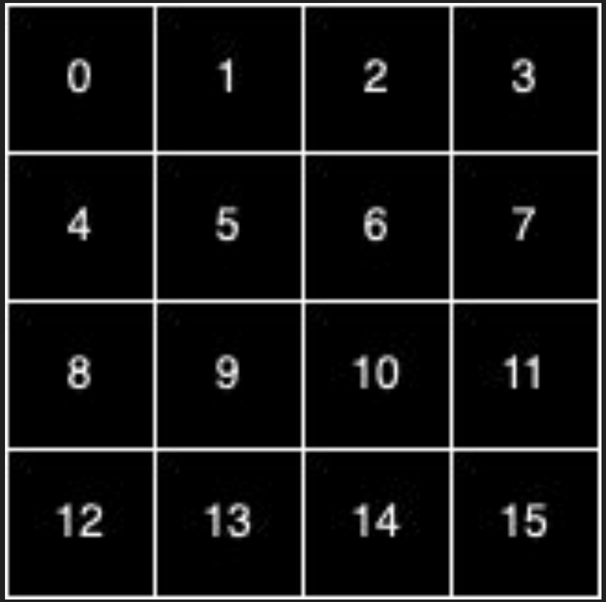
Solving the Sliding Tile Puzzle with Post-Hoc Optimization

