Artificial Intelligence

Planning 2: Abstraction Heuristics

Jendrik Seipp

Linköping University

based on slides by Thomas Keller and Malte Helmert (University of Basel)

Introduction

Planning Heuristics

General Procedure for Obtaining a Heuristic

Solve a simplified version of the problem.

there are many ideas for domain-independent planning heuristics:

- abstraction $\rightsquigarrow$ now
- delete relaxation $\rightsquigarrow$ tomorrow
- landmarks
- critical paths
- network flows
- potential heuristics

Planning Heuristics

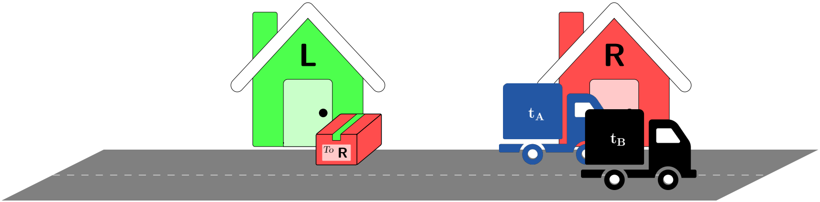
Abstraction: Idea

Estimate solution costs by considering a smaller planning task where states are merged to abstract states.

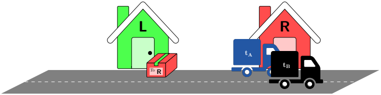
there are many ideas for domain-independent planning heuristics:

- abstraction $\rightsquigarrow$ now
- delete relaxation $\rightsquigarrow$ tomorrow
- landmarks
- critical paths
- network flows
- potential heuristics

Example: Logistics Task with One Package, Two Trucks

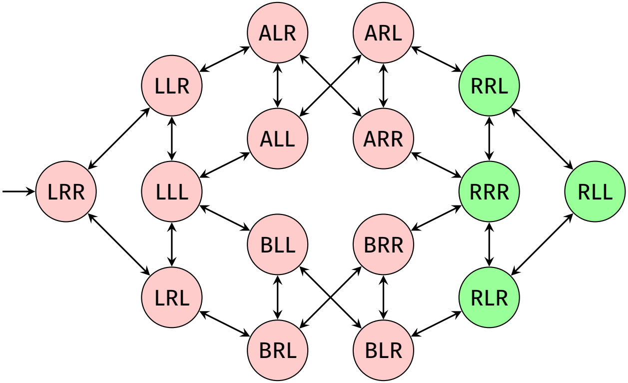


Example: Logistics Task with One Package, Two Trucks



- $V = \{p, t_A, t_B\}$
- $dom(p) = \{L, R, A, B\}$ and $dom(t_A) = dom(t_B) = \{L, R\}$
- $I = \{p \mapsto L, t_A \mapsto R, t_B \mapsto R\}$
- $G = \{p \mapsto R\}$
- $A = \{load_{i,j} \mid i \in \{A, B\}, j \in \{L, R\}\}$
  $\cup \{unload_{i,j} \mid i \in \{A, B\}, j \in \{L, R\}\}$
  $\cup \{move_{i,j,j'} \mid i \in \{A, B\}, j, j' \in \{L, R\}, j \neq j'\}$ with:
  - $load_{i,j}$ has preconditions $\{t_i \mapsto j, p \mapsto j\}$, effects $\{p \mapsto i\}$
  - $unload_{i,j}$ has preconditions $\{t_i \mapsto j, p \mapsto i\}$, effects $\{p \mapsto j\}$
  - $move_{i,j,j'}$ has preconditions $\{t_i \mapsto j\}$, effects $\{t_i \mapsto j'\}$
  - all actions have cost 1

State Space for Example Task



- state $\{p \mapsto i, t_{\mathsf{A}} \mapsto j, t_{\mathsf{B}} \mapsto k\}$ denoted as ijk
- annotations of edges not shown for simplicity
- for example, edge from LLL to ALL has annotation $load_{\mathsf{A},\mathsf{L}}$

Abstractions

Abstraction

abstractions drop distinctions between certain states,
but preserve the state space behavior as well as possible.

