Perspectives on Artificial Intelligence Planning

Héctor Geffner

ICREA & Universitat Pompeu Fabra
Barcelona, Spain

Motivation

- Planning is a form of **general problem solving**

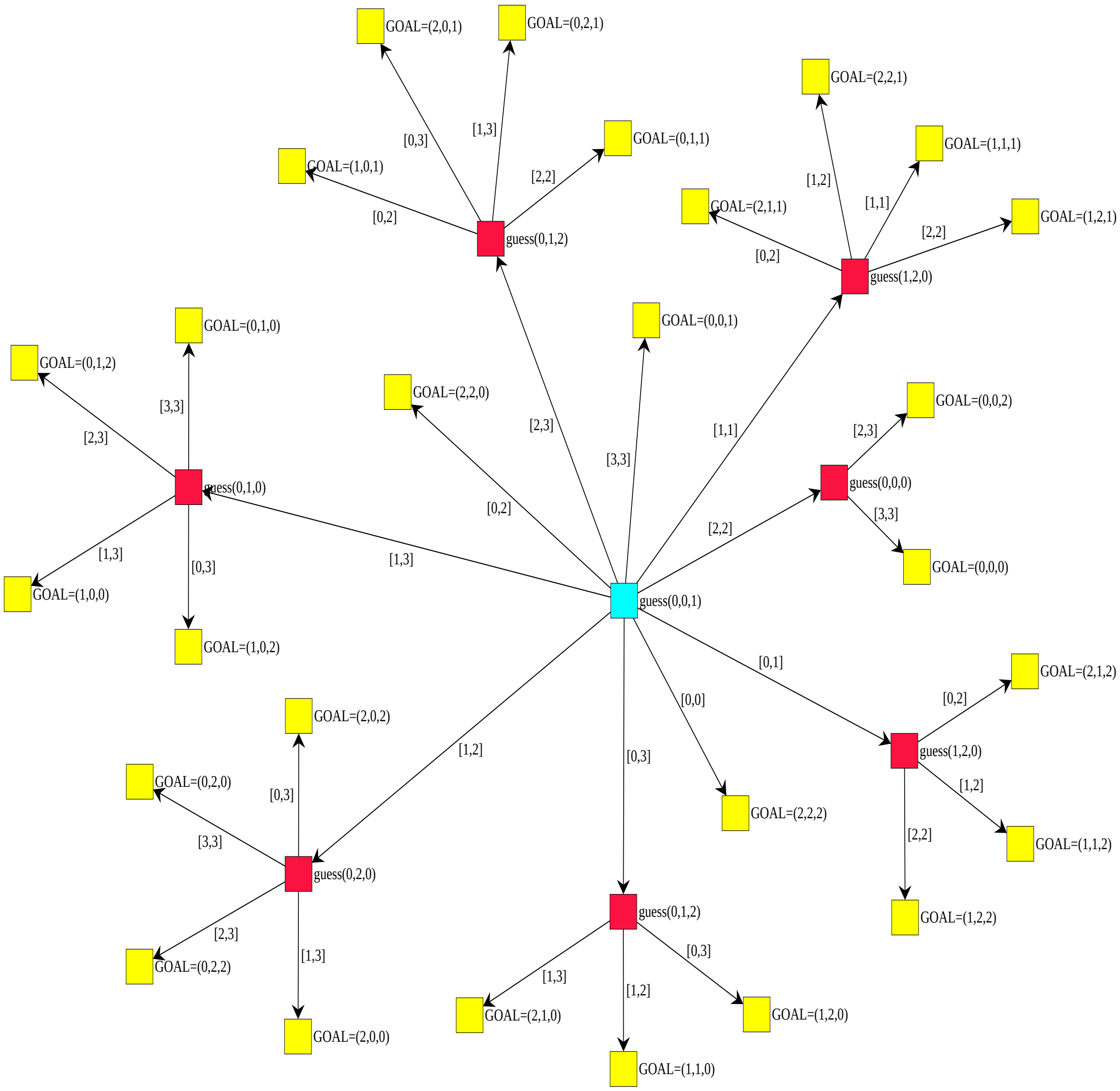
Problem $\Longrightarrow$ *Language* $\Longrightarrow$ $\boxed{\textit{Planner}}$ $\Longrightarrow$ *Solution*

