Compute-Intensive Methods in AI: New Opportunities for Reasoning and Search

Bart Selman

Cornell University
selman@cs.cornell.edu

Introduction

In recent years, we've seen substantial progress in propositional reasoning and search methods.

Boolean satisfiability testing:

1990: 100 variables / 200 clauses (constraints)

1998: 10,000 - 100,000 vars / 10^6 clauses

Novel applications:

e.g. in planning, software / circuit testing, machine learning, and protein folding

Factors in Progress

a) **new algorithms**

  e.g. stochastic methods

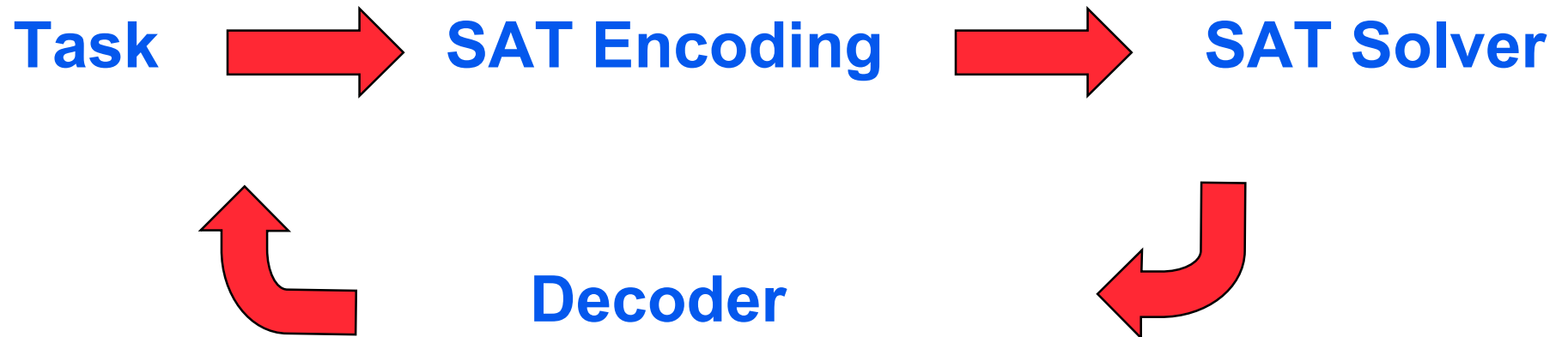b) **better implementations**

  several competitions ---

  Germany 91 / China 96 / DIMACS-93/97/98

c) **faster hardware**

Also, close interplay between theoretical, experimental, and applied work.

Applications: Methodology

Combinatorial
Task ➡ SAT Encoding ➡ SAT Solver

Decoder

Shift work to "encoding phase",
   use fast, off-the-shelf SAT solver and tools.

Compare methodology to the use of
Linear / Integer Programming packages:

--- Emphasis is on mathematical modeling
(e.g. using primal and dual formulations).

--- After modeling phase, problem is handed to a
state-of-the-art solver.

Would specialized solver not be better?

Perhaps theoretically, but often not in practice

--- It's difficult to duplicate efforts put in designing fast solvers.

--- Encodings can compensate for much of the loss due to going to a uniform representation formalism (e.g. SAT, CSP, LP, or MIP).

Outline

I --- **Example application: AI Planning**

                      The SATPLAN system

II --- **Current Themes in SAT Solvers**

                      randomization / scalability

III --- **Current Themes in SAT Encodings**

                      declarative control knowledge

IV --- **Conclusions**

I. Example Application: Planning

Planning: find a (partially) ordered set of actions that transform a given initial state to a specified goal state.

- in most general case, can cover most forms of problem solving

- special case of program synthesis

- scheduling: fixes set of actions, need to find optimal total ordering

    - planning problems typically highly non-linear, require combinatorial search

Some Applications of Planning

Autonomous systems

> **Deep Space One Remote Agent (NASA)**
>
> **mission planning**

Softbots - software robots

- **Internet agents, program assistants**
- **AI "characters" in games, entertainment**

Synthesis, bug-finding (goal = undesirable state), …

Supply Chain Management --- "just-in-time" manufacturing (SAP, I2, PeopleSoft etc. $10 billion)

Proof planning in mathematical domains (Melis 1998)

State-space Planning

Find a sequence of operators that transform an initial state to a goal state

State = complete truth assignment to a set of variables (fluents)

Goal = partial truth assignment (set of states)

Operator = a partial function State → State

specified by three sets of variables:

precondition, add list, delete list

(STRIPS-style, Nilsson & Fikes 1971)

Abdundance of Negative Complexity Results

I. Domain-independent planning: PSPACE-complete or worse

    (Chapman 1987; Bylander 1991; Backstrom 1993)

II. Domain-dependent planning: NP-complete or worse

    (Chenoweth 1991; Gupta and Nau 1992)

III. Approximate planning: NP-complete or worse

    (Selman 1994)

Practice

Traditional domain-independent planners can generate plans of only a few steps.

> **Prodigy, Nonlin, UCPOP, ...**

Practical systems minimize or eliminate search by employing

> **complex search control rules, hand-tailored to the search engine and the particular search space**
> **(Sacerdoti 1975, Slaney 1996, Bacchus 1996)**

> **pre-compiling entire state-space to a reactive finite-state machine**
> **(Agre & Chapman 1997, Williams & Nayak 1997)**

Scaling remains problematic when state space is large or not well understood!

13

Progression

- **Planning as first-order theorem proving**
  (Green 1969)

  computationally infeasible

- **STRIPS** **(Fikes & Nilsson 1971)**

  very hard

- **Partial-order planning**
  (Tate 1977, McAllester 1991, Smith & Peot 1993)

  can be more efficient, but still hard
  (Minton, Bresina, & Drummond 1994)
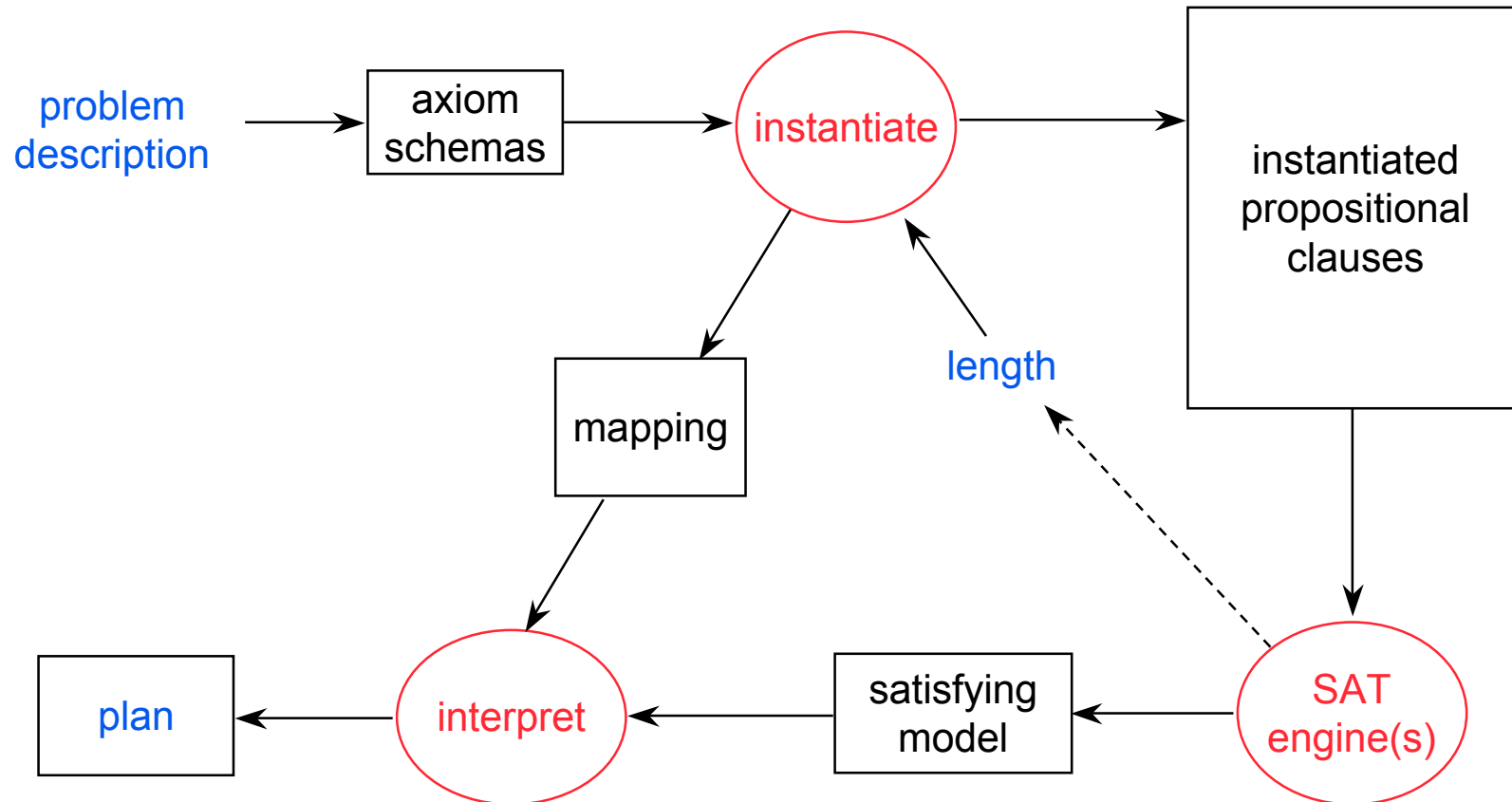
- **Proposal: planning as propositional reasoning**

