

Planning and Optimization

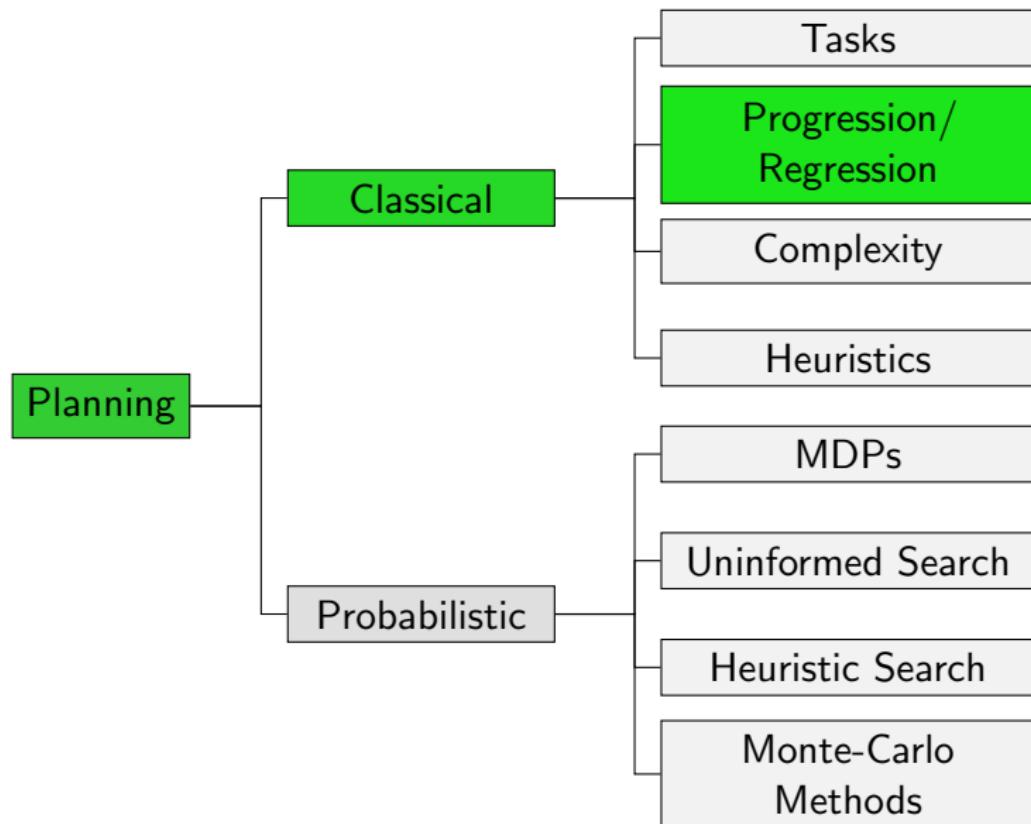
B2. Progression and Regression Search

Gabriele Röger and Thomas Keller

Universität Basel

October 10, 2018

Content of this Course



Introduction
●○○

Progression
○○○○

Regression
○○○○○○

Regression for STRIPS Tasks
○○○○

Summary
○○

Introduction

Search Direction

Search direction

- one dimension for classifying search algorithms
- **forward** search from initial state to goal based on **progression**
- **backward** search from goal to initial state based on **regression**
- **bidirectional** search

Search Direction

Search direction

- one dimension for classifying search algorithms
- **forward** search from initial state to goal based on **progression**
- **backward** search from goal to initial state based on **regression**
- **bidirectional** search

Today we will look into progression and regression planning.

