Planning with Uncertainty

PAH 2015

Classical vs. Uncertainty Planning

- what have you learnt so far?
  - sequential decision making
  - deterministic effects of actions
  - static environment
  - perfect observation
  - perfect sensors

Classical vs. Uncertainty Planning

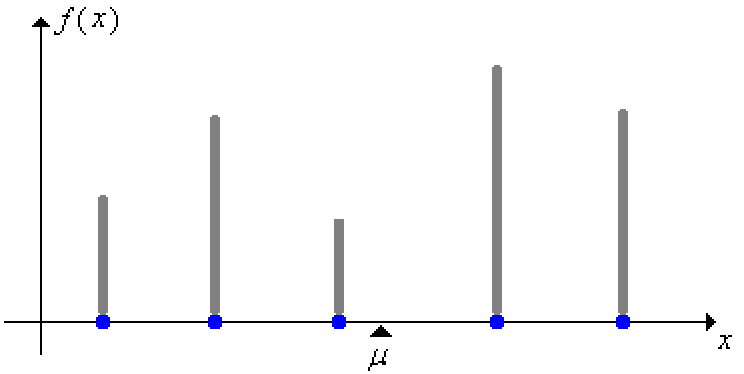
- the world is not perfect
  - actions may fail or yield unexpected results
  - the environment may change due to other agents
  - the agent does not have knowledge about whole situation
  - other agents can have antagonistic objectives
  - sensors are not precise

  - first step towards more realistic setting
  - planning with uncertainty

Classical vs. Uncertainty Planning

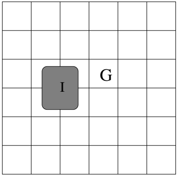
- Uncertainty modeling
  - non-determinism
    - a limited set of outcomes of actions
    - unlimited possible failures (conformant/contingency)
    - limited possible failures (fault tolerant)

  - probability
    - all possible outcomes with probability distribution
    - perfect observability (MDP)
    - partial observability (POMDP)



Conformant Planning

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)

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 P)

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

- $P' = \{Kp, K\neg p \mid p \in P\}$
- $I' = \{Kp \mid \text{ clause } p \in I\}$
- $G' = \{Kp \mid p \in G\}$
- $O' = O$ 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 conformant plan for Π, but not vice versa.

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: $\langle c, p \rangle$	$\Rightarrow$	$\left\{ \begin{array}{l} a : Kc \rightarrow Kp \\ a : K\neg c \rightarrow \emptyset \\ a : \neg K\neg c \rightarrow \neg K\neg p \end{array} \right.$

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

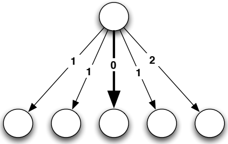
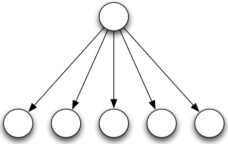
		Planners exec time (s)			
Problem	$\#S_0$	K_{s0}	KP	POND	CFF
Bomb-10-1	1k	648,9	0	1	0
Bomb-10-5	1k	2795,4	0,1	3	0
Bomb-10-10	1k	5568,4	0,1	8	0
Bomb-20-1	1M	$> 1.8G$	0,1	4139	0
Sqr-4-16	4	0,3	fail	1131	13,1
Sqr-4-24	4	1,6	fail	$> 2h$	321
Sqr-4-48	4	57,5	fail	$> 2h$	$> 2h$
Sortnet-6	64	2,2	fail	2,1	fail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

Translation time included in all tables.

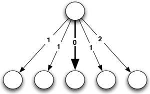
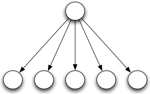
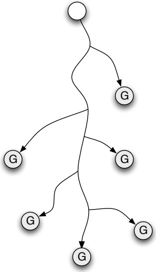
Fault Tolerant Planning: Complexity and Compilation

Carmel Domshlak

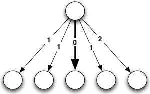
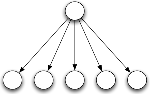
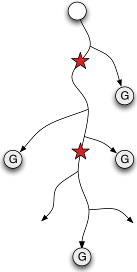
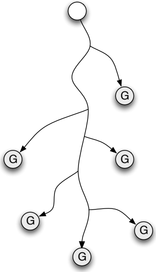
Action Dynamics and Solution Concepts