14

Approach

SAT encodings are designed so that plans correspond to satisfying assignments

Use recent efficient satisfiability procedures (systematic and stochastic) to solve

Evaluation performance on benchmark instances

SATPLAN

SAT Encodings

Propositional CNF: no variables or quantifiers

Sets of clauses specified by axiom schemas

fully instantiated before problem-solving

Discrete time, modeled by integers

state predicates: indexed by time at which they hold

action predicates: indexed by time at which action begins

each action takes 1 time step

many actions may occur at the same step

fly(Plane, City1, City2, i) ⊃ at(Plane, City2, i +1)

Solution to a Planning Problem

A solution is specified by any model (satisfying truth assignment) of the conjunction of the axioms describing the initial state, goal state, and operators

Easy to convert back to a STRIPS-style plan

Satisfiability Testing Procedures

Systematic, complete procedures

> **Depth-first backtrack search**
> **(Davis, Putnam, & Loveland 1961)**
>
> > unit propagation, shortest clause heuristic
>
> **State-of-the-art implementation: ntab**
> **(Crawford & Auton 1997)**
>
> > and many others!   *See SATLIB 1998 / Hoos & Stutzle.*

Stochastic, incomplete procedures

> **GSAT (Selman et. al 1993)**
>
> **Current fastest: Walksat (Selman & Kautz 1993)**
>
> > greedy local search + noise to escape local minima

19

Walksat Procedure

Start with random initial assignment.

Pick a random unsatisfied clause.

Select and flip a variable from that clause:

With probability p, pick a random variable.

With probability 1-p, pick greedily

a variable that minimizes the number of unsatisfied clauses

Repeat to predefined maximum number flips; if no solution found, restart.

Planning Benchmark Test Set

Extension of Graphplan benchmark set

> Graphplan **faster** than **UCPOP** (Weld 1992) **and Prodigy** (Carbonell 1992) **on blocks world and rocket domains**

logistics - complex, highly-parallel transportation domain, ranging up to

> **14 time slots, unlimited parallelism**

> **2,165 possible actions per time slot**

> **optimal solutions containing 150 distinct actions**

Problems of this size (10^18 configurations) not previously handled by any state-space planning system

Solution of Logistics Problems
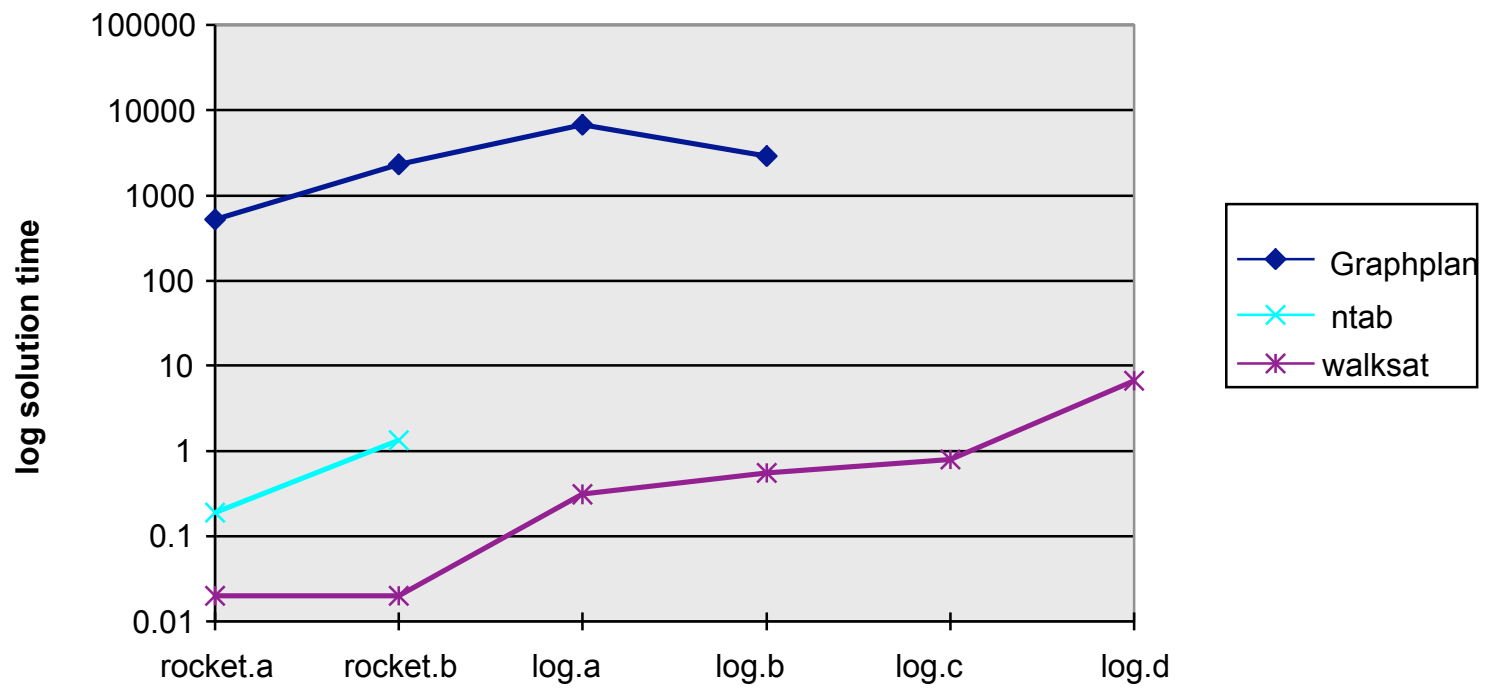
What SATPLAN Shows

A **general** propositional theorem prover can be **competitive** with **specialized** planning systems

Surpise:

"Search direction" does not appear to matter. (Traditional planners generally

backward chain from goal state.)

Fast SAT engines

stochastic search - walksat

large SAT/CSP community sharing ideas and code

specialized engines can catch up, but by then, new general technique

II. Current Themes in Sat Solvers

SAT Solvers

Stochastic local search solvers (walksat)

    when they work, scale well

    cannot show unsat

    fail on certain domains

    must use very simple (fast) heuristics

Systematic solvers (Davis Putnam Loveland style)

    complete

    fail on (often different) domains

    might use more sophisticated (costly) heuristics

    often to scale badly

Can we combine best features of each approach?

