Random 3-SAT and BDDs: The Plot Thickens Further

Alfonso San Miguel Aguirre

Moshe Y. Vardi

Rice University

Pareboy

- Early CS in Russia, 1950's:
  some problems can be solved only via exhaustive search - research program unsuccessful.

- Cook - Levin: NP-completeness, 1970's
  explanation via reduction, worst-case complexity.

- Levin - Gurevich: avg.-case NP-completeness 1980's
  not very successful (only a few problems amenable).

Where the Really Hard Problems Are

Cheeseman + Kanefsky + Taylor, 1991:

- "Worst-case complexity not always useful"

- Instances of NP-complete problems can be quite easy in practice

- Decision problems can be often characterized in terms of "constrainedness"

- The hard problems are those that lie in the transition from underconstrained to overconstrained.

Example: 3-SAT

clause: $\pm x_{i_1} \vee \pm x_{i_2} \vee \pm x_{i_3}$

Instance: $\bigwedge_{i=1}^{m} C_i$

order: # of vars

density: # of clauses / # of vars

Random 3-SAT:
- fix order and density
- choose clauses uniformly

Motivation:
- NP-complete
- useful algorithmic tool
- Basic reasoning task

Mitchell + Selman + Levesque, 1992

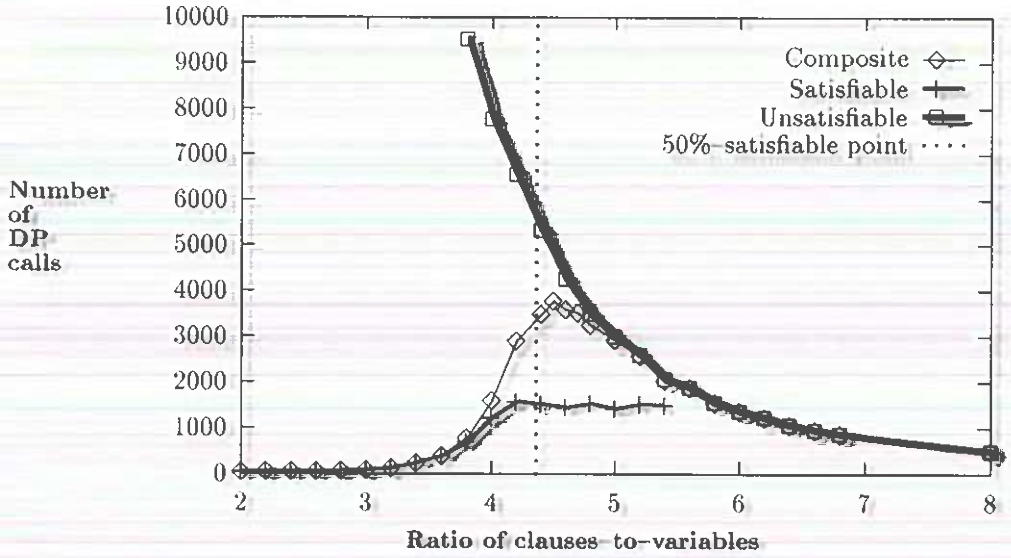


Figure 3: Median DP calls for 50-variable Random 3-SAT as a function of the ratio of clauses-to-variables.