Classical	Fault Tolerant	FOND
		

Action Dynamics and Solution Concepts

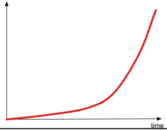
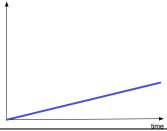
Classical	Fault Tolerant	FOND
		
(weak) plan		strong plan
		

Action Dynamics and Solution Concepts

<table>
<tr><th>Classical</th><th>Fault Tolerant</th><th>FOND</th></tr>
<tr>
<td></td>
<td>
</td>
<td></td>
</tr>
<tr><td>(weak) plan</td><td>κ-plan</td><td>strong plan</td></tr>
<tr>
<td>
</td>
<td>
</td>
<td>
</td>
</tr>
</table>

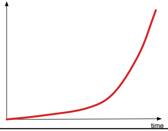
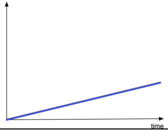
Between Classical and (FOND) Contingent Planning

Between Bold Optimism and Paranoia

Classical	We control the nature. ☺ No bad things will happen!	PSPACE / NP	
FOND	Nature tries to full us. ☹ Bad things always happen ...	EXPTIME	

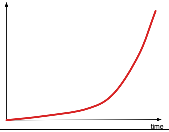
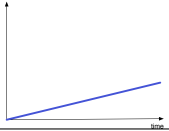
Between Classical and (FOND) Contingent Planning

Between Bold Optimism and Paranoia

Classical	We control the nature. No bad things will happen!	PSPACE / NP	
	Nature tries to full us, but it has other things to do as well.		
FOND	Nature tries to full us. Bad things always happen ...	EXPTIME	

Between Classical and (FOND) Contingent Planning

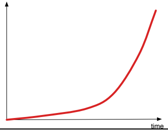
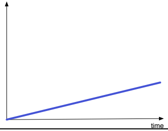
Between Bold Optimism and Paranoia

Classical	We control the nature. ☺ No bad things will happen!	PSPACE / NP	
Fault Tolerant	Nature tries to full us, but it has other things to do as well. 😐 At most κ bad things will happen.		
FOND	Nature tries to full us. ☹ Bad things always happen ...	EXPTIME	

Jensen, Veloso, & Bryant, ICAPS'04

Between Classical and (FOND) Contingent Planning

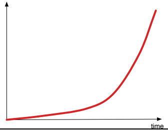
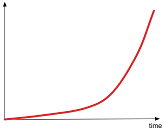
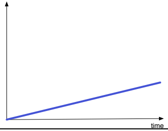
Between Bold Optimism and Paranoia

Classical	We control the nature. No bad things will happen!	PSPACE / NP	
Fault Tolerant	Nature tries to full us, but it has other things to do as well. At most κ bad things will happen.	EXPTIME PSPACE / NP	
FOND	Nature tries to full us. Bad things always happen ...	EXPTIME	

Here: COMPLEXITY

Between Classical and (FOND) Contingent Planning

Between Bold Optimism and Paranoia

Classical	We control the nature. ☺ No bad things will happen!	PSPACE / NP	
Fault Tolerant	Nature tries to full us, but it has other things to do as well. 😐 At most κ bad things will happen.	EXPTIME PSPACE / NP	$\Downarrow$ 
FOND	Nature tries to full us. ☹ Bad things always happen ...	EXPTIME	