Background

Combinatorial search methods often exhibit a remarkable **variability** in performance. It is common to observe significant differences between:

- different heuristics
- same heuristic on different instances
- different runs of same heuristic with different seeds (stochastic methods)

Preview of Strategy

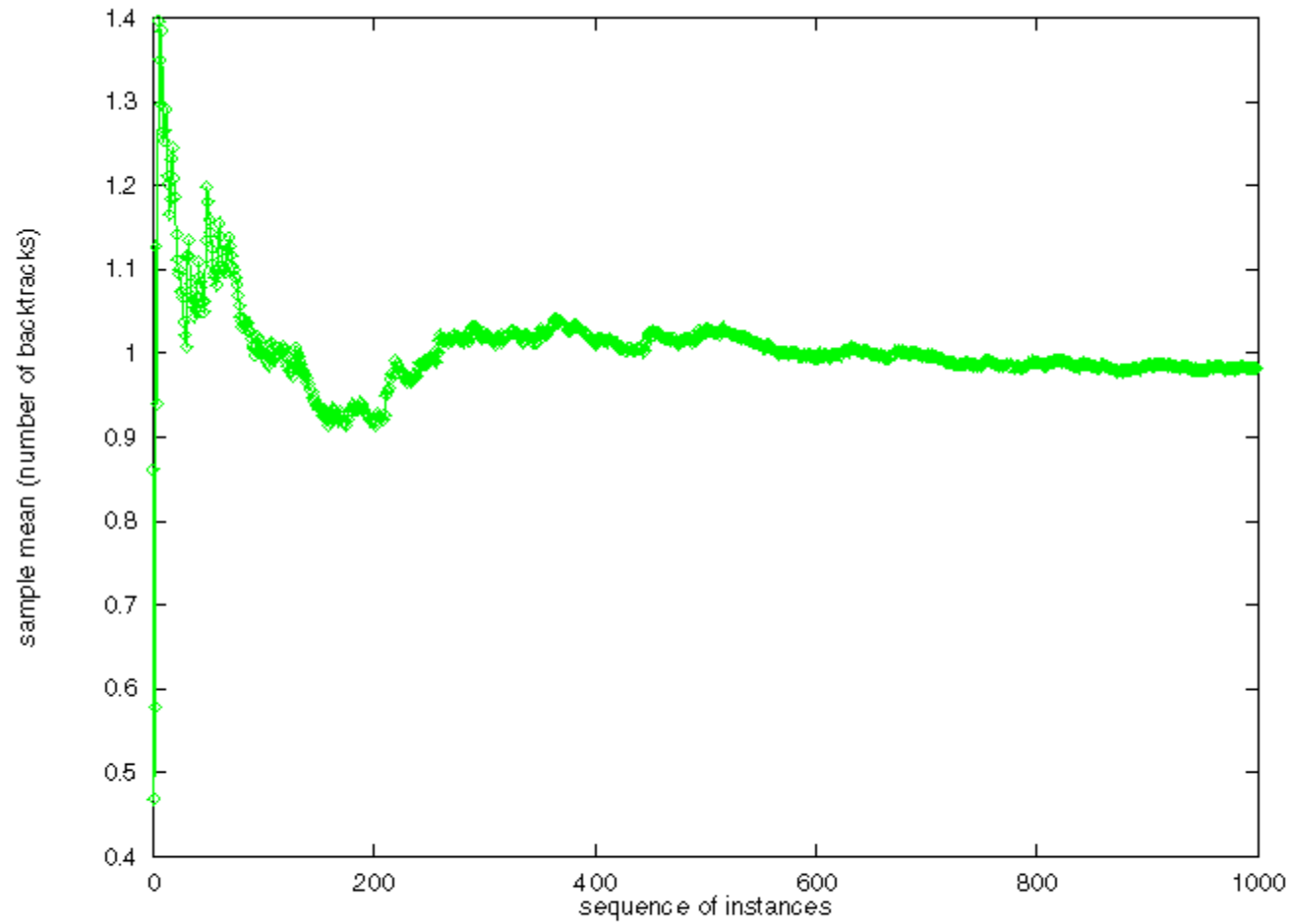We'll put variability / unpredictability to our advantage via randomization / averaging.

Cost Distributions

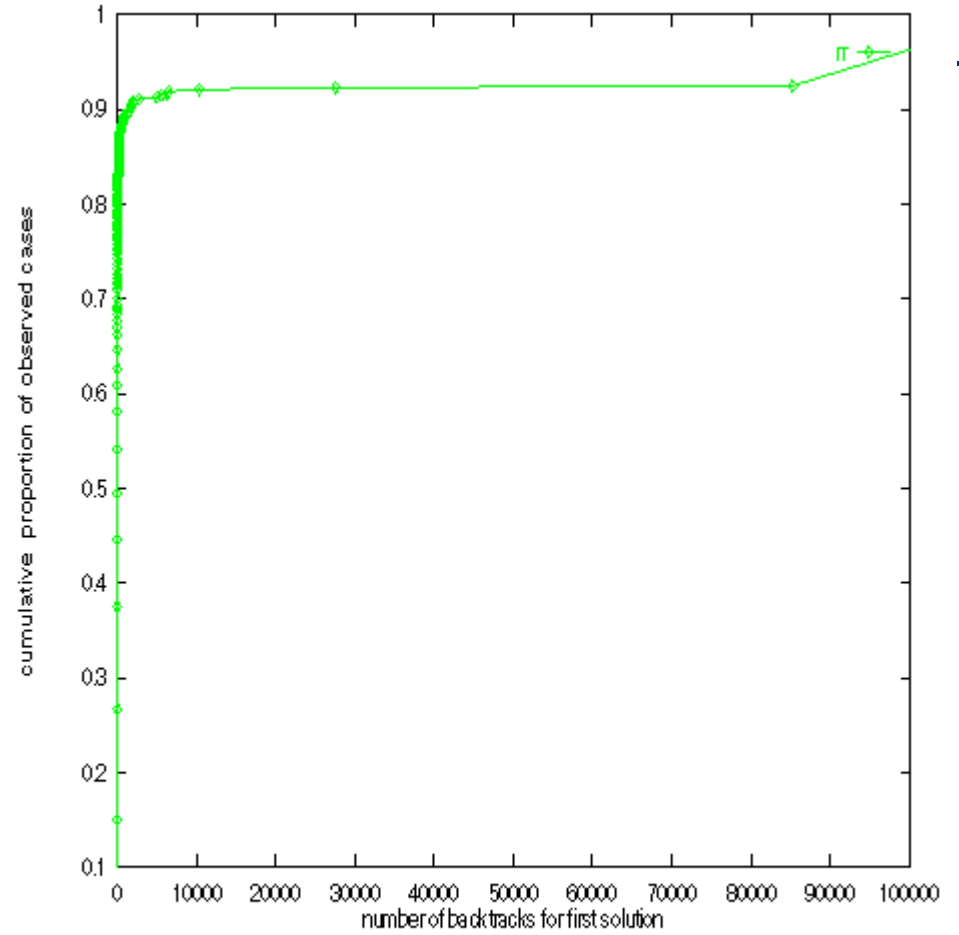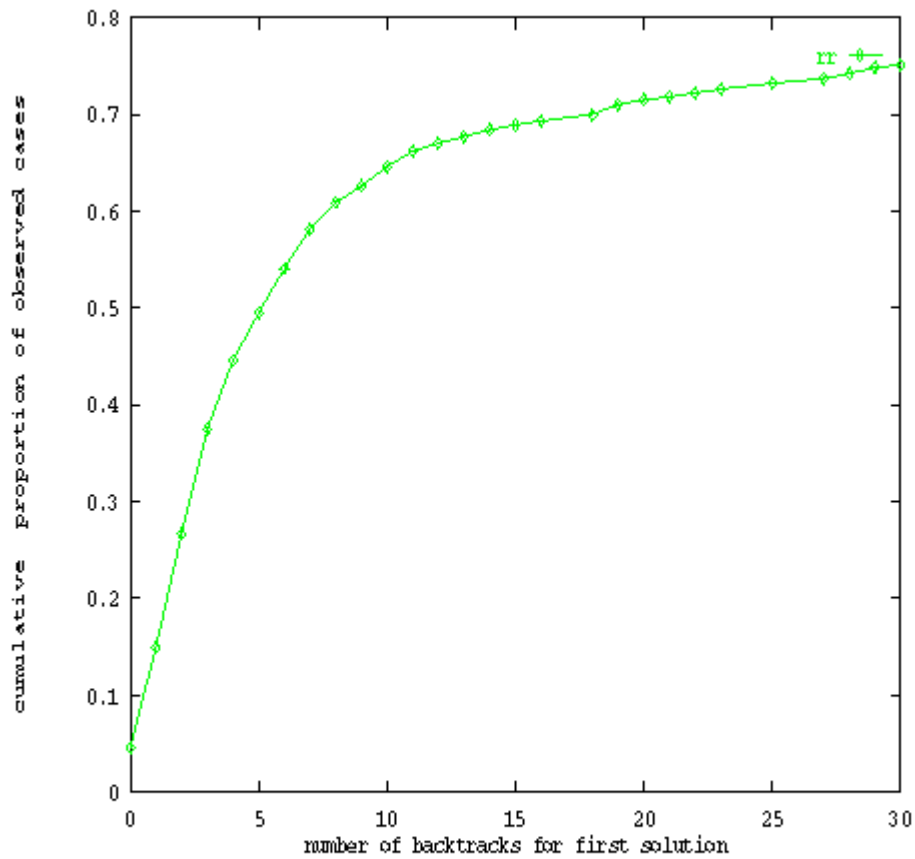Backtrack-style search (e.g. Davis-Putnam) characterized by:

I   Erratic behavior of mean.

