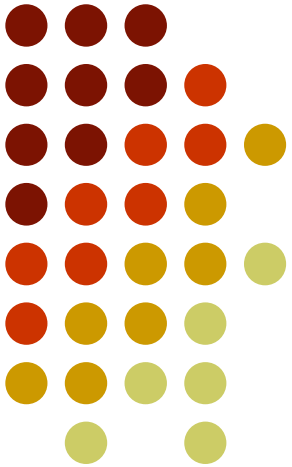
Benchmarking a Model Checker for Algorithmic Improvements and Tuning for Performance

G. Cabodi S.Nocco S.Quer

Politecnico di Torino
Torino, Italy



Outline

- The PdTRAV tool
- Motivations & contributions
- Experiments
- Feedbacks & tuning
- Conclusions

PdTRAV model checker

- **P**olitecnico **d**i **T**orino **R**eachability **A**nalysis & **V**erification (exe now on fmgroup.polito.it)
- NOT a complete verification tool
  - Set of algorithms/engines oriented to evaulation/benchmarking
  - No effort in input language, compiler, GUI (just flat netlist input), etc.
  - Little effort in falsification (we mainly address proofs)
- NO expert system, except ITP-based integrated approach [FMCAD'08])

PdTRAV – low/mid levels

- Low level engines
  - BDDs (CUDD-2.4.1, plus customization)
  - AIGs (our own impl., freely extended from VIS)
  - SAT (Minisat-1.14, no circuit-SAT)
  - ABC (for comb. opt.: rewrite, refactor)
- Mid level library for symbolic manipulation of (Boolean functions, variables, variable sets,, arrays, ..)

PdTRAV – MC engines

- BDD-based: fwd, bwd, fwd/bwd, part (++)
- Interpolants (++)
- Inductive proofs & inductive invariants (.)
- BMC (+)
- Circuit based (AIG) quantification (-)
- CEGAR (-)

PdTRAV – transformations

- Abstractions (localization, 2-phase, …),
- Retiming (minreg+peripheral),
- Inductive equivalences (+ trivial speculations)
- constraints (explicit+hidden),
- relational TR $\rightarrow$circuit transformation

Motivations & goals

- Evaluate different engines
  - we mainly target proof vs. falsification engines
- Build new master module (expert system)
- Classify benchmarks
  - better understanding of problems
- Understand relationships
  - $problem_i$ - $engine_j$ / $engine_i$ - $engine_j$
- Engine tuning (static and dynamic)

Our contributions (preliminary)

- Most of our work revisits common practice
  - Tool benckmarking
  - Engine tuning
  - Pros/cons of BDD- and SAT-based approaches
- Set of classification schemes
- Dynamic tuning
  - engine analyzes his performance and takes/suggests decisions (for speed-up and against other engines)

