Search Methods for Classical and Temporal Planning

Jussi Rintanen

Prague, ECAI 2014

- 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)

- 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

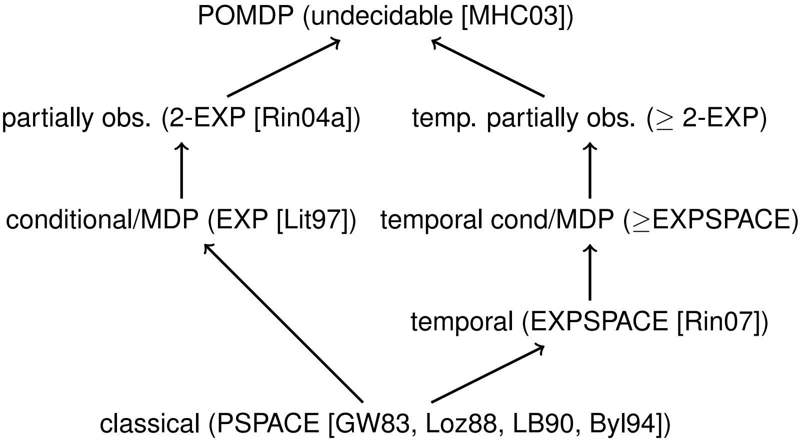
Time

  - 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



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 [GW83, Loz88, LB90, Byl94].
With a polynomial bound on plan length, NP-complete [KS96].

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
- Often very high performance penalty w.r.t. specialized algorithms
- Trade-off between generality and efficiency

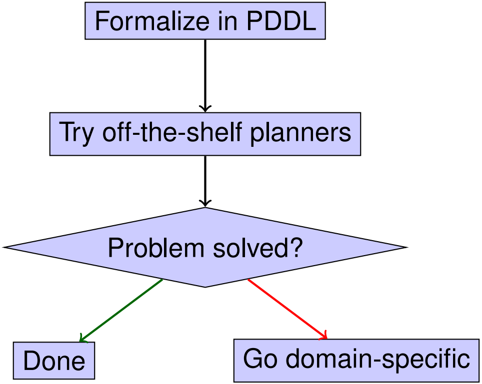
Domain-Specific Planning

What is domain-specific?

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

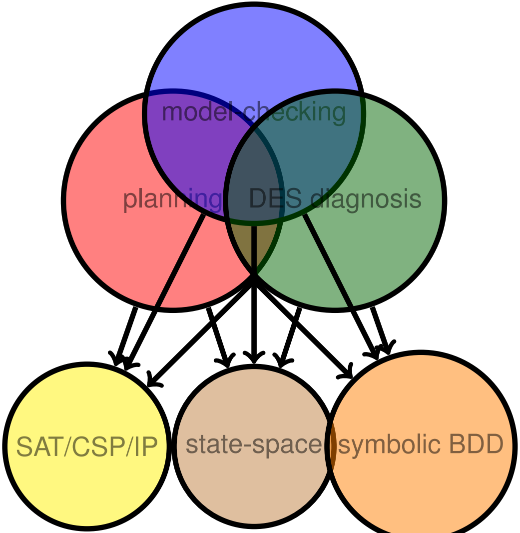
There are some planning systems that have aspects of these, but mostly this means: implement everything from scratch.

Domain-Dependent vs. -Independent Planning
Procedure

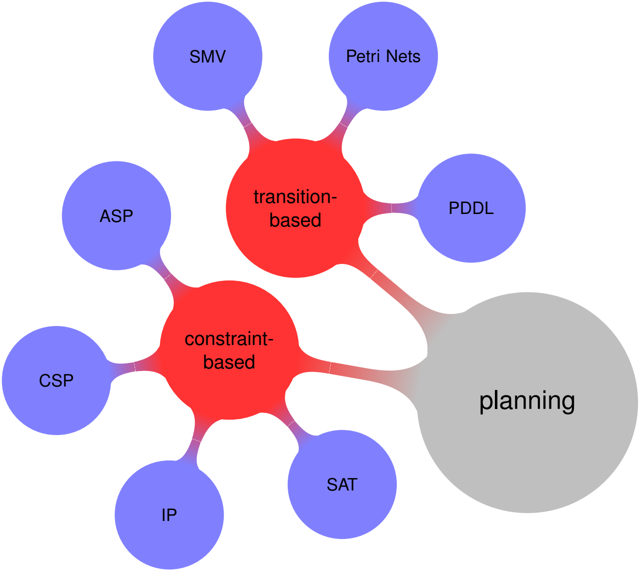


Related Problems, Reductions

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



How to Represent Planning Problems?



PDDL: Planning Domain Description Language

- Defined in 1998 [GHK+98], with several extensions later.
- Lisp-style syntax
- Widely used in the planning (competition) community.
- Most basic version with Boolean state variables only.
- Action sets expressed as schemata instantiated with objects.

```
(:action unload
    :parameters (?obj - obj ?airplane - vehicle ?loc - location)
    :precondition (and (in ?obj ?airplane) (at ?airplane ?loc))
    :effect (and (not (in ?obj ?airplane))))
```

States

States are valuations of state variables.

Example

State variables are

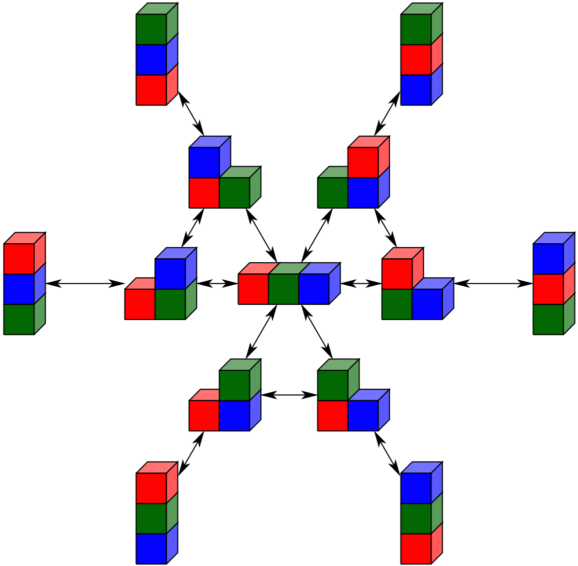
LOCATION:	$\{0,\ldots,1000\}$
GEAR:	$\{R,1,2,3,4,5\}$
FUEL:	$\{0,\ldots,60\}$
SPEED:	$\{-20,\ldots,200\}$
DIRECTION:	$\{0,\ldots,359\}$

One state is

LOCATION =	312
GEAR =	4
FUEL =	58
SPEED =	110
DIRECTION =	90

State-space transition graphs

Blocks world with three blocks



Actions

How values of state variables change

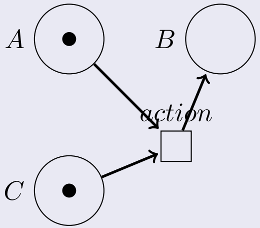
General form

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, inductive definitions.
- 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)

(We will later extend this with time and continuous change.)

The planning problem

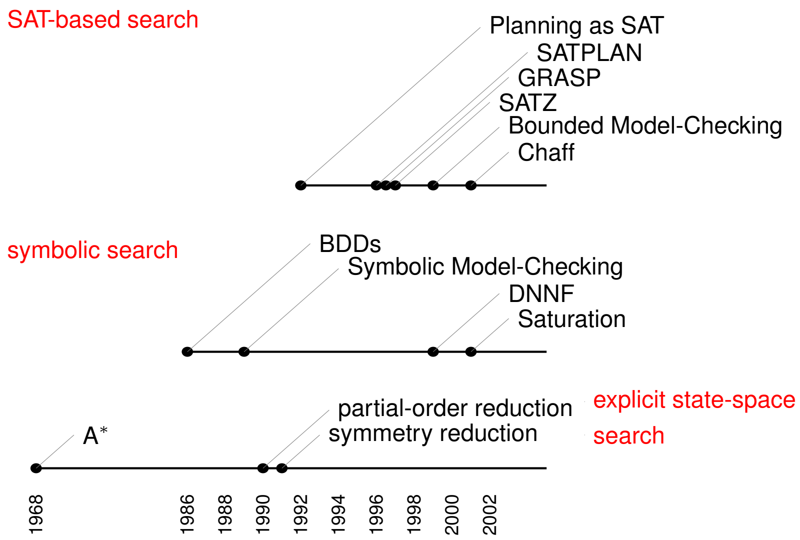
An action $a = \langle p, e \rangle$ is executable in state s iff $s \models p$.
The successor state $s' = \textit{exec}_a(s)$ is defined by

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

Problem

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

Development of state-space search methods



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
- Also known as "forward search" (in contrast to "backward search" with regression [Rin08])
- Pruning methods:
  - symmetry reduction [Sta91, ES96]
  - partial-order reduction [God91, Val91]
  - lower-bounds / heuristics, for informed search [HNR68]

State Representation

Every state represented explicitly $\Rightarrow$ 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

(See [GV03] for references to representative works on compact representations of state sets.)

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

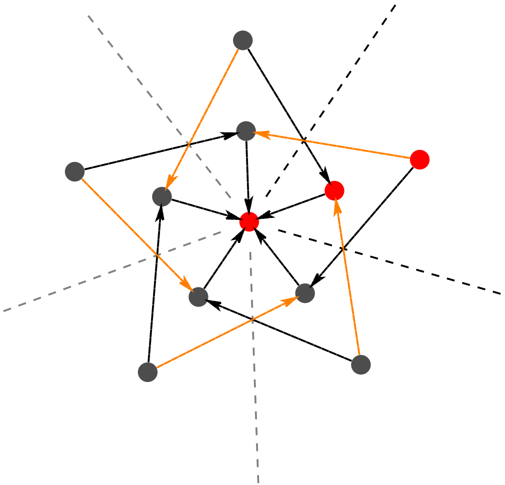
1. Define an equivalence relation $\sim$ on the set of all states: $s_1 \sim s_2$ means that state s_1 is symmetric with s_2.
2. Only one state s_C in each equivalence class $[s_C]$ needs to be considered.
3. If state $s \in [s_C]$ with $s \neq s_C$ is encountered, replace it with s_C.

