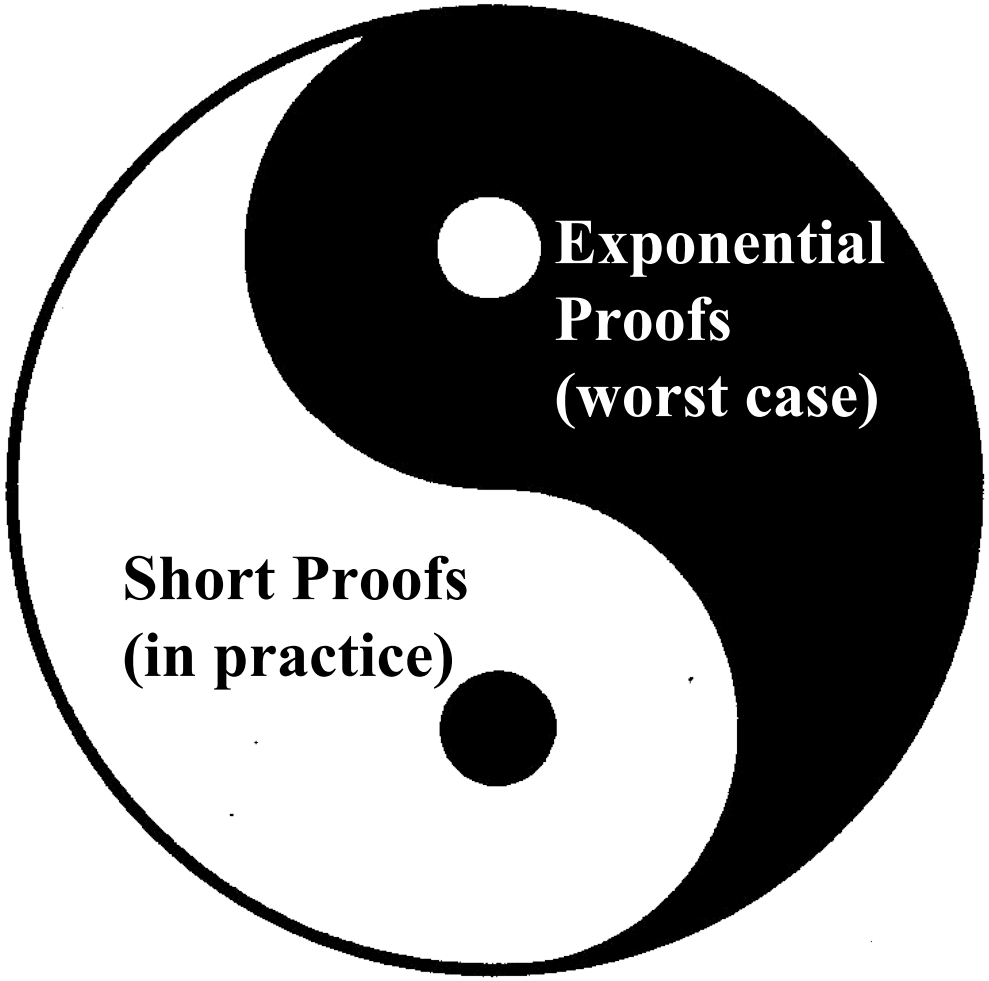
→the problems we are able to solve are much larger than one would predict given that such problems are in general NP complete or harder often

While real-world instances with over 1,000,000 variables are often solved in a few minutes, random SAT instances with only a few hundred variables often cannot be solved!

Intelligent Information Systems Institute

CORNELL UNIVERSITY

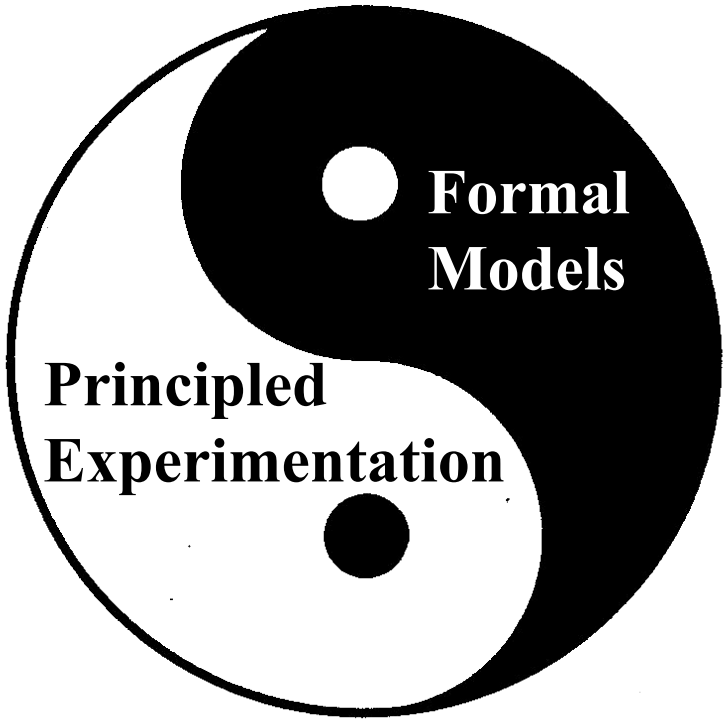
Gap Between Theory and Practice



Intelligent Information Systems Institute

CORNELL UNIVERSITY

Understanding the Gap Between Theory and Practice



Outline

- SAT
  - Random Problems
  - Structured Problems
  - Connections between Heavy-tailed Distributions, Backdoors, and Restart Strategies in Complete Search Methods for Combinatorial Problems:
- #SAT
  - Streamlining Constraint Reasoning, Randomization, and Model Counting;
- QBF - Quantification
- Conclusions


IISI
Intelligent Information
Systems Institute


CORNELL
UNIVERSITY

SAT

Intelligent Information Systems Institute

CORNELL UNIVERSITY

Propositional Satisfiability problem: (SAT)

Satifiability (SAT): Given a formula in propositional calculus, does it have a model, i.e., is there an assignment to its variables making it true?

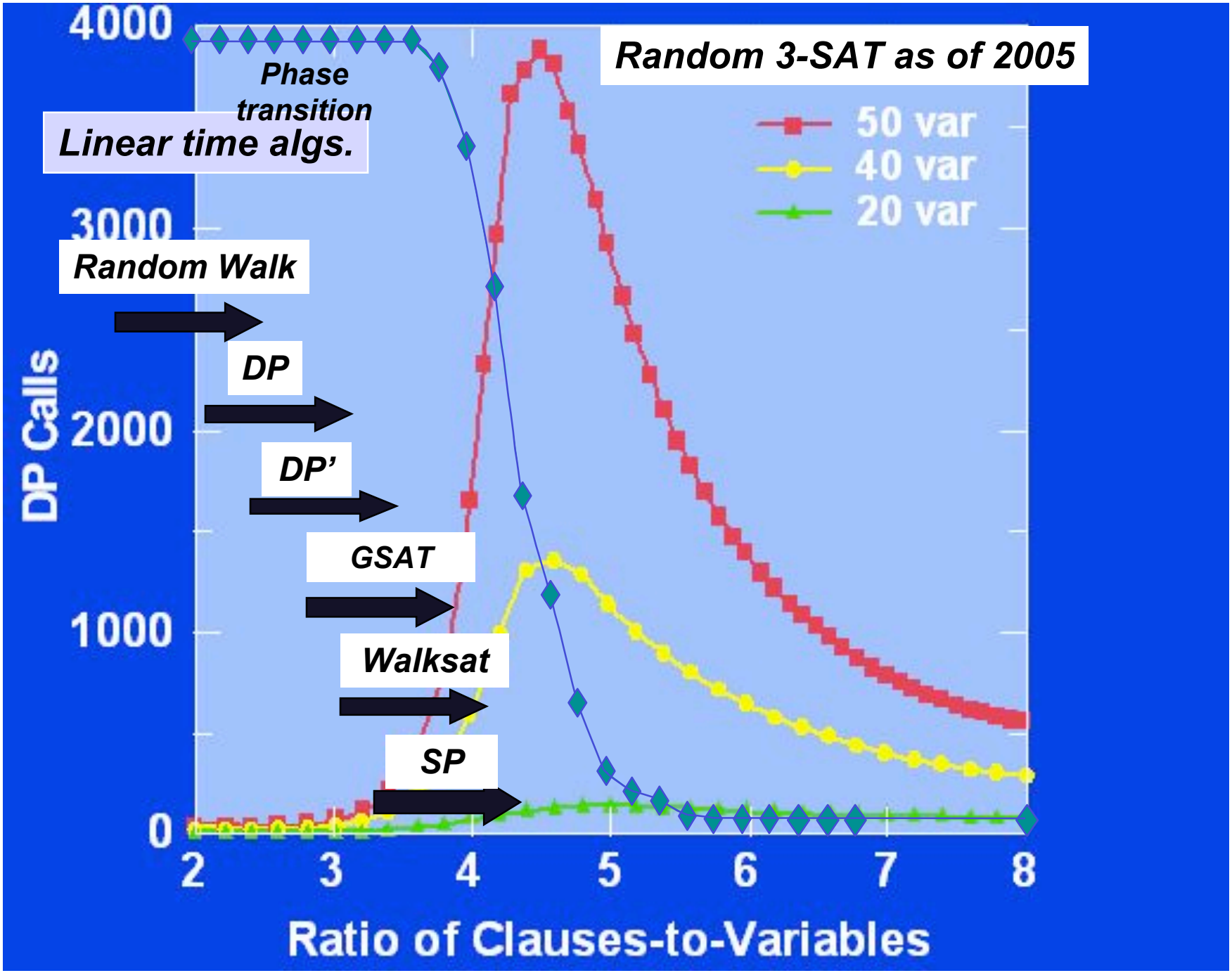
$$(a \vee \neg b \vee \neg c) \text{ AND } (b \vee \neg c) \text{ AND } (a \vee c)$$

possible assignments

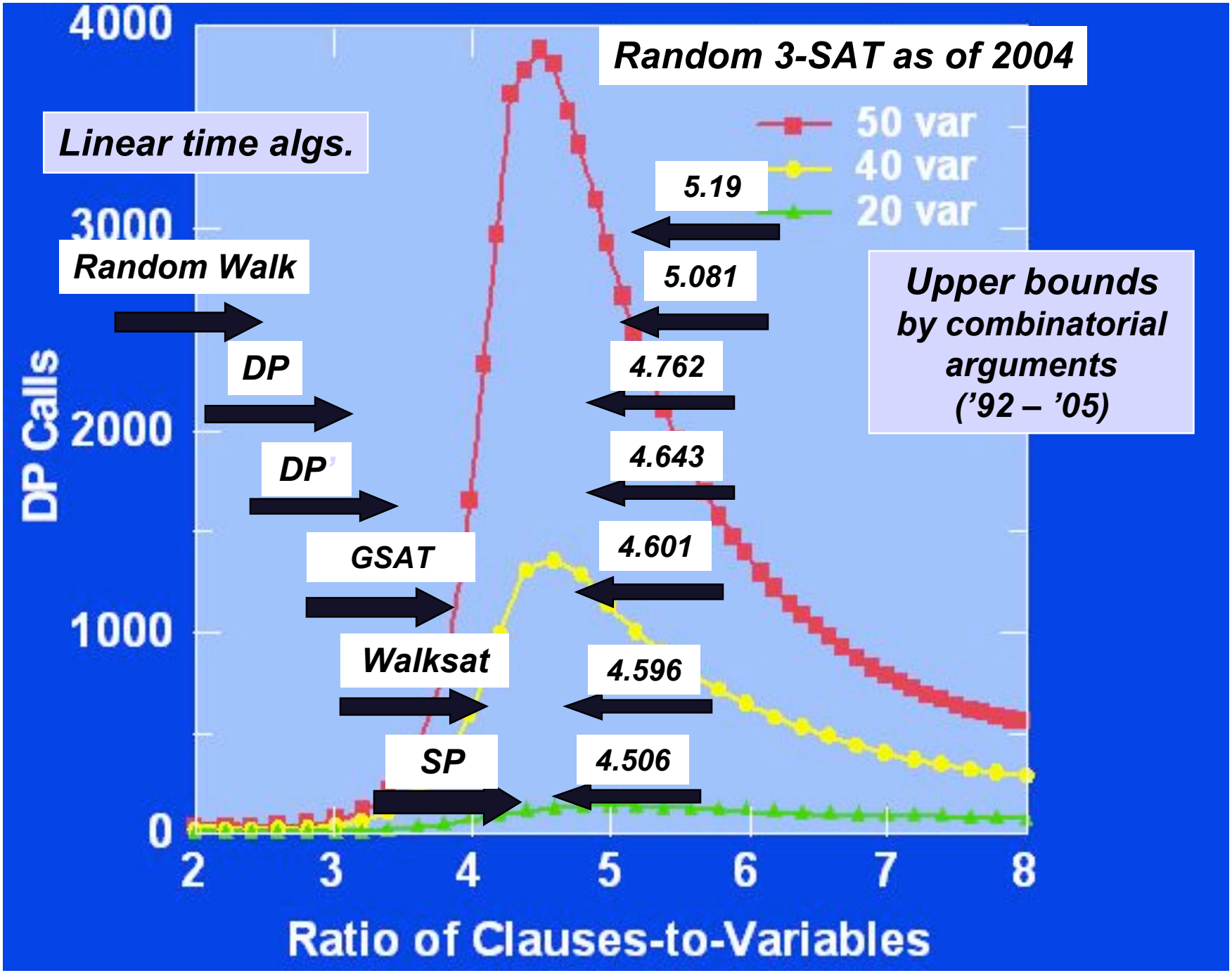
SAT: prototypical hard combinatorial search and reasoning problem. Problem is NP-Complete. (Cook 1971)

SAT

Random Instances



Mitchell, Selman, and Levesque '92


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

Tremendous interaction with other communities
OR, Physics, Mathematics

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.

SAT
Structured Problems

Surprising "power" of SAT for solving certain real world combinatorial problems (clearly outperforming Integer Programming).

From academically interesting to practically relevant.

We now have regular SAT solver competitions.
Germany ’89, Dimacs ’93, China ’96, SAT-02, SAT-03, …, SAT-06

Sat 06 – Seattle Aug. 12-15, 2006
SAT Competitions:

- Classical SAT solvers (CNF)
- Pseudo Boolean Solvers
- QBF
- MAXSAT
- SMT – SAT Mod Theory

SAT Competition 2006: Industrial Instances

- Received 29 solvers by 23 submitters from 13 nations

Australia	1 / 1
Austria	4 / 1
Canada	3 / 3
France	3 / 3
Germany	3 / 2
Israel	1 / 1
Japan	1 / 1

Northern Ireland	1 / 1
Portugal	1 / 1
Spain	1 / 1
Sweden	1 / 1
Netherlands	2 / 1
USA	7 / 6

(X / Y: X solvers, Y submitters)

Europe: 16 solvers, North America: 10, Asia/Australia: 2, Middle East: 1

- 3 industrial solvers, 25 academic, 1 private/amateur

SAT Competition 2006: Industrial Track

- Only industrial category benchmarks
  (no handcrafted and random)
- Short run-times
  (15 minutes timeout per instance)
- Mixture of satisfiable / unsatisfiable instances
  (thus not suitable for local-search solvers)

Intelligent Information Systems Institute

CORNELL UNIVERSITY

SAT Competition 2006: Benchmark Instances Classical SAT Solvers

- 20 instances from bounded model checking
  - IBM's benchmark 2002 and 2004 suites
- 40 instances from pipelined machine verification
  - 20 instances from Velev's benchmark suite
  - 20 instances from Manolios' benchmark suite
- 10 instances from cryptanalysis
  - Collision-finding attacks on reduced-round MD5 and SHA0 (Mironov & Zhang)
- 30 instances from former SAT-Competitions (industrial category)

Instances up to 1,000,000 variables and 15,000,000 clauses – at least one solver could prove SAT/UNSAT

Timelimit per instance: 15 minutes

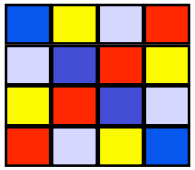
Progress SAT Solvers

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

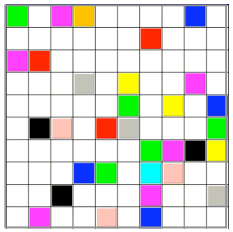
Source: Marques Silva 2002

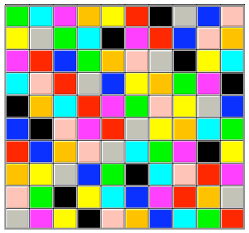
An abstraction of a structured combinatorial problems: Encodings and hardness profiles

Latin Square Completion

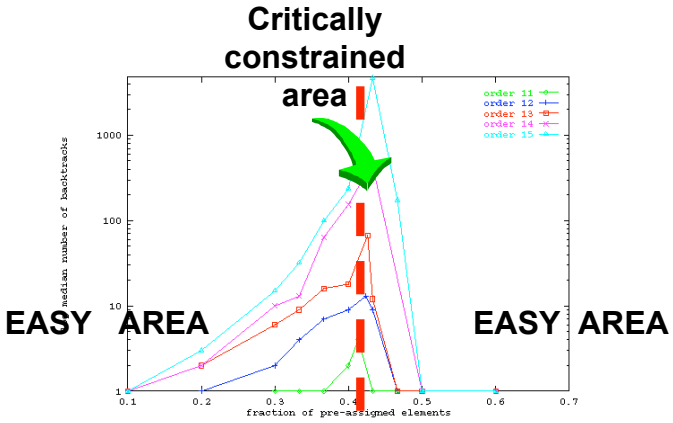


Latin Square
(Order 4)





NP-Complete



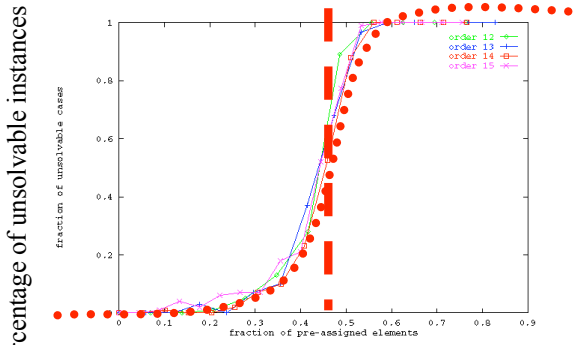
Complexity of Latin Square Completion

Better characterization

beyond worst case?

