Foundations of AI

11.a Recent Approaches to Planning

Wolfram Burgard & Bernhard Nebel
Contents

- Recent Approaches
- Optimal Planning: Planning Graphs
- Fast Planning: Heuristic Search
- Search Space Topology
New Approaches

• Since 1995, a number of new algorithmic approaches have been developed, which are much faster than the POP algorithm:
  – Planning based on planning graphs
  – Satisfiability based planning
  – BDD-based approaches (good for multi-state problems – which we ignore here)
  – Heuristic-search based planning
Planning Graphs

- Parallel execution of actions possible
- Assumption: Only positive preconditions (can be generated by an easy transformation)
- Describe possible developments in a layered graph (fact level/action level)
  - links from (positive) facts to preconditions
  - positive effects generate (positive) facts
  - negative effects are used to mark conflicts
- Extract plan by choosing only non-conflicting parts of graph
Generate a Planning Graph

- Add all applicable actions
- In order to propagate unchanged property $p$, use special action $noop_p$
- Generate all positive effects on next fact level
- Mark conflicts (between actions that cannot be executed in parallel)
- Expand planning graph as long as not all atoms in fact level
Extract a Plan

- Start at last fact level with goal facts
- Select minimal set of non-conflicting actions generating the goals
- Use preconditions of these actions as goals on next lower level
- Backtrack if no non-conflicting choice is possible
Conflict Information

• Two actions interfere (cannot be executed in parallel):
  – one action deletes or asserts the precondition of the other action
  – they have opposite effects on one atomic fact

• They are marked as such
  – and this information is propagated to prune the search early on
Mutex Pairs

- No pair of facts is `mutex` at fact level 0
- A pair of facts is `mutex` at fact level $i > 0$ if all ways of making them true involve actions that are `mutex` at the action level $i-1$
- A pair of actions is `mutex` at action level $i$ if
  - they interfere or
  - one precondition of one action is `mutex` to a precondition of the other action at fact level $i-1$

- `Mutex` pairs cannot be true/executed at the same time
- Note that we do not found all pairs that cannot be true/executed at the same time, but only the easy to spot pairs
Example Mutex Propagation

- Mark all interfering actions
- Then start at action level 1 and propagate conflicts
- Expand planning graph as long the goal nodes are not all generated or some are still mutex
Planning Graphs: General Method

- Expand planning graph until all goal atoms are in fact level and they are not mutex
- If not possible, terminate with failure
- Iterate:
  - Try to extract plan and terminate with plan if successful
  - Expand by another action and fact level
- Termination for unsolvable planning problems can be guaranteed – but is complex
Properties of the *Planning Graph* Approach

- Finds an **optimal solution** (for parallel plans)
- Terminates on **unsolvable** planning instances
- Is **much** faster than POP planning
- Has problems with **symmetries**:
  - Example: Transport $n$ objects from room A to room B using one gripper
  - If shortest plan has $k$ steps, it proves that there is no $k-1$ step plans
Planning as Satisfiability

• Based on planning graphs of depth $k$, one can generated a set of propositional CNF formulae
  – such that each model of these formulae correspond to a $k$-step plan
  – basically, one performs a different kind of search in the planning graph (middle out instead of regression search)
  – Can be considerable faster, sometimes …
Heuristic Search Planning

- Forward state-space search is often considered as too inefficient because of the high branching factor
- Why not use a heuristic estimator to guide the search?
- Could that be automatically derived from the representation of the planning instance?
  ➢ Yes, since the actions are not “black boxes” as in search!
Ignoring Negative Effects

• Ignore all **negative effects** (assuming again we have only positive preconditions)
  – *monotone planning*

• Example for the buyer’s domain:
  – Only *Go* and *Drop* have negative effects (perhaps also *Buy*)
  – Minimal length plan: <*Go(HWS), Buy(Drill), Go(SM), Buy(Bananas), Buy(Milk), Go(Home)*>
  – Ignoring negative effects: <*Go(HWS), Buy(Drill), Go(SM), Buy(Bananas), Buy(Milk)>*

• Usually plans are **shorter**
Monotone Planning

- Monotone planning is easy, i.e., can be solved in polynomial time:
  - While we have not made all goal atoms true:
    - Pick any action that
      - is applicable and
      - has not been applied yet
    - and apply it
    - If there is no such action, return failure
    - otherwise continue
Monotone Optimal Planning

- Finding the *shortest plan* is what we need to get an *admissible heuristic*, though!
- This is NP-hard, even if there are no preconditions!
  - *Minimum Set Cover*, which is NP-complete, can be reduced to this problem
Minimum Set Cover

- **Given:** A set $S$, a collection of subsets $C = \{C_1, \ldots, C_n\}$, $C_i \subseteq S$, and a natural number $k$.

- **Question:** Does there exist a subset of $C$ of size $k$ covering $S$?

- Problem is **NP-complete**

- and obviously a special case of the **monotone planning optimization problem**
Simplifying it Further ...

- Since the monotone planning heuristic is computationally too expensive, simplify it further:
  - compute heuristic distance for each atom (recursively) by assuming independence of sub-goals
  - solve the problem with any planner (i.e. the planning graph approach) and use this as an approximative solution
- both approaches may over-estimate, i.e., it is not an admissible heuristic any longer
The Fast-Forward (FF) System

- **Heuristic:** Solve the monotone planning problem resulting from the relaxation using a planning graph approach
- **Search:** Hill-climbing extended by breadth-first search on plateaus and with
- **Pruning:** Only those successors are considered that are part of a relaxed solution
- **Fall-back strategy:** complete best-first search
Relative Performance of FF

- FF performs very well on the planning benchmarks that are used for the planning competitions
- Examples:
  - Blocks world
  - Logistics
  - Freecell

- Performance data is from the International AI Planning System Competition 2000
Blocksworld (Performance)

CPU time

Solution size
Logistics (Domain)
Logistics (Performance)

CPU time

Solution size
Freecell (Domain)
Freecell (Performance)

CPU time

Solution size
Search Space Topology

• Why works the FF heuristic so well?
• Look for search space properties such as
  – local minima
  – size of plateaus
  – dead ends (detected & undetected)
• Estimate by
  – exploring small instances
  – sampling large instance
• Try to prove conjectures found this way
  ➢ Goes some way in understanding problem structure
Summary & Outlook

• Recent approaches to planning have boosted the efficiency of planning methods significantly
• Heuristic search planning appears to be one of the fastest (non-optimal) methods
• We are starting to understand the difficulty of a planning domain in terms of its search space topology
• Currently, search technology is transferred into the area of formal verification and synthesis (and vice versa)
• While it may still be long time before we can afford to use these techniques instead of domain-specific methods, the progress looks promising