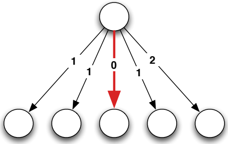
Here: COMPILATION

Task Classification and Decision Problems

FT task classification

Task is α-primary if

each action has at most α primary ($=$ 0-failures) effects

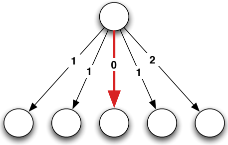


Task Classification and Decision Problems

FT task classification

Task is α-primary if

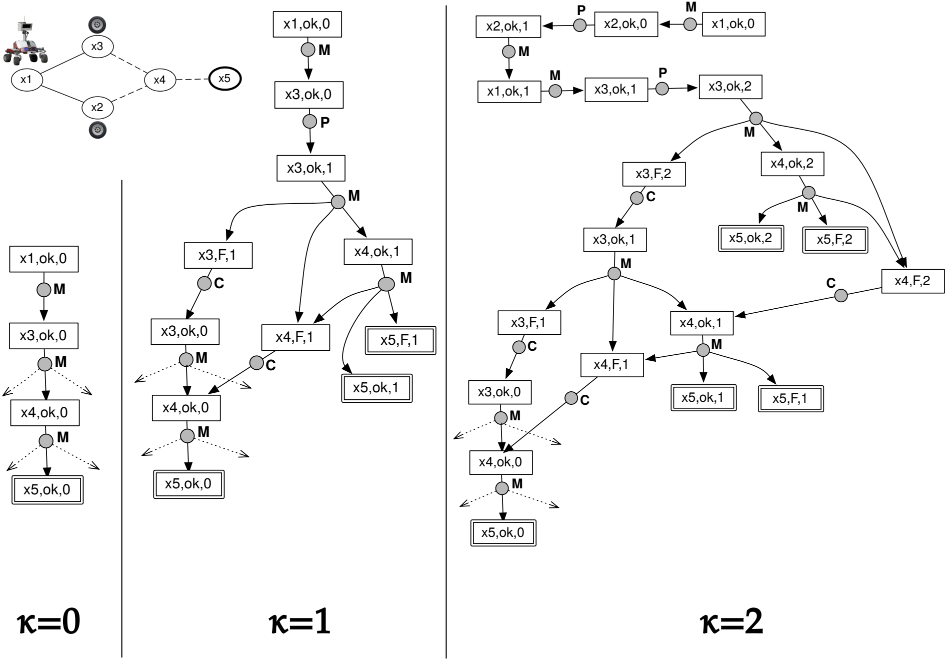
each action has at most α primary ($=$ 0-failures) effects



Decision problems

FT-α-κ: Does α-primary Π have a κ-plan?

POLY-FT-α-κ: Does α-primary Π have a κ-plan such that all its κ-admissible executions reach the goal after a polynomial number of steps?

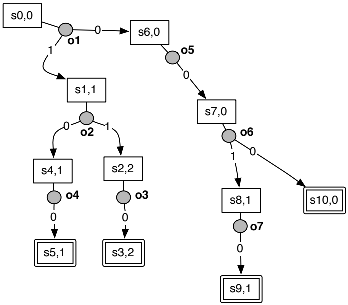

x1
x2
x3
x4
x5
x1,ok,0
M
x3,ok,0
M
x4,ok,0
M
x5,ok,0
κ=0
x1,ok,0
M
x3,ok,0
P
x3,ok,1
M
x3,F,1
C
x3,ok,0
M
x4,ok,0
M
x5,ok,0
x4,F,1
C
x4,ok,1
M
x5,F,1
x5,ok,1
κ=1
x1,ok,0
M
x2,ok,0
P
x2,ok,1
M
x1,ok,1
M
x3,ok,1
P
x3,ok,2
M
x3,F,2
C
x3,ok,1
M
x3,F,1
C
x3,ok,0
M
x4,ok,0
M
x5,ok,0
x4,F,1
C
x4,ok,1
M
x5,ok,1
x5,F,1
x4,ok,2
M
x5,ok,2
x5,F,2
x4,F,2
C
κ=2