Time: 150 1820 165

35% 42% 50%



Phase Transition

20% 42% 50%

Gomes and Selman 97

Underlying Latin Square structure characterizes many real world applications

teams

teams	.1	2	3	4	5
•1	X	•R1	•R4	•R2	•R5	•R3
•2	•R1	X	•R2	•R5	•R3	•R4
•3	•R4	•R2	X	•R3	•R1	•R5
•4	•R2	•R5	•R3	X	•R4	•R1
•5	•R5	•R3	•R1	•R4	X	•R2
•6	•R3	•R4	•R5	•R1	•R2	X

A	D	E	B	C
C	B	A	E	D
D	C	B	A	E
E	A	C	D	B
B	E	D	C	A

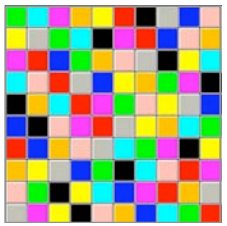
Design of Scientific Experiments

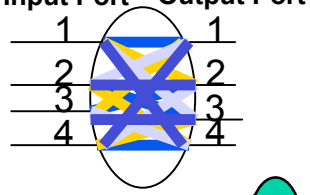
Scheduling and timetabling

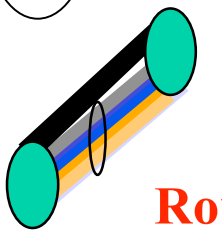
		6						1
	7			6			5	
8			1		3	2		
		5		4		8		
	4		7		2		9	
		8		1		7		
		1	2		5			3
	6			7			8	
2						4		

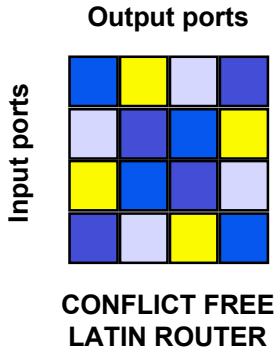
5	3	6	8	2	7	9	4	1
1	7	2	9	6	4	3	5	8
8	9	4	1	5	3	2	6	7
7	1	5	3	4	9	8	2	6
6	4	3	7	8	2	1	9	5
9	2	8	5	1	6	7	3	4
4	8	1	2	9	5	6	7	3
3	6	9	4	7	1	5	8	2
2	5	7	6	3	8	4	1	9

Sudoku









CONFLICT FREE
LATIN ROUTER

Routing in Fiber Optic Networks

Many more applications…

Intelligent Information Systems Institute

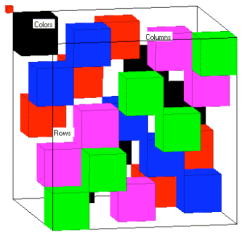
CORNELL UNIVERSITY

Encodings

- Constraint Satisfaction
- Integer Programming
- SAT

1. All the encodings exhibit similar qualitative behavior wrt to hardness profile
2. Scaling varies with encoding;

Integer Programming (Assignment Formulation)



Max number of colored cells

Variables

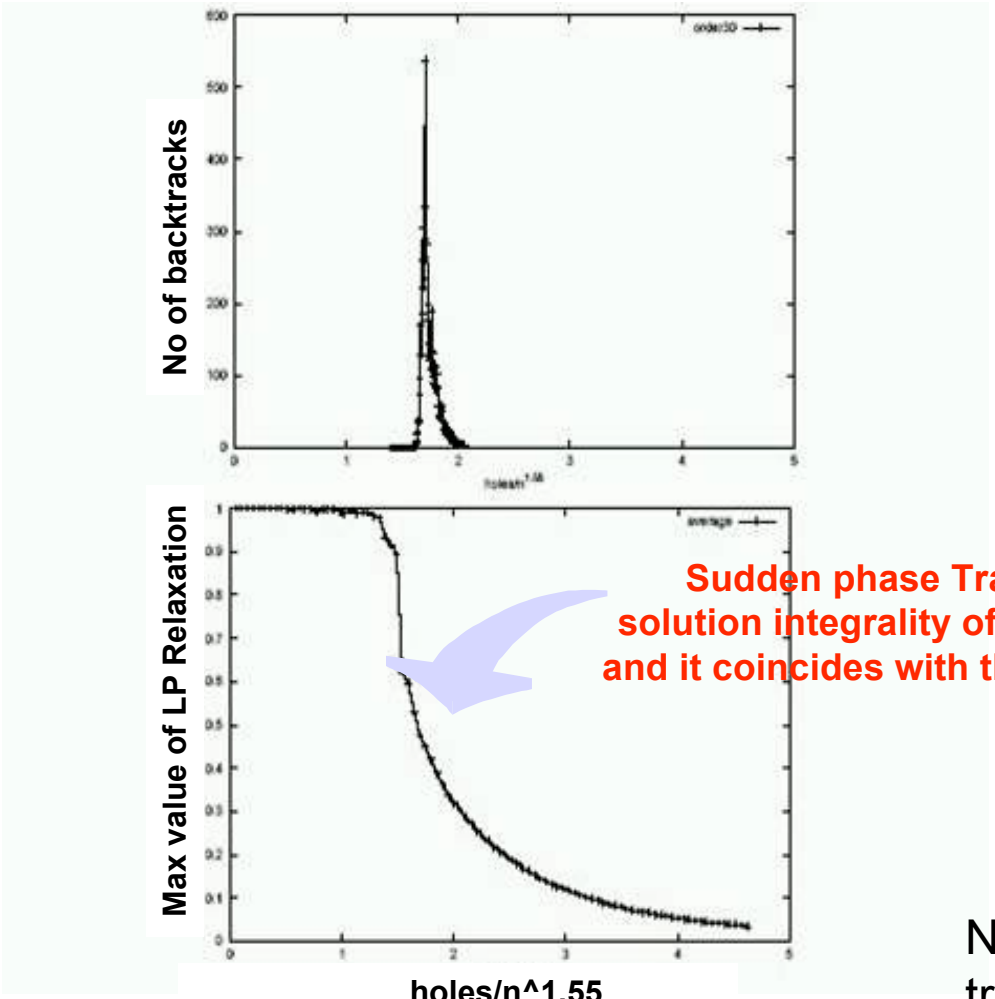
Row/color line

Column/color line

Row/column line

Scaling: up to order 20

New Phase Transition Phenomenon: Integrality of LP

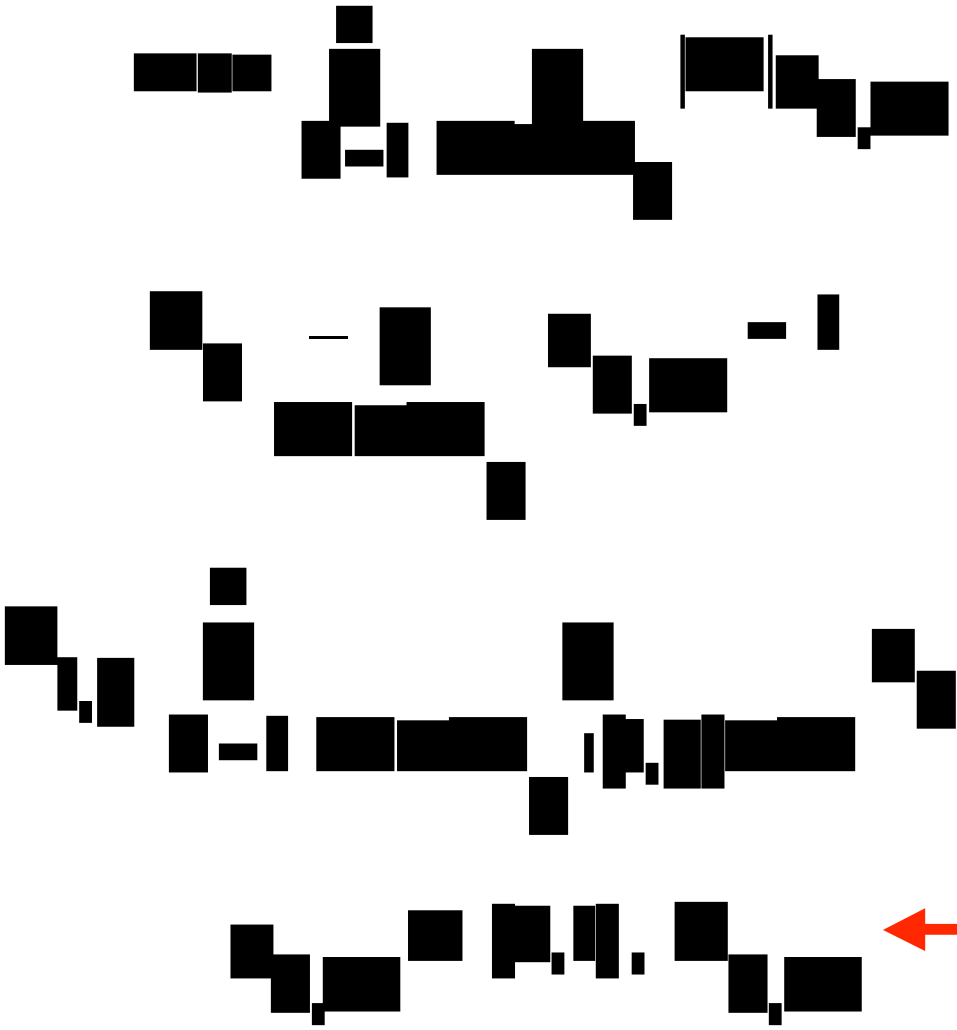


Sudden phase Transition in solution integrality of LP relaxation and it coincides with the hardest area

Note: standard phase transition curves are w.r.t existence of solution)

Gomes and Leahu 04

Integer Programming: Packing Formulation



Max number of colored cells

one pattern per color

at most one pattern covering each cell

(1-1/e) Approximation Algorithm

Gomes and Shmoys 03

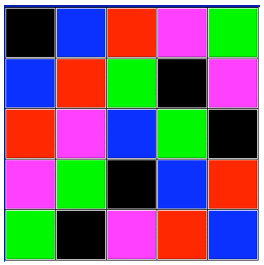
Constraint Satisfaction Problem (CSP)

- Variables

- Constraints

row

column



Scaling:
up to order 33

Intelligent Information Systems Institute

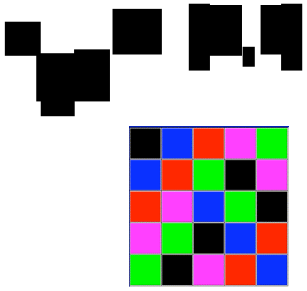
CORNELL UNIVERSITY

- **Hybrid CSP + LP**
  - **CSP propagation**
  - **(1-1/e) - Approximation Algorithm (based on the packing formulation)**

Scaling:
up to order 36

Satisfiability
Minimal Encoding

- Variables: 



Each variables represents a color assigned to a cell.

- Clauses: 

- Some color must be assigned to each cell (clause of length n); 

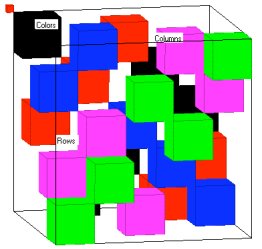
Scaling:
up to order 20

- No color is repeated in the same row (sets of negative binary clauses); 

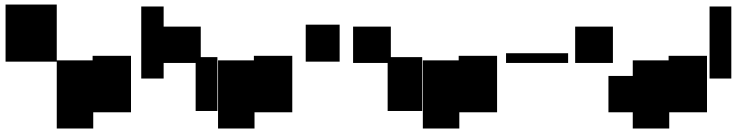
- No color is repeated in the same column (sets of negative binary clauses); 

Satisfiability: Extended Encoding (redundant clauses)

- Variables:  Same as minimal encoding.
- Clauses: Same as the minimal encoding plus:
  - Each color must appear at least once in each row;

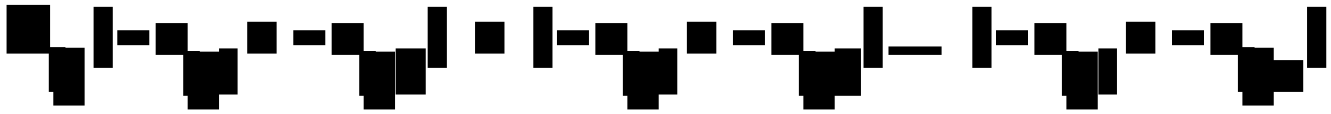


  - Each color must appear at least once in each column;



  - No two colors are assigned to the same cell;

Scaling: up to order 40



The best performing encoding

Encoding is critical when dealing with combinatorial problems:

From order 20 (400) variables to order 40 (1600) variables

The most compact representation is not necessarily the best performing

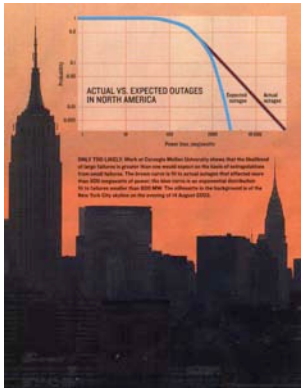
Connections between Heavy-tailed Distributions, Backdoors, and Restart Strategies in Complete Search Methods for Combinatorial Problems

Intelligent Information Systems Institute

CORNELL UNIVERSITY

Phenomena Defying "Standard" Statistical Distributions

Tsunami 2004



Blackout of August 15th 2003
> 50 Million People Affected

All these phenomena are characterized by distributions that have very "heavy" tails

Financial Markets with huge crashes

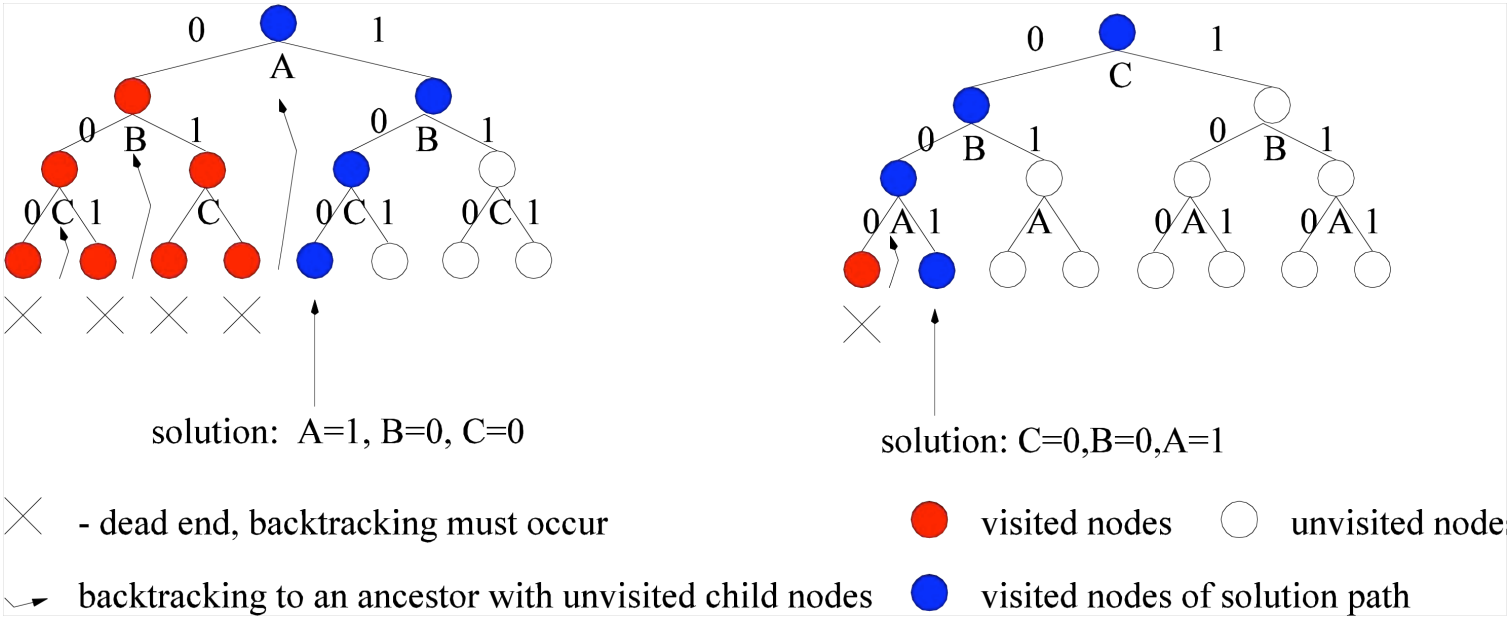
… there are a few billionaires

Intelligent Information Systems Institute

CORNELL UNIVERSITY

Heavy-Tailed Phenomena in Combinatorial Search

(A OR NOT *B* OR NOT C) AND (B OR NOT C) AND (A OR C)



