Algorithms for Classical Planning

Jussi Rintanen

Beijing, IJCAI 2013

Planning

What to do to achieve your objectives?

- Which actions to take to achieve your objectives?
- Number of agents
  - single agent, perfect information: s-t-reachability in succinct graphs
  - + nondeterminism/adversary: and-or tree search
  - + partial observability: and-or search in the space of beliefs
- Time
  - asynchronous or instantaneous actions (integer time, unit duration)
  - rational/real time, concurrency
- Objective
  - Reach a goal state.
  - Maximize probability of reaching a goal state.
  - Maximize (expected) rewards.
  - temporal goals (e.g. LTL)

Planning

What to do to achieve your objectives?

- Which actions to take to achieve your objectives?
- Number of agents
  - single agent, perfect information: s-t-reachability in succinct graphs
  - + nondeterminism/adversary: and-or tree search
  - + partial observability: and-or search in the space of beliefs

Time

  - asynchronous or instantaneous actions (integer time, unit duration)
  - rational/real time, concurrency

Objective

  - Reach a goal state.
  - Maximize probability of reaching a goal state.
  - Maximize (expected) rewards.
  - temporal goals (e.g. LTL)

Planning

What to do to achieve your objectives?

- Which actions to take to achieve your objectives?
- Number of agents
  - single agent, perfect information: s-t-reachability in succinct graphs
  - + nondeterminism/adversary: and-or tree search
  - + partial observability: and-or search in the space of beliefs

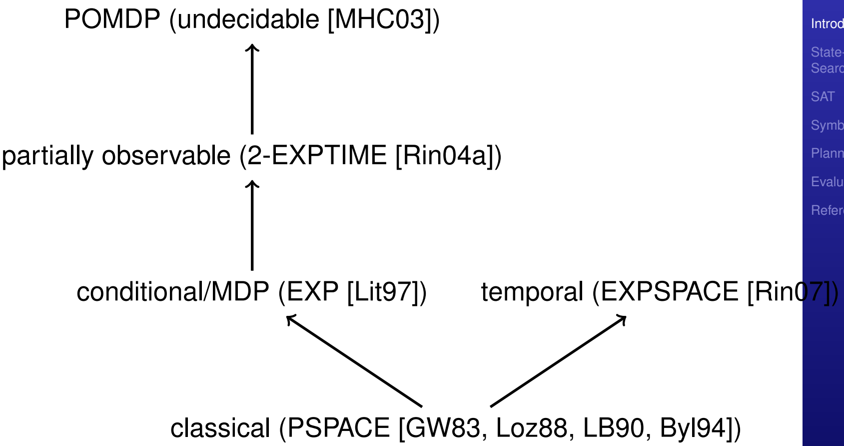
Time

- asynchronous or instantaneous actions (integer time, unit duration)
- rational/real time, concurrency

Objective

- Reach a goal state.
- Maximize probability of reaching a goal state.
- Maximize (expected) rewards.
- temporal goals (e.g. LTL)

Hierarchy of Planning Problems



POMDP (undecidable [MHC03])

partially observable (2-EXPTIME [Rin04a])

conditional/MDP (EXP [Lit97])

temporal (EXPSPACE [Rin07])

classical (PSPACE [GW83, Loz88, LB90, Byl94])

Classical (Deterministic, Sequential) Planning

- states and actions expressed in terms of state variables
- single initial state, that is known
- all actions deterministic
- actions taken sequentially, one at a time
- a goal state (expressed as a formula) reached in the end

Deciding whether a plan exists is PSPACE-complete.
With a polynomial bound on plan length, NP-complete.

Domain-Independent Planning

What is domain-independent?

- general language for representing problems (e.g. PDDL)
- general algorithms to solve problems expressed in it

Advantages and disadvantages:

+ Representation of problems at a high level
+ Fast prototyping
+ Often easy to modify and extend
- Potentially high performance penalty w.r.t. specialized algorithms
- Trade-off between generality and efficiency

Domain-Specific Planning

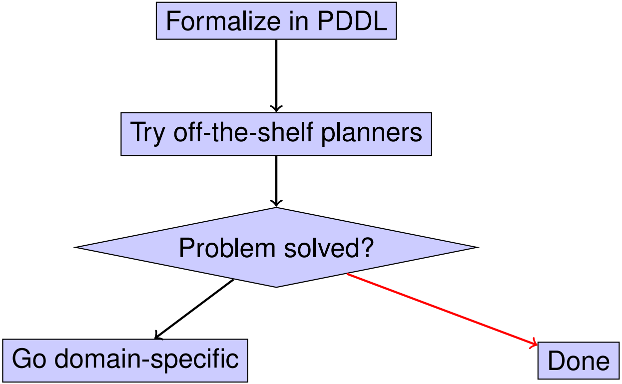
What is domain-specific?

- application-specific representation
- application-specific constraints/propagators
- application-specific heuristics

There are some planning systems that have aspects of these, but mostly this means: implement everything from scratch.

Domain-Dependent vs. -Independent Planning

Procedure



Formalize in PDDL

Try off-the-shelf planners

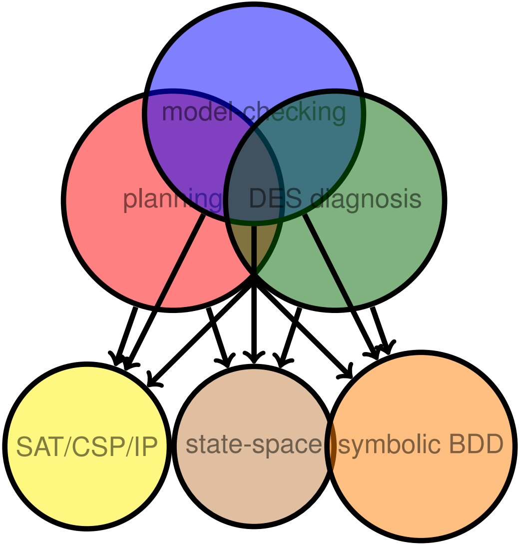
Problem solved?

Go domain-specific

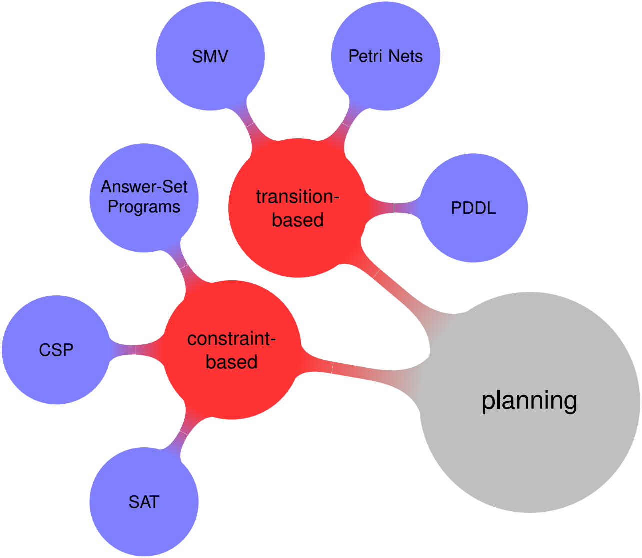
Done

Related Problems, Reductions

planning, diagnosis [SSL^{+}95], model-checking (verification)



How to Represent Planning Problems?



PDDL - Planning Domain Description Language

- Defined in 1998 [McD98], with several extensions later.
- Lisp-style syntax
- Widely used in the planning community.
- Most basic version with Boolean state variables only.
- Action sets expressed as schemata instantiated with objects.

```
(:action analyze-2
   :parameters (?s1 ?s2 - segment ?c1 ?c2 - car)
   :precondition (and (CYCLE-2-WITH-ANALYSIS ?s1 ?s2)
                      (on ?c1 ?s1))
   :effect (and (not (on ?c1 ?s1))
                (on ?c2 ?s1)
                (analyzed ?c1)
                (increase (total-cost) 3)))
```

States

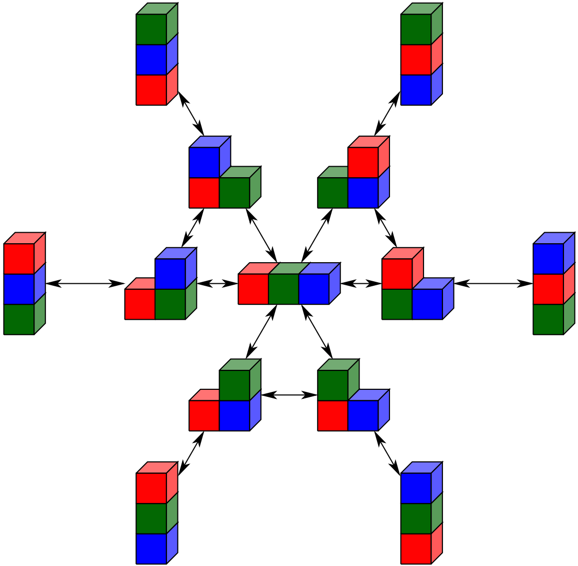
States are valuations of state variables.

Example

State variables are		One state is	
LOCATION:	$\{0, \ldots, 1000\}$	LOCATION =	312
GEAR:	$\{R, 1, 2, 3, 4, 5\}$	GEAR =	4
FUEL:	$\{0, \ldots, 60\}$	FUEL =	58
SPEED:	$\{-20, \ldots, 200\}$	SPEED =	110
DIRECTION:	$\{0, \ldots, 359\}$	DIRECTION =	90

State-space transition graphs

Blocks world with three blocks



Actions

How values of state variables change

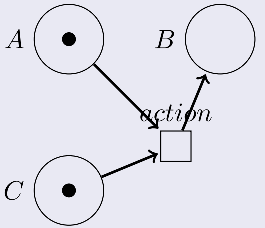
General form

precondition: A=1 $\wedge$ C=1
effect: A := 0; B := 1; C := 0;

STRIPS representation

PRE: A, C
ADD: B
DEL: A, C

Petri net



Weaknesses in Existing Languages

- High-level concepts not easily/efficiently expressible. Examples: graph connectivity, transitive closure.
- Limited or no facilities to express domain-specific information (control, pruning, heuristics).
- The notion of classical planning is limited:
  - Real world rarely a single run of the sense-plan-act cycle.
  - Main issue often uncertainty, costs, or both.
  - Often rational time and concurrency are critical.

Formalization of Planning in This Tutorial

A problem instance in (classical) planning consists of the following.

- set X of state variables
- set A of actions $\langle p, e \rangle$ where
  - p is the precondition (a set of literals over X)
  - e is the effects (a set of literals over X)
- initial state $I : X \to \{0, 1\}$ (a valuation of X)
- goals G (a set of literals over X)

The planning problem

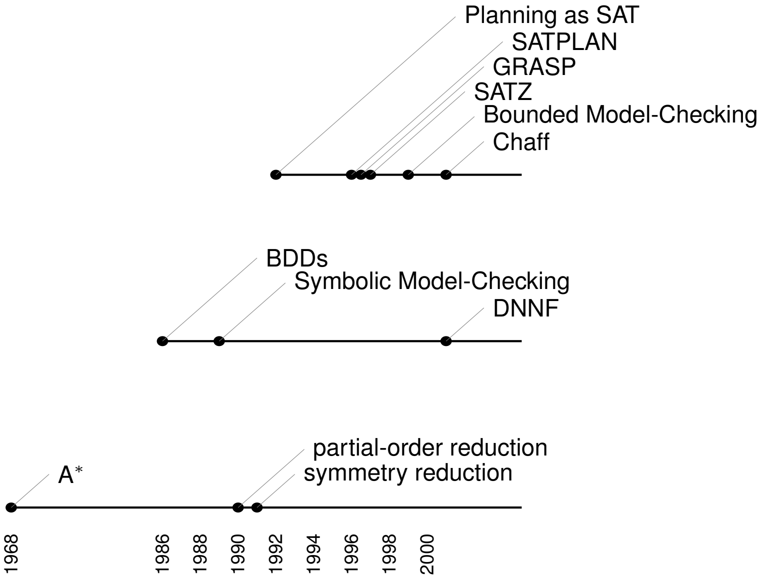
An action $a = \langle p, e \rangle$ is applicable in state s iff $s \models p$.
The successor state $s' = \mathit{exec}_a(s)$ is defined by

- $s' \models e$
- $s(x) = s'(x)$ for all $x \in X$ that don't occur in e.

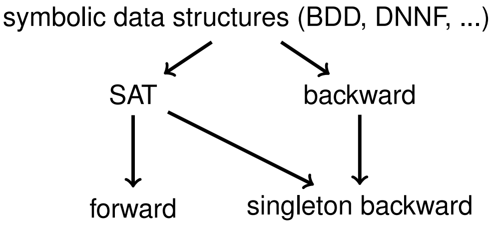
Problem

Find $a_1, \dots, a_n$ such that
$\mathit{exec}_{a_n}(\mathit{exec}_{a_{n-1}}(\cdots \mathit{exec}_{a_2}(\mathit{exec}_{a_1}(I)) \cdots)) \models G$?

Development of state-space search methods



Symbolic Representations vs. Fwd and Bwd Search



symbolic data structures (BDD, DNNF, ...)

SAT

backward

forward

singleton backward

1. symbolic data structures
2. SAT
3. state-space search
4. others: partial-order planning [MR91] (until 1995)

Explicit State-Space Search

- The most basic search method for transition systems
- Very efficient for small state spaces (1 million states)
- Easy to implement
- Very well understood
- Pruning methods:
    - symmetry reduction [Sta91, ES96]
    - partial-order reduction [God91, Val91]
    - lower-bounds / heuristics, for informed search [HNR68]

State Representation

Each state represented explicitly ⇒ compact state representation important

- Boolean (0, 1) state variables represented by one bit
- Inter-variable dependencies enable further compaction:
  - ¬(at(A,L1)∧at(A,L2)) always true
  - automatic recognition of invariants [BF97, Rin98, Rin08]
  - n exclusive variables $x_1, \ldots, x_n$ represented by $1 + \lfloor \log_2(n-1) \rfloor$ bits

Search Algorithms

- uninformed/blind search: depth-first, breadth-first, ...
- informed search: “best first” search (always expand best state so far)
- informed search: local search algorithms such as simulated annealing, tabu search and others [KGJV83, DS90, Glo89] (little used in planning)
- optimal algorithms: A* [HNR68], IDA* [Kor85]

Symmetry Reduction [Sta91, ES96]

Idea

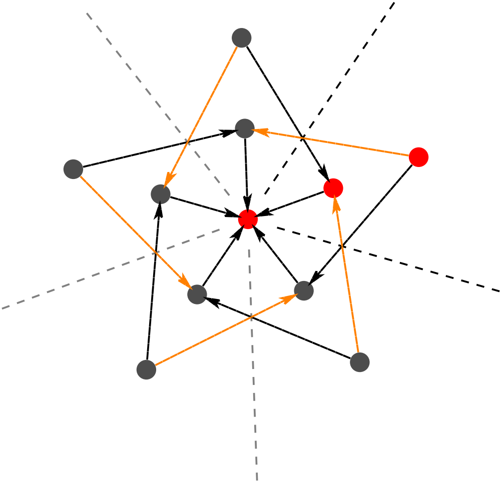
1. Define an equivalence relation $\sim$ on the set of all states: $s_1 \sim s_2$ means that state s_1 is symmetric with s_2.
2. Only one state s_C in each equivalence class C needs to be considered.
3. If state $s \in C$ with $s \neq [s_C]$ is encountered, replace it with s_C.

