Planning

CPS 570
Ron Parr

Some Actual Planning Applications

• Used to fulfill mission objectives in Nasa’s Deep Space One (Remote Agent)
  – Particularly important for space operations due to latency
• Also used for Rovers
• Aircraft assembly schedules
• Logistics for the U.S. Navy
• Observation schedules for Hubble space telescope
• Scheduling of operations in an Australian beer factory

Scheduling

• Many “planning” problems are scheduling problems
• Scheduling can be viewed as a generalization of the planning problem to include resource constraints
  – Time & Space
  – Money & Energy
• Many principles from regular planning generalize, but some extensions (not discussed here) are used

Characterizing Planning Problems

• Start state (group of states)
• Goal – almost always a group of states
• Actions
• Objective: Plan = A sequence of actions that is guaranteed to achieve the goal.
• Like everything else, can view planning as search...
• So, how is this different from generic search?
What makes planning special?
- States typically specified by a set of relations or propositions:
  - On(solar_panels, cargo_floor)
  - arm_broken
- Goal is almost always a set
  - Typically care about a small number of things:
    - at(Ron, airport),
    - parked_in(X, car_of(Ron))
    - airport_parking_stall(X)
  - Many things are irrelevant
    - parked_in(Y, car_of(Bill))
    - adjacent(X,Y)
- Branching factor is large

Planning Algorithms
- Extremely active and rapidly changing area
- Regular competitions pit different algorithms against each other on suites of challenge problems
- Algorithms compete in different categories
  - General vs. Domain specific
  - Optimal vs. Satisficing
- No clearly superior method has emerged, though there are trends – satplan is one such trend

Planning With Logic/Theorem Proving
- Need to describe effects of actions with logic
- Ask for the existence of plans that achieve our goals
- Challenge: Talking about dynamic situations in logic

Why planning w/logic is awkward
- Logic doesn’t tolerate inconsistencies
- Need to talk about not just what is true, but when it is true – add extra argument to every predicate
- Adding an extra argument to every predicate has issues:
  - Complicates representation
  - Need to say what changes in every situation, as well as what doesn’t change – frame problem
- Logic is very general:
  - Theorem proving tends to be slow/inefficient
  - Generality always comes at price of efficiency
Overcoming Limitations of Planning via Theorem Proving

• Simplify the representation

• Avoid generality of full, first order logic in hopes of allowing faster, special purpose algorithms for planning

PDDL

• Actions have a set of preconditions and effects

  •Think of the world as a database
    – Preconditions specify what must be true in the database for the action to be applied
    – Effects specify which things will be changed in the database if the action is taken

• NB: PDDL supersedes an earlier, similar representation called STRIPS

move(obj,from,to)

• Preconditions
  – clear(obj)
  – on(obj,from)
  – clear(to)

• Effects
  – Add
    • on(obj,to)
    • clear(from)
  – Delete
    • on(obj,from)
    • clear(to)

move(y,x,z)

Limitations of PDDL

• Assumes that a small number of things change with each action
  – Dominoes 🧳
  – Pulling out the bottom block from a stack 🧳

• Preconditions and effects are conjunctions

• No quantification

• Closed world assumption (negation in implemented as deletion, no negated preconditions)
How hard is planning?

- Planning is NP hard
- How can we prove this?
  - Use Planning to solve 3SAT
  - Any 3SAT instance can be converted to a planning problem in polynomial time
  - Polynomial time planning algorithm would imply polynomial time solution to 3SAT

Is planning NP-complete?

- NO!
- Consider the towers of Hanoi:
  - PDDL actions are the block moving actions
- Requires exponential number of moves
- Planning is actually PSPACE complete
- Planning with bounded plans is NP-complete

Should plan size worry us?

- What if you have a problem with an exponential length solution?
- Impractical to execute (or even write down) the solution, so maybe we shouldn’t worry
- Sometimes this may just be an artifact of our action representation
  - Towers of Hanoi solution can be expressed as a simple recursive program
  - Nice if planner could find such programs

Planning Using Search

- Forward Search:
  - As with theorem proving, blind forward search is problematic because of the huge branching factor
  - Some success using this method with carefully chosen action pruning heuristics (not covered in class)
- Backward Search:
  - As with theorem proving, tends to focus search on relevant terms
  - Called “Goal Regression” in the planning context
Goal Regression

- Goal regression is a form of backward search from goals
- Basic principle goes back to Aristotle
- Embodied in earliest AI systems
  - GPS: General Problem Solver by Newell & Simon
- Cognitively plausible
- Idea:
  - Pick actions that achieve (some of) your goal
  - Make preconditions of these actions your new goal
  - Repeat until the goal set is satisfied by start state

Greed, decomposition in planning

- Does a greedy approach work for planning?
- Idea:
  - Pick actions that satisfy as many parts of the goal as possible
  - If no single action satisfies any part of the goal, break up the goal into pieces and plan to solve each of them individually
- Bad news: This works poorly in general

Goal Regression Example

Regress on(x,z) through move(z,table,x)
New goal: clear(x)

The Sussman Anomaly

Goal: on(x,y), on(y,z)
Problems with naïve subgoaling

- The number of conjuncts satisfied may not be a good heuristic
- Achieving individual conjuncts in isolation may actually make things harder
- Causes simple planners to go into loops and/or take lots of wasted steps

Summary of Traditional Planners

- Backward search methods were more focused, with careful construction these could be sound and complete generic planners
- Forward search methods worked well when:
  - Search space was very narrow (only a small number of reasonable things to do, which prevented exponential growth in reachable search space)
  - Domain-specific knowledge could be used to narrow the search space

Modern Planners

- One family uses sophisticated heuristics (graphplan)
  - Uses various tricks to narrow search space
  - May use forward or backward planning
- Another family uses forward chaining with domain specific tricks to prune the search space
- Yet another family converts everything into a giant SAT problem and runs a highly optimized SAT solve (SATPlan)

What’s Missing?

- As described, plans are “open loop”
- No provisions for:
  - Actions failing
  - Uncertainty about initial state
  - Observations
- Solutions:
  - Plan monitoring, replanning
  - Conformant/Sensorless planning
  - Contingency planning
Planning Under Uncertainty

• What if there is a probability distribution over possible outcomes?
  – Called: Planning under uncertainty, decision theoretic planning, Markov Decision Processes (MDPs)
  – Much more robust: Solution is a “universal plan”, i.e., a plan for all possible outcomes (monitoring and replanning are implicit)
  – Much more difficult computationally

• What if observations are unreliable?
  – Called: “Partial Observability”, Partially Observable MDPs (POMDPs)
  – Applications to medical diagnosis, defense, sensor planning
  – Way, way harder computationally