Nondeterministic planning (June 6, 2005)

Execution graphs

Definition
- Example
- Bounded reachability

Plans with unbounded executions
- Algorithm idea
- Subprocedure prune
- Algorithm

Maintenance goals
- Definition
- Example
- Algorithm

Summary

Execution graphs

Definition

Let \((S, I, O, G, P)\) be a transition system with full observability and 
\(\pi: S \rightarrow O\) a mapping from states to operators.
Then the execution graph is \((S, E)\) where
1. states \(s \in S\) are the nodes of the graph,
2. \((s, s') \in E\) is an edge if and only if \(s' \in \text{img}_\pi(s)\),
3. the states \(s \in I\) are the initial nodes,
4. the states \(s \in G\) are the goal nodes,
5. nodes \(s \in S\) such that \((s, s') \in E\) for no \(s' \in S\) are terminal nodes.

Plan objectives

Bounded reachability

- The simplest objective for nondeterministic planning is the one we have used in last lectures: reach a goal state with certainty.
- With this objective the nondeterminism can also be understood as an opponent like in 2-player games or in \(n\)-player games in general.
Plan guarantees reaching a goal state no matter what the opponent does: plans are winning strategies.

Need for unbounded executions / looping

- The first planning algorithm finds plans that reach a goal state without visiting any state twice.
- This property guarantees that the length of executions is bounded by some constant (which is smaller than the number of states.)
- Some solvable problems are not solvable this way.
1. Action may fail to have any effect.
   Hit a coconut to break it.
2. Action may fail and take us away from the goals.
   Build a house of cards.
Consequences:
1. It is impossible to avoid visiting some states several times.
2. There is no finite upper bound on execution length.

Execution graphs

- To formalize more complicated planning problems and more complicated forms of plans we define execution graphs of a transition system + plan.
- An execution graph describes the possible states of execution and the transitions between them.
- For memoryless plans the execution states and states of the transition system coincide because the execution mechanism is simple.
- For more complex forms of plans (defined in later lectures when discussing planning without full observability) the execution states also include information that encodes memory from earlier states of execution.

Plan objectives

Bounded reachability

Definition

Let \((S, I, O, G, P)\) be a transition system with full observability and 
\(\pi: S \rightarrow O\) a mapping from states to operators.
Then \(\pi\) is a plan for bounded reachability if all maximal paths starting from an initial node have a finite length and end in a goal node.

This rules out infinite paths.

Need for plans with unbounded executions

Assumption

1. For any nondeterministic effect \(e_1 \cdots e_m\) the probability of every effect \(e_1, \ldots, e_m\) is \(> 0\).
2. For any \(s' \in \text{img}_\pi(s)\) the probability of reaching \(s'\) from \(s\) by \(\pi\) is \(> 0\).

This assumption guarantees that any path in the execution graph has a non-zero probability. This is not compatible with viewing nondeterminism as an opponent in a 2-player game: the opponent's strategy might rule out some of the choices \(e_1, \ldots, e_m\).
Plan objectives
Unbounded reachability

Definition
Let \((S, I, O, G, P)\) be a transition system with full observability and 
\(\pi : S \rightarrow O\) a mapping from states to operators. 
Then \(\pi\) is a plan for unbounded reachability if 
from every node to which there is a path from an initial node 
there is a path to a goal node that is a terminal node.

Looping
These plans may loop i.e., visit and revisit a state an unbounded number of times. 
These plans even allow infinite executions that do not reach a goal state 
but the probability of such executions under the assumption we made is 0.

Need for plans with unbounded executions
Example

Example (Build a house of cards)

- Initial state: all cards lie on the table.
- Goal state: house of cards is complete.
- At every construction step the house may collapse.

Algorithm idea

We give an algorithm that finds plans that may loop (unbounded reachability).

Subprocedure prune

The procedure prune finds a maximal set of states for which reaching goals with looping is possible.

Two nested loops.
1. Inner loop identifies sets \(S_i\) of states from which a goal state can be reached with \(j\) steps without leaving the current set of candidate good states \(W_i\). 
   Limit of \(S_0, S_1, \ldots\) will be \(W_\infty\).
2. Outer loop iterates through \(i = 0, 1, 2, \ldots\) and produces a decreasing sequence of candidate good state sets \(W_0, W_1, \ldots, W_i\) until \(W_\infty = W_\infty + 1\).