- an abstraction of a state space $\mathcal{S}$ is defined by an abstraction function α that determines which states can be distinguished in the abstraction
- based on $\mathcal{S}$ and α, we compute the abstract state space $\mathcal{S}^\alpha$ which is "similar" to $\mathcal{S}$ but smaller

idea of the abstraction heuristic h^α:
use abstract solution costs (solution costs in $\mathcal{S}^\alpha$)
as heuristic values for concrete solution costs (solution costs in $\mathcal{S}$)

Induced Abstraction

Definition (induced abstraction)

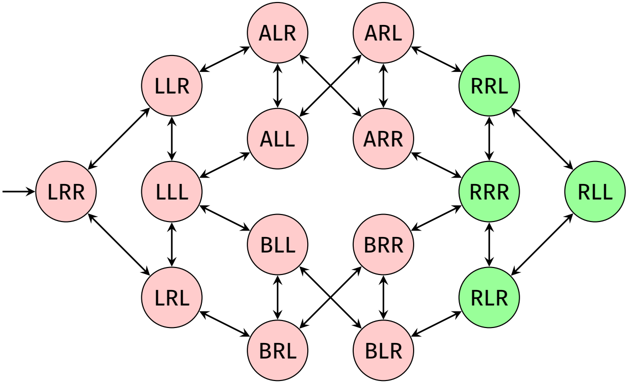
Let $\mathcal{S} = \langle S, A, cost, T, s_0, S_\star \rangle$ be a state space,
and let $\alpha : S \longrightarrow S'$ be a surjective function.

The abstraction of $\mathcal{S}$ induced by α, denoted as $\mathcal{S}^\alpha$,
is the state space $\mathcal{S}^\alpha = \langle S', A, cost, T', s'_0, S'_\star \rangle$ with:

- $T' = \{\langle \alpha(s), a, \alpha(t) \rangle \mid \langle s, a, t \rangle \in T\}$
- $s'_0 = \alpha(s_0)$
- $S'_\star = \{\alpha(s) \mid s \in S_\star\}$

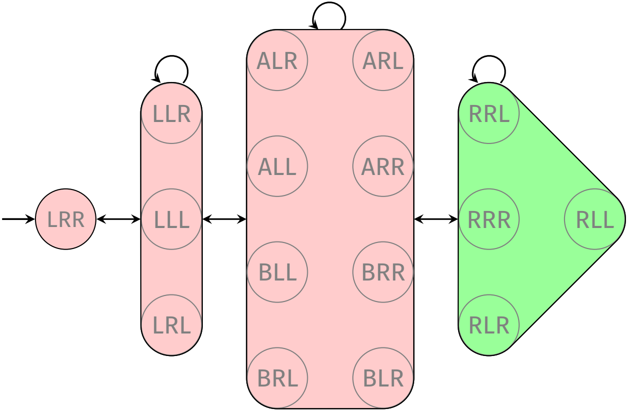
Abstraction: Example

concrete state space



Abstraction: Example

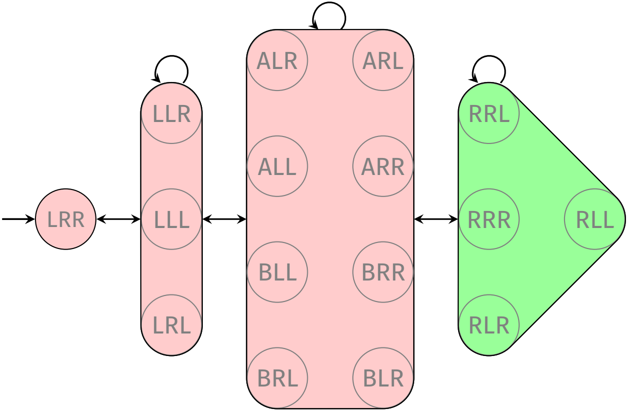
(an) abstract state space



remark: most edges correspond to several (parallel) transitions with different annotations

