

# Planning and Optimization

## E8. Pattern Databases: Pattern Selection

Malte Helmert and Gabriele Röger

Universität Basel

November 13, 2024

# Planning and Optimization

November 13, 2024 — E8. Pattern Databases: Pattern Selection

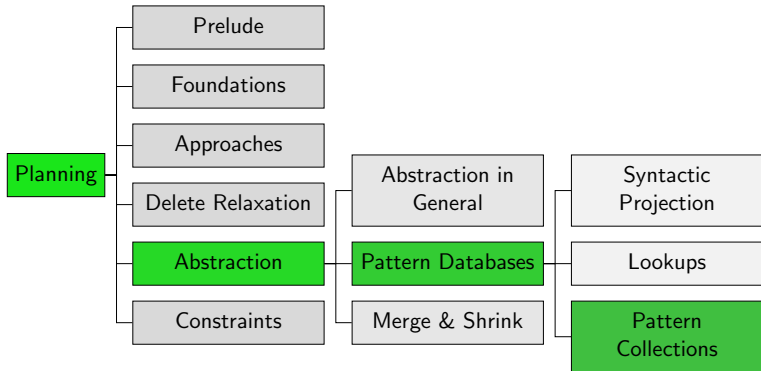
E8.1 Pattern Selection as Local Search

E8.2 Search Neighbourhood

E8.3 Literature

E8.4 Summary

# Content of the Course



# E8.1 Pattern Selection as Local Search

# Pattern Selection as an Optimization Problem

Only one question remains to be answered now  
in order to apply PDBs to planning tasks in practice:

How do we automatically find a good pattern collection?

## The Idea

Pattern selection can be cast as an **optimization problem**:

- ▶ **Given**: a set of **candidates**  
(= pattern collections which fit into a given memory limit)
- ▶ **Find**: a **best possible** candidate, or an approximation  
(= pattern collection with high heuristic quality)

# Pattern Selection as Local Search

How to solve this optimization problem?

- ▶ For problems of interesting size, we cannot hope to find (and prove optimal) a **globally optimal** pattern collection.
  - ▶ **Question:** How many candidates are there?
- ▶ Instead, we try to find **good** solutions by **local search**.

Two approaches from the literature:

- ▶ Edelkamp (2007): using an **evolutionary algorithm**
- ▶ Haslum et al. (2007): using **hill-climbing**

↪ **in the following:** main ideas of the second approach

# Pattern Selection as Hill-Climbing

## Reminder: Hill Climbing

*current* := an **initial candidate**

**loop forever:**

*next* := a **neighbour** of *current* with maximum **quality**

**if**  $quality(next) \leq quality(current)$ :

**return** *current*

*current* := *next*

more on hill climbing:

↪ Foundations of Artificial Intelligence course FS 2024, Ch. C1–C2

# Pattern Selection as Hill-Climbing

## Reminder: Hill Climbing

*current* := an **initial candidate**

**loop forever:**

*next* := a **neighbour** of *current* with maximum **quality**

**if**  $quality(next) \leq quality(current)$ :

**return** *current*

*current* := *next*

Three questions to answer to use this for pattern selection:

- 1 **initial candidate:** What is the initial pattern collection?
- 2 **neighbourhood:** Which pattern collections are considered next starting from a given collection?
- 3 **quality:** How do we evaluate the quality of pattern collections?



## E8.2 Search Neighbourhood

# Search Neighbourhood: Basic Idea

The basic idea is that we

- ▶ start from **small patterns** with only a single variable,
- ▶ grow them by **adding slightly larger patterns**
- ▶ and prefer moving to pattern collections that **improve** the heuristic value of **many states**.

# Initial Pattern Collection

## 1. Initial Candidate

The initial pattern collection is

$\{\{v\} \mid v \text{ is a state variable mentioned in the goal formula}\}$ .

### Motivation:

- ▶ patterns with one variable are the simplest possible ones and hence a natural starting point
- ▶ non-goal patterns are trivial ( $\rightsquigarrow$  Chapter E7), so would be useless

# Which Pattern Collections to Consider Next

From this initial pattern collection, we **incrementally grow** larger pattern collections to obtain an improved heuristic.

## 2. Neighbourhood

The neighbours of  $\mathcal{C}$  are all pattern collections  $\mathcal{C} \cup \{P'\}$  where

- ▶  $P' = P \cup \{v\}$  for some  $P \in \mathcal{C}$ ,
- ▶  $P' \notin \mathcal{C}$ ,
- ▶ all variables of  $P'$  are causally relevant for  $P'$ ,
- ▶  $P'$  is causally connected, and
- ▶ all pattern databases in  $\mathcal{C} \cup \{P'\}$  can be represented within some prespecified space limit.

