CS 4649/7649
Robot Intelligence: Planning

Efficient Planning, PDDL

Sungmoon Joo

School of Interactive Computing
College of Computing
Georgia Institute of Technology

Course Info.

• Course Website: joosm.github.io/RIP2014
• Course Wiki: github.com/RIP2014/RIP2014Wiki/wiki
  - add your contact info, start grouping/filling in project ideas, etc.
• HW#1, due Sept. 29
Efficiency in Planning

- Planning Efficiency: **speed** of a planner

  Smart robots make **good** decisions.
  Smarter robots make **good** decisions **fast**!
  We like smart robots

  **How can we make our robots smarter?**

Properties of Heuristics: $h(s)$

**Informed**
- Does estimate lead to the goal?
- Accuracy of heuristic

**Admissible**
- $h(s) \leq$ true "cost to go"
- Is Best-First Search with Admissible $h(s)$ optimal?
- Is A* with Admissible $h(s)$ optimal?
Heuristics in Planning

- Domain Dependent
  1. B A C
  2. C B A

- Domain Independent
  - Automatically Analyze Problem/Domain
  - Derive Heuristic Estimate

S. Joo (sungmoon.joo@cc.gatech.edu)  9/9/2014

Heuristics in Planning

- Domain Dependent
  1. B A C
  2. C B A

- Domain Independent
  - Automatically Analyze Problem/Domain
  - Derive Heuristic Estimate

S. Joo (sungmoon.joo@cc.gatech.edu)  9/9/2014
Domain Independent Heuristics

Concept
- Solve a relaxed form of the problem
- Use to evaluate states for solving original problem

Approaches
- Assume complete subgoal independence (remember this idea?)
- Assume no negative interactions (new idea)
- Assume limited negative interactions

HSP (Bonet & Geffner 1997)

- Progression or Regression Search
  (We focus on Progression)

- Relax Problem by Eliminating Delete Lists

Expand state in levels by evaluating all valid actions
Continue until no new literals are added

Heuristic cost of literal is number of levels until it first appears

Not quite GraphPlan: level is used as a HEURISTIC – how?
Computing Costs of Literals

HSP Progression Heuristic

- Grow a state space graph from $s$ ignoring negative action effects.
- Heuristic cost of precondition
  $C(p, s) = \text{level of } p \text{ in graph}$
- $HSP\_h(s) = \text{function of } C(p, s)$
HSP Heuristics

$H_0$
- Cost of action is sum of precondition costs
- Informed, but not admissible

$H_1$
- Cost of Action is maximum over costs of preconditions
- Admissible, but not very informed

$H_2$
- Solve for pairs of literals
- Take maximum cost over all pairs
- Informed, and claimed to be admissible

H$_0$

- Assume negative effects do not exist
- The cost of achieving a set of preconditions $\{p_0, ..., p_n\}$
  $= \text{Sum of costs for achieving each precondition}$

$$C(s) = \sum_i C_i$$

Informed? Somewhat
Admissible? No

HSP$_{h(s)}$
- Grow a state space graph from $s$
  ignoring negative action effects.
- Heuristic cost of precondition
  $C(p,s) = \text{level of p in graph}$
- $HSP_{h(s)} = \text{function of C(p,s)}$
**H₁**

- Assume negative effects do not exist
- The cost of achieving a set of preconditions \( \{p₁, \ldots, p_n\} \)
  = Maximum of costs for achieving each precondition

\[
C(s) = \max_i C_i
\]

Informed? Not very
Admissible? Yes

**H₂**

- Assume negative effects do not exist
- The cost of achieving a set of preconditions \( \{p₁, \ldots, p_n\} \)
  = Maximum of costs for achieving PAIRS of preconditions

\[
C(s) = \max_{ij} C_{ij}
\]

Informed? More so
Admissible? Yes
**FF (Hoffman 2000)**

- Hoffman – Best of both worlds: HSP & GraphPlan
- Refines HSP Heuristics

\[
FF_h(s)
\]

- Grow a state space graph from \( s \) ignoring negative action effects.
- Compute a relaxed plan by regression search in the graph.
- \( FF_h(s) = \# \) actions in plan

---

**FF Heuristic**

Suppose our goal is to achieve B and E

1. **Valid Actions w/ precondition A**
   - Add: A
   - \( \text{Add: B} \)
   - \( \text{Add: C} \)

2. **Valid Actions w/ preconditions A,B,C**
   - \( \text{Add: D} \)
   - Add: E

\[
F_{FF}(s)
\]
Suppose our goal is to achieve B and E

If goal exists in level i
Identify it in i-1

---

Otherwise, add action that achieves it to plan
Add its preconditions to Goals at i-1
**FF Heuristic**

