Automated Action Planning

Classical Planning for Non-Classical Planning Formalisms

Carmel Domshlak

Automated Action Planning
— Classical Planning for Non-Classical Planning Formalisms

Beyond Classical Planning

Richer models people are working on

1. Temporal Planning (action have duration)
2. Metric Planning (continuous variables)
3. Planning with Preferences
4. Planning with Resource Constraints
5. Net-benefit Planning (maximize net value of goals achieved)
6. Generalized Planning (complex control structures, such as loops)
7. Multi-agent Planning
8. Planning Under Uncertainty:
   8.1 Conformant Planning
   8.2 Contingent Planning
   8.3 Markov Decision Processes (MDPs)
   8.4 Partially Observable MDPs
   8.5 Conformant Probabilistic Planning (Fully Unobservable POMDPs)

How many courses on planning do we need?

Key Insights:

- ☺ Classical planning offers a wealth of ideas for generating good solutions, fast.
- ☹ Importing these ideas to each of the above non-classical formalisms is difficult, and often simply does not work.

Yet:

- ☺ Goal oriented sequencing of actions is a fundamental computational problem at the heart of all planning problems.
- ☺ Classical planners have reached a certain performance level that makes them attractive for addressing this problem.

So...

Two Strategies

1. Top-down:
   Develop native solvers for more general class of models

   $+$: generality
   $-$: complexity

2. Bottom-up: Extend the scope of 'classical' solvers

   $+$: efficiency
   $-$: generality

We now explore the second approach

Using Classical Planners within Non-Classical Planners

Two Key Techniques:

1. Replanning: the classical problem is an optimistic view of the original problem
2. Compilation: the classical problem is equivalent to the original problem
   (possibly under certain reasonable conditions)

Replanning

An online method for solving planning problems with some uncertainty

1. Make some assumptions $\rightarrow$ get a simpler model
2. Solve simpler model
3. Execute until your observation contradict your assumptions
4. Repeat (Replan)

An established technique:

- Underlies many closed loop controllers
- Used in motion planning under uncertainty

Restrictions on observability

Let $\langle P, I, O, G; P^* \rangle$ be a problem instance in nondeterministic planning.

1. If $P = P^*$, the problem instance is fully observable.
2. If $P^* = \emptyset$, the problem instance is unobservable.
3. If there are no restrictions on P^* then the problem instance is partially observable.

FF-Replan – Yoon, Fern, Given (2007)

Stochastic Shortest Path (SSP)

Imagine a classical planning problem except:

- Actions have stochastic effects
- We get to observe the state following each action
- Special case of a Markov Decision Process (MDP)

FF-Replan

Replanning in SSP

1. Simplify: determinize the effect of actions to get a classical model
2. Solve
3. Execute plan until you observe an unexpected state $=$
   $=$ effect was not the one you assumed in your classical model
4. Replan from new state
5. Repeat until you reach the goal

FF-Replan

Performance

- Base-line planner for IPC 2004 probabilistic planning track
- Won the first place and got some people quite pissed off...
- Very fast thanks to its underlying classical planner (FF)

FF-Replan

Some flaws:

- Choices are not well informed
- Ignores risks: an effect we ignored may trap as in a dead-end
- Ignores numbers: no evaluation of expected path length
- Clearly sub-optimal

Improvements

By selecting more sophisticated sampling/resampling, these problems can be addressed or mitigated!

- Make sure effects of different instances of an action differ
- Hindsight optimization:
  Solve multiple determinization & aggregate results

Replanning

- Solving a simplified problem always carries some risk.
- Can we regain completeness? optimality?

Motivation: Why Analyzing the Expressive Power?

- Expressive power is the motivation for designing new planning languages

$\leadsto$ Often there is the question: *Syntactic sugar* or *essential feature*?

- *Compiling away* or change planning algorithm?
- If a feature can be compiled away, then it is apparently only *syntactic sugar*.
- However, a compilation can lead to much larger planning domain descriptions or to much longer plans.

