Graphplan

José Luis Ambite*

[* based in part on slides by Jim Blythe and Dan Weld]
Basic idea

- Construct a graph that encodes constraints on possible plans
- Use this “planning graph” to constrain search for a valid plan:
  - If valid plan exists, it’s a subgraph of the planning graph
- Planning graph can be built for each problem in polynomial time
Problem handled by GraphPlan*

- Pure STRIPS operators:
  - conjunctive preconditions
  - no negated preconditions
  - no conditional effects
  - no universal effects
- Finds “shortest parallel plan”
- Sound, complete and will terminate with failure if there is no plan.

*Version in [Blum & Furst IJCAI 95, AIJ 97], later extended to handle all these restrictions [Koehler et al 97]
Planning graph

- Directed, leveled graph
  - 2 types of nodes:
    - Proposition: P
    - Action: A
  - 3 types of edges (between levels)
    - Precondition: P -> A
    - Add: A -> P
    - Delete: A -> 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
Constructing the planning graph

- 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)

$P_1 A_1 P_2 A_2 \ldots P_{n-1} A_{n-1} P_n$
Mutual Exclusion relations

- Two actions (or literals) are mutually exclusive (mutex) at some stage if no valid plan could contain both.

- Two actions are mutex if:
  - Interference: one clobbers 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

Competing Needs

Interference (prec-effect)

Inconsistent Support
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

garb

carry

dolly

cleanH

cook

quiet

wrap

¬garb

¬cleanH

¬quiet

dinner

present
Dinner Date example

garb

carry

dolly

cleanH

quiet

wrap

dinner

present

garb

carry

dolly

cleanH

quiet

wrap

dinner

present
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.
Valid plan

A valid plan is a planning graph where:

- Actions at the same level don’t interfere
- Each action’s preconditions are made true by the plan
- Goals are satisfied
GraphPlan algorithm

- Grow the planning graph (PG) until all goals are reachable and not mutex. (If PG levels off first, fail)
- Search the PG for a valid plan
- If non found, add a level to the PG and try again
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

garb

carry

dolly

\neg \text{garb}

\text{cleanH}

\neg \text{cleanH}

quiet

\neg \text{quiet}

wrap

dinner

present

24
Dinner Date example
Dinner Date example