Planning and Optimization

B1. Planning as Search

Malte Helmert and Gabriele Röger

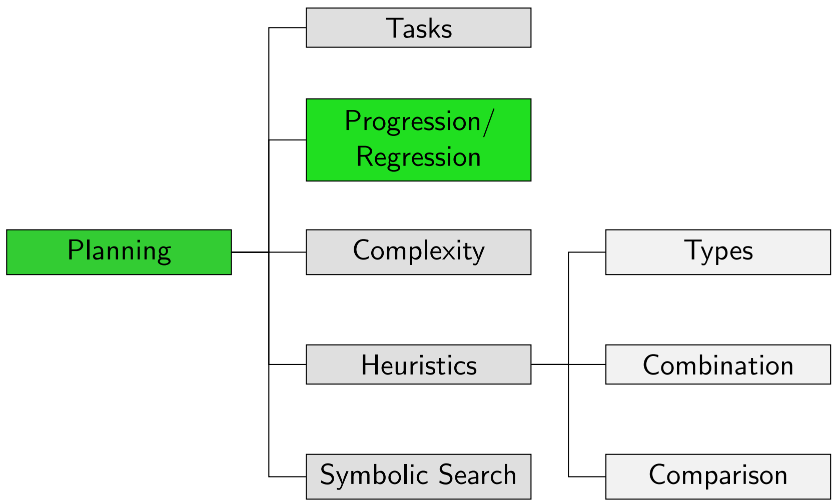
Universität Basel

October 11, 2017

Planning and Optimization

October 11, 2017 — B1. Planning as Search

Content of this Course



B1.1 Introduction

What Do We Mean by Search?

- Search is a very generic term.

⇝ Every algorithm that tries out various alternatives can be said to "search" in some way.

- Here, we mean classical state-space search algorithms.
  - Search nodes are expanded to generate successor nodes.
  - Examples: breadth-first search, greedy best-first search, weighted A^*, A^*, ...
- To be brief, we just say search in the following (not "classical state-space search").

Planning as Search

- search: one of the big success stories of AI
- most state-of-the-art planning systems are based on classical heuristic search algorithms (we will see some other algorithms later, though)
- majority of course focuses on heuristics for planning as search

Reminder: State-Space Search

Need to Catch Up?

- We assume prior knowledge of basic search algorithms:
  - uninformed vs. informed
  - satisficing vs. optimal
- If you are not familiar with them, we recommend Chapters 5–19 of the Foundations of Artificial Intelligence course at `http://cs.unibas.ch/fs2017/foundations-of-artificial-intelligence/`.

Reminder: Interface for Heuristic Search Algorithms

Abstract Interface Needed for Heuristic Search Algorithms

- init() ⇝ returns initial state
- is_goal(s) ⇝ tests if s is a goal state
- succ(s) ⇝ returns all pairs $\langle a, s' \rangle$ with $s \xrightarrow{a} s'$
- cost(a) ⇝ returns cost of action a
- h(s) ⇝ returns heuristic value for state s

⇝ Foundations of Artificial Intelligence course, Chapters 6 and 13

State Space vs. Search Space

- Planning tasks induce transition systems (a.k.a. state spaces) with an initial state, labeled transitions and goal states.
- State-space search searches state spaces with an initial state, a successor function and goal states.

⇝ looks like an obvious correspondence

- However, in planning as search, the state space being searched can be different from the state space of the planning task.
- When we need to make a distinction, we speak of
  - the state space of the planning task whose states are called world states vs.
  - the search space of the search algorithm whose states are called search states.

B1.2 Search-based Planning Algorithm Classification

Satisficing or Optimal Planning?

Must carefully distinguish two different problems:

- satisficing planning: any solution is OK (but cheaper solutions usually preferred)
- optimal planning: plans must have minimum cost

Both are often solved by search, but:

- details are very different
- almost no overlap between good techniques for satisficing planning and good techniques for optimal planning
- many tasks that are trivial to solve for satisficing planners are impossibly hard for optimal planners

Planning as Search

How to apply search to planning? ⇝ many choices to make!

Choice 1: Search Direction

- progression: forward from initial state to goal
- regression: backward from goal states to initial state
- bidirectional search

Planning as Search

How to apply search to planning? ⇝ many choices to make!

Choice 2: Search Space Representation

- search states are identical to world states
  ⇝ explicit-state search
- search states correspond to sets of world states
  ⇝ symbolic search

Planning as Search

How to apply search to planning? ⇝ many choices to make!