Phase 1: experiments

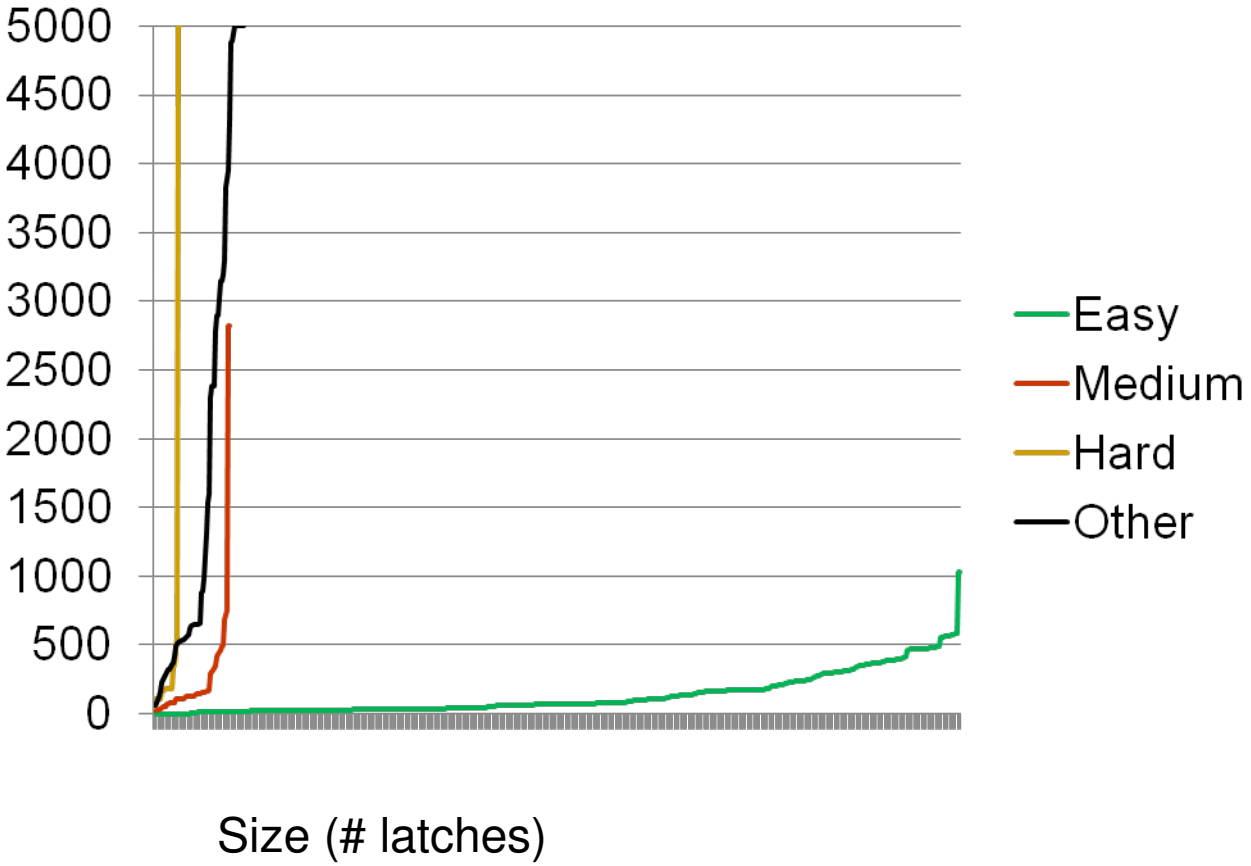
- HWMCC08 benchmark set (645)
- Gather stats on circuits and properties
- 33 runs on engines with different tunings
  - 6 BDDs: fwd, bwd, fb x 2 (no cuts, cuts)
  - 24 ITP: 6 base x 4 Tunings/Transformations
  - 2 Inductive
  - 1 BMC
- Collect stats on engine runs

Phase 2: classification(s)