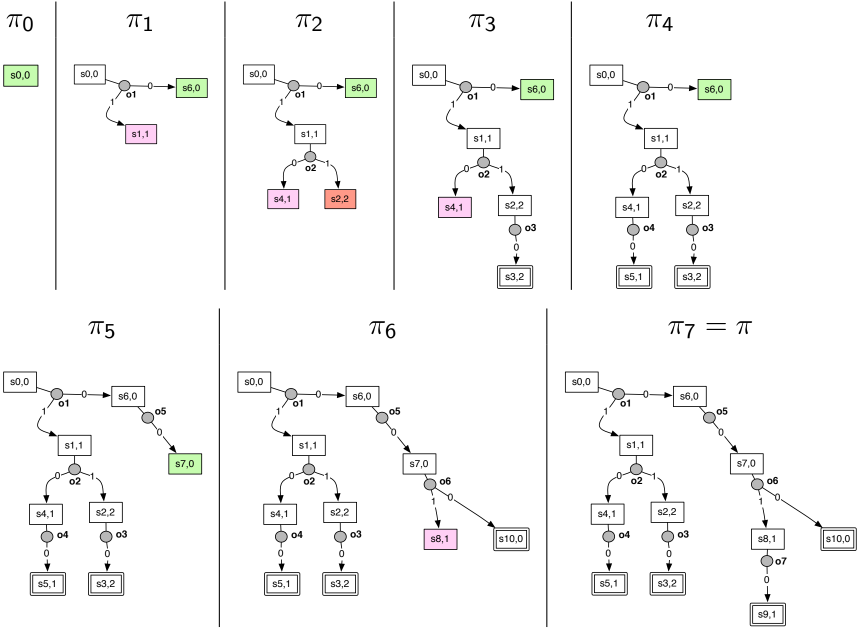
1-or-2-effects fragment of FT

Each action is either

- deterministic, or
- has two possible effects, one primary and one secondary.

Example: 2-plan for a 1-or-2-effects task:




π0
s0,0
π1
s0,0
o1
0
1
s6,0
s1,1
π2
s0,0
o1
s6,0
s1,1
o2
s4,1
s2,2
π3
s0,0
o1
s6,0
s1,1
o2
s4,1
s2,2
o3
s3,2
π4
s0,0
o1
s6,0
s1,1
o2
s4,1
s2,2
o4
o3
s5,1
s3,2
π5
s0,0
o1
s6,0
o5
s7,0
s1,1
o2
s4,1
s2,2
o4
o3
s5,1
s3,2
π6
s0,0
o1
s6,0
o5
s7,0
o6
s8,1
s10,0
s1,1
o2
s4,1
s2,2
o4
o3
s5,1
s3,2
π7 = π
s0,0
o1
s6,0
o5
s7,0
o6
s8,1
s10,0
o7
s9,1
s1,1
o2
s4,1
s2,2
o4
o3
s5,1
s3,2

Key property of $\textsc{ft}$-1-κ (within "1-or-2-effects")

Property

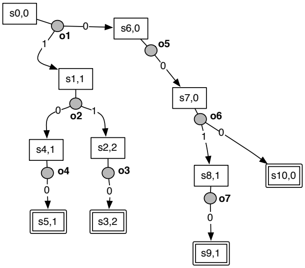
Any irreducible κ-plan induces such a DFS-ordered sequence of sub-plans with *"at most one non-goal leaf with j failures so far."*

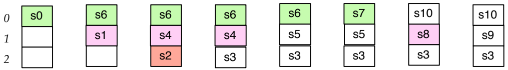
Key property of FT-1-κ (within "1-or-2-effects")

Property

Any irreducible κ-plan induces such a DFS-ordered sequence of sub-plans with *"at most one non-goal leaf with j failures so far."*

- Key enabler for the compilation
- In the paper: Generalization to $O(1)$-effects per action



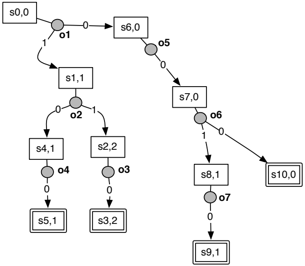


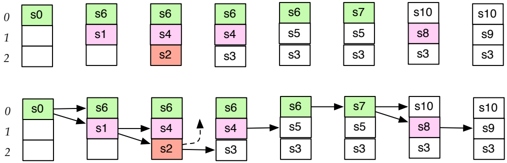
Key property of FT-1-κ (within "1-or-2-effects")

Property

Any irreducible κ-plan induces such a DFS-ordered sequence of sub-plans with *"at most one non-goal leaf with j failures so far."*

- Key enabler for the compilation
- In the paper: Generalization to $O(1)$-effects per action



