AI Planning

6. Progression and Regression

Should We Go Forward or Backward?

Álvaro Torralba, Cosmina Croitoru

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Winter Term 2018/2019

Thanks to Prof. Jörg Hoffmann for slide sources

Agenda

What is "Search"?

Here, we mean classical search: (We'll usually omit the "classical".)

- A search space specifies a start search state, a target-identification function, and a successor search-state function.
- Find a path of search-state transitions from the start state to a state identified as a target.

→ Search state ≠ world state! E.g. regression, cf. later.

→ Classical search is the same as in AI'16 Chapters 4 and 5. But, there, we didn't worry about the search space and just assumed forward search.

Planning as Classical Search: Choices

There are three independent choices to make:

Choice 1: Search Space

- Progression.
  → Search forward from initial state to goal.
  → Search states = world states.
- Regression.
  → Search backward from goal to initial state.
  → Search states = sub-goals we would need to achieve.

→ **This Chapter**

Planning as Classical Search: Choices, ctd.

There are three independent choices to make:

Choice 2: Search Algorithm

- Blind search.
  $\rightarrow$ Depth-first, breadth-first, iterative depth-first, ...
- Heuristic search (systematic). Aka informed search (systematic).
  $\rightarrow$ A*, IDA*, ...
- Heuristic search (local). Aka informed search (local).
  $\rightarrow$ Hill-climbing, simulated annealing, beam search, ...

$\rightarrow$ **Next Chapter**

Planning as Classical Search: Choices, ctd.

There are three independent choices to make:

Choice 3: Search Control

- Heuristic function. (For heuristic searches.)
  → Critical-path heuristics, delete-relaxation heuristics, abstraction heuristics, landmarks heuristics, ...
- Pruning techniques.
  → Helpful actions pruning, symmetry elimination, dominance pruning, partial-order reduction.

→ **Chapters 8–18**

Planning as Classical Search: Example Systems

One of the best satisficing planners:

FF [Hoffmann and Nebel (2001)] → **Chapter 9**

1. Search space: Progression.
2. Search algorithm: Enforced hill-climbing (informed local).
3. Search control: Delete-relaxed plan heuristic h^{FF} (inadmissible), helpful actions pruning (incomplete).

One of the best optimal planners:

Fast Downward + $h^{\mathrm{M\&S}}$ [Nissim *et al.* (2011)] → **Chapter 13**

1. Search space: Progression.
2. Search algorithm: A^* (informed systematic).
3. Search control: Merge-and-shrink abstractions (admissible).

→ Fast Downward is based on FDR, and has become the main implementation basis for heuristic search planning. Its current version implements a great variety of techniques. Our **Programming Exercises** are based on it, too.

What is a "Search Space"?

Search Space for Classical Search

A (classical) search space is defined by the following three operations:

- start(): Generate the start (search) state.
- is-target(s): Test whether a given search state is a target state.
- succ(s): Generates the successor states (a, s') of search state s, along with the actions through which they are reached.

$\rightarrow$ Search state $\neq$ world state! E.g. regression, cf. later.

$\rightarrow$ Progression and regression instantiate this template in different ways.

Our Agenda for This Chapter

2. **Progression:** The (very) simple definition.
3. **Regression:** The less simple definition, for STRIPS vs. FDR and the differences between these two.
4. **Pro and Contra:** So which one should we use?

Progression

. . . is another word for Forward Search:

Search Space for Progression

Let $\Pi = (P, A, c, I, G)$ be a STRIPS planning task. The progression search space of Π is given by:

- start$() = I$
- is-target$(s) = \begin{cases} \textbf{true} & \text{if } G \subseteq s \\ \textbf{false} & \text{otherwise} \end{cases}$
- succ$(s) = \{(a, s') \mid \Theta_\Pi \text{ has the transition } s \xrightarrow{a} s'\}$

The same definition applies to FDR tasks $\Pi = (V, A, c, I, G)$.

$\rightarrow$ Start from initial state, and apply actions until a goal state is reached.

$\rightarrow$ Search space = state space $\Rightarrow$ called state space search.

Regression

. . . is another word for Backward Search:

Search Space for Regression

Let $\Pi = (P, A, c, I, G)$ be a STRIPS planning task. The regression search space of Π is given by:

- start$() = G$
- is-target$(g) = \begin{cases} \textbf{true} & \text{if } g \subseteq I \\ \textbf{false} & \text{otherwise} \end{cases}$
- succ$(g) = \{(a, g') \mid g' = regr(g, a)\}$

The *same definition* applies to FDR tasks $\Pi = (V, A, c, I, G)$.

$\rightarrow$ Start at goal, and regress over actions to produce subgoals, until a subgoal is contained in the initial state.

Condition (*) required: If $g' = regr(g, a)$, then for all s' with $s' \models g'$, we have $s'[\![a]\!] = s$ where $s \models g$.