Lemma (Procedure prune)
Let \(S\) and \(G\) be sets of states and \(O\) a set of operators. 
Then prin\((S, O, G)\) terminates after a finite number of steps and returns 
\(W \subseteq S\) such that there is a plan \(\pi : W \rightarrow O\) such that 
1. for every \(x \in W\) there is an execution \(s_0, s_1, \ldots, s_n\) of \(\pi\) with \(n \geq 1\) such that \(s = s_0\) and \(s_n \in G\), 
2. \(img_{\pi}(\{s\}) \subseteq U \cup G\) for every \(s \in W\), and 
3. for every \(x \in S\) and function \(x' : s \rightarrow O\) there is an execution \(s_0, s_1, \ldots, s_n\) of \(x'\) such that \(s = s_0\) and there is no \(m \geq n\) and execution \(s_m, \ldots, s_n\) such that \(s_m \in G\).
The planning algorithm

```plaintext
1: PROCEDURE FOplanLOOPS(I,O,G)
2:   S := the set of all states;
3:   L := O; prune(S,O,G);
4:   IF I \not\subseteq L THEN RETURN V;
5:   D_0 := G;
6:   i := 1;
7:   REPEAT (* Compute weak backward distances *)
8:     D_i := D_{i-1} \cup \bigcup_{s \in O} \text{preimg}_i(D_{i-1}) \cap \text{spreimg}_i(L);
9:     i := i + 1;
10:    UNTIL D_i = D_{i-1};
11:    FOR EACH s \in D_i \setminus G DO
12:       d := number such that s \in D_d \setminus D_{d-1};
13:       \pi(s) := o such that \text{img}_d(s) \subseteq L and \text{img}_d(s) \cap D_{d-1} \neq \emptyset;
14:    END DO
```

Algorithm for maintenance goals

We can infer rules backwards starting from the death condition.
1. If in desert and thirst = 2 must go to river.
2. If in desert and hunger = 2 must go to pasture.
3. If on pasture and thirst = 1 must go to desert.
4. If at river and hunger = 1 must go to desert.
5. ...

If the above rules conflict, the animal will die.

There is only one plan: go to pasture, go to desert, go to river, go to desert, ...

Plan objectives

Definition

Let \((S,I,O,G,P)\) be a transition system with full observability and \(\pi : S \rightarrow O\) a mapping from states to operators.

Then \(\pi\) is a plan for maintenance if every node in the execution graph to which there is a path from an initial node is a goal node that has a successor node.

The execution graph does not have terminal nodes.

Summary of the algorithm idea

Repeatedly eliminate from consideration those states that in 1 or more steps unavoidably lead to a non-goal state.

Maintainance goals

Example

The state of an animal is determined by three state variables:

- Hunger (0,1,2), thirst (0,1,2) and location (river, pasture, desert).
- There is also a special state called death.
- Thirst grows when not at river; at river it is 0.
- Hunger grows when not on pasture; on pasture it is 0.
- If hunger or thirst exceeds 2, the animal dies.
- The goal of the animal is to not die.

Complexity of the planning algorithm

▸ The procedure prune runs in polynomial time in the number of states because the number of iterations of each loop is at most \(n\) – hence there are \(O(n^2)\) iterations – and computation on each iteration takes polynomial time in the number of states.

▸ Finding conditional plans for full observability under the bounded and unbounded reachability objectives is in the complexity class EXPTIME.

▸ Lecture notes contain proofs showing that the planning problems are also EXPTIME-hard.
**Algorithm for maintenance goals**

**Definition**

1: PROEDURE FOplanMAINTENANCE(I,O,G)
2: \( G' \succ G \);
3: REPEAT
4: \( G' \leftarrow G' \);
5: \( G' \leftarrow \bigcup_{o \in O} \text{spreimg}_o(G') \cap G \);
6: UNTIL \( G' = G' \);
7: IF \( I \not\subseteq G' \) RETURN \( \emptyset \);
8: FOR EACH \( s \in G' \) DO
9: assign \( \pi(s) \leftarrow o \) such that \( \text{img}_o(s) \subseteq G' \);
10: END DO
11: RETURN \( \pi \);

**Summary**

There are several possible objectives a plan can fulfill.

- We have considered **bounded reachability**, **unbounded reachability**, and **maintenance**.
  - The executions are respectively of bounded finite length, unbounded finite length, and infinite.
  - These objectives have all been formalized in terms of the properties of execution graphs.
- We have presented dynamic-programming type (backward search) algorithms for all three planning problems.
- All three algorithms can be implemented by using binary decision diagrams BDDs as a data structure for state sets.