Planning and Optimization

C2. Delete Relaxation: Properties & Finding Relaxed Plans

Malte Helmert and Gabriele Röger

Universität Basel

October 23, 2017

C2.1 The Relaxation Lemma

C2.2 Further Properties

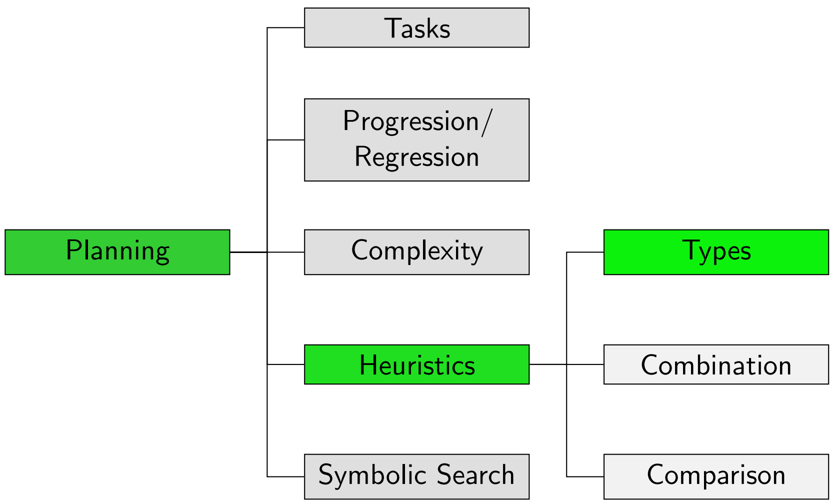
C2.3 Greedy Algorithm

C2.4 Optimal Relaxed Plans

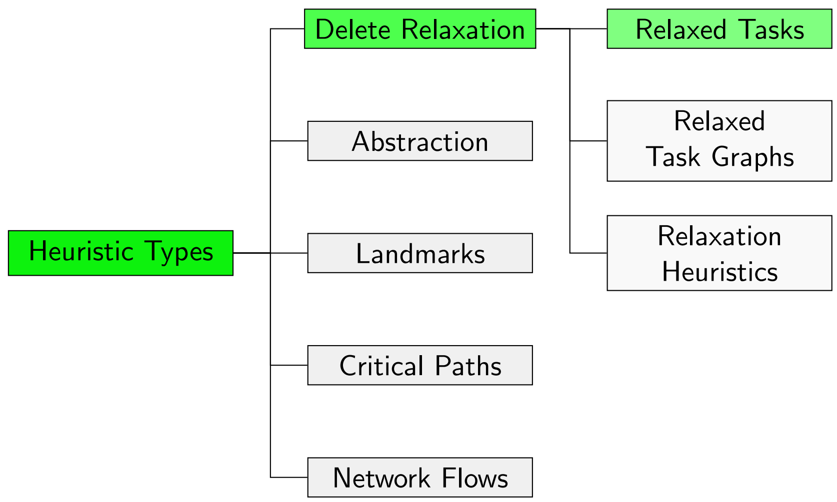
C2.5 Discussion

C2.6 Summary

Content of this Course



Content of this Course: Heuristic Types



C2.1 The Relaxation Lemma

Add Sets and Delete Sets

Definition (Add Set and Delete Set for an Effect)

Consider a propositional planning task with state variables V.
Let e be an effect over V, and let s be a state over V.
The add set of e in s, written $addset(e, s)$,
and the delete set of e in s, written $delset(e, s)$,
are defined as the following sets of state variables:

$$addset(e, s) = \{v \in V \mid s \models effcond(v, e)\}$$
$$delset(e, s) = \{v \in V \mid s \models effcond(\neg v, e)\}$$

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

Relaxation Lemma

For this and the following chapters on delete relaxation,
we assume implicitly that we are working with
propositional planning tasks in positive normal form.

Lemma (Relaxation)

Let s be a state, and let s' be a state that dominates s.

1. *If o is an operator applicable in s, then o^+ is applicable in s' and $s'[\![o^+]\!]$ dominates $s[\![o]\!]$.*
2. *If π is an operator sequence applicable in s, then π^+ is applicable in s' and $s'[\![\pi^+]\!]$ dominates $s[\![\pi]\!]$.*
3. *If additionally π leads to a goal state from state s, then π^+ leads to a goal state from state s'.*

Proof of Relaxation Lemma (1)

Proof.

Let V be the set of state variables.

