Automatically Deriving Abstraction Heuristics

Malte Helmert
Albert-Ludwigs-Universität Freiburg, Germany

STAIR 2008
About This Talk

Abstraction heuristics
Heuristics estimate is goal distance in abstracted state space $S'$ obtained as homomorphism of original state space $S$.

Canonical example: pattern databases

Abstraction heuristics in the search community
A lot of thought has gone into developing (and analyzing) effective abstraction heuristics for particular search problems ($n^2 - 1$-puzzle, Rubik's Cube, Top Spin, ...).

This talk is about applying abstraction heuristics to problems where the search space is unknown to the algorithm designer.
Outline

1. Transition Systems and Abstractions
2. Automatically Derived PDB Abstractions
3. Automatically Derived Explicit-State Abstractions
4. Conclusion
Definition (transition system)

A transition system is a 5-tuple $\langle S, L, A, s_0, S_\ast \rangle$:

- $S$: finite set of states
- $L$: finite set of transition labels
- $A \subseteq S \times L \times S$: labelled transitions
- $s_0 \in S$: initial state
- $S_\ast \subseteq S$: goal states

Objective: Find a shortest path from $s_0$ to some $s_\ast \in S_\ast$. 
Factored Transition Systems

We assume a factored representation of transition systems:

- **states**: assignments to set $\mathcal{V}$ of state variables
- **transitions and labels**: given by set of operators defined in terms of a condition and effect on subsets of $\mathcal{V}$
- **goal states**: given by assignment to $\mathcal{V}' \subseteq \mathcal{V}$
Example: Blocksworld
Example: Pipesworld
Example: FreeCell
Abstractions

Definition (abstraction, homomorphism)

Abstraction of transition system $\mathcal{T}$: pair $\langle \mathcal{T}', \alpha \rangle$ where

- $\mathcal{T}'$ is a transition system with the same labels
- $\alpha$ maps states of $\mathcal{T}$ to states of $\mathcal{T}'$ such that
  - initial state maps to initial state
  - goal states map to goal states
  - transitions $\langle s, l, s' \rangle$ map to transitions $\langle \alpha(s), l, \alpha(s') \rangle$

Abstraction (and $\alpha$) is a homomorphism if $\mathcal{T}'$ only contains necessary goal states and transitions.

Abstraction heuristic: $h(s) = d_\star(\alpha(s))$ admissible, consistent
Generating Abstractions

Conflicting goals in generating abstractions:
- obtain informative heuristic
- keep representation small

Abstractions have small representations if they have
- few abstract states
- succinct encoding for $\alpha$
One idea to get succinct encodings: **projections**

\[ \rightsquigarrow \text{map states to abstract states with perfect hash function} \]

**Definition (projection)**

Projection \( \pi_{\mathcal{V}'} \) to variables \( \mathcal{V}' \subseteq \mathcal{V} \):

homomorphism \( \alpha \) where \( \alpha(s) = \alpha(s') \) iff \( s \) and \( s' \) agree on \( \mathcal{V}' \)

Abstraction heuristics for projections are called **pattern database (PDB)** heuristics.
Example: Transition System

Logistics problem with one package, two trucks, two locations:

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

Project to \{\textit{package}\}:
Automatically Derived Abstraction Heuristics

Our research problem

Automatically derive an effective abstraction heuristic for a given transition system in factored representation.

Some important papers:

- Edelkamp (ECP-01): Planning with PDBs
- Edelkamp (AIPS-02): Symbolic PDBs
- Haslum et al. (AAAI-05): Constrained PDBs
- Haslum et al. (AAAI-07): Pattern selection
- Helmert et al. (ICAPS-07): Explicit-state abstractions
- Katz & Domshlak (ICAPS-08): Optimal cost partitioning
- Katz & Domshlak (ICAPS-08): Structural patterns
Automatically Derived Abstraction Heuristics

Our research problem

Automatically derive an effective abstraction heuristic for a given transition system in factored representation.

Some important papers:

- Edelkamp (ECP-01): Planning with PDBs
- Edelkamp (AIPS-02): Symbolic PDBs
- Haslum et al. (AAAI-05): Constrained PDBs
- Haslum et al. (AAAI-07): Pattern selection
- Helmert et al. (ICAPS-07): Explicit-state abstractions
- Katz & Domshlak (ICAPS-08): Optimal cost partitioning
- Katz & Domshlak (ICAPS-08): Structural patterns
This part based on:

Patrik Haslum, Adi Botea, Malte Helmert, Blai Bonet, Sven Koenig.

PDB Abstractions for Factored Transition Systems

Objective

Automatically derive an effective pattern database heuristic for a given transition system in factored representation.

Guiding questions:

1. What is a pattern for a factored transition system?
2. How can we identify and exploit disjunctive patterns?
3. Which patterns do we choose?
Patterns for Factored Transition Systems

1. What is a pattern for a factored transition system?

Most natural definition:
- identify patterns with sets of state variables to project to

**Definition (abstracted transition system)**

Let $\mathcal{T}$ be a factored transition system and $\mathcal{V}$ its variable set. Let $P \subseteq \mathcal{V}$ be a pattern.