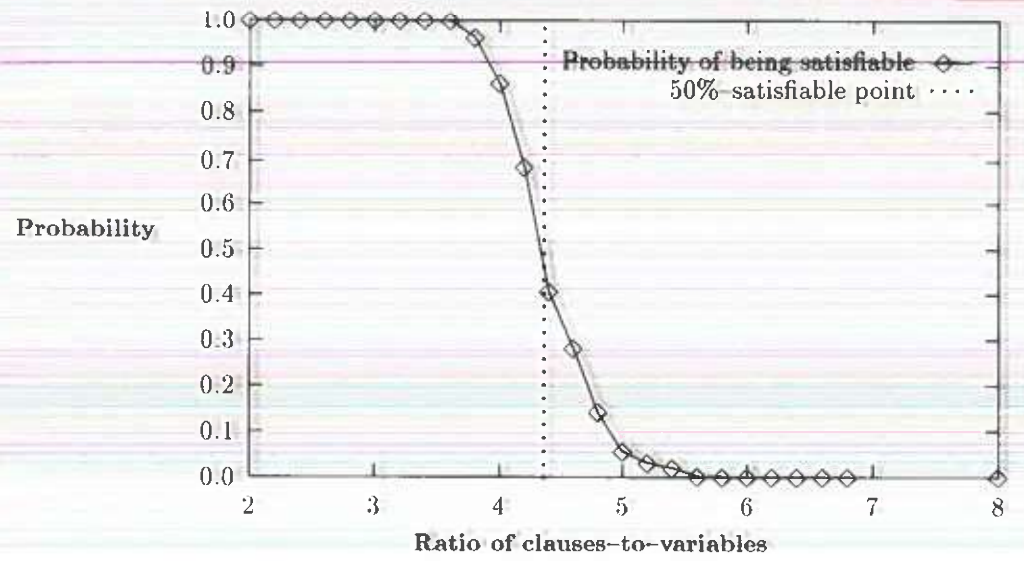


Figure 4: Probability of satisfiability of 50-variable formulas, as a function of the ratio of clauses-to-variables.

easy-hard-easy pattern. The formulas in the hard area appear to be the most challenging for the strategies we have tested, and we conjecture that they will be for every (heuristic) method.

The constant-probability model

We now examine formulas generated using the constant-probability model. The model has three parameters: the number of variables N, and number of clauses L as before; but instead of a fixed clause length, clauses are generated by including a variable in a clause with some fixed probability P, and then negating it with probability 0.5. Large formulas generated this way very often have at least one empty clause and several unit clauses, so that they tend to be either trivially unsatisfiable, or easily shown satisfiable. Thus, the more interesting results are for the modified version in which empty and unit clauses are disallowed. This distribution we call *Random P-SAT*.

Analytic results by Franco and Paull (1983) suggest that one probably cannot generate computationally challenging instances from this model, and our experiments confirm this prediction. In Figure 5, we compare the number of recursive DP calls to solve instances of random P-SAT with the same figures for random 3-SAT. Although we see a slight easy-hard-easy pattern (previously noted by Hooker and Fedjki, 1989), the hard area is not nearly so pronounced, and in absolute terms

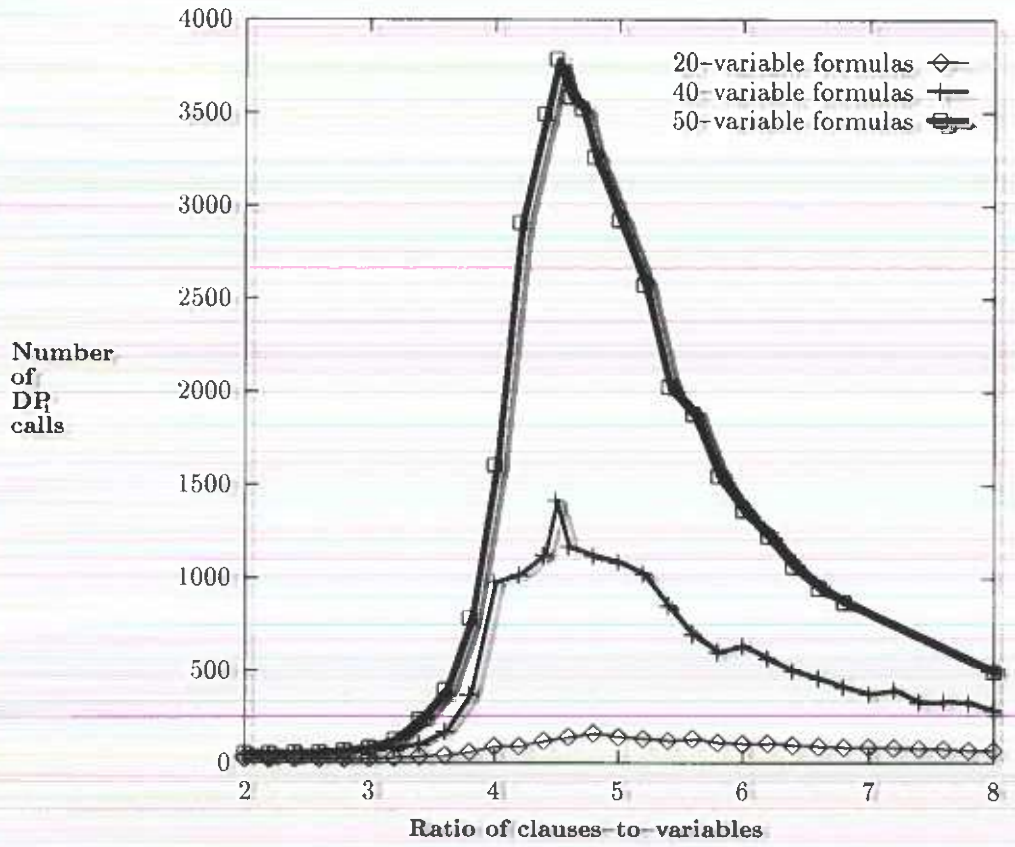


Figure 2: Median number of recursive DP calls for Random 3-SAT formulas, as a function of the ratio of clauses-to-variables.