Example

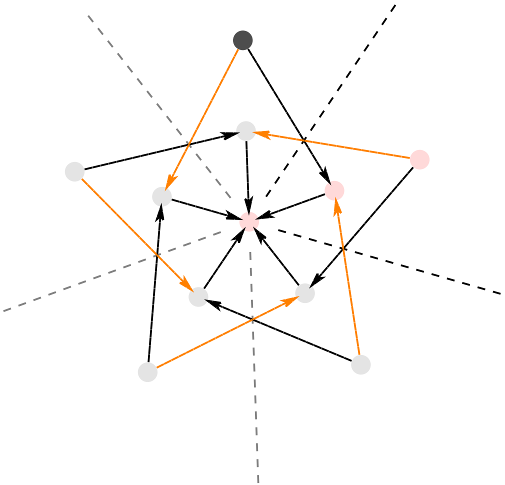
States $P(A) \wedge \neg P(B) \wedge P(C)$ and $\neg P(A) \wedge P(B) \wedge P(C)$ are symmetric because of the permutation $A \mapsto B, B \mapsto A, C \mapsto C$.

Symmetry Reduction

Example: 11 states, 3 equivalence classes

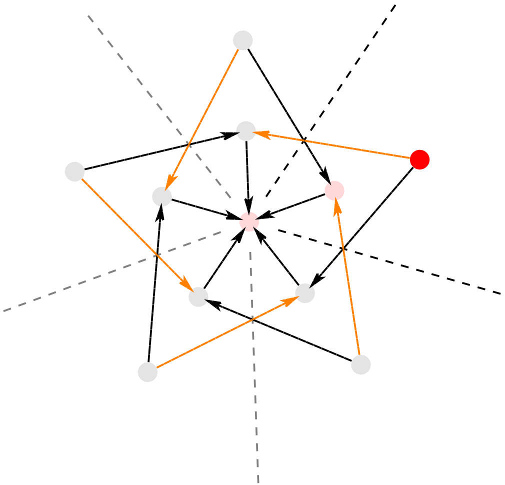


Symmetry Reduction
Example: 11 states, 3 equivalence classes

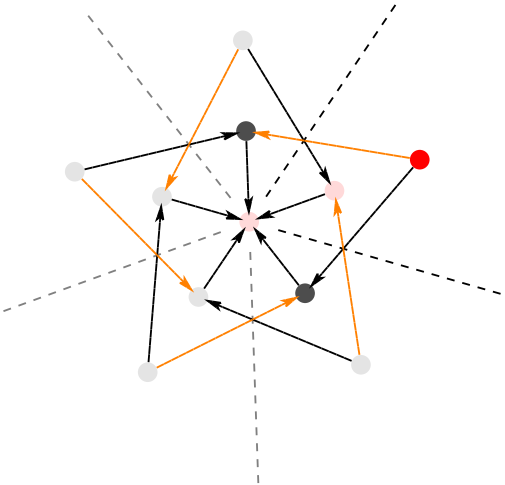


Symmetry Reduction

Example: 11 states, 3 equivalence classes

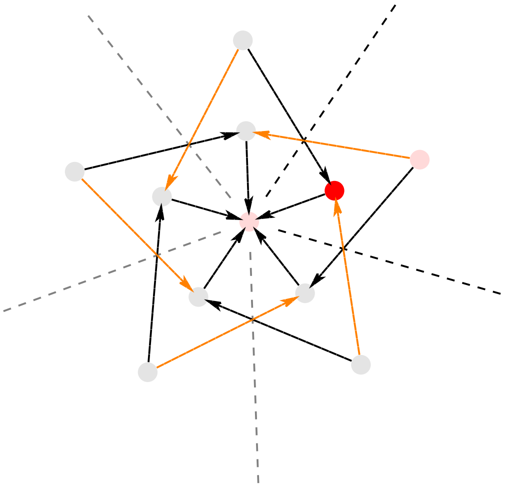


Symmetry Reduction
Example: 11 states, 3 equivalence classes



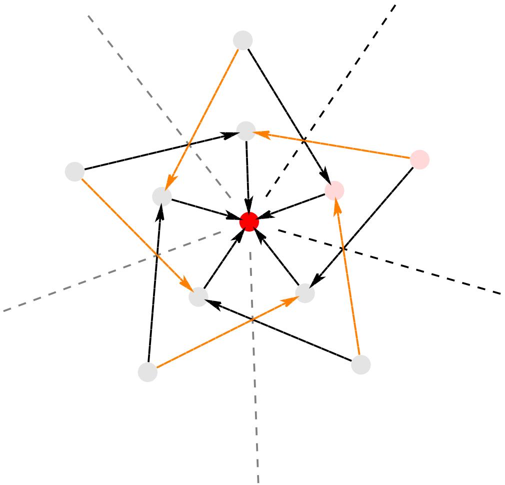
Symmetry Reduction

Example: 11 states, 3 equivalence classes



Symmetry Reduction

Example: 11 states, 3 equivalence classes



Idea [God91, Val91]

Independent actions unnecessary to consider in all orderings, e.g. 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 $m > n$ unnecessary.[a]

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

Heuristics for Classical Planning

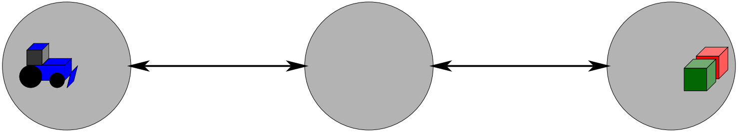
The most basic heuristics used for non-optimal *domain-independent* planning:

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}

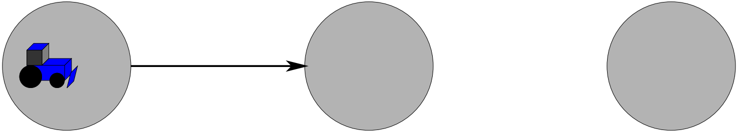
- Basic insight: estimate distances between possible state variable values, not states themselves.
- $g_s(l) = \begin{cases} 0 & \text{if } s \models l \\ \min_{a \text{ with effect } p} (1 + g_s(\text{prec}(a))) & \end{cases}$
- h^+ defines $g_s(L) = \sum_{l \in L} g_s(l)$ for sets S.
- h^{max} defines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

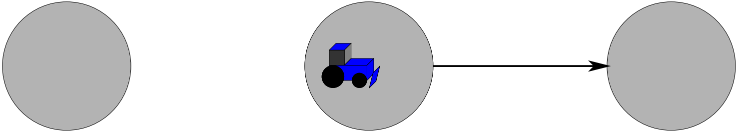


t	T1	T2	T3	A1	A2	A3	B1	B2	B3
0	T	F	F	F	F	T	F	F	T
1	TF	TF	F	F	F	T	F	F	T
2	TF	TF	TF	F	F	T	F	F	T
3	TF	TF	TF	F	TF	TF	F	TF	TF
4	TF	TF	TF	TF	TF	TF	TF	TF	TF

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

Computation of h^{max}

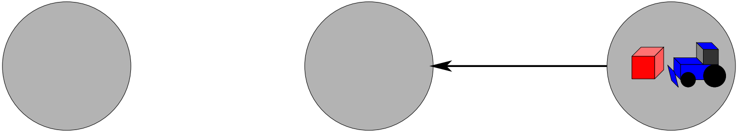
Tractor example



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

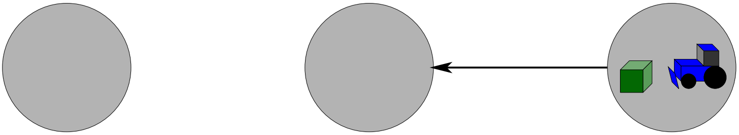
Computation of h^{max}
Tractor example



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

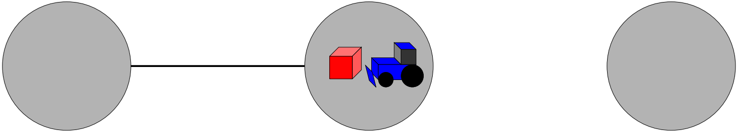
Computation of h^{max}
Tractor example



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

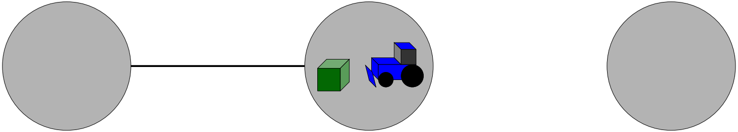


t	T1	T2	T3	A1	A2	A3	B1	B2	B3
0	T	F	F	F	F	T	F	F	T
1	TF	TF	F	F	F	T	F	F	T
2	TF	TF	TF	F	F	T	F	F	T
3	TF	TF	TF	F	TF	TF	F	TF	TF
4	TF	TF	TF	TF	TF	TF	TF	TF	TF

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

Computation of h^{max}

Tractor example

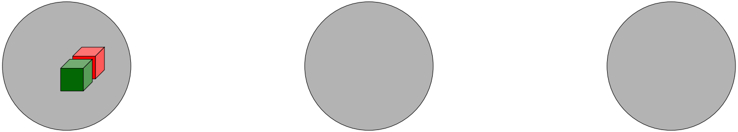


t	T1	T2	T3	A1	A2	A3	B1	B2	B3
0	T	F	F	F	F	T	F	F	T
1	TF	TF	F	F	F	T	F	F	T
2	TF	TF	TF	F	F	T	F	F	T
3	TF	TF	TF	F	TF	TF	F	TF	TF
4	TF	TF	TF	TF	TF	TF	TF	TF	TF

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 $\wedge$ lamp2on $\wedge$ 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 $5 + 5 = 10$.

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).
- The sum-heuristic and its various extensions, including relaxed plan heuristics [HN01, KHH12, KHD13] are used in practice for non-optimal planners.

Heuristic State-space Planners

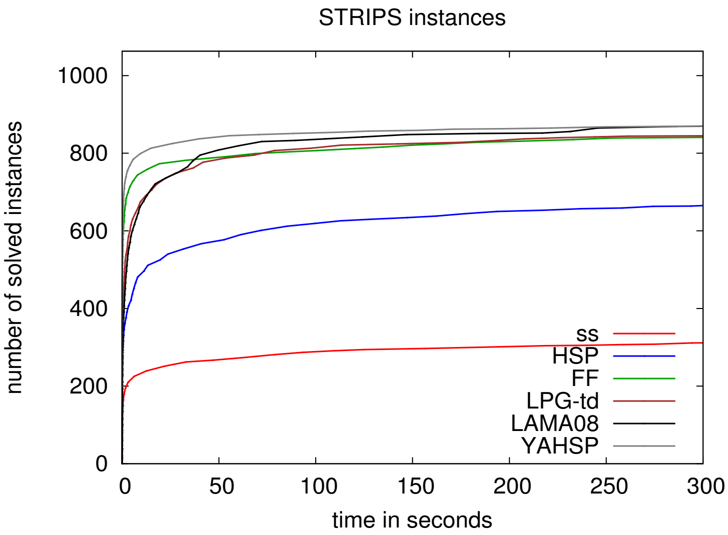
Some planners representing the current state of the art

$$
\begin{tikzcd}
 & \text{HSP [BLG97, BG01]} \arrow[dl] \arrow[d] \arrow[dr] & \\
\text{YAHSP3 [Vid04, Vid11]} & \text{LAMA [RW10]} & \text{PROBE [LG11]}
\end{tikzcd}
$$

- LAMA adds a preference for actions suggested by the computation of heuristic as good “first actions” towards goals [Vid04, RH09].
- YAHSP2/YAHSP3 and PROBE do – from each encountered state with a best-first search with h^+ – incomplete local searches to find shortcuts towards the goals.