Backtrack Search methods also exhibit Heavy-tailed Phenomena

The size of the search tree varies dramatically , depending on the order in which we pick the variables to branch on

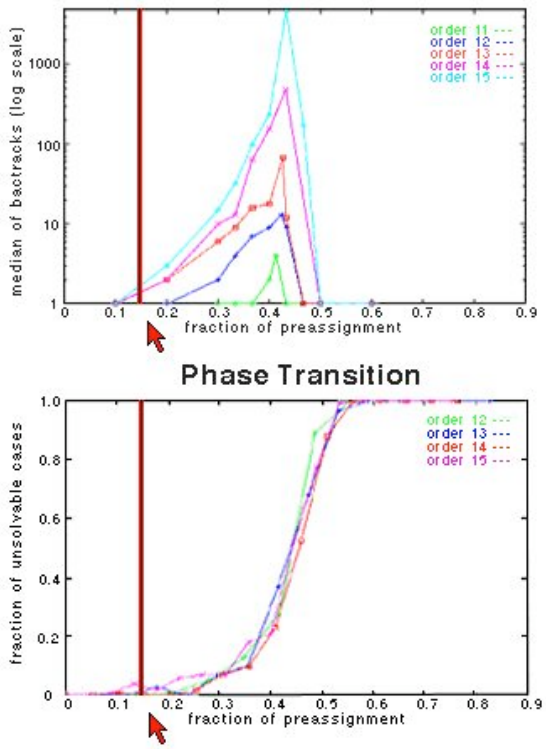
Backtrack Search

Main Underlying Search Mechanisms for Complete Search Methods:

Mathematical Programming (MP)
Constraint Programming (CP)
Satisfiability

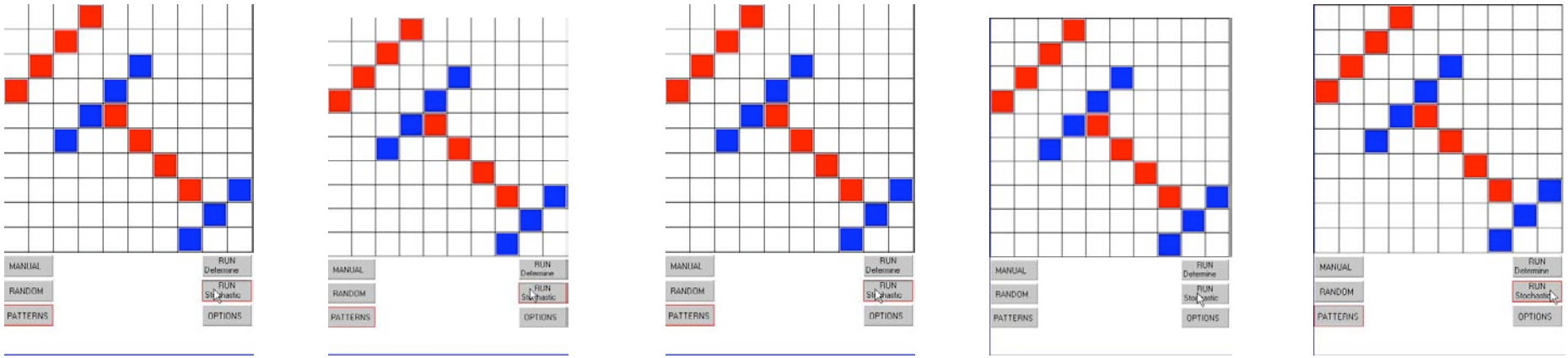
Branch & Bound;
Branch & Cut;
Branch & Price;
Davis-Putnam-Logemann-Lovelan Proc.(DPLL)

What if the we introduce an element of randomness into a complete backtrack search method without losing completeness?



Randomized Backtrack Search

Several runs of the a complete randomized backtrack search procedure (tie breaking only) on the same instance

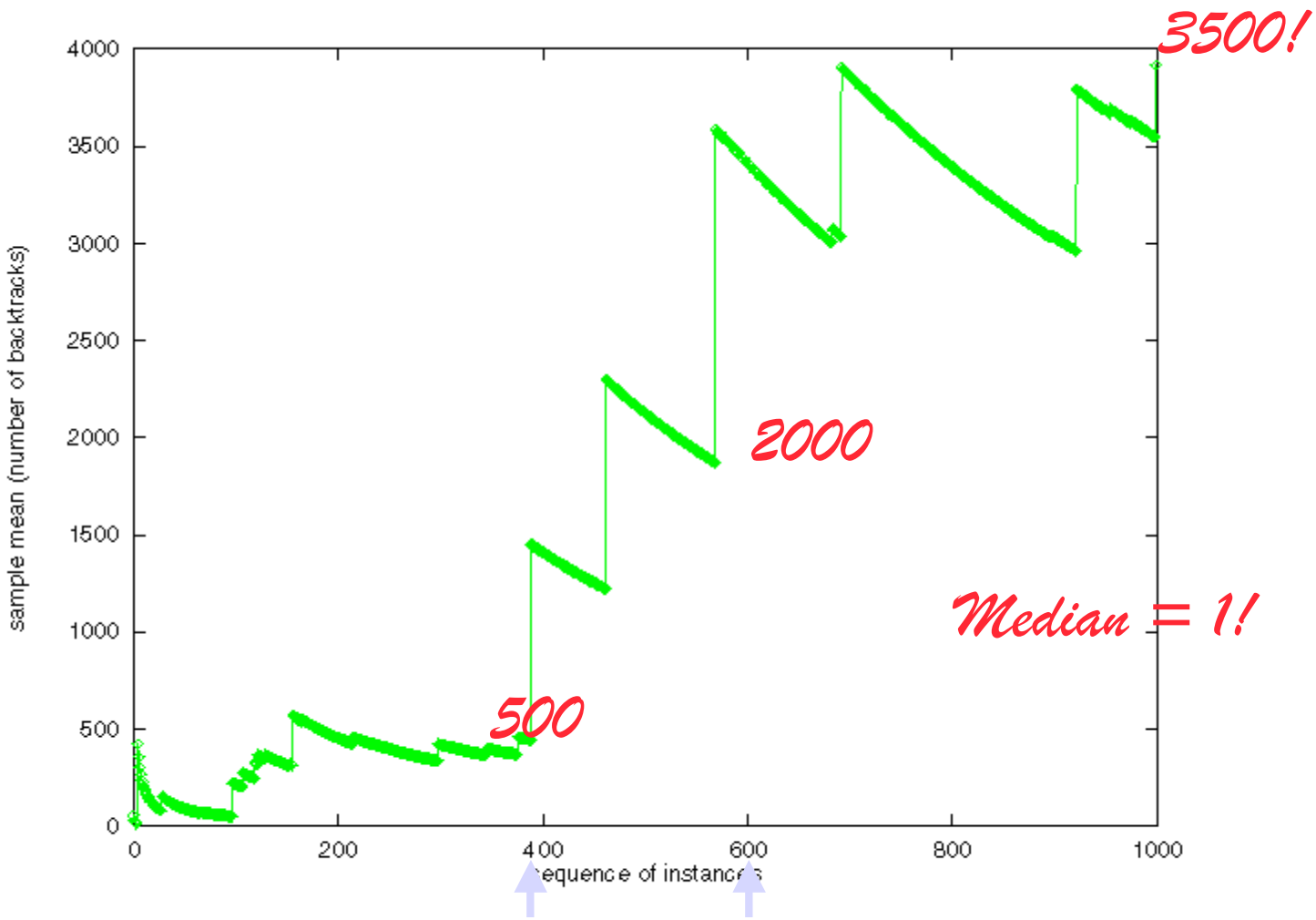


Time: **7** **11** **30** (*) (*)

(*) no solution found - reached cutoff: 2000

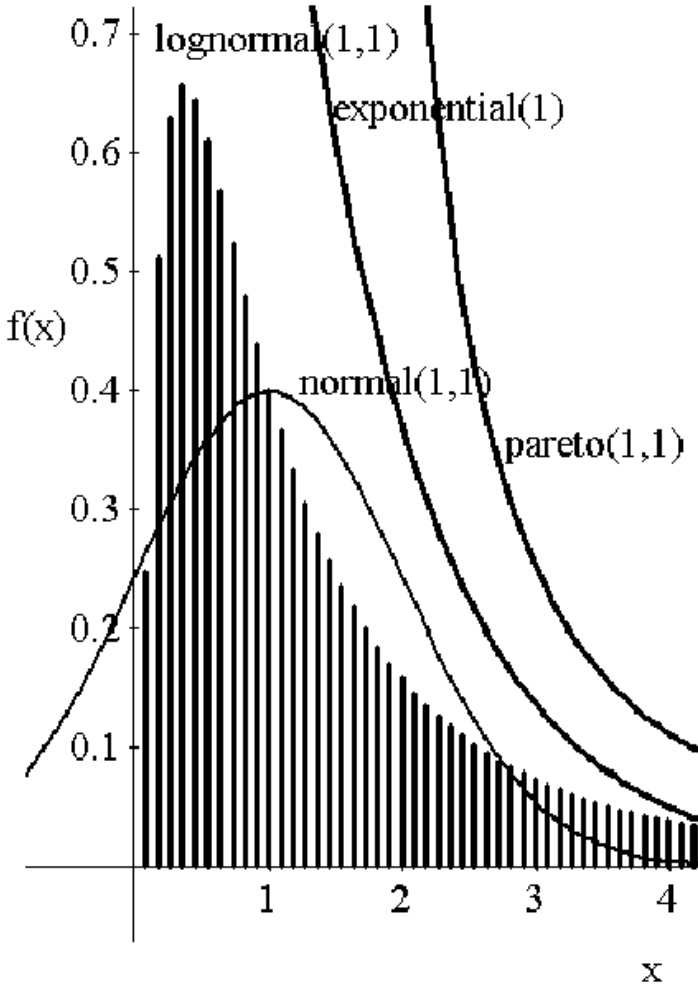
Erratic Behavior of Sample Mean

sample mean



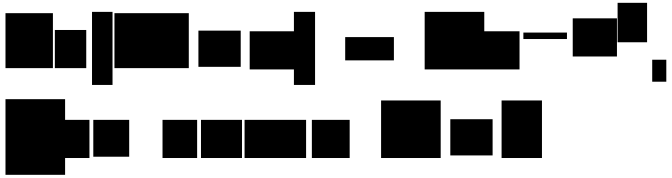
number of runs

Heavy-tailed distributions vs. Standard Distributions



Exponential decay for standard distributions, e.g. Normal, Logonormal, exponential:

Normal →

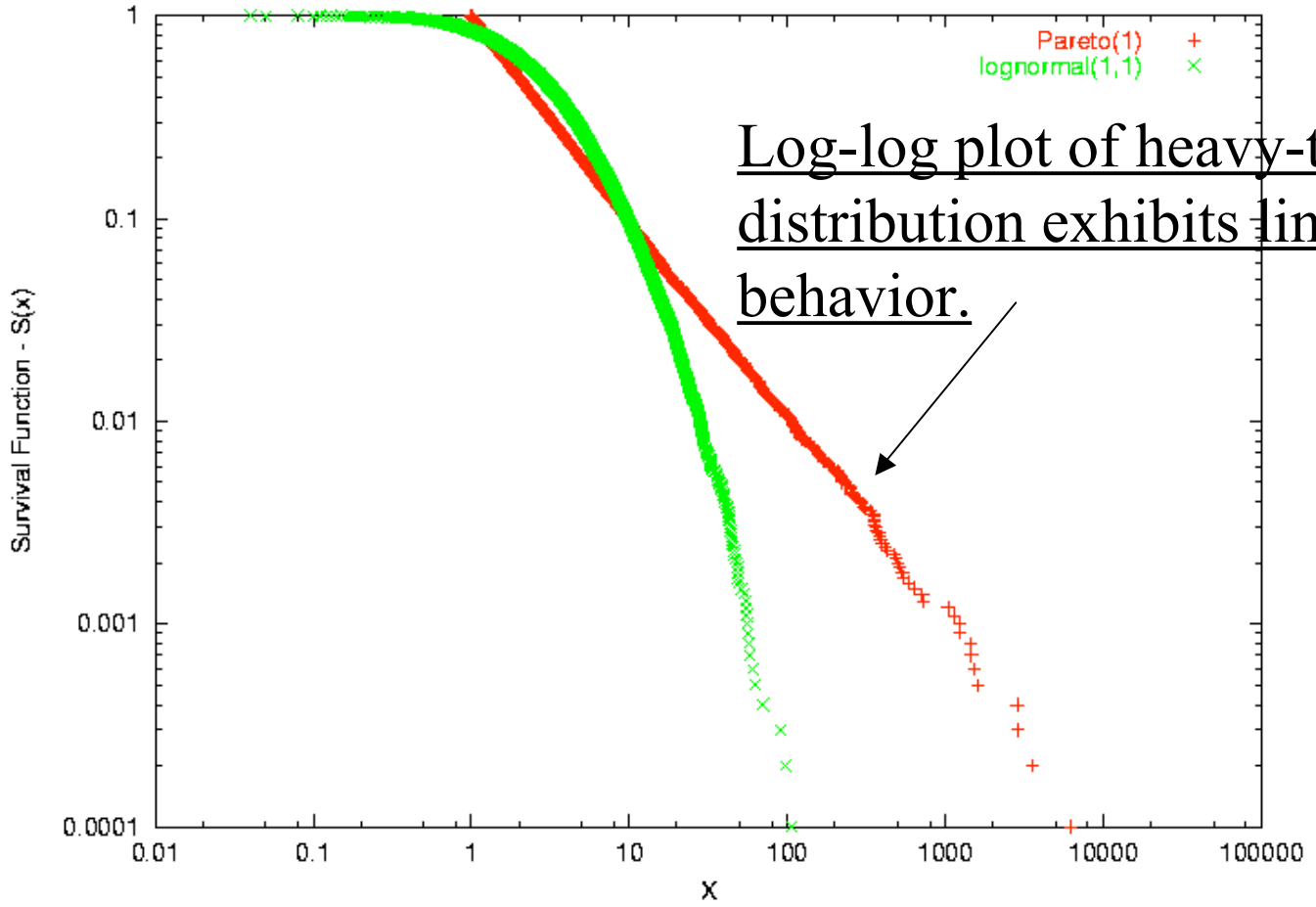


Heavy-Tailed Distributions - Power Law Decay e.g. Pareto-Levy:

Intelligent Information Systems Institute

CORNELL UNIVERSITY

Survival Function (Tail): Heavy-Tailed vs. Non-Heavy-Tailed

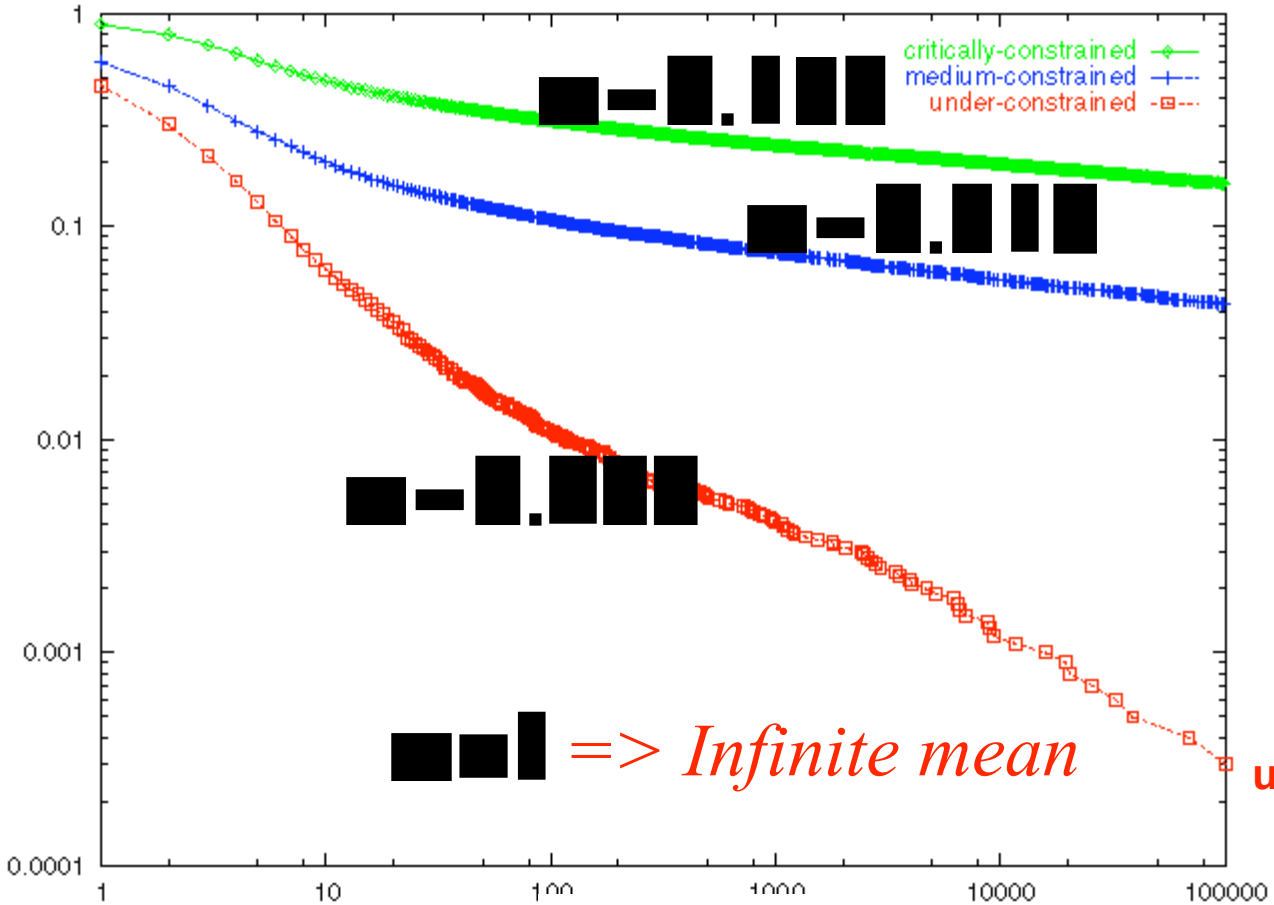


Log-log plot of heavy-tailed distribution exhibits linear behavior.

Intelligent Information Systems Institute

CORNELL UNIVERSITY

Heavy-Tailed Behavior in Latin Square Completion Problem



(1-F(x))(log) Unsolved fraction

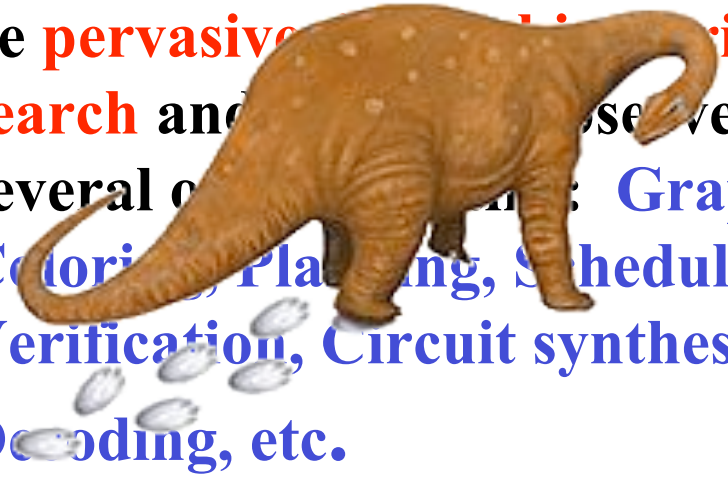
Number backtracks (log)

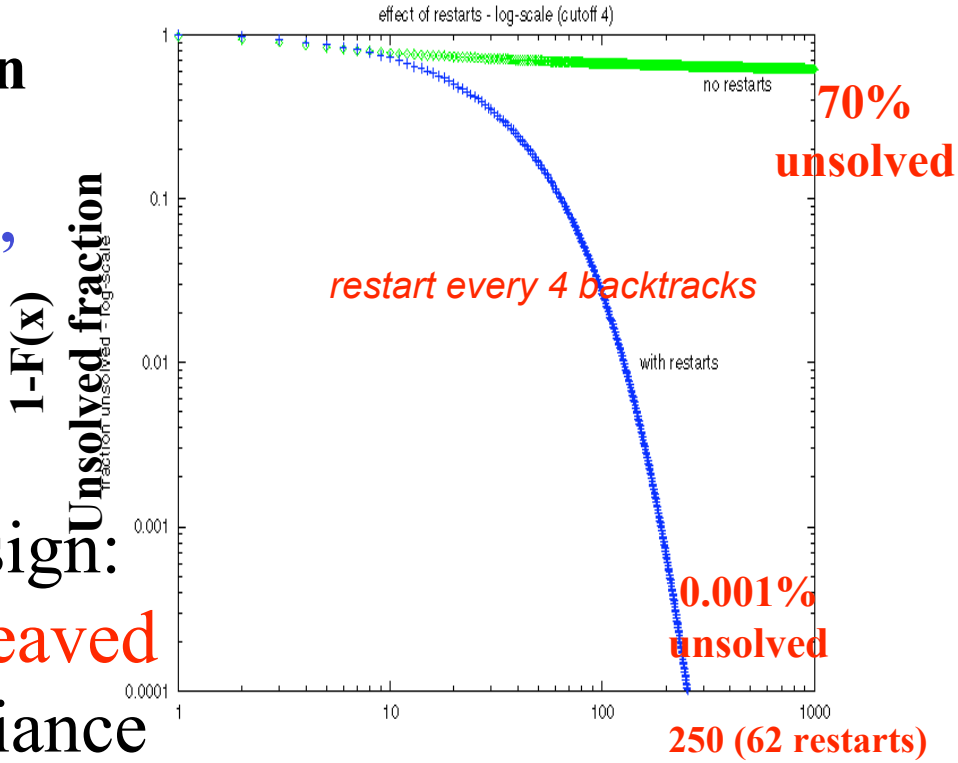
18% unsolved

0.002% unsolved

Restarts: Exploiting Heavy-Tailed Behavior

Heavy Tailed behavior seems to be pervasive in ... search and observed in several ... : Graph Coloring, Planning, Scheduling, Verification, Circuit synthesis, Decoding, etc.





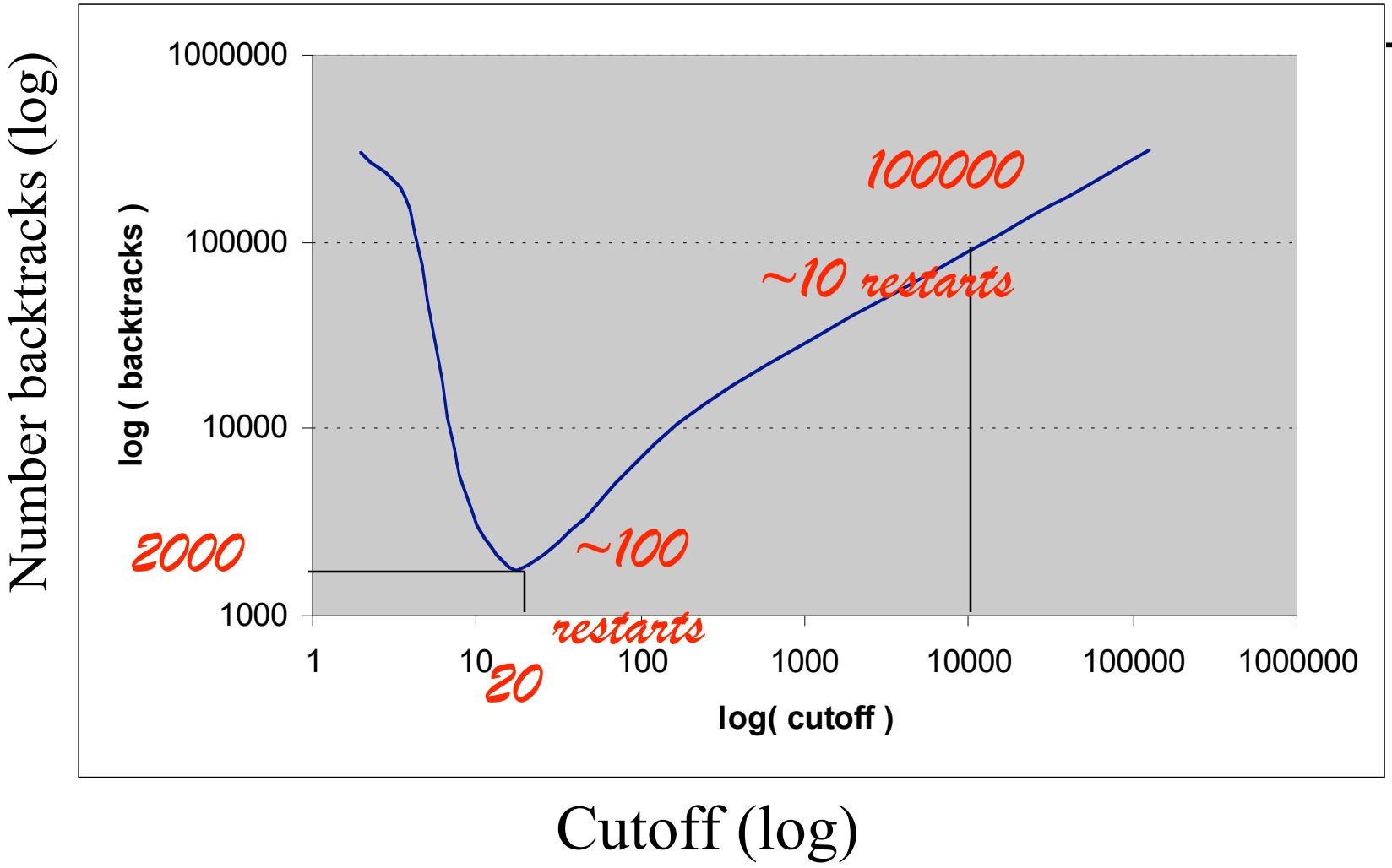
Consequence for algorithm design:
Use restarts or parallel / interleaved runs to exploit the extreme variance in performance ---

most state-of-art SAT solvers use restarts.

(Gomes et al. 97, 98, 2000)

Restarts provably eliminate heavy-tailed behavior

Speedup with Restarts (planning instance)



Formal Models: On the connections between backdoors and heavy-tailedness

Carla P. Gomes
Connection 2
Caltech06

Heavy-tailed distributions

➔Explain very long runs of complete solvers;

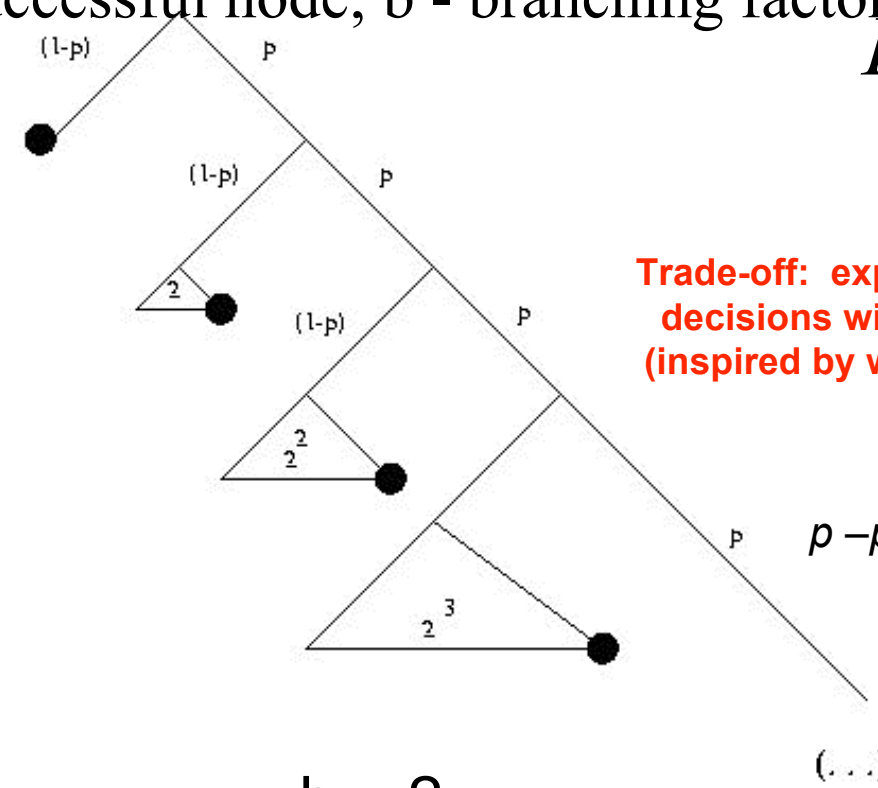
➔But also imply the existence of a wide range of solution times, often from very short runs to very long

How to explain short runs?

Formal Model Yielding Heavy-Tailed Behavior

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

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



b = 2

● successful leaf

Trade-off: exponential decay in making wrong branching decisions with exponential growth in cost of mistakes. (inspired by work in information theory, Berlekamp et al. 1972)

p –probability of not finding the "critical" variable

1 "special/critical variable"

(Gomes 00; Chen, Gomes, and Selman 01)

What is the semantics of these "special/critical variable"

Expected Run Time $p \geq 1/2 \quad E[T] \rightarrow \infty$

(infinite expected time)

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

(infinite variance)

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

(heavy-tailed)

p –probability of not finding the "special/critical variable"

Backdoors

Carla P. Gomes
Connection 2
Caltech06

Backdoors: intuitions

Informally:

A backdoor to a given problem is a subset of "critical" variables such that, once assigned values, the remaining instance simplifies to a tractable class (not necessarily syntactically defined).