assignment is likely to be found early in the search. Formulas with very many clauses are *over-constrained* (and usually unsatisfiable), so contradictions are found easily, and a full search can be completed quickly. Finally, formulas in between are much harder because they have relatively few (if any) satisfying assignments, but the empty clause will only be generated after assigning values to many variables, resulting in a deep search tree. Similar under- and over-constrained areas have been found for random instances of other NP-complete problems (see the discussion of the work of Cheeseman *et al.* (1991), below).

The curves in figure 2 are for *all* formulas of a given size, that is they are composites of satisfiable and unsatisfiable subsets. In figure 3 the median number of calls for 50-variable formulas is factored into satisfiable and unsatisfiable cases, showing that the two sets are quite different. The extremely rare unsatisfiable short formulas are very hard, whereas the rare long satisfiable formulas remain moderately difficult. Thus, the easy parts of the composite distribution appear to be a consequence of a relative abundance of short satisfiable formulas or long unsatisfiable ones.

To understand the hard area in terms of the likelihood of satisfiability, we experimentally determined the probability that a random 50-variable instance is satisfiable (figure 4). There is a remarkable correspondence between the peak on our recursive calls curve and the point where the probability that a formula is satisfiable is 0.5. The main empirical conclusion we draw from this is that *the hardest area for satisfiability is near the point where 50% of the formulas are satisfiable.*

This "50% satisfiable" point seems to occur at a fixed ratio of the number of clauses to the number of variables: when the number of clauses is about 4.3 times the number of variables. There is a boundary effect for small formulas: for formulas with 20 variables, the point occurs at 4.55; for 50 variables, at 4.31; and for 140 variables, at 4.3. While we conjecture that this ratio approaches about 4.25 for very large numbers of variables, it remains a challenging open problem to *analytically* determine the "50% satisfiable" point as a function of the number of variables.

Finally, note that we did not specify a method for choosing which variable to guess in the "splitting" step of DP. In our implementation, we simply set variables in lexical order (except when there are unit clauses.) DP can be made faster by using clever selection strategies (*e.g.*, Zabih and McAllester 1988), but it seems unlikely that such heuristics will *qualitatively* alter the

Hard and Easy Distribution of 3-SAT

MSL, 1992:

- constrainedness = density
- Crossover point ~ 4.3
  probability of satisfiability drops precipitously
- DLL solver peaks at 4.3

Bottom line: easy-hard-easy

- low density: easy
- high density: easy
- crossover density: hard

A closer look

Coarfa et al., 2000:

- What is meant by "easy" and "hard"?
- Where is the transition between "easy" and "hard"?

Proposed approach:

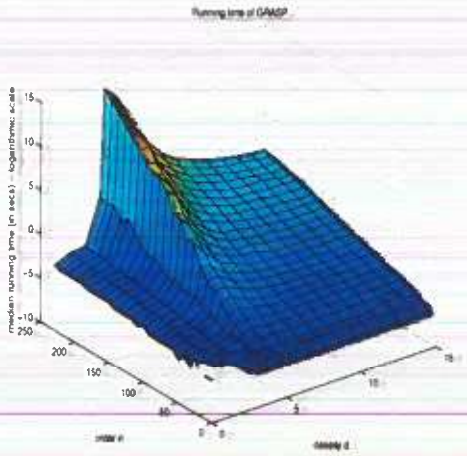
- Focus on scalability
  - + easy = polynomial scalability
  - + hard = exponential scalability
- Fix density, scale order
  - + corresponds to analytical studies
  - + corresponds to applications

Experimental study:

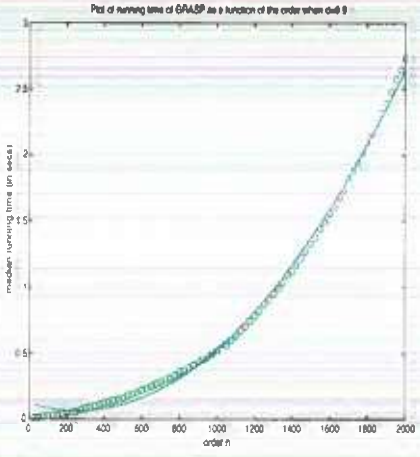
study scalability as function of density.

