Real World Planning and Acting

CS4246/CS5446

AI Planning and Decision Making

This lecture will be recorded!

Please join:
pollev.com/anarayan

Admin: Final Test

- Week 11 Lecture
- Date: April 3, 2024
- Time: 6:30 pm
- Duration: 90 min
- Venue: TBD
- Mode: Sit-in; using Examplify

Topics

- Heuristics for planning (RN 11.3)
  - Problem relaxation and admissible heuristics
  - Domain independent planning (RN 11.3.1)
  - State abstraction in planning (RN 11.3.2)
- Hierarchical planning (RN 11.4)
  - Hierarchical task networks and HTN Planning
  - High-level actions (RN 11.4.1)
  - Searching for primitive solutions (RN 11.4.2)
  - Searching for abstract solutions (RN 11.4.3)

Recap

- Characteristics of problems classical planning can solve
  - Discrete
  - Deterministic
  - Static
  - Fully observable
- Representing a planning problem – PDDL
  - Power to deal with set operations
- Planning as a Search & Satisfiability problem

Quiz

Quiz answer

Quiz answer

Exercise

- Question:
  - If the goal is $P \wedge Q \wedge R$
  - If Action a_1 has effect $P \wedge Q \wedge \neg R$, is a_1 a relevant action?

Solving Planning Problems

Heuristics: for classical planning
HTN: New planning model

- Planning Problem or Model
  - Appropriate abstraction of states, actions, effects, and goals (and costs and values)
- Planning Algorithm
  - Input: a problem
  - Output: a solution in the form of an action sequence
- Planning Solution
  - A plan or path from the initial state(s) to the goal state(s)
    - Any path; OR
    - An optimal path wrt to costs or values
  - A goal state that satisfies certain properties

Heuristics for Planning

Improving Efficiency

Heuristics for Planning

- Heuristic function
  - $h(s)$ estimates distance from a state s to the goal g
- Main idea
  - Find admissible heuristic for distance
  - A* or other heuristic search can then find optimal solutions

Quiz

Quiz answer

Quiz answer

Heuristics for Planning

- Heuristic function
  - $h(s)$ estimates distance from a state s to the goal g
- Main idea
  - Find admissible heuristic for distance
  - A* or other heuristic search can then find optimal solutions
- Approach
  - Define a relaxed problem that is easier to solve
  - Exact cost of solution to easier problem is heuristic for original problem

Types of Heuristics

Real-world problems use a combination of domain-independent and domain-specific heuristics

- Definition and application
  - Facilitated by factored representation for states and action schemas
- Search problem
  - A graph with states as nodes and actions as edges
  - Find a path connecting initial state to goal state
- Domain independent heuristics –Why?
  - Two ways to relax the problem (make it strictly easier):
    - Add more **edges** – easier to find path
    - Group multiple **nodes** together – abstract with fewer states and easier to search
  - Prune away irrelevant branches of search tree

Heuristics of Adding Edges

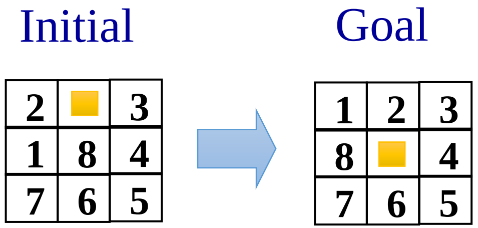
- Ignore preconditions heuristic
  - Drop all preconditions from actions
- Ignore selected preconditions heuristic
  - Derive simpler measures of distance from goal
- Ignore delete list heuristic
  - Allow monotonic progress toward goal
- Caveats:
  - Finding solution to relaxed problem is still NP-hard
  - Trade-off in optimality or admissibility sometimes required
  - Expensive to calculate heuristics
  - Do not reduce state-space size

How to Define an Edge-Based Heuristic?

