Planning and Optimization

C1. Delete Relaxation: Introduction

Malte Helmert and Gabriele Röger

Universität Basel

October 24, 2016

Planning and Optimization

October 24, 2016 — C1. Delete Relaxation: Introduction

C1.1 Heuristics

Planning as Heuristic Search

- Heuristic search is the most common approach to planning.
- ingredients: general search algorithm + heuristic
- heuristic estimates cost from a given state to a given goal
  - progression: from varying states s to fixed goal γ
  - regression: from fixed initial state I to varying subgoals φ
- Over the next weeks, we study the main ideas behind heuristics for planning tasks.

Reminder: Heuristics

Need to Catch Up?

- We assume familiarity with heuristics and their properties:
  - heuristic $h : S \to \mathbb{R}_0^+ \cup \{\infty\}$
  - perfect heuristic h^*: $h^*(s)$ cost of optimal solution from s (∞ if unsolvable)
  - properties of heuristics h:
    - safe: $(h(s) = \infty \Rightarrow h^*(s) = \infty)$ for all states s
    - goal-aware: $h(s) = 0$ for all goal states s
    - admissible: $h(s) \le h^*(s)$ for all states s
    - consistent: $h(s) \le cost(o) + h(s')$ for all transitions $s \xrightarrow{o} s'$
  - connections between these properties
- If you are not familiar with these topics, we recommend Chapters 13–14 of the Foundations of Artificial Intelligence course at `http://informatik.unibas.ch/fs2016/grundlagen-der-kuenstlichen-intelligenz/`.

C1.2 Coming Up with Heuristics

A Simple Heuristic for Planning Tasks

STRIPS (Fikes & Nilsson, 1971) used the number of state variables that differ in current state s and a STRIPS goal $v_1 \wedge \dots \wedge v_n$:

$$h(s) := |\{i \in \{1, \dots, n\} \mid s \not\models v_i\}|.$$

Intuition: more satisfied goal atoms $\rightsquigarrow$ closer to the goal

$\rightsquigarrow$ STRIPS heuristic (a.k.a. goal-count heuristic) (properties?)

Criticism of the STRIPS Heuristic

What is wrong with the STRIPS heuristic?

- quite uninformative:
  the range of heuristic values in a given task is small;
  typically, most successors have the same estimate
- very sensitive to reformulation:
  can easily transform any planning task into an equivalent one
  where $h(s) = 1$ for all non-goal states (how?)
- ignores almost all problem structure:
  heuristic value does not depend on the set of operators!

$\rightsquigarrow$ need a better, principled way of coming up with heuristics

Coming Up with Heuristics in a Principled Way

General Procedure for Obtaining a Heuristic

- Simplify the problem, for example by removing problem constraints.
- Solve the simplified problem (ideally optimally).
- Use the solution cost for the simplified problem as a heuristic for the real problem.

As heuristics are computed for every generated search state, it is important that they can be computed efficiently.

Relaxing a Problem: Example

Example (Route Planning in a Road Network)

The road network is formalized as a weighted graph over points in the Euclidean plane. The weight of an edge is the road distance between two locations.

Example (Relaxation for Route Planning)

Use the Euclidean distance $\sqrt{|x_1 - x_2|^2 + |y_1 - y_2|^2}$ as a heuristic for the road distance between $\langle x_1, y_1 \rangle$ and $\langle x_2, y_2 \rangle$
This is a lower bound on the road distance ($\rightsquigarrow$ admissible).

$\rightsquigarrow$ We drop the constraint of having to travel on roads.

Planning Heuristics: Main Concepts

Major ideas for heuristics in the planning literature:

- delete relaxation
- abstraction
- landmarks
- critical paths
- network flows

We will consider all of them in this course, beginning with delete relaxation.

C1.3 Relaxed Planning Tasks

Delete Relaxation: Idea

In positive normal form (Chapter A6, remember?), good and bad effects are easy to distinguish:

- Effects that make state variables true are good (add effects).
- Effects that make state variables false are bad (delete effects).