Bachelor Thesis Presentation by Benedikt Heuser

The Sliding Tile Puzzle



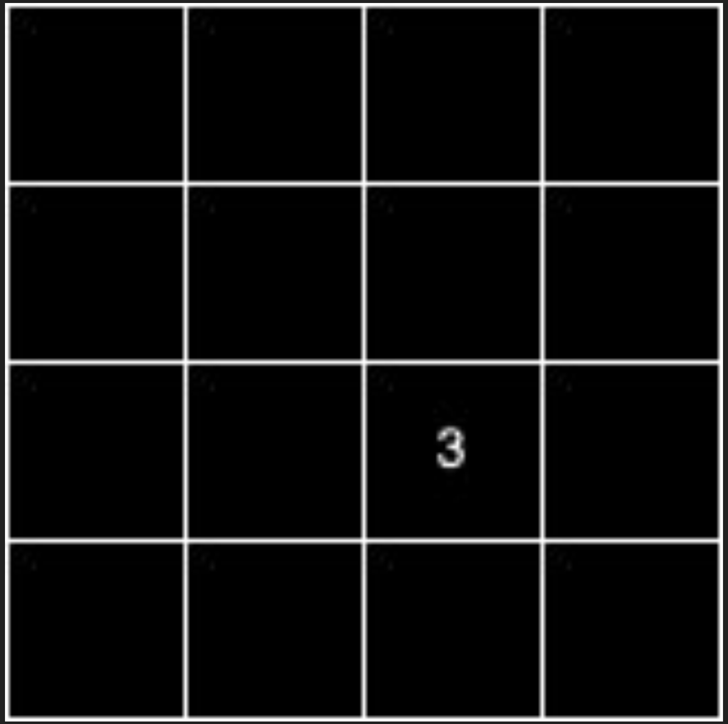
Iterative-Deepening A*

$$f(\text{state}) = g(\text{state}) + h(\text{state})$$

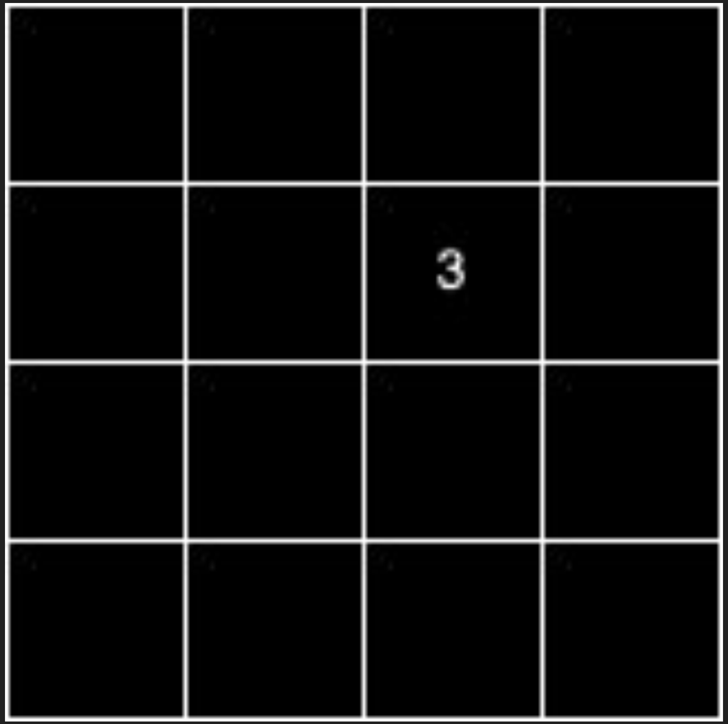
Manhattan Distance

5	2	7	12
9	6	0	14
11	10	3	1
13	15	4	8