Regressing Subgoals Over Actions: FDR

$\rightarrow$ Intuition: a can make the conjunctive subgoal g true if (i) it achieves part of g; (ii) it contradicts none of g; and (iii) the new subgoal we would have to solve is not self-contradictory.

Definition (FDR Regression). *Let (V, A, c, I, G) be an FDR planning task, g be a partial variable assignment, and $a \in A$.*
We say that g is regressable over a if

(i) $\mathit{eff}_a \cap g \neq \emptyset$;

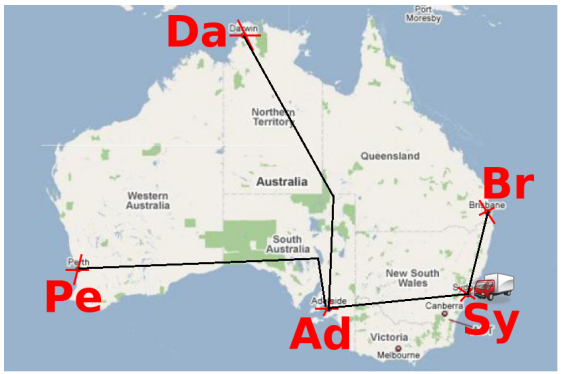
(ii) *there is no $v \in V$ s.t. $v \in V[\mathit{eff}_a] \cap V[g]$ and $\mathit{eff}_a(v) \neq g(v)$; and*

(iii) *there is no $v \in V$ s.t. $v \notin V[\mathit{eff}_a]$, $v \in V[\mathit{pre}_a] \cap V[g]$, and $\mathit{pre}_a(v) \neq g(v)$.*

In that case, the regression of g over a is $\mathit{regr}(g, a) = (g \setminus \mathit{eff}_a) \cup \mathit{pre}_a$; else $\mathit{regr}(g, a)$ is undefined, written $\mathit{regr}(g, a) = \bot$.

Proposition. *This definition of regr satisfies condition (*) on slide 14.*

Regression Example: "TSP" in FDR



- Variables V: $at : \{Sy, Ad, Br, Pe, Da\}$;
  $v(x) : \{T, F\}$ for $x \in \{Sy, Ad, Br, Pe, Da\}$.
- Actions $a \in A$: $drive(x, y)$ where x, y have a road.
- Initial state I: $at = Sy, v(Sy) = T, v(x) = F$ for $x \neq Sy$.
- Goal G: $at = Sy, v(x) = T$ for all x.

Regressing Subgoals Over Actions: STRIPS

Definition (STRIPS Regression). *Let (P, A, c, I, G) be a STRIPS planning task, $g \subseteq P$, and $a \in A$. We say that g is regressable over a if*

(i) $add_a \cap g \neq \emptyset$; *and*

(ii) $del_a \cap g = \emptyset$.

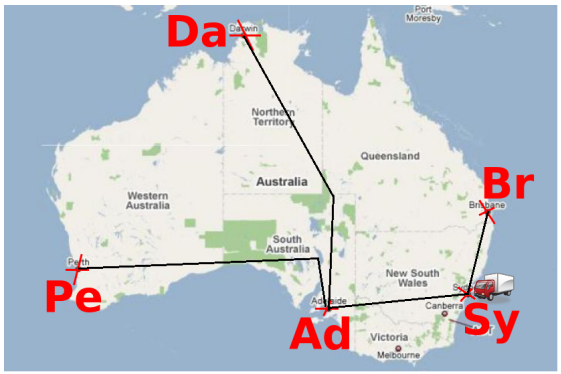
In that case, the regression of g over a is $regr(g, a) = (g \setminus add_a) \cup pre_a$; else $regr(g, a)$ is undefined, written $regr(g, a) = \bot$..

Proposition. *This definition of $regr$ satisfies condition (*) on slide 14.*

Note the difference to FDR:

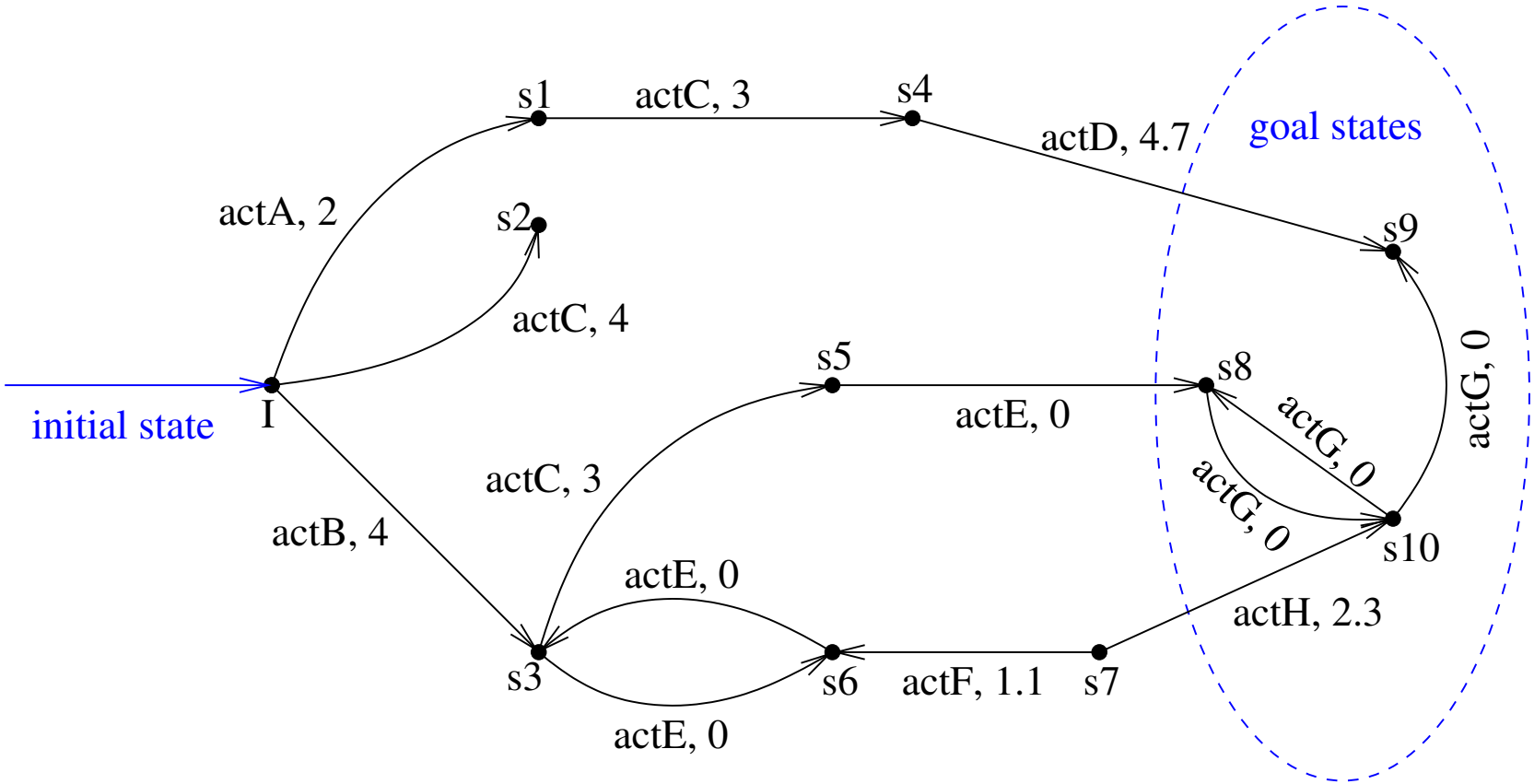
- In (ii), instead of "contradicting variable values" we only look at the action's immediate deletes.
  $\rightarrow$ This is weaker because we fail to see, e.g., that different truck positions yield contradictions as well (see next slide).
- (iii) here is missing completely because in STRIPS there is no possibility for a subgoal to be "self-contradictory".
  $\rightarrow$ This is weaker because we fail to see, e.g., that subgoals requiring several different truck positions are self-contradictory (see next slide).

Regression Example: "TSP" in STRIPS



- Propositions P: $at(x)$ for $x \in \{Sy, Ad, Br, Pe, Ad\}$; $v(x)$ for $x \in \{Sy, Ad, Br, Pe, Ad\}$.
- Actions $a \in A$: $drive(x, y)$ where x, y have a road.
- Initial state I: $at(Sy), v(Sy)$.
- Goal G: $at(Sy), v(x)$ for all x.