Choice 3: Search Algorithm

- uninformed search:
  depth-first, breadth-first, iterative depth-first, ...
- heuristic search (systematic):
  greedy best-first, A^*, weighted A^*, IDA^*, ...
- heuristic search (local):
  hill-climbing, simulated annealing, beam search, ...

Planning as Search

How to apply search to planning? ⇝ many choices to make!

Choice 4: Search Control

- heuristics for informed search algorithms
- pruning techniques: invariants, symmetry elimination, partial-order reduction, helpful actions pruning, ...

Search-based Satisficing Planners: Example (1)

FF (Hoffmann & Nebel, 2001)

- search direction: forward search
- search space representation: explicit-state
- search algorithm: enforced hill-climbing (informed local)
- heuristic: FF heuristic (inadmissible)
- other aspects: helpful action pruning; goal agenda manager

⇝ breakthrough for heuristic search planning;
winner of IPC 2000

Search-based Satisficing Planners: Example (2)

LAMA (Richter & Westphal, 2008)

- search direction: forward search
- search space representation: explicit-state
- search algorithm: restarting Weighted A*
- heuristic: FF heuristic and landmark heuristic (inadmissible)
- other aspects: preferred operators; deferred heuristic evaluation; multi-queue search

⇝ still one of the leading satisficing planners;
winner of IPC 2008 and IPC 2011 (satisficing tracks)

Search-based Optimal Planners: Example

Fast Downward Stone Soup (Helmert et al., 2011)

- search direction: forward search
- search space representation: explicit-state
- search algorithm: A^* (informed systematic)
- heuristic: LM-cut; merge-and-shrink; landmarks; blind (admissible)
- other aspects: sequential portfolio algorithm

⇝ winner of IPC 2011 (optimal track)

Our Plan for the Following Weeks

- progression search ⇝ this chapter
- regression search ⇝ Chapters B2–B4
- heuristics for classical planning ⇝ Parts C–F

B1.3 Progression

Planning by Forward Search: Progression

Progression: Computing the successor state $s[\![o]\!]$ of a state s with respect to an operator o.

Progression planners find solutions by forward search:

- start from initial state
- iteratively pick a previously generated state and **progress it** through an operator, generating a new state
- solution found when a goal state generated

pro: very easy and efficient to implement

Search Space for Progression

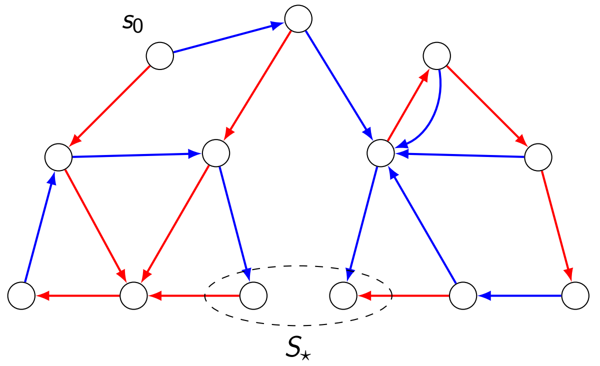
Search Space for Progression

search space for progression in a planning task $\Pi = \langle V, I, O, \gamma \rangle$ (search states are world states s of Π; actions of search space are operators $o \in O$)

- init() $\rightsquigarrow$ returns I
- is_goal(s) $\rightsquigarrow$ tests if $s \models \gamma$
- succ(s) $\rightsquigarrow$ returns all pairs $\langle o, s[\![o]\!] \rangle$ where $o \in O$ and o is applicable in s
- cost(o) $\rightsquigarrow$ returns $cost(o)$ as defined in Π
- h(s) $\rightsquigarrow$ estimates cost from s to γ ($\rightsquigarrow$ Parts C–F)

