Probabilistic Planning

Jim Blythe
Some ‘classical planning’ assumptions

- Atomic time
- All effects are immediate
- Deterministic effects
- Omniscience
- Sole agent of change is the planning agent
- Goals of attainment
Some ‘classical planning’ assumptions

- Atomic time                      (10/02/03)
- All effects are immediate         (10/02/03)
- Deterministic effects
- Omniscience                      (10/28/03)
- Sole agent of change              (10/16/03)
- Goals of attainment               (11/13/03)
Sources of uncertainty

When we try to execute plans, uncertainty from several different sources can affect success.

Firstly, we might have uncertainty about the state of the world.
Sources of uncertainty

Actions we take might have uncertain effects even when we know the world state.
Sources of uncertainty

*External agents* might be changing the world while we execute our plan.
Dealing with uncertainty: re-planning

- Make a plan assuming nothing bad will happen
- Monitor for problems during execution
- Build a new plan if a problem is found
  - Either re-plan to the goal state
  - Or try to patch the existing plan
Dealing with uncertainty: Conditional planning

- Deal with *contingencies* (bad outcomes) at planning time before they occur

- Universal planning might be viewed as conditional planning where every possible contingency is covered (somehow) in the policy.
Tradeoffs in strategies for uncertainty

- My **re-planner** housemate: “Why are you taking an umbrella? It’s not raining!”
  - Can’t find plans that require steps taken **before** the contingency is discovered

- My **conditional planner** housemate: “Why are you leaving the house? Class may be cancelled. It might rain. You might have won the lottery. Was that an earthquake?....”
  - Impossible to plan for every contingency. Need a representation that captures tradeoffs.
Probabilistic planning lets us explore the middle ground

- Partial knowledge about our uncertainties: different contingencies have different probabilities of occurring.

- Plan ahead for *likely* contingencies that may need steps taken before they occur.

- Use probability theory to judge plans that address some contingencies:
  seek a plan that is above some minimum probability of success.
Some issues to think about

- How do we figure out the probability of a plan succeeding? Is it expensive to do?

- How do we know what the most likely contingencies are?

- Can we distinguish bad outcomes (not holding the cup) from really bad outcomes (broken the cup, spilled the sulfuric acid..)?
More issues

Why not just do all this with MDPs/POMDPs?

- We’ll look at MDP-based approaches in a later class
- With MDPs, need to control potential state space explosion
- Most approaches (of both kinds) use compact action representations
- MDP-based approaches can find optimal solutions and deal elegantly with costs.
Representing actions with uncertain outcomes

Deterministic effects

Non-deterministic effects

Non-deterministic effects with conditional probabilities
Reminder: POP algorithm

POP((A, O, L), agenda, PossibleActions):

- If agenda is empty, return (A, O, L)
- Pick (Q, An) from agenda
- choose an action Ad that adds Q.
  
  ![Diagram](Ap \xrightarrow{Q} Ac)
Buridan (an SNLP-based planner)

- An SNLP-based planner might come up with this plan for a deterministic action representation:
A plan that works 70% of the time..
Modifications to the UCPOP algorithm

- Allow more than one causal link for each condition in the plan.

- Confront a threat by decreasing the probability that it will happen. (By adding conditions negating the trigger of the threat).

- Terminate when sufficient probability reached (may still have threats).
Computing probability of plan success
1: forward projection

- Simulate the plan, keep track of possible states and their probabilities, finally sum the probabilities of states that satisfy the goal.

- Here, the china is packed in the initial state with probability 0.5 (and is not packed with probability 0.5)

<table>
<thead>
<tr>
<th>Step</th>
<th>State</th>
<th>Features</th>
<th>Parent</th>
<th>Prob</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pack-china</td>
<td>$S^1_{pack}$</td>
<td>packed</td>
<td>I</td>
<td>0.5</td>
</tr>
<tr>
<td>Pack-china</td>
<td>$S^2_{pack}$</td>
<td>¬ packed</td>
<td>I</td>
<td>0.5</td>
</tr>
<tr>
<td>Put-in-china</td>
<td>$S^1_{put}$</td>
<td>packed, in-car</td>
<td>$S^1_{pack}$</td>
<td>0.5</td>
</tr>
<tr>
<td>Put-in-china</td>
<td>$S^2_{put}$</td>
<td>¬ packed, in-car</td>
<td>$S^2_{pack}$</td>
<td>0.5</td>
</tr>
<tr>
<td>Drive-china</td>
<td>$S^1_{d}$</td>
<td>packed, ¬ broken</td>
<td>$S^1_{put}$</td>
<td>0.475</td>
</tr>
<tr>
<td>Drive-china</td>
<td>$S^2_{d}$</td>
<td>packed, broken</td>
<td>$S^2_{put}$</td>
<td>0.025</td>
</tr>
<tr>
<td>Drive-china</td>
<td>$S^1_{d}$</td>
<td>¬ packed, ¬ broken</td>
<td>$S^1_{put}$</td>
<td>0.35</td>
</tr>
<tr>
<td>Drive-china</td>
<td>$S^2_{d}$</td>
<td>¬ packed, broken</td>
<td>$S^2_{put}$</td>
<td>0.15</td>
</tr>
</tbody>
</table>

