Search and Inference in AI Planning

Héctor Geffner

ICREA & Universitat Pompeu Fabra
Barcelona, Spain

Joint work with V. Vidal, B. Bonet, P. Haslum, H. Palacios, . . .

AI Planning

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

Problem $\Longrightarrow$ *Language* $\Longrightarrow$ *Planner* $\Longrightarrow$ *Solution*

- **Idea:** problems **described** at high-level and **solved** automatically

- **Goal:** facilitate modeling, maintain performance

Planning and General Problem Solving: How general?

For which **class of problems** a planner should work?

- **Classical planning** focuses on problems that map into **state models**
  - state space S
  - initial state $s_0 \in S$
  - goal states $S_G \subseteq S$
  - actions $A(s)$ applicable in each state s
  - transition function $s' = f(a, s)$, $a \in A(s)$
  - action costs $c(a, s) > 0$
- A **solution** of this class of models is a **sequence of applicable actions** mapping the inital state s_0 into a goal state S_G
- It is **optimal** if it minimizes sum of action costs
- Other **models** for planning with **uncertainty** (conformant, contingent, Markov Decision Processes, etc), **temporal planning**, etc.

Planning Languages

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 $\mathcal{S}(P)$

The Talk

- Focus on approaches for **optimal sequential/parallel/temporal domain-independent planning** (SAT, Graphplan, Heuristic Search, CP)

- Significant progress in last decade as a result of empirical methodology and novel ideas

- Three messages:

  1. It is all (or mostly) **branching** and **pruning**
  2. Yet novel and powerful techniques developed in planning context
  3. Some of these techniques potentially applicable in other contexts

Planning as SAT

Theory with **horizon** n for Strips problem $P = \langle A, O, I, G \rangle$:

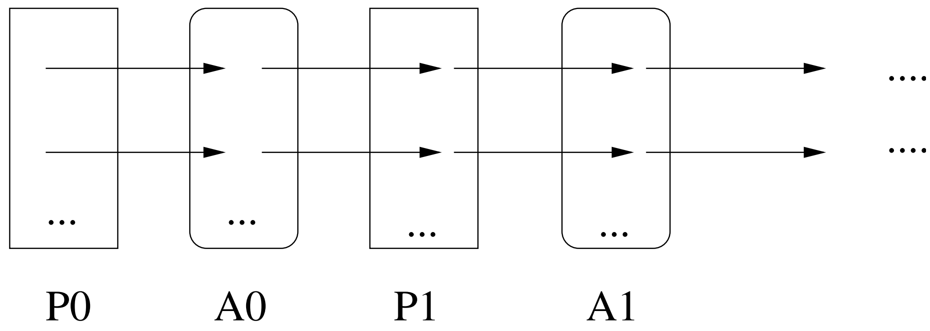
1. **Init:** p_0 for $p \in I$, $\neg q_0$ for $q \not\in I$
2. **Goal:** p_n for $p \in G$
3. **Actions:** For $i = 0, 1, \ldots, n-1$ (including NO-OPs)

   $a_i \supset p_i$ for $p \in Prec(a)$ (Preconds)
   $a_i \supset p_{i+1}$ for each $p \in Add(a)$ (Adds)
   $a_i \supset \neg p_{i+1}$ for each $p \in Del(a)$ (Deletes)
4. **Frame:** $\bigwedge_{a:p \in Add(a)} \neg a_i \quad \supset \quad \neg p_{i+1}$
5. **Concurrency:** If a and a' incompatible, $\neg(a_i \wedge a'_i)$

In practice, however, SAT and CSP planner build theory from Graphplan's **planning graph** that encodes useful **lower bounds**

Planning Graphs and Lower Bounds

- Build layered graph $P_0,\ A_0,\ P_1,\ A_1,\ \ldots$



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

Heuristic $h_1(G)$ defined as time where G becomes **reachable** is a **lower bound** on **number of time steps** to actually achieve G:

$$h_1(G) \stackrel{\text{def}}{=} \min\ i \ \text{ s.t. } G \subseteq P_i$$

The Planning Graph and Variable Elimination

- Graphplan actually builds more complex layered graph by keeping track of atom and action **pairs** that cannnot be reached simultaneously (mutexes)

- Resulting heuristic h_2 is more informed than h_1; i.e., $0 \leq h_1 \leq h_2 \leq h^*$

- Graphplan builds graph **forward** in first phase, then extracts plan **backwards** by backtracking

