A Beginner’s Introduction to Heuristic Search Planning

5. Abstraction Heuristics and Pattern Databases

Malte Helmert       Gabriele Röger

AAAI 2015 Tutorial

January 25, 2015
Abstraction Heuristics

Projections and Pattern Database Heuristics

Pattern Collections and iPDB

Summary
Abstraction Heuristics: Idea

- Heuristic estimate = plan cost in simplified state space
- Simplification: Do not distinguish all states
Abstraction: Example

- **state variable** package: \{L, R, A, B\}
- **state variable** truck A: \{L, R\}
- **state variable** truck B: \{L, R\}
Abstraction: Example

(an) abstract state space

Remark: Most edges correspond to several parallel transitions with different labels.
Abstraction: Example

\[ h^\alpha(\{p \mapsto L, t_A \mapsto R, t_B \mapsto R\}) = 3 \]
Abstract state space is derived from original state space as specified by an abstraction function.

Abstraction function defines which states should be distinguished.
Abstraction Heuristics

- Abstract state space is derived from original state space as specified by an abstraction function
- Abstraction function defines which states should be distinguished
- Preserve all original paths in abstract state space
- Do not relax more than required by abstraction function
Induced Abstraction

**Definition (induced abstraction)**

Let $S = \langle S, s_0, S_*, A, cost, T \rangle$ be a state space and let $\alpha : S \rightarrow S'$ be a surjective function.

The **abstraction of $S$ induced by $\alpha$** is the state space $S^\alpha = \langle S', s'_0, S'_*, A, cost, T' \rangle$ with:

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

For state space $S$ and abstraction function $\alpha$, the heuristic estimate $h^{\alpha}(s)$ for state $s$ is the cost of a cheapest path from $\alpha(s)$ to a goal state in $S^{\alpha}$. 
Abstraction Heuristic

**Definition (abstraction heuristic)**

For state space $S$ and abstraction function $\alpha$, the heuristic estimate $h^{\alpha}(s)$ for state $s$ is the cost of a cheapest path from $\alpha(s)$ to a goal state in $S^\alpha$.

Abstraction heuristics are admissible and consistent.


Classes of Abstractions

- **Projections**
  - Simple abstractions used for pattern databases
    (Culberson & Schaeffer, Computational Intelligence 1998; Edelkamp, ECP 2001; Haslum et al., AAAI 2007)

- **Merge & Shrink abstractions**
  - Can represent arbitrary abstractions
    (Dräger et al., SPIN 2006; Helmert et al., ICAPS 2007; Sievers et al. AAAI, 2014)

- **Cartesian abstractions**
  - Generalization of projections
    (Seipp & Helmert, ICAPS 2013; ICAPS 2014)

- **Structural patterns**
  - Easy to solve despite being large
    (Katz & Domshlak, ICAPS 2008)
Classes of Abstractions

- **Projections**
  - Simple abstractions used for pattern databases
    (Culberson & Schaeffer, Computational Intelligence 1998; Edelkamp, ECP 2001; Haslum et al., AAAI 2007)

- **Merge & Shrink abstractions**
  - Can represent arbitrary abstractions
    (Dräger et al., SPIN 2006; Helmert et al., ICAPS 2007; Sievers et al. AAAI, 2014)

- **Cartesian abstractions**
  - Generalization of projections
    (Seipp & Helmert, ICAPS 2013; ICAPS 2014)

- **Structural patterns**
  - Easy to solve despite being large
    (Katz & Domshlak, ICAPS 2008)
Projections and Pattern Database Heuristics
Pattern database heuristics

Pattern database (PDB) heuristics

- represent some aspects (\(\equiv\) state variables) perfectly, but
- entirely ignore all other aspects

Example (15-puzzle)

- Choose subset \(P\) of tiles (the pattern).
- In the abstract state space... 
  - consider the exact position of all tiles in \(P\),
  - assume that all other tiles and the blank position can be everywhere.
PDB heuristics are abstraction heuristics where the abstraction function is a **projection**.