Performance of State-Space Search Planners

Planning Competition Problems 2008-2011



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 (compose and abstract) [DFP09]

A generalization of pattern databases, allowing more complex aggregation of states (not just identification of ones agreeing on a subset of state variables.)
Planning people call it “merge and shrink”.

Landmark-cut [HD09] has worked well with standard benchmarks.

- 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 $\Rightarrow$ reductions to many other problems possible, often simple.

Other NP-complete search frameworks

constraint satisfaction (CSP)	[vBC99, DK01]
NM logic programs / answer-set programs	[DNK97]
Mixed Integer Linear Programming (MILP)	[DG02]

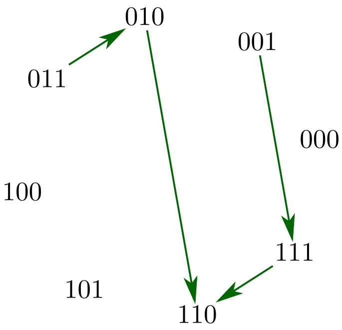
Transition relations in propositional logic

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

	000	001	010	011	100	101	110	111
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



Encoding of Actions as Formulas
for Sequential Plans

Actions as propositional formulas

New value of state variable x_i is a function of the old values of $x_1, \ldots, x_n$:
action j = 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 solvers

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

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

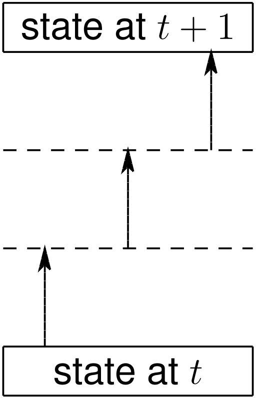
is satisfiable.

Remark

Most SAT solvers require formulas to be in CNF. There are efficient transformations to achieve this [Tse68, JS05, MV07].

Parallel Plans: Motivation

- Don't represent all intermediate states of a sequential plan.
- Don't represent the relative ordering of some consecutive actions.
- Reduced number of explicitly represented states $\Rightarrow$ smaller formulas



Parallel plans ($\forall$-step plans)

Blum and Furst [BF97], Kautz and Selman 1996 [KS96]

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

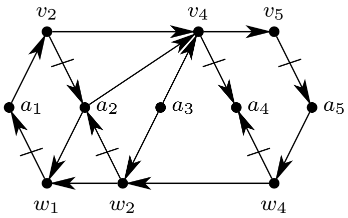
$$a@t \rightarrow l@t \text{ for all } l \text{ in the precondition of } a,$$

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.

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.

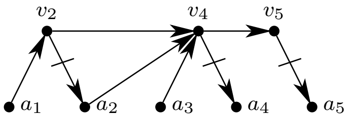
- $\bigcup_{i=1}^{n} p_i$ is consistent.
- $\bigcup_{i=1}^{n} e_i$ is consistent.
- There is a total ordering $a_1, \ldots, a_n$ such that $e_i \cup p_j$ is consistent whenever $i \leq j$: disabling an action earlier in the ordering is allowed.

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

$\exists$-step plans: linear encoding
Rintanen et al. 2006 [RHN06]

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

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
$\forall$-parallel	[BF97, KS96]	parallel actions independent
$\exists$-parallel	[DNK97, RHN06]	executable in at least one order

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

- $\forall$-parallel plans: the disables relation is empty.
- $\exists$-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.

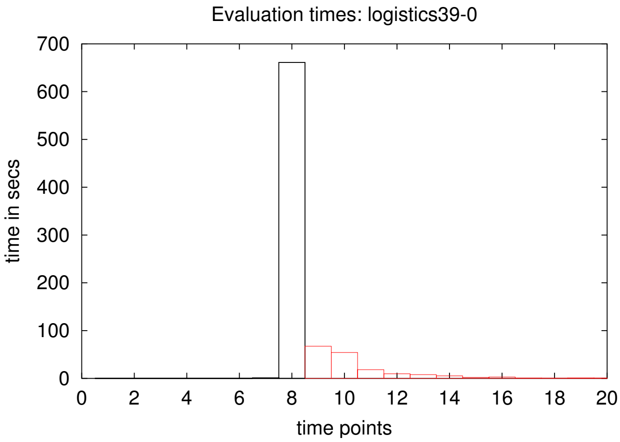
Q: How to schedule these satisfiability tests?

Search through Horizon Lengths

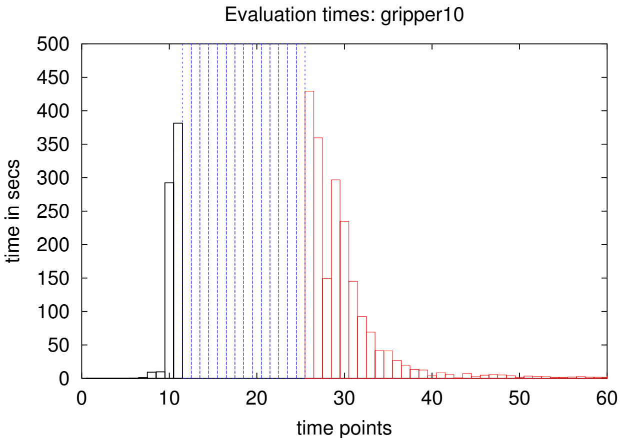
algorithm	reference	comment
sequential	[KS92, KS96]	slow, guarantees min. horizon
binary search	[SS07]	prerequisite: “tight” length UB
n processes	[Rin04b, Zar04]	fast, more memory needed
geometric	[Rin04b]	fast, more memory needed