II Distributions have "heavy tails".
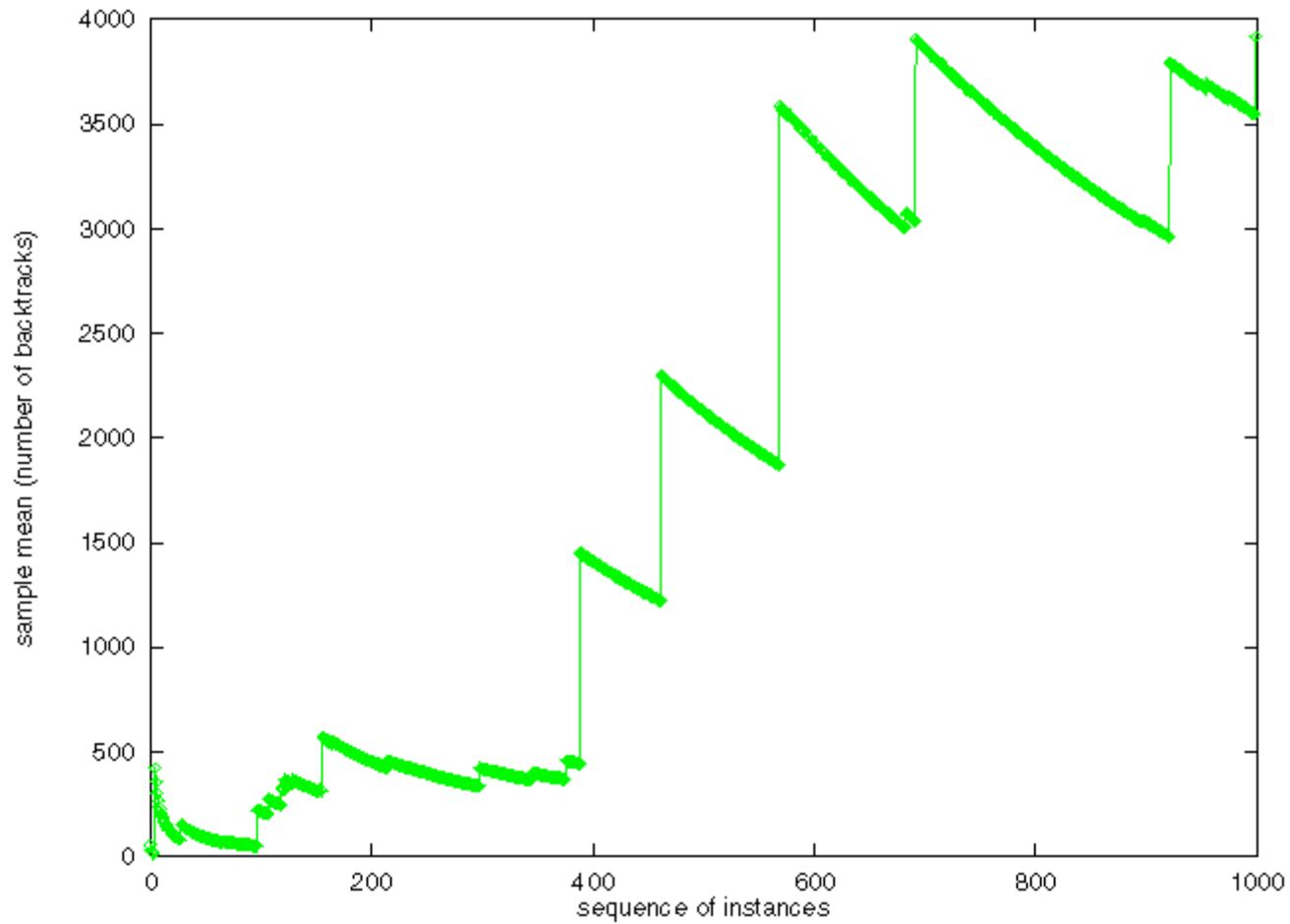
Erratic Mean Cost Behavior

Standard Mean Cost Behavior (Gamma)

Heavy-Tailed Behavior

Heavy-Tailed Distributions

… infinite variance … infinite mean

Introduced by Pareto in the 1920's
--- "probabilistic curiosity."

Mandelbrot established the use of heavy-tailed distributions to model real-world fractal phenomena.

Examples: stock-market, earth-quakes, weather,...

Decay of Distributions
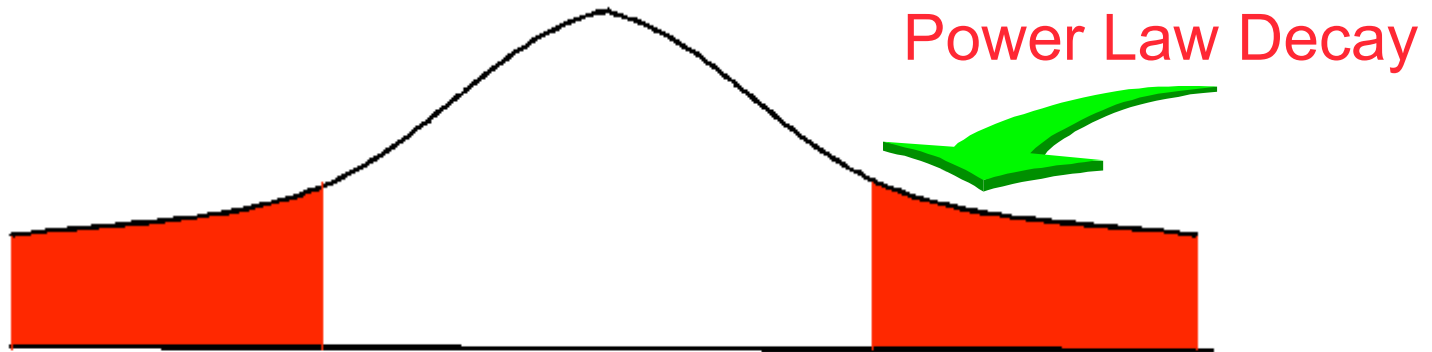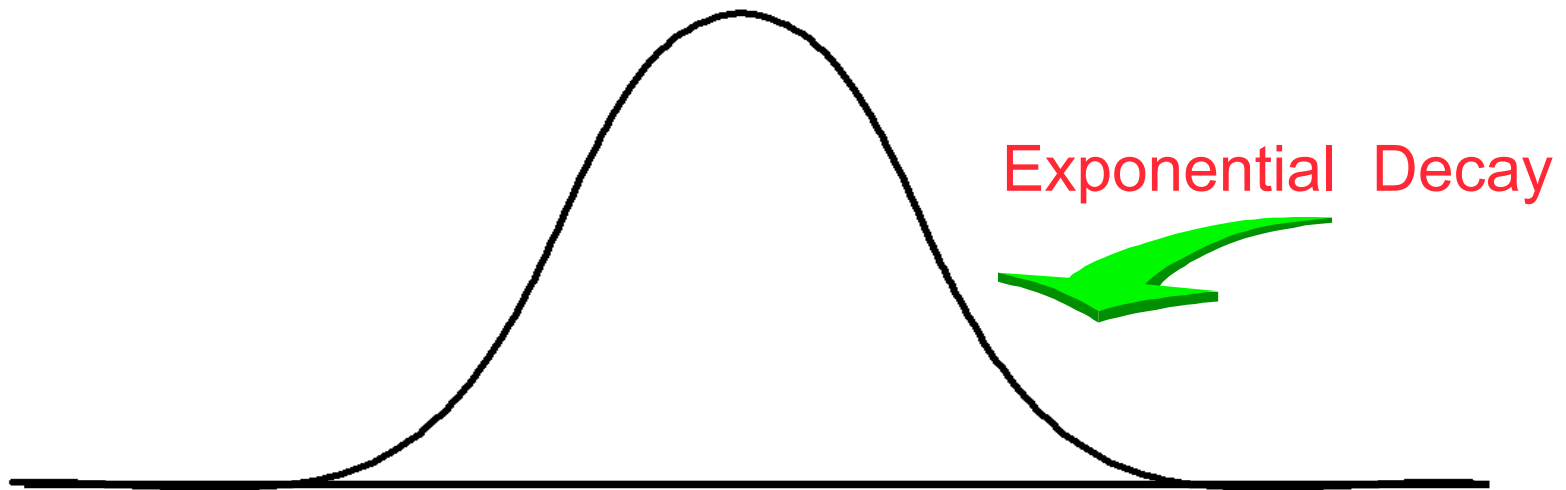
Standard --- Exponential Decay

   e.g. Normal:

$$P_N(|X_n - \mu|) \sim K e^{-n^2}, \quad \text{[illegible]}$$

Heavy-Tailed --- Power Law Decay

   e.g. Pareto-Levy:

$$P_H(|X_n - \mu|) \sim K n^{-\alpha}, \quad \text{[illegible]}$$

Power Law Decay

HEAVY TAILED DISTRIBUTION
(infinite mean & variance)

Exponential Decay

Standard Distribution
(finite mean & variance)

34

How to Check for "Heavy Tails"?