**Definition (Projection)**

Let $\Pi$ be a SAS$^+$ planning task with variables $V$ and states $S$. For $P \subseteq V$ let $S'_P$ be the set of partial variable assignments that are defined exactly on $P$.

The **projection** $\pi_P : S \rightarrow S'_P$ is defined as $\pi_P(s) := s|_P$

(with $s|_P(v) := s(v)$ for all $v \in P$).

Put differently: $\pi_P$ maps two concrete states to the same abstract state iff they agree on all variables in $P$. 


Example: concrete state space
Example: Projection (1)

Abstraction induced by $\pi\{\text{package}\}$:

$h\{\text{package}\}(\text{LRR}) = 2$
Example: Projection (2)

Abstraction induced by $\pi\{\text{package, truck A}\}$:

$$h^{\{\text{package, truck A}\}}(\text{LRR}) = 2$$
Example: Projection (2)

Abstraction induced by $\pi\{\text{package, truck A}\}$:

$h\{\text{package, truck A}\}(\text{LRR}) = 2$
Example: Projection (3)

Abstraction induced by $\pi\{\}$:

$h_{\{package, truck A\}}(LRR) = 0$
Abstraction induced by $\pi\{\text{package, truck A, truck B}\}$:

$h\{\text{package, truck A}\}(\text{LRR}) = 4$
Pattern Collections and iPDB
Pattern Collections

- Multiple PDB heuristics can be combined into one heuristic estimate
Pattern Collections

- **Multiple PDB heuristics** can be combined into one heuristic estimate
- **Pattern collection**: Set of patterns defining a collection of PDB heuristics
Pattern Collections

- **Multiple PDB heuristics** can be combined into one heuristic estimate
- **Pattern collection**: Set of patterns defining a collection of PDB heuristics
- **Canonical heuristic**: Maximizes where necessary to stay admissible, sums up where possible *(Haslum et al., AAAI 2007)*
Pattern Collections

- **Multiple PDB heuristics** can be combined into one heuristic estimate
- **Pattern collection**: Set of patterns defining a collection of PDB heuristics
- **Canonical heuristic**: Maximizes where necessary to stay admissible, sums up where possible (Haslum et al., AAAI 2007)
- **Post-hoc optimization**: Solves linear program (LP) to determine admissible weighted sum of individual estimates (Pommerening et al., IJCAI 2013)
Pattern Collections

- Multiple PDB heuristics can be combined into one heuristic estimate

- Pattern collection: Set of patterns defining a collection of PDB heuristics

- Canonical heuristic: Maximizes where necessary to stay admissible, sums up where possible (Haslum et al., AAAI 2007)

- Post-hoc optimization: Solves linear program (LP) to determine admissible weighted sum of individual estimates (Pommerening et al., IJCAI 2013)

- Optimal cost partitioning: Solves very large LP to suitably adjust cost functions of the individual abstractions (Katz & Domshlak, AIJ 2010)
How to Find a Good Pattern Collection for a Task?

Example: iPDB (Haslum et al., AAAI 2007)

- **Hill-climbing search** in the space of pattern collections
- Optimizing estimates of **canonical heuristic**
- **Initial pattern collection**: \( \{ \{ v \} \mid v \text{ is goal variable} \} \)
- **Search neighborhood**: Add one new pattern which
  - extends an already included pattern with one variable...
  - ...that can improve the heuristic estimate according to some relevance criterion, and...
  - ...the resulting PDBs fit into a prespecified memory limit
- **Stop** if no successor improves the heuristic estimate

In Fast Downward

ipdb(...)
Hands on: iPDB in Fast Downward

Hands-On

$ cd hands-on
$ ./fd ipc/logistics00/probLOGISTICS-6-0.pddl \
   --search "astar(ipdb())"
Summary
Summary

- **abstraction heuristics**: map state space to smaller space
- **projections**: abstraction functions that perfectly represent some state variables (given by the pattern) and entirely ignore all others
- **pattern database (PDB) heuristic**: abstraction heuristic using a projection
- **combining PDBs**: better heuristic estimates
- **pattern selection**
  - precomputation time and memory vs. heuristic quality
  - usually task-specific, e.g. with iPDB