- 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

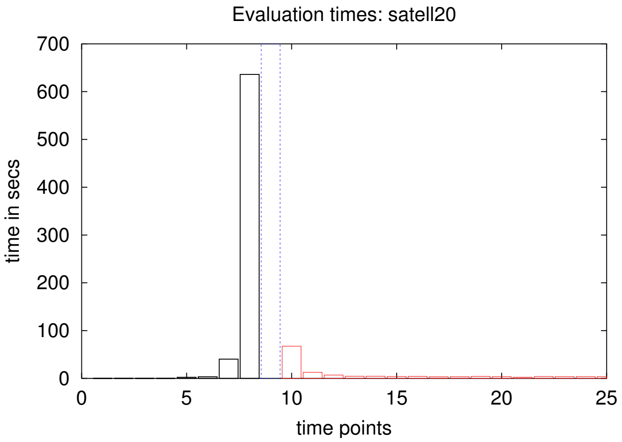
Some runtime profiles



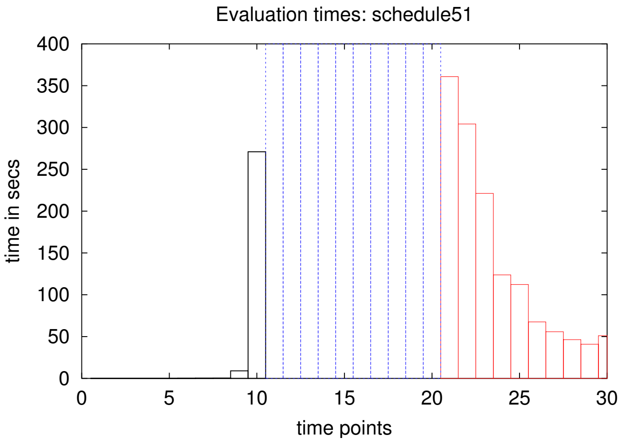
Some runtime profiles



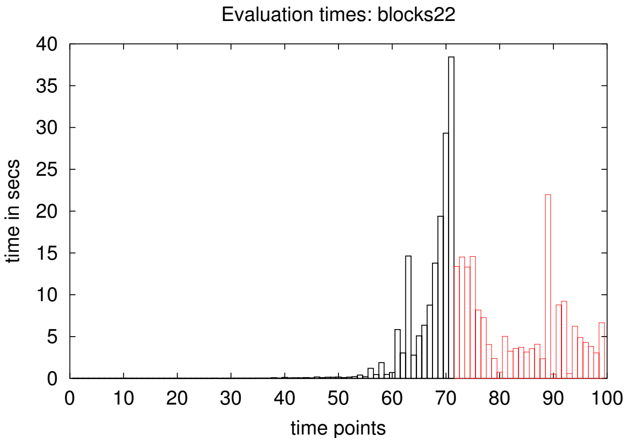
Some runtime profiles



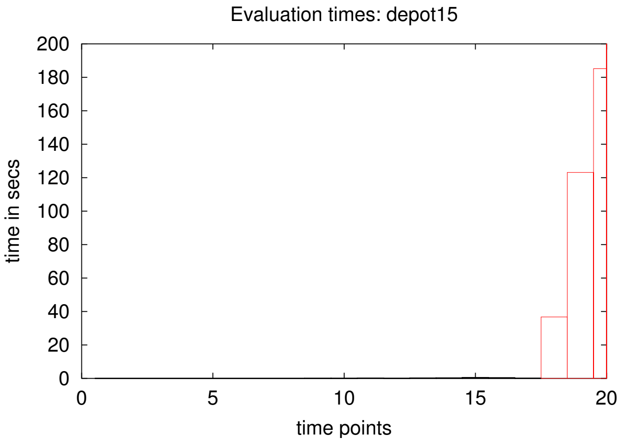
Some runtime profiles



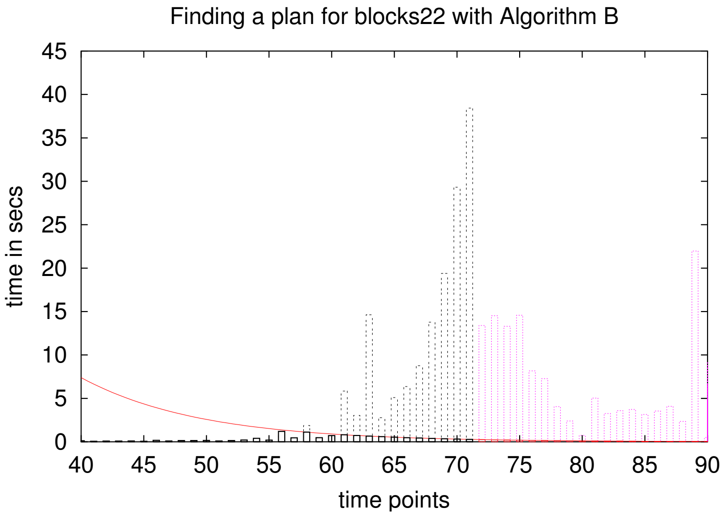
Some runtime profiles



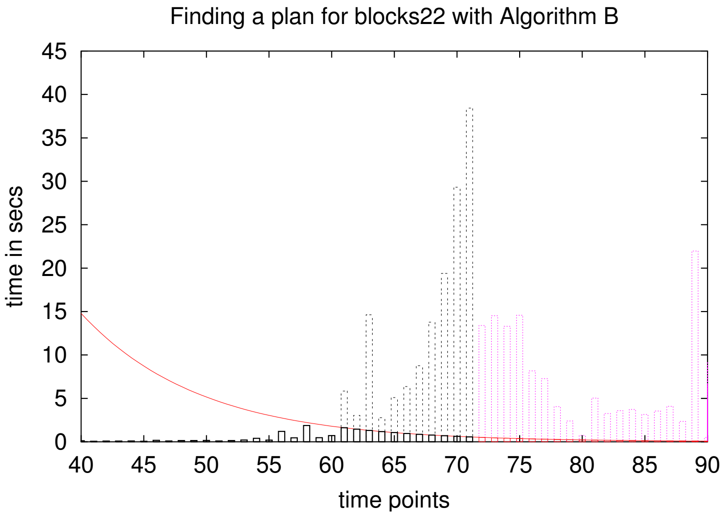
Some runtime profiles



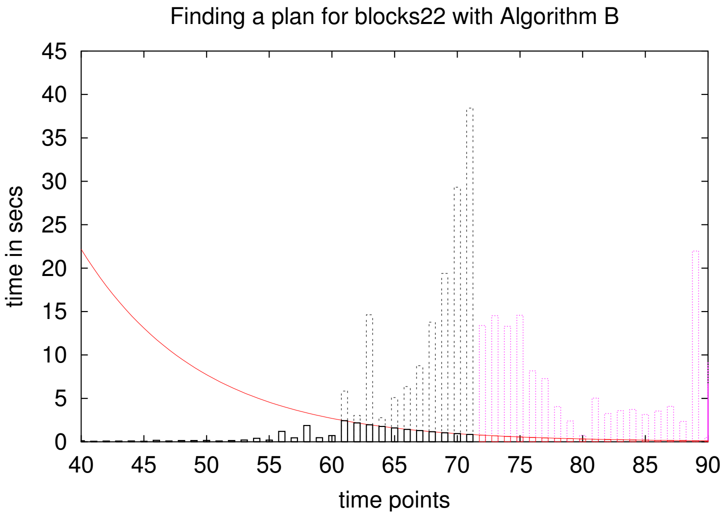
Geometric Evaluation



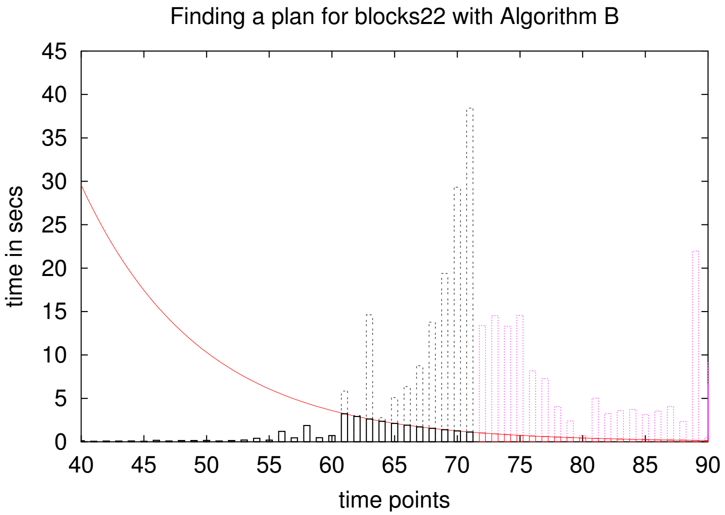
Geometric Evaluation



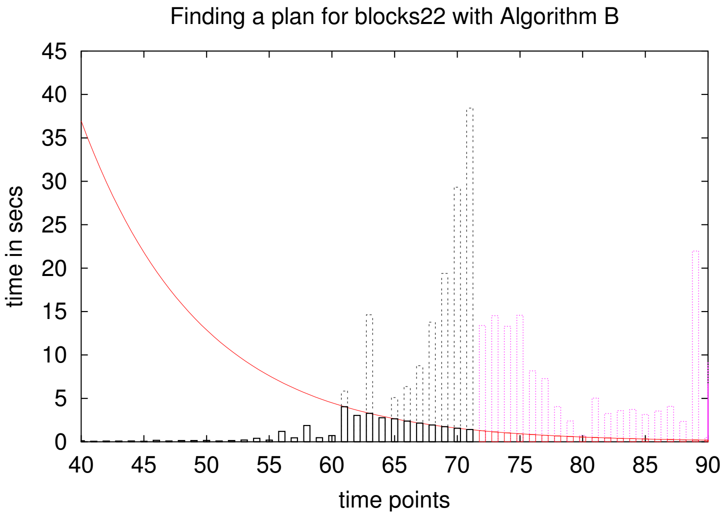
Geometric Evaluation



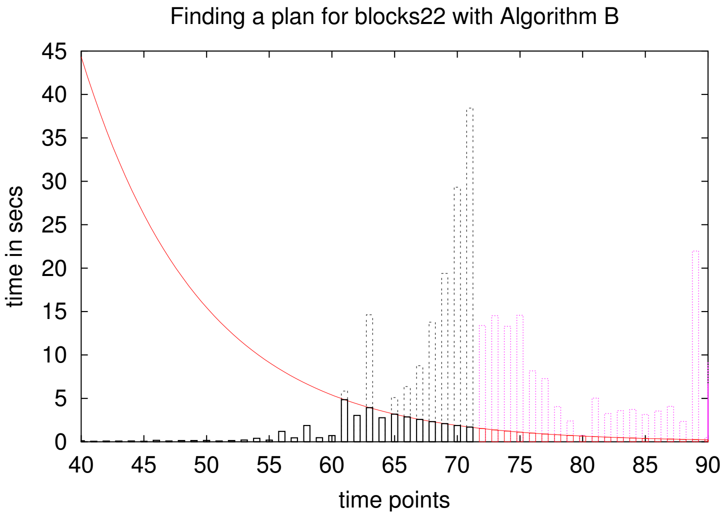
Geometric Evaluation



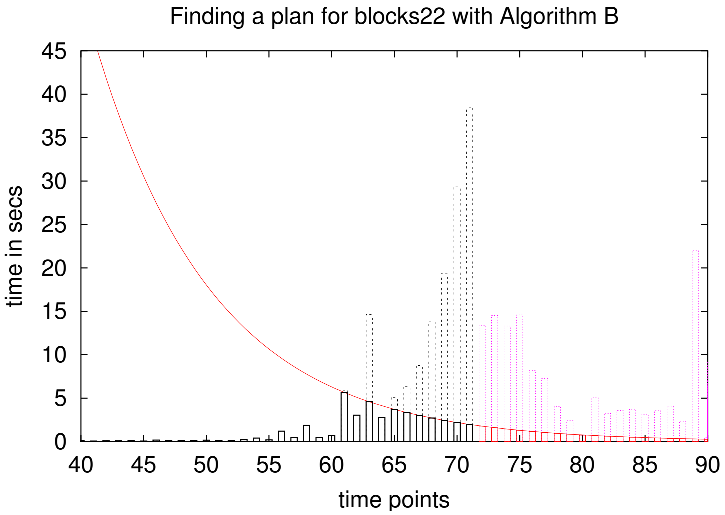
Geometric Evaluation



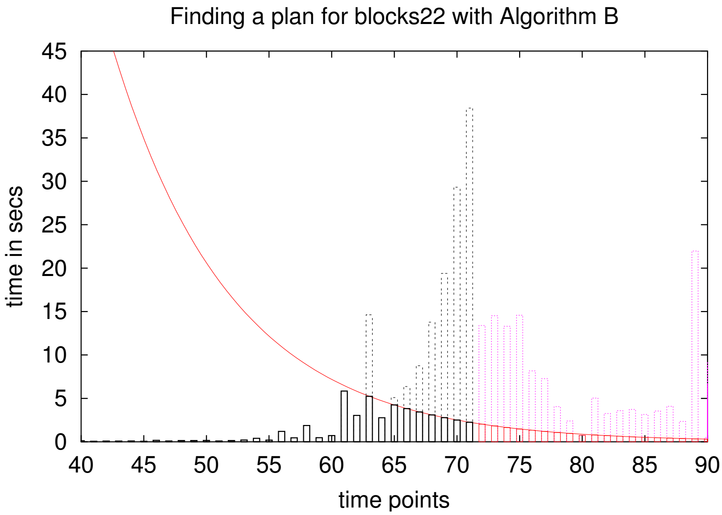
Geometric Evaluation



Geometric Evaluation



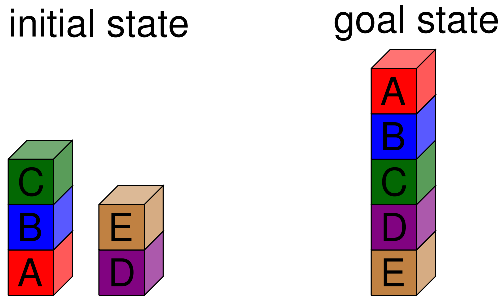
Geometric Evaluation



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].
  - Very 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 [Rin10, Rin12]
  - Replace standard CDCL heuristics with planning-specific ones.
  - For certain problem classes substantial improvement
  - New research topic: lots of unexploited potential



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
clear(a)	F	F				
clear(b)	F					F
clear(c)	T	T			F	F
clear(d)	F	T	T	F	F	F
clear(e)	T	T	F	F	F	F
on(a,b)	F	F	F			T
on(a,c)	F	F	F	F	F	F
on(a,d)	F	F	F	F	F	F
on(a,e)	F	F	F	F	F	F
on(b,a)	T	T			F	F
on(b,c)	F	F			T	T
on(b,d)	F	F	F	F	F	F
on(b,e)	F	F	F	F	F	F
on(c,a)	F	F	F	F	F	F
on(c,b)	T			F	F	F
on(c,d)	F	F	F	T	T	T
on(c,e)	F	F	F	F	F	F
on(d,a)	F	F	F	F	F	F
on(d,b)	F	F	F	F	F	F
on(d,c)	F	F	F	F	F	F
on(d,e)	F	F	T	T	T	T
on(e,a)	F	F	F	F	F	F
on(e,b)	F	F	F	F	F	F
on(e,c)	F	F	F	F	F	F
on(e,d)	T	F	F	F	F	F
ontable(a)	T	T	T			F
ontable(b)	F	F			F	F
ontable(c)	F			F	F	F
ontable(d)	T	T	F	F	F	F
ontable(e)	F	T	T	T	T	T