The abstracted transition system $\mathcal{T}(P)$ is obtained from $\mathcal{T}$ by:
- restricting the initial state to $P$
- restricting operator conditions and effects to $P$
- removing goal conditions on variables not in $P$
Definition (pattern heuristic)

Let $T$ be a factored transition system and $\mathcal{V}$ its variable set. Let $P \subseteq \mathcal{V}$ be a pattern.

The pattern heuristic $h^P$ assigns to each state $s$ of $T$ the length of an optimal solution for $T(P)$, starting from the state obtained by restricting $s$ to $P$.

For all choices of $P$, heuristic $h^P$ is admissible and consistent.

What can we do if we have multiple patterns $P_1, \ldots, P_k$?
How can we identify and exploit disjunctive patterns?

**Theorem (disjunctive patterns)**

Let $\mathcal{C}$ be a pattern collection, i.e. a set of patterns of task $\mathcal{T}$. We say that an operator affects a pattern $P$ if it can assign a new value to some variable $v \in P$.

If no operator in $\mathcal{T}$ affects more than one pattern in $\mathcal{C}$, then $\sum_{P \in \mathcal{C}} h^P$ is admissible and consistent.
Finding Disjunctive Patterns

Finding sets of disjunctive patterns in a pattern collection $C$:

- build *compatibility graph* for $C$
  - vertices correspond to patterns $P \in C$
  - edge between two vertices iff no operator affects both
- compute *all maximal cliques* of the graph
  using the algorithm of Tomita, Tanaka & Takahashi
The Canonical Heuristic Function

Definition (canonical heuristic function)

Let $\mathcal{T}$ be a factored transition system and $\mathcal{V}$ its variable set. Let $\mathcal{C}$ be a pattern collection.

The canonical heuristic $h^C$ for pattern collection $\mathcal{C}$ is defined as

$$h^C(s) = \max_{D \in \text{cliques}(\mathcal{C})} \sum_{P \in D} h^P(s),$$

where $\text{cliques}(\mathcal{C})$ is the set of all maximal cliques in the compatibility graph for $\mathcal{C}$.

For all choices of $\mathcal{C}$, heuristic $h^C$ is admissible and consistent.

It is the best possible admissible heuristic that can be derived from the information in the pattern databases in $\mathcal{C}$.

The full story includes “dominance pruning” to optimize speed.
Building the Pattern Collection

3. **Which patterns do we choose?**

~~ perform local search in the space of pattern collections

- **search method:**
  - hill-climbing (steepest ascent)

- **initial collection:**
  - \{\{v\} \mid v \in V, v \text{ has a goal condition}\}

- **search neighbourhood:**
  - for each pattern \(P \in C\) and each variable \(v \in V\), the collection \(C \cup \{P \cup \{v\}\}\) is a neighbour
  - unless the pattern database for \(P \cup \{v\}\) would exceed a pre-specified memory limit

- **evaluation function:**
  - estimate heuristic quality of collection (next slide)
for pattern collection $C$, want to estimate the quality of $h^C$

only need to estimate degree of improvement:

- For neighbours $C_1$ and $C_2$ of $C$, which of $h^{C_1}$ and $h^{C_2}$ leads to the larger improvement over $h^C$?

using an analytical model for heuristic search performance by Korf, Reid & Edelkamp (and some simplifying assumptions), this is reduced to:

- Which of $h^{C_1}$ and $h^{C_2}$ has a higher probability of giving a better estimate than $h^C$ for randomly drawn states?

    using a particular non-uniform random distribution that I do not want to discuss in detail...

$\leadsto$ problem reduces to computing $h^{C_i}(s)$ for some states $s$
Estimating Heuristic Quality (ctd.)

\[ \sim \rightarrow \text{problem reduces to computing } h^{C_i}(s) \text{ for some states } s \]

- Idea: compute \( h^{C_i}(s) \) without computing the pattern database for each new pattern (too expensive)

\[ \sim \rightarrow \text{perform } A^* \text{ searches in the state space of } h^{P \cup \{v\}} \]
using \( h^P \) as a heuristic
Empirical Evaluation

We tested the approach on
- 24 instances of the 15-puzzle and
- 40 Sokoban instances.
Empirical Evaluation: Results

15-puzzle:
- we can solve all 24 instances optimally
- compared to the only other known general algorithm which manages this (Haslum, Bonet & Geffner 2005):
  - their technique: 2,559,508 node expansions
  - our technique: 549,147 node expansions

Sokoban:
- we can solve 23 out of 40 instances optimally
- we are not aware of any other general algorithm which can solve any of these optimally
Outline

1. Transition Systems and Abstractions
2. Automatically Derived PDB Abstractions
3. Automatically Derived Explicit-State Abstractions
4. Conclusion
This part based on:

Malte Helmert, Patrik Haslum, Jörg Hoffmann. Flexible Abstraction Heuristics for Optimal Sequential Planning.