GRASP

DLL-based learning backjumping



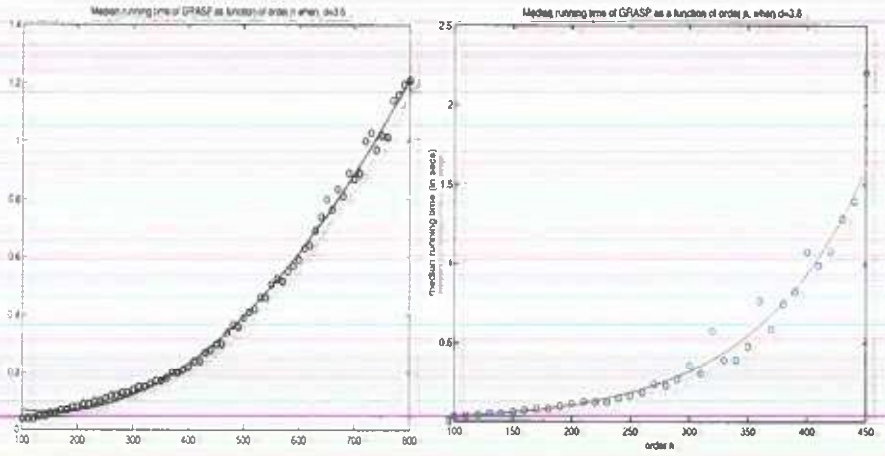
3-D Plot of median running time



Quadratic median running time for density 0.9 as a function of the order

8

GRASP - Poly2Exp



GRASP – median running time for density 3.6 (left) and density 3.8 (right) as a function of the order

Poly2Exp Phase Transition:

- Density 3.6: quadratic running time
- Density 3.8: exponential running time

Evolution of Scalability

GRASP:

- Below density 3.8 : polynomial
- Above density 3.8 : exponential
  - 3.8 - 4.3 : exponent rises
  - > 4.3 : exponent decreases

Bottom line:

- Two phase transitions
- First one (3.8) more significant
- Patten: easy - hard - less hard

Question: what about other SAT solvers?

e.g. BDD based.

