2025-11-03

1.6-Bit Pattern Databases

Sebastian Lukas

University of Basel

Pattern Databases

- **Pattern Databases**

1.6-Bit Pattern Databases

Implementation

Experiment

Pattern Databases (PDBs)

- A lookup table for optimal solution costs
- Abstraction needed to limit PDB size
- Used as heuristic
- Multiple PDBs can be combined

1.6-Bit Pattern Databases

Pattern Databases

- **1.6-Bit Pattern Databases**

Implementation

Experiment

1.6-Bit Pattern Databases

- Idea: only store the heuristic value modulo-three
  - 5 compressed values in a byte
- Only possible if search space is undirected and unit cost
- Retrieval requires heuristic value of predecessor state

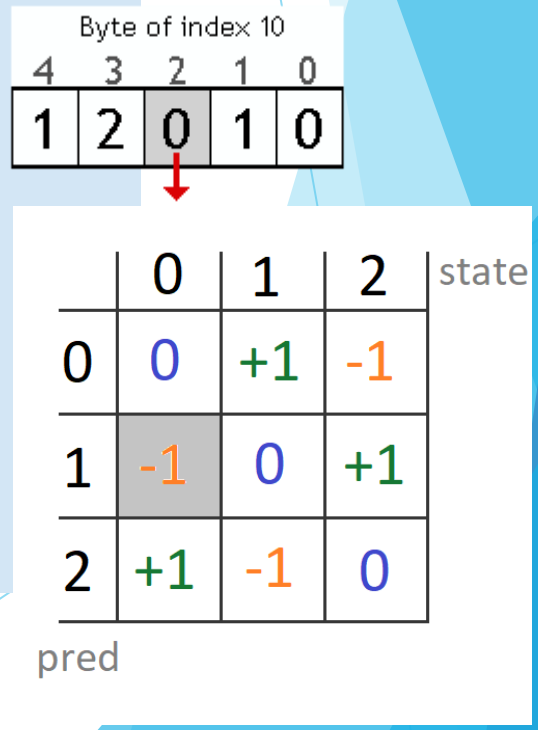
1.6-Bit Pattern Databases - Retrieval

- Find index of byte and subindex within byte
- Case distinction between predecessor state and new state
  - Difference between neighboring states

pred \ state	0	1	2
0	0	+1	-1
1	-1	0	+1
2	+1	-1	0

1.6-Bit PDB Example

1. State s, Predecessor state s_{pred} with concrete value of 7
2. Uncompressed index i = 52
3. Find the relevant byte at index $i_{byte} = i/5 = 10$
   - Sub-index $i_{sub} = i\%5 = 2$
4. Determine the difference between s and s_{pred}
5. $h(s) = 7 - 1 = 6$



Implementation Details

Pattern Databases

1.6-Bit Pattern Databases

- **Implementation**

Experiment

Implementation Details

- Implementation in Fast Downward
- Heuristic value of initial state stored
- Values of already visited states are cached
  - Entries for abstract states
  - Alternative: store path

Implementation Details - Heuristics

- Single PDB heuristic
  - Generates entire PDB, then compresses it
- iPDB
  - Use hill climbing to generate a pattern collection
  - Each PDB has ist own cache
  - Generates entire PDB collection, then compresses it

Experiment

Pattern Databases

1.6-Bit Pattern Databases

Implementation

- **Experiment**

Experiment

- IPC benchmarks
  - blocks, driverlog, elevators, gripper, logistics, termes & transport
  - Reversibility of operators checked with Python script
- Memory limit of 3947 MiB
- 3 questions

Experiment – Question 1

- How well does the 1.6-Bit compression work with the single PDB heuristic?
- Run the search with both the compressed and uncompressed PDB over the same tasks

Results – Single PDB

- Average compression rate of 79.1%
  - Compression rate varies
- Total memory usage not improved
- PDB only a small part of total memory usage
  - Avg. 7.1% for uncompressed
  - Max. 19.7% for uncompressed

PDB memory usage	$h^{PDB}_{compressed}$	h^{PDB}
blocks (21)	**0.006**	0.100
driverlog (11)	**0.005**	0.073
elevators-opt08-strips (20)	**0.005**	0.068
elevators-opt11-strips (16)	**0.005**	0.073
gripper (8)	**0.010**	0.056
logistics00 (16)	**0.004**	0.061
logistics98 (3)	**0.005**	0.106
termes-opt18-strips (12)	**0.003**	0.023
transport-opt08-strips (12)	**0.006**	0.075
transport-opt11-strips (8)	**0.007**	0.080
transport-opt14-strips (8)	**0.006**	0.062
Arithmetic mean (135)	**0.006**	0.071

Experiment – Question 2

- How well does the 1.6-Bit compression work with the iPDB heuristic?
- Run the search with both the compressed and uncompressed PDB collections over the same tasks

Results – iPDB

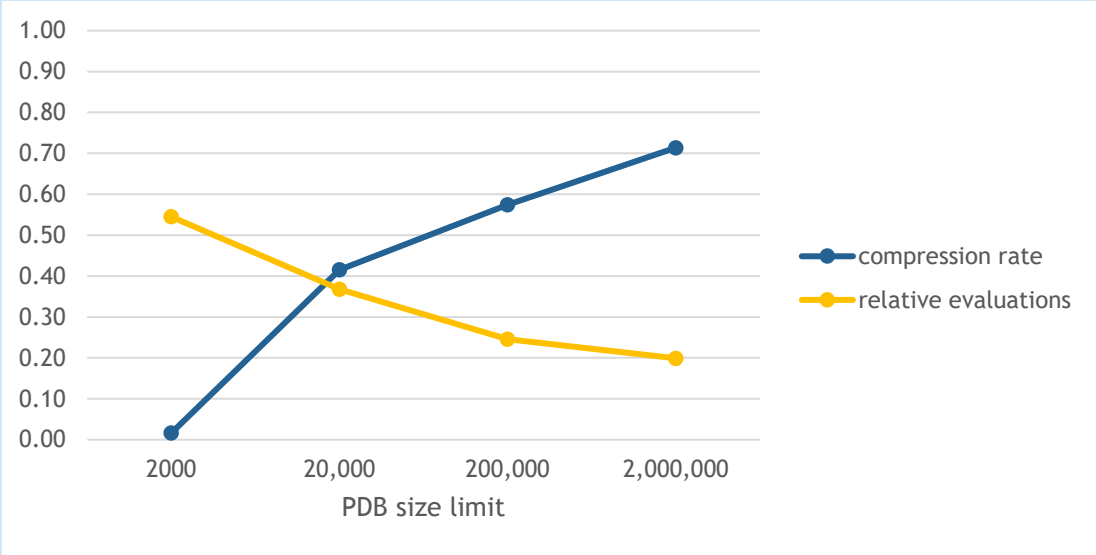
- Average compression rate of 71.4%
- Total memory usage was not improved

Experiment – Question 3

- Does the size of the PDB collection matter for the compression?
- Run the search with both compressed and uncompressed PDB collections at different size limits
  - Limits between 2000 and 2 million PDB states

Results – PDB Size

- Larger PDB collections can be compressed more efficiently
  - Less cached values relative to collection size



Key Points

- A 1.6-Bit PDB can store 5 modulo-three compressed heuristic values in a single byte
- Visited states are cached
- Compression works but total memory usage not improved
- Larger PDB collections are compressed more efficiently