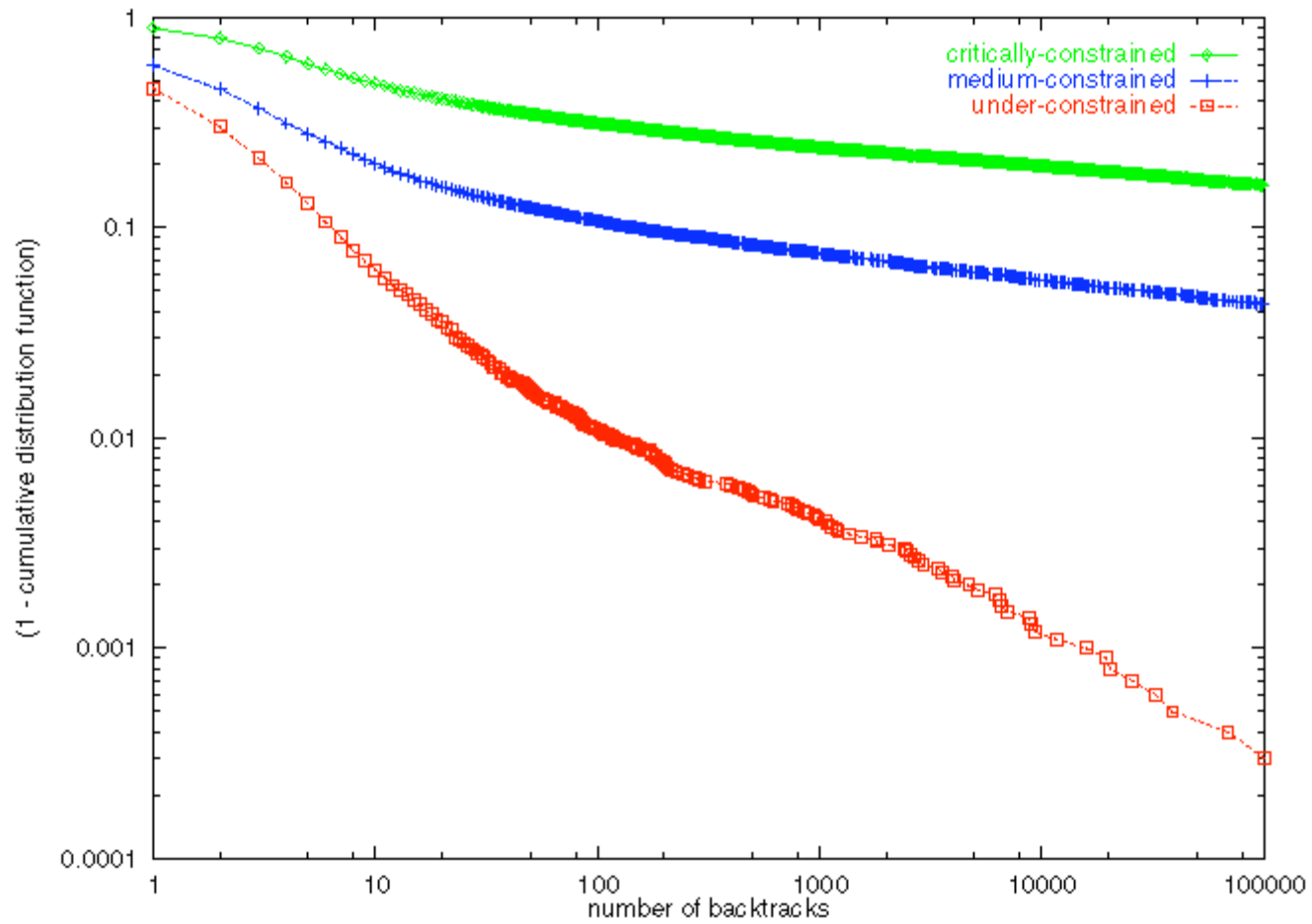**Log-Log plot of tail of distribution should be approximately linear.**
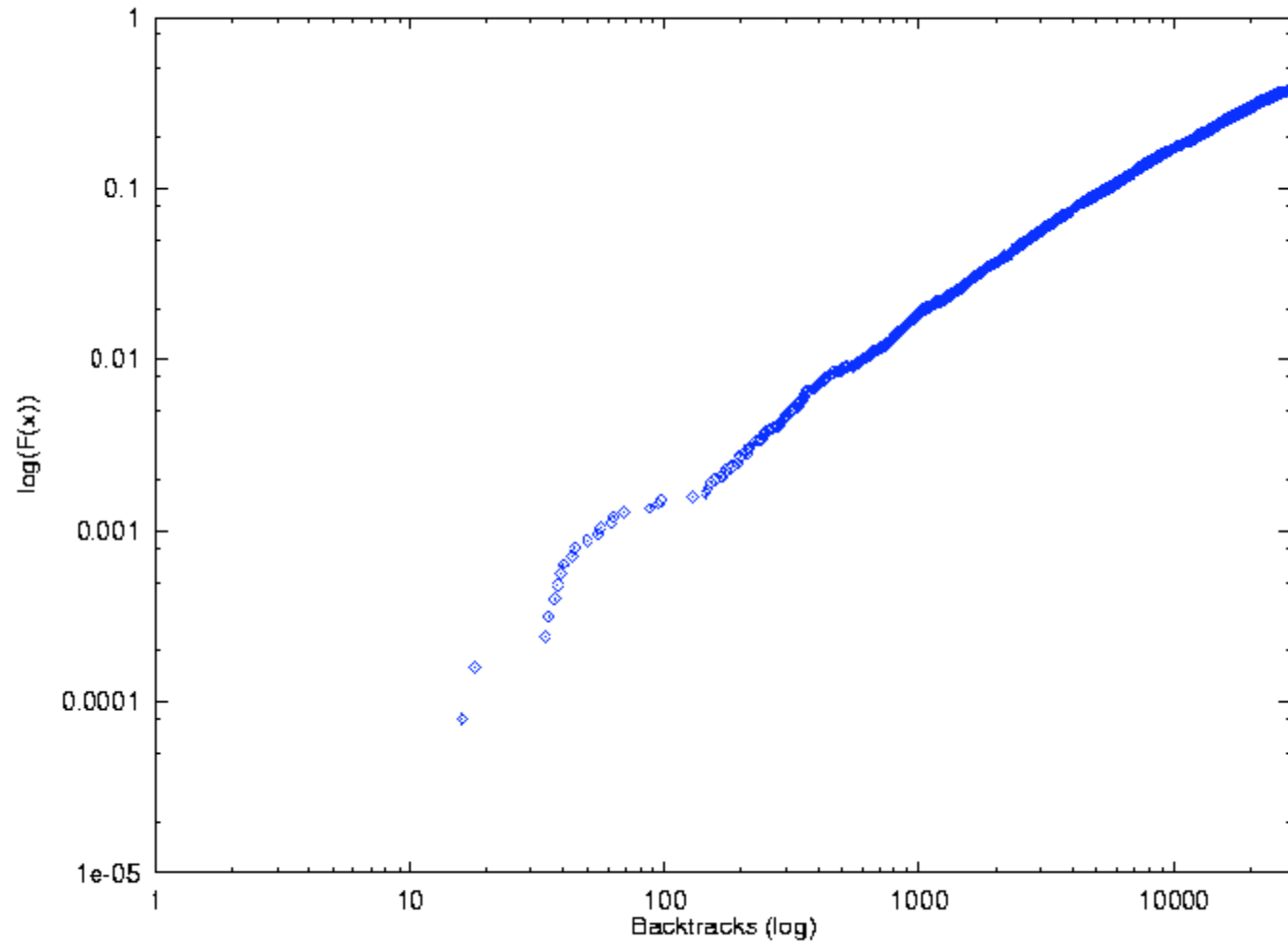
Slope gives value of ■