Example

States $P(A) \wedge \neg P(B) \wedge P(C)$ and $\neg P(A) \wedge P(B) \wedge P(C)$ are symmetric because of the permutation $A \mapsto B, B \mapsto A, C \mapsto C$.

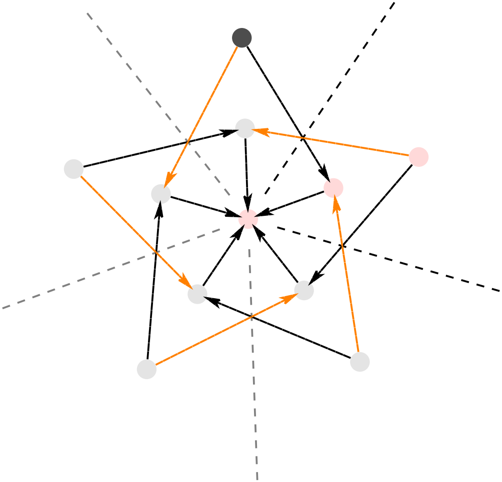
Symmetry Reduction

Example: 11 states, 3 equivalence classes



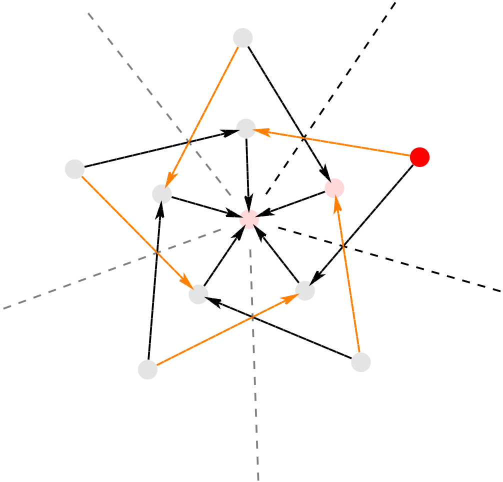
Symmetry Reduction

Example: 11 states, 3 equivalence classes



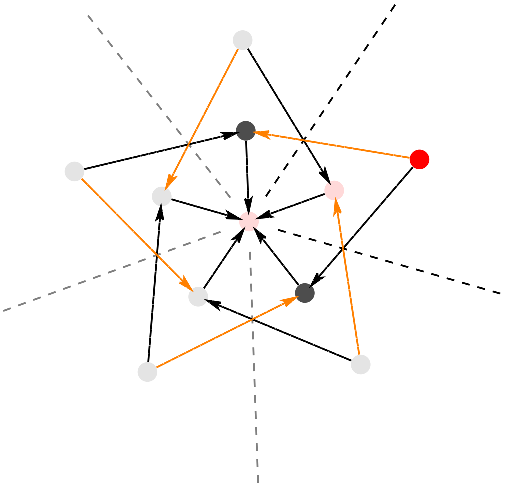
Symmetry Reduction

Example: 11 states, 3 equivalence classes



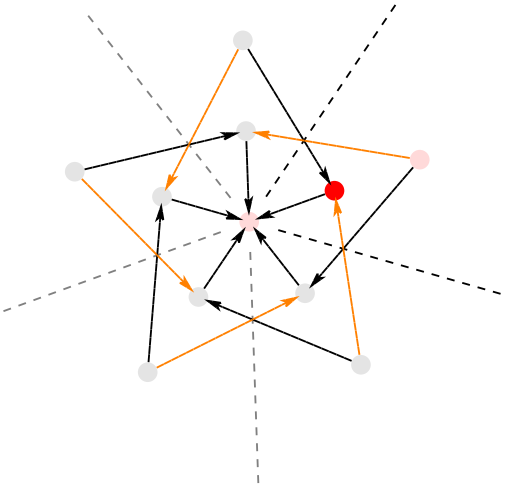
Symmetry Reduction

Example: 11 states, 3 equivalence classes



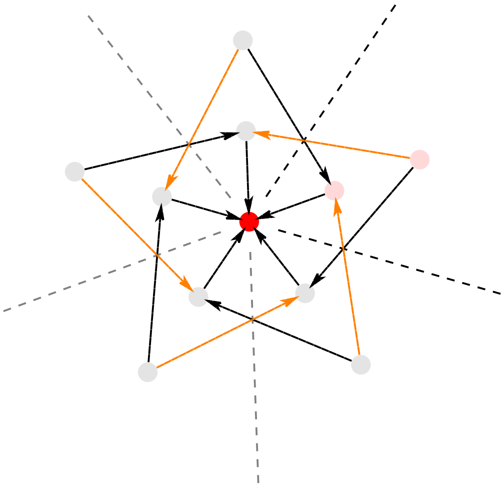
Symmetry Reduction

Example: 11 states, 3 equivalence classes



Symmetry Reduction

Example: 11 states, 3 equivalence classes



Partial Order Reduction

Stubborn sets and related methods

Idea [God91, Val91]

Independent actions unnecessary to consider in all orderings, e.g. both A_1, A_2 and A_2, A_1.

Example

Let there be lamps $1, 2, \ldots, n$ which can be turned on. There are no other actions. One can restrict to plans in which lamps are turned on in the ascending order: switching lamp n after lamp $m > n$ needless.[a]

---

[a] The same example is trivialized also by symmetry reduction!

Heuristics for Classical Planning

The most basic heuristics widely used for non-optimal planning:

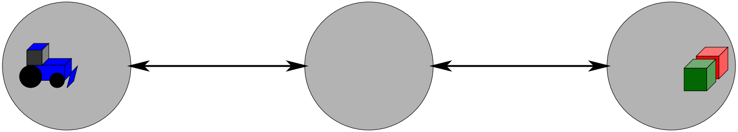
h^{max}	[BG01, McD96]	best-known admissible heuristic
h^{+}	[BG01]	still state-of-the-art
h^{relax}	[HN01]	often more accurate, but performs like h^{+}

Definition of h^{max}, h^{+} and h^{relax}

- Basic insight: estimate distances between possible state variable values, not states themselves.
- $g_s(l) = \begin{cases} 0 & \text{if } s \models l \\ \min_{a \text{ with effect } p} (1 + g_s(\text{prec}(a))) \end{cases}$
- h^{+} defines $g_s(L) = \sum_{l \in L} g_s(l)$ for sets S.
- h^{max} defines $g_s(L) = \max_{l \in L} g_s(l)$ for sets S.
- h^{relax} counts the number of actions in computation of h^{max}.

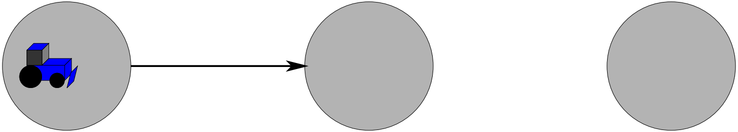
Computation of h^{max}

Tractor example



1. Tractor moves:
   - from 1 to 2: $T12 = \langle T1, \{T2, \neg T1\}\rangle$
   - from 2 to 1: $T21 = \langle T2, \{T1, \neg T2\}\rangle$
   - from 2 to 3: $T23 = \langle T2, \{T3, \neg T2\}\rangle$
   - from 3 to 2: $T32 = \langle T3, \{T2, \neg T3\}\rangle$
2. Tractor pushes A:
   - from 2 to 1: $A21 = \langle T2 \wedge A2, \{T1, A1, \neg T2, \neg A2\}\rangle$
   - from 3 to 2: $A32 = \langle T3 \wedge A3, \{T2, A2, \neg T3, \neg A3\}\rangle$
3. Tractor pushes B:
   - from 2 to 1: $B21 = \langle T2 \wedge B2, \{T1, B1, \neg T2, \neg B2\}\rangle$
   - from 3 to 2: $B32 = \langle T3 \wedge B3, \{T2, B2, \neg T3, \neg B3\}\rangle$

Computation of h^{max}
Tractor example

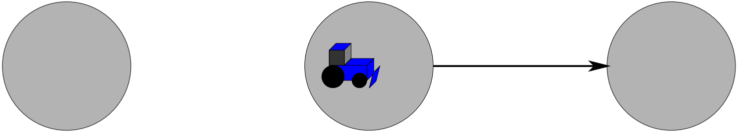


t	T1	T2	T3	A1	A2	A3	B1	B2	B3
0	T	F	F	F	F	T	F	F	T
1	TF	TF	F	F	F	T	F	F	T
2	TF	TF	TF	F	F	T	F	F	T
3	TF	TF	TF	F	TF	TF	F	TF	TF
4	TF	TF	TF	TF	TF	TF	TF	TF	TF

Apply $T12 = \langle T1, \{T2, \neg T1\} \rangle$

Computation of h^{max}

Tractor example

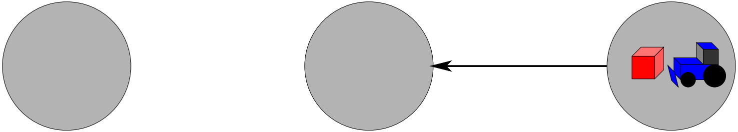


t	T1	T2	T3	A1	A2	A3	B1	B2	B3
0	T	F	F	F	F	T	F	F	T
1	TF	TF	F	F	F	T	F	F	T
2	TF	TF	TF	F	F	T	F	F	T
3	TF	TF	TF	F	TF	TF	F	TF	TF
4	TF	TF	TF	TF	TF	TF	TF	TF	TF

Apply $T23 = \langle T2, \{T3, \neg T2\}\rangle$

Computation of h^{max}

Tractor example

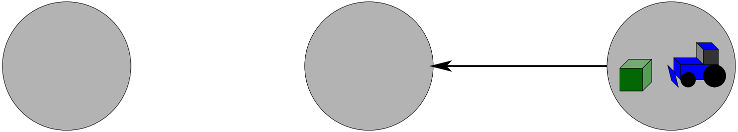


t	T1	T2	T3	A1	A2	A3	B1	B2	B3
0	T	F	F	F	F	T	F	F	T
1	TF	TF	F	F	F	T	F	F	T
2	TF	TF	TF	F	F	T	F	F	T
3	TF	TF	TF	F	TF	TF	F	TF	TF
4	TF	TF	TF	TF	TF	TF	TF	TF	TF

Apply $A32 = \langle T3 \wedge A3, \{T2, A2, \neg T3, \neg A3\}\rangle$

Computation of h^{max}

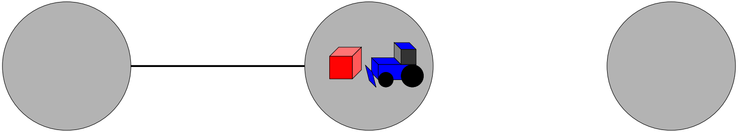
Tractor example



t	T1	T2	T3	A1	A2	A3	B1	B2	B3
0	T	F	F	F	F	T	F	F	T
1	TF	TF	F	F	F	T	F	F	T
2	TF	TF	TF	F	F	T	F	F	T
3	TF	TF	TF	F	TF	TF	F	TF	TF
4	TF	TF	TF	TF	TF	TF	TF	TF	TF

Apply $B32 = \langle T3 \wedge B3, \{T2, B2, \neg T3, \neg B3\}\rangle$

Computation of h^{max}
Tractor example

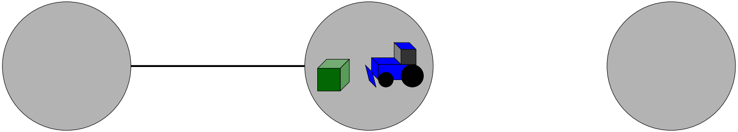


t	T1	T2	T3	A1	A2	A3	B1	B2	B3
0	T	F	F	F	F	T	F	F	T
1	TF	TF	F	F	F	T	F	F	T
2	TF	TF	TF	F	F	T	F	F	T
3	TF	TF	TF	F	TF	TF	F	TF	TF
4	TF	TF	TF	TF	TF	TF	TF	TF	TF

Apply $A21 = \langle T2 \wedge A2, \{T1, A1, \neg T2, \neg A2\}\rangle$

Computation of h^{max}

Tractor example

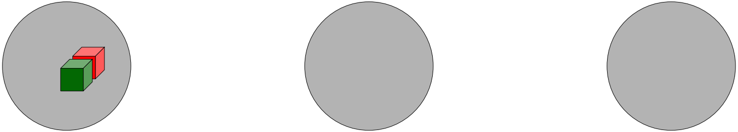


t	T1	T2	T3	A1	A2	A3	B1	B2	B3
0	T	F	F	F	F	T	F	F	T
1	TF	TF	F	F	F	T	F	F	T
2	TF	TF	TF	F	F	T	F	F	T
3	TF	TF	TF	F	TF	TF	F	TF	TF
4	TF	TF	TF	TF	TF	TF	TF	TF	TF

Apply $B21 = \langle T2 \wedge B2, \{T1, B1, \neg T2, \neg B2\}\rangle$

Computation of h^{max}

Tractor example



t	T1	T2	T3	A1	A2	A3	B1	B2	B3
0	T	F	F	F	F	T	F	F	T
1	TF	TF	F	F	F	T	F	F	T
2	TF	TF	TF	F	F	T	F	F	T
3	TF	TF	TF	F	TF	TF	F	TF	TF
4	TF	TF	TF	TF	TF	TF	TF	TF	TF

Distance of $A1 \wedge B1$ is 4.

h^{max} Underestimates

Example

Estimate for lamp1on ∧ lamp2on ∧ lamp3on with

$$\langle\top, \{\text{lamp1on}\}\rangle$$
$$\langle\top, \{\text{lamp2on}\}\rangle$$
$$\langle\top, \{\text{lamp3on}\}\rangle$$

is 1. Actual shortest plan has length 3.
By definition, $h^{max}(G_1 \wedge \cdots \wedge G_n)$ is the maximum of $h^{max}(G_1), \ldots, h^{max}(G_n)$.
If goals are independent, the sum of the estimates is more accurate.