Backdoors explain how a solver can get "clever" and solve very large instances

Formally:

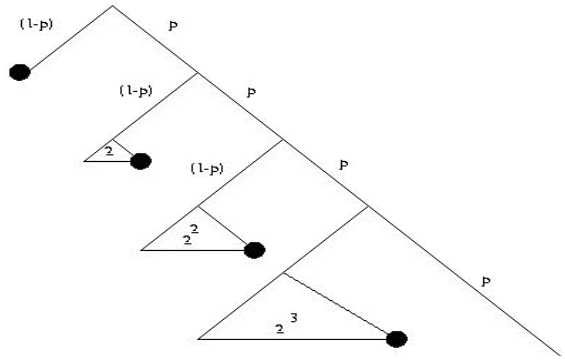
We define notion of a "sub-solver" (handles tractable substructure of problem instance) and *Backdoors* and *strong backdoors*

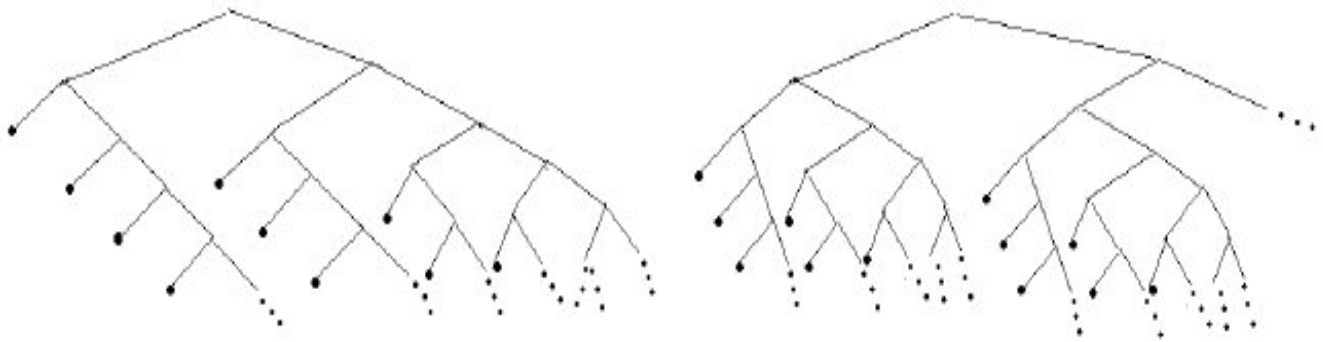
Note on Definition of Sub-solver

- Definition is general enough to encompass any **polynomial time propagation** methods used by state of the art solvers:
  - –Unit propagation
  - –Arc consistency
  - –ALLDIFF
  - –Linear programming
  - –…
  - –Any polynomial time solver

- Definition is also general to include even polytime solvers for which there does not exist a clean syntactical characterization of the tractable subclass.
- Applies to CSP, SAT, MIP, etc

More than 1 backdoor



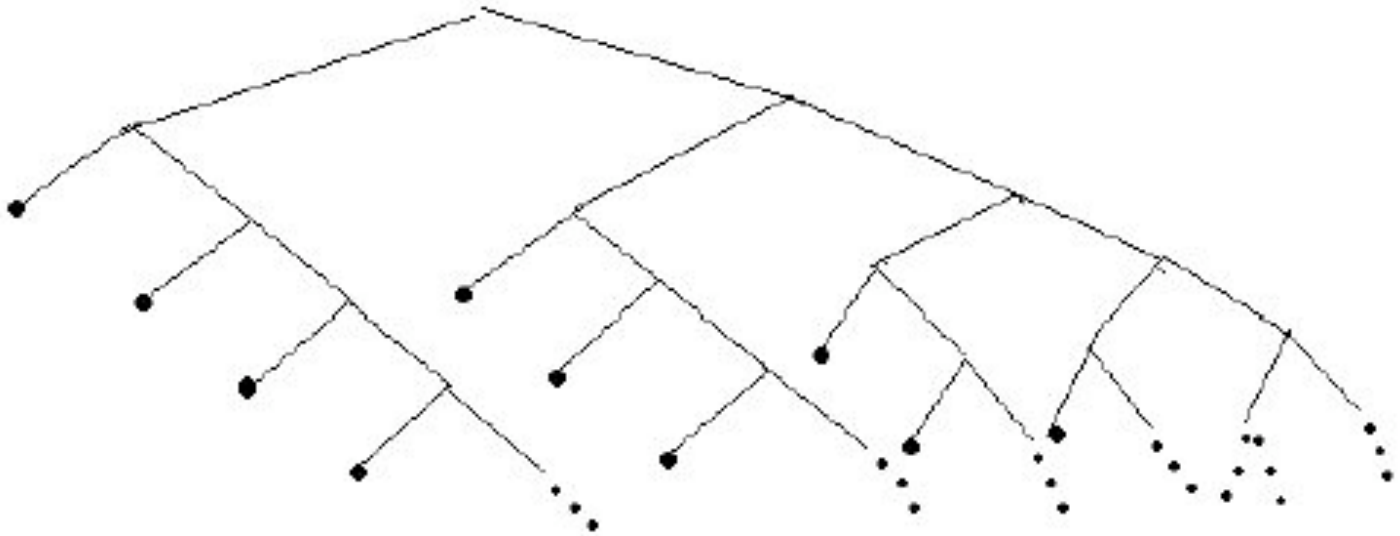


(Williams, Gomes, Selman 03)

Intelligent Information Systems Institute

CORNELL UNIVERSITY

Backdoors provide detailed formal model for heavy-tailed search behavior.



Theorem 3. *(Heavy-tail lower bound) If* $B \in o(N/\log N)$ *and the probability of heuristic failure* $(1-1/h)$ *is less than* $1/b^{\alpha}$ *for* $\alpha \in (0,2)$*, then the tail probability of the search cost on* $V(B)$ *is lower bounded by a heavy-tail.*

Can formally relate size of backdoor and strength of heuristics (captured by its failure probability to identify backdoor variables) to occurrence of heavy tails in backtrack search.

Backdoors in real-world problems instances

Carla P. Gomes
Connection 2
Caltech06

Gap between Theory and Practice: Tractable Problem Sub-structure

Backdoors →

Hidden *tractable* substructure in real-world problems

Backdoors can be surprisingly small:

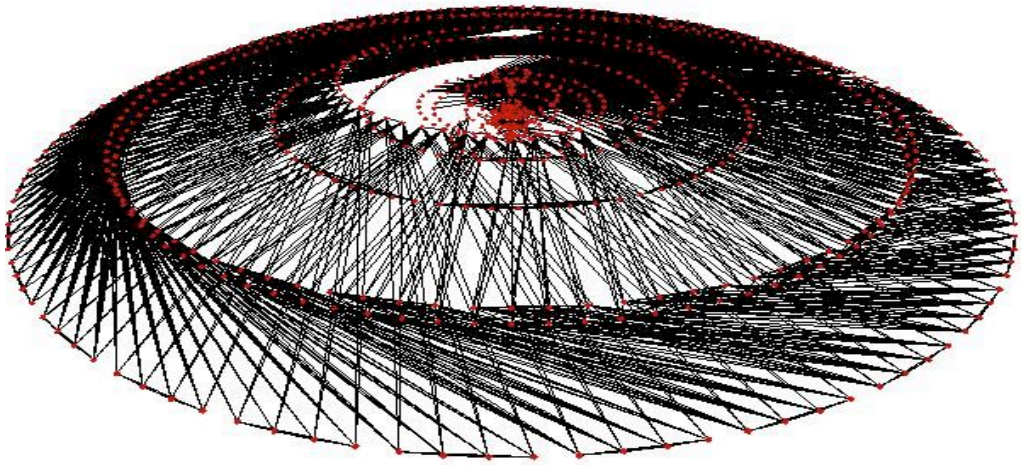
Backdoors explain how a solver can get "lucky" on certain runs, when the backdoors are identified early on in the search.

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

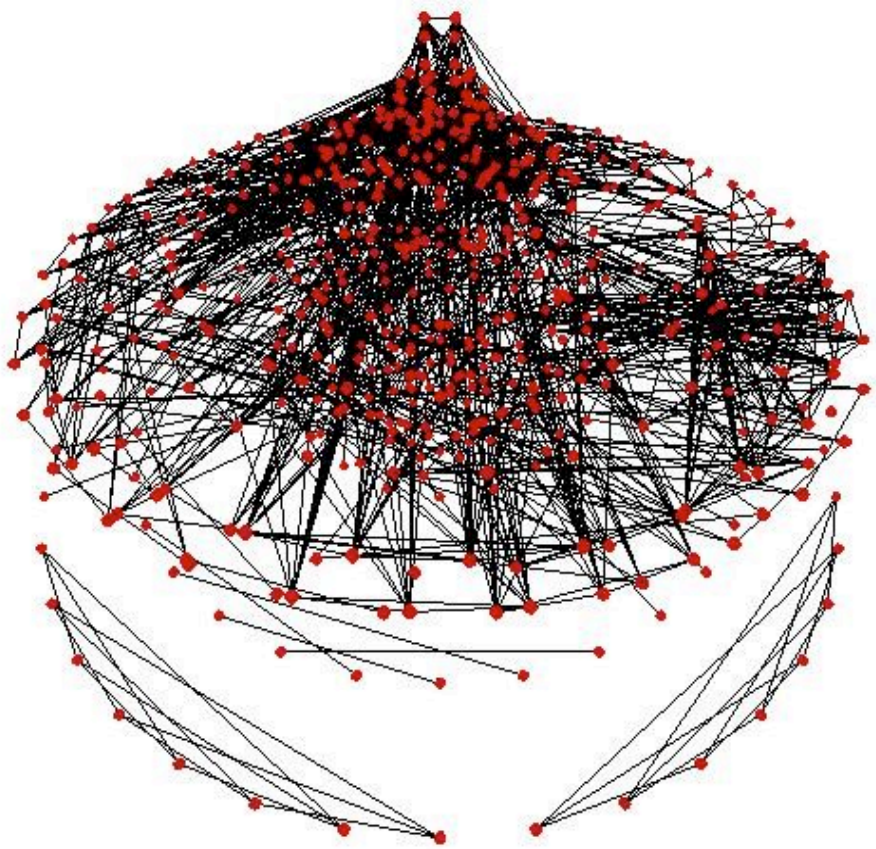
Most recent: Other combinatorial domains. E.g. graphplan planning, near constant size backdoors (2 or 3 variables) and log(n) size in certain domains. (Hoffmann, Gomes, Selman '05)

Backdoors capture critical problem resources (bottlenecks).

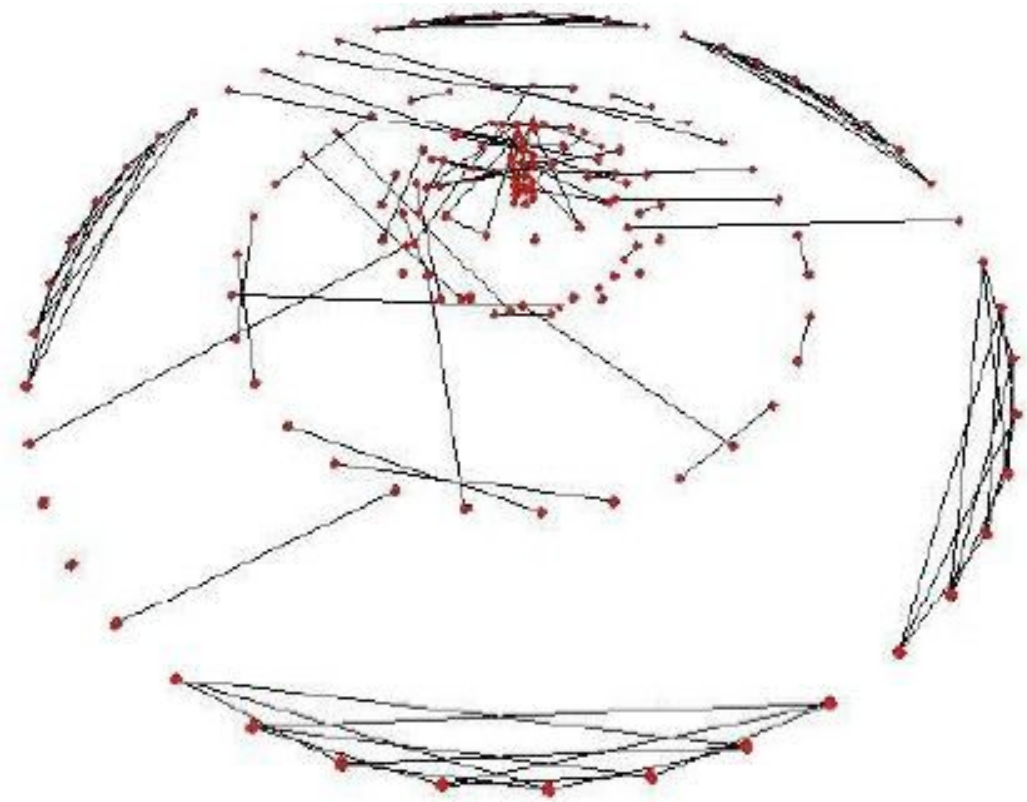
Backdoors --- “seeing is believing”



Logistics_b.cnf planning formula.
843 vars, 7,301 clauses, approx min backdoor 16

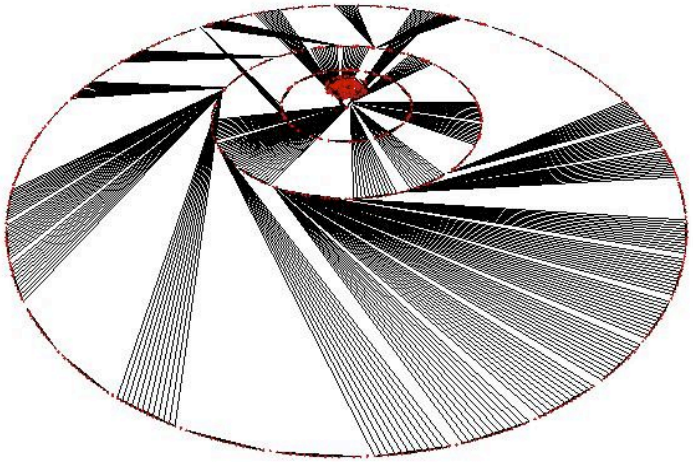


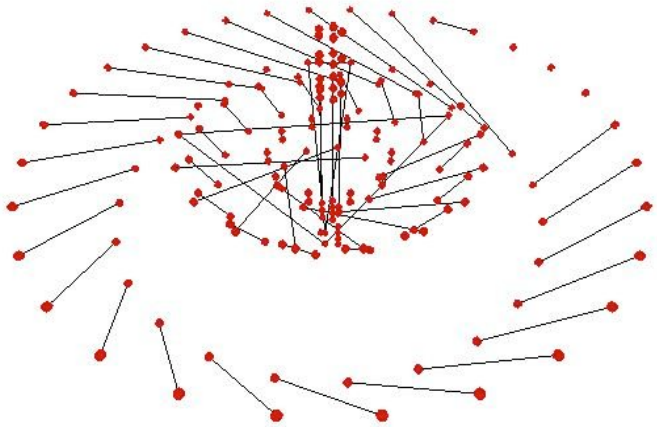
Logistics.b.cnf after setting 5 backdoor vars (result after propagation);

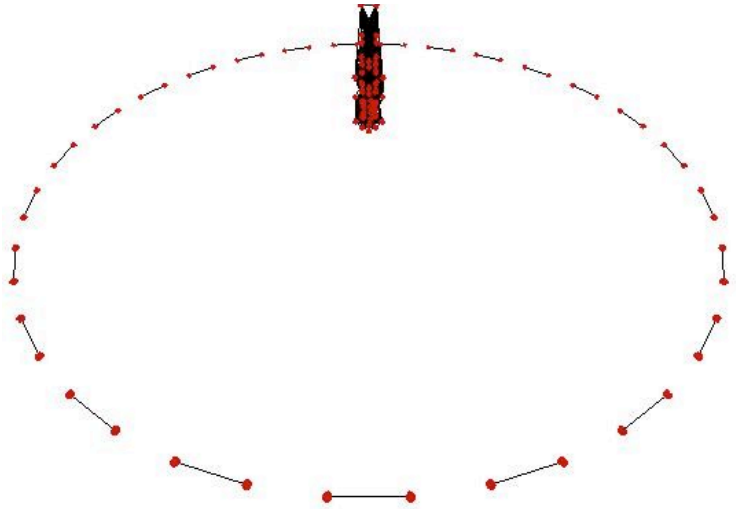


After setting just 12 (out of 800+) backdoor vars – problem almost solved.

