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,\ldots).

- Theory $C(P, n)$ includes vars $p_0, p_1, \ldots, p_{n+1}$ and $a_0, a_1, \ldots, a_n$ 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 $q \not\in I$
2. Goal: $p_{n+1}$ for $p \in G$
3. Actions: For $i = 0, 1, \ldots, n - 1$
   \begin{align*}
   a_i \supset p_i & \text{ for } p \in Prec(a) \\
   a_i \supset p_{i+1} & \text{ for each } p \in Add(a) \\
   a_i \supset \neg p_{i+1} & \text{ for each } p \in Del(a)
   \end{align*}
4. NO-OPs: For each $p$, and $i = 0, 1, \ldots, n - 1$, ‘dummy’ NO-OP$(p)$ action added, with
   precondition and add list $p$ and empty delete list.
5. Frame: If $a^1, \ldots, a^m$ are the actions that add $p$, then for $i = 0, \ldots, n - 1$:
   \[
   \neg a^1_i \land \cdots \land \neg a^m_i \subset \neg p_{i+1}
   \]
6. Mutex: If $a$ and $a'$ mutex, $\neg(a_i \land a'_i)$

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

• Initial Situation $I = \{q\}$

• Actions:
  - $a : \text{Pre}(a) = \{q\} ; \text{Add}(a) = \{p\} ; \text{Del}(a) = \{q\}$
  - $b : \text{Pre}(b) = \{q\} ; \text{Add}(b) = \{r\} ; \text{Del}(b) = \{\}$
  - $a : \text{Pre}(c) = \{r\} ; \text{Add}(c) = \{q\} ; \text{Del}(c) = \{r\}$

• Goal: $G = \{p, q\}$

Solve problem by using SAT formulation and horizon $n = 0, 1, 2, \ldots$
Back to State Planning: Regression Planning

Search backward from goal rather than forward from initial state:

- initial state $\sigma_0$ is $G$
- $a$ applicable in $\sigma$ 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 $\sigma$ if $\sigma \subseteq I$

Advantages/Problems:

+ Heuristic $h(\sigma)$ for any $\sigma$ 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 $\sigma$ 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: $Add(A) = \bigcup_{a \in A} Add(a)$, $Del(A) = \bigcup_{a \in A} Del(a)$, $Prec(A) = \bigcup_{a \in A} Prec(a)$.

• 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 $\text{NO-OP}(p)$ for each $p$ with $\text{Prec} = \text{Add} = \{p\}$ and $\text{Del} = \emptyset$

- A set of non-mutex actions $A$ (possibly including NO-OPs) applicable in $\sigma$ if $\sigma \subseteq \text{Add}(A)$ and $\text{Del}(A) \cap \sigma = \emptyset$

- Resulting state is $\sigma = \text{Prec}(A)$

- 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 $A_i$, $i = 0, \ldots, n$
  - $P_0$ 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 $\sigma$

- This heuristic, as defined above, corresponds to the $h_{max} = h_1$ heuristic; Graphplan actually uses a more powerful admissible heuristic akin to $h_2$ . . .

- 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 $\text{Prec}(a) \cup \text{Prec}(a')$ 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 $\text{Solve}(P, i)$ 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 \( p_1, \ldots, p_n \) into \( n \) regressions with goals \( p_i, \ i = 1, \ldots, n \);

2. **combine** resulting plans so that they **do not interfere** with each other

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

\[
\begin{align*}
a1: \quad & \text{Prec}(a1) = \{r\}, \ Add(a1) = \{p\}, \ Del(a1) = \{r\} \\
a2: \quad & \text{Prec}(a2) = \{r\}, \ Add(a2) = \{q\}, \ Del(a2) = \{} \\
\end{align*}
\]

-- \( P1 = \{a1\} \) 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 $(Steps, O, CLs)$ where

- $Steps$ is a set of actions $a_i$
- $O$ is a set of precedence constraints $a_i \prec a_j$
- $CLs$ is a set of causal links $(a_1, p, a_2)$ meaning that that precondition $p$ of $a_2$ is achieved by action $a_1$

- POCL extends partial plans til they become ‘complete’; i.e., states $\sigma$ 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 \( P = (\text{Steps}, \mathcal{O}, CLs) \) will be complete when it contains no ‘flaw’ of the form:

1. **unsupported precondition**: a precond \( p \in Prec(a) \) for \( a \in \text{Steps} \) s.t. no \( \text{CL} \ (a', p, a) \) in \( CLs \)

2. **threatened causal link**: a CL \( (a', p, a) \) for \( b \in \text{Steps} \) 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 \( \text{Steps} \), \( a' \prec a \) 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