Example: Transition System

Logistics problem with one package, two trucks, two locations:

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

Project to \{\text{package}\}:
Example: Projection (2)

Project to \{\text{package, truck A}\}:
Example: Projection (2)

Project to \{package, truck A\}:
Limitations of Projections

How accurate is the PDB heuristic?

- consider generalization of the example:
  $N$ trucks, $M$ locations (still one package)
- consider any pattern that is proper subset of $\mathcal{V}$
- $h(s_0) \leq 2 \leadsto$ no better than atomic projection to package

(maximizing over patterns or disjunctive patterns do not help either)
Explicit-State Abstraction Heuristics: Main Idea

Main idea

(due to Dräger, Finkbeiner & Podelski, 2006):

Instead of perfectly reflecting a few state variables, reflect all state variables, but in a potentially lossy way.
Explicit-State Abstraction Heuristics: Key Insights

Key insights:

1. Information of two abstractions $\mathcal{A}$ and $\mathcal{A}'$ of the same transition system can be composed by a simple graph-theoretic operation (synchronized product $\mathcal{A} \otimes \mathcal{A}'$).

2. The complete state space can be recovered using only atomic projections:

$$\bigotimes_{v \in \mathcal{V}} \pi_v$$ is isomorphic to $\pi_{\mathcal{V}}$.

$\leadsto$ build fine-grained abstractions from coarse ones

3. When intermediate results become too big, we can shrink them by aggregating some abstract states.
Generic abstraction computation algorithm

\[
\text{abs} := \text{all atomic projections } \pi_v \ (v \in \mathcal{V}).
\]

\[
\text{while abs contains more than one abstraction:}
\]

\[
\begin{align*}
\text{select } A_1, A_2 \text{ from abs} \\
\text{shrink } A_1 \text{ and/or } A_2 \text{ until } \text{size}(A_1) \cdot \text{size}(A_2) \leq N \\
\text{abs} := \text{abs} \setminus \{A_1, A_2\} \cup \{A_1 \otimes A_2\}
\end{align*}
\]

\[
\text{return the remaining abstraction}
\]

\[N: \text{parameter bounding number of abstract states}\]

Questions for practical implementation:

- Which abstractions to select? \(\rightsquigarrow\) composition strategy
- How to shrink an abstraction? \(\rightsquigarrow\) shrinking strategy
- How to choose \(N\)?
Experimental Evaluation

### Comparison to state of the art

**Is this competitive with the state of the art?**

- Compare scaling behaviour to other heuristics: blind, $h^{\text{max}}$, PDB

<table>
<thead>
<tr>
<th>Domain</th>
<th>abs</th>
<th>blind</th>
<th>$h^{\text{max}}$</th>
<th>PDB</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pipes-NoTankage</strong></td>
<td>19</td>
<td>14</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td><strong>Pipes-Tankage</strong></td>
<td>13</td>
<td>10</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td><strong>Satellite</strong></td>
<td>6</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td><strong>Logistics</strong></td>
<td>18</td>
<td>6</td>
<td>6</td>
<td>16</td>
</tr>
<tr>
<td><strong>PSR</strong></td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td><strong>TPP</strong></td>
<td>7</td>
<td>5</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td><strong>total</strong></td>
<td>68</td>
<td>42</td>
<td>46</td>
<td>54</td>
</tr>
</tbody>
</table>
### Theoretical Properties

<table>
<thead>
<tr>
<th>As powerful as PDBs</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDB heuristics are a special case of our abstraction heuristics, and arise naturally as a side product.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Get additivity for free</th>
</tr>
</thead>
<tbody>
<tr>
<td>If $P$ and $P'$ are disjunctive patterns, then for all $h$-preserving abstractions $A$ of $\pi_P$ and $A'$ of $\pi_{P'}$, the abstraction heuristic for $A \otimes A'$ dominates $h^P + h^{P'}$.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Greater representational power</th>
</tr>
</thead>
<tbody>
<tr>
<td>In some planning domains where PDBs have unbounded error (Gripper, Schedule, two Promela variants), we can obtain perfect heuristics in polynomial time with suitable composition/shrinking strategies.</td>
</tr>
</tbody>
</table>
Outline

1. Transition Systems and Abstractions
2. Automatically Derived PDB Abstractions
3. Automatically Derived Explicit-State Abstractions
4. Conclusion
Conclusion

Take-home message

Abstraction heuristics are equally useful in settings where they must be automatically derived as in settings where they can be hand-tailored.

Moreover, there are plenty of research opportunities in transferring results from one area to the other.
Come to ICAPS 2008!

Tutorial:
- Abstraction Heuristics for Planning: PDBs and Beyond (Patrik Haslum and myself)

Papers:
- The Compression Power of Symbolic Pattern Databases (Marcel Ball and Robert C. Holte)
- Optimal Additive Composition of Abstraction-based Admissible Heuristics (Michael Katz and Carmel Domshlak)
- Structural Patterns Heuristics via Fork Decomposition (Michael Katz and Carmel Domshlak)