$\leadsto$ This means the planning algorithm will probably choke, i.e., it cannot be considered as a compilation

Example: DNF Preconditions

- Assume we have **DNF preconditions** in STRIPS operators
- This can be **compiled away** as follows
- Split each operator with a DNF precondition $c_1 \vee \ldots \vee c_n$ into n operators with the same effects and c_i as preconditions

$\rightsquigarrow$ If there exists a plan for the original planning task there is one for the new planning task and *vice versa*

$\longrightarrow$ The planning task has almost the same size

$\longrightarrow$ The shortest plans have the same size

Example: Conditional effects

- Can we compile away **conditional effects** to STRIPS?
- Example operator: $\langle a, b \rhd d \wedge \neg c \rhd e \rangle$
- Can be translated into four operators:
  $\langle a \wedge b \wedge c, d \rangle, \langle a \wedge b \wedge \neg c, d \wedge e \rangle, \ldots$
- Plan existence and plan size are identical
- Exponential blowup of domain description!

$\longrightarrow$ Can this be avoided?

FDR Planning with Soft Goals

- Planning with soft goals aimed at plans π that maximize utility

$$u(\pi) = \sum_{p \in app_{\pi}(I)} u(p) \quad - \quad \sum_{a \in \pi} cost(a)$$

- Best plans achieve best tradeoff between action costs and rewards
  $\leadsto$ Note: "do nothing" is always a valid plan.
  $\rightarrow$ Suggests conceptual difference?
- Model used in recent planning competitions; net-benefit track 2008 IPC
- Yet soft goals do not add expressive power; they can be compiled away

FDR Planning with Soft Goals

- For each soft goal p, create new hard goal p' initially false, and two new actions:
  - $\mathit{collect}(p)$ with precondition p, effect p' and cost 0, and
  - $\mathit{forgo}(p)$ with an empty precondition, effect p' and cost $u(p)$
- Plans π maximize $u(\pi)$ iff minimize $\mathit{cost}(\pi) = \sum_{a \in \pi} \mathit{cost}(a)$ in resulting problem
- Any helpful in practice?
- Compilation yields better results that native soft goal planners in 2008 IPC [KG07]

	IPC-2008 Net-Benefit Track			Compiled Problems			
Domain	Gamer	HSP^*_{P}	Mips-XXL	Gamer	HSP^*_{F}	HSP^*_{0}	Mips-XXL
crewplanning(30)	4	16	8	-	8	21	8
elevators (30)	11	5	4	18	8	8	3
openstacks (30)	7	5	2	6	4	6	1
pegsol (30)	24	0	23	22	26	14	22
transport (30)	12	12	9	-	15	15	9
woodworking (30)	13	11	9	-	23	22	7
total	71	49	55		84	86	50

Planning without observability: conformant planning

- Here we consider the second special case of planning with partial observability: planning without observability.
- Plans are sequences of actions because observations are not possible, actions cannot depend on the nondeterministic events or uncertain initial state, and hence the same actions have to be taken no matter what happens.
- Techniques needed for planning without observability can often be generalized to the general partially observable case.

Why acting without observation?

- Conformant planning is like planning to act in an environment while you are blind and deaf.
- Observations could be expensive or it is preferable to have a simple plan.
- Example: Finding synchronization sequences in hardware circuits
- Example: Initializing a system consisting of many components that are in unknown states.
- Internal motivation: try to understand the unobservable case so that one can better deal with the more complicated partially observable case.

Belief states and the belief space

- The current state is not in general known during plan execution. Instead, a set of possible current states is known.
- The set of possible current states forms the belief state.
- The set of all belief states is the belief space.
- If there are n states and none of them can be observationally distinguished from another, then there are $2^n - 1$ belief states.

The belief space

1. Let B be a belief state (a set of states).
2. Operator o is executable in B if it is executable in every $s \in B$.
3. When o is executed, possible next states are $T = img_o(B)$.
4. Belief states can be succinctly represented using Boolean formulae or BDDs.