- **Idea:** problems **described** at high-level and **solved** automatically
- **Goal:** facilitate modeling with small penalty in performance

Example: Mastermind

```
(define (domain mastermind)
  (:objects d0 d1 d2 d3 - [0,3] ..)

  (:action guess ?x0 ?x1 ?x2 ?x3 - [0,3]

      :observation (+ (:if (= ?x0 d0) 1 0)
                      (+ (:if (= ?x1 d1) 1 0)..))

                   (+ (:if (hit ?x0) 1 0)
                      (+ (:if (hit ?x1) 1 0)..)))

  (:predicate (hit ?x - [0,3])
     (:or (= ?x d0) (= ?x d1) (= ?x d2) .. )))

(define (problem m4)
  (:domain mastermind)
  (:init (:set d0 :in [0,3])
         (:set d1 :in [0,3]) ..)
  (:goal :certainty))
```

Optimal Strategy for Mastermind (3,3)



Planning and Action Selection

Three approaches in AI to the problem of **action selection** or **control**

1. **Programming:** specify control by hand
2. **Planning:** specify problem by hand, derive control automatically
3. **Learning:** learn control from experience

All have strengths and weaknesses; approaches not exclusive and often complementary

Three key ingredients in Planning

- **languages** for representing problems
- **algorithms** for solving them
- **models** for understanding them

Models needed to define **scope** of a planner:

- what is a **planning problem**
- what is a **solution** (plan), and
- what is an **optimal** solution

State model for Classical AI Planning

- finite and discrete state space S
- an initial state $s_0 \in S$
- a set $S_G \subseteq S$ of goal states
- actions $A(s) \subseteq A$ applicable in each $s \in S$
- a transition function $s' = f(a, s)$ for $a \in A(s)$
- action costs $c(a, s) > 0$

A **solution** is a sequence of applicable actions that maps s_0 into S_G

An **optimal** solution minimizes sum of action costs

Models in Planning and elsewhere

- Mathematical models in OR, Control, ...
  - *linear programs*
  - *integer programs*
  - *differential equations* + *quadratic costs*
  - ...
- AI Planning differs in two ways
  - concerned with **different** class of models
  - models represented implicitly in **language**

Planning Languages

Play two roles

- **specification:** concise model description
- **computation:** reveal useful heuristic info

- A **problem** in Strips is a tuple $\langle A, O, I, G \rangle$:
  - A stands for set of all **atoms** (boolean vars)
  - O stands for set of all **operators** (actions)
  - $I \subseteq A$ stands for **initial situation**
  - $G \subseteq A$ stands for **goal situation**
- Operators $o \in O$ **represented** by three lists
  - the **Add** list $Add(o) \subseteq A$
  - the **Delete** list $Del(o) \subseteq A$
  - the **Precondition** list $Pre(o) \subseteq A$

Strips: From Language to Models

Strips problem $P = \langle A, O, I, G \rangle$ determines **state model** $\mathcal{S}(P)$ where

- the states $s \in S$ are **collections of atoms**
- the initial state s_0 is I
- the goal states s are such that $G \subseteq s$
- the actions a in $A(s)$ are s.t. $Prec(a) \subseteq s$
- the next state is $s' = s - Del(a) + Add(a)$
- action costs $c(a, s)$ are all 1

The (optimal) **solution** of problem P is the (optimal) **solution** of State Model $\mathcal{S}(P)$

Model for Planning with Uncertainty 1 (No Feedback)

- finite and discrete state space S
- a **set** of possible initial states $S_0 \subseteq S$
- a set $S_G \subseteq S$ of goal states
- actions $A(s) \subseteq A$ applicable in each $s \in S$
- a **non-deterministic** transition function F such that $F(a, s)$ is a **set** of states, $a \in A(s)$
- action costs $c(a, s) > 0$

