

Planning as SAT (Kautz & Selman)

- Maps planning problem $P = \langle A, O, I, G \rangle$ with horizon n into a **set of clauses** $C(P, n)$, solved by **SAT solver** (satz, chaff, . . .).
- Theory $C(P, n)$ includes vars p_0, p_1, \dots, p_{n+1} and a_0, a_1, \dots, a_n for each $p \in A$ and $a \in O$
- $C(P, n)$ satisfiable iff there is a parallel plan with length n ; in that case, plan extracted from satisfying assignment
- In parallel plan, **non-mutex** actions can be executed in parallel; two actions are **mutex** if one deletes precs/adds of the other
- **Optimal** parallel plans minimize number of time steps; they are obtained by starting with optimistic horizon n (lower bound), and increasing it by 1 til $C(P, n)$ satisfiable

Theory $C(P, n)$ for Problem $P = \langle A, O, I, G \rangle$

1. **Init:** p_0 for $p \in I$, $\neg q_0$ for $q \notin I$
2. **Goal:** p_{n+1} for $p \in G$
3. **Actions:** For $i = 0, 1, \dots, n - 1$
 - $a_i \supset p_i$ for $p \in \text{Prec}(a)$
 - $a_i \supset p_{i+1}$ for each $p \in \text{Add}(a)$
 - $a_i \supset \neg p_{i+1}$ for each $p \in \text{Del}(a)$
4. **NO-OPs:** For each p , and $i = 0, 1, \dots, n - 1$, 'dummy' **NO-OP**(p) action added, with precondition and add list p and empty delete list.
5. **Frame:** If a^1, \dots, a^m are the actions that add p , then for $i = 0, \dots, n - 1$:

$$\neg a_i^1 \wedge \dots \wedge \neg a_i^m \subset \neg p_{i+1}$$

6. **Mutex:** If a and a' mutex, $\neg(a_i \wedge a'_i)$

- Current SAT/CSP formulations (Blackbox) built on top of **Graphplan**; translation exploits lower bounds in planning graph

Example

- Initial Situation $I = \{q\}$
- Actions:
 - $a : Pre(a) = \{q\} ; Add(a) = \{p\} ; Del(a) = \{q\}$
 - $b : Pre(b) = \{q\} ; Add(b) = \{r\} ; Del(b) = \{\}$
 - $c : Pre(c) = \{r\} ; Add(c) = \{q\} ; Del(c) = \{r\}$
- Goal: $G = \{p, q\}$

Solve problem by using SAT formulation and horizon $n = 0, 1, 2, \dots$

Back to State Planning: Regression Planning

Search backward from **goal** rather than forward from initial state:

- initial state σ_0 is G
- a **applicable** in σ if $Add(a) \cap \sigma \neq \emptyset$ and $Del(a) \cap \sigma = \emptyset$
- resulting state is $\sigma_a = \sigma - Add(a) + Prec(a)$
- terminal states σ if $\sigma \subseteq I$

Advantages/Problems:

- + Heuristic $h(\sigma)$ for any σ can be computed by simple aggregation (max,sum, . . .) of estimates $g(p, s_0)$ for $p \in \sigma$ computed only **once** from s_0
- Spurious states σ not reachable from s_0 often generated (e.g., where a block is on two blocks at the same time). A good h should make $h(\sigma) = \infty$. . .

Variation: Parallel Regression Search

Search backward from goal assuming that **non-mutex actions** can be done in **parallel**

- The regression search is similar, except that **sets of non-mutex actions A allowed**: $Add(A) = \cup_{a \in A} Add(a)$, $Del(A) = \cup_{a \in A} Del(a)$, $Prec(A) = \cup_{a \in A} Prec(a)$.
- Resulting state from regression is $\sigma_A = \sigma - Add(A) + Prec(a)$

Advantages/Problems:

- + Sometimes easier to compute optimal **parallel plans** than optimal **serial plans**
- + Some heuristics provide tighter estimates of **parallel cost** than **serial cost** (e.g., $h = h1$)
- **Branching factor** in parallel search (either forward or backward) can be very large (2^n if n applicable actions).