Idea of delete relaxation heuristics: ignore all delete effects.

Delete-Relaxed Planning Tasks

Definition (Delete Relaxation of Operators)

The delete relaxation o^+ of an operator o in positive normal form is the operator obtained by replacing all negative effects $\neg a$ within $\mathit{eff}(o)$ by the do-nothing effect $\top$.

Definition (Delete Relaxation of Planning Tasks)

The delete relaxation Π^+ of a planning task $\Pi = \langle V, I, O, \gamma \rangle$ in positive normal form is the planning task $\Pi^+ := \langle V, I, \{o^+ \mid o \in O\}, \gamma \rangle$.

Definition (Delete Relaxation of Operator Sequences)

The delete relaxation of an operator sequence $\pi = \langle o_1, \dots, o_n \rangle$ is the operator sequence $\pi^+ := \langle o_1^+, \dots, o_n^+ \rangle$.

Note: "delete" is often omitted: relaxation, relaxed

Relaxed Planning Tasks: Terminology

- Planning tasks in positive normal form without delete effects are called relaxed planning tasks.
- Plans for relaxed planning tasks are called relaxed plans.
- If Π is a planning task in positive normal form and π^+ is a plan for Π^+, then π^+ is called a relaxed plan for Π.

C1.4 The Domination Lemma

On-Set and Dominating States

Definition (On-Set)

The on-set of a valuation s is the set of propositional variables that are true in s, i.e., $on(s) = s^{-1}(\{\mathbf{T}\})$.

$\rightsquigarrow$ for states: states can be viewed as sets of (true) state variables

Definition (Dominate)

A valuation s' dominates a valuation s if $on(s) \subseteq on(s')$.

$\rightsquigarrow$ all state variables true in s are also true in s'

Domination Lemma (1)

Lemma (Domination)

Let s and s' be valuations of a set of propositional variables V, and let χ be a propositional formula over V which does not contain negation symbols.

If $s \models \chi$ and s' dominates s, then $s' \models \chi$.

Proof.

Proof by induction over the structure of χ.

- Base case $\chi = \top$: then $s' \models \top$.
- Base case $\chi = \bot$: then $s \not\models \bot$.

...

Domination Lemma (2)

Proof (continued).

- Base case $\chi = v \in V$: if $s \models v$, then $v \in on(s)$. With $on(s) \subseteq on(s')$, we get $v \in on(s')$ and hence $s' \models v$.
- Inductive case $\chi = \chi_1 \wedge \chi_2$: by induction hypothesis, our claim holds for the proper subformulas χ_1 and χ_2 of χ.

$$\begin{aligned} s \models \chi &\implies s \models \chi_1 \wedge \chi_2 \\ &\implies s \models \chi_1 \text{ and } s \models \chi_2 \\ &\overset{\text{I.H. (twice)}}{\implies} s' \models \chi_1 \text{ and } s' \models \chi_2 \\ &\implies s' \models \chi_1 \wedge \chi_2 \\ &\implies s' \models \chi. \end{aligned}$$

- Inductive case $\chi = \chi_1 \vee \chi_2$: analogous

$\square$

C1.5 The Relaxation Lemma

Add Sets and Delete Sets

Definition (Add Set and Delete Set for an Effect)

For all effects e and states s over state variables V, the add set of e in s, written $[e]_s^+$, and the delete set of e in s, written $[e]_s^-$, are defined as the following sets of state variables:

$$[e]_s^+ = \{v \in V \mid v \in [e]_s\}$$
$$[e]_s^- = \{v \in V \mid \neg v \in [e]_s\}$$

Note: For all states s and operators o applicable in s, we have $on(s[\![o]\!]) = (on(s) \setminus [eff(o)]_s^-) \cup [eff(o)]_s^+$.