Computation of h^+

Tractor example

t	T1	T2	T3	A1	A2	A3	B1	B2	B3
0	T	F	F	F	F	T	F	F	T
1	TF	TF	F	F	F	T	F	F	T
2	TF	TF	TF	F	F	T	F	F	T
3	TF	TF	TF	F	TF	TF	F	TF	TF
4	TF	TF	TF	F	TF	TF	F	TF	TF
5	TF	TF	TF	TF	TF	TF	TF	TF	TF

Apply $A21 = \langle T2 \wedge A2, \{T1, A1, \neg T2, \neg A2\}\rangle$.
$h^+(T2 \wedge A2)$ is 1+3.
$h^+(A1)$ is 1+3+1 = 5 (h^{max} gives 4.)

Computation of h^+

Tractor example

t	T1	T2	T3	A1	A2	A3	B1	B2	B3
0	T	F	F	F	F	T	F	F	T
1	TF	TF	F	F	F	T	F	F	T
2	TF	TF	TF	F	F	T	F	F	T
3	TF	TF	TF	F	TF	TF	F	TF	TF
4	TF	TF	TF	F	TF	TF	F	TF	TF
5	TF	TF	TF	TF	TF	TF	TF	TF	TF

h^+ of $A1 \wedge B1$ is $5 + 5 = 10$.

Computation of h^{relax}
Motivation

actions	estimate for $a \wedge b \wedge c$ max	sum	actual
$\langle \top, \{a, b, c\} \rangle$	1	3	1
$\langle \top, \{a\} \rangle, \langle \top, \{b\} \rangle, \langle \top, \{c\} \rangle$	1	3	3

- Better estimates with h^{relax} (but: performance is similar to h^{+}).
- Application: directing search with preferred actions [Vid04, RH09]

Computation of h^{relax}

t	T1	T2	T3	A1	A2	A3	B1	B2	B3
0	T	F	F	F	F	T	F	F	T
1	TF	TF	F	F	F	T	F	F	T
2	TF	TF	TF	F	F	T	F	F	T
3	TF	TF	TF	F	TF	TF	F	TF	TF
4	TF	TF	TF	TF	TF	TF	TF	TF	TF

Estimate for $A1 \wedge B1$ with relaxed plans:

t	relaxed plan
0	
1	
2	
3	

Computation of h^{relax}

t	T1	T2	T3	A1	A2	A3	B1	B2	B3
0	T	F	F	F	F	T	F	F	T
1	TF	TF	F	F	F	T	F	F	T
2	TF	TF	TF	F	F	T	F	F	T
3	TF	TF	TF	F	TF	TF	F	TF	TF
4	TF	TF	TF	TF	TF	TF	TF	TF	TF

Estimate for $A1 \wedge B1$ with relaxed plans:

t	relaxed plan
0	
1	
2	
3	A21, B21

Computation of h^{relax}

t	T1	T2	T3	A1	A2	A3	B1	B2	B3
0	T	F	F	F	F	T	F	F	T
1	TF	TF	F	F	F	T	F	F	T
2	TF	TF	TF	F	F	T	F	F	T
3	TF	TF	TF	F	TF	TF	F	TF	TF
4	TF	TF	TF	TF	TF	TF	TF	TF	TF

Estimate for $A1 \wedge B1$ with relaxed plans:

t	relaxed plan
0	
1	
2	
3	A21, B21

Computation of h^{relax}

t	T1	T2	T3	A1	A2	A3	B1	B2	B3
0	T	F	F	F	F	T	F	F	T
1	TF	TF	F	F	F	T	F	F	T
2	TF	TF	TF	F	F	T	F	F	T
3	TF	TF	TF	F	TF	TF	F	TF	TF
4	TF	TF	TF	TF	TF	TF	TF	TF	TF

Estimate for $A1 \wedge B1$ with relaxed plans:

t	relaxed plan
0	
1	
2	A32, B32
3	A21, B21

Computation of h^{relax}

t	T1	T2	T3	A1	A2	A3	B1	B2	B3
0	T	F	F	F	F	T	F	F	T
1	TF	TF	F	F	F	T	F	F	T
2	TF	TF	TF	F	F	T	F	F	T
3	TF	TF	TF	F	TF	TF	F	TF	TF
4	TF	TF	TF	TF	TF	TF	TF	TF	TF

Estimate for $A1 \wedge B1$ with relaxed plans:

t	relaxed plan
0	
1	
2	A32, B32
3	A21, B21

Computation of h^{relax}

t	T1	T2	T3	A1	A2	A3	B1	B2	B3
0	T	F	F	F	F	T	F	F	T
1	TF	TF	F	F	F	T	F	F	T
2	TF	TF	TF	F	F	T	F	F	T
3	TF	TF	TF	F	TF	TF	F	TF	TF
4	TF	TF	TF	TF	TF	TF	TF	TF	TF

Estimate for $A1 \wedge B1$ with relaxed plans:

t	relaxed plan
0	
1	T23
2	A32, B32
3	A21, B21

Computation of h^{relax}

t	T1	T2	T3	A1	A2	A3	B1	B2	B3
0	T	F	F	F	F	T	F	F	T
1	TF	TF	F	F	F	T	F	F	T
2	TF	TF	TF	F	F	T	F	F	T
3	TF	TF	TF	F	TF	TF	F	TF	TF
4	TF	TF	TF	TF	TF	TF	TF	TF	TF

Estimate for $A1 \wedge B1$ with relaxed plans:

t	relaxed plan
0	
1	T23
2	A32, B32
3	A21, B21

Computation of h^{relax}

t	T1	T2	T3	A1	A2	A3	B1	B2	B3
0	T	F	F	F	F	T	F	F	T
1	TF	TF	F	F	F	T	F	F	T
2	TF	TF	TF	F	F	T	F	F	T
3	TF	TF	TF	F	TF	TF	F	TF	TF
4	TF	TF	TF	TF	TF	TF	TF	TF	TF

Estimate for $A1 \wedge B1$ with relaxed plans:

t	relaxed plan
0	T12
1	T23
2	A32, B32
3	A21, B21

Computation of h^{relax}

t	T1	T2	T3	A1	A2	A3	B1	B2	B3
0	T	F	F	F	F	T	F	F	T
1	TF	TF	F	F	F	T	F	F	T
2	TF	TF	TF	F	F	T	F	F	T
3	TF	TF	TF	F	TF	TF	F	TF	TF
4	TF	TF	TF	TF	TF	TF	TF	TF	TF

Estimate for $A1 \wedge B1$ with relaxed plans:

t	relaxed plan
0	T12
1	T23
2	A32, B32
3	A21, B21

estimate = number of actions in relaxed plan = 6

Comparison of the Heuristics

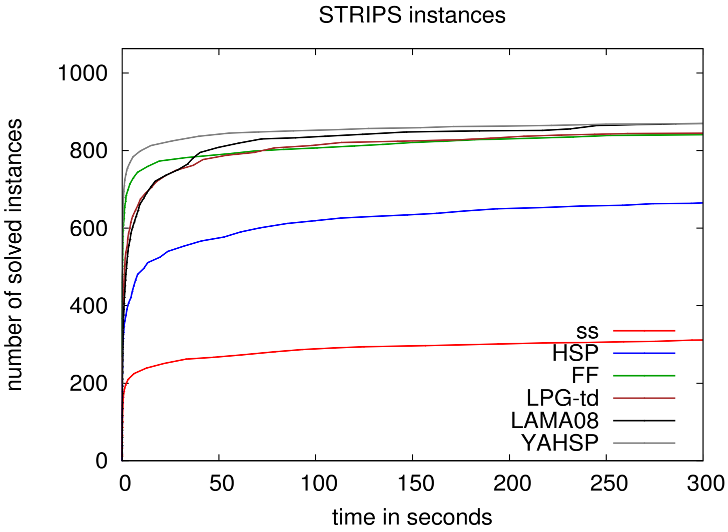
- For the Tractor example:
    - actions in the shortest plan: 8
    - h^{max} yields 4 (never overestimates).
    - h^{+} yields 10 (may under or overestimate).
    - h^{relax} yield 6 (may under or overestimate).
- The sum-heuristic and the relaxed plan heuristic are used in practice for non-optimal planners.

Preferred Actions

- h^+ and h^{relax} boosted with preferred/helpful actions.
- Preferred actions on the first level $t = 0$ in a relaxed plan.
- Several possibilities:
  - Always expand with a preferred action when possible [Vid04].
  - A tie-breaker when the heuristic values agree [RH09].
- Planners based on explicit state-space search use them: YAHSP, LAMA.

Performance of State-Space Search Planners

Planning Competition Problems



Heuristics for Optimal Planning

Admissible heuristics are needed for finding optimal plans, e.g with A* [HNR68]. Scalability much poorer.

Pattern Databases [CS96, Ede00]

Abstract away many/most state variables, and use the length/cost of the optimal solution to the remaining problem as an estimate.

Generalized Abstraction (merge and shrink) [DFP09, HHH07]

A generalization of pattern databases, allowing more complex aggregation of states (not just identification of ones agreeing on a subset of state variables.)

Landmark-cut [HD09] has been doing well with planning competition problems.

Planning with SAT
Background

- Proposed by Kautz and Selman [KS92].
- Idea as in Cook's proof of NP-hardness of SAT [Coo71]: encode each step of a plan as a propositional formula.
- Intertranslatability of NP-complete problems ⇒ reductions to many other problems possible.

Related solution methods

constraint satisfaction (CSP)	[vBC99, DK01]
NM logic programs / answer-set programs	[DNK97]

Translations from SAT into other formalisms often simple. In terms of performance, SAT is usually the best choice.

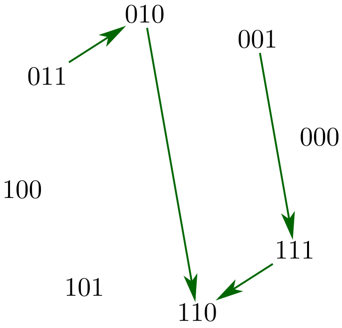
Transition relations in propositional logic

State variables are
$X = \{a, b, c\}$.

$$\begin{array}{l}
(\neg a \wedge b \wedge c \wedge \neg a' \wedge b' \wedge \neg c') \vee \\
(\neg a \wedge b \wedge \neg c \wedge a' \wedge b' \wedge \neg c') \vee \\
(\neg a \wedge \neg b \wedge c \wedge a' \wedge b' \wedge c') \vee \\
(a \wedge b \wedge c \wedge a' \wedge b' \wedge \neg c')
\end{array}$$

The corresponding matrix is

$$\begin{array}{c|cccccccc}
 & 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \\
\hline
000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
001 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
010 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
011 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
100 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
101 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
110 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
111 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{array}$$



Encoding of Actions as Formulas
for Sequential Plans

An action j corresponds to the conjunction of the precondition $P_j@t$ and

$$x_i@(t+1) \leftrightarrow F_i(x_1@t, \ldots, x_n@t)$$

for all $i \in \{1, \ldots, n\}$. Denote this by $E_j@t$.

Example (move-from-X-to-Y)

$$\overbrace{atX@t}^{\text{precond}} \wedge \overbrace{\begin{array}{l}(atX@(t+1) \leftrightarrow \bot) \wedge (atY@(t+1) \leftrightarrow \top) \\ \wedge(atZ@(t+1) \leftrightarrow atZ@t) \wedge (atU@(t+1) \leftrightarrow atU@t)\end{array}}^{\text{effects}}$$

Choice between actions $1, \ldots, m$ expressed by the formula

$$\mathcal{R}@t = E_1@t \vee \cdots \vee E_m@t.$$

Finding a Plan with SAT

Let

- I be a formula expressing the initial state, and
- G be a formula expressing the goal states.

Then a plan of length T exists iff

$$I@0 \land \bigwedge_{t=0}^{T-1} \mathcal{R}@t \land G_T$$

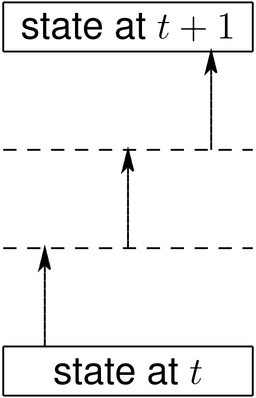
is satisfiable.

Remark

Most SAT solvers require formulas to be in CNF. There are efficient transformations to achieve this [Tse62, JS05, MV07].

Parallel Plans: Motivation

- Don't represent all intermediate states of a sequential plan.
- Ignore relative ordering of consecutive actions.
- Reduced number of explicitly represented states ⇒ smaller formulas



Parallel plans (∀-step plans)

Kautz and Selman 1996

Allow actions $a_1 = \langle p_1, e_1 \rangle$ and $a_2 = \langle p_2, e_2 \rangle$ in parallel whenever they don't interfere, i.e.

- both $p_1 \cup p_2$ and $e_1 \cup e_2$ are consistent, and
- both $e_1 \cup p_2$ and $e_2 \cup p_1$ are consistent.

Theorem

If $a_1 = \langle p_1, e_1 \rangle$ and $a_2 = \langle p_1, e_1 \rangle$ don't interfere and s is a state such that $s \models p_1$ and $s \models p_2$, then $\mathit{exec}_{a_1}(\mathit{exec}_{a_2}(s)) = \mathit{exec}_{a_2}(\mathit{exec}_{a_1}(s))$.