Example

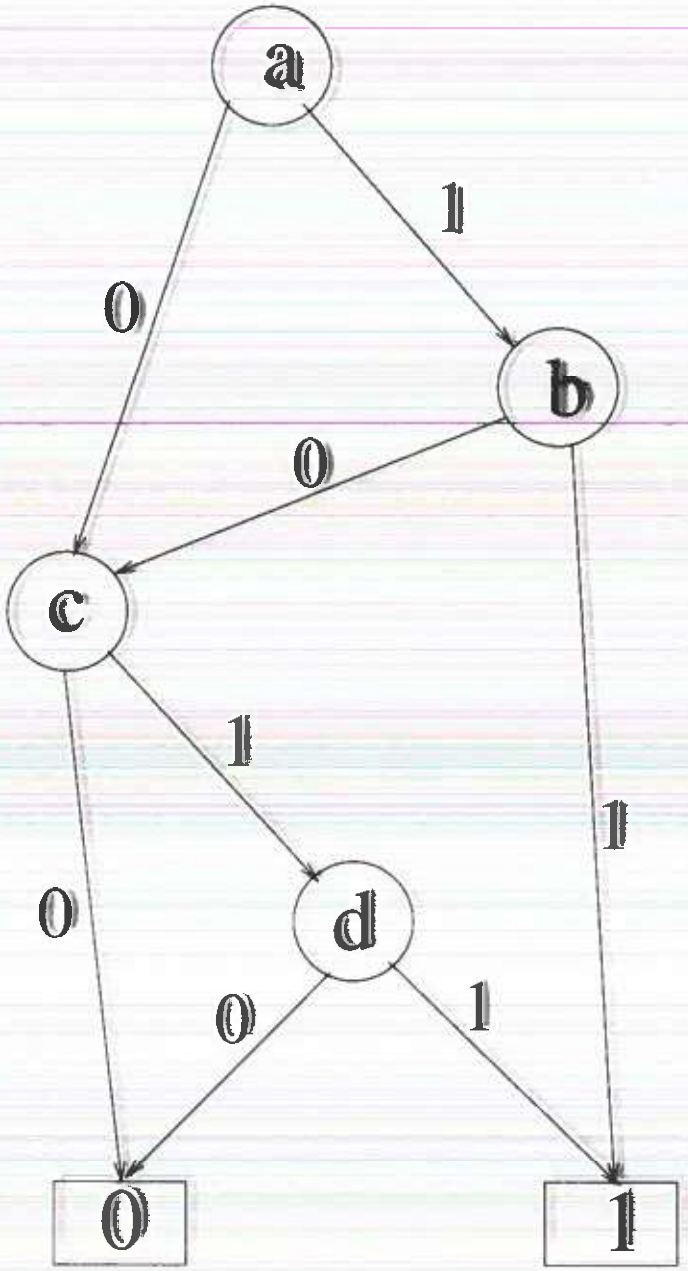


Figure 1: BDD for $(a \wedge b) \vee (c \wedge d)$

Ordered Binary Decision Diagrams

OBDDs: efficient way to manipulate Boolean functions

- directed acyclic graph ("folded decision tree")
- internal nodes correspond to Boolean variables
- all paths lead to one of the two terminal vertices labeled by the values 0 and 1

Properties:

- canonical representation for a given variable ordering
- easy equivalence check
- polynomial Boolean operations

BDD-based SAT

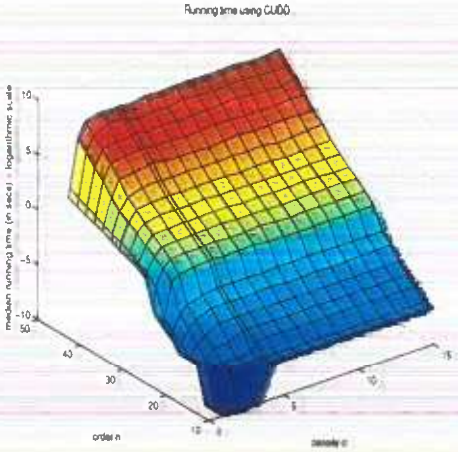
BDD(φ)

- Construct, using Boolean operations, a BDD B_φ such that $models(\varphi) = models(B_\varphi)$.
- check whether $B_\varphi \neq 0$.

Note:

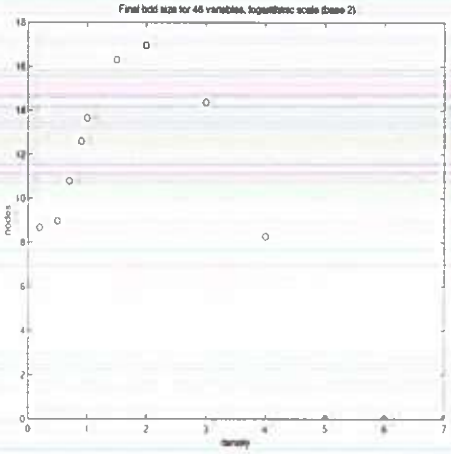
- Complexity depends on size of intermediate BDDs.
- Used in hardware verification
- Provably orthogonal to DLL.

BDDs



Flattens at density 2.0

3-D Plot of median running time

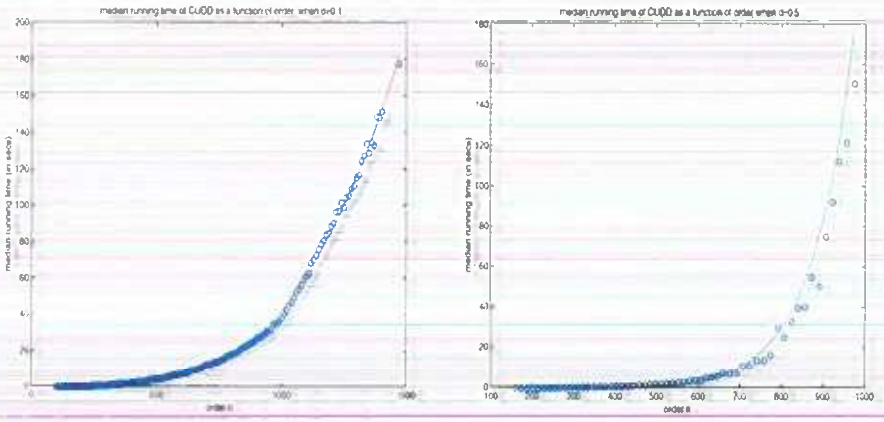


Final BDD size as a function of the density

Phase transition at density 2.0.

