Relaxed Planning Graph Heuristic

Excerpt from CSC384, winter 2014

Planning

- We will look at one technique:
  Relaxed Plan heuristics used with heuristic search.

  The heuristics are domain independent. As such they are part of a class of so-called domain-independent heuristic search for planning

Reachability Analysis.

- The idea is to consider what happens if we ignore the delete lists of actions.
- This is yields a "relaxed problem" that can produce a useful heuristic estimate.

Reachability Analysis

- In the relaxed problem actions add new facts, but never delete facts.
- Then we can do reachability analysis, which is much simpler than searching for a solution.

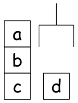
Reachability

- We start with the initial state S_0.
- We alternate between state and action layers.
- We find all actions whose preconditions are contained in S_0. These actions comprise the first action layer A_0.
- The next state layer contains:
  - S_0 U all states added by the actions in A_0.
- In general:
  - A_i … set of actions whose preconditions are in S_i.
  - $S_i = S_{i-1}$ U the add lists of all of the actions in A_i

STRIPS Blocks World Operators.

- pickup(X)
  Pre: {handempty, ontable(X), clear(X)}
  Add: {holding(X)}
  ~~Del: {handempty, ontable(X), clear(X)}~~
- putdown(X)
  Pre: {holding(X)}
  Add: {handempty, ontable(X), clear(X)}
  ~~Del: {holding(X)}~~
- unstack(X,Y)
  Pre: {handempty, clear(X), on(X,Y)}
  Add: {holding(X), clear(Y)}
  ~~Del: {handempty, clear(X), on(X,Y)}~~
- stack(X,Y)
  Pre: {holding(X),clear(Y)}
  Add: {handempty, clear(X), on(X,Y)}
  ~~Del: {holding(X),clear(Y)}~~

Example

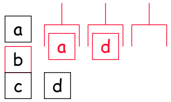


on(a,b),
on(b,c),
ontable(c),
ontable(d),
clear(a),
clear(d),
handempty

S_0

unstack(a,b)
pickup(d)

A_0

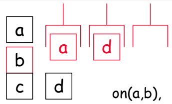


on(a,b),
on(b,c),
ontable(c),
ontable(d),
clear(a),
handempty,
clear(d),
holding(a),
clear(b),
holding(d)

S_1

this is not a state as some of these facts cannot be true at the same time!

Example



on(a,b),
on(b,c),
ontable(c),
ontable(d),
clear(a),
clear(d),
handempty,
holding(a),
clear(b),
holding(d)

S_1

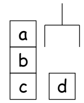
unstack(a,b)
pickup(d)
} from A_0

putdown(a),
putdown(d),
stack(a,b),
stack(a,a),
stack(d,a),
stack(d,b),
stack(d,d),
unstack(b,c)
…

Impossible, but we don't know because we ignore dels.

A_1

Example

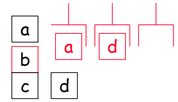


on(a,b),
on(b,c),
ontable(c),
ontable(d),
clear(a),
clear(d),
handempty

S_0

unstack(a,b)
pickup(d)

A_0



on(a,b),
on(b,c),
ontable(c),
ontable(d),
clear(a),
handempty,
clear(d),
holding(a),
clear(b),
holding(d)

S_1

this is not a state!

Example

on(a,b),
on(b,c),
ontable(c),
ontable(d),
clear(a),
clear(d),
handempty,
holding(a),
clear(b),
holding(d)

S_1

putdown(a),
putdown(d),
stack(a,b),
stack(a,a),
stack(d,b),
stack(d,a),
pickup(d),
…
unstack(b,c)
…

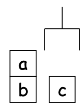
A_1

Reachabilty

- We continue until:
  - the goal G is contained in the state layer, or
  - until the state layer no longer changes (reached fix point).
- Intuitively:
  - the actions at level A_i are the actions that could be executed at the i-th step of some plan, and
  - the facts in level S_i are the facts that could be made true within a plan of length i.
- Some of the actions/facts have this property. But not all!