Reminder: Interface for Heuristic Search Algorithms

Abstract Interface Needed for Heuristic Search Algorithms

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

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

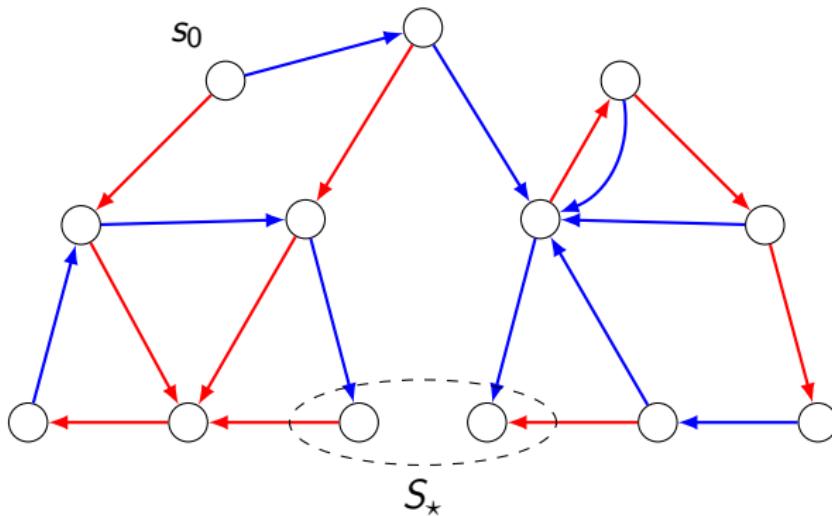
Search Space for Progression

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)