Some other intermediate stages:

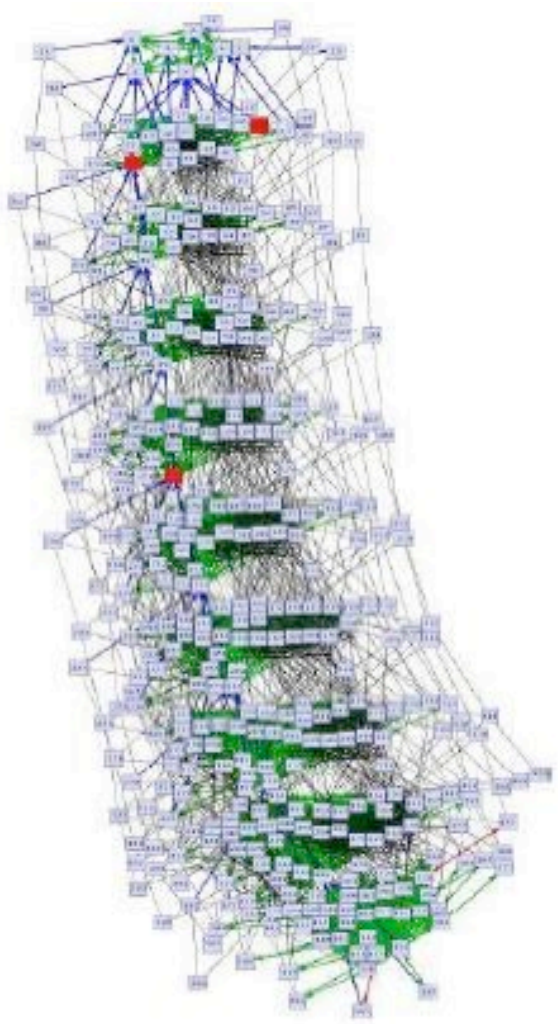






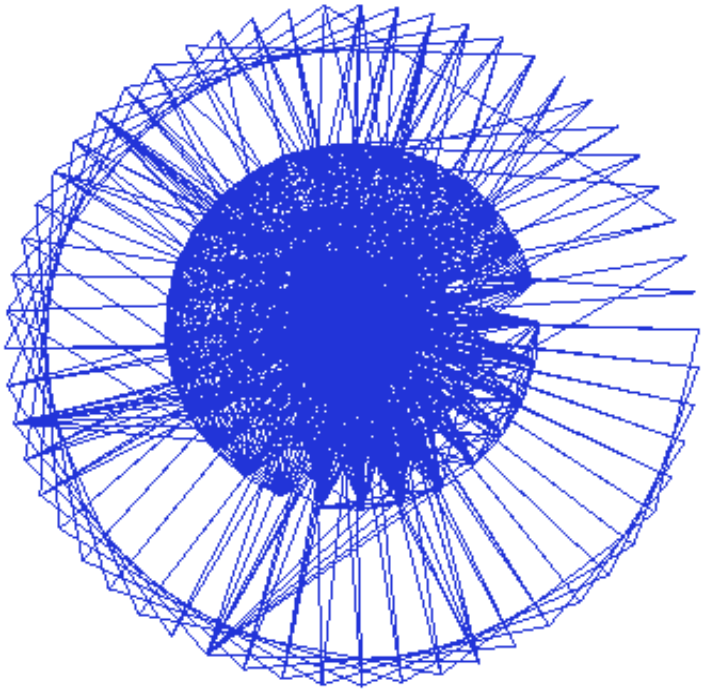
After setting 38 (out of 1600+) backdoor vars:

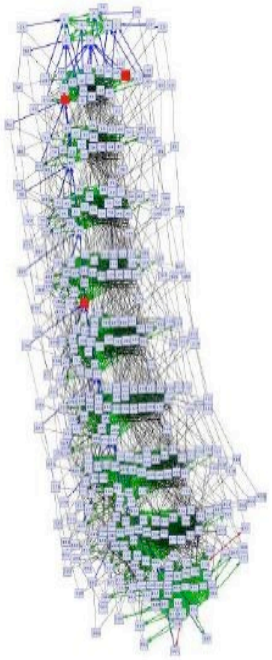
Tractable structure *hidden* in the network. Related to small-world networks etc.



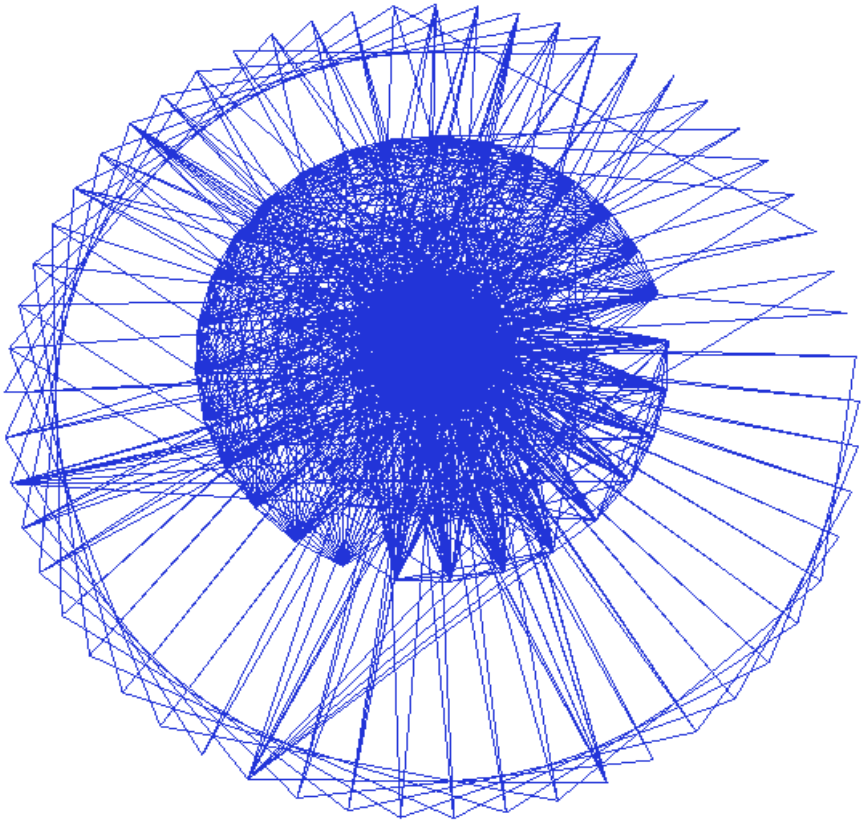
MAP-6-7.cnf infeasible planning instances. Strong backdoor of size 3.
392 vars, 2,578 clauses.

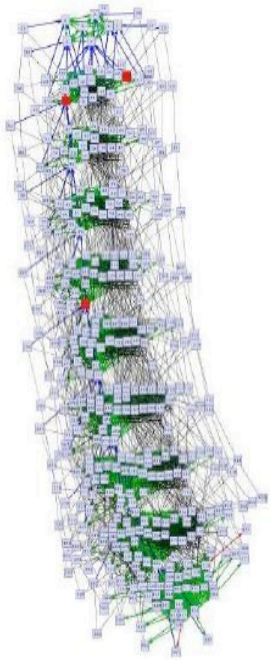
Map Top: running without backdoor



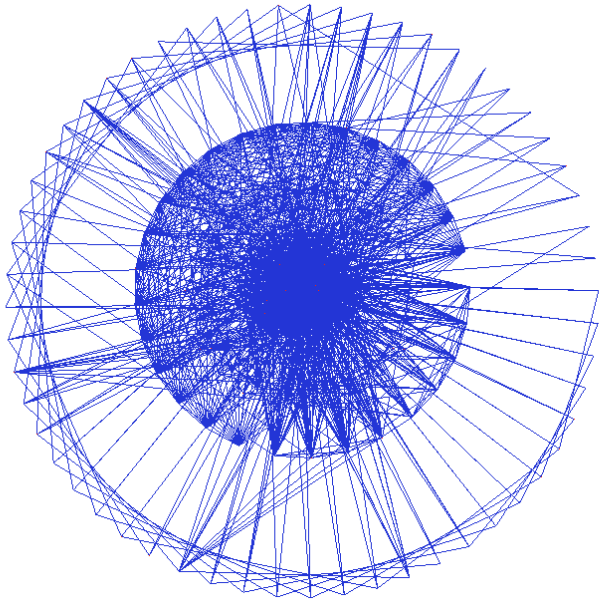


Map Top: running with “a” backdoor
(size 9 – not minimum)

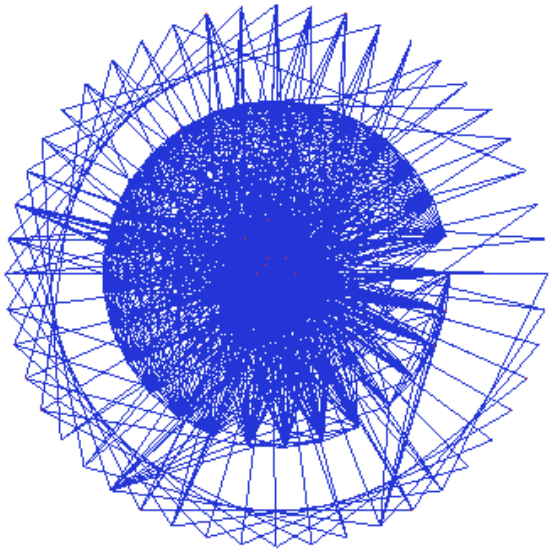




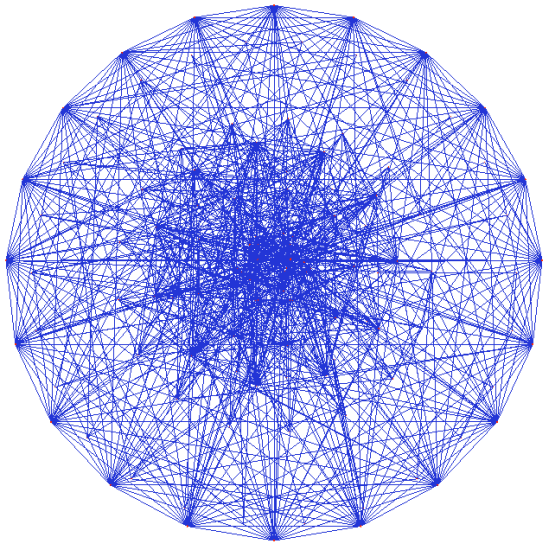
Map Top: running with backdoor (minimum – size 3)



Initial Graph



After setting one backdoor



After setting two backdoors

Instance is Unsatisfiable

After setting three backdoors

How to Exploit Backdoors?

We need to take into account
the cost of finding the backdoor!

We considered several complete algorithms:

- **Generalized Iterative Deepening**
- **Randomized Generalized Iterative Deepening**
- **Complete randomized backtrack search with variable and value selection heuristics**

  (as in current solvers)

Formal results

Backtrack Search with randomized heuristic for Var/Value selection ↓

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

(Williams, Gomes, and Selman '04)

Current solvers

Backdoor set detection is fixed-parameter tractable for HORN and 2CNF (Nishimura, et. al 04)

Streamlining Constraint Reasoning

Carla P. Gomes
Connection 2
Caltech06

Intelligent Information Systems Institute

CORNELL UNIVERSITY

Design of Scientific Experiments (4 Treatments)

Standard Approach:
Analysis of Variance Based on Latin Squares

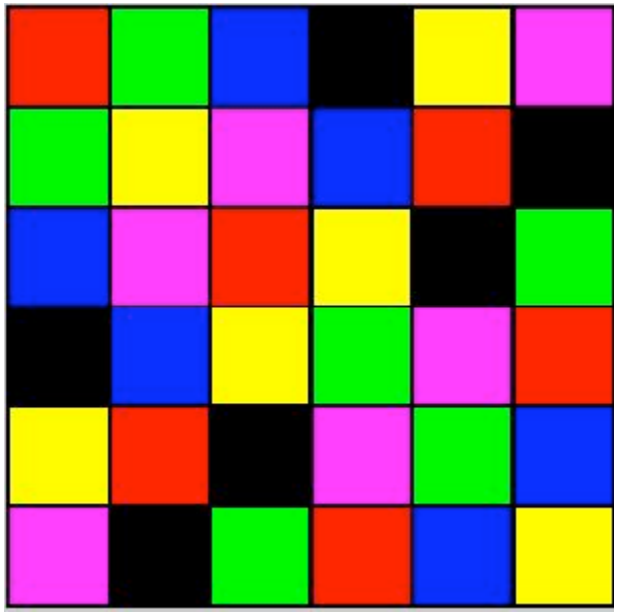
Carla P. Gomes
Connection 2
Caltech06

Design of Scientific Experiments (4 Treatments: A,B,C,D)

For some experiments: need so called Spatially Balanced Latin Squares

Carla P. Gomes
Connection 2
Caltech06

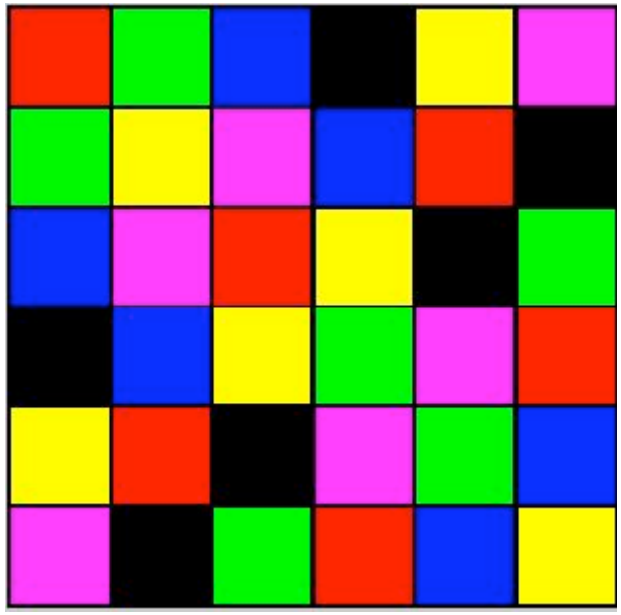
Spatially Balanced Latin Squares (SBLS)

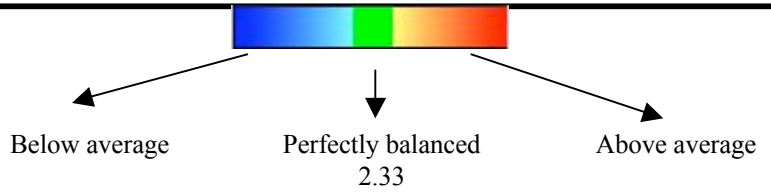


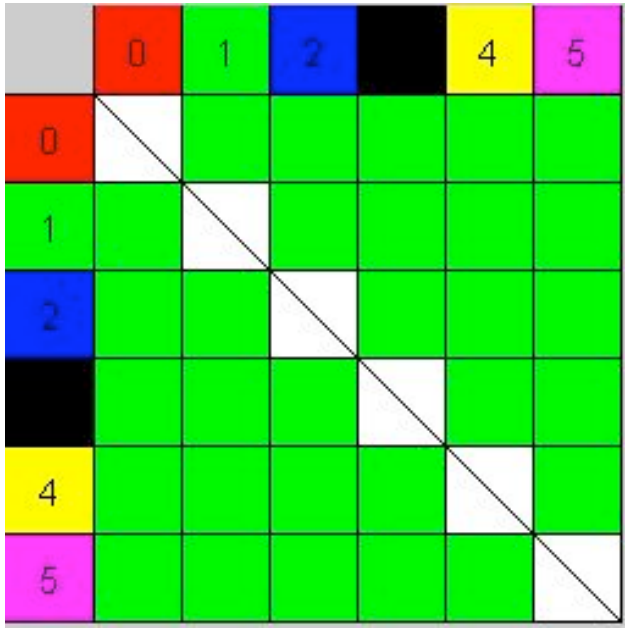
	1	2	3	4	5
	4	1	1	3	2
	3	2	2	1	1
	2	4	5	3	1
	3	4	4	1	2
	1	1	1	2	3
Total Row Distance (pair)	14	14	14	14	14
Average Row Distance (pair)	2.33	2.33	2.33	2.33	2.33

Carla P. Gomes
Connection 2
Caltech06

Spatially Balanced Latin Squares (SBLS)







A totally spatially balanced Latin square (TSBLS):
the total row distance is
the same for all pairs of colors or symbols

Carla P. Gomes
Connection 2
Caltech06

Standard Approaches to SBLS

- Hybrid IP/CSP based
  - Assignment formulation
  - Packing formulation
  - Different CSP models
- CSP based approach
  - State of the art model for Latin Squares

  + symmetry breaking by initializing first row and column
  (SBDD doesn't help; this is not a completion problem)
- Local search based approach

  (Gomes and Sellmann CPAIOR 04)

These approaches do not scale up

max order 6.