14

BDDs - Poly2Exp



CUDD – median running time for (left) density 0.1 and (right) density 0.5

Poly2Exp Phase Transition:

- Density 0.1: cubic running time
- Density 0.5: exponential running time

15

BDDs vs. GRASP

Very different profile:

- No peaks
- Flattens at density 2.0
- Poly2exp transition between 0.1 – 0.5

But:

- GRASP searches for one solution
- BDDs enumerates all solution

Question: Can we have a "fairer" comparison?

Early Quantification

SAT: $\varphi = C_1 \wedge C_2 \wedge \dots \wedge C_m$

Pure SAT: $\varphi = (\exists x_1) \dots (\exists x_m)(C_1 \wedge \dots \wedge C_m)$

- Dichotomy: $B_\varphi = 0$ or $B_\varphi = 1$
- Quantification:

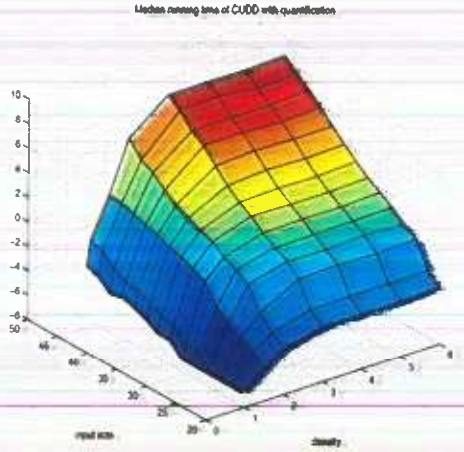
$$B_{(\exists x)\varphi} = B_{\varphi|x=0} \vee B_{\varphi|x=1}$$

Early Quantification: quantify asap

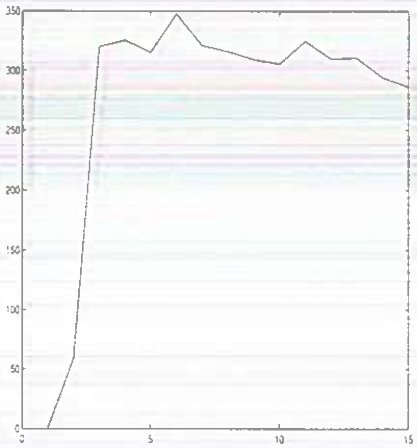
Example

$$(\exists x_1) \dots (\exists x_4)(\neg x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_2 \vee \neg x_3 \vee x_4)) \equiv$$
$$(\exists x_2)(\exists x_3)((\exists x_1)(\neg x_1 \vee \neg x_2 \vee x_3)) \wedge (\exists x_4)(\neg x_2 \vee \neg x_3 \vee x_4))$$

BDDs with Early Quantification



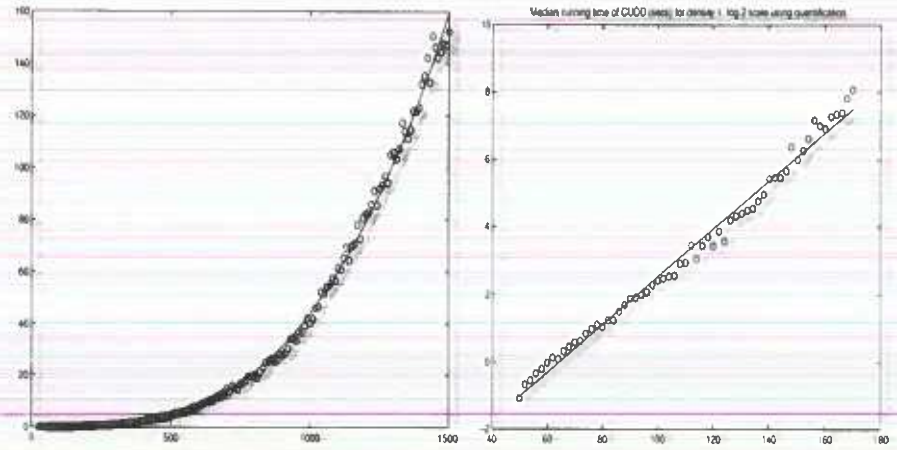
BDD(Q) – 3-D Plot of median running time



BDD(Q) – median running time as a function of the density for order 46

Flatenning: at density 4.0.