■ ■ ▮   **infinite mean and infinite variance**

▮ ■ ■ ■ ▮   **infinite variance**

Heavy-Tailed Behavior (log-log scale)

37

Heavy Tails

Bad scaling of systematic solvers can be caused by heavy tailed distributions

Deterministic algorithms get stuck on particular instances

> *but that same instance might be easy for a different deterministic algorithm!*

Expected (mean) solution time increases without limit over large distributions

Randomized Restarts

Solution: randomize the systematic solver

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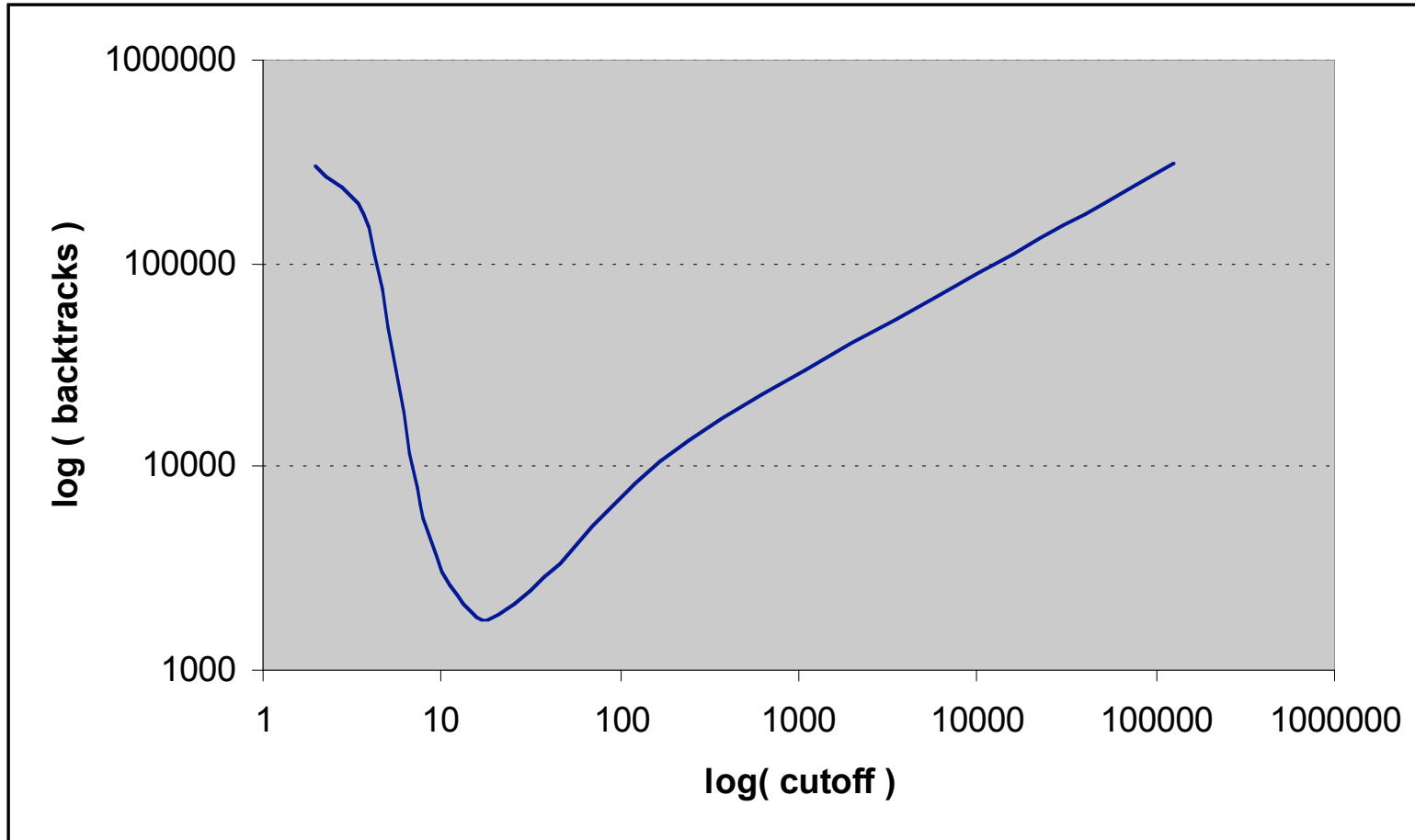
Cutoff and **restart** search after a fixed number of backtracks

Provably Eliminates heavy tails

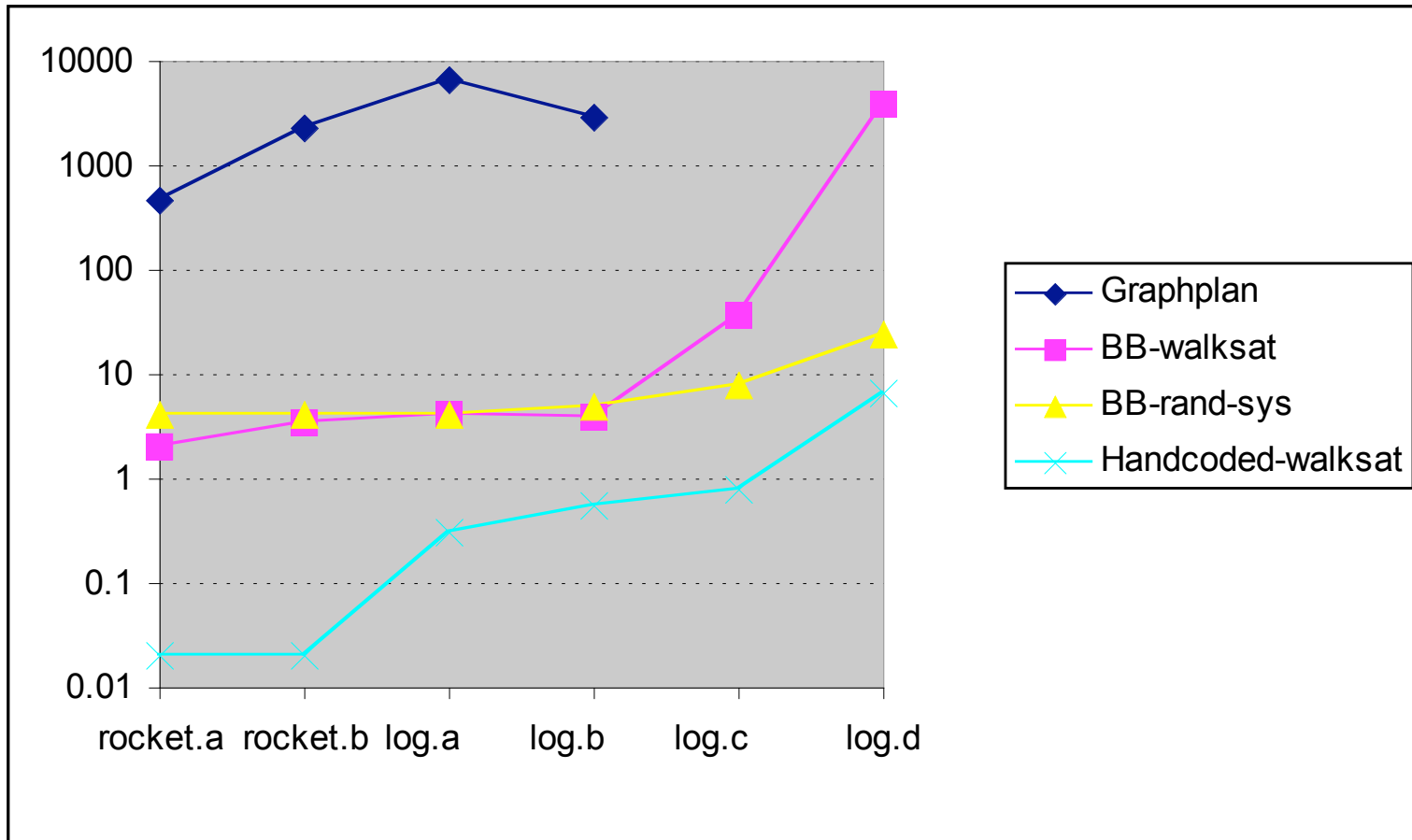*In practice: rapid restarts with low cutoff can dramatically improve performance*

(Gomes and Selman 1997, 1998)

Rapid Restart on LOG.D

Note Log Scale: Exponential speedup!

40

SATPLAN Results

Overall insight:

Randomized tie-breaking with rapid restarts gives powerful bactrack-style search strategy. (sequential / interleaved / parallel)

Related analysis: Luby & Zuckerman 1993; Alt & Karp 1996.