The belief space

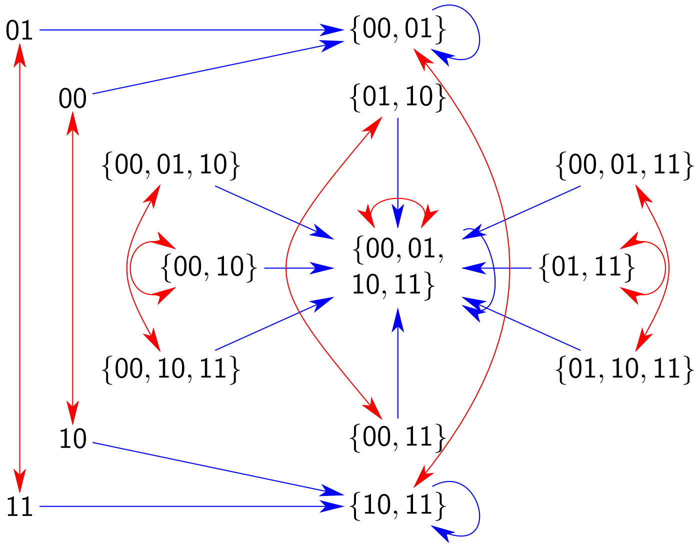
Example

Example (Next slide)

Belief space generated by states over two Boolean state variables.
$n = 2$ state variables, $2^n = 4$ states, $2^{2^n} - 1 = 15$ belief states
red action: complement the value of the first state variable
blue action: assign a random value to the second state variable

The belief space

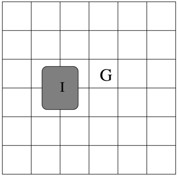
Example



Algorithms for unobservable problems

1. Find an operator sequence $o_1, \ldots, o_n$ that reaches a state satisfying G starting from any state satisfying I.
2. o_1 must be applicable in all states $B_0 = \{s \in S | s \models I\}$ satisfying I.
   o_2 must be applicable in all states in $B_1 = img_{o_1}(B_0)$.
   o_i must be applicable in all states in $B_i = img_{o_i}(B_{i-1})$ for all $i \in \{1, \ldots, n\}$.
   Terminal states must be goal states: $B_n \subseteq \{s \in S | s \models G\}$.

Conformant vs. Classical Planning



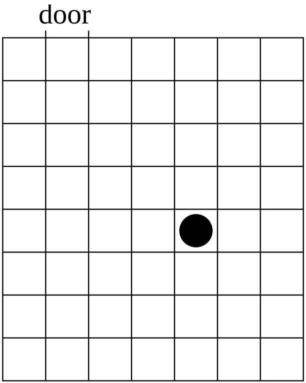
Problem: A robot must move from an **uncertain** I into G with **certainty**, one cell at a time, in a grid nxn

- Conformant and classical planning look similar except for uncertain I (assuming actions are deterministic).
- Yet plans can be quite different:
  best **conformant plan** must move robot to a corner first! (in order to localize)

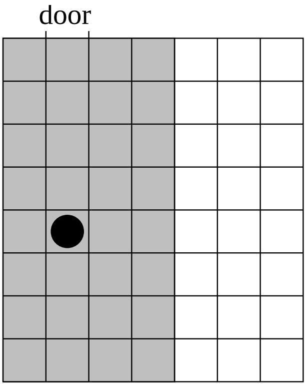
The belief space

Example

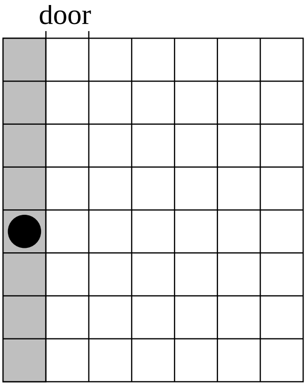
- A robot without any sensors, anywhere in a room of size 7×8.
- Actions: go North, South, East, West; if no way, just stay where you are
- Plan for getting out: $6 \times$ West, $7 \times$ North, $1 \times$ East, $1 \times$ North
- On the next slides we depict one possible location of the robot (•) and the change in the belief state at every execution step by gray fields.



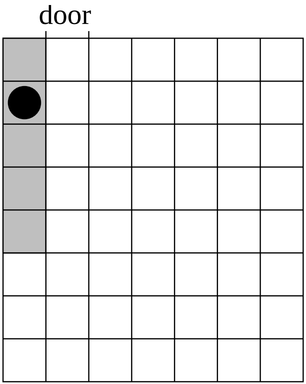
Example: after WWW



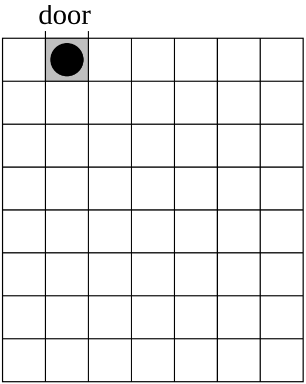
Example: after WWWWWW



Example: after WWWWWWWNNN



Example: after WWWWWWNNNNNNNE



Empirical Troubles with Conformant Planning

Problems with top-down approach

- effective representation of belief states b
- effective heuristic $h(b)$ for estimating cost from b to b_G

Now show: both tackled by translation into classical planning!