- Approach:
  - Relax actions by removing all preconditions and all effects except goal literals
  - Count minimum no. of actions required for union effects to satisfy the goal
- Note:
  - An instance of the set cover problem – NP-hard
- Potential solution:
  - Simple greedy algorithm guaranteed to return a set covering whose size is within a factor of $\log(n)$ of the true minimum covering
    - where n is the number of literals in the goal
  - Loses guarantee of admissibility

Example: 8-Puzzle as Planning

- Planning as path search



- Game objective:
  - To rearrange a given initial configuration (state) of eight numbered tiles arranged on a 3×3 board into given final or goal configuration (state)

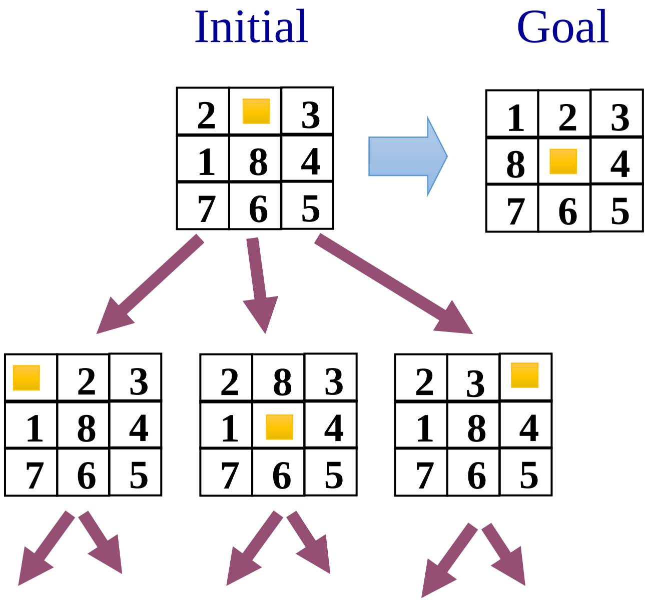
$Action(Slide(t, s_1, s_2)$

Precond: $On(t, s_1) \wedge Tile(t) \wedge Blank(s_2) \wedge Adjacent(s_1, s_2)$

Effect: $On(t, s_2) \wedge Blank(s_1) \wedge \neg On(t, s_1) \wedge \neg Blank(s_2)$)

Example: 8-Puzzle as Planning

- States
  - A state description specifies the location of each of the eight tiles in one of the nine squares
- Actions or operators
  - Tile slides left, right, up, or down
- Goal test
  - State matches the goal configuration
- Path cost
  - Each step cost 1, so path cost is just the length of the path



$Action(Slide(t, s_1, s_2)$

Precond: $On(t, s_1) \wedge Tile(t) \wedge Blank(s_2) \wedge Adjacent(s_1, s_2)$

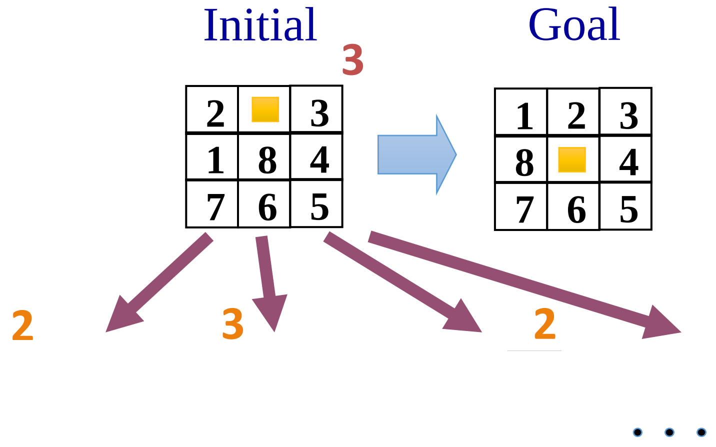
Effect: $On(t, s_2) \wedge Blank(s_1) \wedge \neg On(t, s_1) \wedge \neg Blank(s_2)\ )$

Ignore Preconditions Heuristic

- Main idea:
  - All actions become applicable in all states
  - Any single goal fluent can be achieved in one step (if there is an applicable action)
  - No. of steps required to solve relaxed problem ≈ no. of unsatisfied goals
    1. Some actions may achieve multiple goals
    2. Some actions may undo the effects of others
  - Possible accurate heuristic – consider 1 and ignore 2

