Lecture 6 - Planning under Certainty

Jesse Hoey
School of Computer Science
University of Waterloo

February 3, 2020

Readings: Poole & Mackworth (2nd ed.) Chapt. 6.1-6.4

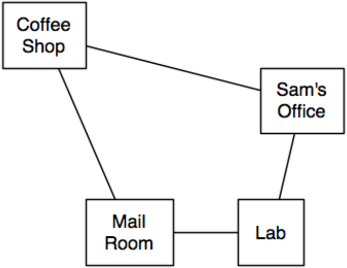
Planning

- Planning is deciding what to do based on an agent's ability, its goals, and the state of the world.
- Planning is finding a sequence of actions to solve a goal.
- Initial assumptions:
  - A single agent
  - The world is deterministic.
  - There are no exogenous events outside of the control of the agent that change the state of the world.
  - The agent knows what state it is in (full observability)
  - Time progresses discretely from one state to the next.
  - Goals are predicates of states that need to be achieved or maintained.

Actions

- A deterministic action is a partial function from states to states.
- partial function: some actions not possible in some states
- The preconditions of an action specify when the action can be carried out.
- The effect of an action specifies the resulting state.

Delivery Robot Example



Features (Variables):

RLoc – Rob's location
(4-valued: {cs,off,mr,lab})
RHC – Rob has coffee (binary)
SWC – Sam wants coffee (binary)
MW – Mail is waiting (binary)
RHM – Rob has mail (binary)

Actions:

mc – move clockwise
mcc – move counterclockwise
puc – pickup coffee
dc – deliver coffee
pum – pickup mail
dm – deliver mail

Explicit State-space Representation

State	Action	Resulting State
$\langle lab, \neg rhc, swc, \neg mw, rhm \rangle$	mc	$\langle mr, \neg rhc, swc, \neg mw, rhm \rangle$
$\langle lab, \neg rhc, swc, \neg mw, rhm \rangle$	mcc	$\langle off, \neg rhc, swc, \neg mw, rhm \rangle$
$\langle off, \neg rhc, swc, \neg mw, rhm \rangle$	dm	$\langle off, \neg rhc, swc, \neg mw, \neg rhm \rangle$
$\langle off, \neg rhc, swc, \neg mw, rhm \rangle$	mcc	$\langle cs, \neg rhc, swc, \neg mw, rhm \rangle$
$\langle off, \neg rhc, swc, \neg mw, rhm \rangle$	mc	$\langle lab, \neg rhc, swc, \neg mw, rhm \rangle$
...	...	...

Feature-based representation of actions

For each action:

- precondition is a proposition that specifies when the action can be carried out.

For each feature:

- causal rules that specify when the feature gets a new value and
- frame rules that specify when the feature keeps its value.

Notation:

- Features are capitalized (e.g. $Rloc$, RHC)
- Values of the features are not (e.g. $Rloc = cs$, rhc, $\neg rhc$)
- If X is a feature, then X' is the feature after an action is carried out

Example feature-based representation

Precondition of pick-up coffee (*puc*):

$$RLoc = cs \wedge \neg rhc$$

Rules for location is *cs* (specifies RLoc'):

$$RLoc' = cs \leftarrow RLoc = off \wedge Act = mcc$$
$$RLoc' = cs \leftarrow RLoc = mr \wedge Act = mc$$
$$RLoc' = cs \leftarrow RLoc = cs \wedge Act \neq mcc \wedge Act \neq mc$$

Rules for "robot has coffee" (specifies rhc'):

$$(frame \quad rule): \quad RHC' = true \leftarrow RCH = true \wedge Act \neq dc$$
$$(causal \quad rule): \quad RHC' = true \leftarrow Act = puc$$

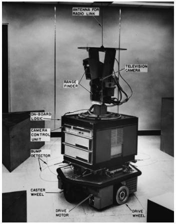
also write as:

$$rhc' \leftarrow rhc \wedge Act \neq dc$$
$$rhc' \leftarrow Act = puc$$

STRIPS Representation

- Previous representation was feature-centric: specify how each feature changes for each action that satisfies a precondition.
- STRIPS is action-centric: specify effects and preconditions for each action. For each action:
  - precondition that specifies when the action can be carried out.
  - effect a set of assignments of values to features that are made true by this action.

STRIPS:
STanford Research Institute Problem Solver



Example STRIPS representation

Frame assumption: all non-mentioned features stay the same.
Therefore, $V = v$ after *act* if:

- if $V = v$ was on effect list of *act* or
- if V is not on the effect list of *act*, and $V = v$ immediately before *act*

Example STRIPS representation

Pick-up coffee (puc):

- precondition: $[cs, \neg rhc]$
- effect: $[rhc]$

Deliver coffee (dc):

- precondition: $[off, rhc]$
- effect: $[\neg rhc, \neg swc]$

Planning

Given:

- A description of the effects and preconditions of the actions
- A description of the initial state
- A goal to achieve

find a sequence of actions that is possible and will result in a state satisfying the goal.

Forward Planning

Idea: search in the state-space graph.

