Motivation

Phase transition

Formalization

Experiments

Approaches

SAT Planning
State-space search
LPG

1st test series

Runtimes
Plan lengths

2nd test series

Phase transition
Runtimes
Plan lengths
LPG, HSP, FF

Discussion

Conclusions

Phase Transitions in Classical Planning: An Experimental Study

Jussi Rintanen

Albert-Ludwigs-Universität Freiburg, Germany

June 7, ICAPS’04

(Albert-Ludwigs-Universität Freiburg)

June 7, ICAPS’04

Motivation

◮ Almost all of the standard benchmarks are solvable by simple polynomial-time problem-specific algorithms.
  ◦ Narrow class, not representative (in general; applications)
  ◦ Say little about performance of planners in general
◮ How were difficult instances obtained: increase the number of packages, airplanes, ... (≥ 2000 state variables, ≥ 4000 operators,)
◮ Actually, 20 state variables and 40 operators is a challenge to many planners!!!

Phase transition

Planning phase transition

How to solve the easiest problems

Bylander 1996: insolubility by a simple syntactic test

Bylander 1996: solvable by a simple hill−climbing algorithm

Problem instances

Characterized by the following parameters.
1. number $n$ of state variables (size of state space)
2. number of operators
3. number of effect literals in operators (our experiments: 2)
4. number of precondition literals (our experiments: 3)
5. number of goal literals (our experiments: $n$)
6. number of goal literals with value differing from the initial value (our experiments: $n$).

Further restrictions

◮ Model B (Bylander 1996): no restrictions.
◮ Model C: each literal occurs as effect at least once. Otherwise very likely some goal literals cannot be made true: many trivially insoluble instances.
◮ Model A: each literal occurs as effect about the same number of times. Model C does not fully fix the problem in Model B, so we go a bit further in Model A.

Analogy: SAT benchmarks

1. Notoriously difficult to come by just by inventing some.
2. Prove that for any algorithm the problem is difficult (pigeon-hole formulas for DPLL/resolution!): not very interesting...
3. Go to Intel and ask for problems that resist solution. (Which company is the Intel of planning?)
4. Experiment with the set of all instances, identifying problem parameters that make planning difficult.
Experiments

Experimental set-up

- Fix other parameters, and vary the number of operators.
  \[ \implies \text{What happens to difficulty when the number of arcs (} \sim \text{operators) in the transition graph is varied?} \]
- Number of instances for given parameter values is astronomical, so we sample the space of all problem instances.
- Evaluate runtimes and plan lengths of different planners.

Approach: satisfiability planning

- First developed by Kautz and Selman (1992, 1996)
- Translate planning into formulae, find plans with a SAT solver.
- The commercially most successful planning technology (outside planning!!!): bounded model-checking since 1999 a leading technology for model-checking, mega-USD business
- Has not been considered competitive on current benchmarks. Main reason: “faster” planners give no quality guarantees.

Planner: SP

- Our own (here: SP for Satisfiability Planning)
- Improved problem encodings: formula size often \[ \leq \frac{1}{5} \text{of BLACKBOX}\] and runtimes \[ \frac{1}{10} \text{to } 1 \text{ on big problems.} \]
- With novel evaluation strategies very good on standard benchmarks without any benchmark-specific tricks!! See ECAI’04 paper.
- BLACKBOX about as good as SP on the small problem instance we discuss in this talk.

Approach: heuristic state-space search

- Heuristic search in the state space + distance heuristics
- Reference: Bonet and Geffner (2001)
- Favored by the planning competition community.

Planners: HSP an FF

1. HSP (Bonet and Geffner, 2001)
2. FF (Hoffmann and Nebel, 2001)
   - additional techniques inspired by the standard benchmarks
   - very good on standard benchmarks

Approaches LPG

- Developed by Gerevini and Serina (1999-)
- Basic data structure: planning graph from Graphplan (Blum & Furst, 1995)
- Local search with incomplete plans (\( \sim \) planning graphs)
- Advantage over earlier planning graph approaches: length increased dynamically during search (optimality given up!)

First test series

- Model A (Results on Model C are similar.)
- 20 state variables, from 36 to 120 operators at interval \( \sim 6 \)
- About 500 soluble instance for each operators / variable ratio (about 8000 soluble instances out of 100000, identified by a BDD-based breadth-first search planner)
- Measure runtimes and plan lengths (timeout 10 minutes)

Runtimes: SP

Model A: Distribution of runtimes on SP
2nd test series

Further tests: scalability

- 20, 40 and 60 state variables (∼10^6, 10^12, 10^18 states)
- No efficient insolubility test: could not distinguish between insoluble and very difficult instances.
- Main results for SP only (SP scales up by far the best.)
- LPG, HSP and FF: proportion of solved instances wrt SP (timeout 10 minutes)
Runtimes: mean

Model A: Runtimes on on bigger problems

proportion of soluble instances vs. ratio # operators / # state variables

average time to find plan in secs

Runtimes: median

Model A: Median runtimes on bigger problems

proportion of soluble instances vs. ratio # operators / # state variables

median time to find plan in secs

Plan lengths

Model A: Plan lengths on bigger problems

proportion of soluble instances vs. ratio # operators / # state variables

average plan length

LPG timeouts

Model A: Success rate of LPG

percentage of instances solved vs. ratio # operators / # state variables

FF timeouts

Model A: Success rate of FF

percentage of instances solved vs. ratio # operators / # state variables

HSP timeouts

Model A: Success rate of HSP

percentage of instances solved vs. ratio # operators / # state variables

Discussion

Why does SP scale up best?

1. Like LPG, SP's problem representation explicitly uses state variables. (a fundamental difference to HSP and FF).
2. Powerful general-purpose inferences: unit resolution, clause learning, ..., as implemented by SAT solvers. (a main difference to LPG)
3. Systematic search algorithm (a main difference to LPG)

Discussion

Why does LPG scale up better than HSP, FF?

1. LPG's problem representation explicitly uses state variables.
2. State-space search in HSP and FF ignores the structural information in the state variables (and operators).
3. HSP and FF look at the state variables only when computing the distance estimates.
Why does HSP scale up better than FF?

- FF has “Helpful Actions Pruning”: ignore operators considered “not helpful” (as suggested by computation of heuristic).
- HAP is a factor in FF’s good performance on many of the big-and-easy benchmarks.
- On easy problems performance improves and equals to HSP when HAP is disabled.
- So HAP is a big drawback when distance heuristics do not work well (all difficult problems and many easy ones.)

Discussion

- Are problems in the phase transition region difficult?
  - Yes, for all of the four planners.
- And outside it they are easy?
  - Yes, for most of the planners. (exception: FF)
- Do the results agree with what is known about the algorithms?
  1. Yes! Bounded model checking (~ satisfiability planning) good in challenging real-world problems: scalability not a direct function of the cardinality of the state space.
  2. Yes! State-space search has not been considered a feasible approach to solve difficult problems with big state spaces (> 10 million states).
  3. Yes/No! Standard planning benchmarks have huge state spaces and are efficiently solved by some state-space planners. But, the benchmarks are actually rather easy.

Conclusions

- We have proposed variants of Bylander’s model of problem instances in classical planning.
- We have tested some of the main approaches to planning on instances inside and outside the phase transition region.
- Results clarify what the strengths of different approaches are. ⇒ Interesting complement to standard benchmarks.