Seminar: Search and Optimization

Domain-Independent Construction of Pattern Database Heuristics for Cost-Optimal-Planning

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

Silvan Sievers

Universität Basel

November 21, 2013

Repetition: Pattern Databases (PDBs)

- Abstractions and PDBs:
  - Subset $A \subseteq \mathcal{V}$ of the problem's variables called the pattern
  - Abstraction defined by A ignores all $v \in \mathcal{V} \setminus A$
  - $h^A(s)$ is the minimum cost of reaching a goal state from state s in the abstract state space
  - PDB for pattern A contains h^A for all abstract states
- PDB heuristics:
  - Successfully used for optimal planning
  - Several abstractions can be combined (maximum or sum)

Pattern Databases ctd.

- Issues and open questions:
  - Quality of PDB heuristics strongly depends on choice of patterns
  - Difficult to predict what pattern suits problem domain or even instance
  - In domain-independent planning: not enough time/memory to try lots of patterns
- Contribution of this work:
  - Completely automatic and general pattern selection procedure

Agenda

Pattern Collections

Pattern Collections

- Motivation
  - Limited usefulness of single PDBs due to exponential growth rate
  - Want to use collections of multiple patterns
  - Can always use maximum over PDBs and stay admissible
  - Want to sum over PDBs whenever possible
- How do we best combine several PDBs?

Canonical Heuristic Function

Definition (Additivity criterion)

Let A and B be two patterns. If there exists no operator that affects variables from both patterns, then $h(s) = h^A(s) + h^B(s)$ is an admissible and consistent heuristic.

Definition (Canonical Heuristic Function)

Let $C = \{P_1, \ldots, P_k\}$ be a collection of patterns. Let A be the collection of all maximal (w.r.t. set inclucsion) additive subsets of C. Then the canonical heuristic function is defined as:

$$h^C(s) = \max_{S \in A} \sum_{P \in S} h^P(s)$$

Canonical Heuristic Function (ctd.)

Example

Planning task with $V = \{v_1, v_2, v_3\}$ and pattern collection $C = \{P_1, \dots, P_4\}$ with $P_1 = \{v_1, v_2\}$, $P_2 = \{v_1\}$, $P_3 = \{v_2\}$, $P_4 = \{v_3\}$. Operators affect single variables or v_1 and v_3 at the same time.

- Maximal additive subsets?
- Canonical heuristic function?

$\rightarrow$ Whiteboard

From planning course Universität Freiburg, WS2008

Pattern Selection

Pattern Selection as Search

- Local search:
  - Search space: pattern collections
  - Starting point: one pattern for each goal variable
  - Neighborhood: from $C = \{P_1, \ldots, P_k\}$, select $P_i \in C$, $v \notin P_i$ and add $P_{k+1} = P_i \cup \{v\}$ to C, resulting in C'.
  - End: memory limit is reached or no improvement possible

Evaluating the Neighborhood

- How to rank the relative quality of candidate pattern collections?
  - Estimate search effort of the candidates
  - Choose neighbor with the highest improvement in search effort
- What is "search effort"? Theoretical answer:
  - Number of node expansions of a tree search (IDA^*)
  - Depends on parameters of the search that can only be estimated

Evaluating the Neighborhood (ctd.)

- Observations:
  - No need for exact values, we are only interested in the best candidate collection
  - Good heuristics for IDA^* should be good for A^*
  - Use sampling to approximate the search effort
- Under several assumptions and simplifications, evalution reduces to:
  - Sample m states $s_1, \ldots, s_m$ through random walks in the search space
  - Improvement of C' over C: number of sample states s_i for which $h^{C'}(s_i) > h^{C}(s_i)$

Comparing $h^{C}(s)$ and $h^{C'}(s)$

- Evaluating the comparison:
  - C' contains C and P_{k+1}:

$$h^{C'}(s) > h^{C}(s) \text{ iff } h^{P_{k+1}}(s) + \sum_{P_i \in S - \{P_{k+1}\}} h^{P_i}(s) > h^{C}(s)$$

    for some additive subset $S \subseteq C'$ that includes P_{k+1}
- What do we need:
  - $h^{C}(s)$ is a simple look-up
  - $h^{P_{k+1}}$: want to avoid computing the PDB
  - Instead: Compute $h^{P_{k+1}}$ by searching with PDB for P_i serving as heuristic

Experiments

Comparison against mean value evaluation[1]

- Sokoban:
  - Search effort evaluation: solves 80 problems with 418730 nodes expanded
  - Mean value evalution: solves 66 problems with 657380 nodes expanded
- Logistics 2000:
  - Same coverage
  - 23992 vs 176850 nodes expanded

[1] Edelkamp, 2006

Conclusion

Conclusion

- Summary:
  - New approach of automatically constructing good pattern collections
  - Better resulting heuristic compared to previous work
- Additional work on PDBs:
  - Change the additivity criterion: cost partitioning[2]
  - Middle ground: post-hoc optimization[3]

[2] Katz and Domshlak, 2010
[3] Pommerening et al., 2013