Planning

CPS 270
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 \textit{guaranteed} to achieve the goal.

- Like everything else, can view planning as search...
- \textit{So, how is this different from generic search?}
What makes planning special?

- States typically specified by a set of grounded relations or propositions (negations not allowed):
  - 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

How Do Search, Planning & CSPs fit together?

- All very general frameworks
- Search is the most general:
  - Start
  - Goal
  - Actions
  - Costs
- We can formulate anything as search, even in a not entirely unnatural way:
  - Shortest path
  - Sorting
  - Planning
  - CSPs
- The fact that you are holding a hammer doesn’t make everything a nail.

Algorithm vs. Concept

- There are times when we will talk about search as a specific algorithm, i.e., something maintains a queue, pops things off the queue, expands them, etc.
- Other times we will talk about search as a more abstract concept, e.g., finding a minimum of a function by gradient descent can be thought of as a kind of search, even though we don’t maintain a queue

Planning Algorithms

- Extremely active and rapidly changing area
- Annual 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
PDDL – A Language for Planning

- 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\( (\text{obj}, \text{from}, \text{to}) \)

- Preconditions
  - clear\( (\text{obj}) \)
  - on\( (\text{obj}, \text{from}) \)
  - clear\( (\text{to}) \)

- Effects
  - Add
    - on\( (\text{obj}, \text{to}) \)
    - clear\( (\text{from}) \)
  - Delete
    - on\( (\text{obj}, \text{from}) \)
    - clear\( (\text{to}) \)

Nota bene: STRIPS broke down effects into add and delete lists, while PDDL will lump them together and express deletions with negation. Note that negations are NOT entered into the database.

Comparison With Search

- Consider a table with \( n \) objects
- One planning action: move\( (x, y, z) \)

- How many realizations of this action?
- What is the branching factor if we approach this as a generic search problem?

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 effects only implemented as deletion)
How hard is planning?

- Planning is NP hard

- Decision problems:
  - Does there exist a plan to achieve the goal?
  - Does there exist a plan to achieve the goal in k or fewer steps?

- Optimization: What is the shortest plan to achieve the goal? (Also hard, though not a decision problem as posed.)

Is planning NP-complete?

- NO!

- Consider the towers of Hanoi:
  - PDDL actions are the block moving actions

- Requires exponential number of moves

- Planning existence is actually PSPACE complete

- Bounded plan existence 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

Dealing With Intractability

- So, you’re problem is NP-hard/complete

- Do you give up?

- Remember: These are worst case results

- Not all domains are worst case

- Blocks world:
  - Plan existence: Trivially yes (if blocks exist)
  - Plan of size 2n or less: Trivially yes (if plan exists)
  - Short(est) plan: NP-hard
Planning Using Search

- **Forward Search:**
  - 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:**
  - Called “Goal Regression” in the planning context
  - Must be done in a clever way because goal is usually a state set (no single goal state)

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

Goal Regression Example

Regress on(x,z)
through move(z,table,x)

New goal:
clear(x)

Goal: on(x,z)

Goal Regression Properties

- Can ignore irrelevant details
  (e.g. blocks not appearing in or connected to blocks in the goal configuration)
- Viewed as a search through state sets
- Can still blow up – many possible choices for actions to regress through
Heuristics in planning

- How close are we to the goal?

- Idea:
  - If goal is a conjunction, does satisfying parts bring us closer to achieving the entire goal?
  - Example: Can we think about individual pairs of blocks in a stack, or do we need to reason about the entire stack?

- Bad news: Planning doesn’t decompose easily

---

The Sussman Anomaly

![Diagram of blocks with 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 are more focused; with careful construction these can 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 domain neutral tricks to narrow search space
  – Tracks subgoal interactions to avoid doing silly things with problems like the Sussman anomaly
  – May use forward or backward planning
• Another family uses forward search 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