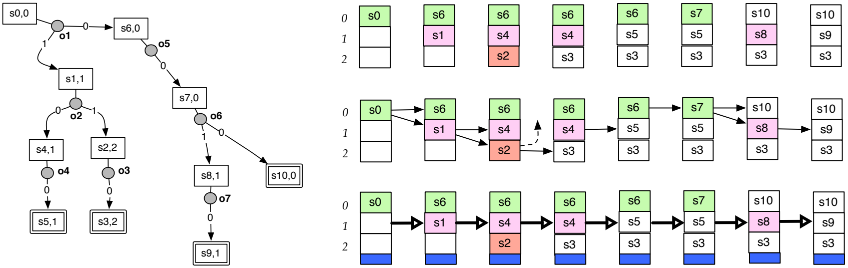


Key property of FT-1-κ (within "1-or-2-effects")

Property

Any irreducible κ-plan induces such a DFS-ordered sequence of sub-plans with *"at most one non-goal leaf with j failures so far."*

- Key enabler for the compilation
- In the paper: Generalization to $O(1)$-effects per action



Tiny evaluation

- Robot to move from BL to TR of a 7×7, 4-connected grid
- Edges: unsafe/safe with $p/(1-p)$
  - Safe $\rightsquigarrow$ deterministic
  - Unsafe $\rightsquigarrow$ can get a flat and stay
- 10 spare tires placed randomly on the grid

	CFF	CFF $\left(\Pi^{(\mathcal{F},\kappa)}\right)$				FD (Π')			
task		0	1	2	4	0	1	2	4
$p = 0.1$	0.12	0.00	–	–	–	0.00	0.02	0.03	0.06
	0.13	0.00	2.10	–	–	0.00	1.67	0.04	0.07
	0.13	0.00	–	–	–	0.00	0.21	0.03	0.07
	0.13	0.00	–	–	–	0.00	0.02	0.03	0.06
	0.13	0.00	–	–	–	0.00	0.09	0.04	0.07
$p = 0.2$	0.13	0.00	–	–	–	0.00	27.32	0.04	0.08
	0.13	0.00	–	–	–	0.00	0.01	0.03	0.06
	0.13	0.00	–	–	–	0.00	0.02	0.03	0.06
	0.13	0.00	–	–	–	0.00	0.01	0.03	0.06
	0.13	0.00	–	–	–	0.00	5.96	0.04	0.07
$p = 0.5$	0.13	0.00	–	–	–	0.00	0.38	0.05	0.09
	0.13	0.00	3.32	4.13	–	0.00	0.04	0.63	11.56
	0.13	0.00	–	–	–	0.00	0.31	38.86	–
	0.13	0.00	0.14	0.15	0.15	0.00	0.01	0.03	0.06
	0.13	0.00	–	–	–	0.00	0.89	17.37	1.25