Complexity: Classical vs. Conformant Planning

- **Complexity:** conformant planning harder than classical planning
  - *because* **verification** *of a conformant plan* **intractable** *in worst case*
- **Idea:** focus on computation of conformant plans that are **easy to verify** (e.g., in linear time in the plan length)
  - *computation of such plans* **no more complex than classical planning**

Basic Translation: Move to Knowledge Level

Given conformant problem $\Pi = \langle P, I, O, G \rangle$

- P – set of (all unobservable) *propositional* state variables
- O – set of operators with conditional effects $\langle c, e \rangle$
- I – *prior knowledge* about the initial state (clauses over P)
- G – goal description (conjunction over A)

Define classical problem $K_0(\Pi) = \langle P', I', O', G' \rangle$

- $P' = \{Kp, K\neg p \mid p \in P\}$
- $I' = \{Kp \mid \text{ clause } L \in I\}$
- $G' = \{Kp \mid p \in G\}$
- $O' = O$ but preconds p replaced by Kp, and effects $\langle c, e \rangle$ replaced by $Kc \rightarrow Ke$ (supports) and $\neg K \neg c \rightarrow \neg K \neg e$ (cancellation)

$K_0(\Pi)$ is sound but incomplete: every classical plan that solves $K_0(\Pi)$ is a

Basic Translation: Move to Knowledge Level

Conformant Π	$\Rightarrow$	Classical $K_0(\Pi)$
$\langle P, I, O, G \rangle$	$\Rightarrow$	$\langle P', I', O', G' \rangle$
variable p	$\Rightarrow$	$Kp, K\neg p$ *(two vars)*
Init: known var p	$\Rightarrow$	$Kp \wedge \neg K\neg p$
Init unknown var p	$\Rightarrow$	$\neg Kp \wedge \neg K\neg p$ *(both false)*
Goal p	$\Rightarrow$	Kp
Operator a has prec p	$\Rightarrow$	a has prec Kp
Operator a: $\langle c, p \rangle$	$\Rightarrow$	$\begin{cases} a : Kc \rightarrow Kp \\ a : K\neg c \rightarrow \emptyset \\ a : \neg K\neg c \rightarrow \neg K\neg p \end{cases}$

Basic Properties and Extensions

- Translation $K_0(\Pi)$ is sound:
  - If π is a **classical plan** that solves $K_0(\Pi)$, then π is a **conformant plan** for Π.
- But way too incomplete
  - often $K_0(\Pi)$ will have no solution while Π does
  - works when uncertainty is irrelevant
- Extension $K_{T,M}(\Pi)$ we present now **can** be both complete and polynomial

Idea

- Given literal L and tag t, atom KL/t means
  - $K(t_0 \supset L)$: *KL true if t is true* **initially**

Example

- Conformant Problem Π:
  - Init: $x_1 \vee x_2, \neg g$
  - Goal: g
  - Actions: $a_1 : x_1 \rightarrow g$, $a_2 : x_2 \rightarrow g$
- Classical Problem $K_{T,M}(\Pi)$:
  - Init: $Kx_1/x_1, Kx_2/x_2, K\neg g, \neg Kg, \neg Kx_1, \neg K\neg x_1, \ldots$
  - After a_1: Kg/x_1, $Kx_1/x_1, Kx_2/x_2$, $\neg K\neg g$, $\neg Kg, \ldots$
  - After a_2: Kg/x_2, $Kg/x_1, Kx_1/x_2, Kx_2/x_2, \neg K\neg g, \neg Kg, \ldots$
    - New action $merge_g$: $Kg/x_1 \wedge Kg/x_2 \rightarrow Kg$
  - After $merge_g$: Kg, $Kg/x_2, Kg/x_1, Kx_1/x_2, Kx_2/x_2, \neg K\neg g, \ldots$
  - Goal satisfied: Kg