Ignore Selected Preconditions Heuristic

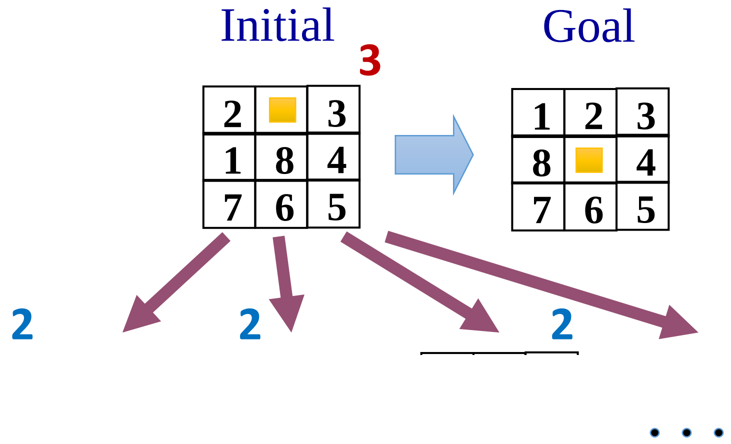
- Remove Preconditions
  - $Blank(s_2)\ \wedge\ Adjacent(s_1, s_2)$
- What does the heuristic do?
  - No. of misplaced tiles heuristic
- What are the estimates?



$Action(Slide(t, s_1, s_2)$
Precond: $On(t, s_1) \wedge\ Tile(t) \wedge Blank(s_2) \wedge Adjacent(s_1, s_2)$
Effect: $On(t, s_2) \wedge Blank(s_1) \wedge \neg On(t, s_1) \wedge \neg Blank(s_2))$

Ignore Selected Preconditions Heuristic

- Remove Preconditions
  - $Blank(s_2)$
- What does the heuristic do?
  - Manhattan-distance heuristic
- What are the estimates?



- Caveat:
  - Unclear which preconditions can be selectively ignored in general

$Action(Slide(t, s_1, s_2)$

Precond: $On(t, s_1) \wedge Tile(t) \wedge Blank(s_2) \wedge Adjacent(s_1, s_2)$

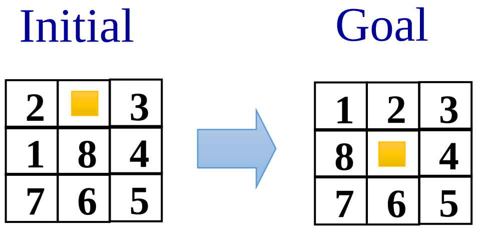
Effect: $On(t, s_2) \wedge Blank(s_1) \wedge \neg On(t, s_1) \wedge \neg Blank(s_2))$

Ignore Delete List Heuristic

- Main idea:
  - Assume only positive literals in all plans and actions
  - Remove delete list from all actions (all negative literals from effects) to make monotonic progression toward goal
  - Create a relaxed version of original problem that is easier to solve, where solution length will serve as good heuristics
  - Approximate solution can be found in polynomial time by hill-climbing

Ignore Delete Lists Heuristic

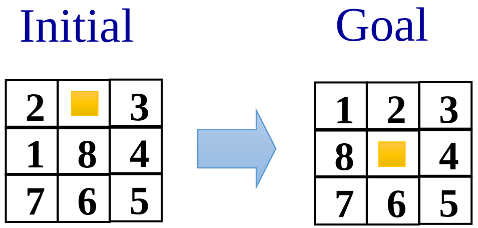
- Ignore delete lists
  - How does the problem become easier when ignore delete lists heuristic is applied to this schema?
  - Goals are never undone
  - Blanks always increase



$$\begin{aligned}&Action(Slide(t, s_1, s_2)\\&\qquad \text{Precond: } On(t, s_1) \land Tile(t) \land Blank(s_2) \land Adjacent(s_1, s_2)\\&\qquad \text{Effect: } On(t, s_2) \land Blank(s_1) \land \boxed{\neg On(t, s_1) \land \neg Blank(s_2))}\end{aligned}$$

Can We Do Better?