42

Heavy-Tailed Distributions
in Other Domains

Quasigroup Completion Problem

Graph Coloring

Logistic Planning

Circuit Synthesis

Gomes, Selman, and Crato 1997 - Proc. CP97;

Gomes, Selman, McAloon, and Tretkoff 1998 - Proc AIPS98;

Gomes, Kautz, and Selman 1998 - Proc. AAAI98.

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		

Gomes, Kautz, Selman 1998

SAT Solvers: Themes, cont.

Randomization (as discussed)

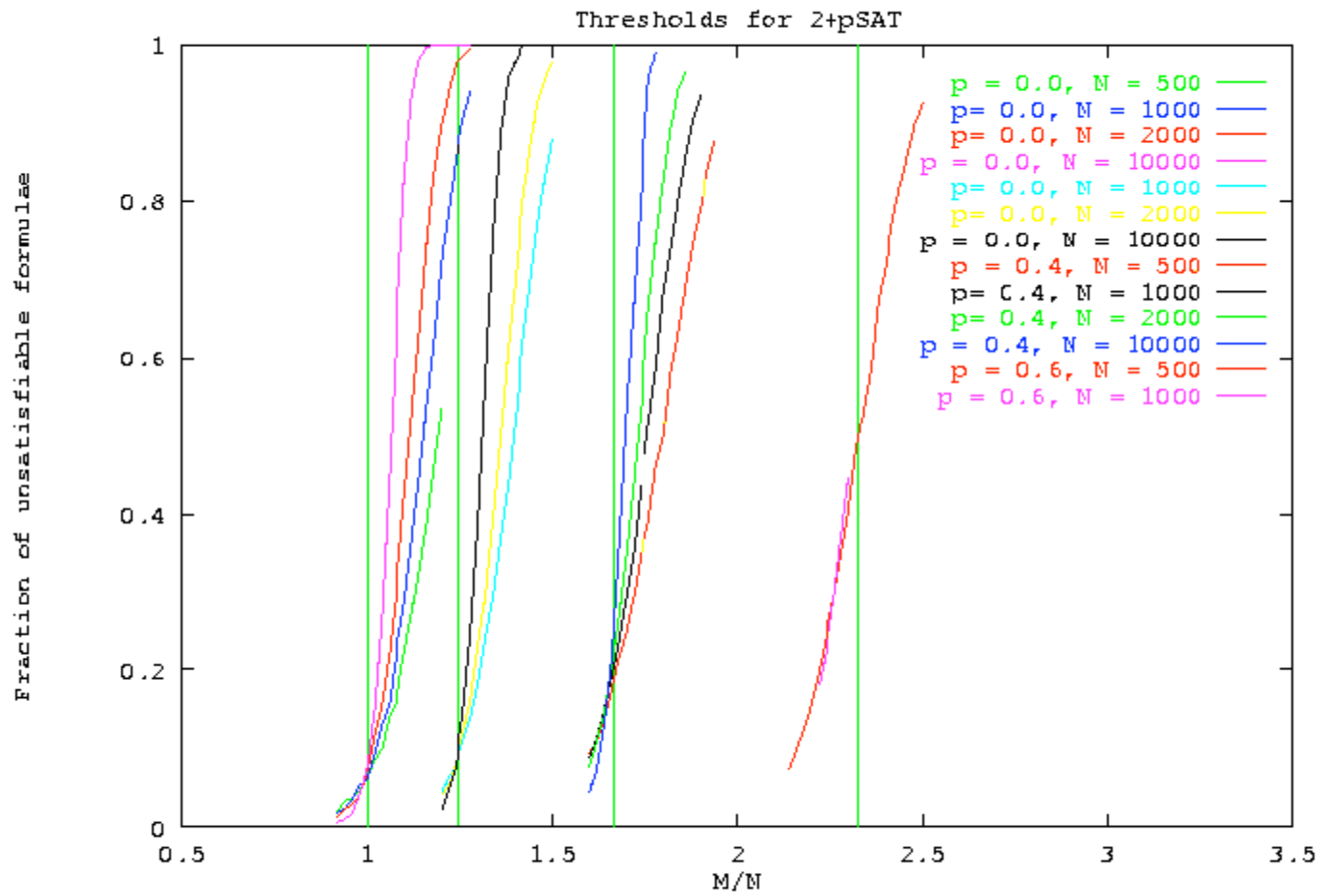**Hybrid solvers --- Algorithm Portfolios**

> **(Hogg & Hubermann 1997; Gomes & Selman 1997)**
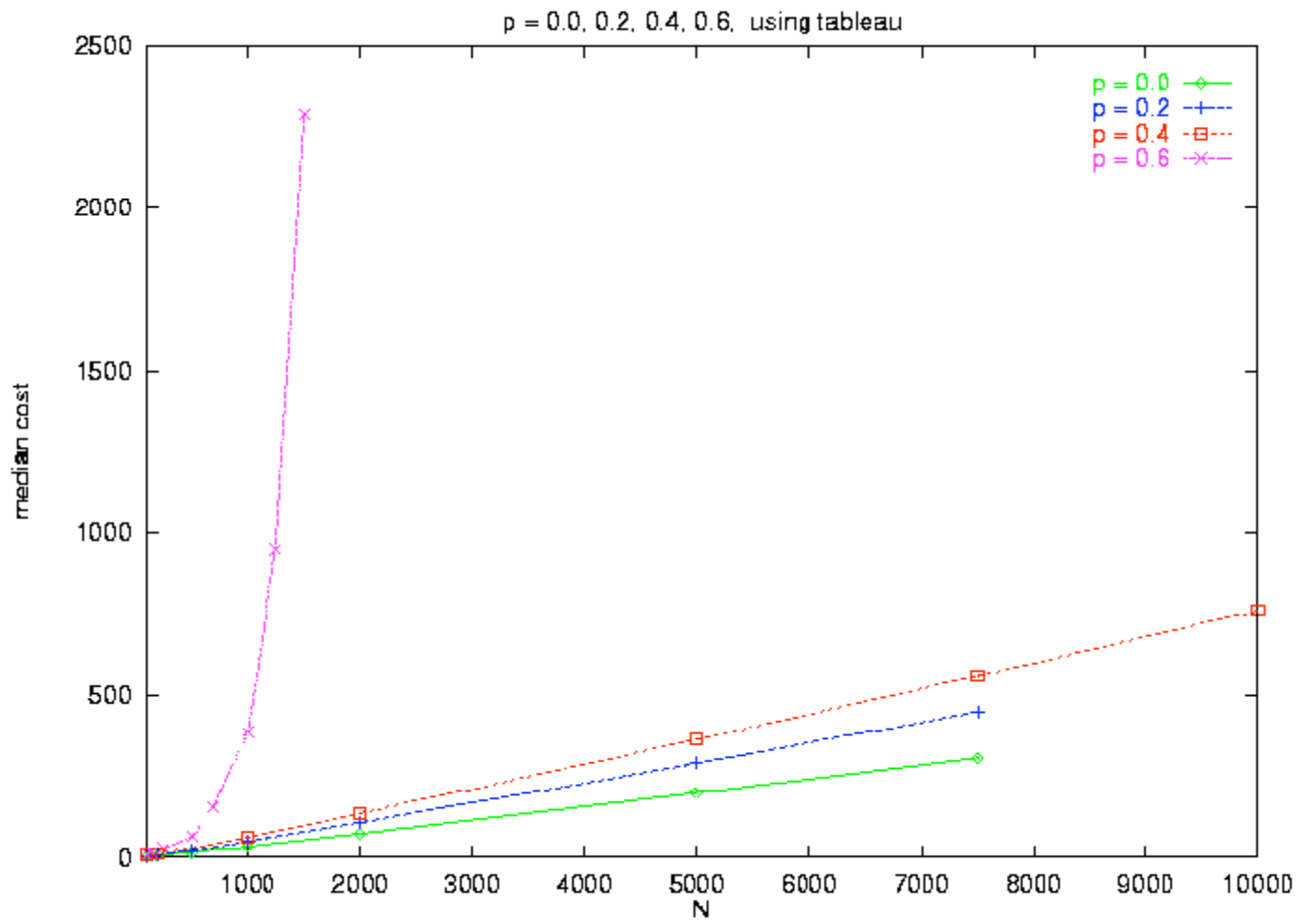
Using LP relaxations (Warners & van Maaren 1998)

Between 2SAT / 3SAT:

> **Mixture can behave as pure 2SAT!**
>
> **(Kirkpatrick, Selman, et al. 1996 / 1998)**