Parallel Regression Search with NO-OPs

- Assumes 'dummy' operator **NO-OP(p)** for each p with $Prec = Add = \{p\}$ and $Del = \emptyset$
- A set of non-mutex actions A (possibly including NO-OPs) applicable in σ if $\sigma \subseteq Add(A)$ and $Del(A) \cap \sigma = \emptyset$
- Resulting state is $\sigma = Prec(A)$
- Starting state $\sigma_0 = G$ and terminal states $\sigma \subseteq I$

Advantages/Problems:

- More actions to deal with
- + Enables certain compilation techniques as in Graphplan . . .

Graphplan (Blum & Furst): First Version

- Graphplan does an IDA* parallel regression search with NO-OPs over **planning graph** containing **propositional** and **action layers** P_i and A_i , $i = 0, \dots, n$
 - P_0 contains the atoms true in I
 - A_i contains the actions whose precs are true in P_i
 - P_{i+1} contains the **positive** effects of the actions in A_i
- planning graph built til layer P_n where G appears, then **search** for plans with horizon $n - 1$ invoked with $Solve(G, n)$ where
 - $Solve(G, 0)$ succeeds if $G \subseteq I$ and fails otherwise, and
 - $Solve(G, n)$ mapped into $Solve(Prec(A), n - 1)$, where A is a **set of non-mutex actions in layer** in A_{n-1} that covers G , i.e., $G \subseteq Add(A)$.
- If search fails, n increased by 1, and process is repeated

Graphplan: Real version

- The IDA* search is **implicit**; heuristic $h(\sigma)$ encoded in planning graph as **index of first layer P_i that contains σ**
- This heuristic, as defined above, corresponds to the $hmax = h1$ heuristic; Graphplan actually uses a more powerful admissible heuristic akin to $h_2 \dots$
- **Basic idea: extend mutex relations to pairs of actions and propositions in each layer $i > 0$ as follows:**
 - p and q **mutex in P_i** if p and q are in P_i and the actions in A_{i-1} that support p and q are **mutex in A_{i-1}** ;
 - a and a' **mutex in A_i** if a and a' are in A_i , and they are **mutex** or $Prec(a) \cup Prec(a')$ contains a **mutex in P_i**
- The **index of first layer** in planning graph that contains a set of atoms P or actions A **without a mutex**, is a **lower bound**
- Thus, search can be **started** at level in which G appears without a

mutex, and $Solve(P, i)$ needs to consider only sets of actions A in A_{i-1} that **do not contain a mutex**.

Graphplan is first 'modern' planner, preceded SATPlan and HSP, and still source of useful ideas . . .

POP: Regression + Decomposition

Basic idea in Partial Order Planning:

1. recursively **decompose** regression with goal p_1, \dots, p_n into n regressions with goals $p_i, i = 1, \dots, n$;
2. **combine** resulting plans so that they **do not interfere** with each other

E.g.: let $G = \{p, q\}$, $I = \{r\}$, and two actions

$$a1: \text{Prec}(a1) = \{r\}, \text{Add}(a1) = \{p\}, \text{Del}(a1) = \{r\}$$

$$a2: \text{Prec}(a2) = \{r\}, \text{Add}(a2) = \{q\}, \text{Del}(a2) = \{\}$$

-- $P1 = \{a1\}$ is a plan for p , and $P2 = \{a2\}$ a plan for q

-- Yet $a1$ in $P1$ **deletes** a precondition of $a2$

-- This 'threat' can be solved by forcing $a1$ **after** $a2$, i.e., $a2 \prec a1$.

Partial Order Causal Link planning is a formulation of POP that pursues 1 and 2 concurrently

Partial Plans and Causal Links

A **partial plan** P in POCL is a triple $(Steps, \mathcal{O}, CLs)$ where

- $Steps$ is a set of **actions** a_i
 - \mathcal{O} is a set of **precedence constraints** $a_i \prec a_j$
 - CLs is a set of **causal links** $(a1, p, a2)$ meaning that that precondition p of $a2$ is achieved by action $a1$
-
- POCL extends partial plans til they become 'complete'; i.e., states σ in the search are partial plans
 - Initial partial plan is $P_0 = (\{Start, End\}, \{Start \prec End\}, \{\})$, where $Start$ and End are actions that summarize I and G : $Add(Start) = I$, $Prec(End) = G$, $Rest(*) = \emptyset$

