Automated Planning and Decision Making
Prof. Ronen Brafman

Graphplan

Based on slides by:
Ambite, Blyth and Weld
Basic Idea

- Construct a graph that encodes constraints on possible plans based on smart reachability analysis (Reachable-2)
- Use this "planning graph" to constrain search for a valid plan:
  - If a valid plan exists, it’s a sub-graph of the planning graph.
- A planning graph can be built for any problem in polynomial time.
Properties:

- Finds "shortest parallel plan".
- Sound, complete and will terminate with failure if no plan exists.
Planning Graph

- Directed, leveled graph
  - 2 types of nodes:
    - Proposition: P
    - Action: A
  - 3 types of edges (between levels)
    - Precondition: $P \rightarrow A$
    - Add: $A \rightarrow P$
    - Delete: $A \rightarrow P$

- Proposition and action levels alternate.
- Action level includes actions whose preconditions are satisfied in previous level plus no-op actions (to solve frame problem).
Planning Graph

No-Op

Proposition

Precondition

Action

Effect

...
Planning Graph Example - Rocket Problem
Constructing The Graphplan

- Level $P_1$: all literals from the initial state.
- Add an action in level $A_i$ if all its preconditions are present in level $P_i$.
- Add a precondition in level $P_i$ if it is the effect of some action in level $A_{i-1}$ (including no-ops).
- Maintain a set of exclusion relations to eliminate incompatible propositions and actions (thus reducing the graph size).

$$P_1A_1P_2A_2\cdots P_{n-1}A_{n-1}P_n$$
Mutual Exclusion Relations

- Idea: if two actions (or literals) are mutually exclusive (mutex) at some stage, no valid plan could contain both.
- Two actions are mutex if:
  - Interference: one destroys the others’ effect or precondition.
  - Competing needs: mutex preconditions.
- Two propositions are mutex if:
  - All ways of achieving them are mutex.
Mutual Exclusion Relations

Inconsistent Effects

Interference (prec-effect)

Competing Needs

Inconsistent Support
Dinner Date Example

- Initial Conditions:
  \[(\text{and (garbage) (cleanHands) (quiet)})\]
- Goal:
  \[(\text{and (dinner) (present) (not (garbage)})\]
- Actions:
  - Cook
    precondition: \(\text{(cleanHands)}\)
    effect: \(\text{(dinner)}\)
  - Wrap
    precondition: \(\text{(quiet)}\)
    effect: \(\text{(present)}\)
  - Carry
    precondition: 
    effect: \(\text{(and (not (garbage)) (not (cleanHands)))}\)
  - Dolly
    precondition: 
    effect: \(\text{(and (not (garbage)) (not (quiet)))}\)
Dinner Date Example
Dinner Date Example
Observation 1

Propositions monotonically increase
(always carried forward by no-ops)
Observation 2

Actions monotonically increase
Observation 3

Proposition mutex relationships monotonically decrease
Observation 4

Action mutex relationships monotonically decrease
Observation 5

Planning Graph 'levels off'.

- After some time $k$ all levels are identical.
- Because it’s a finite space, the set of literals never decreases and mutexes don’t reappear.
A valid plan is a sub-graph of the planning graph where:

- Actions at the same level don’t interfere.
- Each action’s preconditions are made true by the plan.
- Goals are made true by some action.
Searching For A Solution Plan

- Backward chain on the planning graph.
- Achieve goals level by level.
- At level k, pick a subset of non-mutex actions to achieve current goals. Their preconditions become the goals for k-1 level.
- Build goal subset by picking each goal and choosing an action to add. Use one already selected if possible. Do forward checking on remaining goals (backtrack if can’t pick non-mutex action).
Plan Graph Search

If goals are present & non-mutex:
- Choose action to achieve each goal.
- Add preconditions to next goal set.
Dinner Date Example

- **Initial Conditions:**
  
  (and (garbage) (cleanHands) (quiet))

- **Goal:**
  
  (and (dinner) (present) (not (garbage))

- **Actions:**
  
  - **Cook**
    
    precondition: (cleanHands)
    
    effect: (dinner)

  - **Wrap**
    
    precondition: (quiet)
    
    effect: (present)

  - **Carry**
    
    precondition:
    
    effect: (and (not (garbage)) (not (cleanHands)))

  - **Dolly**
    
    precondition:
    
    effect: (and (not (garbage)) (not (quiet))))
Dinner Date Example
Dinner Date Example

[Diagram showing relationships between actions such as "eats", "scores", etc., with arrows indicating sequence and outcomes.]
Dinner Date Example
Theorem 1:

- The size of the t-level PG and the time to create it are polynomial in:
  - $t =$ number of levels
  - $n =$ number of objects
  - $m =$ number of operators
  - $k =$ largest arity of actions and predicates

- Max nodes proposition level: $O((t + 1)mn^k)$
- Max nodes action level: $O(tmn^k)$
GraphPlan algorithm

1. Grow the planning graph (PG) until all goals are reachable and not mutex. (If PG levels off first, fail)
2. Search the PG for a valid plan...
3. If none found, add a level to the PG and try again.
Termination For Unsolvable Problems

- Graphplan records (memoizes) sets of unsolvable goals:
  - $U(i,t) =$ unsolvable goals at level $i$ after stage $t$.
- More efficient: early backtracking...
- Also provides necessary and sufficient conditions for termination:
  - Assume plan graph levels off at level $n$, stage $t > n$.
  - If $U(n, t-1) = U(n, t)$ then we know no plan exists.