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

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

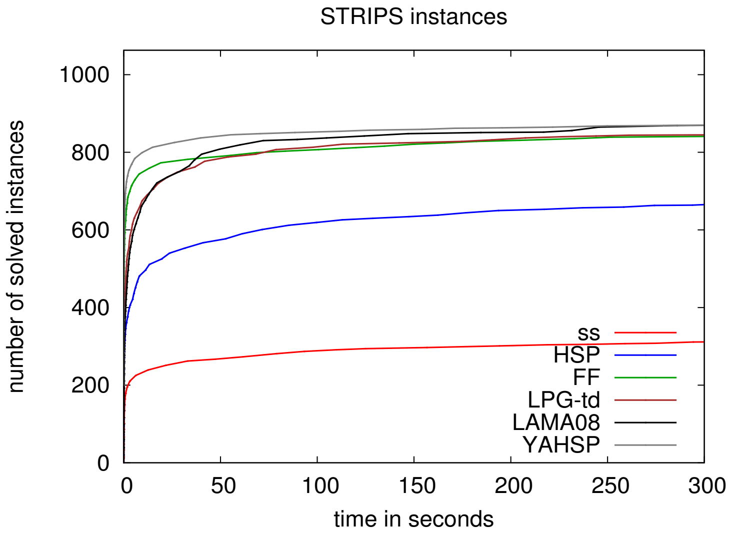
Solving the SAT Problem
Example

	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: $\neg\text{clear(b)}^1$.
3. Branch: clear(a)^3.
4. Plan found:

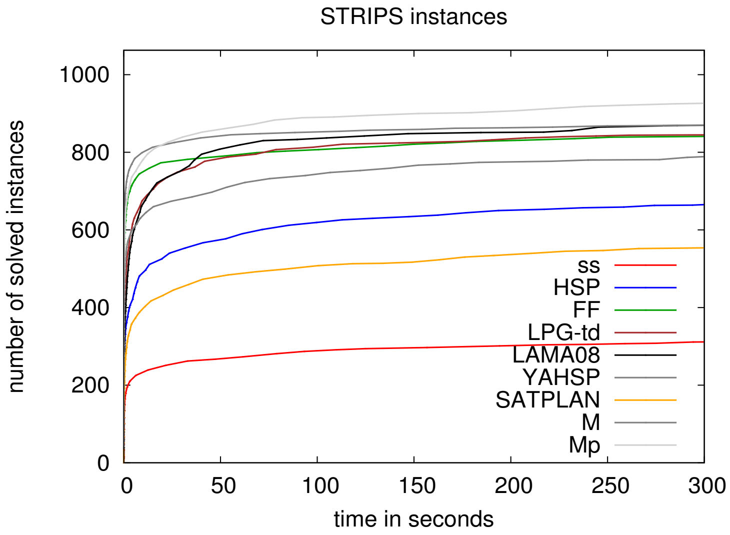
	0	1	2	3	4
fromtable(a,b)	F	F	F	F	T
fromtable(b,c)	F	F	F	T	F
fromtable(c,d)	F	F	T	F	F
fromtable(d,e)	F	T	F	F	F
totable(b,a)	F	F	T	F	F
totable(c,b)	F	T	F	F	F
totable(e,d)	T	F	F	F	F

Performance of SAT-Based Planners
Planning Competition Problems 1998-2008



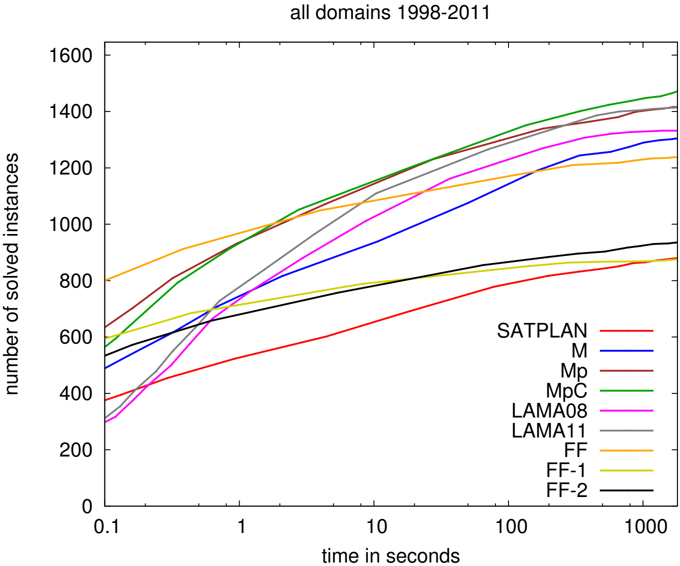
Performance of SAT-Based Planners

Planning Competition Problems 1998-2008



Performance of SAT-Based Planners

Planning Competition Problems 1998-2011 (revised)



Symbolic Search Methods
Motivation

- logical formulas as data structure for sets, relations
- state-space search (planning, model-checking, diagnosis, ...) in terms of set & relational operations
- Algorithms that can handle very large state sets, bypassing inherent limitations of enumerative methods.

- SAT and explicit state-space search: primary use finding one path from an initial state to a goal state
- "Symbolic" search methods can be used for more general problems:
  - Finding set of all reachable states
  - Distances/plans from the initial state to all states
  - Distances/plans to goal states from all states
- Competitive for optimal planning and detecting unsolvability.
- BDDs are a representation of belief states [BCRT01, Rin05].
- Algebraic Decision Diagrams (ADD) [FMY97, BFG+97] can represent value functions in probabilistic planning [HSAHB99].

Transition relations in propositional logic

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

$$
\begin{aligned}
&(\neg a \wedge b \wedge c \wedge \neg a' \wedge b' \wedge \neg c')\vee \\
&(\neg a \wedge b \wedge \neg c \wedge a' \wedge b' \wedge \neg c')\vee \\
&(\neg a \wedge \neg b \wedge c \wedge a' \wedge b' \wedge c')\vee \\
&(a \wedge b \wedge c \wedge a' \wedge b' \wedge \neg c')
\end{aligned}
$$

The corresponding matrix is

	000	001	010	011	100	101	110	111
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

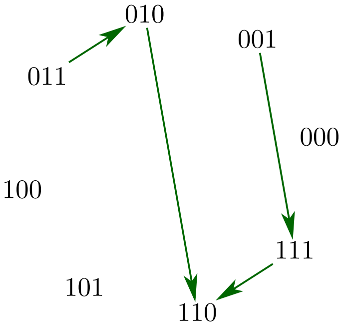


Image operations

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

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

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

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

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

(Pre-image corresponds to regression used with backward-search [Rin08].)

Finding All Plans with a Symbolic Algorithm
[BCL+94]

All reachable states with breadth-first search

$$S_0 = \{I\}$$
$$S_{i+1} = S_i \cup \bigcup_{a \in A} \mathit{img}_a(S_i)$$

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 (linear in plan length and number of state variables)
(Approximations of the above algorithm compute invariants [Rin08]).

Symbolic State-Space Search Algorithms

- Symbolic Breadth-First [BCL$^+$94]
- Symbolic (BDD) versions of A*:
  - BDDA* [ER98]
  - SetA* [JVB08]
  - ADDA* [HZF02]
- The Saturation algorithm [CLS01, CLM07, YCL09] trades optimality (as obtained with breadth-first) to far better scalability: find all reachable states, without accurate distance information.

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 $\emptyset$	$\bot$ (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 $\overset{a\,b\,a'b'}{1\,0\,1\,0}$ 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

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

$$\exists c.(a \vee (b \wedge c)) \equiv a \vee b \qquad \forall c.(a \vee (b \wedge c)) \equiv a$$

a	b	c	$a \vee (b \wedge c)$
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	$\exists c.(a \vee (b \wedge c))$
0	0	0
0	1	1
1	0	1
1	1	1

a	b	$\forall c.(a \vee (b \wedge c))$
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 $x@(t+1)$.

Images as Relational Operations

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

Images as Relational Operations

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

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

Images as Relational Operations

$$
\boxed{\begin{array}{l} s_0 00 \\ s_2 10 \end{array}}
\bowtie
\boxed{\begin{array}{l} s_0 s_1 00\ 01 \\ s_0 s_2 00\ 10 \\ s_1 s_0 01\ 00 \\ s_1 s_2 01\ 10 \\ s_2 s_0 10\ 00 \end{array}}
=
\boxed{\begin{array}{l} s_0 s_1 00\ 01 \\ s_0 s_2 00\ 10 \\ s_2 s_0 10\ 00 \end{array}}
\quad
\begin{array}{l|l} x_0 x_1 & \\ \hline 00 & 1 \\ 01 & 0 \\ 10 & 1 \\ 11 & 0 \end{array}
\bowtie
\begin{array}{l|l} x_0 x_1 x_0' x_1' & \\ \hline 0000 & 0 \\ 0001 & 1 \\ 0010 & 1 \\ 0011 & 0 \\ 0100 & 1 \\ 0101 & 0 \\ 0110 & 1 \\ 0111 & 0 \\ 1000 & 1 \\ 1001 & 0 \\ 1010 & 0 \\ 1011 & 0 \\ 1100 & 0 \\ 1101 & 0 \\ 1110 & 0 \\ 1111 & 0 \end{array}
=
\begin{array}{l|l} x_0 x_1 x_0' x_1' & \\ \hline 0001 & 1 \\ 0010 & 1 \\ 1000 & 1 \end{array}
$$

Computation of Successor States

Let

- $X = \{x_1, \ldots, x_n\}$,
- $X' = \{x'_1, \ldots, x'_n\}$,
- 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

- $X = \{x_1, \ldots, x_n\}$,
- $X' = \{x'_1, \ldots, x'_n\}$,
- 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.

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
d-DNNF deterministic DNNF	[Dar02]	

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

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

Remark

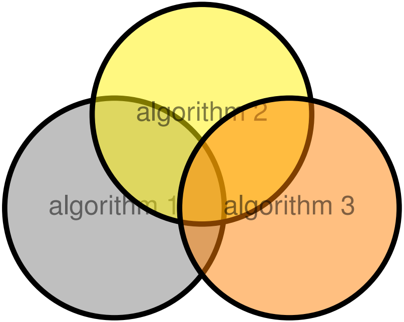
For BDDs one $\vee / \wedge$ is polynomial time/size (size is doubled) but repeated $\vee / \wedge$ lead to exponential size.