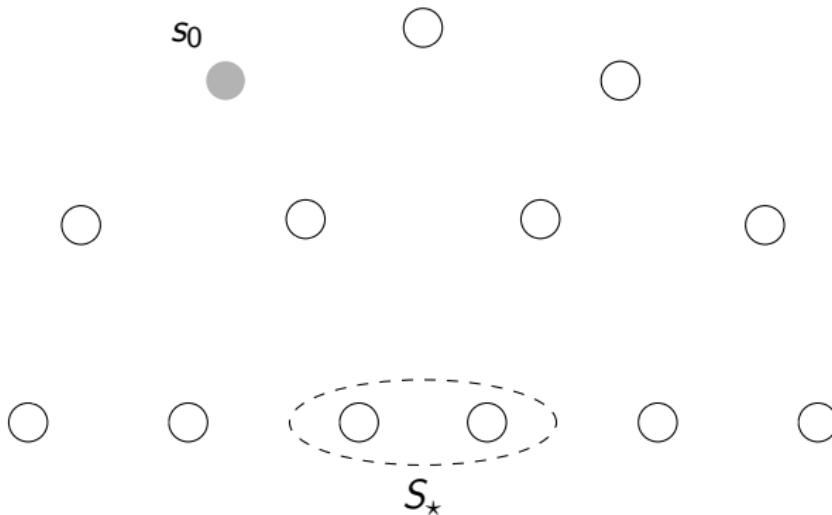
Progression Planning Example

Example of a progression search



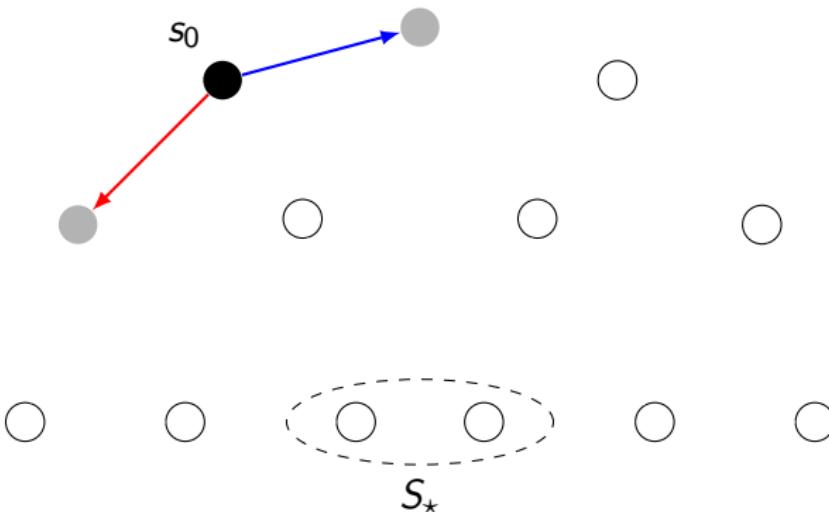
Progression Planning Example

Example of a progression search



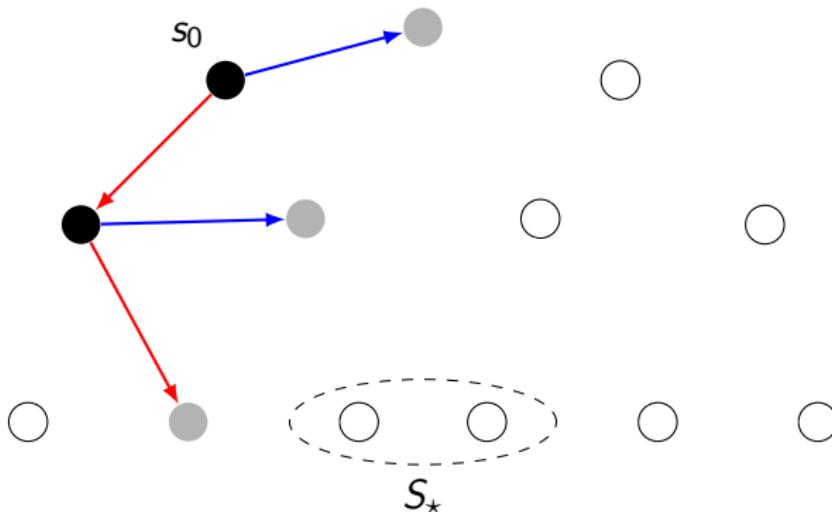
Progression Planning Example

Example of a progression search



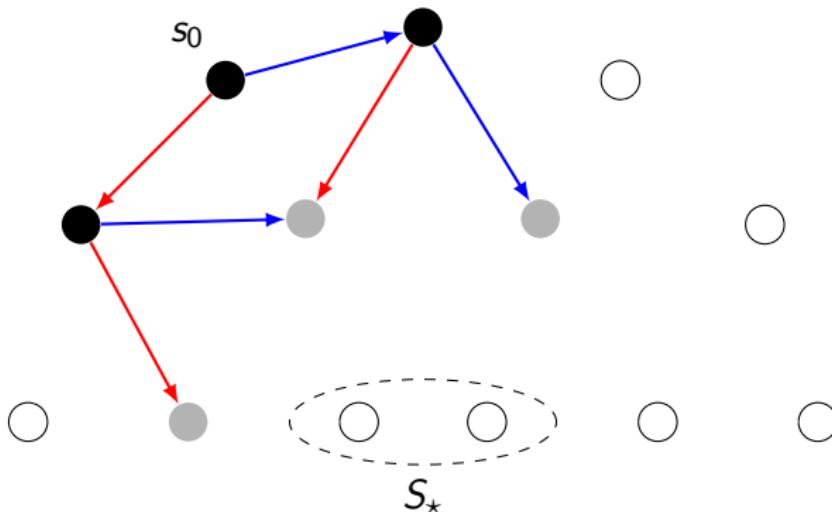
Progression Planning Example

Example of a progression search



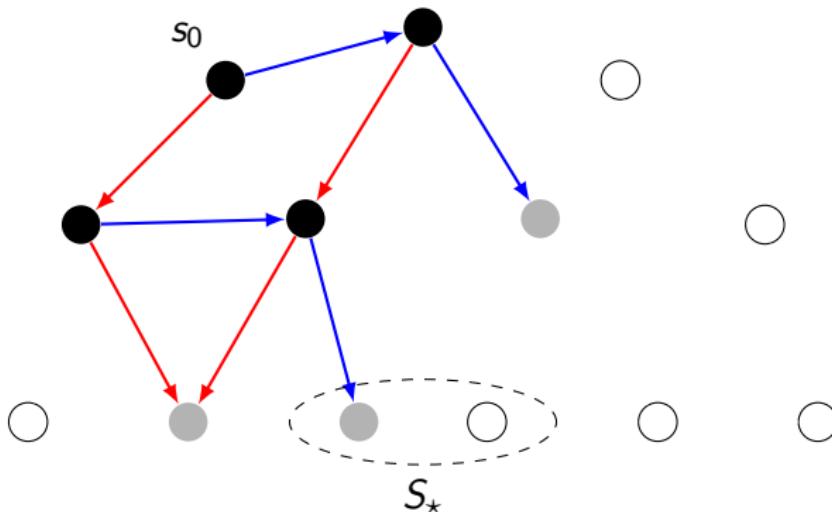
Progression Planning Example

Example of a progression search



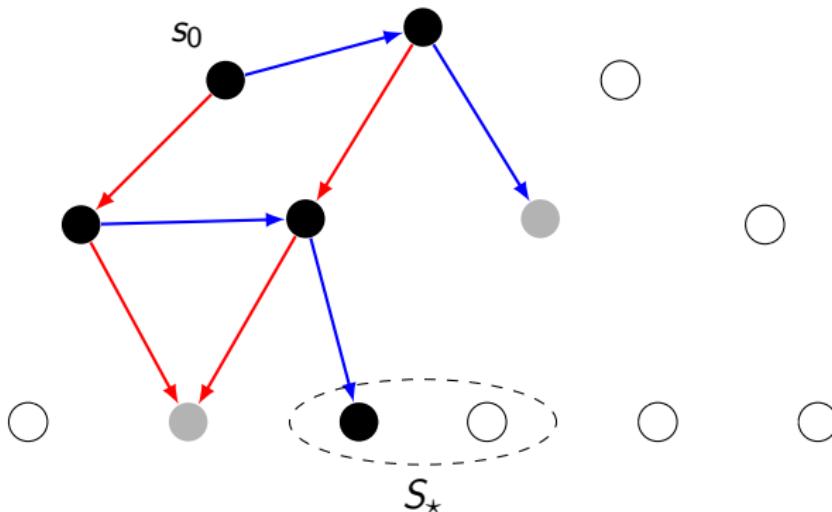
Progression Planning Example

Example of a progression search



Progression Planning Example

Example of a progression search



Introduction
ooo

Progression
oooo

Regression
●ooooo

Regression for STRIPS Tasks
oooo

Summary
oo

Regression

Forward Search vs. Backward Search

Searching planning tasks in forward vs. backward direction is **not symmetric**:

- forward search starts from a **single** initial state;
backward search starts from a **set** of goal states
- when applying an operator o in a state s in forward direction,
there is a **unique successor state s'** ;
if we just applied operator o and ended up in state s' ,
there can be **several possible predecessor states s**

~~> in most natural representation for backward search in planning,
each search state corresponds to a **set of world states**

Planning by Backward Search: Regression

Regression: Computing the possible predecessor states $regr(S', o)$ of a set of states S' ("subgoal") given the last operator o that was applied.

~ formal definition in next chapter

Regression planners find solutions by backward search:

- start from set of goal states
- iteratively pick a previously generated subgoal (state set) and regress it through an operator, generating a new subgoal
- solution found when a generated subgoal includes initial state

pro: can handle many states simultaneously

con: basic operations complicated and expensive

Search Space Representation in Regression Planners

identify state sets with **logical formulas** (again):

- each **search state** corresponds to a **set of world states** (“subgoal”)
- each search state is represented by a **logical formula**:
 φ represents $\{s \in S \mid s \models \varphi\}$
- many basic search operations like detecting duplicates are NP-complete or coNP-complete