What to do when:

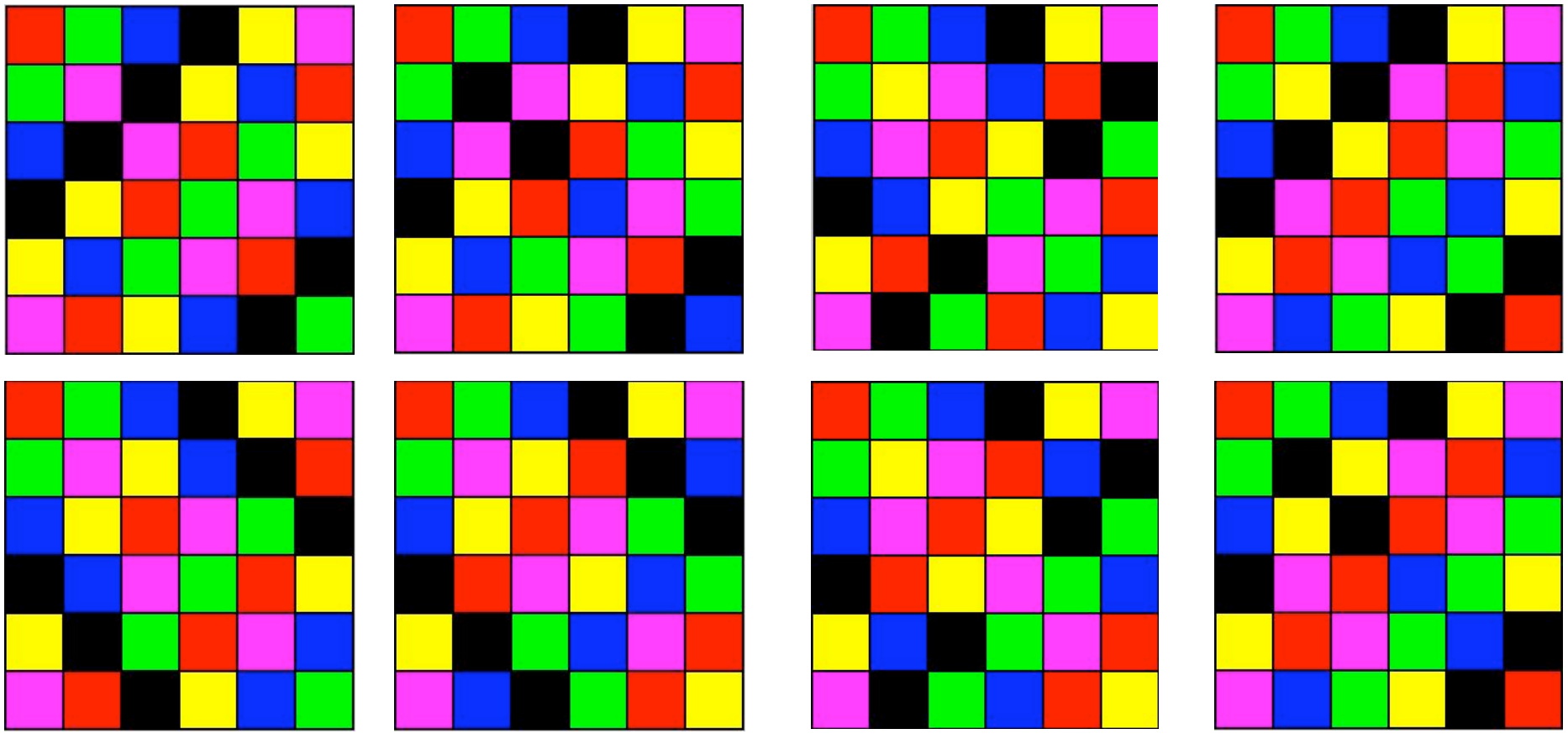
Local search doesn't help;

Backtrack search with all sophisticated enhancements performs very poorly;

…

and you believe there are lots of solutions that we cannot find?

Streamlined Diagonally Symmetric SBLS: Order 6



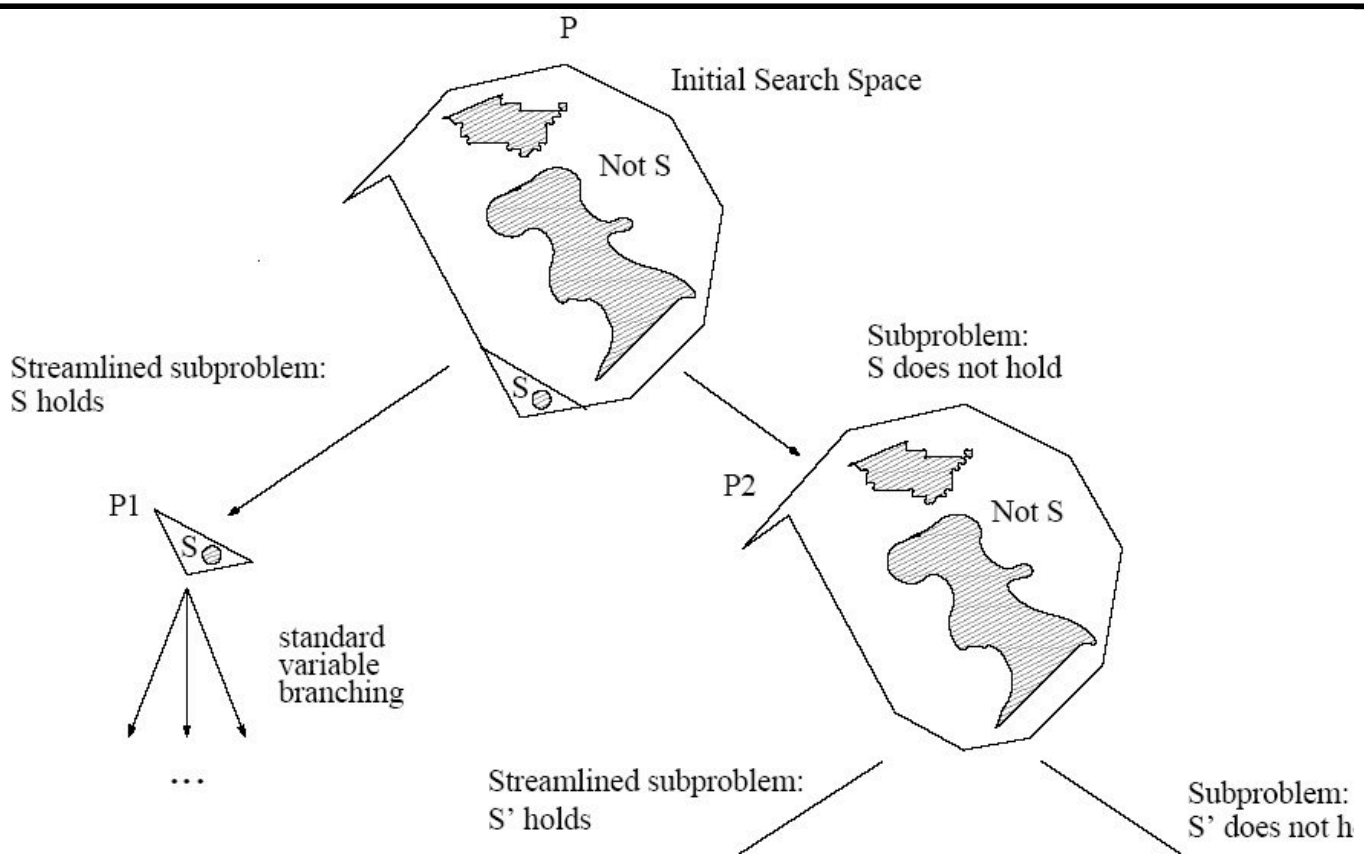
Carla P. Gomes
Connection 2
Caltech06

Streamlining Reasoning: Key ideas

Goal: Exploit the structure of solutions
to dramatically
boost the effectiveness
of the propagation mechanisms

Note:
Hard practical limit on the effectiveness of constraint propagation methods,
if one insists on maintaining the full solution set;
Often there is no compact representation for all the solutions
(e.g., Latin Squares)

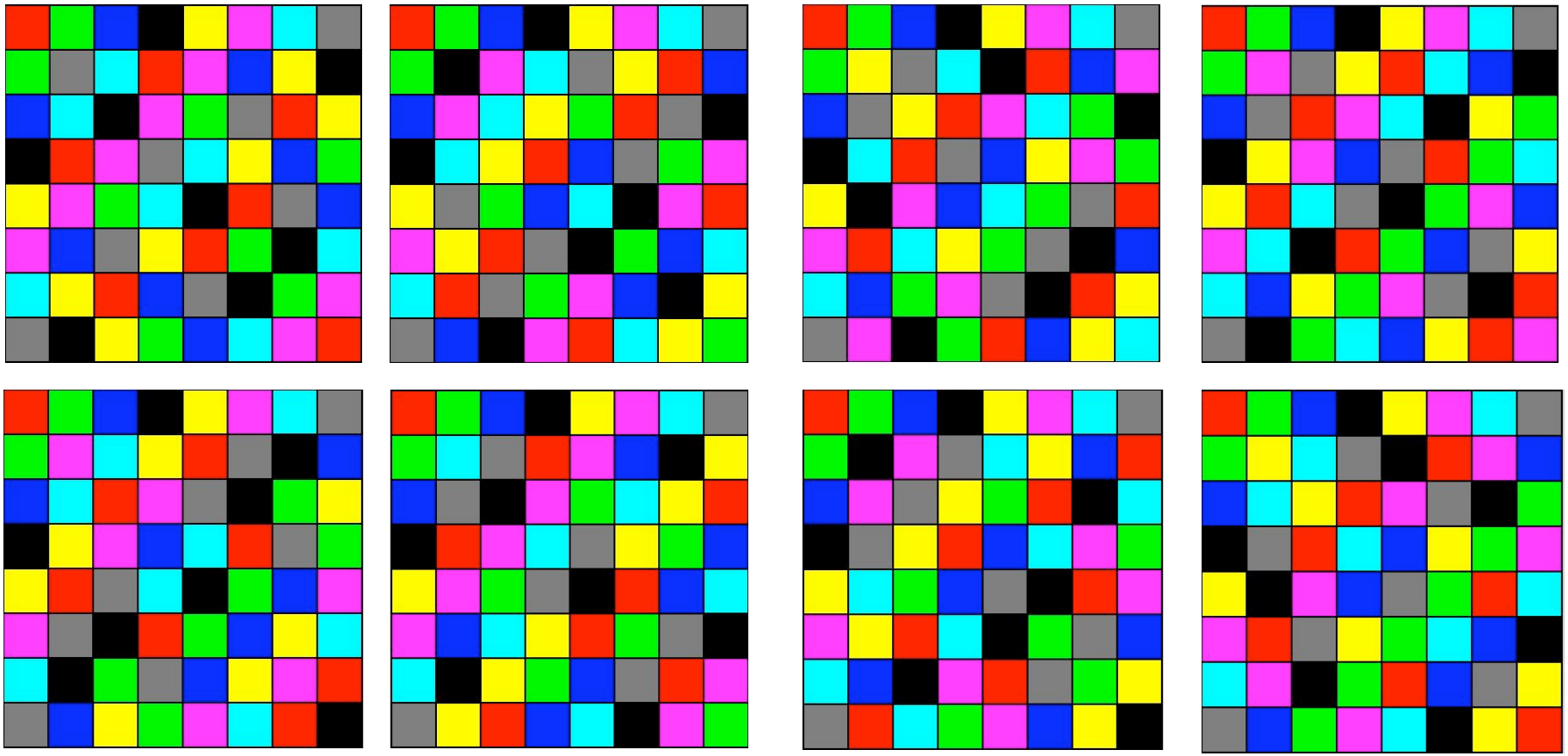
Streamlining in Terms of Global Search



P1
Substantially smaller than its complement
P2

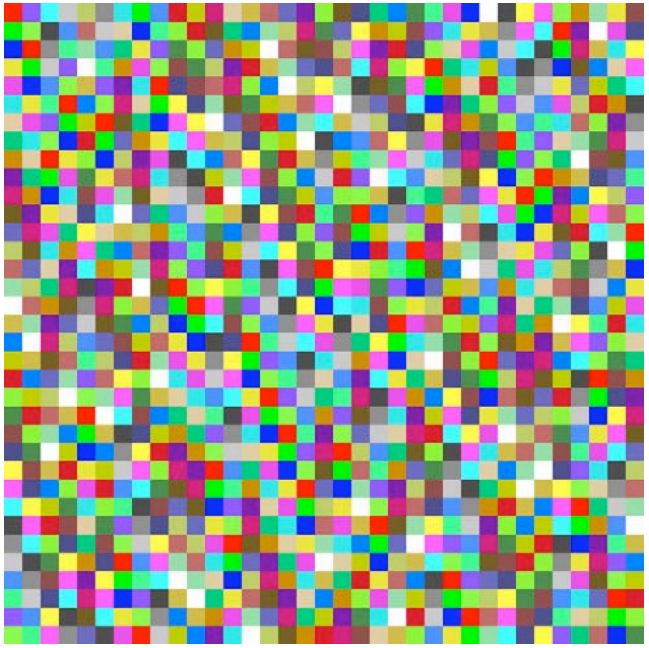
Streamlining:
strong branching mechanisms at high levels of the search tree.

Streamlined Diagonally Symmetric SBLS: Order 8



Carla P. Gomes
Connection 2
Caltech06

A SBLS of Order 35: The Largest Ever Found



Carla P. Gomes
Connection 2
Caltech06

Intelligent Information Systems Institute

CORNELL UNIVERSITY

Cornell can now provide the templates for experimental design to NYS Ag. Laboratories (Land-grant mission) ☺

But this is highly domain dependent…
Can we find an domain independent way of streamlining?

YES
XOR-Streamlining

Not only does it allow us to FIND solutions it also allow us to COUNT solutions.

Beyond Satisfaction: Model Counting

Counting and Sampling

The ability to count/sample solutions effectively opens up a wide range of new applications.

SAT testing → **counting/sampling**

↓ ↓

logic inference **probabilistic reasoning**

Counting Solutions is much harder than finding a single model

#P-complete **vs.** ***NP / co-NP-complete***

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

Related work: Bayardo 98, Kautz et al. '04; Bacchus et al. '03; Darwich '04 & '05; Littman '03, Nishimura, Radge, Szeider, 06.

#SAT: Counting solutions

- # SAT is very hard in the worst case
- NP-Hard to approximate within $2^{n^{1-\varepsilon}}$ $\varepsilon > 0$
- Hardness results apply even for tractable cases of SAT (e.g., Horn, and 2CNF)

Model Counting: Two Paradigms

1. **Approximate counters.** E.g. Markov Chain Monte Carlo methods. Based on setting up a Markov chain with a predefined stationary distribution. E.g. simulated annealing. **Markov chain takes exponential time to converge to its stationary distribution.** NO guarantees on quality of approximation.

2. **Exact counters.** Modifications of DPLL (backtrack style) SAT solvers. **Need to traverse full exponential search space.** Very Expensive.

Model Counting: A New Approach

Can we count solutions in a "totally" different way using a state-of-the-art SAT solver AS-IS?

SAT solver says just "satisfiable" or "unsatisfiable".
Hmm???

YES!!!

Recent work with Ashish Sabharwal and Bart Selman 06

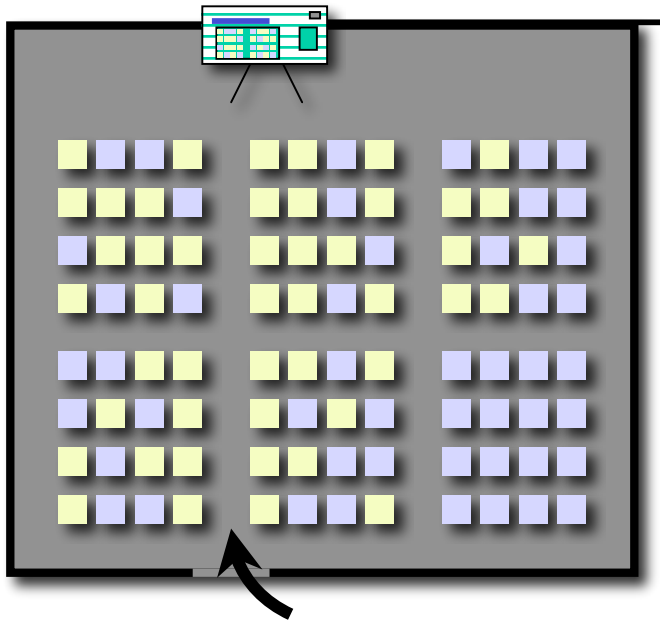
Intelligent Information Systems Institute

CORNELL UNIVERSITY

Model Counting: Intuition of Our Approach

How many people are present in this room?

Model Counting: Intuition of Our Approach



Does this work?

- On average, YES.

Everyone starts with a hand up

- Everyone tosses a coin
- If heads, keep hand up, if tails, bring hand down
- Repeat till only one hand is up

Return $2^{\#(\text{rounds})}$

Making the Intuitive Idea Concrete

- How can we implement this coin flipping strategy given that we know nothing about the solution space structure
  - Solutions are "hidden" in the formula
- How do we transform the average behavior into a robust method with provable correctness guarantees?

Somewhat surprisingly, all these issues can be resolved!

Intelligent Information Systems Institute

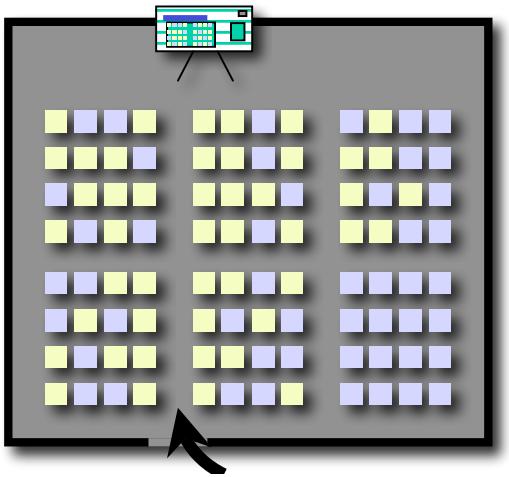
CORNELL UNIVERSITY

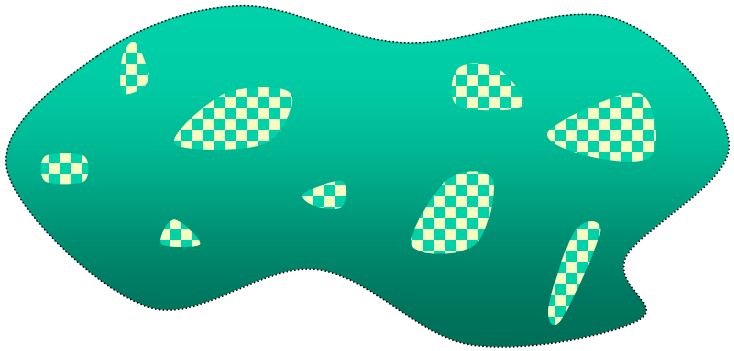
From Counting People to #SAT

Given a formula F over n variables,

- Auditorium : search space for F
- Seats : 2^n truth assignments
- Occupied seats : satisfying assignments

Bring hands down : add a constraint eliminating those satisfying assignments





Model Counting: A New Approach

Approach inspired by

- (1) "Streamlining Constraint Reasoning" Divide search space by adding streamlining constraints.
- (2) Work of Valiant and Vazirani (1986) on "Unique SAT"

"Unique SAT" problem.