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

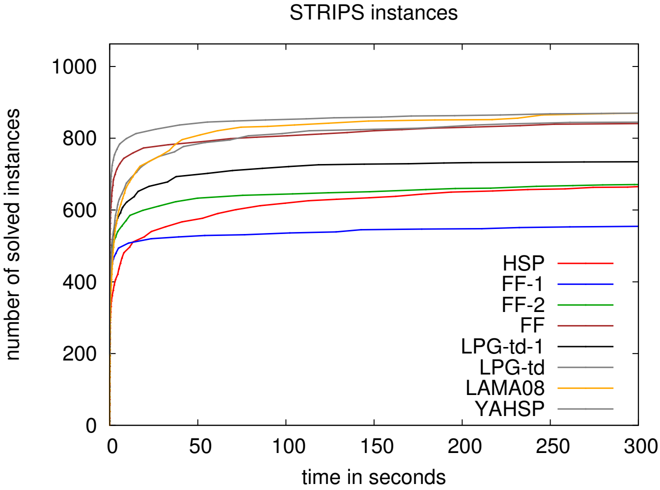
Methods for composing a portfolio

selection	choose one for current instance [XHHLB08]
parallel	run components in parallel [GS97, HLH97]
sequential	run consecutively, according to a schedule

Other variations of the above [HDH+00].

Early uses in planning: BLACKBOX [KS99] (manual configuration), FF [HN01] and LPG [GS02] (fixed configuration)

Many works on automated portofolio construction in the SAT area, directly applicable to planning as they are not specific to SAT or planning.



FF	=	FF-1 followed by FF-2 ($\sim$ HSP)
LPG-td	=	LPGT-td-1 followed by FF-2 ($\sim$ HSP)

Evaluation of Planners

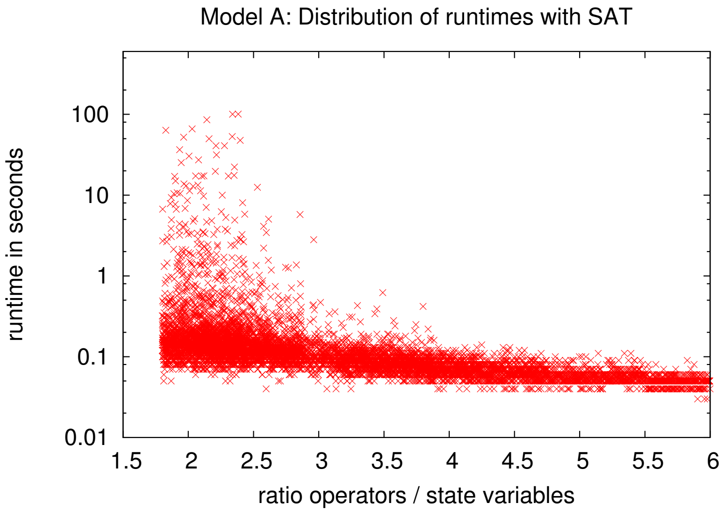
Evaluation of planning systems is based on

- 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, Rin04c].
  - \+ Easy to control problem hardness.
  - \- No direct real-world relevance (but: core of any “hard” problem)

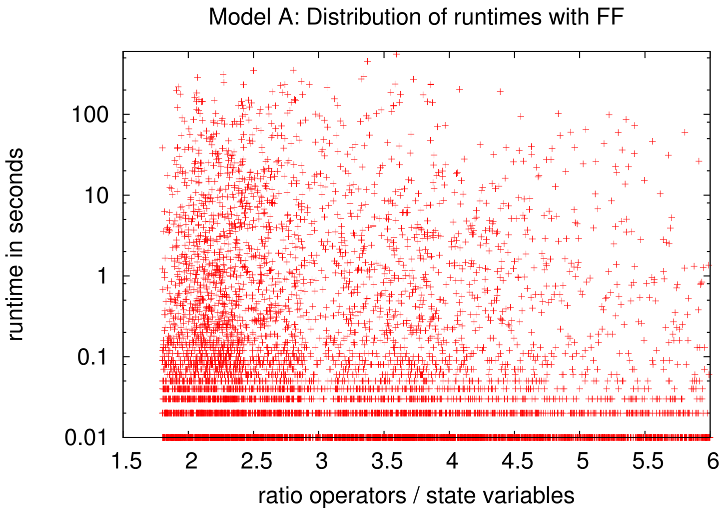
Sampling from the Set of All Instances
[Byl96, Rin04c]

- 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 (2^{20} states) too difficult for many planners.

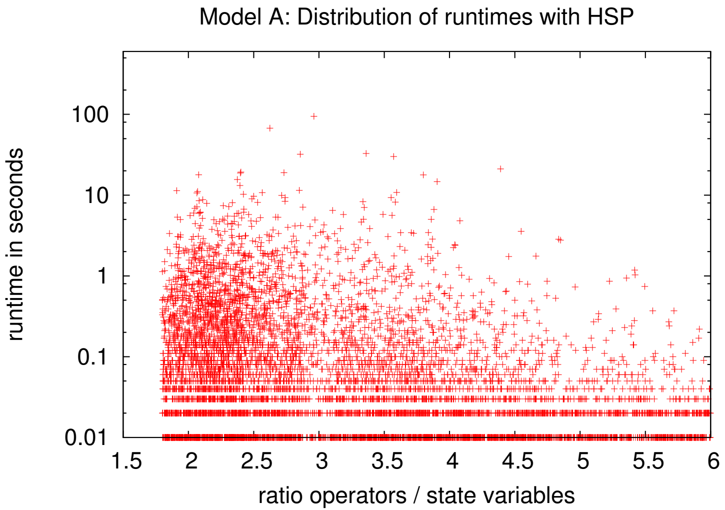
Sampling from the Set of All Instances
Experiments with planners



Sampling from the Set of All Instances
Experiments with planners



Sampling from the Set of All Instances
Experiments with planners



Minimization of the duration of the execution phase:

- Two short actions may be better than one long one.
- Actions can be taken in parallel.
- Connection to scheduling problems [SFJ00].

This is a core consideration in most mixed planning+scheduling problems. (Duration and especially concurrency ignored in classical planning and basic state-space search methods.)

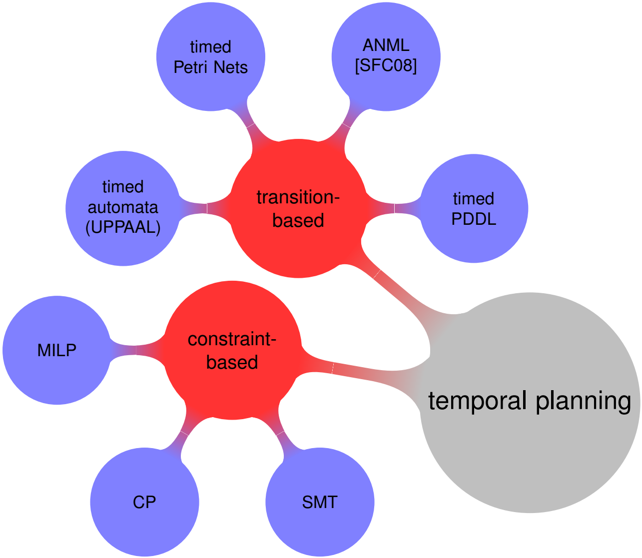
Inherent concurrency of actions

- Taking an action may require other concurrent actions.
- Some effects may only be achieved as joint effects of multiple actions.

Less important in practice: can often (always?) be avoided by modelling problem differently.

- Actions that must be used concurrently can be combined.
- Replace one complex action by several simpler ones: go to Paris = go to airport, board plane, fly, exit, take train to city

How to Represent Temporal Planning Problems?



Basic Modelling Concepts

Actions	Taken at a given time point t
Precondition	Must be satisfied at t.
Effects	Assignments $x := v$ at time points $t' > t$.
Dependencies	If action 1 taken at t, action 2 cannot be at $[t_1, t_2]$.

Action Dependencies through Resources

- n-ary resources
  Simultaneous use of resource can be at most n units.
  If each action needs 1 unit of the resource, no more than n actions can be using it simultaneously.
  Example: n identical tools or machines
- state resources
  A resource is in at most one state at a time.
  Multiple actions can use the resource in the same state.
  Example: generator that can produce 110V,60Hz or 220V,50Hz

Relation to scheduling

- Planning = action selection + scheduling.
- Scheduling = assignment of starting times to tasks/actions, respecting resource constraints
- Expressive languages for temporal planning include scheduling and hence support the representation of resources.
- Resources and ordering constraints are the mechanism for guaranteeing that plans are executable.

Complexity

Most important scheduling problems are NP-complete [GJ79].
Temporal planning complete for PSPACE or EXPSPACE [Rin07].
Action selection is the main difference between them.

Embedding of Scheduling in Temporal Planning

Representation of a simple job-shop scheduling problem in temporal planning.

1. For each job j = a sequence of tasks $t_1^j, \ldots, t_{n_j}^j$, introduce state variable $p_j : \{1, \ldots, n+1\}$.
2. Each task is mapped to action a_i^j with
   - precondition $p_j = i$,
   - effect $p_j = i + 1$ after the duration of t_i^j,
   - resource requirements as in the scheduling problem.
3. In the initial state $p_j = 1$ for every job j.
4. In the goal we have $p_j = n_{j+1}$.

Tasks and their ordering inside the job are fixed. Remaining problem is scheduling the tasks/actions for different jobs relative to other jobs' tasks/actions and minimizing the makespan. Solutions of the temporal planning problem are exactly the solutions to the job-shop scheduling problem.

Timed State-Space

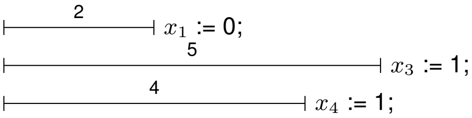
- state = values of state variables + values of clocks
- Clocks induce a schedule of future events.
- Actions initialize clocks.
- Time progresses, affecting all clocks.
- Reaching a critical clock value triggers scheduled events:
  - effects taking place later than the action's "starting" time point
  - resources allocated and later freed

This is the model behind all search methods.

Seemingly simple route to temporal planning with explicit state-space search.

Updates to the timed state
Scheduling (taking) an action

$$x_1 = 1$$
$$x_2 = 1$$
$$x_3 = 0$$
$$x_4 = 0$$

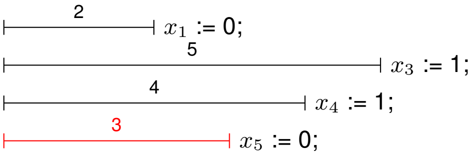


Updates to the timed state

Scheduling (taking) an action

Take action with precondition $x_2 = 1$ and effect $x_5 := 0$ at time 3.