What is the worst-case time complexity of this algorithm?
Computing the probability of success
2: Bayes nets

What is the worst-case time complexity of this algorithm?
Tradeoffs in computing probability of success

- Belief net approach is often faster because it ignores irrelevant differences in the state.

- Neither approach is guaranteed to be faster.

- Often, the time to compute the probability of success dominates the planning time.
Conditional planning in this framework: CNLP and C-Buridan

- Tricky to represent conditional branches in partially-ordered plans.

- Actions can produce “observation labels” as well as effects, e.g. “the weather is good”.

- After introducing an action with observation labels, the possible values can be used as “context labels” assigned to actions ordered after the observation step.
Example: drive around the mountain

Initial

(at car old-apartment)
(at car old-apartment)
\sim(broken china)

Put-in ?x

(at car ?x loc)
(at ?x ?x loc)

Get-weather-report

ok
bad

Drive-china-over-mountain ?x ?y

[ok]

Drive-china-around-mountain ?x ?y

\sim(broken china)
(at car new-apartment)

Goal

(at car ?x)
(at ?x ?x)

(in china car)

[in china car]

[bad]
DRIPS:
(Decision-theoretic Refinement Planner)

- Considers plan utility, taking into account action costs, benefits of different states.

- Searches for a plan with Maximum Expected Utility (MEU), not just above a threshold.

- A skeletal planner, makes use of ranges of utility of abstract plans in order to search efficiently.

- Prune abstract plans whose utility range is completely below the range of some alternative (dominated plans)
Abstract action for moving china

- Utility ranges used to compute dominance between alternative plans.
Abstract plan

move china

load-up-china

pack-and-load

load

pack-china

load

drive-china

drive-around-mountain

drive-over-mountain
MAXPLAN

- Inspired by SATPLAN. Compile planning problem to an instance of E-MAJSAT

- E-MAJSAT: given a boolean formula with variables that are either *choice* variables or *chance* variables, find an assignment to the choice variables that maximizes the probability that the formula is true.

- Choice variables: we can control them
  - e.g. which action to use

- Chance variables: we cannot control them
  - e.g. the weather, the outcome of each action, ..

- Then use standard algorithm to compute and maximize probability of success
Example operators with chance variables

**dig-moat**

1: moat
   - moat
     - T: 1.0, F: 0.5

2: castle
   - castle
     - T: 1.0, F: 0.0

**erect-castle**

1: castle
   - castle
     - T: 1.0
     - F: 0.75
       - moat
         - e₁: 0.67, e₂: 0.25
       - castle: new
         - e₃: 1.0
         - e₄: 0.5

2: moat
   - moat
     - T: 1.0, F: 0.0
Clauses with chance variables

\[ \text{dig-moat-1} \land \neg \text{moat-0} \land d_{1-1} \rightarrow \text{moat-1} \]

- Solved with Davis-Putnam-Logemann-Loveland algm
Thinking about MAXPLAN

- As it stands, does MAXPLAN build conditional plans?
- How could we make MAXPLAN build conditional plans?
Other approaches that have been used

- Graphplan
  (pointers to Weld and Smith’s work in paper)

- Prodigy (more in next class)

- HTN planning (Cypress)

- Markov decision problems
  (more in the class after next)
With all approaches, we must consider the same issues

- **Tractability**
  - Plans can have many possible outcomes
  - How to reason about when to add sensing

- **Plan utility**
  - Is probability of success enough?
  - What measures of cost and benefit can be used tractably?
  - Can operator costs be summed? What difference do time-based utilities like deadlines make?

- **Observability and conditional planning**
  - Classical planning is “open-loop” with no sensing
  - A “policy” assumes we can observer everything
  - Can we model limited observability, noisy sensors, bias..?