Thresholds for 2+pSAT

p = 0.0, 0.2, 0.4, 0.6, using tableau

47

SAT Solvers: Recent Theory

Minimal size of search tree

   (Beame, Karp, et al. 1998)

Better worst-case: less than O(2^n)

  backtrack style: O(2^(0.387n))

   (Schiermeyer 1997; Paturi, et al. 1998)

  local search: O(2^(c.n)) with c < 1

  (Hirsch, 1998)

48

IV. Current Themes in Encodings

Add Declarative Domain Knowledge

Efficient representations and (randomized) SAT engines extend the range of domain-independent planning

Ways for further **improvement:**

Better general search algorithms

Incorporate (more) domain dependent knowledge

Kinds of Knowledge

* About domain itself

> *a truck is only in one location*
>
> *airplanes are always at some airport*

* About good plans

> *do not remove a package from its destination location*
>
> *do not unload a package and immediate load it again*

X About how to search

> *plan air routes before land routes*
>
> *work on hardest goals first*

Expressing Knowledge

Such information is traditionally incorporated in the planning algorithm itself

or in a special programming language

Instead: use additional declarative axioms

Problem instance: operator axioms + initial and goal axioms + heuristic axioms

Domain knowledge ≈ constraints on search and solution spaces

Independent of any search engine strategy

Logical Status of Heuristics

1. **Entailed by operator axioms: conflicts and derived effects**

   fly(plane,d1,i) and fly(plane,d2,i) conflict

2. **Entailed by operators + initial state axioms: state invariants**

   a truck is at only one location

3. **Entailed by operators + initial + goal + length: optimality conditions**

   do not return a package to a location

4. **New constraints on problem instance: simplifying assumptions**

   Once a truck is loaded, it should immediately move

Axiomatic Form

Invariant: A truck is at only one location

at(truck,loc1,i) & loc1 ≠ loc2 ⊃
¬ at(truck,loc2,i)

Optimality: Do not return a package to a location

at(pkg,loc,i) & ¬ at(pkg,loc,i+1) & i<j ⊃
¬ at(pkg,loc,j)

Simplifying: Once a truck is loaded, it should immediately move

¬ in(pkg,truck,i) & in(pkg,truck,i+1) &
at(truck,loc,i+1) ⊃
¬ at(truck,loc,i+2)

Questions

Does it work?

> **Additional axioms might just blow up instance with redundant information**

Is effect independent of search engine?

Can we predict the most useful level of heuristic axioms?

> **What is relation of difficulty to problem size?**

Experiment: Logistics

h1: Optimality conditions

> *Once a package leaves a location, it never returns*

h2, h3: Simplifying assumptions

> *A package is never in any city other than its origin or destination cities*
>
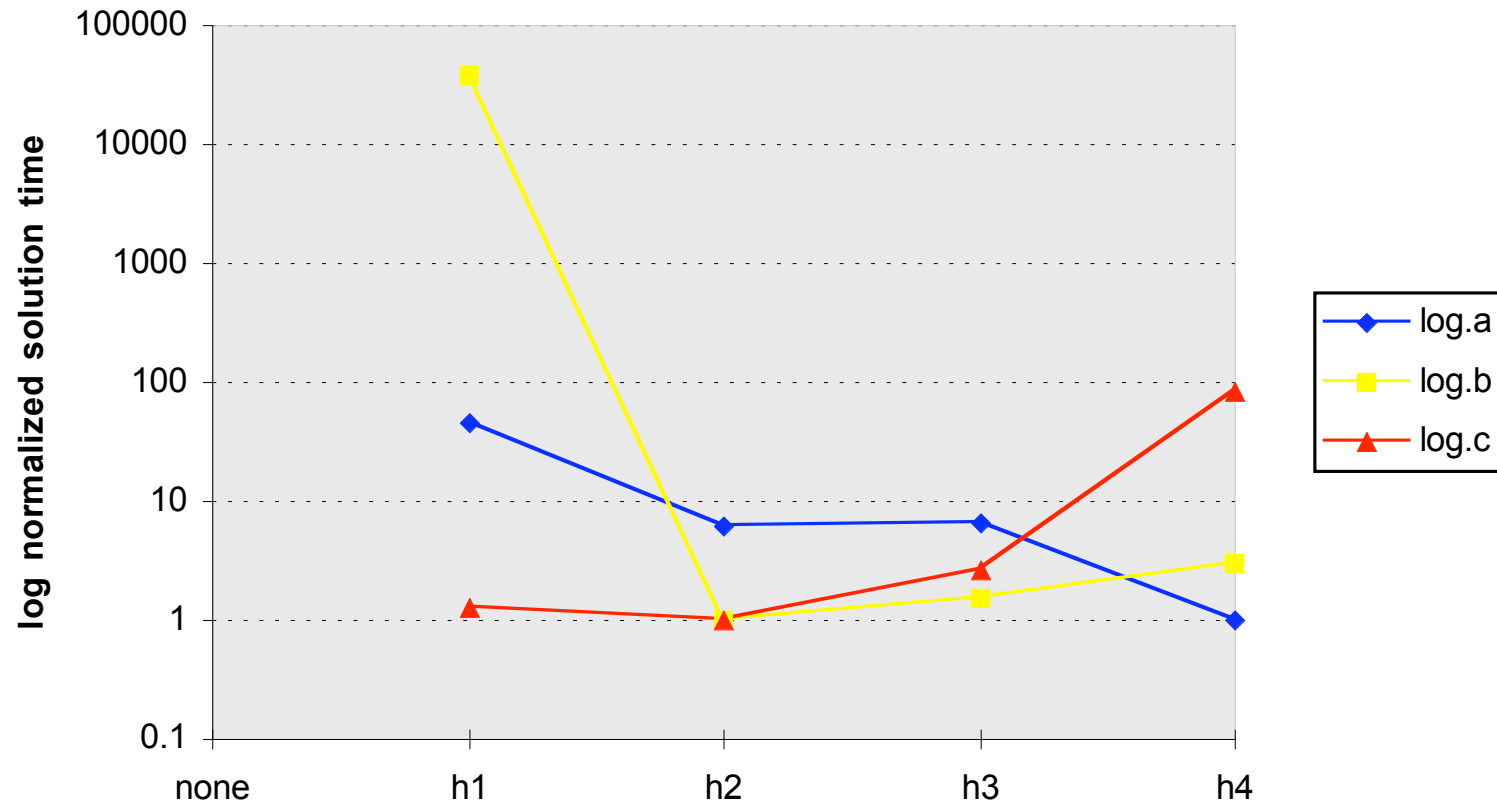> > **rules out solutions where packages are transferred between airplanes in an intermediate city**
>
> *Once a vehicle is loaded, it should immediately move*
>
> > **rules out solutions where vehicles are loaded incrementally**

h4: More optimality conditions

> *A package never leaves its destination city*

ntab solution of logistics

Answers

Does it work?

> **YES, 10 to 100+ times speedup**

Is effect independent of search engine?

> **YES, same heuristics best for systematic and stochastic engines --- but needs more investigation**

Can we predict the most useful level of heuristic axioms?

> **USUALLY point at which problem size is minimized after simplification by unit propagation (40% - 70% reduction)**

How to Generate Control Knowledge --- Automatically

Polytime preprocessing

> Try to add "obvious" inferences (McAllester, Crawford)

Compilation

> Fix operators and initial or goal state, generate tractable equivalent theory (Kautz & Selman)

Learning strategies (Minton, Kambhampati, Etzioni, Weld, Smith)

Use automatic type inference to derive invariants.

> (Fox & Long --- STAN system 1998; Rintanen 1998;
> Koehler & Nebel --- IPP system 1998)

Encodings: Themes cont.

Add declarative control knowledge (as discussed)

Robustness

> Small change in original formulation, small change in encoding.

Add numeric information / "soft constraints"

> Weighted MAXSAT?

More compact encodings.

> E.g. causal.

Conclusions

Discussed current state-of-the-art in propositional reasoning and search.

Shift to 10,000+ variables and 10^6 clauses has opened up new applications.

Methodology: *Find compact SAT encoding;*
Use off-the-shelf SAT Solver.
Analogous to LP and MIP approaches.

Conclusions, cont.

Example: AI planning / SATPLAN system

One order of magnitude improvement (last 3yrs):

10 step to 200 step plans

Need two more:

up to 20,000 step ...

Discussed themes in SAT Sovers / Encodings

Heavy-tails / Randomization / Declarative domain knowledge

62