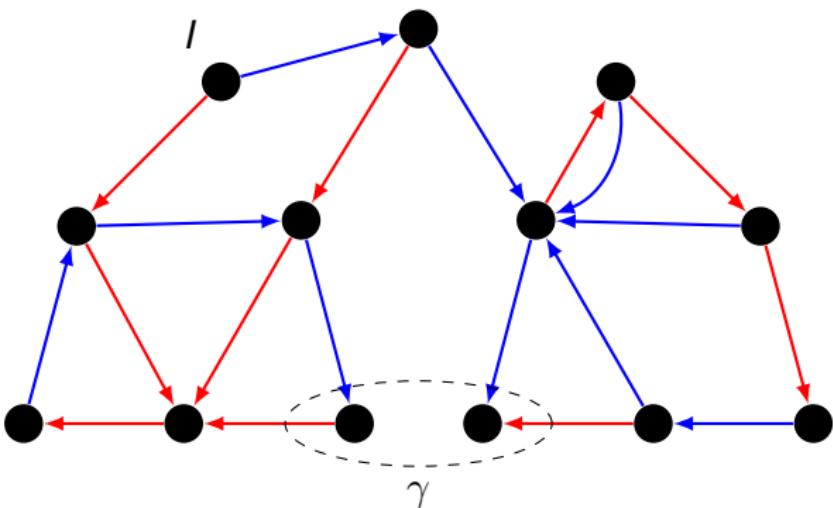
Search Space for Regression

Search Space for Regression

search space for regression in a planning task $\Pi = \langle V, I, O, \gamma \rangle$
(search states are formulas φ describing sets of world states;
actions of search space are operators $o \in O$)

- **init()** \rightsquigarrow returns γ
- **is_goal(φ)** \rightsquigarrow tests if $I \models \varphi$
- **succ(φ)** \rightsquigarrow returns all pairs $\langle o, \text{regr}(\varphi, o) \rangle$
where $o \in O$ and $\text{regr}(\varphi, o)$ is defined
- **cost(o)** \rightsquigarrow returns $\text{cost}(o)$ as defined in Π
- **h(φ)** \rightsquigarrow estimates cost from I to φ (\rightsquigarrow Parts C–F)

Regression Planning Example (Depth-first Search)



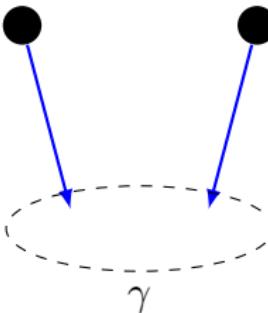
Regression Planning Example (Depth-first Search)

 γ 

Regression Planning Example (Depth-first Search)

$$\varphi_1 = \text{regr}(\gamma, \longrightarrow)$$

$$\varphi_1 \longrightarrow \gamma$$

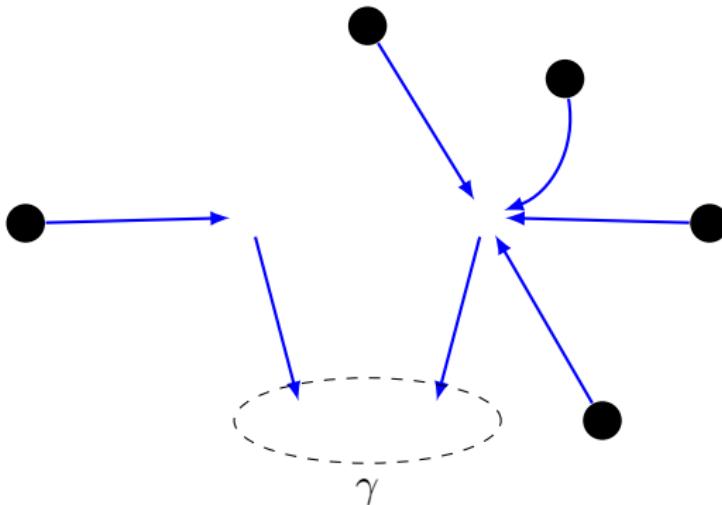


Regression Planning Example (Depth-first Search)