- What would YOU do?
  - Can planning system emulate human moves?
  - Would that help?



$$
\begin{aligned}
&Action(Slide(t, s_1, s_2)\\
&\qquad \text{Precond: } On(t, s_1) \wedge Tile(t) \wedge Blank(s_2) \wedge Adjacent(s_1, s_2)\\
&\qquad \text{Effect: } On(t, s_2) \wedge Blank(s_1) \wedge \neg On(t, s_1) \wedge \neg Blank(s_2)
\end{aligned}
$$

Quiz

Quiz answer

Quiz answer

Domain-independent Pruning

- Symmetry reduction
  - Prune all symmetric branches of the search tree except for one
  - For many domains, efficiently solve intractable problems
- Forward pruning
  - Accept risk of pruning away an optimal solution, in order to focus search on promising branches
- Rule-out negative interactions
  - A problem has serializable subgoals if there exists an order of subgoals such that the planner can achieve them in that order without having to undo any of the previously achieved subgoals

How do you find preferred action(s)?

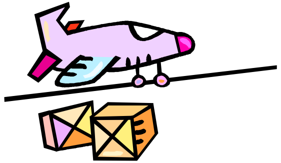
Serializable Subgoals

- Planning examples in the real world:
  - Build a tower of blocks on Table, in any order
  - Switch on all n lights with independent switches, in any order
  - Remote Agent Planner that commands NASA’s Deep Space One spacecraft (1998) – serializable by design
    - able to command spacecraft in real-time

Heuristics of State Abstraction

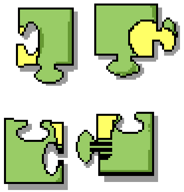
- State abstraction:
  - Many-to-one mapping from states in the ground representation of the problem to the abstract representation
- Main approach:
  - Ignore some fluents
  - Solution in abstract state space will be shorter than a solution in the original space (assumes admissible heuristic)
  - Abstract solution extensible to solution for original problem

Example: Air Cargo Transportation



- Original problem:
  - 10 airports, 50 planes and 200 cargos
  - Total no. of states =
- Assumption:
  - All cargos are just in 5 airports, instead of 10, and all cargos in the same airport have the same destination.
- Reformulation:
  - Drop all the irrelevant At fluents (What are the relevant ones?)
  - Total no. of states =
- Solution:
  - Shorter than that for original problem (admissible heuristic)
  - Extensible by adding relevant actions

Decomposition



- Main idea:
  - Divide problem into parts, solving each part independently, and then combining the parts - Key idea in defining heuristics
- How to choose the right abstraction to reduce total cost?
  - Defining abstraction, doing abstract search, mapping abstraction back to original problem
  - Can the cost be less than original planning cost?
  - Example:
    - Pattern databases – cost of creation amortized over many problem instances

Planning Cost with Abstraction

- Problem definition:
  - Suppose the goal is a set of fluents G, divided into disjoint subsets $G_1, \dots, G_n$.
  - Find plans $P_1, \dots, P_n$ that solve the respective subgoals
  - What is the estimated cost of the plan for achieving all of G?
- Heuristic estimation:
  - Think of each $Cost(P_{\mathrm{i}})$ as a heuristic estimate
  - If each subproblem uses an admissible heuristic, taking Max is admissible
- Assuming subgoal independence:
  - Sum the cost of solving each independent subgoal; if admissible, Sum is better than Max – Why?
  - Solution optimistic when there are negative interactions between subplans for each subgoal; e.g.,action in one subplan deletes goals in another subproblem.
  - Solution pessimistic (not admissible) when there are positive interactions; e.g., actions in one subplan achieves goals in another subproblem