Probabilistic Planning

Classical vs. Probabilistic Planning

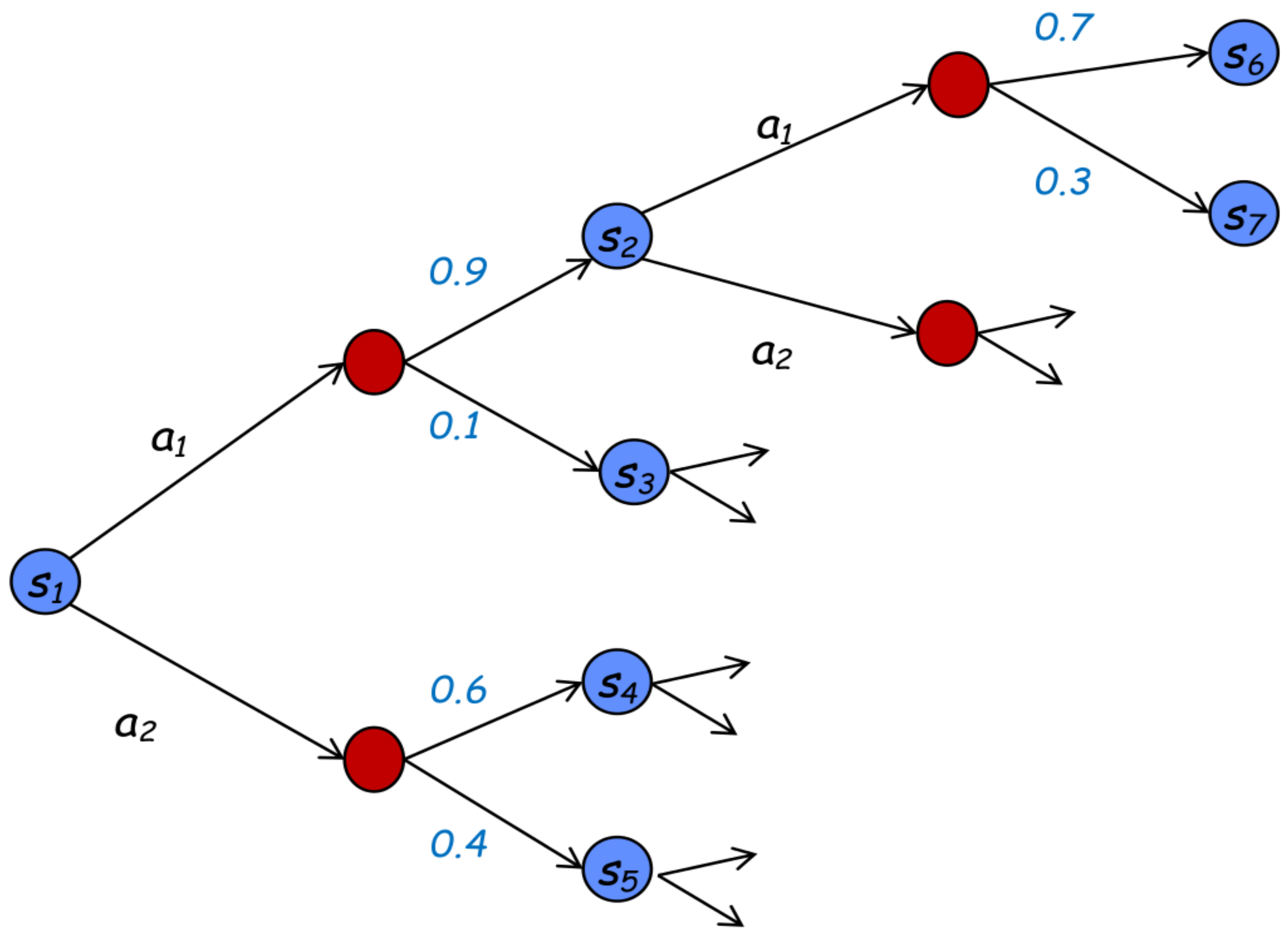
- Classical Planning: $\langle S, s_0, S_G, A, f, c \rangle$
  - states, initial state, goal state(s)
  - actions
  - transition function $f: S \times A \rightarrow S$
  - cost function

- Probabilistic Planning
  - probabilistic transition function $P: S \times A \times S \rightarrow [0,1]$

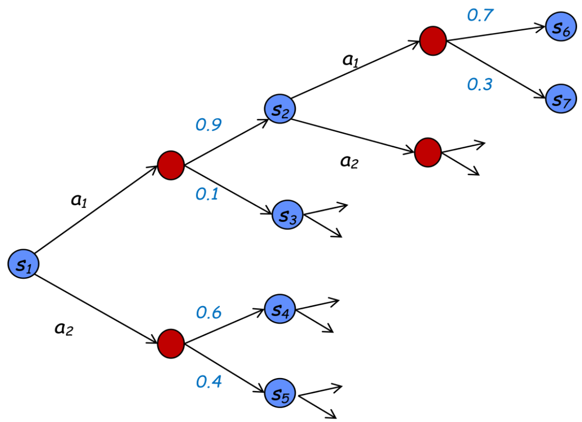
$$\sum_{s' \in S} P(s, a, s') = 1$$

Q: why is this enough for modelling uncertainty in environment?