BDDs with EQ - Poly2Exp



BDD(Q) – (left) median running time for (left) density 0.5 and (right) density 1

Poly2Exp Phase Transition:

- Density 0.5: quadratic running time
- Density 1.0: exponential running time

BDD vs. BDD(Q)

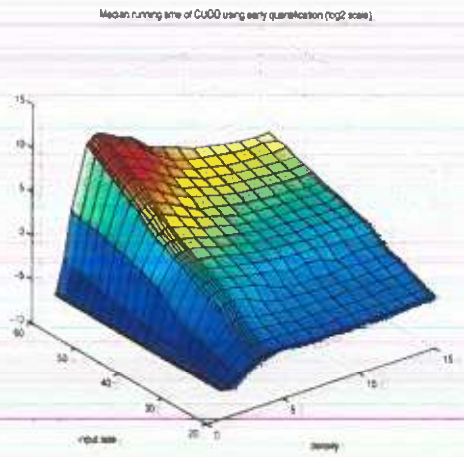
	BDD	BDD(Q)
Flattening	2.0	4.0
Poly2exp	0.1-0.5	0.5-1.0

Another idea from verification:

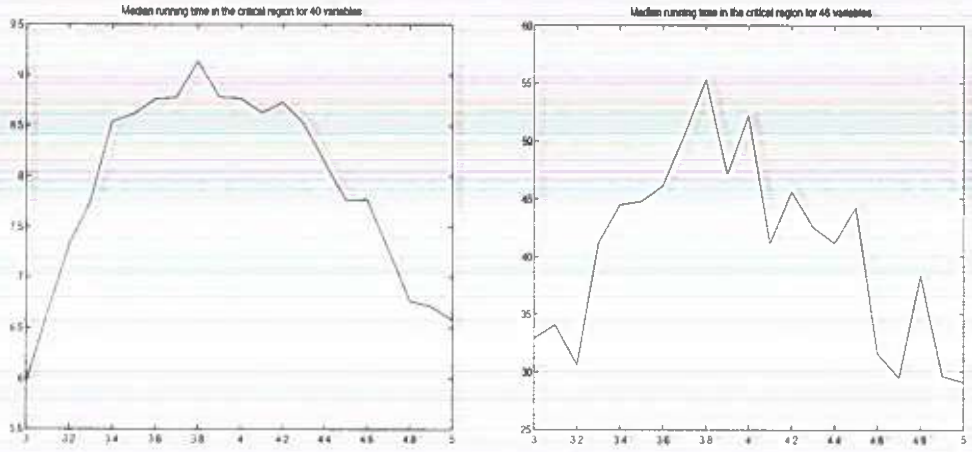
clause reordering

- Early quantification crucial
- Reorder clauses to maximize early quantification
  - optimal order NP-hard
  - greedy heuristics used

BDDs with EQ and Reordering



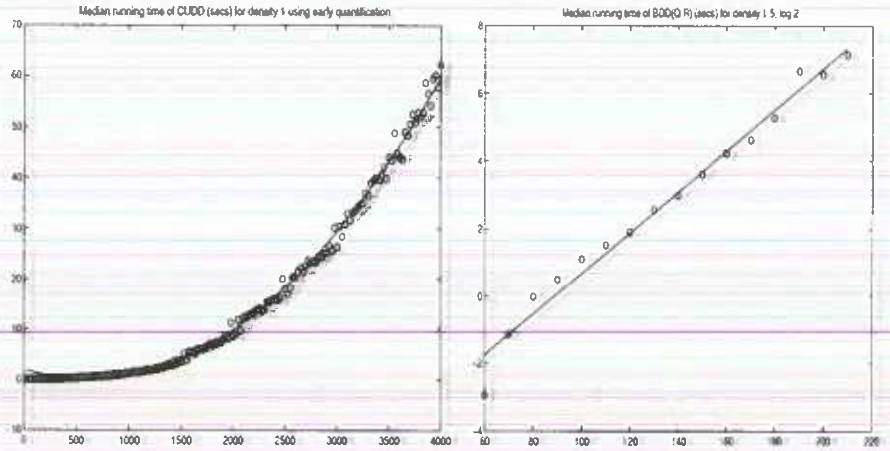
BDD(Q,R) – 3-D Plot of median running time



BDD(Q,R) – median running time, for order 40 (left) and 46 (right)

Peak: at density 3.8.