Abstraction Heuristic: Example

(an) abstract state space

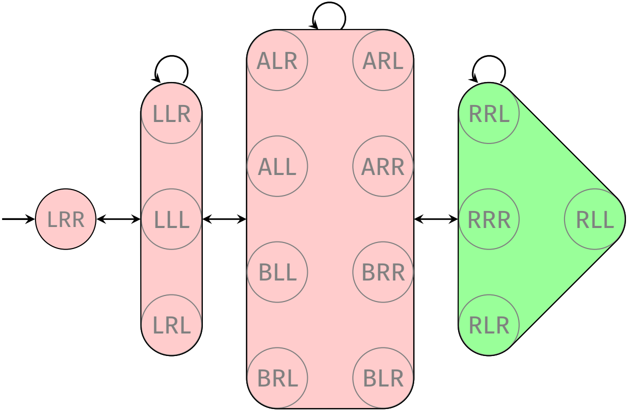


$h^{\alpha}(LRR) = ?,$
$h^{\alpha}(LLR) = ?,$
$h^{\alpha}(LLL) = ?,$
$h^{\alpha}(LRL) = ?,$
$h^{\alpha}(ALR) = ?,$
$\ldots$

remark: most edges correspond to several (parallel) transitions with different annotations

Abstraction: Example

(an) abstract state space



$h^{\alpha}(LRR) = 3,$
$h^{\alpha}(LLR) = 2,$
$h^{\alpha}(LLL) = 2,$
$h^{\alpha}(LRL) = 2,$
$h^{\alpha}(ALR) = 1,$
$\ldots$

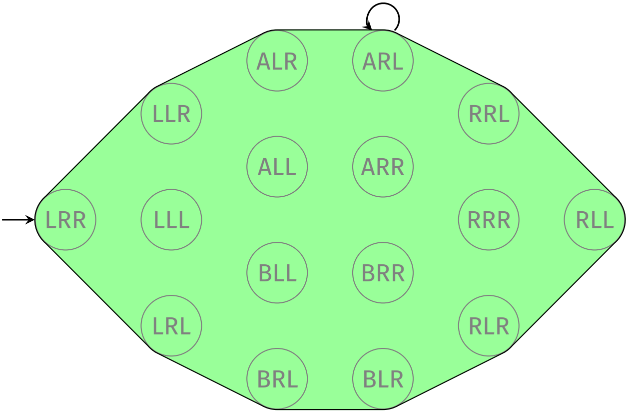
remark: most edges correspond to several (parallel) transitions with different annotations

Abstraction Heuristics: Discussion

- every abstraction heuristic is admissible and consistent
- the choice of the abstraction function α is very important
  - every α yields an admissible and consistent heuristic
  - but most α lead to poor heuristics
- a “good” α must yield an informative heuristic ...
- ...as well as being efficiently computable

How can we find a suitable α?

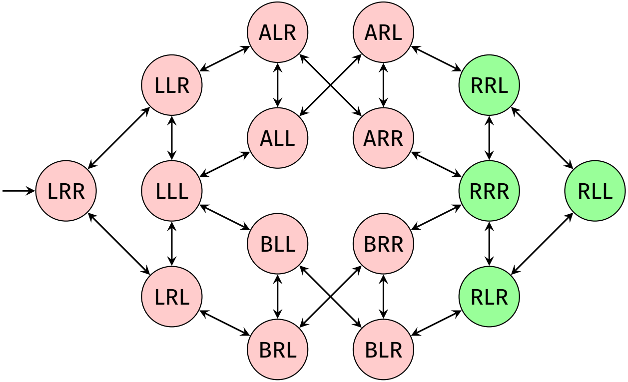
Usually a Bad Idea: Single-State Abstraction



one state abstraction: $\alpha(s) :=$ const

- + compactly representable and α easy to compute
- – very uninformed heuristic

Usually a Bad Idea: Identity Abstraction



identity abstraction: $\alpha(s) := s$

- \+ perfect heuristic and α easy to compute
- − too many abstract states $\rightsquigarrow$ computation of h^α too hard

Automatic Computation of Suitable Abstractions

Main Problem with Abstraction Heuristics

How to find a good abstraction?

several successful methods:

- pattern databases (PDBs) (Culberson & Schaeffer, 1996)
- merge-and-shrink abstractions (Dräger, Finkbeiner & Podelski, 2006)
- Cartesian abstractions (Seipp & Helmert, 2013)

Pattern Databases

Pattern Databases: Background

- the most common abstraction heuristics are pattern database heuristics
- originally introduced for the 15-puzzle (Culberson & Schaeffer, 1996) and for Rubik's Cube (Korf, 1997)
- introduced for automated planning by Edelkamp (2001)
- for many search problems the best known heuristics
- many research papers studying
  - theoretical properties
  - efficient implementation and application
  - pattern selection
  - ...