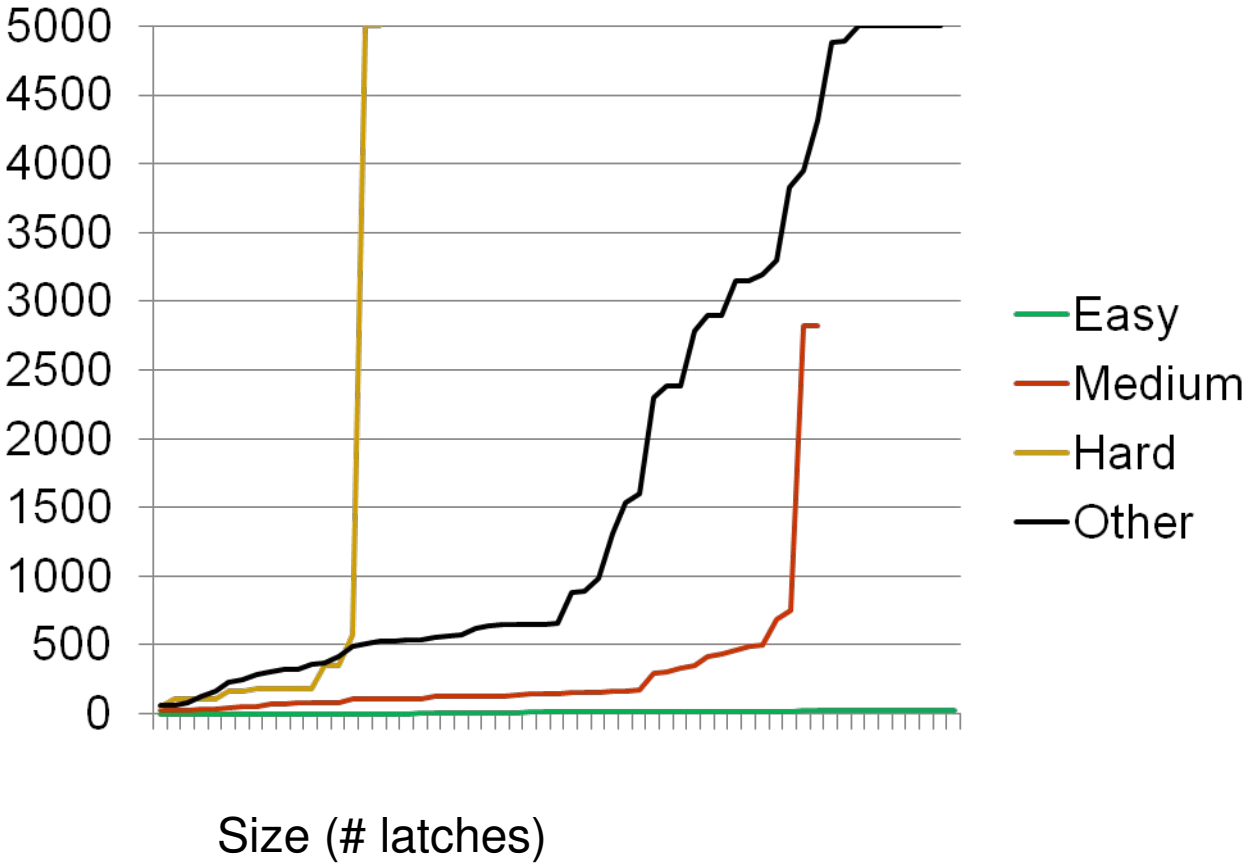
- By execution times (easy to hard)

	Easy (<10 s)	Medium (<2 min)	Hard (< 15 min)	TO
SAT	215	16	7	13
UNSAT	304	33	10	9
?				38
TOT	585 (238+347)			60

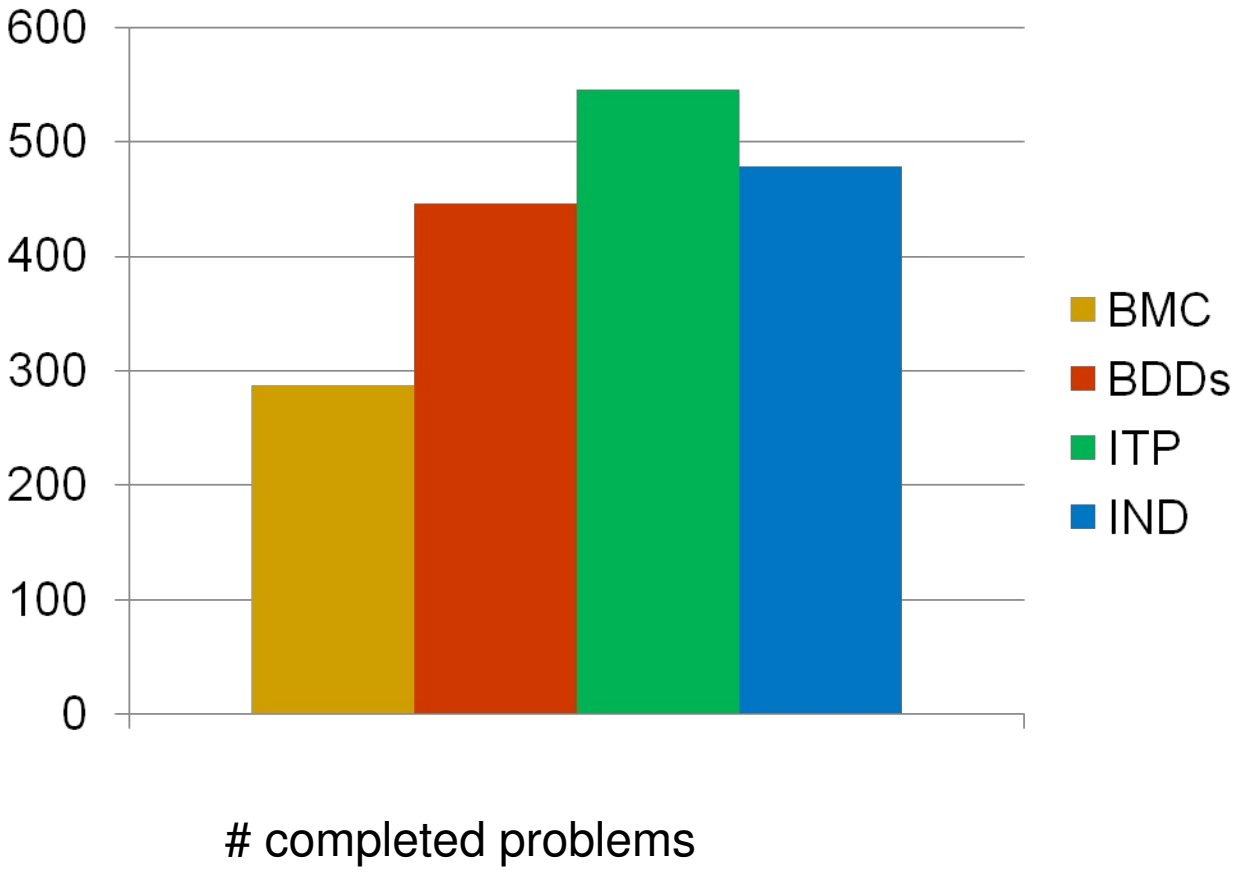
Phase 2: classification(s)