- This is analogous to **bounded variable elimination** (Dechter et al):
  - In VE, variables eliminated in one order (inducing constraints of size up to n) and solved backtrack-free in reverse order

  - In Bounded VE, var elimation phase yields constraints of **bounded** size m, followed by backtrack search in reverse

The planning Graph and Variable Elimination (cont'd)

- Graphplan does actually a precise form of **Bounded-m Block Elimination** where **whole layers** are eliminated in one step inducing constraints of size m over next layer

- While Bounded-m Block Elimination is **exponential** in the size of the blocks/layers in the worst case; Graphplan does it in **polynomial time** exploiting simple **stratified** structure of Strips theories [Geffner KR-04]

Two reconstructions of Graphplan

Graphplan can thus be understood **fully** as either

- a **CSP** planner that does **Bounded-2 Layer Elimination** followed by **Backtrack** search, or

- an **Heuristic Search** Planner that first computes an **admissible heuristic** and then uses it to drive an **IDA* search** from the goal

It is interesting that both approaches yield **equivalent account** in this setting

Temporal Planning: the Challenge

- We can extract lower bounds h automatically from problems, and get a reasonable optimal **sequential** planner by using an **heuristic search** algorithm like IDA*

- We can translate the planning graph into SAT, and get a reasonable optimal **parallel** planner using a state-of-the-art **SAT solver**

- Neither approach, however, extends naturally to **temporal** planning:
  - in HS approaches, the **branching scheme** is not suitable
  - in SAT approaches, the **representation** is not suitable

- These limitations were the motivation for **CPT**, a CP-based temporal planner that
  - **minimizes makespan**, and
  - is **competitive with SAT planners** when durations are uniform

Semantics of Temporal Plans

A temporal (Strips) plan is a set of **actions** $a \in Steps$ with their start times $T(a)$ such that:

1 **Truth** Every precondition p of a is true at $T(a)$

2 **Mutex:** Interfering actions in the plan do not overlap in time

Assuming 'dummy' actions $Start$ and End in plan, 1 decomposed as

1.1 **Precond:** Every precond p of $a \in Steps$ is **supported** in the plan by an **earlier** action a'

1.2 **Causal Link:** If a' supports precond p of a in plan, then all actions a'' in plan that **delete** p must come **before** a' or **after** a

Partial Order Causal Link (POCL) Branching

POCL planners (temporal and non-temporal alike), start with a partial plan with $Start$ and End and then loop:

- adding actions, supports, and precedences to enforce 1.1 (fix open supports)
- adding precedences to enforce 1.2 and 2 (fix threats)
- **backtracking** when resulting precedences in the plan form an inconsistent **Simple Temporal Network (STP)** [Meiri et al], or no other fix

The problem with POCL Planning (and Dynamic CSP!)

- POCL branching yields a **simple** and **elegant** algorithm for temporal planning; the problem is that it is just . . . **branching!**

- Pruning partial plans whose STP network is not consistent **does not suffice to match performance of modern planners**

- For this, it is crucial to **predict failures earlier**; the question is how to do it.

- The key part is to be able to **reason with all possible actions, and not only those in current partial plan.**

- This is indeed what Graphplan and SAT approaches do in non-temporal setting

  (Similar problem in **Dynamic CSPs**; need to reason about all possible vars, not only those in 'current' CSP)

CPT: A CP-based POCL Planner

- Key novelty in CPT are the strong mechanisms for reasoning about **all actions in the domain** (start times, precedences, supports, etc), and not only **those in current plan**.

- This involves novel constraint-based **representation** and **propagation rules**, as in particular, an action can occur 0, 1, 2, or many times in the plan!

- CPT provides effective solution to the underlying **Dynamic CSP**

CPT: Formulation

- Variables
- Preprocessing
- Constraints
- Branching

Variables

For **all actions in the domain** $a \in O$ and preconditions $p \in Pre(a)$:

- $T(a) :: [0, \infty]$ = **starting time** of a
- $S(p, a) :: \{a' \in O | p \in Add(a')\}$ = **support** of p for a
- $T(p, a) :: [0, \infty]$ = **starting time of support** $S(p, a)$
- $InPlan(a) :: [0, 1]$ = **presence of** a **in the plan**

