Planning: Regression Planning

CPSC 322 Lecture 16

February 8, 2006
Textbook §11.2
Lecture Overview

Recap

Regression Planning
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.
Example state-space graph

**Actions**
- mc: move clockwise
- mac: move anticlockwise
- nm: no move
- puc: pick up coffee
- dc: deliver coffee
- pum: pick up mail
- dm: deliver mail

**Locations:**
- cs: coffee shop
- off: office
- lab: laboratory
- mr: mail room

**Feature values**
- rhc: robot has coffee
- swc: Sam wants coffee
- mw: mail waiting
- rhm: robot has mail

**Planning:** Regression Planning
Forward search can use **domain-specific knowledge** specified as:

- a **heuristic function** that estimates the number of steps to the goal
- **domain-specific pruning of neighbors**:
  - don’t go to the coffee shop unless “Sam wants coffee” is part of the goal and Rob doesn’t have coffee
  - don’t pick-up coffee unless Sam wants coffee
  - unless the goal involves time constraints, don’t do the “no move” action.
Lecture Overview

Recap

Regression Planning
Regression Planning

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

- **Nodes** are propositions: partial assignments to state variables
- **Start node**: the goal condition
- **Arcs** correspond to actions
- A node that **neighbours** \( N \) via arc \( A \) is a variable assignment that specifies what must be true immediately before \( A \) so that \( N \) is true immediately after.
- The **goal test** is true if \( N \) is a proposition that is true of the initial state.
Defining nodes and arcs

- A node $N$ is a partial assignment of values to variables:
  
  \[
  [X_1 = v_1, \ldots, X_n = v_n]
  \]

- An action which can be taken to this node is one that achieves one of the $X_i = v_i$, and does not achieve any $X_j = v_j$ where $v'_j$ is different from $v_j$.

- Any node that neighbours $N$ via arc $A$ must contain:
  - The prerequisites of action $A$
  - All of the elements of $N$ that were not achieved by $A$

  $N$ must be consistent.
If we’re currently at a node \([X_1 = v_1, \ldots, X_n = v_n]\) then an arc labeled \(A\) exists to another node \(N\) if

- There exists some \(i\) for which \(X_i = v_i\) is on the effects list of action \(A\)
- For all \(j\), \(X_j = v'_j\) is not on the effects list for \(A\), where \(v'_j \neq v_j\)
- \(N\) is \(\text{preconditions}(A) \cup \{X_k = v_k : X_k = v_k \notin \text{effects}(A)\}\) and \(N\) is consistent in that it does not assign multiple values to any one variable.
Regression example

**Actions**
- mc: move clockwise
- mac: move anticlockwise
- nm: no move
- puc: pick up coffee
- dc: deliver coffee
- pum: pick up mail
- dm: deliver mail

**Locations:**
- cs: coffee shop
- off: office
- lab: laboratory
- mr: mail room

**Feature values**
- rhc: robot has coffee
- swc: Sam wants coffee
- mw: mail waiting
- rhm: robot has mail

**Planning: Regression Planning**
Find the errors (none involve room locations)

Locations:
cs: coffee shop
off: office
lab: laboratory
mr: mail room

Feature values
rhc: robot has coffee
swc: Sam wants coffee
mw: mail waiting
rhm: robot has mail

Actions
mc: move clockwise
mac: move anticlockwise
nm: no move
puc: pick up coffee
dc: deliver coffee
pum: pick up mail
dm: deliver mail
Loop detection and multiple-path pruning

- Goal $G_1$ is simpler than goal $G_2$ if $G_1$ is a subset of $G_2$.
  - It is easier to solve [$cs$] than [$cs, rhc$].

- Loop detection: if during the search we encounter a node $N$, but one of its ancestors $N'$ is the same or simpler, you can prune $N$.

- Multiple path pruning: if during the search we encounter a node $N$, but elsewhere in the search tree (not as a descendent of $N$) we have encountered a node $N'$ which is the same or simpler, you can prune $N$. 
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.

- E.g., it may not be obvious from the action description that the agent can only hold one item at any time.
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)