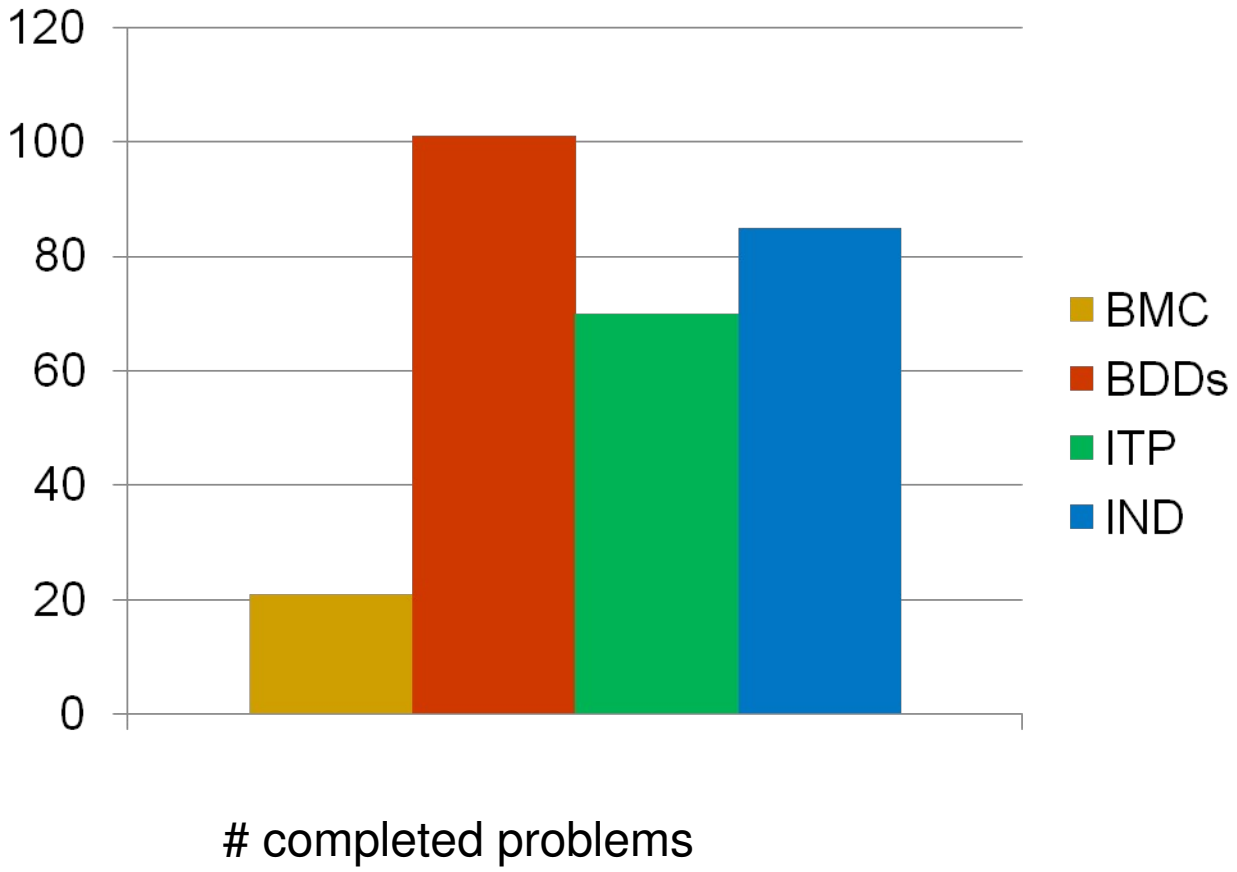
Phase 2: classification(s)