- Grow a state space graph from ignoring negative action effects.
- Compute a relaxed plan by regression search in the graph.
- $FF_{H(s)} = \#$ steps in plan

Suppose our goal is to achieve B and E

```
Goals at i
A
B
C
D
E

Add: B

Add: C

PC: C Add: E
```

Otherwise, add action that achieves it to plan
Add its preconditions to Goals at i-1

---

**FF vs. HSP**

- FF's heuristic is more informed and admissible
- Takes into account positive interactions by propagating constraints back through the layers
- FF uses "Enforced Hill Climbing" which appears to work well
  $EHC = \text{Hill Climbing} + \text{BFS when stuck}$

Really though – which one is faster?

**International Planning Competitions at ICAPS**
**FF vs. HSP**

- FF's heuristic is more informed
- Takes into account positive interactions by propagating constraints back through the layers
- FF uses "Enforced Hill Climbing" which appears to work well
  
  \[ \text{EHC} = \text{Hill Climbing} + \text{BFS when stuck} \]

  **The Hoffman Story:**

  2000 FF = International Competition Winner

  2002 FF is Also very Successful

  2004 Hoffman becomes International Competition Chair

---

**Plan Space Planning?**

- Catching up! Various heuristics have been proposed.
- "Flaw Selection Strategies” (Which subgoal(or threat) should be worked on?)
  - Continue working on a single subproblem
  - Choose easiest subproblems to work on
  - In particular solve ones with forced solutions (reduces branching factor)
  - Put off threats that may be forced by future constraints

- VHPOP (Younes & Simmons)
  - Similar to FFs heuristic for partial order planning (additive)
Propositional Satisfiability: Since 1960s

SAT = First PROBLEM shown NP-Complete! 1971 Stephen Cook

Given a boolean formula:

\[(P \lor Q) \land (\neg R \lor S) \land (R \lor Q \lor S)\]

SAT: Exists a model (truth assignment) that makes this true?

Many algorithms for solving it
Especially for achieving average-case polynomial time!
Planning As Satisfiability

- Let $P$ be a planning problem
  - Let $(P,k)$ be a *bounded planning problem*
  - A solution for $P$ of length $k$ is a solution to $(P,k)$

- For each $k = 0, 1, \ldots, N$ (some bound)
  - Encode $(P,k)$ as a SAT problem
  - If resulting SAT problem is satisfiable, extract solution plan

Encoding Plans as SAT

Use Only propositional logic

- Turn predicates $\text{on}(A,\text{Table})$ into propositions $\text{on-}A\text{-\emph{Table}}$
- Modify action descriptions accordingly
## Encoding Plans as SAT

- **Initial State:** \(\text{on-A-Table-0} \land \text{on-C-A-0} \land \text{on-B-Table-0} \ldots\)
- **Goal State:** \(\text{on-A-B-k} \land \text{on-B-C-k} \land \text{on-C-Table-k}\)
- **For every action and for every step** \(i (0 < i < k)\):
  \[\text{Action-}i \Rightarrow Pr_{1-i} \land Pr_{2-i} \land \ldots \land Erf_{1-(i+1)} \land Erf_{2-(i+1)}\]
- **Complete Exclusion Axioms** (for every action and every step \(i\))
  \[\neg \text{ActionA-}i \lor \neg \text{ActionB-}i\]
  only one action occurs at each time point
- **Frame Axioms!** (for example: for \(A, B\ldots\) that have effect on \(P\))
  describe the predicate that are not affected by the actions
  
  ...
Solving SAT

- Davis-Putman
  - Transform to CNF (Conjunctive Normal Form)
  - Backtrack Search for Assignments to Literals

- Hillclimbing Local Search!

- Walksat
  - Select random truth assignment
  - With probability $p$ flip a random variable
  - With probability $(1-p)$ flip the variable that minimized unsatisfied clauses

The Last Classical Story

- 1992 Kautz and Selman show first competitive SAT planner
  Still takes too much memory and time compared to modern planners

- 1995-1997 Blum and Furst present GraphPlan
  GraphPlan propagates plan constraints

- 1998 **Blackbox**
  Combines Mutex Constraints of GraphPlan with SAT Solvers
  One of the best planners in 1998 Planning Competition!
The Last Classical Story

- 1992 Kautz and Selman show first competitive SAT planner
  
  Still takes too much memory and time compared to modern planners

- 1995-1997 Blum and Furst present GraphPlan
  
  GraphPlan propagates plan constraints

- 1998 Blackbox
  Combines Mutex Constraints of GraphPlan with SAT Solvers
  One of the best planners in 1998 Planning Competition!