Reachability

to reach on(c,b) requires 4 actions



on(a,b),
ontable(c),
ontable(b),
clear(a),
clear(c),
handempty

S_0

unstack(a,b)
pickup(c)

A_0

on(a,b),
ontable(c),
ontable(b),
clear(a),
clear(c),
handempty,
holding(a),
clear(b),
holding(c)

S_1

stack(c,b)
…

A_1

on(c,b),
…

but stack(c,b) cannot be executed after one step

Heuristics from Reachability Analysis

Grow the levels until the goal is contained in the final state level S_K.

- If the state level stops changing and the goal is not present: The goal is unachievable under the assumption that (a) the goal is a set of positive facts, and (b) all preconditions are positive facts.

- Then do the following

Heuristics from Reachability Analysis

CountActions(G,S_K):

/ Compute the number of actions contained in a relaxed plan achieving the goal. */*

- Split G into facts in S_{K-1} and elements in S_K only.
  - G_P contains the previously achieved (in S_{K-1}) and
  - G_N contains the just achieved parts of G (only in S_K).
- Find a minimal set of actions A whose add effects cover G_N.
  - may contain no redundant actions,
  - but may not be the minimum sized set (computing the minimum sized set of actions is the set cover problem and is NP-Hard)
- NewG := S_{K-1} U preconditions of A.
- return CountAction(NewG,S_{K-1}) + size(A)

Heuristics from Reachability Analysis

CountActions(G,S_K):

/ Compute the number of actions contained in a relaxed plan achieving the goal. */*

- Split G into facts in S_{K-1} and elements in S_K only.
  - G_P contains the previously achieved (in S_{K-1}) and
  - G_N contains the just achieved parts of G (only in S_K).
- Find a minimal set of actions A whose add effects cover G_N.
  - may contain no redundant actions,
  - but may not be the minimum sized set (computing the minimum sized set of actions is the set cover problem and is NP-Hard)
- NewG := G_P U preconditions of A.
- return CountAction(NewG,S_{K-1}) + size(A)

Example

legend: [pre]act[add]

$S_0 = \{f_1, f_2, f_3\}$
$A_0 = \{[f_1]a_1[f_4], [f_2]a_2[f_5]\}$

Goal: f_6,f_5,f_1
Actions:
$[f_1]a_1[f_4]$
$[f_2]a_2[f_5]$
$[f_2,f_4,f_5]a_3[f_6$

Example

legend: [pre]act[add]

$S_0 = \{f_1, f_2, f_3\}$
$A_0 = \{[f_1]a_1[f_4], [f_2]a_2[f_5]\}$
$S_1 = \{f_1,f_2,f_3,f_4,f_5\}$

Goal: f_6,f_5,f_1
Actions:
$[f_1]a_1[f_4]$
$[f_2]a_2[f_5]$
$[f_2,f_4,f_5]a_3[f_6$

Example

legend: [pre]act[add]

$S_0 = \{f_1, f_2, f_3\}$
$A_0 = \{[f_1]a_1[f_4], [f_2]a_2[f_5]\}$
$S_1 = \{f_1,f_2,f_3,f_4,f_5\}$
$A_1 = \{[f_2,f_4,f_5]a_3[f_6]\}$

Goal: f_6,f_5,f_1
Actions:
$[f_1]a_1[f_4]$
$[f_2]a_2[f_5]$
$[f_2,f_4,f_5]a_3[f_6$

Example

legend: [pre]act[add]

$S_0 = \{f_1, f_2, f_3\}$
$A_0 = \{[f_1]a_1[f_4],\ [f_2]a_2[f_5]\}$
$S_1 = \{f_1,f_2,f_3,f_4,f_5\}$
$A_1 = \{[f_2,f_4,f_5]a_3[f_6]\}$
$\mathbf{S_2} = \{f_1,f_2,f_3,f_4,f_5,f_6\}$

Goal: f_6,f_5,f_1
Actions:
$[f_1]a_1[f_4]$
$[f_2]a_2[f_5]$
$[f_2,f_4,f_5]a_3[f_6$