$\forall$-step plans: encoding

Define $\mathcal{R}^{\forall}@t$ as the conjunction of

$$x@(t+1) \leftrightarrow ((x@t \wedge \neg a_1@t \wedge \cdots \wedge \neg a_k@t) \vee a'_1@t \vee \cdots \vee a'_{k'}@t)$$

for all $x \in X$, where $a_1, \ldots, a_k$ are all actions making x false, and $a'_1, \ldots, a'_{k'}$ are all actions making x true, and

$$a@t \rightarrow l@t \text{ for all } l \text{ in the precondition of } a,$$

and

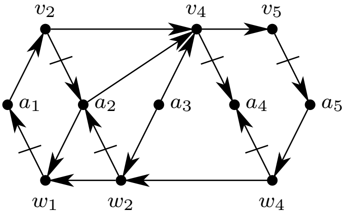
$$\neg(a@t \wedge a'@t) \text{ for all } a \text{ and } a' \text{ that interfere.}$$

This encoding is quadratic due to the interference clauses.

$\forall$-step plans: linear encoding

Rintanen et al. 2006 [RHN06]

Action a with effect l disables all actions with precondition $\bar{l}$, except a itself.
This is done in two parts: disable actions with higher index, disable actions with lower index.



This is needed for every literal.

$\exists$-step plans

Dimopoulos et al. 1997 [DNK97]

Allow actions $\{a_1, \ldots, a_n\}$ in parallel if they can be executed in at least one order.

- $\bigcup_{i=1}^{n} p_i$ is consistent.
- $\bigcup_{i=1}^{n} e_i$ is consistent.
- There is a total ordering $a_1, \ldots, a_n$ such that $e_i \cup p_j$ is consistent whenever $i \leq j$: disabling an action earlier in the ordering is allowed.

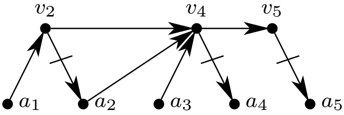
Several compact encodings exist [RHN06].
Fewer time steps are needed than with $\forall$-step plans. Sometimes only half as many.

∃-step plans: linear encoding

Rintanen et al. 2006 [RHN06]

Choose an arbitrary fixed ordering of all actions $a_1, \dots, a_n$.

Action a with effect l disables all later actions with precondition $\bar{l}$.



This is needed for every literal.

Disabling graphs

Rintanen et al. 2006 [RHN06]

Define a disabling graph with actions as nodes and with an arc from a_1 to a_2 (a_1 disables a_2) if $p_1 \cup p_2$ and $e_1 \cup e_2$ are consistent and $e_1 \cup p_2$ is inconsistent.

The test for valid execution orderings can be limited to strongly connected components (SCC) of the disabling graph.

In many structured problems all SCCs are singleton sets.
$\Longrightarrow$ No tests for validity of orderings needed during SAT solving.

Summary of Notions of Plans

plan type	reference	comment
sequential	[KS92]	one action per time point
∀-parallel	[BF97, KS96]	parallel actions independent
∃-parallel	[DNK97, RHN06]	executable in at least one order

The last two expressible in terms of the relation disables restricted to applied actions:

- ∀-parallel plans: the disables relation is empty.
- ∃-parallel plans: the disables relation is acyclic.

Search through Horizon Lengths

The planning problem is reduced to the satisfiability tests for

$$
\begin{aligned}
&\Phi_0 = I@0 \wedge G@0 \\
&\Phi_1 = I@0 \wedge \mathcal{R}@0 \wedge G@1 \\
&\Phi_2 = I@0 \wedge \mathcal{R}@0 \wedge \mathcal{R}@1 \wedge G@2 \\
&\Phi_3 = I@0 \wedge \mathcal{R}@0 \wedge \mathcal{R}@1 \wedge \mathcal{R}@2 \wedge G@3 \\
&\vdots \\
&\Phi_u = I@0 \wedge \mathcal{R}@0 \wedge \mathcal{R}@1 \wedge \cdots \mathcal{R}@(u-1) \wedge G@u
\end{aligned}
$$

where u is the maximum possible plan length.

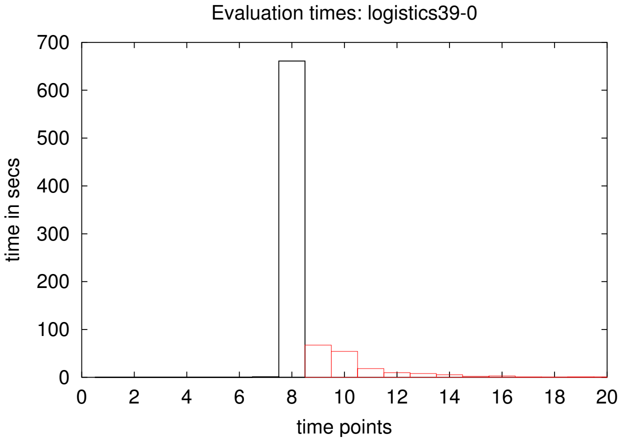
Q: How to schedule these satisfiability tests?

Search through Horizon Lengths

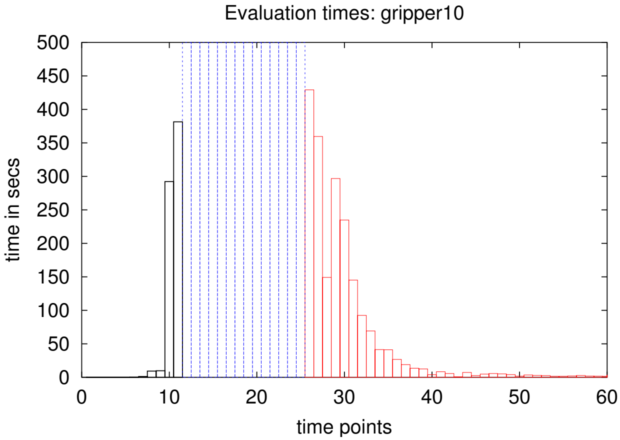
algorithm	reference	comment
sequential	[KS92, KS96]	slow, guarantees min. horizon
binary search	[SS07]	prerequisite: length UB
n processes	[Rin04b, Zar04]	fast, more memory needed
geometric	[Rin04b]	fast, more memory needed

- sequential: first test Φ_0, then Φ_1, then Φ_2, $\ldots$
  - This is breadth-first search / iterative deepening.
  - Guarantees shortest horizon length, but is slow.
- parallel strategies: solve several horizon lengths simultaneously
  - depth-first flavor
  - usually much faster
  - no guarantee of minimal horizon length