If we have a formula with at most unique/single satisfying assignment, is it easier than an arbitrary (satisfiable) formula?

SAT and UNIQUE SAT are essentially equivalent in terms of hardness.

Valiant and Vazirani (1986): **Mainly viewed as a negative result: "knowing more does not help."**

XOR/Parity Constraints

Unique Sat Proof (very clever): Add random parity / XOR constraints to formula to cut down #solns to 1 with high probability.

XOR/parity constraints

- *E.g.* $a \oplus b \oplus c \oplus d = 1$
  (satisfied if an odd number of variables set to True)
- Translates into a small set of CNF clauses
- probabilistically streamline the search space (XOR-Streamlining)

→ an XOR constraint cuts down the number of satisfying assignments in half (in expectation), acting as a "hash function", splitting the set of truth assignments in an "accept" and "reject" bucket.

Model Bound (MBound): Counting with XOR Streamlining Constraints

Algorithm 1: MBound

Params: k, s, t, δ, α : k, s, t positive integers,
$k \leq n/2,\ 0 < \delta \leq 1/2,\ \alpha \geq 1$
Input : A CNF formula F
Output: A lower or upper bound on $MC(F)$, or Failure
begin
  $numSat \leftarrow 0$
  for $i \leftarrow 1$ **to** t **do**
    $Q_s \leftarrow \{s$ random constraints from $\mathbb{X}^k\}$
    $F_s^k \leftarrow F \cup Q_s$
    $result \leftarrow$ SATSolve(F_s^k)
    if $result =$ TRUE **then** $numSat \leftarrow numSat + 1$
  if $numSat \geq t \cdot (1/2 + \delta)$ **then**
    return Lower bound: $MC(F) > 2^{s-\alpha}$
  else if $numSat \leq t \cdot (1/2 - \delta)$ **then**
    return Upper bound: $MC(F) < 2^{s+\alpha}$
  else return Failure
end

Basically:

Add s XOR parity constraints to the original formula F

Thm: If F is still satisfiable after s random XOR constraints, then F has $\geq 2^{s-\alpha}$ solutions with prob. $\geq$ $(1-1/2^{\alpha})$

Gomes, Sabharval, Selman AAAI06

Key Features of MBound and Hybrid Bound

- Can use any off-the-shelf state-of-the-art SAT solver –only one solution needs to be found.
- Random XOR constraints independent of both the problem domain and the SAT solver used
- Adding XORs further constrain the problem
  - Can model count formulas that couldn't even be solved!
  - An effective way of "streamlining" [Gomes-Sellmann '04]
    → XOR streamlining
- Provable upper and lower bounds on the model counts, with confidence that can be boosted arbitrarily by repeated runs.
- Further boost in performance → Hybrid Model Count = xor-streamlining + exact counter

Our approach

Experimental results

Instance	MBound		Relsat		Cachet		ApproxCount	
	Models	Time	Models	Time	Models	Time	Models	Time
Ramsey-20-4-5	$\geq \mathbf{1.2 \times 10^{30}}$	**< 2 hrs**	$\geq 7.1 \times 10^{8}$	12 hrs	—	12 hrs	$\approx 1.8 \times 10^{19}$	4 hrs
Ramsey-23-4-5	$\geq \mathbf{1.8 \times 10^{19}}$	**< 2 hrs**	$\geq 1.9 \times 10^{5}$	12 hrs	—	12 hrs	$\approx 7.7 \times 10^{12}$	5 hrs
Schur-5-100	$\geq \mathbf{2.8 \times 10^{14}}$	**< 2 hrs**	—	12 hrs	—	12 hrs	$\approx 2.3 \times 10^{11}$	7 hrs
Schur-5-140	$\geq \mathbf{6.7 \times 10^{7}}$	**< 5 hrs**	—	12 hrs	—	12 hrs	—	12 hrs
fclqcolor-18-14-11	$\geq \mathbf{2.1 \times 10^{40}}$	**3 mins**	$\geq 2.8 \times 10^{26}$	12 hrs	—	12 hrs	—	12 hrs
fclqcolor-20-15-12	$\geq \mathbf{2.2 \times 10^{46}}$	**9 mins**	$\geq 2.3 \times 10^{20}$	12 hrs	—	12 hrs	—	12 hrs

1) **Instances: Very hard combinatorial problems.**
2) **E.g. Schur_5_140 formula cannot be solved at all without XOR streamlining.**
3) **Confidence in bounds: >= 99.9% (can be arbitrarily boosted). Approxcount no guarantees.**
4) **Further results in Gomes, Sabharwal, abd Selman 06.**

Backdoors for #SAT

- #SAT is tractable for hitting formulas where any two clauses clash (i.e., have a complementary pair of literals)
- #SAT is tractable for clustering formulas (i.e., variable disjoint sets of hitting formulas)
- Fixed-parameter tractable algorithm for #SAT based on clustering formula (clustering width) backdoor set

Nishimura, Radge, Szeider, 2006

Quantification

Intelligent Information Systems Institute

CORNELL UNIVERSITY

Quantified Reasoning

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

Quantifiers prefix **Matrix**

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

•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.

QBF encodes adversarial tasks: 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.

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

Can we transfer successful SAT techniques to QBF?

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

Modern SAT solvers scale very well
(1M + variables),
QBF solvers don't! (~10 K vars)

Challenges to QBF

QBF is much more sensitive to problem encoding.

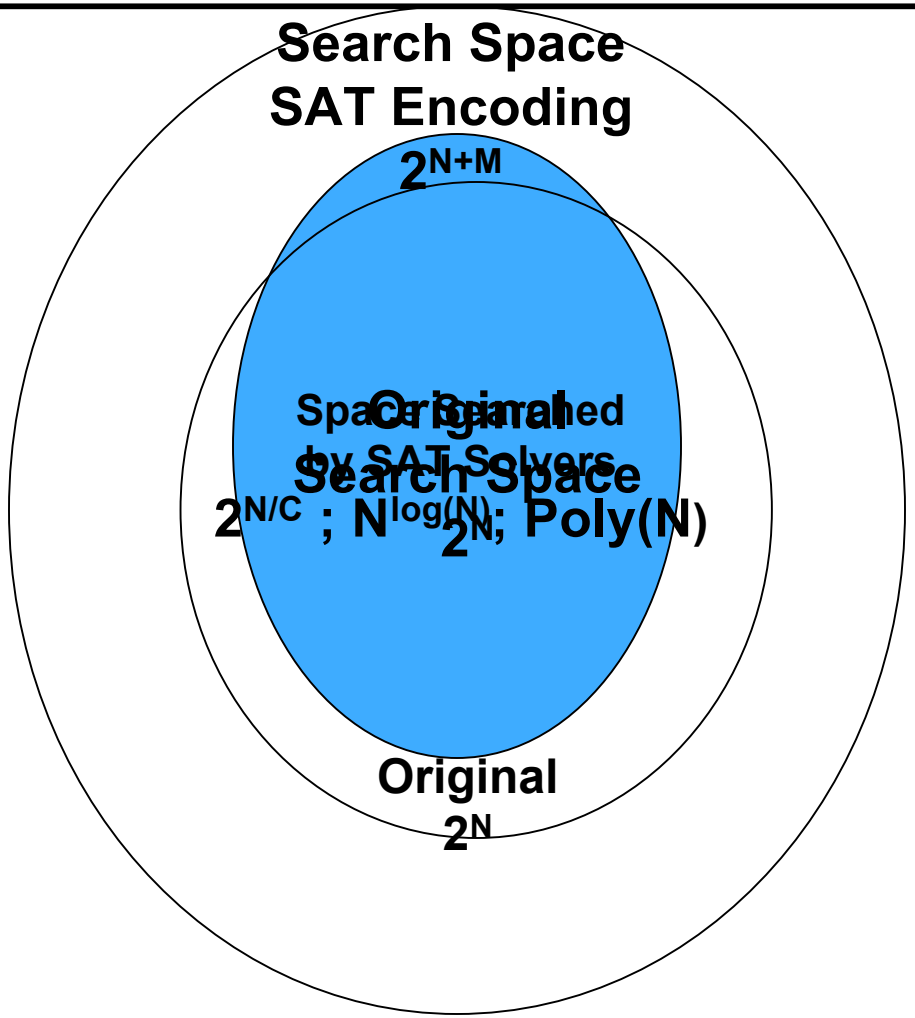
SAT/QBF encodings require auxiliary variables.
These variables significantly increase the raw combinatorial search space.

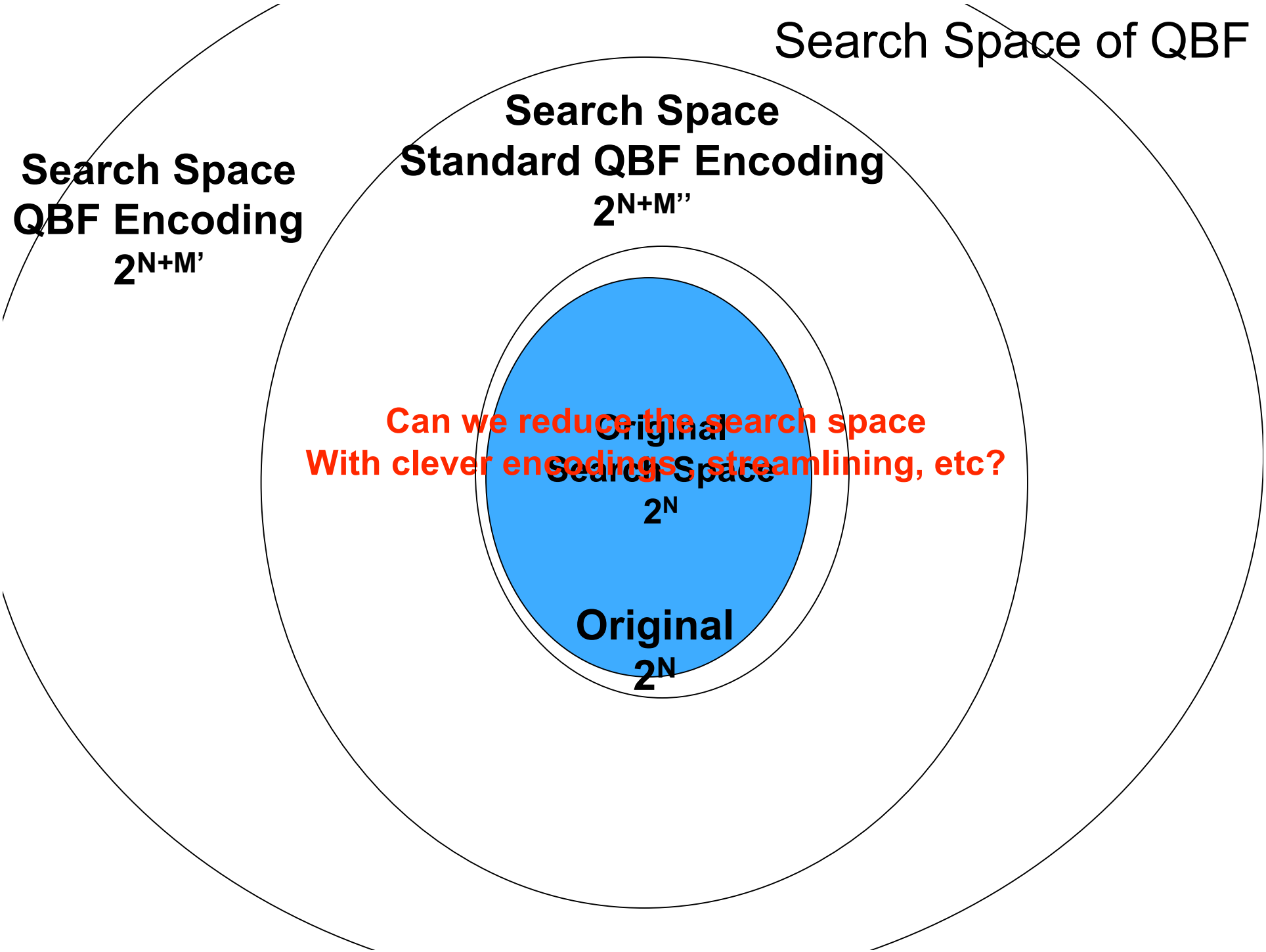
SAT: Propagation forces search to stay within combinatorial space of original task.

QBF – more problematic! Universal player pushes to violate domain constraints (trying to violate one or more clauses). Search leads quickly outside of search space of original problems.

→encodings have to be carefully engineered.

Search Space for SAT Approaches




Search Space of QBF
Search Space
QBF Encoding
2N+M'
Search Space
Standard QBF Encoding
2N+M''
Original
Search Space
2N
Original
2N
Can we reduce the search space
With clever encodings, streamlining, etc?

Boosting QBF Reasoning Using A Dual CNF-DNF Representation

- Pure CNF QBF encodings of games are fairly complex
  [e.g. Madhusudan-Nam-Alur 2003; Ansotegui-Gomes-Selman AAAI'05]
  - Hinder propagation across quantifiers
  - Lead to "illegal search space" issues
- A good **dual CNF-DNF encoding** can alleviate many problems
  - Symmetric structure and succinctness
  - **Orders of magnitude improvement in runtime**

QBF Modeling: Exploiting Player Symmetry for Simplicity and Efficiency [Sabharwal-Ansotegui-Gomes-Hart-Selman SAT'06]

Experimental Results

	xChess instance		Pure CNF Encoding				Dual Encoding
	name	T/F	Sempro p	sKizzo	Quaffle	Cond-Quaffle	Duaffle (without learning!)
5-15 quantifier levels (reachability)	conf-r1	F	12	4.0	15	1.3	**0.01**
	conf-r2	F	25	5.86	33	2.5	**0.02**
	conf-r3	F	55	9.3	62	4.1	**0.03**
	conf-r4	F	85	26	124	6.4	**0.04**
	conf-r5	F	985	84	676	34	**0.08**
	conf-r6	F	2042	73	713	49	**0.10**
7-9 quantifier levels	conf01	F	1225	492	--	539	**6.4**
	conf02	F	93	30	6.0	1.0	**0.0**
	conf03	T	--	1532	--	83	**1.4**
	conf04	T	--	--	2352	100	**3.5**
	conf05	F	3290	448	510	196	**0.1**
	conf06	F	--	memout	--	633	**30.6**
	conf07	F	261	42	78	3.5	**0.0**
	conf08	T	--	1509	--	1088	**31.2**

Experimental Results, contd.

xChess instance		Pure CNF Encoding				Dual Encoding
name	T/F	Sempro p	sKizzo	Quaffle	Cond-Quaffle	Duaffle (without learning!)
conf1a	T	627	83	--	161	**1.8**
conf1b	F	682	176	2939	124	**1.3**
conf1c	T	659	804	--	156	**2.1**
conf1d	F	706	1930	1473	148	**2.2**
conf2a	T	--	--	--	438	**65.9**
conf2b	F	--	--	--	275	**56.9**
conf3a	T	--	memout	--	653	**5.2**
conf3b	F	--	--	2128	327	**2.2**
conf4	F	--	--	--	274	**32.0**
conf5	F	1018	427	142	11	**0.1**

7-9 quantifier levels

Duaffle (even without learning) on the dual encoding dramatically outperforms all leading CNF-based QBF solvers on these challenging instances

Summary

SAT progress →
Path **from 100 var instances** (early 90's) to **1,000,000+ var instances** (current).

Still moving forward …

Capturing and exploiting structure is key when dealing with Large Real-world instances:

connections between heavy-tails, backdoor sets, randomization, and restarts.

streamlining and xor streamlining

Beyond satisfaction / New applications:

Model counting and **Quantification.**

The End

www.cs.cornell.edu/gomes