Key elements in Translation $K_{T,M}(\Pi)$

- a set T of tags t: consistent set of assumptions (literals) about the initial situation I

$$I \not\models \neg t$$

- a set M of merges m: valid subsets of tags

$$I \models \bigvee_{L \in m} L$$

- Semantics of var KL/t: L is true given that initially t (*i.e.* $K(t_0 \supset L)$)

Example of T, M

Example

Given $I = \{p \vee q, v \vee \neg w\}$, T and M can be:

$$\begin{aligned} T &= \{\{\}, p, q, v, \neg w\} \\ M &= \{\{p, q\}, \{v, \neg w\}\} \end{aligned} \qquad \begin{aligned} T' &= \{\{\}, \{p, v\}, \{q, v\}, \ldots\} \\ M' &= \ldots \end{aligned}$$

Translation $K_{T,M}(\Pi)$

For conformant $\langle P, I, O, G\rangle$, $K_{T,M}(\Pi)$ is $\langle P', I', O', G'\rangle$

- **P**$'$: KL/t for every literal L and tag $t \in T$
- **I**$'$: KL/t if $I \models (t \supset L)$
- **G**$'$: KL for $L \in G$
- For every tag t in T and $a : L_1 \wedge \cdots \wedge L_n \rightarrow L$ in O, add to O'
  - $a : KL_1/t \wedge \cdots \wedge KL_n/t \rightarrow KL/t$
  - $a : \neg K\neg L_1/t \wedge \cdots \wedge \neg K\neg L_n/t \rightarrow \neg K\neg L/t$
- **prec** $L \Rightarrow$ **prec** KL
- **Merge** actions in O': for each lit L and merge $m \in M$ with $m = \{t_1, \ldots, t_n\}$

$$merge_{L,m} : KL/t_1 \wedge \ldots \wedge KL/t_n \rightarrow KL$$

Properties of Translation $K_{T,M}$

- If T contains only the empty tag, $K_{T,M}(\Pi)$ reduces to $K_0(\Pi)$
- $K_{T,M}(\Pi)$ is always sound

We will see that...

- For suitable choices of T,M translation is **complete**
- ... and sometimes **polynomial** as well

Intuition of soundness

- Idea:
  - if sequence of actions π makes KL/t true in $K_{T,M}(\Pi)$
  - π makes L true in Π over all **trajectories** starting at initial states satisfying t

Theorem (Soundness $K_{T,M}(\Pi)$)

If π is a **plan that solves the classical** *planning problem $K_{T,M}(\Pi)$, then the action sequence π' that results from π by dropping the merge actions is a* **plan that solves the conformant** *planning problem Π.*

A complete but exponential instance of $K_{T,M}(\Pi)$: K_{s0}

If possible initial states are $s_0^1, \ldots, s_0^n$, scheme K_{s0} is the instance of $K_{T,M}(\Pi)$ with

- $T = \{\ \{\}, s_0^1, \ldots, s_0^n\ \}$
- $M = \{\ \{s_0^1, \ldots, s_0^n\}\ \}$
  i.e., only one merge for the disjunction of possible initial states
- Intuition: applying actions in K_{s0} keeps track of each fluent for each possible initial states
- This instance is complete, but exponential in the number of fluents
  - ...although not a bad conformant planner

Performance of K_{s0} + FF

		Planners exec time (s)			
Problem	$\#S_0$	K_{s0}	KP	POND	CFF
Bomb-10-1	1k	648,9	0	1	0
Bomb-10-5	1k	2795,4	0,1	3	0
Bomb-10-10	1k	5568,4	0,1	8	0
Bomb-20-1	1M	$> 1.8G$	0,1	4139	0
Sqr-4-16	4	0,3	fail	1131	13,1
Sqr-4-24	4	1,6	fail	$> 2h$	321
Sqr-4-48	4	57,5	fail	$> 2h$	$> 2h$
Sortnet-6	64	2,2	fail	2,1	fail
Sortnet-7	128	27,9	fail	17,98	fail
Sortnet-8	256	$> 1.8G$	fail	907,1	fail

Translation time included in all tables.