$$\varphi_1 = \text{regr}(\gamma, \longrightarrow)$$

$$\varphi_2 = \text{regr}(\varphi_1, \longrightarrow)$$

$$\varphi_2 \longrightarrow \varphi_1 \longrightarrow \gamma$$



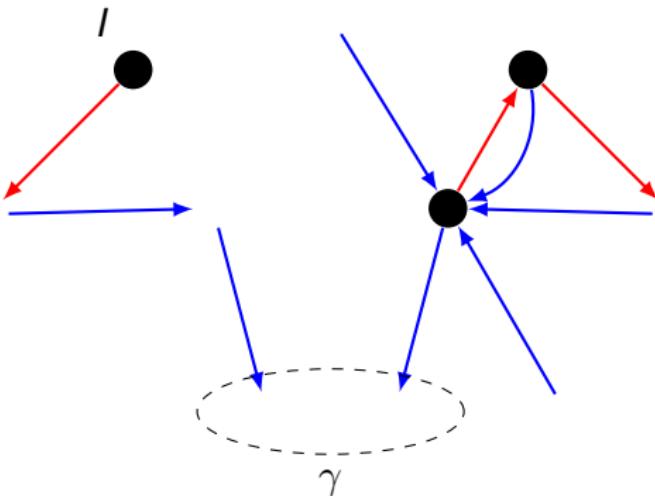
Regression Planning Example (Depth-first Search)

$$\varphi_1 = \text{regr}(\gamma, \xrightarrow{\quad})$$

$$\varphi_3 \xrightarrow{\quad} \varphi_2 \xrightarrow{\quad} \varphi_1 \xrightarrow{\quad} \gamma$$

$$\varphi_2 = \text{regr}(\varphi_1, \xrightarrow{\quad})$$

$$\varphi_3 = \text{regr}(\varphi_2, \xrightarrow{\quad}), I \models \varphi_3$$



Regression for STRIPS Tasks

Regression for STRIPS Planning Tasks

Regression for STRIPS planning tasks is much simpler than the general case:

- Consider subgoal φ that is conjunction of atoms $a_1 \wedge \dots \wedge a_n$ (e.g., the original goal γ of the planning task).
- **First step:** Choose an operator o that deletes no a_i .
- **Second step:** Remove any atoms added by o from φ .
- **Third step:** Conjoin $pre(o)$ to φ .

↝ Outcome of this is regression of φ w.r.t. o .
It is again a conjunction of atoms.

optimization: only consider operators adding at least one a_i

STRIPS Regression

Definition (STRIPS Regression)

Let $\varphi = \varphi_1 \wedge \dots \wedge \varphi_n$ be a conjunction of atoms, and let o be a STRIPS operator which adds the atoms a_1, \dots, a_k and deletes the atoms d_1, \dots, d_l .

The **STRIPS regression** of φ with respect to o is

$$sregr(\varphi, o) := \begin{cases} \perp & \text{if } \varphi_i = d_j \text{ for some } i, j \\ pre(o) \wedge \bigwedge (\{\varphi_1, \dots, \varphi_n\} \setminus \{a_1, \dots, a_k\}) & \text{else} \end{cases}$$

Note: $sregr(\varphi, o)$ is again a conjunction of atoms, or \perp .

Does this Capture the Idea of Regression?

For our definition to capture the concept of **regression**, it should satisfy the following property:

Regression Property

For all sets of states described by a conjunction of atoms φ , all states s and all STRIPS operators o ,

$$s \models s\text{regr}(\varphi, o) \quad \text{iff} \quad s\llbracket o \rrbracket \models \varphi.$$

This is indeed true. We do not prove it now because we prove this property for general regression (not just STRIPS) later.

Introduction
○○○

Progression
○○○○

Regression
○○○○○○

Regression for STRIPS Tasks
○○○○

Summary
●○

Summary

Summary

- Progression search proceeds forward from the initial state.
- In progression search, the search space is identical to the state space of the planning task.
- Regression search proceeds backwards from the goal.
- Each search state corresponds to a set of world states, for example represented by a formula.
- Regression is simple for STRIPS operators.
- The theory for general regression is more complex.
This is the topic of the following chapters.