Phase 2a: comparing classes



Phase 2a: comparing classes (easy runs removed)



Correlation classes/engines

- Solved by just one engine (using different settings):

	BDD	ITP	BMC	IND
Full set	9	7	8	1
ITP+ excluded	28	4	8	2

Correlation classes/engines: affinity

- $A(M_i,B_j) = f$ (time,mem,stats)
- Very preliminary
- Aim:
  - high value for engine M_i able to solve problem B_j in low time/memory, with good statistics
  - Could also (indirectly) relate engines, if they have comparable affinity with given problem

Classification: other heuristics

- Analyze property (AIG circuit):
  - Equivalence checking (uncover miters)
  - Multiple properties: $P = \bigwedge_i p_i$
  - Hidden constraints
- Exploit ternary simulation:
  - Guess (rough) on depth (diameter)
- Try BDD encoding
  - Overall BDD size
  - # cut vars

Phase 3: expert system?

- Many easy problems, few medium/hard solved
- Straightforward heuristic:
  - Select 7 engines
  - Run in sequence for 2 min
  - Stop if solved
  - 569 solved problems !!!
- Little room for improvement
- BUT! Could compare on execution times.

Phase 4: package tuning

- Focus on difficult & unolved instances
- Deeper investigation and finer tuning can add more solved problems:
  - we got 12 more !
  - Similar experiences on industrial benchmarks

Dynamic tuning (learning)

- Engine is given initial time/memory resources
- Engine dynamically evaluates performance
  - traversal iterations / time
  - Peak BDD / sifting
  - Size increase / traversal iterations
- Use statistics to drive heuristics:
  - To dynamically change settings (e.g. reclustering, cutpoint merging)
  - To extend / reduce time limits

Transformations

- We have several transformations
  - Abstractions
  - Eq-preserving transformations
  - Reductions
- Generally all transformations simplify the problem
- Experiment show mixed results

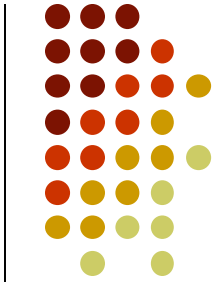
Example: inductive equiv.

- Once an equivalence is given, merge equiv. nodes
- OK !
  - No extra constraint required
  - Circuit simplified
- BUT
  - Equivalence guaranteed on all reachable states
  - Behavior is changed on UNREACHABLE states.
  - Possible impact on backward reachability !

Ad Hoc transformations

- We have implemented a specific transformation from relational to circuit representation
  - Uncover hidden constraints:
    - Equivalences
    - Functional dependencies:
      - NS $\leftarrow$ PI
      - PI $\leftrightarrow$ F(…)
    - Apply transformation
      - NS $\leftarrow$ F(…)

Conclusions



- Target
  - Better understand problem set
  - Build expert system
  - Tune package
- Approach
  - Extensive experimentation with different engines/ settings
  - Classify problems & Correlate engines/problems
- Result (preliminary)
  - Improvement w.r.t. HWMCC08 (40+)

Conclusions (2)

- HWMCC08 benchmarks
  - Many easy problems
  - Difference made on few benchmarks
  - Winner(s) (probably) depends on new benchmarks
- 15 min time limit
  - Good for productivity
  - Low for corner cases & difficult instances
  - Some problems solved in hours
- What if sub-competition on fewer problems with higher TO ?

Thank you!