Preprocessing

- **Initial lower bounds:** $T_{min}(a) = h^2_T(a)$
- **Structural mutexes:** pairs of atoms p, q for which $h^2_T(\{p, q\}) = \infty$
- **e-deleters:** extended deletes computed from structural mutexes
- **Distances:**
  - $dist(a, a') = h^1_T(a')$ with $I = I_a$
  - $dist(Start, a) = h^2_T(a)$
  - $dist(a, End)$: shortest-path algorithm on a 'relevance graph'
- E-deleters and Distances used to make constraints tighter;
  $\delta(a', a) \stackrel{\text{def}}{=} duration(a') + dist(a', a)$

Constraints

- **Bounds:** for all $a \in O$

$$T(Start) + dist(Start, a) \leq T(a)$$

$$T(a) + dist(a, End) \leq T(End)$$

- **Preconditions:** supporter a' of precondition p of a must precede a:

$$T(a) \;\; \geq \;\; \min_{a' \in [D(S(p,a)]} [T(a') + \delta(a', a)]$$

$$T(a') + \delta(a', a) > T(a) \rightarrow S(p, a) \neq a'$$

- **Causal Link Constraints:** for all $a \in O$, $p \in pre(a)$ and a' that e-deletes p, a' precedes $S(p, a)$ or follows a:

$$T(a') + dur(a') + \min_{a'' \in D[S(p,a)]} dist(a', a'') \leq T(p, a) \quad \vee \quad T(a) + \delta(a, a') \leq T(a')$$

Constraints (cont'd)

- **Mutex Constraints:** for effect-interfering a and a'

$$T(a) + \delta(a, a') \leq T(a') \;\; \vee \;\; T(a') + \delta(a', a) \leq T(a)$$

- **Support Constraints:** $T(p, a)$ and $S(p, a)$ related by

$$S(p, a) = a' \rightarrow T(p, a) = T(a')$$

$$\min_{a' \in D[S(p,a)]} T(a') \;\; \leq \;\; T(p, a) \;\; \leq \;\; \max_{a' \in D[S(p,a)]} T(a')$$

$$T(p, a) \neq T(a') \rightarrow S(p, a) \neq a'$$

Branching

- A **Support Threat** $\langle a', S(p,a) \rangle$ generates the split

$$[T(a') + dur(a') + \min_{a'' \in D[S(p,a)]} dist(a', a'') \leq T(p,a);$$

$$T(a) + \delta(a, a') \leq T(a')]$$

- An **Open Condition** $S(p,a)$ generates the split

$$[S(p,a) = a'; S(p,a) \neq a']$$

- A **Mutex Threat** $\langle a, a' \rangle$ generates the split

$$[T(a) + \delta(a, a') \leq T(a'); T(a') + \delta(a', a) \leq T(a)]$$

Two subtle issues and their solutions in CPT

1. **Conditional variables:** variables associated with actions not yet included or excluded from current plan
   - propagate **into** those variables but never **from** them
   - domains meaningful **under assumption** that action eventually in plan

2. **Action Types vs. Tokens:** dealing with unknown number of tokens?
   - Variables associated with both **action types** and **action tokens**
   - **Action tokens** generated dynamically from action types by **cloning**
   - **Action types** summarize all tokens of same type not yet in plan

-- 1 relevant for **Dynamic CSP:** need to reason about **all** potential vars and not only those in '**current**' CSP

-- 2 relevant for certain **Symmetries**; e.g., hammers in box 'symmetrical' til one picked

Current Status of CPT

1. It currently appears as the best optimal **temporal** planner

2. Competitive with SAT **parallel** planners in the **special case** when action durations are uniform

3. Recent extension solves wide range of benchmark domains **backtrack-free!** (Blocks, Logistics, Satellite, Gripper, Miconic, Rovers, etc).

4. In such a case, **optimality** is not enforced (see Vincent presentation later today for details)

Summary

- Optimal planners (Graphplan, SAT, Heuristic Search) can all be understood as **branching** and **pruning**
- Big performance jump in last decade is the result of **pruning**; til Graphplan search was basically **blind**, although useful **branching** schemes
- Planning theories have **stratified** structure which is exploited in construction of **planning graph** and used by SAT approaches
- Temporal planning particularly suited for CP; CPT combines **POCL branching**, **lower bounds** obtained at preprocessing, and **pruning** based on CP formulation that reasons about **all actions in the domain**
- Some ideas in CPT potentially relevant for dealing with **Dynamic CSPs** and certain classes of **symmetries**