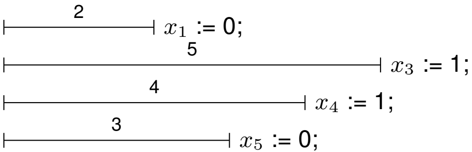
$$x_1 = 1$$
$$x_2 = 1$$
$$x_3 = 0$$
$$x_4 = 0$$
$$x_5 = 1$$



Updates to the timed state

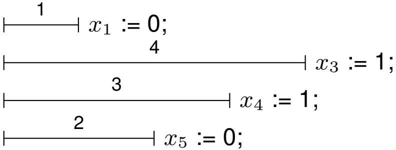
Advancing time

$$x_1 = 1$$
$$x_2 = 1$$
$$x_3 = 0$$
$$x_4 = 0$$
$$x_5 = 1$$



Updates to the timed state
Advancing time

$$x_1 = 1$$
$$x_2 = 1$$
$$x_3 = 0$$
$$x_4 = 0$$
$$x_5 = 1$$



Updates to the timed state

Advancing time

$$x_1 = 1$$
$$x_2 = 1$$
$$x_3 = 0$$
$$x_4 = 0$$
$$x_5 = 1$$

x_1 := 0;

3 x_3 := 1;

2 x_4 := 1;

1 x_5 := 0;

Updates to the timed state
Advancing time

$$
\begin{aligned}
x_1 &= 0 \\
x_2 &= 1 \\
x_3 &= 0 \\
x_4 &= 0 \\
x_5 &= 1
\end{aligned}
$$

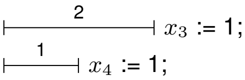
2
x_3 := 1;

1
x_4 := 1;

x_5 := 0;

Updates to the timed state
Advancing time

$$x_1 = 0$$
$$x_2 = 1$$
$$x_3 = 0$$
$$x_4 = 0$$
$$x_5 = 0$$



Completeness of Timed State-Space Search

- Since time is continuous, an action can be started at any of an infinite number of time points. $\Longrightarrow$ search space and branching factor infinite
- Simplistic policies for advancing time lead to incompleteness [MW06]. Most early temporal planners are incomplete. Few temporal planners have been proved to be complete.
- region abstraction [AD94] abstracts an infinite number of timed states to finitely many behaviorally equivalent regions.

Separation of planning and scheduling
CPT planner [VG06]

- Separate two problems
  1. selection of actions (only ordering, no timing)
  2. scheduling of these actions

  and interleave their solution.
- Action selection induces temporal constraints [DMP91]
- These temporal constraints can be solved separately.
- Completeness regained.

Systems for Temporal Planning

- Probably the most powerful verification tool based on explicit state-space search in the state-space induced by timed automata and their extension hybrid automata is UPPAAL [BLL+96].
  UPPAAL has been used in modelling and solving planning scenarios for example in robotics [QBZ04] and autonomous embedded systems [AAG+07, KMH01].
- CPT [VG06]
- Temporal Fast-Downward, based on the Fast-Downward planner for classical planning

Temporal Planning by Constraint Satisfaction

- Temporal planning can be encoded in
  - SAT modulo Theories (SMT) [WW99, ABC+02].
  - Constraint Programming [RvBW06]
  - Mixed Integer Linear Programming [DG02]

  (Similarly to scheduling [ABP+11].)
- The encoding methods for all are essentially the same. Differences in surface structure of the encoding, especially the types of constraints that can be encoded directly.
- In this tutorial we focus on SMT, due to its closeness to SAT.
- Differences in performance and pragmatic differences:
  - CP: support for customized search (heuristics, propagators, ...)
  - SMT: fully automatic, powerful handling of Boolean constraints.
  - MILP: for problems with intensive linear optimization

Each SMT instance fixes the number of steps i analogously to untimed (asynchoronous) state-space problems in SAT.

variables in SMT encoding

var	type	description
Δ_i	real	time between steps $i-1$ and i
$a@i$	bool	Is action a taken at step i?
$c_a@i$	real	Value of clock for action a at step i
$x@i$	bool	Value of Boolean state variable at step i

Executability of an action

Action cannot be taken if it is already active:

$$a@i \rightarrow (c_a@(i-1) \geq \textit{dur}(a)) \tag{1}$$

($\textit{dur}(a)$ denotes the duration a).

If actions actions a_1 and a_2 use the same unary resource respectively at $[t_1, t'_1]$ and at $[t_2, t'_2]$ then we have

$$t_2 + t'_2 - c_{a_1}@i \leq t_1 \tag{2}$$

$$t_1 + t'_1 \leq t_2 - c_{a_1} \tag{3}$$

Additionally, if $[t_1, t'_1]$ and $[t_2, t'_2]$ overlap, we have

$$\neg a_1@i \vee \neg a_2@i \tag{4}$$

Encodings of Timed Problems in SMT

Formula ϕ with every variable x replaced by $x@i$ is denoted by $\phi@i$.

Action with precondition p:

$$a@i \rightarrow p@i \tag{5}$$

If action is taken, its clock is initialized to 0:

$$a@i \rightarrow (c_a@i = 0) \tag{6}$$

If action is not taken, its clock advances:

$$\neg a@i \rightarrow (c_a@i = c_a@(i-1) + \Delta_i) \tag{7}$$

Encodings of Timed Problems in SMT

Effects of an action

An effect l scheduled at relative time t:

$$(c_a@i = t) \rightarrow l@i \tag{8}$$

Encodings of Timed Problems in SMT

Passage of time

Time may not pass a scheduled effect at relative time t:

$$c_a@(i-1) < t \rightarrow c_a@i \leq t \quad (9)$$

Time always passes by a non-zero amount:

$$\Delta_i > 0 \quad (10)$$

Let $(a_1, t_1), \ldots, (a_k, t_k)$ be all actions and times such that action a_i makes x true at time t relative to its start.

$$(\neg x@(i-1) \wedge x@i) \rightarrow ((c_{a_1}@i = t_1) \vee \cdots \vee (c_{a_k}@i = t_k)) \tag{11}$$

The frame axiom for x becoming false is analogous.

Encodings of Timed Problems in SMT

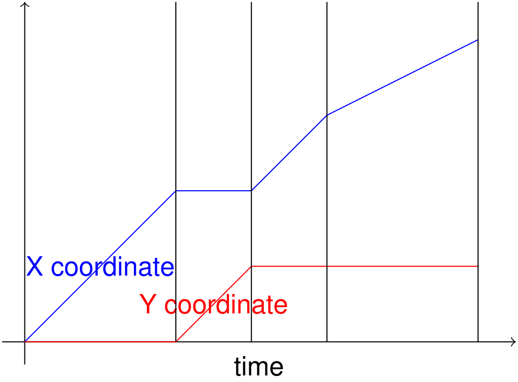
- Real variables in SMT incur a performance penalty.
- The encoding we gave is very general. In many practical cases (e.g. unit durations, small integer durations) more efficient encodings possible (SAT rather than SMT), similarly to scheduling problems.

Planning with Continuous Change

Hybrid systems = discrete change + continuous change

- Physical systems have continuous change.
  - movement of physical objects, substances, liquids (velocity, acceleration)
  - chemical and biological processes
  - light, electromagnetic radiation
  - electricity: voltage, charge, AC frequency, AC phase
- Discrete parts make the overall system piecewise continuous:
  - Discrete changes triggered by continuous change.
  - Continuous change controlled by discrete changes.
- Inherent issues with physical systems: lack of predictability, inaccuracy of control actions
- Problems primarily researched in control theory: Hybrid Systems Control, Model Predictive Control ("Planning" with continuous change not a separate research problem!)

Planning with Continuous Change
Example



actions: 2 east, 1 north, 1 east, $\frac{1}{2}$ east half speed

Hybrid Systems Modeling

- Continuous change a function of time.
- Type of change determined by discrete parts of the system.
- Example: heater on, heater off, temperature $f(w_0, \Delta)$
- Example: object in free fall, on ground, altitude $f(h_0, \Delta)$
- Both actions and continuous values trigger discrete change.
- Example: Falling object reaches ground.
- Example: Container becomes full of liquid.

Hybrid Systems with SMT

- Basic framework exactly as in the discrete timed case.
- Value of continuous variables directly a function of Δ.

law	explanation
$f(x, \Delta) = x + c\Delta$	linear change proportional to Δ
$f(x, \Delta) = x \cdot r^{c\Delta}$	exponential change
$f(x, \Delta) = c$	new constant value
$f(x, \Delta) = x$	no change, previous value

- Other forms of change require a clock variable and an initial value. For example polynomials $c + x^n$.

Hybrid systems: computational properties

- Simple decision problems about hybrid systems undecidable [HKPV95, CL00, PC07]: complete algorithms only for narrow problem classes.
- decidable cases for reachability: rectangular automata [HKPV95], 2-d PCD [AMP95], planar multi-polynomial systems [ČV96]
- semi-decision procedures: no termination when plans don't exist.
- stability: sensitivity to small inaccuracies in control [YMH98]

Hybrid systems: reasoning and analysis

- Main approaches generalize those for discrete timed systems.
    - explicit state-space search (e.g. HyTech [HHWT97])
    - SAT, constraints [SD05]
- Linear systems handled by efficient standard methods (MILP, linear arithmetics) in tools like MILP solvers and SAT modulo Theories solvers [SD05, ABCS05].
- Challenge: non-linear change
    - non-linear programming a very wide subarea of mathematical optimization. mixed integer nonlinear programming solvers (MINLP):
        - AIMMS
        - MAPLE
        - Mathematica
        - MATLAB
    - SMT solvers with non-linear arithmetic [JDM12, GKC13].

Model Predictive Control
Inaccuracy of control, uncertainty, unpredictability

Model Predictive Control [GPM89] (“Dynamical Matrix Control”, “Generalized Predictive Control”, “Receding Horizon Control”)

- Physical systems often not predictable enough for deterministic control.
- Continuous observation - prediction - control cycle.
- Predictions over a finite receding horizon
- Hybrid Model Predictive Control, integrating discrete variables.

Mixed Logical Dynamical (MLD) systems [BM99]

References I

Yasmina Abdeddaïm, Eugene Asarin, Matthieu Gallien, Félix Ingrand, Charles Lesire, Mihaela Sighireanu, et al.
Planning robust temporal plans: A comparison between CBTP and TGA approaches.
In *ICAPS 2007. Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling*, pages 2–10. AAAI Press, 2007.