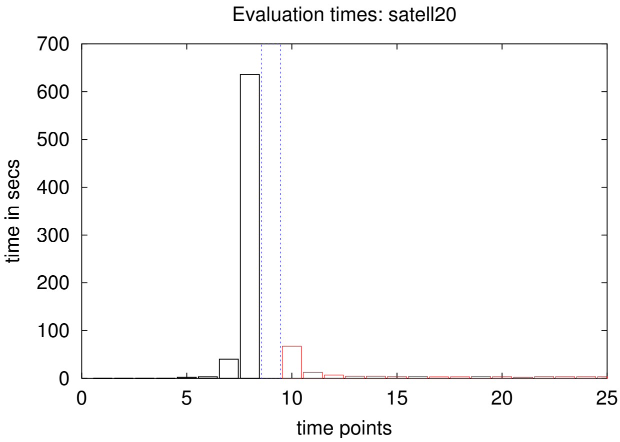
Some runtime profiles



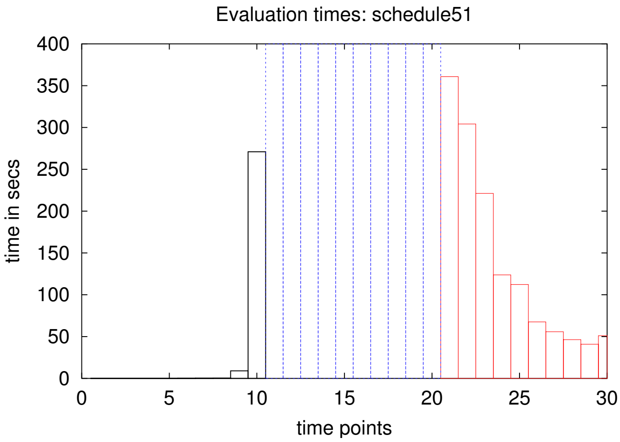
Some runtime profiles



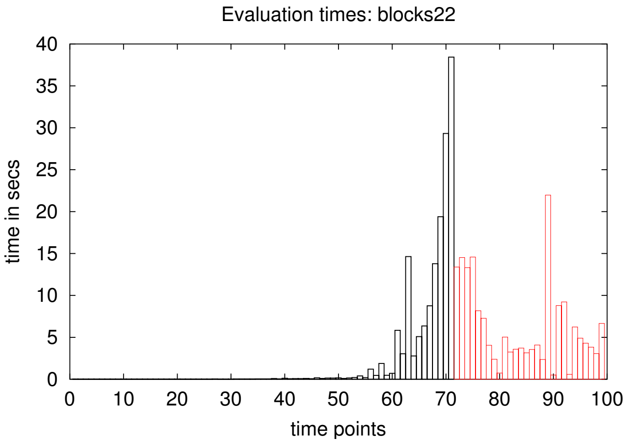
Some runtime profiles



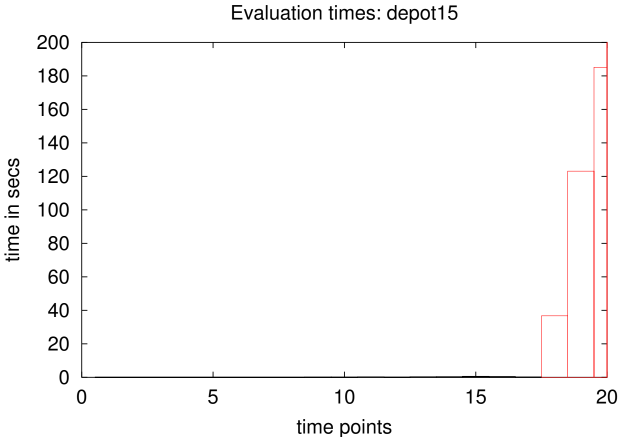
Some runtime profiles



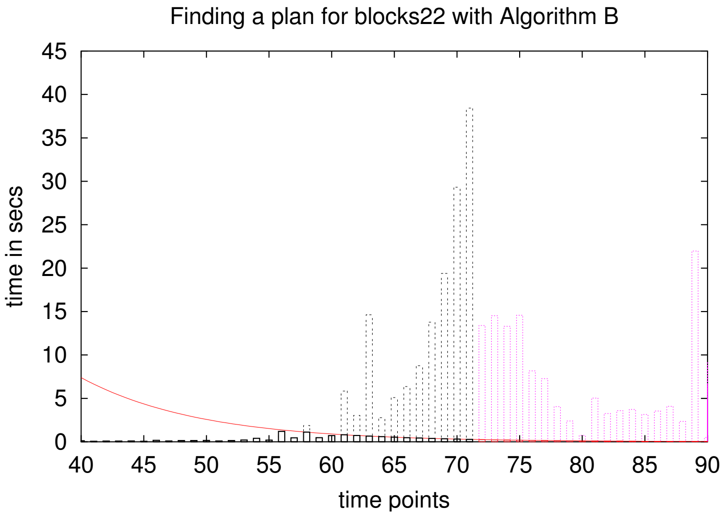
Some runtime profiles



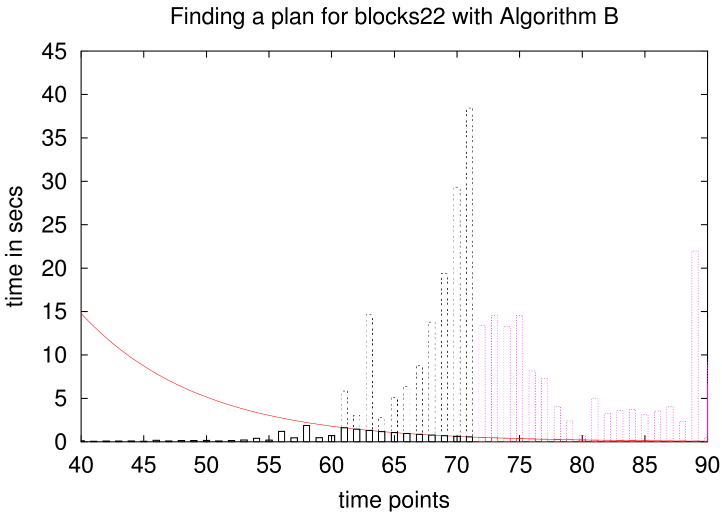
Some runtime profiles



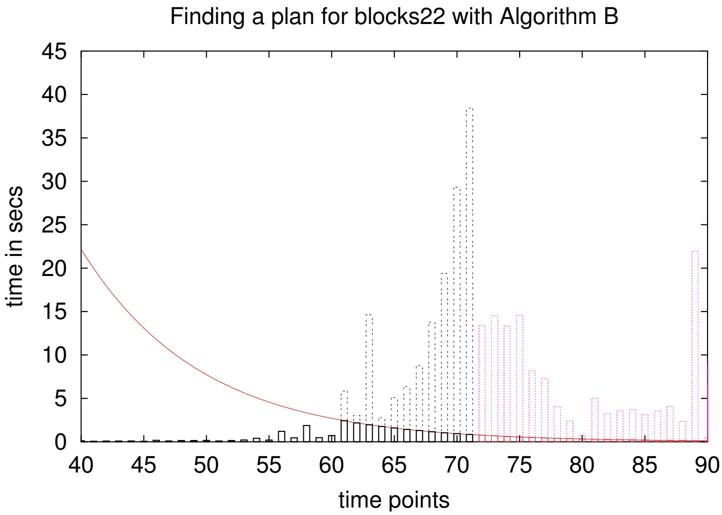
Geometric Evaluation



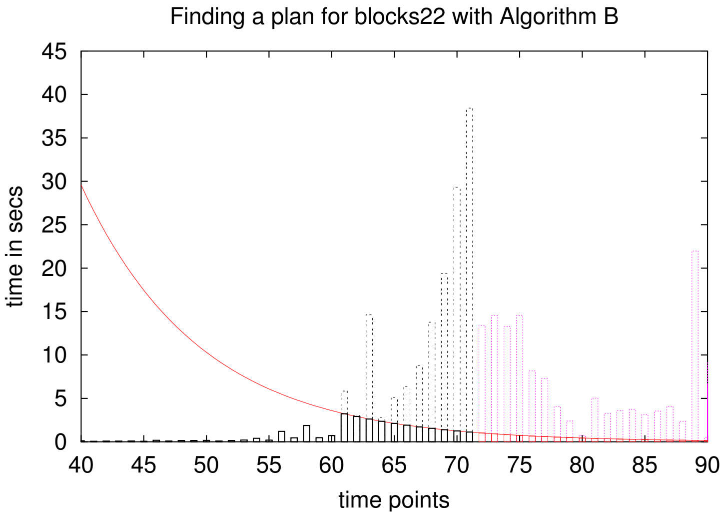
Geometric Evaluation



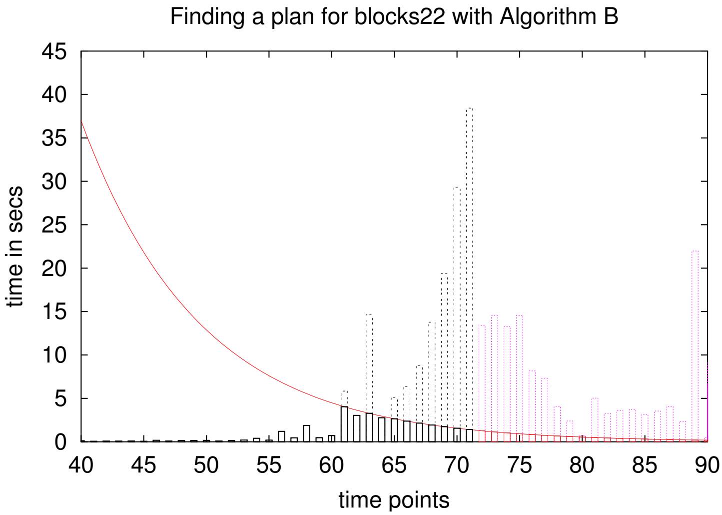
Geometric Evaluation



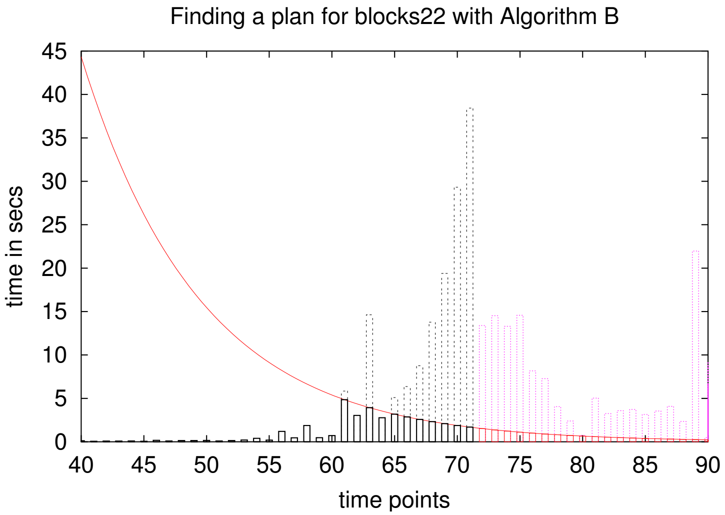
Geometric Evaluation



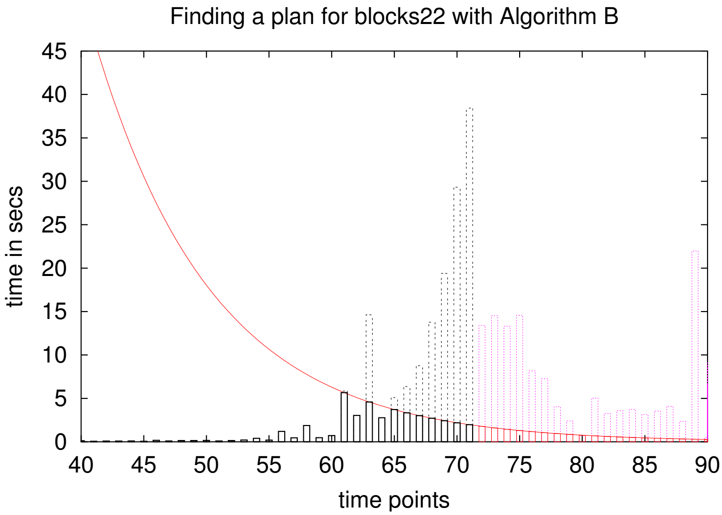
Geometric Evaluation



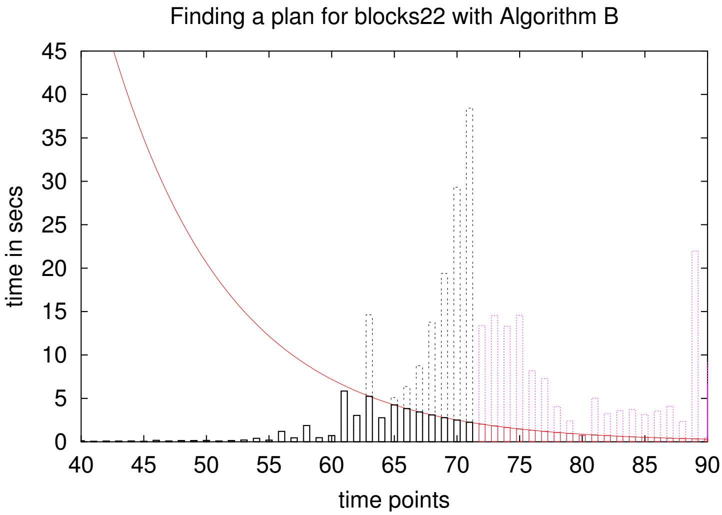
Geometric Evaluation



Geometric Evaluation



Geometric Evaluation



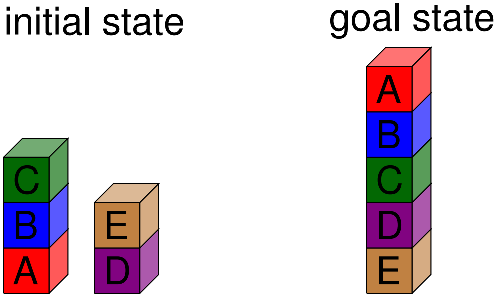
Solving the SAT Problem

SAT problems obtained from planning are solved by

- generic SAT solvers
    - Mostly based on Conflict-Driven Clause Learning (CDCL) [MMZ$^+$01].
    - Extremely good on hard combinatorial planning problems.
    - Not designed for solving the extremely large but “easy” formulas (arising in some types of benchmark problems).
- specialized SAT solvers [Rin10b, Rin10a]
    - Replace standard CDCL heuristics with planning-specific ones.
    - For certain problem classes substantial improvement
    - New research topic: lots of unexploited potential

Solving the SAT Problem

Example



Problem solved almost without search:

- Formulas for lengths 1 to 4 shown unsatisfiable without any search.
- Formula for plan length 5 is satisfiable: 3 nodes in the search tree.
- Plans have 5 to 7 operators, optimal plan has 5.

Solving the SAT Problem
Example

	0	1	2	3	4	5	0	1	2	3	4	5	0	1	2	3	4	5
clear(a)	F	F					F	F	F		T	T	F	F	F	T	T	T
clear(b)	F					F	F	F		T	T	F	F	F	T	T	T	F
clear(c)	T	T			F	F	T	T	T	T	F	F	T	T	T	T	F	F
clear(d)	F	T	T	F	F	F	F	T	T	F	F	F	F	T	T	F	F	F
clear(e)	T	T	F	F	F	F	T	T	F	F	F	F	T	T	F	F	F	F
on(a,b)	F	F	F			T	F	F	F	F	F	T	F	F	F	F	F	T
on(a,c)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(a,d)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(a,e)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(b,a)	T	T			F	F	T	T	T		F	F	T	T	T	F	F	F
on(b,c)	F	F			T	T	F	F	F	F	T	T	F	F	F	F	T	T
on(b,d)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(b,e)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(c,a)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(c,b)	T			F	F	F	T	T		F	F	F	T	T	F	F	F	F
on(c,d)	F	F	F	T	T	T	F	F	F	T	T	T	F	F	F	T	T	T
on(c,e)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(d,a)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(d,b)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(d,c)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(d,e)	F	F	T	T	T	T	F	F	T	T	T	T	F	F	T	T	T	T
on(e,a)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(e,b)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(e,c)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(e,d)	T	F	F	F	F	F	T	F	F	F	F	F	T	F	F	F	F	F
ontable(a)	T	T	T			F	T	T	T	T	T	F	T	T	T	T	T	F
ontable(b)	F	F			F	F	F	F	F		F	F	F	F	F	T	F	F
ontable(c)	F			F	F	F	F	F		F	F	F	F	F	T	F	F	F
ontable(d)	T	T	F	F	F	F	T	T	F	F	F	F	T	T	F	F	F	F
ontable(e)	F	T	T	T	T	T	F	T	T	T	T	T	F	T	T	T	T	T

1. State variable values inferred from initial values and goals.
2. Branch: $\neg$clear(b)^1.
3. Branch: clear(a)^3.
4. Plan found:

	0	1	2	3	4
fromtable(a,b)	F	F	F	F	T
fromtable(b,c)	F	F	F	T	F
fromtable(c,d)	F	F	T	F	F
fromtable(d,e)	F	T	F	F	F
totable(b,a)	F	F	T	F	F
totable(c,b)	F	T	F	F	F
totable(e,d)	T	F	F	F	F

Solving the SAT Problem
Example

	0	1	2	3	4	5	0	1	2	3	4	5	0	1	2	3	4	5
clear(a)	F	F					F	F	F		T	T	F	F	F	T	T	T
clear(b)	F					F	F	F		T	T	F	F	F	T	T	T	F
clear(c)	T	T			F	F	T	T	T	T	F	F	T	T	T	T	F	F
clear(d)	F	T	T	F	F	F	F	T	T	F	F	F	F	T	T	F	F	F
clear(e)	T	T	F	F	F	F	T	T	F	F	F	F	T	T	F	F	F	F
on(a,b)	F	F	F			T	F	F	F	F	F	T	F	F	F	F	F	T
on(a,c)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(a,d)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(a,e)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(b,a)	T	T			F	F	T	T	T		F	F	T	T	T	F	F	F
on(b,c)	F	F			T	T	F	F	F	F	T	T	F	F	F	F	T	T
on(b,d)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(b,e)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(c,a)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(c,b)	T			F	F	F	T	T		F	F	F	T	T	F	F	F	F
on(c,d)	F	F	F	T	T	T	F	F	F	T	T	T	F	F	F	T	T	T
on(c,e)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(d,a)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(d,b)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(d,c)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(d,e)	F	F	T	T	T	T	F	F	T	T	T	T	F	F	T	T	T	T
on(e,a)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(e,b)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(e,c)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(e,d)	T	F	F	F	F	F	T	F	F	F	F	F	T	F	F	F	F	F
ontable(a)	T	T	T			F	T	T	T	T	T	F	T	T	T	T	T	F
ontable(b)	F	F			F	F	F	F	F		F	F	F	F	F	T	F	F
ontable(c)	F			F	F	F	F	F		F	F	F	F	F	T	F	F	F
ontable(d)	T	T	F	F	F	F	T	T	F	F	F	F	T	T	F	F	F	F
ontable(e)	F	T	T	T	T	T	F	T	T	T	T	T	F	T	T	T	T	T

1. State variable values inferred from initial values and goals.
2. Branch: $\neg$clear(b)^1.
3. Branch: clear(a)^3.
4. Plan found:

	0	1	2	3	4
fromtable(a,b)	F	F	F	F	T
fromtable(b,c)	F	F	F	T	F
fromtable(c,d)	F	F	T	F	F
fromtable(d,e)	F	T	F	F	F
totable(b,a)	F	F	T	F	F
totable(c,b)	F	T	F	F	F
totable(e,d)	T	F	F	F	F

Solving the SAT Problem

Example

	0	1	2	3	4	5	0	1	2	3	4	5	0	1	2	3	4	5
clear(a)	F	F					F	F	F		T	T	F	F	F	T	T	T
clear(b)	F					F	F	F		T	T	F	F	F	T	T	T	F
clear(c)	T	T			F	F	T	T	T	T	F	F	T	T	T	T	F	F
clear(d)	F	T	T	F	F	F	F	T	T	F	F	F	F	T	T	F	F	F
clear(e)	T	T	F	F	F	F	T	T	F	F	F	F	T	T	F	F	F	F
on(a,b)	F	F	F			T	F	F	F	F	F	T	F	F	F	F	F	T
on(a,c)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(a,d)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(a,e)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(b,a)	T	T			F	F	T	T	T		F	F	T	T	T	F	F	F
on(b,c)	F	F			T	T	F	F	F	F	T	T	F	F	F	F	T	T
on(b,d)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(b,e)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(c,a)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(c,b)	T			F	F	F	T	T		F	F	F	T	T	F	F	F	F
on(c,d)	F	F	F	T	T	T	F	F	F	T	T	T	F	F	F	T	T	T
on(c,e)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(d,a)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(d,b)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(d,c)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(d,e)	F	F	T	T	T	T	F	F	T	T	T	T	F	F	T	T	T	T
on(e,a)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(e,b)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(e,c)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(e,d)	T	F	F	F	F	F	T	F	F	F	F	F	T	F	F	F	F	F
ontable(a)	T	T	T			F	T	T	T	T	T	F	T	T	T	T	T	F
ontable(b)	F	F			F	F	F	F	F		F	F	F	F	F	T	F	F
ontable(c)	F			F	F	F	F	F		F	F	F	F	F	T	F	F	F
ontable(d)	T	T	F	F	F	F	T	T	F	F	F	F	T	T	F	F	F	F
ontable(e)	F	T	T	T	T	T	F	T	T	T	T	T	F	T	T	T	T	T

1. State variable values inferred from initial values and goals.
2. Branch: $\neg$clear(b)^1.
3. Branch: clear(a)^3.
4. Plan found:

	0	1	2	3	4
fromtable(a,b)	F	F	F	F	T
fromtable(b,c)	F	F	F	T	F
fromtable(c,d)	F	F	T	F	F
fromtable(d,e)	F	T	F	F	F
totable(b,a)	F	F	T	F	F
totable(c,b)	F	T	F	F	F
totable(e,d)	T	F	F	F	F