↪ add **one pattern** with **one additional variable** at a time

↪ use criteria for **redundant** patterns (↪ Chapter E7)  
to avoid neighbours that cannot improve the heuristic

# Checking Causal Relevance and Connectivity

**Remark:** For causal relevance and connectivity, there is a sufficient and necessary criterion which is easy to check:

- ▶  $v$  is a predecessor of some  $u \in P$  in the causal graph, **or**
- ▶  $v$  is a successor of some  $u \in P$  in the causal graph and is mentioned in the goal formula.

# Evaluating the Quality of Pattern Collections

- ▶ The last question we need to answer is how to evaluate the **quality** of pattern collections.
- ▶ This is perhaps the most critical point: without a good evaluation criterion, pattern collections are chosen blindly.

# Approaches for Evaluating Heuristic Quality

Three approaches have been suggested:

- ▶ estimating the **mean heuristic value** of the resulting heuristic (Edelkamp, 2007)
- ▶ estimating **search effort** under the resulting heuristic using a model for predicting search effort (Haslum et al., 2007; Franco et al., 2017)
- ▶ **sampling** states in the state space and counting **how many** of them have **improved** heuristic values compared to the current pattern collection (Haslum et al., 2007)

The last approach is most commonly used and has been shown to work well experimentally.

# Heuristic Quality by Improved Sample States

## 3. Quality

- ▶ Generate  $M$  states  $s_1, \dots, s_M$  through random walks in the state space from the initial state (according to certain parameters not discussed in detail).
- ▶ The **degree of improvement** of a pattern collection  $\mathcal{C}'$  which is generated as a successor of collection  $\mathcal{C}$  is the **number of sample states**  $s_i$  for which  $h^{\mathcal{C}'}(s_i) > h^{\mathcal{C}}(s_i)$ .
- ▶ Use the degree of improvement as the **quality measure** for  $\mathcal{C}'$ .



## Computing $h^{\mathcal{C}'}(s)$

- ▶ So we need to compute  $h^{\mathcal{C}'}(s)$  for some states  $s$  and each candidate successor collection  $\mathcal{C}'$ .
- ▶ We have PDBs for all patterns in  $\mathcal{C}$ , but not for the new pattern  $P' \in \mathcal{C}'$  (of the form  $P \cup \{v\}$  for some  $P \in \mathcal{C}$ ).
- ▶ If possible, we want to avoid fully computing all PDBs for all neighbours.

### Idea:

- ▶ For SAS<sup>+</sup> tasks  $\Pi$ ,  $h^{P'}(s)$  is identical to the **optimal solution cost for the syntactic projection  $\Pi|_{P'}$** .
- ▶ We can use **any optimal planning algorithm** for this.
- ▶ In particular, we can use **A\*** search using  $h^P$  as a heuristic.

## E8.3 Literature

# References (1)

References on planning with pattern databases:



Stefan Edelkamp.

Planning with Pattern Databases.

*Proc. ECP 2001*, pp. 13–24, 2001.

First paper on planning with pattern databases.



Stefan Edelkamp.

Symbolic Pattern Databases in Heuristic Search Planning.

*Proc. AIPS 2002*, pp. 274–283, 2002.

Uses BDDs to store pattern databases more compactly.

## References (2)

References on planning with pattern databases:



Patrik Haslum, Blai Bonet and Héctor Geffner.

New Admissible Heuristics for Domain-Independent Planning.

*Proc. AAAI 2005*, pp. 1164–1168, 2005.

Introduces **constrained PDBs**.

First **pattern selection methods** based on **heuristic quality**.

## References (3)

References on planning with pattern databases:



Stefan Edelkamp.

Automated Creation of Pattern Database Search Heuristics.  
*Proc. MoChArt 2006*, pp. 121–135, 2007.

First **search-based** pattern selection method.



Patrik Haslum, Adi Botea, Malte Helmert, Blai Bonet and  
Sven Koenig.

Domain-Independent Construction of Pattern Database  
Heuristics for Cost-Optimal Planning.

*Proc. AAAI 2007*, pp. 1007–1012, 2007.

Introduces **canonical heuristic** for pattern collections.

Search-based pattern selection based on **Korf, Reid &  
Edelkamp's theory** for search effort estimation.

## References (4)

References on planning with pattern databases:



Santiago Franco, Álvaro Torralba, Levi H. S. Leis and Mike Barley.

On Creating Complementary Pattern Databases  
*Proc. IJCAI 2017*, pp. 4302–4309, 2017.

Improved version of Edelkamp's pattern collection selection approach evaluating pattern collections based on a **prediction of A\* search effort**.

## E8.4 Summary

# Summary

- ▶ One way to automatically find a good pattern collection is by searching in the space of pattern collections.
- ▶ One such approach uses hill-climbing search
  - ▶ starting from single-variable patterns
  - ▶ adding patterns with one additional variable at a time
  - ▶ evaluating patterns by the number of improved sample states
- ▶ By exploiting what we know about redundant patterns, the hill-climbing search space can be reduced significantly.