Gilles Audemard, Piergiorgio Bertoli, Alessandro Cimatti, Artur Korniłowicz, and Roberto Sebastiani.
A SAT based approach for solving formulas over Boolean and linear mathematical propositions.
In Andrei Voronkov, editor, *Automated Deduction - CADE-18, 18th International Conference on Automated Deduction, Copenhagen, Denmark, July 27-30, 2002, Proceedings*, number 2392 in Lecture Notes in Computer Science, pages 195–210. Springer-Verlag, 2002.

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.

Carlos Ansótegui, Miquel Bofill, Miquel Palahı, Josep Suy, and Mateu Villaret.
Satisfiability modulo theories: An efficient approach for the resource-constrained project scheduling problem.
In *Proceedings of the 9th symposium on abstraction, reformulation and approximation (SARA 2011)*, pages 2–9, 2011.

References II

Rajeev Alur and David L. Dill.
A theory of timed automata.
Theoretical Computer Science, 126(2):183–235, 1994.

Eugene Asarin, Oded Maler, and Amir Pnueli.
Reachability analysis of dynamical systems having piecewise-constant derivatives.
Theoretical Computer Science, 138(1):35–65, 1995.

Jerry R. Burch, Edmund M. Clarke, David E. Long, Kenneth L. MacMillan, and David L. Dill.
Symbolic model checking for sequential circuit verification.
IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 13(4):401–424, 1994.

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 III

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.

Blai Bonet, Gábor Loerincs, and Héctor Geffner.
A robust and fast action selection mechanism for planning.
In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-97) and 9th Innovative Applications of Artificial Intelligence Conference (IAAI-97)*, pages 714–719. AAAI Press, 1997.

Johan Bengtsson, Kim Larsen, Fredrik Larsson, Paul Pettersson, and Wang Yi.
UPPAAL - a tool suite for automatic verification of real-time systems.
In *Hybrid Systems III*, volume 1066 of *Lecture Notes in Computer Science*, pages 232–243. Springer-Verlag, 1996.

Alberto Bemporad and Manfred Morari.
Control of systems integrating logic, dynamics, and constraints.
Automatica, 35(3):407–427, 1999.

References IV

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.

Franck Cassez and Kim Larsen.
The impressive power of stopwatches.
In Catuscia Palamidessi, editor, *CONCUR 2000 - Concurrency Theory*, volume 1877 of *Lecture Notes in Computer Science*, pages 138–152. Springer-Verlag, 2000.

Gianfranco Ciardo, Gerald Lüttgen, and Andrew S. Miner.
Exploiting interleaving semantics in symbolic state-space generation.
Formal Methods in System Design, 31(1):63–100, 2007.

References V

Gianfranco Ciardo, Gerald Lüttgen, and Radu Siminiceanu.
Saturation: An efficient iteration strategy for symbolic state-space generation.
In Tiziana Margaria and Wang Yi, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 2031 of *Lecture Notes in Computer Science*, pages 328–342. Springer-Verlag, 2001.

Stephen 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.

Kārlis Čerāns and Juris Vīksna.
Deciding reachability for planar multi-polynomial systems.
In Rajeev Alur, Thomas A. Henzinger, and Eduardo D. Sontag, editors, *Hybrid Systems III*, volume 1066 of *Lecture Notes in Computer Science*, pages 389–400. Springer-Verlag, 1996.

References VI

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

Adnan Darwiche.
A compiler for deterministic, decomposable negation normal form.
In *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI-2002) and the 14th Conference on Innovative Applications of Artificial Intelligence (IAAI-2002)*, pages 627–634, 2002.

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.

Yannis Dimopoulos and Alfonso Gerevini.
Temporal planning through mixed integer programming: A preliminary report.
In Pascal Van Hentenryck, editor, *Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming*, volume 2470 of *Lecture Notes in Computer Science*, pages 47–62. Springer-Verlag, 2002.

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 VII

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

Rina Dechter, Itay Meiri, and Judea Pearl.
Temporal constraint networks.
Artificial Intelligence, 49(1):61–95, 1991.

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 Amedeo Cesta, editor, *Recent Advances in AI Planning. Sixth European Conference on Planning (ECP'01)*, pages 13–24. AAAI Press, 2000.

References VIII

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.

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.

M. Ghallab, A. Howe, C. Knoblock, D. McDermott, A. Ram, M. Veloso, D. Weld, and D. Wilkins.
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.

M. R. Garey and D. S. Johnson.
Computers and Intractability.
W. H. Freeman and Company, San Francisco, 1979.

References IX

Sicun Gao, Soonho Kong, and Edmund M. Clarke.
dreal: An SMT solver for nonlinear theories over the reals.
In Maria Paola Bonacina, editor, *Automated Deduction - CADE-24*, volume 7898 of *Lecture Notes in Computer Science*, pages 208–214. Springer-Verlag, 2013.

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.

Carlos E. Garcìa, David M. Prett, and Manfred Morari.
Model predictive control: Theory and practice – a survey.
Automatica, 25(3):335–348, 1989.

Carla P. Gomes and Bart Selman.
Algorithm portfolio design: theory vs. practice.
In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence (UAI-97)*, pages 190–197. Morgan Kaufmann Publishers, 1997.

References X

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.

Jaco Geldenhuys and Antti Valmari.
A nearly memory-optimal data structure for sets and mappings.
In Thomas Ball and Sriram K. Rajamani, editors, *Model Checking Software*, volume 2648 of *Lecture Notes in Computer Science*, pages 136–150. Springer-Verlag, 2003.

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.

References XI

Adele E. Howe, Eric Dahlman, Christopher Hansen, Michael Scheetz, and Anneliese von Mayrhauser.
Exploiting competitive planner performance.
In Susanne Biundo and Maria Fox, editors, *Recent Advances in AI Planning. 5th European Conference on Planning, ECP'99, Durham, UK, September 8-10, 1999. Proceedings*, volume 1809 of *Lecture Notes in Computer Science*, pages 62–72, 2000.

Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi.
HYTECH: a model checker for hybrid systems.
International Journal on Software Tools for Technology Transfer (STTT), 1:110–122, 1997.

Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya.
What's decidable about hybrid automata?
In *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, pages 373–382, 1995.

Bernardo A. Huberman, Rajan M. Lukose, and Tad Hogg.
An economics approach to hard computational problems.
Science, 275(5296):51–54, 1997.

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

References XII

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.

Dejan Jovanović and Leonardo De Moura.
Solving non-linear arithmetic.
In Bernhard Gramlich, Dale Miller, and Uli Sattler, editors, *Automated Reasoning*, volume 7364 of *Lecture Notes in Computer Science*, pages 339–354. Springer-Verlag, 2012.

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.

References XIII

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.

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

Michael Katz, Jörg Hoffmann, and Carmel Domshlak.
Red-black relaxed plan heuristics.
In *Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI-13)*, pages 489–495. AAAI Press, 2013.

Emil Ragip Keyder, Jörg Hoffmann, and Patrik Haslum.
Semi-relaxed plan heuristics.
In *ICAPS 2012. Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling*, pages 128–136. AAAI Press, 2012.

Lina Khatib, Nicola Muscettola, and Klaus Havelund.
Mapping temporal planning constraints into timed automata.
In *Temporal Representation and Reasoning, 2001. TIME 2001. Proceedings. Eighth International Symposium on*, pages 21–27. IEEE, 2001.

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

References XIV

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.

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.

Nir Lipovetzky and Hector Geffner.
Searching for plans with carefully designed probes.
In *ICAPS 2011. Proceedings of the Twenty-First International Conference on Automated Planning and Scheduling*, pages 154–161, 2011.

References XV

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.

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.

References XVI

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.

Mausam and Daniel S. Weld.
Probabilistic temporal planning with uncertain durations.
In *Proceedings of the 21th National Conference on Artificial Intelligence (AAAI-2006)*, pages 880–887. AAAI Press, 2006.

André Platzer and Edmund M. Clarke.
The image computation problem in hybrid systems model checking.
In Alberto Bemporad, Antonio Bicchi, and Giorgio Buttazzo, editors, *Hybrid Systems: Computation and Control*, volume 4416 of *Lecture Notes in Computer Science*, pages 473–486. Springer-Verlag, 2007.

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.

References XVII

Michael Melholt Quottrup, Thomas Bak, and R. I. Zamanabadi.
Multi-robot planning: A timed automata approach.
In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04*, volume 5, pages 4417–4422. IEEE, 2004.

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 XVIII

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 XIX

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.
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.

Jussi Rintanen.
Planning as satisfiability: heuristics.
Artificial Intelligence, 193:45–86, 2012.

Francesca Rossi, Peter van Beek, and Toby Walsh.
Handbook of Constraint Programming.
Elsevier Science Publishers, 2006.

Silvia Richter and Matthias Westphal.
The LAMA planner: guiding cost-based anytime planning with landmarks.
Journal of Artificial Intelligence Research, 39:127–177, 2010.

References XX

Ji-Ae Shin and Ernest Davis.
Processes and continuous change in a SAT-based planner.
Artificial Intelligence, 166(1):194–253, 2005.

David E. Smith, Jeremy Frank, and William Cushing.
The ANML language.
ICAPS-08 Workshop on Knowledge Engineering for Planning and Scheduling (KEPS), 2008.

David Smith, Jeremy Frank, and Ari Jonsson.
Bridging the gap between planning and scheduling.
Knowledge Engineering Review, 15(1):47–83, 2000.

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.

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.

References XXI

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 derivations in propositional calculus.
In A. O. Slisenko, editor, *Studies in Constructive Mathematics and Mathematical Logic, Part II*, pages 115–125. Consultants Bureau, 1968.

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.

Vincent Vidal and Héctor Geffner.
Branching and pruning: an optimal temporal POCL planner based on constraint programming.
Artificial Intelligence, 170:298–335, 2006.

References XXII

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.

Vincent Vidal.
YAHSP2: Keep it simple, stupid.
Notes for The 2011 International Planning Competition (unpublished), 2011.

Steven A. Wolfman and Daniel S. Weld.
The LPSAT engine & its application to resource planning.
In Thomas Dean, editor, *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 310–315. Morgan Kaufmann Publishers, 1999.

Lin Xu, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown.
SATzilla: portfolio-based algorithm selection for SAT.
Journal of Artificial Intelligence Research, 32:565–606, 2008.

Andy Jinqing Yu, Gianfranco Ciardo, and Gerald Lüttgen.
Decision-diagram-based techniques for bounded reachability checking of asynchronous systems.
International Journal on Software Tools for Technology Transfer, 11(2):117–131, 2009.

References XXIII

Hui Ye, Anthony N. Michel, and Ling Hou.
Stability theory for hybrid dynamical systems.
IEEE Transactions on Automatic Control, 43(4):461–474, 1998.

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.