Solving the SAT Problem

Example

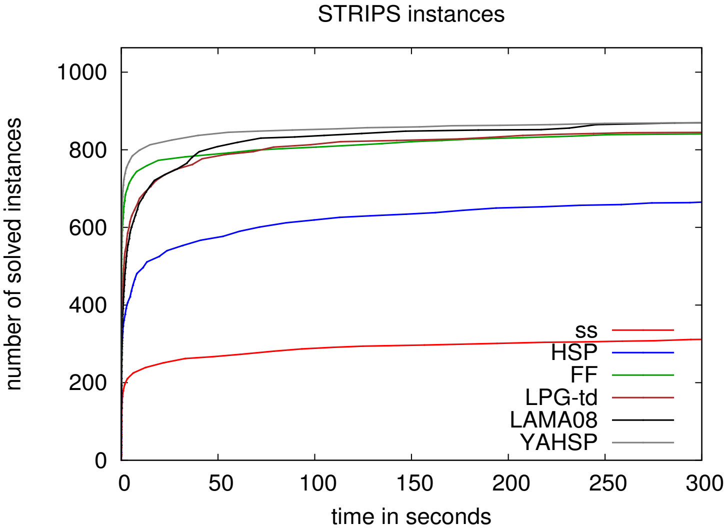
	0	1	2	3	4	5	0	1	2	3	4	5	0	1	2	3	4	5
clear(a)	F	F					F	F	F		T	T	F	F	F	T	T	T
clear(b)	F					F	F	F		T	T	F	F	F	T	T	T	F
clear(c)	T	T			F	F	T	T	T	T	F	F	T	T	T	T	F	F
clear(d)	F	T	T	F	F	F	F	T	T	F	F	F	F	T	T	F	F	F
clear(e)	T	T	F	F	F	F	T	T	F	F	F	F	T	T	F	F	F	F
on(a,b)	F	F	F			T	F	F	F	F	F	T	F	F	F	F	F	T
on(a,c)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(a,d)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(a,e)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(b,a)	T	T			F	F	T	T	T		F	F	T	T	T	F	F	F
on(b,c)	F	F			T	T	F	F	F	F	T	T	F	F	F	F	T	T
on(b,d)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(b,e)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(c,a)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(c,b)	T			F	F	F	T	T		F	F	F	T	T	F	F	F	F
on(c,d)	F	F	F	T	T	T	F	F	F	T	T	T	F	F	F	T	T	T
on(c,e)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(d,a)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(d,b)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(d,c)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(d,e)	F	F	T	T	T	T	F	F	T	T	T	T	F	F	T	T	T	T
on(e,a)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(e,b)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(e,c)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
on(e,d)	T	F	F	F	F	F	T	F	F	F	F	F	T	F	F	F	F	F
ontable(a)	T	T	T			F	T	T	T	T	T	F	T	T	T	T	T	F
ontable(b)	F	F			F	F	F	F	F		F	F	F	F	F	T	F	F
ontable(c)	F			F	F	F	F	F		F	F	F	F	F	T	F	F	F
ontable(d)	T	T	F	F	F	F	T	T	F	F	F	F	T	T	F	F	F	F
ontable(e)	F	T	T	T	T	T	F	T	T	T	T	T	F	T	T	T	T	T

1. State variable values inferred from initial values and goals.
2. Branch: $\neg$clear(b)^1.
3. Branch: clear(a)^3.
4. Plan found:

	0	1	2	3	4
fromtable(a,b)	F	F	F	F	T
fromtable(b,c)	F	F	F	T	F
fromtable(c,d)	F	F	T	F	F
fromtable(d,e)	F	T	F	F	F
totable(b,a)	F	F	T	F	F
totable(c,b)	F	T	F	F	F
totable(e,d)	T	F	F	F	F

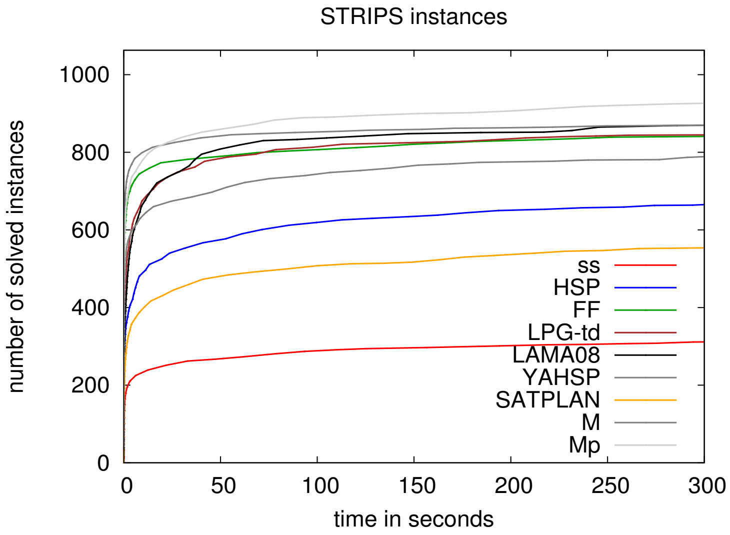
Performance of SAT-Based Planners

Planning Competition Problems 1998-2008



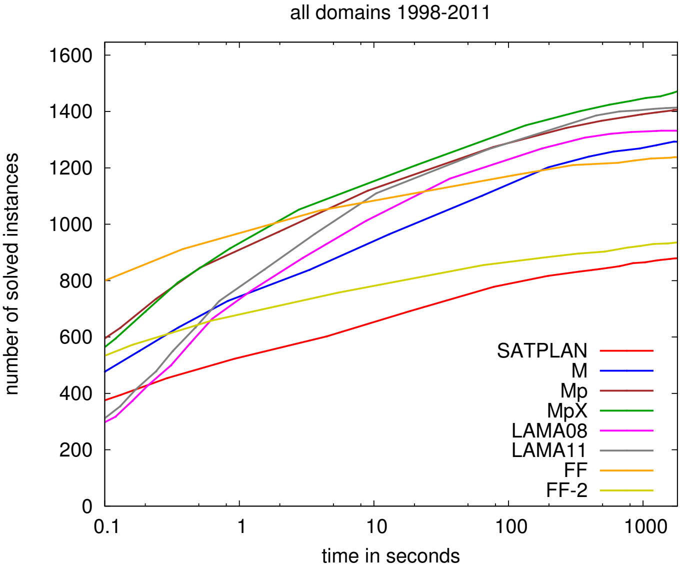
Performance of SAT-Based Planners

Planning Competition Problems 1998-2008



Performance of SAT-Based Planners

Planning Competition Problems 1998-2011 (revised)



Extensions

MathSAT [BBC$^+$05] and other SAT modulo Theories (SMT) solvers extend SAT with numerical variables and equalities and inequalities.
Applications include:

- timed systems [ACKS02], temporal planning
- hybrid systems [GPB05, ABCS05], temporal planning + continuous change

Symbolic Search Methods

Motivation

- logical formulas as a data structure for sets, relations
- Planning (model-checking, diagnosis, ...) algorithms in terms of set & relational operations.
- Algorithms that can handle very large state sets efficiently, bypassing inherent limitations of explicit state-space search.
- Complementary to explicit (enumerative) representations of state sets: strengths in different types of problems.

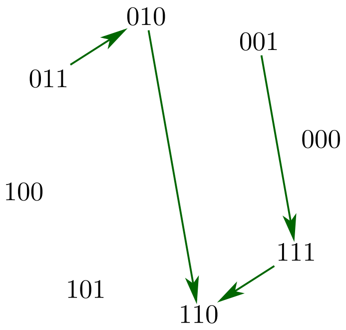
Transition relations in propositional logic

State variables are
$X = \{a, b, c\}$.

$$
\begin{array}{l}
(\neg a \wedge b \wedge c \wedge \neg a' \wedge b' \wedge \neg c')\vee \\
(\neg a \wedge b \wedge \neg c \wedge a' \wedge b' \wedge \neg c')\vee \\
(\neg a \wedge \neg b \wedge c \wedge a' \wedge b' \wedge c')\vee \\
(a \wedge b \wedge c \wedge a' \wedge b' \wedge \neg c')
\end{array}
$$

The corresponding matrix is

$$
\begin{array}{c|cccccccc}
 & 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \\
\hline
000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
001 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
010 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
011 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
100 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
101 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
110 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
111 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{array}
$$



Operations

The image of a set T of states w.r.t. action a is

$$\textit{img}_a(T) = \{s' \in S | s \in T, sas'\}.$$

The pre-image of a set T of states w.r.t. action a is

$$\textit{preimg}_a(T) = \{s \in S | s' \in T, sas'\}.$$

These operations reduce to the relational join and projection operations with a logic-representation of sets (unary relations) and binary relations.

Finding Plans with a Symbolic Algorithm

Computation of all reachable states

$$S_0 = \{I\}$$
$$S_{i+1} = S_i \cup \bigcup_{x \in X} \mathit{img}_x(S_i)$$

If $S_i = S_{i+1}$, then $S_j = S_i$ for all $j \geq i$, and the computation can be terminated.

- $S_i, i \geq 0$ is the set of states with distance $\leq i$ from the initial state.
- $S_i \backslash S_{i-1}, i \geq 1$ is the set of states with distance i.
- If $G \cap S_i$ for some $i \geq 0$, then there is a plan.

Action sequence recovered from sets S_i by a sequence of backward-chaining steps.

Use in Connection with Heuristic Search Algorithms

Symbolic (BDD) versions of heuristic algorithms in the state-space search context:

- SetA* [JVB08]
- BDDA* [ER98]
- ADDA* [HZF02]

Use in Connection with More General Problems

- BDDs and other normal forms standard representation in planning with partial observability [BCRT01, Rin05]. Also, probabilistic planning [HSAHB99] with value functions represented as Algebraic Decision Diagrams (ADD) [FMY97, BFG$^+$97].
- A belief state is a set of possible current states.
- These sets are often very large, best represented as formulas.

Significance of Symbolic Representations

- Much more powerful framework than SAT or explicit state-space search.
- Unlike other methods, allows exhaustive generation of reachable states.
- Problem 1: e.g. with BDDs, size of transition relation may explode.
- Problem 2: e.g. with BDDs, size of sets S_i may explode.
- Important research topic: symbolic search with less restrictive normal forms than BDD.

Images as Relational Operations

$$\begin{array}{|c|}\hline s_0 \\ s_2 \\ \hline\end{array} \bowtie \begin{array}{|cc|}\hline s_0 & s_1 \\ s_0 & s_2 \\ s_1 & s_0 \\ s_1 & s_2 \\ s_2 & s_0 \\ \hline\end{array} = \begin{array}{|cc|}\hline s_0 & \mathbf{s_1} \\ s_0 & \mathbf{s_2} \\ s_2 & \mathbf{s_0} \\ \hline\end{array}$$

Images as Relational Operations

$$\boxed{\begin{array}{l} s_0 \\ s_2 \end{array}} \bowtie \boxed{\begin{array}{ll} s_0 & s_1 \\ s_0 & s_2 \\ s_1 & s_0 \\ s_1 & s_2 \\ s_2 & s_0 \end{array}} = \boxed{\begin{array}{ll} s_0 & \mathbf{s_1} \\ s_0 & \mathbf{s_2} \\ s_2 & \mathbf{s_0} \end{array}}$$

$$\boxed{\begin{array}{l} s_0 00 \\ s_2 10 \end{array}} \bowtie \boxed{\begin{array}{l} s_0 s_1 00\ 01 \\ s_0 s_2 00\ 10 \\ s_1 s_0 01\ 00 \\ s_1 s_2 01\ 10 \\ s_2 s_0 10\ 00 \end{array}} = \boxed{\begin{array}{l} s_0 s_1 00\ 01 \\ s_0 s_2 00\ 10 \\ s_2 s_0 10\ 00 \end{array}}$$

Images as Relational Operations

$$\boxed{\begin{array}{l} s_0 00 \\ s_2 10 \end{array}} \bowtie \boxed{\begin{array}{l} s_0 s_1 00\ 01 \\ s_0 s_2 00\ 10 \\ s_1 s_0 01\ 00 \\ s_1 s_2 01\ 10 \\ s_2 s_0 10\ 00 \end{array}} = \boxed{\begin{array}{l} s_0 s_1 00\ 01 \\ s_0 s_2 00\ 10 \\ s_2 s_0 10\ 00 \end{array}}$$

$$\begin{array}{l|l} x_0 x_1 & \\ \hline 00 & 1 \\ 01 & 0 \\ 10 & 1 \\ 11 & 0 \end{array} \bowtie \begin{array}{l|l} x_0 x_1 x_0' x_1' & \\ \hline 0000 & 0 \\ 0001 & 1 \\ 0010 & 1 \\ 0011 & 0 \\ 0100 & 1 \\ 0101 & 0 \\ 0110 & 1 \\ 0111 & 0 \\ 1000 & 1 \\ 1001 & 0 \\ 1010 & 0 \\ 1011 & 0 \\ 1100 & 0 \\ 1101 & 0 \\ 1110 & 0 \\ 1111 & 0 \end{array} = \begin{array}{l|l} x_0 x_1 x_0' x_1' & \\ \hline 0001 & 1 \\ 0010 & 1 \\ 1000 & 1 \end{array}$$

Representation of Sets as Formulas

state sets	**formulas over X**
those $\frac{2^{\lvert X \rvert}}{2}$ states where x is true	$x \in X$
$\overline{E}$ (complement)	$\neg E$
$E \cup F$	$E \vee F$
$E \cap F$	$E \wedge F$
$E \setminus F$ (set difference)	$E \wedge \neg F$
the empty set $\emptyset$	$\perp$ (constant *false*)
the universal set	$\top$ (constant *true*)

question about sets	**question about formulas**
$E \subseteq F$?	$E \models F$?
$E \subset F$?	$E \models F$ and $F \not\models E$?
$E = F$?	$E \models F$ and $F \models E$?

Sets (of states) as formulas

Formulas over X represent sets

$a \vee b$ over $X = \{a, b, c\}$

represents the set $\{\overset{a\,b\,c}{010}, 011, 100, 101, 110, 111\}$.

Formulas over $X \cup X'$ represent binary relations

$a \wedge a' \wedge (b \leftrightarrow b')$ over $X \cup X'$ where $X = \{a, b\}$, $X' = \{a', b'\}$

represents the binary relation $\{(\overset{a\,b}{10}, \overset{a'b'}{1\,0}), (11, 11)\}$.

