

Foundations of Artificial Intelligence

35. Automated Planning: Delete Relaxation

Malte Helmert

University of Basel

April 29, 2019

Foundations of Artificial Intelligence

April 29, 2019 — 35. Automated Planning: Delete Relaxation

35.1 How to Design Heuristics?

35.2 Delete Relaxation

35.3 Examples

35.4 Summary

35.1 How to Design Heuristics?

A Simple Planning Heuristic

The STRIPS planner (Fikes & Nilsson, 1971) uses the **number of goals not yet satisfied** in a STRIPS planning task as heuristic:

$$h(s) := |G \setminus s|.$$

intuition: fewer unsatisfied goals \rightsquigarrow closer to goal state

\rightsquigarrow **STRIPS heuristic** (properties?)

Problems of STRIPS Heuristic

drawback of STRIPS heuristic?

- ▶ rather **uninformed**:
For state s , if there is no applicable action a in s such that applying a in s satisfies strictly more (or fewer) goals, then all successor states have the same heuristic value as s .
- ▶ ignores almost the whole **task structure**:
The heuristic values do not depend on the actions.

↔ we need better methods to design heuristics

Planning Heuristics

We consider **three basic ideas** for general heuristics:

- ▶ **delete relaxation** ↔ this and next chapter
- ▶ **abstraction** ↔ later
- ▶ **landmarks** ↔ later

Delete Relaxation: Basic Idea

Estimate solution costs by considering a **simplified planning task**, where all **negative action effects are ignored**.

Automated Planning: Overview

Chapter overview: automated planning

- ▶ 33. Introduction
- ▶ 34. Planning Formalisms
- ▶ 35.–36. Planning Heuristics: Delete Relaxation
 - ▶ 35. **Delete Relaxation**
 - ▶ 36. Delete Relaxation Heuristics
- ▶ 37. Planning Heuristics: Abstraction
- ▶ 38.–39. Planning Heuristics: Landmarks

35.2 Delete Relaxation

Relaxed Planning Tasks: Idea

In STRIPS planning tasks,
good and bad effects are easy to distinguish:

- ▶ **Add effects** are always **useful**.
- ▶ **Delete effects** are always **harmful**.

Why?

idea for designing heuristics: **ignore all delete effects**

Relaxed Planning Tasks

Definition (relaxation of actions)

The **relaxation** a^+ of STRIPS action a is the action with
 $pre(a^+) = pre(a)$, $add(a^+) = add(a)$, $cost(a^+) = cost(a)$,
and $del(a^+) = \emptyset$.

German: Relaxierung von Aktionen

Definition (relaxation of planning tasks)

The **relaxation** Π^+ of a STRIPS planning task $\Pi = \langle V, I, G, A \rangle$
is the task $\Pi^+ := \langle V, I, G, \{a^+ \mid a \in A\} \rangle$.

German: Relaxierung von Planungsaufgaben

Relaxed Planning Tasks

Definition (relaxation of action sequences)

The **relaxation** of action sequence $\pi = \langle a_1, \dots, a_n \rangle$
is the action sequence $\pi^+ := \langle a_1^+, \dots, a_n^+ \rangle$.

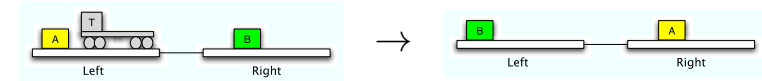
German: Relaxierung von Aktionsfolgen

Relaxed Planning Tasks: Terminology

- ▶ STRIPS planning tasks without delete effects
are called **relaxed planning tasks**
or **delete-free planning tasks**.
- ▶ Plans for relaxed planning tasks are called **relaxed plans**.
- ▶ If Π is a STRIPS planning task and π^+ is a plan for Π^+ ,
then π^+ is called **relaxed plan for Π** .
- ▶ $h^+(\Pi)$ denotes the cost of an **optimal plan** for Π^+ ,
i.e., of an **optimal relaxed plan**.
- ▶ analogously: $h^+(s)$ cost of optimal relaxed plan
starting in state s (instead of initial state)
- ▶ h^+ is called **optimal relaxation heuristic**.