Part 1: Because o is applicable in s, we have $s \models pre(o)$.

Because $pre(o)$ is negation-free and s' dominates s,
we get $s' \models pre(o)$ from the domination lemma.

Because $pre(o^+) = pre(o)$, this shows that o^+ is applicable in s'.

...

Proof of Relaxation Lemma (2)

Proof (continued).

To prove that $s'[\![o^+]\!]$ dominates $s[\![o]\!]$,
we first compare the relevant add sets:

$$\begin{aligned} \mathit{addset}(\mathit{eff}(o), s) &= \{v \in V \mid s \models \mathit{effcond}(v, \mathit{eff}(o))\} \\ &= \{v \in V \mid s \models \mathit{effcond}(v, \mathit{eff}(o^+))\} && (1) \\ &\subseteq \{v \in V \mid s' \models \mathit{effcond}(v, \mathit{eff}(o^+))\} && (2) \\ &= \mathit{addset}(\mathit{eff}(o^+), s'), \end{aligned}$$

where (1) uses $\mathit{effcond}(v, \mathit{eff}(o)) \equiv \mathit{effcond}(v, \mathit{eff}(o^+))$
and (2) uses the dominance lemma (note that effect conditions are negation-free for operators in positive normal form). ...

Proof of Relaxation Lemma (3)

Proof (continued).

We then get:

$$\begin{aligned} \mathit{on}(s[\![o]\!]) &= (\mathit{on}(s) \setminus \mathit{delset}(\mathit{eff}(o), s)) \cup \mathit{addset}(\mathit{eff}(o), s) \\ &\subseteq \mathit{on}(s) \cup \mathit{addset}(\mathit{eff}(o), s) \\ &\subseteq \mathit{on}(s') \cup \mathit{addset}(\mathit{eff}(o^+), s') \\ &= \mathit{on}(s'[\![o^+]\!]), \end{aligned}$$

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

This concludes the proof of Part 1. ...

Proof of Relaxation Lemma (4)

Proof (continued).

Part 2: by induction over $n = |\pi|$

Base case: $\pi = \langle\rangle$
The empty plan is trivially applicable in s', and
$s'[\![\langle\rangle^+]\!] = s'$ dominates $s[\![\langle\rangle]\!] = s$ 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 $t = s[\![\langle o_1, \dots, o_n\rangle]\!]$.
Also, o_{n+1} is applicable in t.

Using Part 1, o_{n+1}^+ is applicable in t' and $s'[\![\pi^+]\!] = t'[\![o_{n+1}^+]\!]$ dominates $s[\![\pi]\!] = t[\![o_{n+1}]\!]$.

This concludes the proof of Part 2. ...

Proof of Relaxation Lemma (5)

Proof (continued).

Part 3: Let γ be the goal formula.

From Part 2, we obtain that $t' = s'[\![\pi^+]\!]$ dominates $t = s[\![\pi]\!]$.
By prerequisite, t is a goal state and hence $t \models \gamma$.

Because the task is in positive normal form, γ is negation-free,
and hence $t' \models \gamma$ because of the domination lemma.

Therefore, t' is a goal state. □

C2.2 Further Properties

Further Properties of Delete Relaxation

- The relaxation lemma is the main technical result that we will use to study delete relaxation.
- Next, we derive some further properties of delete relaxation that will be useful for us.
- Two of these are direct consequences of the relaxation lemma.

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 tasks.

Monotonicity of Relaxed Planning Tasks

Lemma (Monotonicity)

Let s be a state in which relaxed operator o^+ is applicable. Then $s[\![o^+]\!]$ dominates s.

Proof.

Since relaxed operators only have positive effects, we have $on(s) \subseteq on(s) \cup addset(eff(o^+), s) = on(s[\![o^+]\!])$. □

$\rightsquigarrow$ Together with our previous results, this means that making a transition in a relaxed planning task never hurts.

Finding Relaxed Plans

Using the theory we developed, we are now ready to study the problem of finding plans for relaxed planning tasks.