- The nodes represent the states
- The arcs correspond to the actions: The arcs from a state s represent all of the actions that are legal in state s.
- A plan is a path from the state representing the initial state to a state that satisfies the goal.
- Can use any of the search techniques from Chap. 3

Example state-space graph

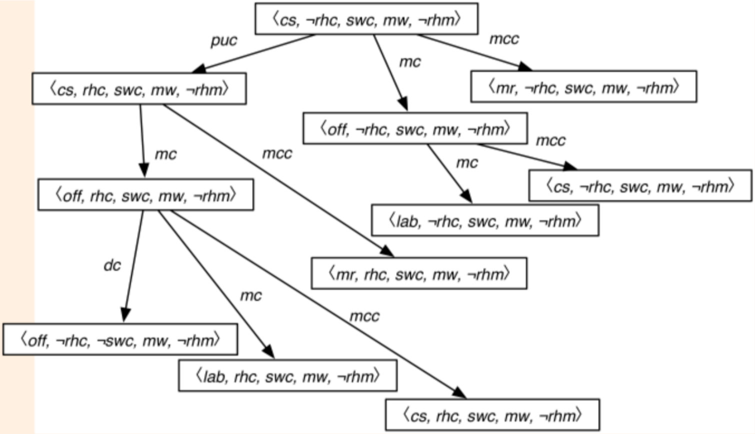


Figure 6.2: Part of the search space for a state-space planner

Regression Planning

Idea: search backwards from the goal description: nodes correspond to subgoals, and arcs to actions.

- Nodes are propositions: a formula made up of assignments of values to features
- Arcs correspond to actions that can achieve one of the goals
- Neighbors of a node N associated with arc A specify what must be true immediately before A so that N is true immediately after.
- The start node is the goal to be achieved.
- $goal(N)$ is true if N is a proposition that is true of the initial state.

Regression example

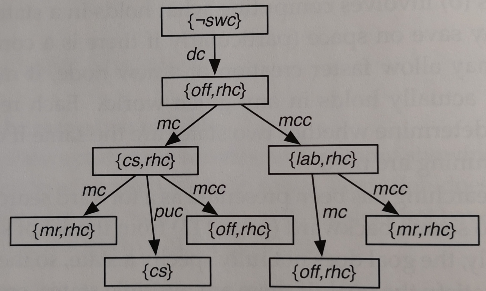


Figure 6.3: Part of the search space for a regression planner

Improving Efficiency

- You can define a heuristic function that estimates how difficult it is to solve the goal from the initial state.
- You can use domain-specific knowledge to remove impossible goals.
  - It is often not obvious from an action description to conclude that an agent can only hold one item at any time.
  - e.g. if we have $(\neg rhc, rhm)$ and we regress through deliver coffee, then we will have an impossible state (rhc,rhm). There is no sequence of actions from the initial state $(\neg rhc, \neg rhm)$ that will acheive this.
- A tutorial by Malte Helmert on Heuristics for Deterministic Planning: `https://ai.dmi.unibas.ch/misc/tutorial_aaai2015/`

Comparing forward and regression planners

- Which is more efficient depends on:
  - The branching factor
  - How good the heuristics are
- Forward planning is unconstrained by the goal (except as a source of heuristics).
- Regression planning is unconstrained by the initial state (except as a source of heuristics)

Planning as a CSP

- Search over planning horizons.
- For each planning horizon, create a CSP constraining possible actions and features
  - Choose a planning horizon k.
  - Create a variable for each state feature and each time from 0 to k.
  - Create a variable for each action feature for each time in the range 0 to $k-1$.

Constraints

- state constraints: between variables at the same time step.
- precondition constraints: between state variables at time t and action variables at time t that specify what actions are available from a state.
- effect constraints: between state variables at time t, action variables at time t and state variables at time $t+1$.
- frame constraints: between state variables at time t, action variables at time t and state variables at time $t+1$ specify that a variable does not change
- initial state constraints that are usually domain constraints on the initial state (at time 0).
- goal constraints that constrains the final state to be a state that satisfies the goals that are to be achieved.

CSP for Delivery Robot (horizon=2)

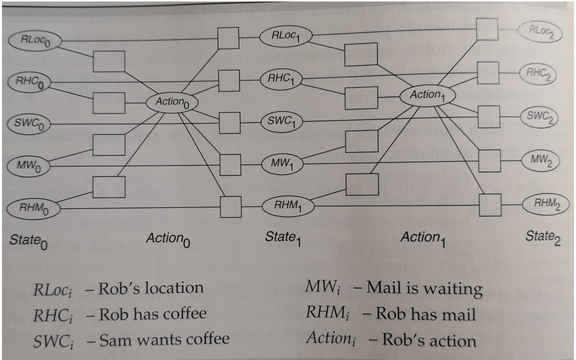


Figure 6.4: The delivery robot CSP planner for a planning horizon of $k = 2$

at time i:

$RLoc_i$ — Rob's location
RHC_i — Rob has coffee
SWC_i — Sam wants coffee
MW_i — Mail is waiting
RHM_i — Rob has mail

$Action_i$ — Rob's action
$SWC_0 = true$ — initial state
$RHC_0 = false$ — initial state
$SWC_2 = false$ — Goal

Next:

- Supervised Learning (Poole & Mackworth (2nd ed.) Chapter 7.1-7.6)
- Uncertainty (Poole & Mackworth (2nd ed.) Chapter 8)