35.3 Examples

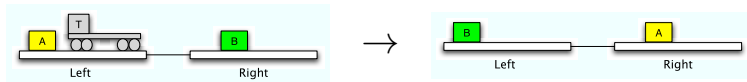
Example: Logistics



Example (Logistics Task)

- ▶ variables: $V = \{at_{AL}, at_{AR}, at_{BL}, at_{BR}, at_{TL}, at_{TR}, in_{AT}, in_{BT}\}$
- ▶ initial state: $I = \{at_{AL}, at_{BR}, at_{TL}\}$
- ▶ goals: $G = \{at_{AR}, at_{BL}\}$
- ▶ actions: $\{move_{LR}, move_{RL}, load_{AL}, load_{AR}, load_{BL}, load_{BR}, unload_{AL}, unload_{AR}, unload_{BL}, unload_{BR}\}$
- ▶ ...

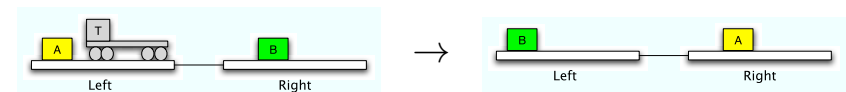
Example: Logistics



Example (Logistics Task)

- ▶ $pre(move_{LR}) = \{at_{TL}\}$, $add(move_{LR}) = \{at_{TR}\}$,
 $del(move_{LR}) = \{at_{TL}\}$, $cost(move_{LR}) = 1$
- ▶ $pre(load_{AL}) = \{at_{TL}, at_{AL}\}$, $add(load_{AL}) = \{in_{AT}\}$,
 $del(load_{AL}) = \{at_{AL}\}$, $cost(load_{AL}) = 1$
- ▶ $pre(unload_{AL}) = \{at_{TL}, in_{AT}\}$, $add(unload_{AL}) = \{at_{AL}\}$,
 $del(unload_{AL}) = \{in_{AT}\}$, $cost(unload_{AL}) = 1$
- ▶ ...

Example: Logistics



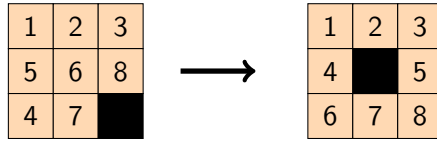
▶ optimal plan:

- 1 $load_{AL}$
- 2 $move_{LR}$
- 3 $unload_{AR}$
- 4 $load_{BR}$
- 5 $move_{RL}$
- 6 $unload_{BL}$

▶ optimal relaxed plan: ?

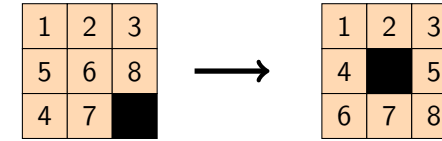
- ▶ $h^*(I) = 6$, $h^+(I) = ?$

Example: 8-Puzzle



- ▶ (original) task:
 - ▶ A tile can be moved from cell A to B if A and B are adjacent and B is free.
- ▶ simplification (basis for Manhattan distance):
 - ▶ A tile can be moved from cell A to B if A and B are adjacent.
- ▶ relaxed task:
 - ▶ A tile can be moved from cell A to B if A and B are adjacent and B is free.
 - ▶ ... where delete effects are ignored (in particular: free cells at earlier time remain free)

Example: 8-Puzzle



- ▶ actual goal distance: $h^*(s) = 8$
- ▶ Manhattan distance: $h^{\text{MD}}(s) = 6$
- ▶ optimal delete relaxation: $h^+(s) = 7$

relationship:

h^+ **dominates** the Manhattan distance in the sliding tile puzzle (i.e., $h^{\text{MD}}(s) \leq h^+(s) \leq h^*(s)$ for all states s)

Relaxed Solutions: Suboptimal or Optimal?

- ▶ For general STRIPS planning tasks, h^+ is an **admissible and consistent heuristic**.
- ▶ Can h^+ be computed efficiently?
 - ▶ It is **easy** to solve delete-free planning tasks **suboptimally**. (How?)
 - ▶ optimal solution (and hence the computation of h^+) is **NP-hard** (reduction from SET COVER)
- ▶ In practice, heuristics approximate h^+ from below or above.

35.4 Summary

Summary

delete relaxation:

- ▶ ignore **negative effects** (delete effects) of actions
- ▶ use **solution costs of relaxed planning task** as **heuristic** for solution costs of the original planning task
- ▶ computation of optimal relaxed solution costs h^+ is NP-hard, hence usually **approximated** from below or above