Valuations $\overset{a\,b\,a'b'}{1010}$ and 1111 of $X \cup X'$ can be viewed respectively as pairs of valuations $(\overset{a\,b}{10}, \overset{a'b'}{1\,0})$ and $(11, 11)$ of X.

Relation Operations

relation operation	logical operation
projection	abstraction
join	conjunction

Normal Forms

normal form	**reference**	**comment**
NNF Negation Normal Form		
DNF Disjunctive Normal Form		
CNF Conjunctive Normal Form		
BDD Binary Decision Diagram	[Bry92]	most popular
DNNF Decomposable NNF	[Dar01]	more compact

Darwiche's terminology: knowledge compilation languages [DM02]

Trade-off

- more compact $\mapsto$ less efficient operations
- But, "more efficient" is in the size of a correspondingly inflated formula. (Also more efficient in terms of wall clock?) BDD-SAT is $\mathcal{O}(1)$, but e.g. translation into BDDs is (usually) far less efficient than testing SAT directly.

Complexity of Operations

Operations offered e.g. by BDD packages:

	$\vee$	$\wedge$	$\neg$	$\phi \in$TAUT?	$\phi \in$SAT?	$\phi \equiv \phi'$?
NNF	poly	poly	poly	co-NP-hard	NP-hard	co-NP-hard
DNF	poly	exp	exp	co-NP-hard	in P	co-NP-hard
CNF	exp	poly	exp	in P	NP-hard	co-NP-hard
BDD	exp	exp	poly	in P	in P	in P

Remark

For BDDs one $\vee/\wedge$ *is polynomial time/size (size is doubled) but repeated* $\vee/\wedge$ *lead to exponential size.*

Existential and Universal Abstraction

Definition

Existential abstraction of a formula ϕ with respect to $x \in X$:

$$\exists x.\phi = \phi[\top/x] \vee \phi[\bot/x].$$

Universal abstraction is defined analogously by using conjunction instead of disjunction.

Definition

Universal abstraction of a formula ϕ with respect to $x \in X$:

$$\forall x.\phi = \phi[\top/x] \wedge \phi[\bot/x].$$

∃-Abstraction

Example

$$\begin{aligned}
&\exists b.((a \rightarrow b) \wedge (b \rightarrow c)) \\
&= ((a \rightarrow \top) \wedge (\top \rightarrow c)) \vee ((a \rightarrow \bot) \wedge (\bot \rightarrow c)) \\
&\equiv c \vee \neg a \\
&\equiv a \rightarrow c
\end{aligned}$$

$$\begin{aligned}
&\exists ab.(a \vee b) = \exists b.(\top \vee b) \vee (\bot \vee b) \\
&= ((\top \vee \top) \vee (\bot \vee \top)) \vee ((\top \vee \bot) \vee (\bot \vee \bot)) \\
&\equiv (\top \vee \top) \vee (\top \vee \bot) \equiv \top
\end{aligned}$$

$\forall$ and $\exists$-Abstraction in Terms of Truth-Tables

$\forall c$ and $\exists c$ correspond to combining lines with the same valuation for variables other than c.

Example

$$\exists c.(a \lor (b \land c)) \equiv a \lor b \qquad \forall c.(a \lor (b \land c)) \equiv a$$

a	b	c	$a \lor (b \land c)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

a	b	$\exists c.(a \lor (b \land c))$
0	0	0
0	1	1
1	0	1
1	1	1

a	b	$\forall c.(a \lor (b \land c))$
0	0	0
0	1	0
1	0	1
1	1	1

Encoding of Actions as Formulas

Let X be the set of all state variables. An action a corresponds to the conjunction of the precondition P_j and

$$x' \leftrightarrow F_i(X)$$

for all $x \in X$. Denote this by $\tau_X(a)$.

Example (move-from-A-to-B)

$$atA \wedge (atA' \leftrightarrow \bot) \wedge (atB' \leftrightarrow \top) \wedge (atC' \leftrightarrow atC) \wedge (atD' \leftrightarrow atD)$$

This is exactly the same as in the SAT case, except that we have x and x' instead of $x@t$ and $x@(t+1)$.

Computation of Successor States

Let

- $X = \{x_1, \dots, x_n\}$,
- $X' = \{x'_1, \dots, x'_n\}$,
- ϕ be a formula over X that represents a set T of states.

Image Operation

The image $\{s' \in S | s \in T, sas'\}$ of T with respect to a is

$$\mathit{img}_a(\phi) = (\exists X.(\phi \wedge \tau_X(a)))[X/X'].$$

The renaming is necessary to obtain a formula over X.

Computation of Predecessor States

Let

- $X = \{x_1, \dots, x_n\}$,
- $X' = \{x'_1, \dots, x'_n\}$,
- ϕ be a formula over X that represents a set T of states.

Preimage Operation

The pre-image $\{s \in S | s' \in T, sas'\}$ of T with respect to a is

$$\textit{preimg}_a(\phi) = (\exists X'.(\phi[X'/X] \wedge \tau_X(a))).$$

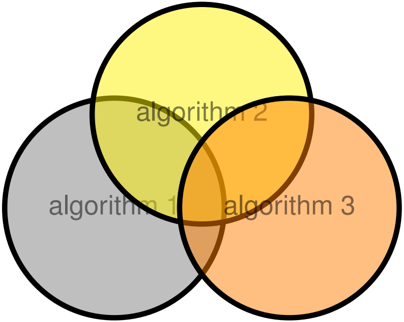
The renaming of ϕ is necessary so that we can start with a formula over X.

Engineering Efficient Planners

- Gap between Theory and Practice large: engineering details of implementation critical for performance in current planners.
- Few of the most efficient planners use textbook methods.
- Explanations for the observed differences between planners lacking: this is more art than science.

Algorithm Portfolios

- Algorithm portfolio = combination of two or more algorithms
- Useful if there is no single “strongest” algorithm.



Algorithm Portfolios
Composition methods

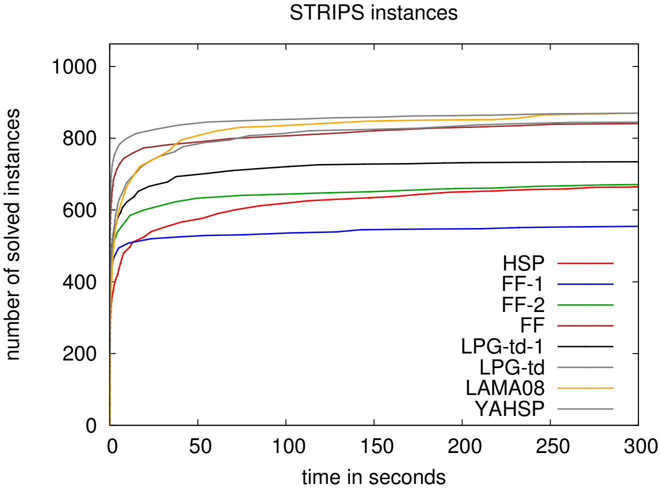
Composition methods:

- selection = choose one, for the instance in question
- parallel composition = run components in parallel
- sequential composition = run consecutively, according to a schedule

Examples: BLACKBOX [KS99], FF [HN01], LPG [GS02] (all use sequential composition)

Algorithm Portfolios

An Illustration of Portfolios



FF	=	FF-1 followed by FF-2
LPG-td	=	LPGT-td-1 followed by FF-2

Evaluation of Planners

Evaluation of planning systems is based on

- Hand-crafted problems (from the planning competitions)
  - This is the most popular option.
  - \+ Problems with (at least moderately) different structure.
  - \- Real-world relevance mostly low.
  - \- Instance generation uncontrolled: not known if easy or difficult.
  - \- Many have a similar structure: objects moving in a network.
- Benchmark sets obtained by translation from other problems
  - graph-theoretic problems: cliques, colorability, ... [PMB11]
- Instances sampled from all instances [Byl96].
  - \+ Easy to control problem hardness.
  - \- No direct real-world relevance (but: core of any “hard” problem)

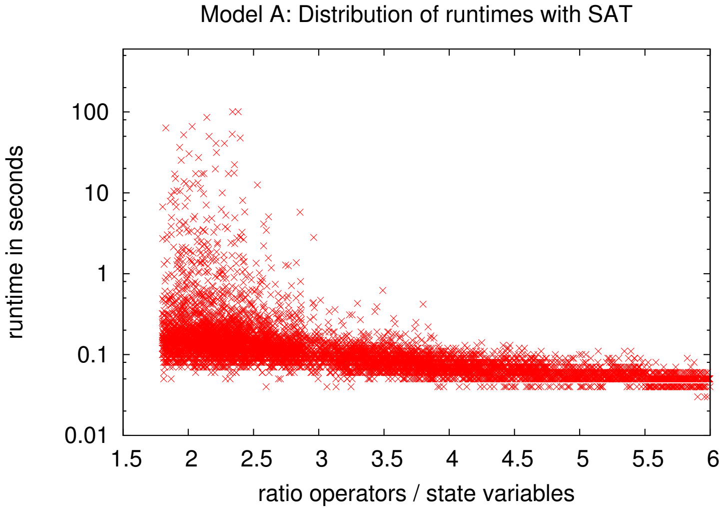
Sampling from the Set of All Instances

[Byl96, Rin04c]

- Generation:
  1. Fix number N of state variables, number M of actions.
  2. For each action, choose preconditions and effects randomly.
- Has a phase transition from unsolvable to solvable, similarly to SAT [MSL92] and connectivity of random graphs [Bol85].
- Exhibits an easy-hard-easy pattern, for a fixed N and an increasing M, analogously to SAT [MSL92].
- Hard instances roughly at the 50 per cent solvability point.
- Hardest instances are very hard: 20 state variables too difficult for many planners, as their heuristics don't help.

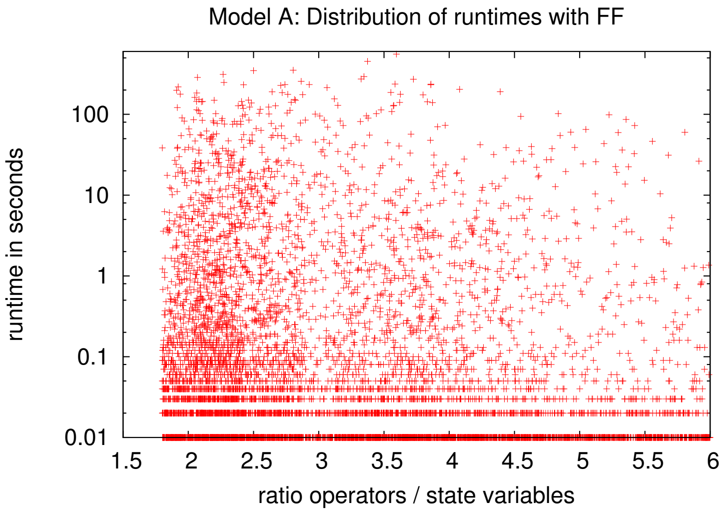
Sampling from the Set of All Instances

Experiments with planners



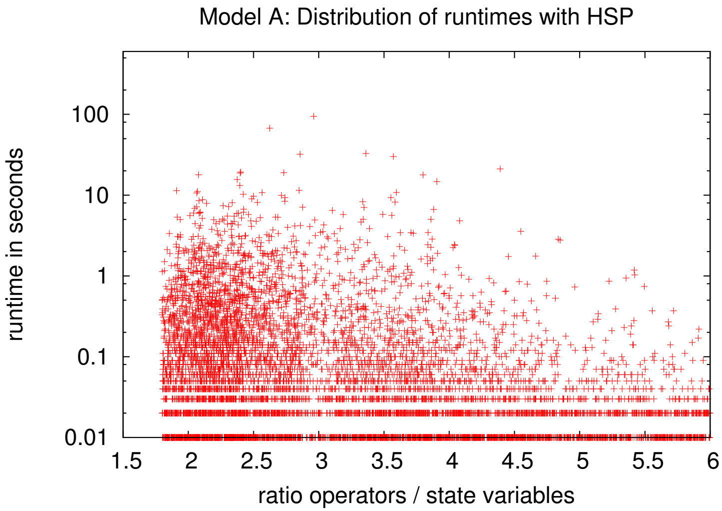
Sampling from the Set of All Instances

Experiments with planners



Sampling from the Set of All Instances

Experiments with planners



References I

Gilles Audemard, Marco Bozzano, Alessandro Cimatti, and Roberto Sebastiani.
Verifying industrial hybrid systems with MathSAT.
Electronic Notes in Theoretical Computer Science, 119(2):17–32, 2005.

Gilles Audemard, Alessandro Cimatti, Artur Korniłowicz, and Roberto Sebastiani.
Bounded model checking for timed systems.
In *Formal Techniques for Networked and Distributed Systems - FORTE 2002*, number 2529 in Lecture Notes in Computer Science, pages 243–259. Springer-Verlag, 2002.

Marco Bozzano, Roberto Bruttomesso, Alessandro Cimatti, Tommi Junttila, Peter van Rossum, Stephan Schulz, and Roberto Sebastiani.
The MathSAT 3 system.
In *Automated Deduction - CADE-20*, volume 3632 of *Lecture Notes in Computer Science*, pages 315–321. Springer-Verlag, 2005.

Piergiorgio Bertoli, Alessandro Cimatti, Marco Roveri, and Paolo Traverso.
Planning in nondeterministic domains under partial observability via symbolic model checking.
In Bernhard Nebel, editor, *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pages 473–478. Morgan Kaufmann Publishers, 2001.

Avrim L. Blum and Merrick L. Furst.
Fast planning through planning graph analysis.
Artificial Intelligence, 90(1-2):281–300, 1997.

References II

R. I. Bahar, E. A. Frohm, C. M. Gaona, G. D. Hachtel, E. Macii, A. Pardo, and F. Somenzi.
Algebraic decision diagrams and their applications.
Formal Methods in System Design: An International Journal, 10(2/3):171–206, 1997.

Blai Bonet and Héctor Geffner.
Planning as heuristic search.
Artificial Intelligence, 129(1-2):5–33, 2001.

B. Bollobás.
Random graphs.
Academic Press, 1985.

R. E. Bryant.
Symbolic Boolean manipulation with ordered binary decision diagrams.
ACM Computing Surveys, 24(3):293–318, September 1992.