– A **solution** is a sequence of actions that lead to S_G for **any** possible initial state and transition

– Planning over this class of models called **Conformant Planning**

Model for Planning with Uncertainty 2 (Full Feedback)

Same model but states assumed **observable**

- finite and discrete state space S
- a **set** of possible initial states $S_0 \subseteq S$
- a set $S_G \subseteq S$ of goal states
- actions $A(s) \subseteq A$ applicable in each $s \in S$
- a **non-deterministic** transition function F ...
- action costs $c(a, s) > 0$

– **Solutions** become **functions** mapping states into actions (closed-loop control policies)

– **Optimal** solutions minimize cost in **worst case**; similar **probabilistic** model for minimizing **expected costs** (MDPs)

Model for Planning with Uncertainty 3 (Partial Feedback)

Same model but with **sensor**

- finite and discrete state space S
- a set of possible initial states $S_0 \subseteq S$
- a set $S_G \subseteq S$ of goal states
- actions $A(s) \subseteq A$ applicable in each $s \in S$
- a non-deterministic transition function ...
- **sensor** O **mapping states into observations**
- action costs $c(a, s) > 0$

– **Solutions** map **belief states** into actions

– A **belief state** represents **set** of states deemed possible, and summarizes past actions and observations

Framework

- **models** for defining the planning tasks
- **languages** for defining models concisely
- **algorithms** for solving them

Next

Algorithms for **classical planning** where significant progress achieved

- **empirical methodology**
  - standard PDDL language
  - planners and benchmarks available
  - AIPS planning competition, ...
- **novel formulations**
  - Graphplan, SAT, HSP, ...
- **large problems solved**

Heuristic Search Planning

Simple, powerful, and explains success of recent approaches in relation to previous approaches

- Explicitly **searches** state space with heuristic $h(s)$ that estimates cost from s to Goal
- Heuristic h extracted **automatically** from problem encoding

Heuristic Functions

- Heuristics derived as **optimal** cost function of **relaxed** problems
- Simple relaxations used in Planning
  - ignore delete lists
  - reduce large goal sets to subsets
  - ignore certain atoms

Additive Heuristic

- For all **atoms** p:

$$g(p;s) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } p \in s \text{, else} \\ \min_{a \in O(p)} [1 + g(Prec(a);s)] \end{cases}$$

- For **sets** of atoms C, assume **independence**:

$$g(C;s) \stackrel{\text{def}}{=} \sum_{r \in C} g(r;s)$$

- Resulting **heuristic function** $h_{add}(s)$:

$$h_{add}(s) \stackrel{\text{def}}{=} g(Goals;s)$$

Heuristic not admissible but informative and fast

Max Heuristic

- For all **atoms** p:

$$g(p;s) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } p \in s\text{, else} \\ \min_{a \in O(p)}[1 + g(Prec(a);s)] \end{cases}$$

- For **sets** of atoms C, replace **sum** by **max**

$$g(C;s) \stackrel{\text{def}}{=} max_{r \in C}\, g(r;s)$$

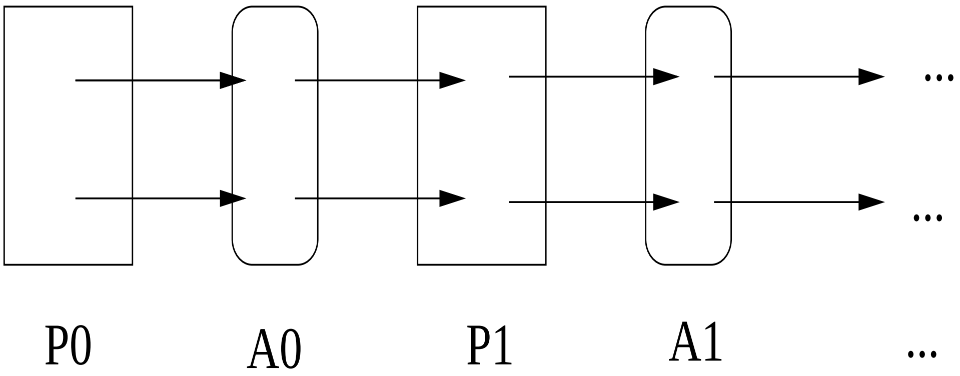
- Resulting **heuristic function** $h_{max}(s)$:

$$h_{max}(s) \stackrel{\text{def}}{=} g(Goals;s)$$

Heuristic admissible but not very informative ...