Probabilistic Planning - Visualization



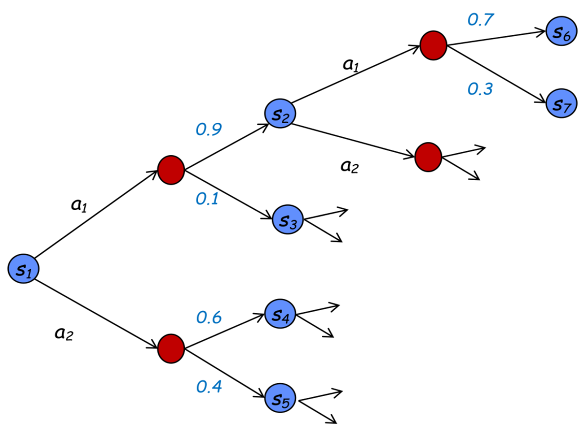
Probabilistic Planning - Solution



- what is the solution in classical planning?
  - sequence of (partially) ordered actions leading from initial state to the goal state
- this is not sufficient in the probabilistic case
  - what if the plan fails?
- we need **a (partial) policy**

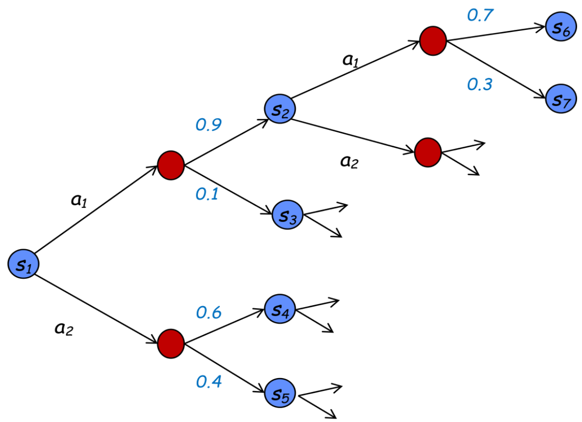
Probabilistic Planning - Solution

- in general we seek for a probabilistic history-dependent policy
  - $\pi: H \times A \rightarrow [0,1]$
  - where $h = s_1 a_1 s_2 a_2 \ldots s_t$
  - note that the policy may prescribe randomization over actions



- now we have a representation for plans (policy)
  - we need a method for plan evaluation

Probabilistic Planning - Evaluation



- costs are assigned to triplets (s, a, s')
- typically termed rewards (i.e., positive sense)
- executing a policy yields a sequence of rewards
- policy value – linear additive utility
  - $u(R_1, R_2, \dots) = R_1 + \gamma R_2 + \gamma^2 R_3 + \cdots$
  - $u(\pi(s_0)) = E[u(R_1, \dots)]$
- expected utility – what can happen?
  - optimal only for risk-neutral agent

Probabilistic Planning – Optimal Solution

- If the quality of every policy can be measured by its expected linear additive utility, there is a policy that is optimal at every time step.

(Stated in various forms by Bellman, Denardo, and others)

- we seek for π^* s.t. $u(\pi^*) \geq u(\pi)$ for all other policies π

- note: can be the case that the policy cannot be measured by expected linear additive utility?
  - yes (infinite state-space with non-discounted rewards, dead-ends, …)

Probabilistic Planning – Algorithms

- this lecture
  - using classical planning to probabilistic planning
  - straightforward approach (FF-replan)
  - improved approach (Robust FF)
  - “multi-layered” approach (FF-Hindsight Optimization)
- next lectures
  - algorithms that directly use probability and uncertainty
    - formal definition MDP, strategy/policy iteration
    - current approaches for solving MDPs
  - uncertainty in observations
    - formal definition and current approaches for solving POMDPs

Probabilistic Planning – First Approach

- 2004 – first international probabilistic planning competition
- several participants, mainly based on MDP solvers
- winner?
  - FF-Replan
  - possibly the simplest algorithm you can think of …

FF-Replan

- outline of the algorithm
  1. determinizes the input domain (remove all probabilistic information from the problem)
  2. synthesizes a plan
  3. executes the plan
  4. should an unexpected state occur, replans

FF-Replan - Determinization

- what information can be discarded?
- two main heuristics
  - keep only one from all probabilistic outcomes of an action in a state (e.g., using the outcome with the highest probability)
  - keep all outcomes
    - generate a separate action for each possible outcome
- very simple, not sound, not optimal, but still good enough for simple domains
- (outperformed also all participants in IPPC-06)