Example

legend: [pre]act[add]

$S_0 = \{f_1, f_2, f_3\}$
$A_0 = \{[f_1]a_1[f_4],\ [f_2]a_2[f_5]\}$
$S_1 = \{f_1,f_2,f_3,f_4,f_5\}$
$A_1 = \{[f_2,f_4,f_5]a_3[f_6]\}$
$\mathbf{S_2} = \{f_1,f_2,f_3,f_4,f_5,f_6\}$

$G = \{f_6,f_5,f_1\}$

Example

legend: [pre]act[add]

$S_0 = \{f_1, f_2, f_3\}$
$A_0 = \{[f_1]a_1[f_4],\ [f_2]a_2[f_5]\}$
$S_1 = \{f_1,f_2,f_3,f_4,f_5\}$
$A_1 = \{[f_2,f_4,f_5]a_3[f_6]\}$
$\mathbf{S_2} = \{f_1,f_2,f_3,f_4,f_5,f_6\}$

Goal: f_6,f_5,f_1
Actions:
$[f_1]a_1[f_4]$
$[f_2]a_2[f_5]$
$[f_2,f_4,f_5]a_3[f_6$

$G = \{f_6,f_5, f_1\}$

We split G into G_P and G_N:

Example

legend: [pre]act[add]

$S_0 = \{f_1, f_2, f_3\}$
$A_0 = \{[f_1]a_1[f_4],\ [f_2]a_2[f_5]\}$
$S_1 = \{f_1,f_2,f_3,f_4,f_5\}$
$A_1 = \{[f_2,f_4,f_5]a_3[f_6]\}$
$\mathbf{S_2} = \{f_1,f_2,f_3,f_4,f_5,f_6\}$

$G = \{f_6,f_5,f_1\}$
$G_N = \{f_6\}$ (newly achieved)
$G_p = \{f_5, f_1\}$ (achieved before)

Example

legend: [pre]act[add]

$S_0 = \{f_1, f_2, f_3\}$
$A_0 = \{[f_1]a_1[f_4],\ [f_2]a_2[f_5]\}$
$S_1 = \{f_1,f_2,f_3,f_4,f_5\}$
$A_1 = \{[f_2,f_4,f_5]a_3[f_6]\}$
$\mathbf{S_2} = \{f_1,f_2,f_3,f_4,f_5,f_6\}$

$G = \{f_6,f_5, f_1\}$

We split G into G_P and G_N:

CountActs(G,S_2)
$G_P = \{f_5, f_1\}$ **//already in S1**
$G_N = \{f_6\}$ **//New in S2**
$A = \{a_3\}$ **//adds all in G_N**

//the new goal: $G_P \cup Pre(A)$
$G_1 = \{f_5,f_1,f_2,f_4\}$
Return
1 + CountActs(G_1,S_1)

Example

Now, we are at level S1

$S_0 = \{f_1, f_2, f_3\}$
$A_0 = \{[f_1]a_1[f_4],\ [f_2]a_2[f_5]\}$
$\mathbf{S_1} = \{f_1,f_2,f_3,f_4,f_5\}$
$A_1 = \{[f_2,f_4,f_5]a_3[f_6]\}$
$S_2 = \{f_1,f_2,f_3,f_4,f_5,f_6\}$

$G_1 = \{f_5,f_1,f_2,f_4\}$

CountActs(G_1,S_1)

Example

Now, we are at level S1

$S_0 = \{f_1, f_2, f_3\}$

$A_0 = \{[f_1]a_1[f_4], [f_2]a_2[f_5]\}$

$\mathbf{S_1} = \{f_1,f_2,f_3,f_4,f_5\}$

$A_1 = \{[f_2,f_4,f_5]a_3[f_6]\}$

$S_2 = \{f_1,f_2,f_3,f_4,f_5,f_6\}$

$G_1 = \{f_5,f_1,f_2,f_4\}$

We split G_1 into G_P and G_N:

CountActs(G_1,S_1)

Example

Now, we are at level S1

$S_0 = \{f_1, f_2, f_3\}$

$A_0 = \{[f_1]a_1[f_4], [f_2]a_2[f_5]\}$

$\mathbf{S_1} = \{f_1,f_2,f_3,f_4,f_5\}$

$A_1 = \{[f_2,f_4,f_5]a_3[f_6]\}$

$S_2 = \{f_1,f_2,f_3,f_4,f_5,f_6\}$

$G_1 = \{f_5,f_1,f_2,f_4\}$

We split G_1 into G_P and G_N:

$\mathbf{G_N} = \{f_5,f_4\}$

$\mathbf{G_P} = \{f_1,f_2\}$

CountActs(G_1,S_1)

$G_P = \{f_1,f_2\}$ **//already in S0**

$G_N = \{f_4,f_5\}$ **//New in S1**

$A = \{a_1,a_2\}$ **//adds all in G_N**

//the new goal: $G_P \cup$ Pre(A)

$G_2 = \{f_1,f_2\}$

Return

2 + CountActs(G_2,S_0)

Example

Now, we are at level S1

$S_0 = \{f_1, f_2, f_3\}$

$A_0 = \{[f_1]a_1[f_4], [f_2]a_2[f_5]\}$

$S_1 = \{f_1,f_2,f_3,f_4,f_5\}$

$A_1 = \{[f_2,f_4,f_5]a_3[f_6]\}$

$S_2 = \{f_1,f_2,f_3,f_4,f_5,f_6\}$

$G_2 = \{f_1,f_2\}$

We split G_2 into G_P and G_N:

$\mathbf{G_N} = \{f_1,f_2\}$

$\mathbf{G_P} = \{\}$

CountActs(G_2,S_0)

$G_N = \{f_1,f_2\}$ **//already in S0**

$G_P = \{\}$ **//New in S1**

$A = \{\}$ **//No actions needed.**

Return

0

Example

Now, we are at level S1

$S_0 = \{f_1, f_2, f_3\}$

$A_0 = \{[f_1]a_1[f_4], [f_2]a_2[f_5]\}$

$\mathbf{S_1} = \{f_1,f_2,f_3,f_4,f_5\}$

$A_1 = \{[f_2,f_4,f_5]a_3[f_6]\}$

$S_2 = \{f_1,f_2,f_3,f_4,f_5,f_6\}$

$G_2 = \{f_1,f_2\}$

We split G_2 into G_P and G_N:

$\mathbf{G_N} = \{f_1,f_2\}$

$\mathbf{G_P} = \{\}$

CountActs(G_2,S_0)

$G_N = \{f_1,f_2\}$ **//already in S0**

$G_P = \{\}$ **//New in S1**

$A = \{\}$ **//No actions needed.**

Return

0

So, in total CountActs(G,S2)=1+2+0=3

Using the Heuristic

- First, build a layered structure from a state S that reaches a goal state.
- CountActions: counts how many actions are required in a relaxed plan.
  - Use this as our heuristic estimate of the distance of S to the goal.
  - This heuristic tends to work better with greedy best-first search rather than A* search
    - That is when we ignore the cost of getting to the current state.

Admissibility

- A minimum sized plan in the delete relaxed problem would be a lower bound on the optimal size of a plan in the real problem. And could serve as an admissible heuristic for A*.
- However, CountActions **does NOT compute** the length of the optimal relaxed plan.
  - The choice of which *action set* to use to achieve G_P ("just achieved part of G") is not necessarily optimal – it is minimal, but not necessary a minimum.
  - Furthermore even if we picked a true minimum set A at each stage of CountActions, we might not obtain a minimum set of actions for the entire plan---the set A picked at each state influences what set can be used at the next stage!

Admissibility

- It is NP-Hard to compute the optimal length plan even in the relaxed plan space.
  - So CountActions cannot be made into an admissible heuristic without making it much harder to compute.
  - Empirically, refinements of CountActions performs very well on a number of sample planning domains.