C2.3 Greedy Algorithm

Greedy Algorithm for Relaxed Planning Tasks

The relaxation and monotonicity lemmas suggest the following algorithm for solving relaxed planning tasks:

Greedy Planning Algorithm for $\langle V, I, O^+, \gamma\rangle$

$s := I$
$\pi^+ := \langle\rangle$
loop forever:
  if $s \models \gamma$:
    return π^+
  else if there is an operator $o^+ \in O^+$ applicable in s
      with $s[\![o^+]\!] \neq s$:
    Append such an operator o^+ to π^+.
    $s := s[\![o^+]\!]$
  else:
    return unsolvable

Correctness of the Greedy Algorithm

The algorithm is sound:

- If it returns a plan, this is indeed a correct solution.
- If it returns "unsolvable", the task is indeed unsolvable
  - Upon termination, there clearly is no relaxed plan from s.
  - By iterated application of the monotonicity lemma, s dominates I.
  - By the relaxation lemma, there is no solution from I.

What about completeness (termination) and runtime?

- Each iteration of the loop adds at least one atom to $on(s)$.
- This guarantees termination after at most $|V|$ iterations.
- Thus, the algorithm can clearly be implemented to run in polynomial time.
  - A good implementation runs in $O(\|\Pi\|)$.

Using the Greedy Algorithm as a Heuristic

We can apply the greedy algorithm within heuristic search:

- When evaluating a state s in progression search, solve relaxation of planning task with initial state s.
- When evaluating a subgoal φ in regression search, solve relaxation of planning task with goal φ.
- Set $h(s)$ to the cost of the generated relaxed plan.

Is this an admissible heuristic?

- Yes if the relaxed plans are optimal (due to the plan preservation corollary).
- However, usually they are not, because our greedy relaxed planning algorithm is very poor.

(What about safety? Goal-awareness? Consistency?)

C2.4 Optimal Relaxed Plans

The Set Cover Problem

To obtain an admissible heuristic, we must compute optimal relaxed plans. Can we do this efficiently?

This question is related to the following problem:

Problem (Set Cover)

Given: *a finite set U, a collection of subsets $C = \{C_1, \dots, C_n\}$ with $C_i \subseteq U$ for all $i \in \{1, \dots, n\}$, and a natural number K.*
Question: *Is there a set cover of size at most K, i.e., a subcollection $S = \{S_1, \dots, S_m\} \subseteq C$ with $S_1 \cup \dots \cup S_m = U$ and $m \leq K$?*

The following is a classical result from complexity theory:

Theorem (Karp 1972)

The set cover problem is NP-complete.

Complexity of Optimal Relaxed Planning (1)

Theorem (Complexity of Optimal Relaxed Planning)

The BCPLANEX *problem restricted to delete-relaxed planning tasks is NP-complete.*

Proof.

For membership in NP, guess a plan and verify.

It is sufficient to check plans of length at most $|V|$ where V is the set of state variables, so this can be done in nondeterministic polynomial time.

For hardness, we reduce from the set cover problem. . . .

Complexity of Optimal Relaxed Planning (2)

Proof (continued).

Given a set cover instance $\langle U, C, K\rangle$, we generate the following relaxed planning task $\Pi^+ = \langle V, I, O^+, \gamma\rangle$:

- $V = U$
- $I = \{v \mapsto \mathbf{F} \mid v \in V\}$
- $O^+ = \{\langle \top, \bigwedge_{v \in C_i} v, 1\rangle \mid C_i \in C\}$
- $\gamma = \bigwedge_{v \in U} v$

If S is a set cover, the corresponding operators form a plan. Conversely, each plan induces a set cover by taking the subsets corresponding to the operators. There exists a plan of cost at most K iff there exists a set cover of size K.

Moreover, Π^+ can be generated from the set cover instance in polynomial time, so this is a polynomial reduction. □

C2.5 Discussion

Using Relaxations in Practice

How can we use relaxations for heuristic planning in practice?

Different possibilities:

- Implement an optimal planner for relaxed planning tasks and use its solution costs as estimates, even though optimal relaxed planning is NP-hard.
  $\rightsquigarrow$ h^+ heuristic
- Do not actually solve the relaxed planning task, but compute an approximation of its solution cost.
  $\rightsquigarrow$ h^{max} heuristic, h^{add} heuristic, $h^{\mathrm{LM\text{-}cut}}$ heuristic
- Compute a solution for relaxed planning tasks which is not necessarily optimal, but "reasonable".
  $\rightsquigarrow$ h^{FF} heuristic

$\rightsquigarrow$ more in the following chapters

C2.6 Summary

Summary

- 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.
- Because of their monotonicity property, delete-relaxed tasks can be solved in polynomial time by a greedy algorithm.
- However, the solution quality of this algorithm is poor.
- For an informative heuristic, we would ideally want to find optimal relaxed plans.
- However, the bounded-cost plan existence problem for relaxed planning tasks is NP-complete.