POCL Planning Algorithm

By construction, a partial plan $P = (Steps, \mathcal{O}, CLs)$ will be **complete** when it contains no 'flaw' of the form:

1. **unsupported precondition:** a precondition $p \in Prec(a)$ for $a \in Steps$ s.t. no CL (a', p, a) in CLs
 2. **threatened causal link:** a CL (a', p, a) for $b \in Steps$ s.t. $p \in Del(b)$ and $a' \prec b \prec a$ is consistent with \mathcal{O}
 3. **inconsistency:** \mathcal{O} is inconsistent
-
- **Flaw #1:** fixed by selecting an action a' , $p \in Add(a)$, and adding a' to $Steps$, $a' \prec a$ to \mathcal{O} , and (a', p, a) to CLs
 - **Flaw #2:** fixed by adding $b \prec a'$ or $a \prec b$ to \mathcal{O}
 - **Flaw #3:** cannot be fixed

- POCL search **starts** with the plan $P = P_0$ above, selecting a flaw in P , and trying each one of the repairs.
- The **terminal** plans (states) are the complete plans (solutions), or the partial plans that cannot be fixed (dead ends)

Status of POCL Planning

- POP/POCL dominated planning research for 10-15 years, until Graphplan
- Unlike other approaches, can work with action schemas
- In recent years lost favor to Graphplan/SAT/CSP/HSP
- Recent comeback combined with heuristics as in RePOP
- Holds promise as **branching scheme for temporal planning . . .**

Selected Bibliography for Current Research

- Bacchus, F. The 2000 AI Planning Systems Competition. *Artificial Intelligence Magazine* 22(3). 2001.
- Blum, A., and Furst, M. Fast planning through planning graph analysis. In *Proceedings of IJCAI-95*, 1636--1642. 1995
- Bonet, B., and Geffner, H. Planning as heuristic search. *Artificial Intelligence* 129(1--2):5--33. 2001.
- Fikes, R., and Nilsson, N. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 1:27--120. 1971
- Fox, M., and Long, D. PDDL2.1: An extension to PDDL for expressing temporal planning domains. At www.dur.ac.uk/d.p.long/competition.html. 2001.
- Geffner, H. Perspectives on AI Planning. *Proceedings AAAI-2002*. MIT Press. 2002.
- Haslum, P., and Geffner, H. Admissible heuristics for optimal planning. In *Proc. of the Fifth International Conference on AI Planning Systems (AIPS-2000)*, 70--82. 2000.
- Hoffmann, J., and Nebel, B. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 2001:253--302.
- Kautz, H., and Selman, B. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proceedings of AAAI-96*, 1194--1201.
- Kautz, H., and Selman, B. Unifying SAT-based and Graph-based planning. *Proceedings IJCAI-99*, 318--327.
- D. McAllester and D. Rosenblitt. Systematic Non-linear Planning. In *Proceedings of AAAI-91*, pp 634-639, 1991.
- McDermott, D. A heuristic estimator for means-ends analysis in planning. In *Proc. Third Int. Conf. on AI Planning Systems (AIPS-96)*.
- McDermott, D. The 1998 AI Planning Systems Competition. *Artificial Intelligence Magazine* 21(2):35--56. 2000.

- Nebel, B. Compilation schemes: A theoretical tool for assessing the expressive power of planning formalisms. In *KI-99: Advances in Artificial Intelligence*, 183--194. k Springer-Verlag.
- Newell, A., and Simon, H. GPS: a program that simulates human thought. In Feigenbaum, E., and Feldman, J., eds., *Computers and Thought*. McGraw Hill. 1963. 279--293.
- Nguyen, X. L., and Kambhampati, S. Reviving partial order planning. In *Proc. IJCAI-01*.
- Penberthy, J., and Weld, D. Ucpop: A sound, complete, partial order planner for adl. In *Proceedings KR'92*.
- J. Rintanen and Jörg Hoffmann. An Overview of Recent Algorithms for AI Planning. In *Proc. KI-2001*.
- Weld, D. S. An introduction to least commitment planning. *AI Magazine* 15(4):27--61. 1994
- Daniel S. Weld. Recent Advances in AI Planning. *AI Magazine*. 20(2): 93--123. 1999.
- Wilkins, D. E., and des Jardins, M. A call for knowledge-based planning. *AI Magazine* 22(1):99--115. 2001.