Tom Bylander.
The computational complexity of propositional STRIPS planning.
Artificial Intelligence, 69(1-2):165–204, 1994.

Tom Bylander.
A probabilistic analysis of propositional STRIPS planning.
Artificial Intelligence, 81(1-2):241–271, 1996.

References III

S. A. Cook.
The complexity of theorem proving procedures.
In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.

Joseph C. Culberson and Jonathan Schaeffer.
Searching with pattern databases.
In Gordon I. McCalla, editor, *Advances in Artificial Intelligence, 11th Biennial Conference of the Canadian Society for Computational Studies of Intelligence, AI '96, Toronto, Ontario, Canada, May 21-24, 1996, Proceedings*, volume 1081 of *Lecture Notes in Computer Science*, pages 402–416. Springer-Verlag, 1996.

Adnan Darwiche.
Decomposable negation normal form.
Journal of the ACM, 48(4):608–647, 2001.

Klaus Dräger, Bernd Finkbeiner, and Andreas Podelski.
Directed model checking with distance-preserving abstractions.
International Journal on Software Tools for Technology Transfer, 11(1):27–37, 2009.

Minh Binh Do and Subbarao Kambhampati.
Planning as constraint satisfaction: Solving the planning graph by compiling it into CSP.
Artificial Intelligence, 132(2):151–182, 2001.

References IV

Adnan Darwiche and Pierre Marquis.
A knowledge compilation map.
Journal of Artificial Intelligence Research, 17:229–264, 2002.

Yannis Dimopoulos, Bernhard Nebel, and Jana Koehler.
Encoding planning problems in nonmonotonic logic programs.
In S. Steel and R. Alami, editors, *Recent Advances in AI Planning. Fourth European Conference on Planning (ECP'97)*, number 1348 in Lecture Notes in Computer Science, pages 169–181. Springer-Verlag, 1997.

G. Dueck and T. Scheuer.
Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing.
Journal of Computational Physics, 90:161–175, 1990.

Stefan Edelkamp.
Planning with pattern databases.
In *Proceedings of the 6th European Conference on Planning (ECP-01)*, pages 13–24, 2000.
Unpublished.

Stefan Edelkamp and Frank Reffel.
OBDDs in heuristic search.
In *KI-98: Advances in Artificial Intelligence*, number 1504 in Lecture Notes in Computer Science, pages 81–92. Springer-Verlag, 1998.

References V

E. Allen Emerson and A. Prasad Sistla.
Symmetry and model-checking.
Formal Methods in System Design: An International Journal, 9(1/2):105–131, 1996.

M. Fujita, P. C. McGeer, and J. C.-Y. Yang.
Multi-terminal binary decision diagrams: an efficient data structure for matrix representation.
Formal Methods in System Design: An International Journal, 10(2/3):149–169, 1997.

Fred Glover.
Tabu search – part I.
ORSA Journal on Computing, 1(3):190–206, 1989.

P. Godefroid.
Using partial orders to improve automatic verification methods.
In Kim Guldstrand Larsen and Arne Skou, editors, *Proceedings of the 2nd International Conference on Computer-Aided Verification (CAV '90), Rutgers, New Jersey, 1990*, number 531 in Lecture Notes in Computer Science, pages 176–185. Springer-Verlag, 1991.

Nicolò Giorgetti, George J. Pappas, and Alberto Bemporad.
Bounded model checking of hybrid dynamical systems.
In *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference 2005*, pages 672–677. IEEE, 2005.

References VI

Alfonso Gerevini and Ivan Serina.
LPG: a planner based on local search for planning graphs with action costs.
In Malik Ghallab, Joachim Hertzberg, and Paolo Traverso, editors, *Proceedings of the Sixth International Conference on Artificial Intelligence Planning Systems, April 23-27, 2002, Toulouse, France*, pages 13–22. AAAI Press, 2002.

Hana Galperin and Avi Wigderson.
Succinct representations of graphs.
Information and Control, 56:183–198, 1983.
See [Loz88] for a correction.

Malte Helmert and Carmel Domshlak.
Landmarks, critical paths and abstractions: What's the difference anyway.
In Alfonso Gerevini, Adele Howe, Amedeo Cesta, and Ioannis Refanidis, editors, *ICAPS 2009. Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling*, pages 162–169. AAAI Press, 2009.

Malte Helmert, Patrik Haslum, and Joerg Hoffmann.
Flexible abstraction heuristics for optimal sequential planning.
In *ICAPS 2007. Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling*, pages 176–183. AAAI Press, 2007.

J. Hoffmann and B. Nebel.
The FF planning system: fast plan generation through heuristic search.
Journal of Artificial Intelligence Research, 14:253–302, 2001.

References VII

P. E. Hart, N. J. Nilsson, and B. Raphael.
A formal basis for the heuristic determination of minimum-cost paths.
IEEE Transactions on System Sciences and Cybernetics, SSC-4(2):100–107, 1968.

Jesse Hoey, Robert St-Aubin, Alan Hu, and Craig Boutilier.
SPUDD: Stochastic planning using decision diagrams.
In Kathryn B. Laskey and Henri Prade, editors, *Uncertainty in Artificial Intelligence, Proceedings of the Fifteenth Conference (UAI-99)*, pages 279–288. Morgan Kaufmann Publishers, 1999.

E. Hansen, R. Zhou, and Z. Feng.
Symbolic heuristic search using decision diagrams.
In *Abstraction, Reformulation, and Approximation*, pages 83–98. Springer-Verlag, 2002.

Paul Jackson and Daniel Sheridan.
Clause form conversions for Boolean circuits.
In Holger H. Hoos and David G. Mitchell, editors, *Theory and Applications of Satisfiability Testing, 7th International Conference, SAT 2004, Vancouver, BC, Canada, May 10-13, 2004, Revised Selected Papers*, volume 3542 of *Lecture Notes in Computer Science*, pages 183–198. Springer-Verlag, 2005.

R. M. Jensen, M. M. Veloso, and R. E. Bryant.
State-set branching: Leveraging BDDs for heuristic search.
Artificial Intelligence, 172(2-3):103–139, 2008.

References VIII

S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi.
Optimization by simulated annealing.
Science, 220(4598):671–680, May 1983.

R. E. Korf.
Depth-first iterative deepening: an optimal admissible tree search.
Artificial Intelligence, 27(1):97–109, 1985.

Henry Kautz and Bart Selman.
Planning as satisfiability.
In Bernd Neumann, editor, *Proceedings of the 10th European Conference on Artificial Intelligence*, pages 359–363. John Wiley & Sons, 1992.

Henry Kautz and Bart Selman.
Pushing the envelope: planning, propositional logic, and stochastic search.
In *Proceedings of the 13th National Conference on Artificial Intelligence and the 8th Innovative Applications of Artificial Intelligence Conference*, pages 1194–1201. AAAI Press, 1996.

Henry Kautz and Bart Selman.
Unifying SAT-based and graph-based planning.
In Thomas Dean, editor, *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 318–325. Morgan Kaufmann Publishers, 1999.

References IX

Antonio Lozano and José L. Balcázar.
The complexity of graph problems for succinctly represented graphs.
In Manfred Nagl, editor, *Graph-Theoretic Concepts in Computer Science, 15th International Workshop, WG'89*, number 411 in Lecture Notes in Computer Science, pages 277–286. Springer-Verlag, 1990.

Michael L. Littman.
Probabilistic propositional planning: Representations and complexity.
In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-97) and 9th Innovative Applications of Artificial Intelligence Conference (IAAI-97)*, pages 748–754. AAAI Press, 1997.

Antonio Lozano.
NP-hardness of succinct representations of graphs.
Bulletin of the European Association for Theoretical Computer Science, 35:158–163, June 1988.

Drew McDermott.
A heuristic estimator for means-ends analysis in planning.
In Brian Drabble, editor, *Proceedings of the Third International Conference on Artificial Intelligence Planning Systems*, pages 142–149. AAAI Press, 1996.

Drew McDermott.
The Planning Domain Definition Language.
Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, Yale University, October 1998.

References X

Omid Madani, Steve Hanks, and Anne Condon.
On the undecidability of probabilistic planning and related stochastic optimization problems.
Artificial Intelligence, 147(1–2):5–34, 2003.

Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik.
Chaff: engineering an efficient SAT solver.
In *Proceedings of the 38th ACM/IEEE Design Automation Conference (DAC'01)*, pages 530–535. ACM Press, 2001.

David A. McAllester and David Rosenblitt.
Systematic nonlinear planning.
In *Proceedings of the 9th National Conference on Artificial Intelligence*, volume 2, pages 634–639. AAAI Press / The MIT Press, 1991.

David Mitchell, Bart Selman, and Hector Levesque.
Hard and easy distributions of SAT problems.
In William Swartout, editor, *Proceedings of the 10th National Conference on Artificial Intelligence*, pages 459–465. The MIT Press, 1992.

Panagiotis Manolios and Daron Vroon.
Efficient circuit to CNF conversion.
In Joao Marques-Silva and Karem A. Sakallah, editors, *Proceedings of the 8th International Conference on Theory and Applications of Satisfiability Testing (SAT-2007)*, volume 4501 of *Lecture Notes in Computer Science*, pages 4–9. Springer-Verlag, 2007.

References XI

Aldo Porco, Alejandro Machado, and Blai Bonet.
Automatic polytime reductions of NP problems into a fragment of STRIPS.
In *ICAPS 2011. Proceedings of the Twenty-First International Conference on Automated Planning and Scheduling*, pages 178–185. AAAI Press, 2011.

S. Richter and M. Helmert.
Preferred operators and deferred evaluation in satisficing planning.
In *ICAPS 2009. Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling*, pages 273–280, 2009.

Jussi Rintanen, Keijo Heljanko, and Ilkka Niemelä.
Planning as satisfiability: parallel plans and algorithms for plan search.
Artificial Intelligence, 170(12-13):1031–1080, 2006.

Jussi Rintanen.
A planning algorithm not based on directional search.
In A. G. Cohn, L. K. Schubert, and S. C. Shapiro, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixth International Conference (KR '98)*, pages 617–624. Morgan Kaufmann Publishers, 1998.

Jussi Rintanen.
Complexity of planning with partial observability.
In Shlomo Zilberstein, Jana Koehler, and Sven Koenig, editors, *ICAPS 2004. Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling*, pages 345–354. AAAI Press, 2004.

References XII

Jussi Rintanen.
Evaluation strategies for planning as satisfiability.
In Ramon López de Mántaras and Lorenza Saitta, editors, *ECAI 2004. Proceedings of the 16th European Conference on Artificial Intelligence*, pages 682–687. IOS Press, 2004.

Jussi Rintanen.
Phase transitions in classical planning: an experimental study.
In Shlomo Zilberstein, Jana Koehler, and Sven Koenig, editors, *ICAPS 2004. Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling*, pages 101–110. AAAI Press, 2004.

Jussi Rintanen.
Conditional planning in the discrete belief space.
In Leslie Pack Kaelbling, editor, *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pages 1260–1265. Morgan Kaufmann Publishers, 2005.

Jussi Rintanen.
Complexity of concurrent temporal planning.
In *ICAPS 2007. Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling*, pages 280–287. AAAI Press, 2007.

References XIII

Jussi Rintanen.
Regression for classical and nondeterministic planning.
In Malik Ghallab, Constantine D. Spyropoulos, and Nikos Fakotakis, editors, *ECAI 2008. Proceedings of the 18th European Conference on Artificial Intelligence*, pages 568–571. IOS Press, 2008.

Jussi Rintanen.
Heuristic planning with SAT: beyond uninformed depth-first search.
In Jiuyong Li, editor, *AI 2010 : Advances in Artificial Intelligence: 23rd Australasian Joint Conference on Artificial Intelligence, Adelaide, South Australia, December 7-10, 2010, Proceedings*, number 6464 in Lecture Notes in Computer Science, pages 415–424. Springer-Verlag, 2010.

Jussi Rintanen.
Heuristics for planning with SAT.
In David Cohen, editor, *Principles and Practice of Constraint Programming - CP 2010, 16th International Conference, CP 2010, St. Andrews, Scotland, September 2010, Proceedings.*, number 6308 in Lecture Notes in Computer Science, pages 414–428. Springer-Verlag, 2010.

Matthew Streeter and Stephen F. Smith.
Using decision procedures efficiently for optimization.
In *ICAPS 2007. Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling*, pages 312–319. AAAI Press, 2007.

References XIV

Meera Sampath, Raja Sengupta, Stéphane Lafortune, Kasim Sinnamohideen, and Demosthenis Teneketzis.
Diagnosability of discrete-event systems.
IEEE Transactions on Automatic Control, 40(9):1555–1575, 1995.

P. H. Starke.
Reachability analysis of Petri nets using symmetries.
Journal of Mathematical Modelling and Simulation in Systems Analysis, 8(4/5):293–303, 1991.

G. S. Tseitin.
On the complexity of derivation in propositional calculus.
In A. O. Slisenko, editor, *Studies in Constructive Mathematics and Mathematical Logic, Part 2*, pages 115–125. Consultants Bureau, New York - London, 1962.

Antti Valmari.
Stubborn sets for reduced state space generation.
In Grzegorz Rozenberg, editor, *Advances in Petri Nets 1990. 10th International Conference on Applications and Theory of Petri Nets, Bonn, Germany*, number 483 in Lecture Notes in Computer Science, pages 491–515. Springer-Verlag, 1991.

Peter van Beek and Xinguang Chen.
CPlan: a constraint programming approach to planning.
In *Proceedings of the 16th National Conference on Artificial Intelligence (AAAI-99) and the 11th Conference on Innovative Applications of Artificial Intelligence (IAAI-99)*, pages 585–590. AAAI Press, 1999.

References XV

Vincent Vidal.
A lookahead strategy for heuristic search planning.
In Shlomo Zilberstein, Jana Koehler, and Sven Koenig, editors, *ICAPS 2004. Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling*, pages 150–160. AAAI Press, 2004.

Emmanuel Zarpas.
Simple yet efficient improvements of SAT based bounded model checking.
In Alan J. Hu and Andrew K. Martin, editors, *Formal Methods in Computer-Aided Design: 5th International Conference, FMCAD 2004, Austin, Texas, USA, November 15-17, 2004. Proceedings*, number 3312 in Lecture Notes in Computer Science, pages 174–185. Springer-Verlag, 2004.