- 2000-2002 FF (Hoffman)

- 2004-2006 SatPlan extends Blackbox and takes 1st place!

Who Won in 2008? FF/HSP Strike Back!

![Sequential satisficing track](image)

- Winner: LAMA
- Runner-up: FF(ha)
- Jury Award: C³

![Sequential optimization track](image)

- Winner: Gamer
- Runner-up: HSPf

S. Joo (sungmoon.joo@cc.gatech.edu)  9/9/2014
Who Won in 2011?

- Optimization Track: Fast Downward Stone Soup-1
- Sequential Track: Winner - LAMA 2011
  Runner up - Stone Soup

55 planners in total from 55 different people from 11 countries: Australia, Canada, China, France, Germany, India, Israel, Italy, Spain, UK and USA

Who Won in 2014?

- Sequential Optimal track: Winner - SymBA*-2, SymBA*-1
  Runner up – cGamer
- Sequential Multi-core track: Winner - ArvandHerd
  Runner up – IBaCoP
- Sequential Satisficing track: Winner – IBaCoP2
  Runner up - Mercury
- Agile track: Winner – YAHSP3
  Runner up – Madagascar-pC
- Temporal track: Winner – YAHSP3-MT
  Runner up – Temporal Fast Downward

67 planners in total from 66 people from 14 countries: Australia, Canada, Czech Republic, Finland, France, Germany, Iran, Israel, New Zealand, Spain, Switzerland, UK, Venezuela, and USA

*China(X), India(X), Italy(X)
### PDDL

**PDDL = Planning Domain Definition Language**

- Components of a PDDL planning task:
  - Objects: Things in the world
  - Predicates: Object properties
  - Initial state: The state of the world that we start in
  - Goal specification: Things that we want to be true
  - Actions/Operators: Ways of changing the state of the world
How to use PDDL for planning problems?

- Two files
- A domain file: predicates and actions
- A problem file: objects, initial state and goal specification

(define (domain hanoi-domain)
  (:requirements :equality)
  (:action move-disk
    :parameters (?disk ?below ?new ?below)
    :precondition (and (disk ?disk)
                      (smaller ?disk ?new ?below)
                      (not (= ?new ?below ?disk))
                      (not (= ?new ?below ?disk))
                      (on ?disk ?below ?disk))
    :effect (and (clear ?below ?disk)
                (on ?disk ?new ?below)
                (not (on ?disk ?below ?disk))
                (not (clear ?new ?below ?disk))))

(define (problem hanoi-problem)
  (:domain hanoi-domain)
  (:objects p1 p2 p3 d1 d2 d3)
  (:init (smaller d1 p1) (smaller d2 p1) (smaller d3 p1)
          (smaller d1 p2) (smaller d2 p2) (smaller d3 p2)
          (smaller d1 p3) (smaller d2 p3) (smaller d3 p3) (smaller d1 d2)
          (clear d1) (clear d2) (clear d3)
  (:goal (and (on d1 d2) (on d2 d3) (on d3 p1))))
PDDL

- Domain description

Requirements

(define (domain hanoi-domain)
  (:requirements :equality)
  (:action move-disk
    :parameters (?disk ?below -disk ?new -below -disk)
    :precondition (and (disk ?disk)
      (smaller ?disk ?new -below -disk)
      (not (= ?new -below -disk ?below -disk))
      (not (= ?new -below -disk ?disk))
      (on ?disk ?below -disk)
      (clear ?disk)
      (clear ?new -below -disk))
    :effect (and (clear ?below -disk)
      (on ?disk ?new -below -disk)
      (not (on ?disk ?below -disk))
      (not (clear ?new -below -disk)))))

*Requirement flags allow a planner to quickly tell if it is likely to be able to handle the domain
*Action effects can include universal quantifiers (i.e. forall), conditionals (e.g. when)

PDDL

- Problem description

(define (problem hanoi-problem)
  (:domain hanoi-domain)
  (:objects p1 p2 p3 d1 d2 d3)
  (:init (smaller d1 p1) (smaller d2 p1) (smaller d3 p1) (smaller d1 p2) (smaller d2 p2) (smaller d3 p2) (smaller d1 p3) (smaller d2 p3) (smaller d3 p3) (clear p1) (clear p2) (clear p3) (disk d1) (disk d2) (disk d3) (on d1 d2) (on d2 d3) (on d3 p1) (on d3 p2) (on d3 p3))
  (:goal (and (on d1 d2) (on d2 d3) (on d3 p1) )))

*domain must match the corresponding domain name