BDDs with EQ and Reordering – Poly2Exp



BDD(Q,R) – median running time for (left) density 1.0 and (right) density 1.5

Poly2Exp Phase Transition:

- Density 1.0: quadratic running time
- Density 1.5: exponential running time

BDD(Q,R) vs. GRASP

	BDD(Q,R)	GRASP
peak	3.8	4.3
poly2exp	1.0-1.5	3.8

Bottom line:

- common profile: easy-hard-less hard
- different phase transitions: peak
  poly2exp
- BDD(Q,R) not affected by probability phase transition.

Variable Ordering

Well known:

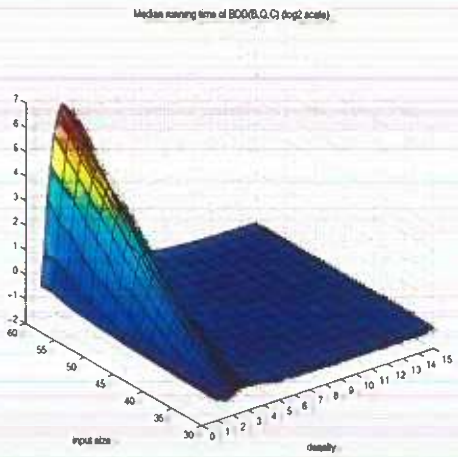
- Canonicity of BDDs is order-dependent
- Size of BDDs is order-dependent

So far: variable ordering selected by BDD package (CUDD).

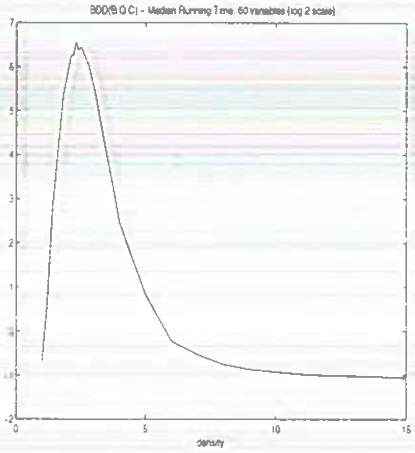
Now: order variables to minimize dependence

- due to Bouquet
- reminiscent of Freuder
- we also use here EQ
- and clause reordering
- and also clause clustering

BDDs with Variable Ordering



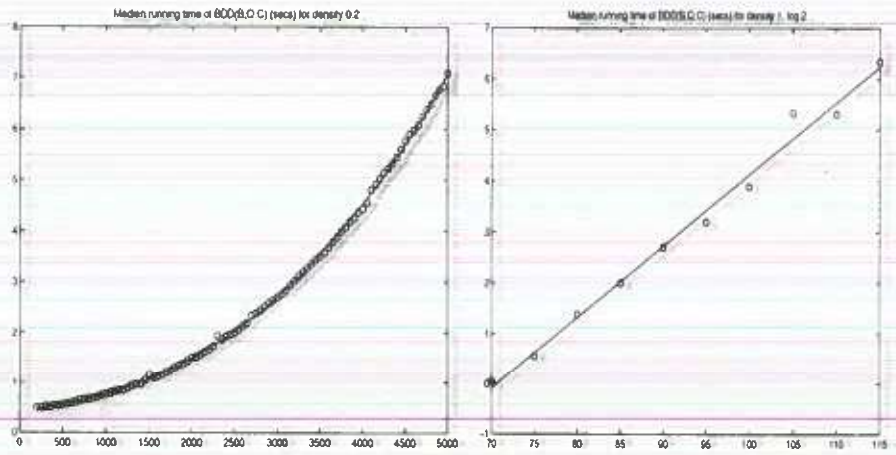
BDD(B,Q,C) – 3-D Plot of median running time



BDD(B,Q,C) – median running times as a function of the density for order 60

Peak: at density 2.3.

BDDs with Variable Ordering - Poly2Exp



BDD(B,Q,C) – median running time for (left) density 0.2 and (right) density 1.0

Poly2Exp Phase Transition:

- Density 0.2: quadratic running time
- Density 1.0: exponential running time

LC

CONCLUSIONS

- "Easy-hard-easy around crossover point" – very simplistic
- Where are the really hard problems? It depends!
  - No evidence of intrinsic hardness
- Is common profile fundamental?
- Should we build density-aware solvers?
- Should we build structure-aware solvers?
- Another puzzle: relationship between polynomial transition and heavy-tail phenomena.