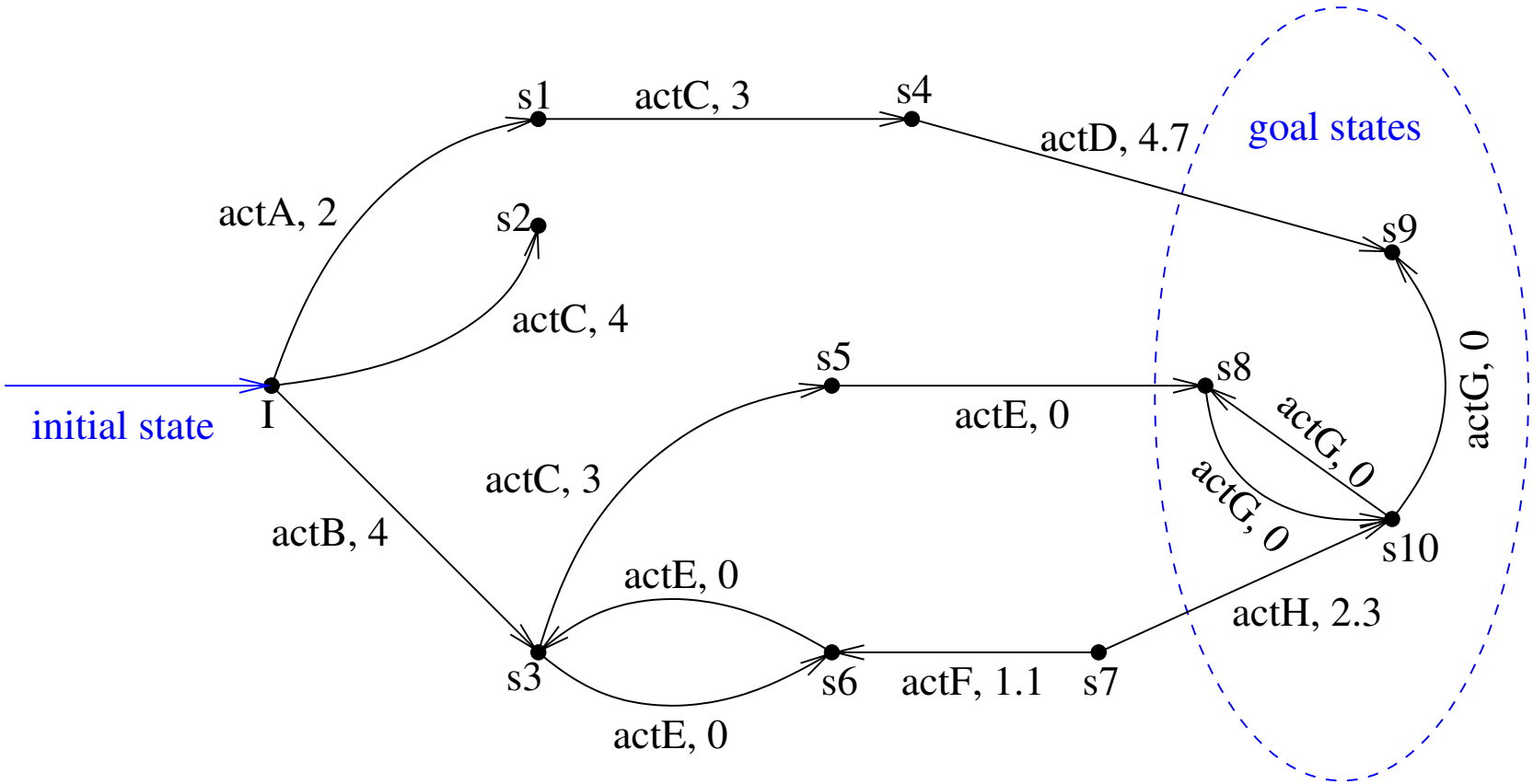
Reachable and Solvable States in Progression



- Does progression explore the state "at bottom, 2nd from right"?
- Does progression explore the state "near top, 2nd from left"?

Progression and Relevance

Reachable and Solvable States in Regression



- Does regression explore the state "near top, 2nd from left"?
- Does regression explore the state "at bottom, 2nd from right"?

Regression and Reachability

Questionnaire

→ Regarding fruitless parts of the forward search space:

Question!

In which of these STRIPS domains can the state space contain unsolvable (dead-end) but reachable states?

(A): "Logistics"
(B): "TSP" in Australia
(C): 15-Puzzle
(D): FreeCell

Questionnaire

→ Regarding fruitless parts of the backward search space: (recall that the "state space" contains all subsets of facts)

Question!

In which of these STRIPS domains can the state space contain solvable (non-dead-end) but unreachable states?

(A): "TSP" in Australia

(B): "Logistics"

(C): 15-Puzzle

(D): FreeCell

So Which One Should We Use?

Summary

- Search is required in planning because the problem is computationally hard.
- We consider classical search, that finds a path through a search space implicitly defined in terms of the operations start(), is-target(s), and succ(s).
- Progression is forward search from the initial state to the goal, in the state space. To be effective, it needs to be informed about relevance.
- Regression is backward search from the goal to the initial state, in a space of subgoals that correspond to sets of world states. To be effective, it needs to be informed about reachability.
- FDR regression is a lot more effective than STRIPS regression, because its search space contains less unreachable states.

Remarks

References I

Jörg Hoffmann and Bernhard Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

Raz Nissim, Jörg Hoffmann, and Malte Helmert. Computing perfect heuristics in polynomial time: On bisimulation and merge-and-shrink abstraction in optimal planning. In Toby Walsh, editor, *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI'11)*, pages 1983–1990. AAAI Press/IJCAI, 2011.