Example: FF- FastForward

- Characteristics:
  - Forward state-space searcher making use of effective heuristics
  - Ignore-delete-list heuristic with graph plan for heuristic estimation
  - Hill-climbing search (modified to keep track of plan) with heuristic to find a solution
    - Non-standard hill-climbing algorithm: avoids local maxima by running a breadth-first search from the current state until a better one is found
    - If this fails, FF switches to greedy best-first search instead
  - [Hoffmann, 2001]

Classical Planning Today

Hint! Hint! Think about the project ☺

- Classical Planning research: Examples:
  - Families of systems in use – Heuristic Search Planner (HSP), Fast Forward (FF), Fast Downward
  - https://planning.wiki/ref/planners/
- International Planning Competitions
  - https://www.icaps-conference.org/competitions/
  - FastForward (FF) [Hoffmann, 2001] was the winner in the 2002 International Planning Competition (IPC)
  - SATPlan was the winner in the 2004 and 2006 IPC
  - IPC 2014 was won by SymBA*, based on bidirectional search using heuristics and abstraction.
  - IPC 2018 was won by Delfi, using deep learning to select a planner from a collection of planners including Fast Downward (uses forward search) with various heuristics and SymBA*
  - IPC 2020 – Hierarchical planning!
- Classical Planning in practice: Examples
  - Commercial video games [Neufeld, X., et al., 2019]
  - AI planning for enterprise [Sohrabi,S, 2019]
  - Space exploration [Estlin, T., et al. 07] [Rabideau, G., et al., 2020]

Summary

- Classical or deterministic planning
    - No consensus on the best approach; competition and cross-fertilization induce progress
- Using domain-independent heuristics
    - Transform planning problems into relaxed problem spaces
    - Effective heuristics derived with subgoal independence assumptions by:
        - relaxation of planning problem
        - pruning repeated or irrelevant branches
    - Solve with efficient algorithms

Homework

- Readings:
  - Heuristics: RN: 11.3
  - Hierarchical: RN: 11.4
- Reviews:
  - RN: 3.3, 3.5, 3.6 (Review of search and heuristics in problem-solving)

References

- Content:

  - Nau, Dana. Lecture Slides from Automate Planning: Licensed under the Creative Commons Attribution-NonCommercial-ShareAlike License: http://creativecommons.org/licenses/by-nc-sa/2.0/

- Classical planning systems: (Journal articles publicly available online or through NUS Library e-Resources)

  - Hoffmann, J. and B. Nebel, *The FF planning system: fast plan generation through heuristic search.* J. Artif. Int. Res., 2001. **14**(1): p. 253–302.
  - Helmert, M., *The fast downward planning system.* J. Artif. Int. Res., 2006. **26**(1): p. 191–246.
  - Georgievski, I. and M. Aiello, *HTN planning: Overview, comparison, and beyond.* Artificial Intelligence, 2015. **222**: p. 124-156.

- Classical planning in the Real World:

  - Neufeld, X., et al., Building a Planner: A Survey of Planning Systems Used in Commercial Video Games. IEEE Transactions on Games, 2019. 11(2): p. 91-108. https://ieeexplore.ieee.org/document/8214211
  - Sohrabi, S., AI Planning for Enterprise: Putting Theory Into Practice, in Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19. 2019, International Joint Conferences on Artificial Intelligence Organization. p. 6408--6410. https://www.ijcai.org/proceedings/2019/0897.pdf
  - Estlin, T., et al. *Increased Mars Rover Autonomy using AI Planning, Scheduling and Execution*. in *Proceedings 2007 IEEE International Conference on Robotics and Automation*. 2007. https://ai.jpl.nasa.gov/public/documents/papers/estlin_icra07_marsrover.pdf
  - Rabideau, G.; Wong, V.; Gaines, D.; Agrawal, J.; Chien, S.; Kuhn, S.; Fosse, E.; and Biehl, J. Onboard Automated Scheduling for the Mars 2020 Rover. In *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation for Space*, of *i-SAIRAS'2020*, Noordwijk, NL, 2020. European Space Agency https://ai.jpl.nasa.gov/public/documents/papers/M2020_OBP_i-SAIRAS2020_camera.pdf