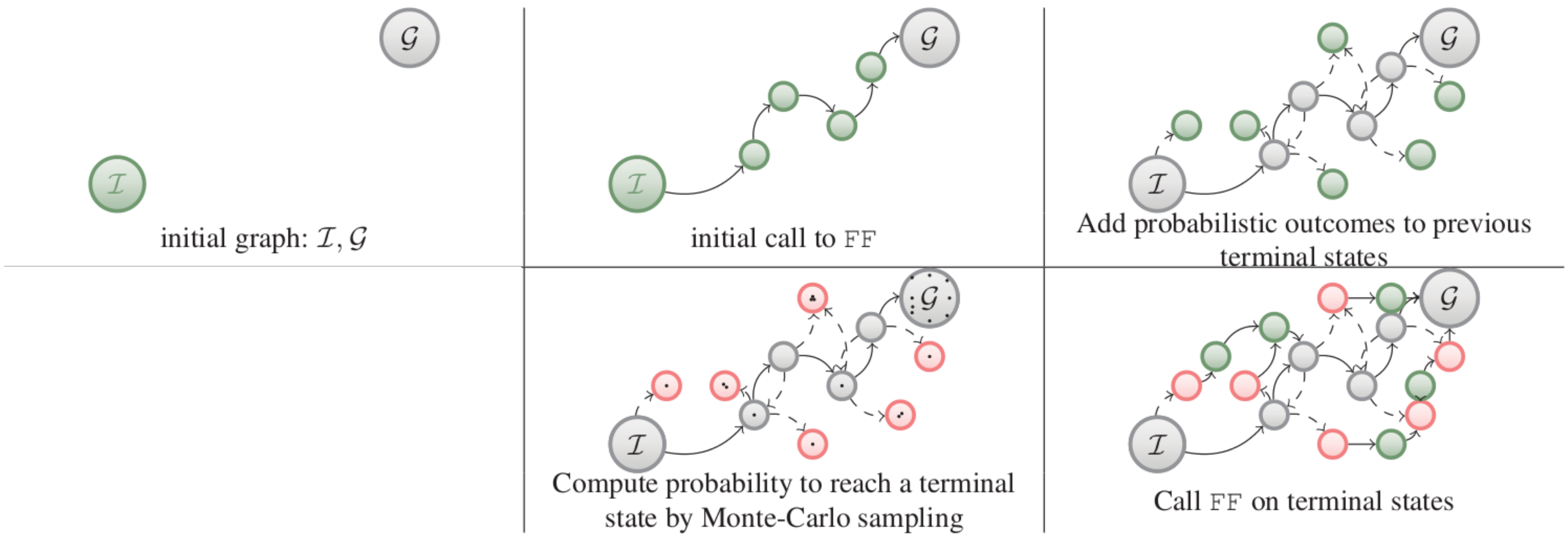
Probabilistic Planning (2)

- winner of IPPC 2008
  - Robust-FF
    - (Incremental Plan Aggregation for Generating Policies in MDPs, Konigsbuch, Kuter, Infantes 2010)
  - generalizes FF-Replan

1. determinize the problem
2. use classical planner to find partial plans
3. aggregate these plans into the partial policy
4. continue until the probability of replanning is below given threshold

Robust-FF

- outline of the algorithm



Robust-FF

- number of options
  - selecting determinization (most probable, all outcomes)
  - selecting goals (only problem goals, random goals, best goals)
    - random/best goals – include also expanded states into G_{FF}; either k random, or k "best ones"
  - calculating probability of reaching terminal states (dynamic programming, Monte Carlo simulations)
- soundness vs. completeness of the algorithm?
  - only with selected methods (RFF_{AO})
- not (approximately) optimal in general

Hindsight Optimization (HOP) – FF-Hindsight

- Approximate the value of a state
  - sample a set of determinized problems originating from that state
  - then solve the problems "in hindsight" and combine their values
  - if the deterministic problems are easier → computational gains
- Optimal value function

$$V^*(s,T) = \max_{\pi} \boldsymbol{E}[R(s,F,\pi)]$$

  - state *s*, horizon *T*, (non-stationary) policy π, total reward *R* and random variable *F* uniformly distributed over all futures
- HOP value function approximation

$$V_{hs}(s,T) = \boldsymbol{E}[\max_{\pi} R(s,F,\pi)]$$