Pattern Databases: Projections

a PDB heuristic is an abstraction heuristic where

- some aspects (= state variables) of the task are preserved with perfect precision while
- all other aspects are not preserved at all

formalized as projections on a pattern P, e.g.:

- $s = \{v_1 \mapsto d_1, v_2 \mapsto d_2, v_3 \mapsto d_3\}$
- projection on $P = \{v_1\}$ (= ignore v_2, v_3):
  $\alpha(s) = s|_P = \{v_1 \mapsto d_1\}$
- projection on $P = \{v_1, v_3\}$ (= ignore v_2):
  $\alpha(s) = s|_P = \{v_1 \mapsto d_1, v_3 \mapsto d_3\}$

Pattern Databases: Definition

Definition (pattern database heuristic)

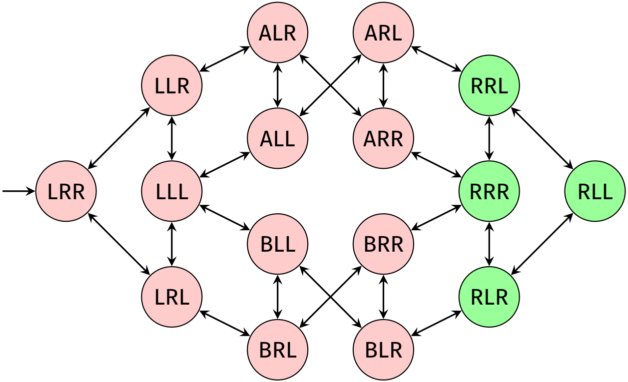
Let P be a subset of the variables of a planning task.

The abstraction heuristic induced by the projection π_P on P is called pattern database heuristic (PDB heuristic) with pattern P.

abbreviated notation: h^P for h^{π_P}

remark: “pattern databases” in analogy to endgame databases (which have been successfully applied in 2-person-games)

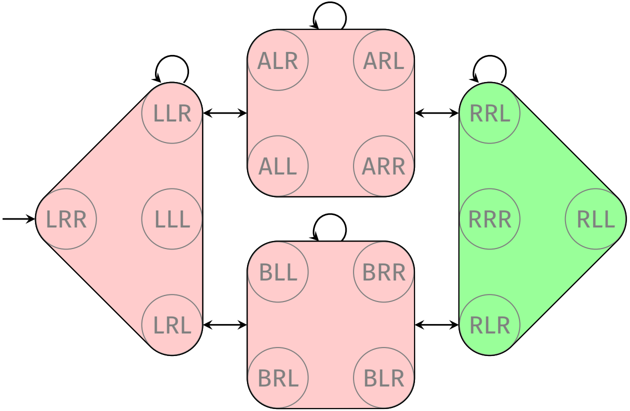
Example: Concrete State Space



- state variable for package, p: {L, R, A, B}
- state variable for truck A, t_A: {L, R}
- state variable for truck B, t_B: {L, R}

Example: Projection (1)

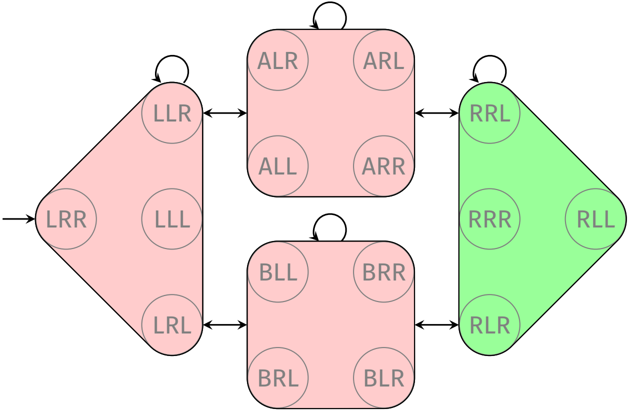
abstraction induced by $\pi_{\{p\}}$:



$$
\begin{aligned}
h^{\{p\}}(LRR) &= 2, \\
h^{\{p\}}(LLR) &= 2, \\
h^{\{p\}}(LLL) &= 2, \\
h^{\{p\}}(LRL) &= 2, \\
h^{\{p\}}(ALR) &= 1, \\
&\ldots
\end{aligned}
$$