Max Heuristic and Planning Graphs

- Build reachability graph P_0, A_0, P_1, A_1, ...



$$
\begin{aligned}
P_0 &= \{p \in Init\} \\
A_i &= \{a \in O \mid Prec(a) \subseteq P_i\} \\
P_{i+1} &= \{p \in Add(a) \mid a \in A_i\}
\end{aligned}
$$

- Graph implicitly **represents** max heuristic:

$$h_{max}(s) = \text{min } i \text{ such that } s \subseteq P_i$$

Need No-op(p) action for each p: $Prec = Add = \{p\}$

More Informed h in Graphplan

- Graphplan also keeps track of **pairs** that cannnot be reached **simultaneously** in i steps, $i = 0, 1, \ldots$
  - **action** pair **mutex** at i if incompatible or preconditions **mutex** at i
  - **atom** pair **mutex** at $i + 1$ if supporting action pair **mutex** at i

- **Mutexes** computed with planning graph and yield **more informed** admissible h

$h_G(s) \stackrel{\text{def}}{=}$ min i s.t. $s \subseteq P_i$ & s **not** mutex at i

Generalization: h^m heuristics

For fixed $m = 1, 2, \ldots$, assume cost of achieving set C given by cost of **most costly subset of size** m

- For $m = 1$, $h^m = h_{max}$
- For $m = 2$, $h^m = h_G$ (Graphplan)
- For any m, h^m admissible and polynomial

$$h^m(C) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } C \subseteq s \\ \min_a [1 + h^m(R(C,a))] & |C| \leq m \\ \max_{D \subseteq C, |D| = m} h^m(D) & |C| > m \end{cases}$$

where **regression** $R(C,a) = C - Add(a) + Prec(a)$ for a such that $Del(a) \cap C = \emptyset$

Heuristic and Constraint-based Search

- **All** modern planners rely on **heur. estimators**
- Yet some perform **constraint-based search** on SAT/CSP translation of planning graph
- **Optimal** test in every node σ **intractable**

$$f^*(\sigma) \leq Bound$$

- In Heuristic Search, **polynomial lower bound** $f \leq f^*$ used
- In Constraint-based Search, **polynomial but incomplete consistency test** used (**constraint propagation**)

Beyond Classical Planning

Heuristics h required but not enough

- In **temporal planning**
  - **branching** *also key (POP, constraints, . . .)*
- In **planning with uncertainty**
  - **heuristic-search** & **dynamic programming** *integration (RTDP, LAO*, . . .)*
- In presence of **partial sensing**
  - **belief representation** *(*e.g., BDDs)

Planning in Uncertain Worlds with 'Strips'?

Often you can; two misconceptions

1. uncertain **world** implies uncertain **model**
2. **actions** in Strips must be **'primitive'**

1. **Model $\neq$ World:** often **closed-loop deterministic planning** (replanning from each state) adequate

2. **Actions can be closed-loop policies;** planning for assembling **low-level behaviors** for achieving high level goals

Summary

- Three key ingredients in planning:

  languages, **models**, **algorithms**

- Significant progress in classical setting

  novel approaches, empirical methodology

- Convergence of number of threads concerned with **modeling and problem solving:**

  search, planning, constraints, uncertainty, kr, dynamic programming, combinatorial optimization, ...

ACK: Blum & Furst for Graphplan, Kautz and Selman for SATPlan, Blai Bonet for collaboration, Judea Pearl