Relaxation Lemma

For this and the following chapters on delete relaxation, we always assume tasks are in positive normal form.

Lemma (Relaxation)

Let s be a state, let s' be a state that dominates s, and let π be an operator sequence which is applicable in s.

Then π^+ is applicable in s' and $s'[\![\pi^+]\!]$ dominates $s[\![\pi]\!]$.

Moreover, if π leads to a goal state from s, then π^+ leads to a goal state from s'.

Proof of Relaxation Lemma (1)

Proof.

The "moreover" part follows from the rest by the domination lemma. Prove the rest by induction over the length of π.

Base case: $\pi = \langle\rangle$
$s'[\![\pi^+]\!] = s'$ dominates $s[\![\pi]\!] = s$ by prerequisite.

Inductive case: $\pi = \langle o_1, \dots, o_{n+1}\rangle$
By the induction hypothesis, $\langle o_1^+, \dots, o_n^+\rangle$ is applicable in s', and $t' = s'[\![\langle o_1^+, \dots, o_n^+\rangle]\!]$ dominates $t = s[\![\langle o_1, \dots, o_n\rangle]\!]$.

o_{n+1} is applicable in t, and thus $t \models pre(o_{n+1})$.
By the domination lemma, we get $t' \models pre(o_{n+1})$ and hence o_{n+1}^+ is applicable in t'. Therefore, π^+ is applicable in s'.

Because o_{n+1} is in positive normal form, all effect conditions satisfied by t are also satisfied by t' (by the domination lemma).
Therefore, $[eff(o_{n+1})]_t^+ \subseteq [eff(o_{n+1}^+)]_{t'}$. ...

Proof of Relaxation Lemma (2)

Proof (continued).

We get:

$$\begin{aligned} on(s[\![\pi]\!]) &= (on(t) \setminus [eff(o_{n+1})]_t^-) \cup [eff(o_{n+1})]_t^+ \\ &\subseteq on(t) \cup [eff(o_{n+1})]_t^+ \\ &\subseteq on(t') \cup [eff(o_{n+1}^+)]_{t'} \\ &= on(s'[\![\pi^+]\!]), \end{aligned}$$

and thus $s'[\![\pi^+]\!]$ dominates $s[\![\pi]\!]$. □

Consequences of the Relaxation Lemma (1)

Corollary (Relaxation Preserves Plans and Leads to Dominance)

Let π be an operator sequence that is applicable in state s. Then π^+ is applicable in s and $s[\![\pi^+]\!]$ dominates $s[\![\pi]\!]$. If π is a plan for Π, then π^+ is a plan for Π^+.

Proof.

Apply relaxation lemma with $s' = s$. □

- $\rightsquigarrow$ Relaxations of plans are relaxed plans.
- $\rightsquigarrow$ Delete relaxation is no harder to solve than original task.
- $\rightsquigarrow$ Optimal relaxed plans are never more expensive than optimal plans for original tasks.

Consequences of the Relaxation Lemma (2)

Corollary (Relaxation Preserves Dominance)

Let s be a state, let s' be a state that dominates s, and let π^+ be a relaxed operator sequence applicable in s. Then π^+ is applicable in s' and $s'[\![\pi^+]\!]$ dominates $s[\![\pi^+]\!]$.

Proof.

Apply relaxation lemma with π^+ for π, noting that $(\pi^+)^+ = \pi^+$. □

- $\rightsquigarrow$ If there is a relaxed plan starting from state s, the same plan can be used starting from a dominating state s'.
- $\rightsquigarrow$ Dominating states are always "better" in relaxed planning tasks.

C1.6 Summary

Summary

- A general way to come up with heuristics: solve a simplified version of the real problem, for example by removing problem constraints.
- delete relaxation: given a task in positive normal form, discard all delete effects
- Delete relaxation is a simplification in the sense that it is never harder to solve a relaxed task than the original one.
- Delete-relaxed tasks have a domination property: it is always beneficial to make more state variables true.