Example: Projection (1)

abstraction induced by $\pi_{\{p\}}$:



PDB :

$\{p \mapsto L\} = 2,$

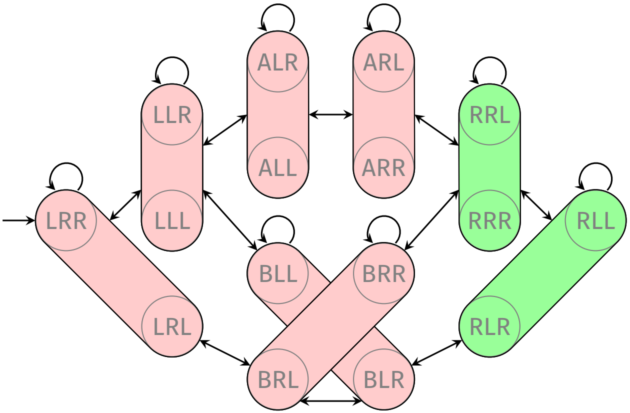
$\{p \mapsto A\} = 1,$

$\{p \mapsto B\} = 1,$

$\{p \mapsto R\} = 0$

Example: Projection (2)

abstraction induced by $\pi_{\{p,t_A\}}$:



$h^{\{p,t_A\}}(LRR) = 2,$

$h^{\{p,t_A\}}(LLR) = 2,$

$h^{\{p,t_A\}}(LLL) = 2,$

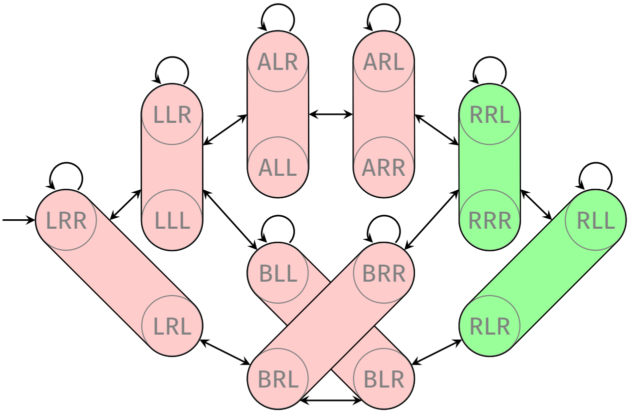
$h^{\{p,t_A\}}(LRL) = 2,$

$h^{\{p,t_A\}}(ALR) = 2,$

$\ldots$

Example: Projection (2)

abstraction induced by $\pi_{\{p, t_A\}}$:



PDB :

$\{p \mapsto L, t_A \mapsto L\} = 2,$

$\{p \mapsto L, t_A \mapsto R\} = 2,$

$\{p \mapsto A, t_A \mapsto L\} = 2,$

$\{p \mapsto A, t_A \mapsto R\} = 1,$

$\{p \mapsto B, t_A \mapsto L\} = 1,$

$\{p \mapsto B, t_A \mapsto R\} = 1,$

$\{p \mapsto R, t_A \mapsto L\} = 0,$

$\{p \mapsto R, t_A \mapsto R\} = 0$

Pattern Databases in Practice

practical aspects which we do not discuss in detail:

- How to automatically find good patterns?
- How to combine multiple PDB heuristics?
- How to implement PDB heuristics efficiently?
  - good implementations efficiently handle abstract state spaces with 10^7, 10^8 or more abstract states
  - effort independent of the size of the concrete state space
  - usually all heuristic values are precomputed
    $\rightsquigarrow$ space complexity = number of abstract states

Quiz

Kahoot!

Summary

planning formalisms:

- STRIPS: particularly simple, easy to handle for algorithms
  - binary state variables
  - preconditions, add and delete effects, goals: sets of variables
- SAS^+: extension of STRIPS
  - state variables with arbitrary finite domains
- PDDL: input language used in practice
  - based on predicate logic (more compact than propositional logic)
  - only partly supported by most algorithms

abstraction heuristics:

- estimate solution cost by considering a smaller planning task
- Pattern database heuristics are abstraction heuristics based on projections onto state variable subsets (patterns): states are distinguishable iff they differ on the pattern.