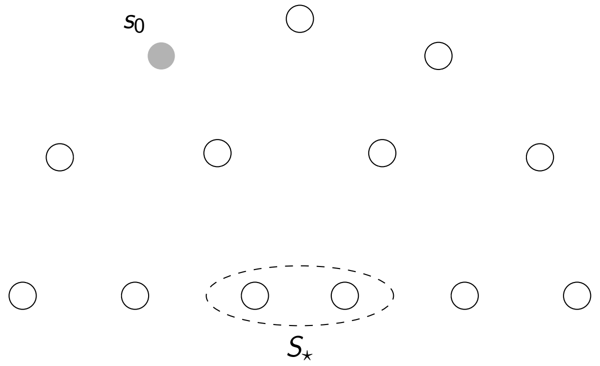
Progression Example

Example of a progression search



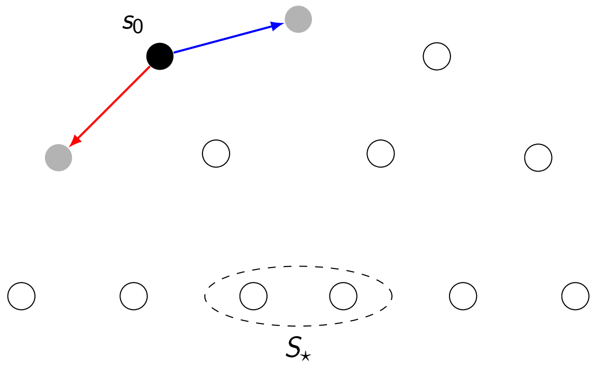
Progression Example

Example of a progression search



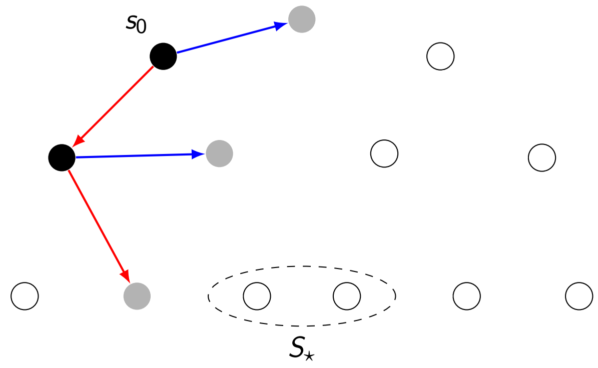
Progression Example

Example of a progression search



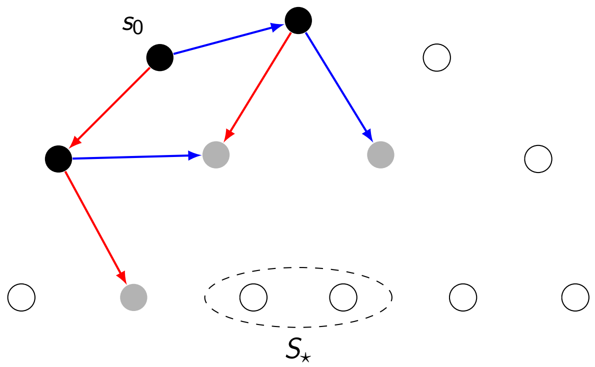
Progression Example

Example of a progression search



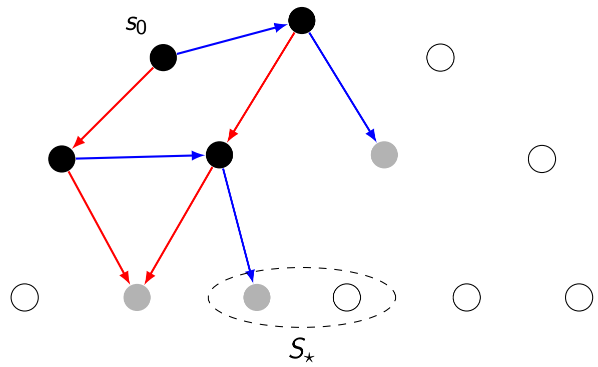
Progression Example

Example of a progression search



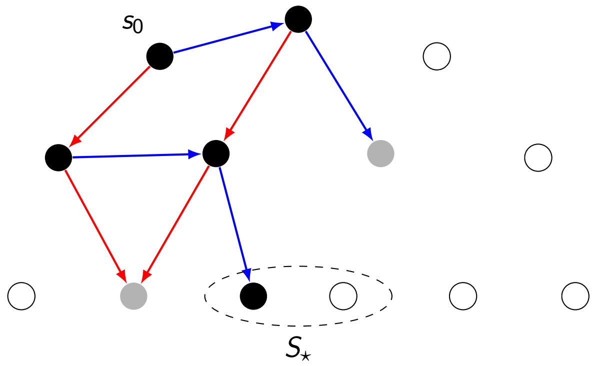
Progression Example

Example of a progression search



Progression Example

Example of a progression search



B1.4 Summary

Summary

- (Classical) search is a very important planning approach.
- Search-based planning algorithms differ along many dimensions, including
  - search direction (forward, backward)
  - what each search state represents (a world state, a set of world states)
- Progression search proceeds forward from the initial state.
- In progression search, the search space is identical to the state space of the planning task.