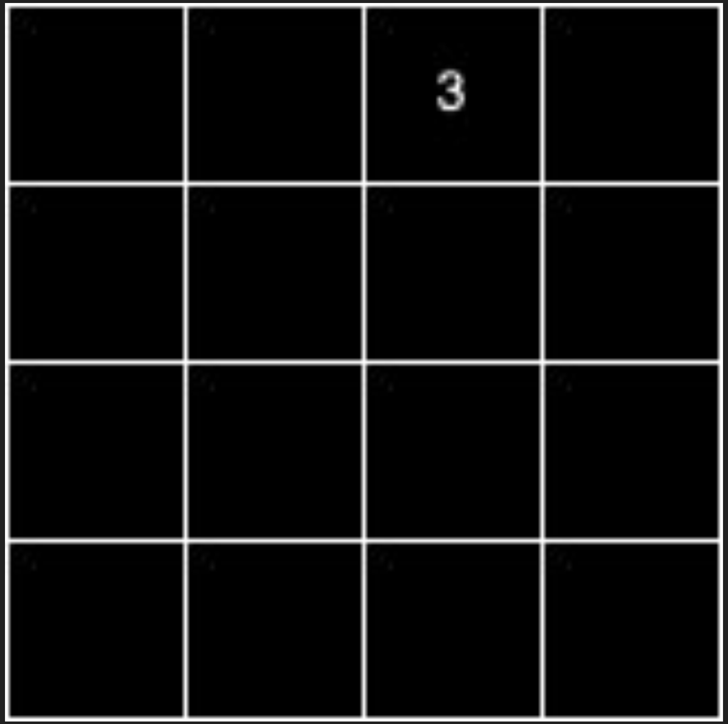
Manhattan Distance



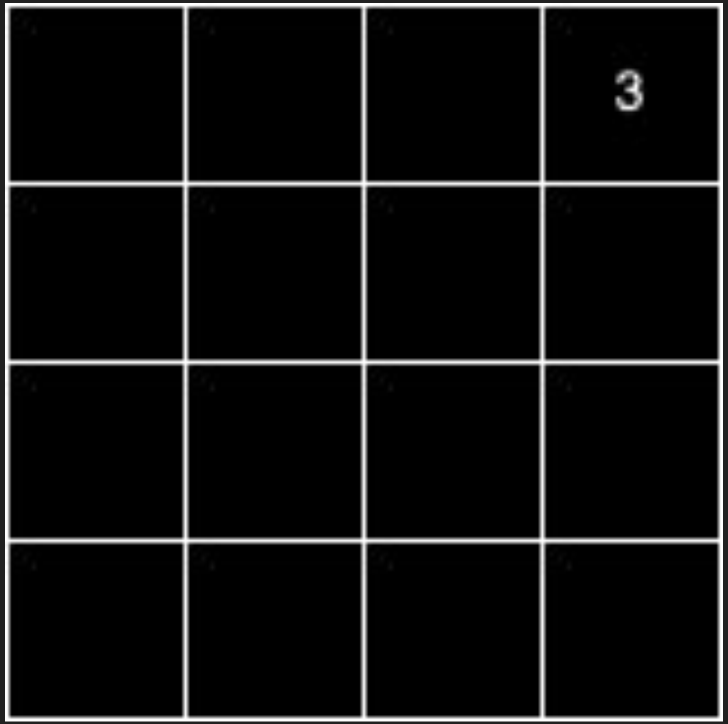
Manhattan Distance



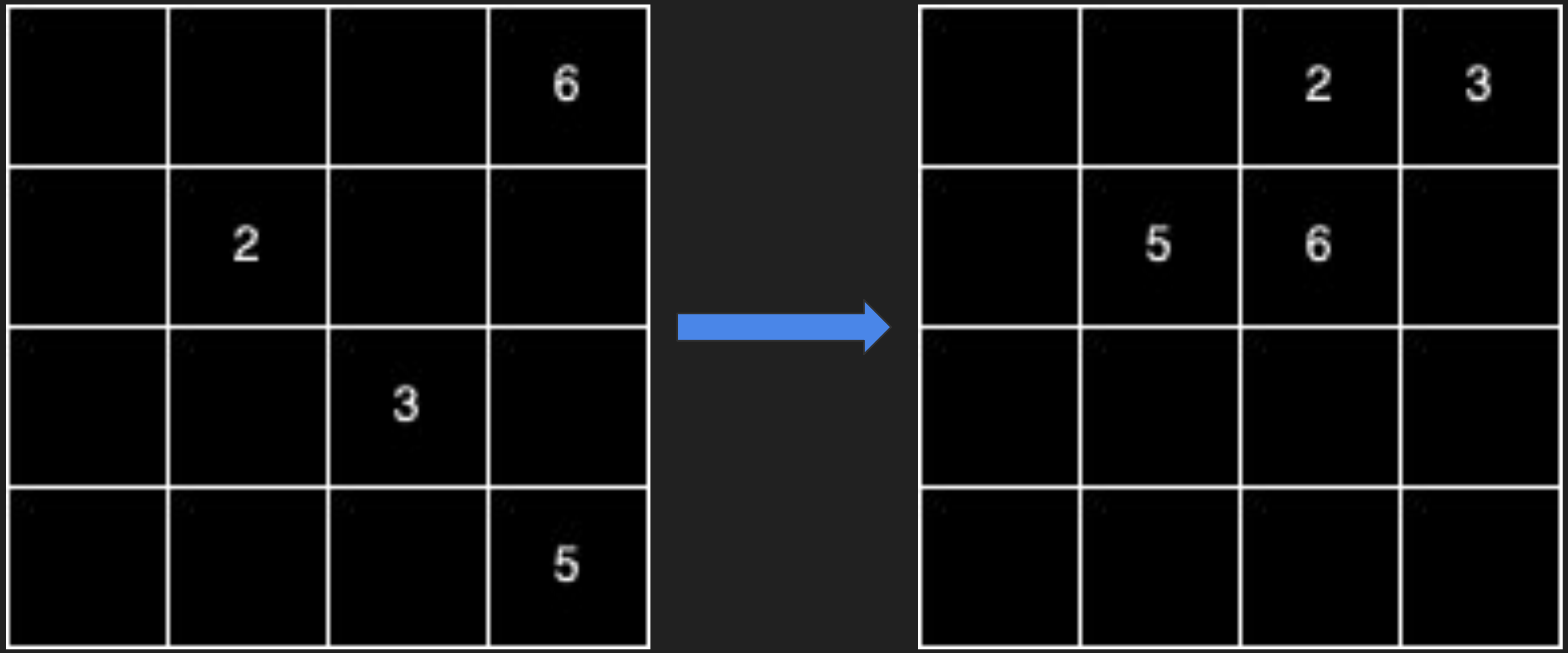
Manhattan Distance



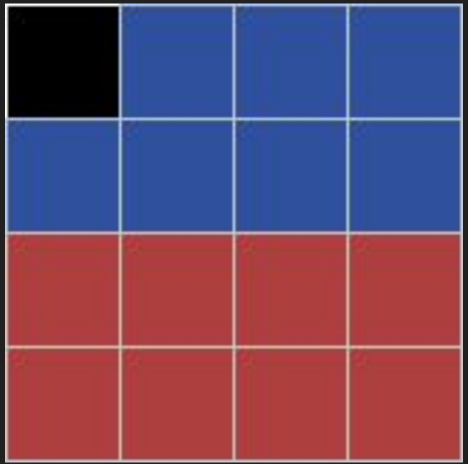
Manhattan Distance

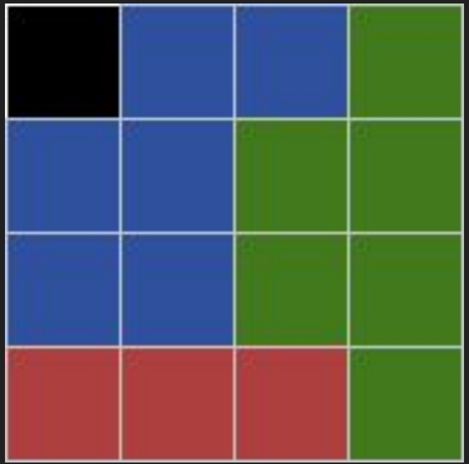


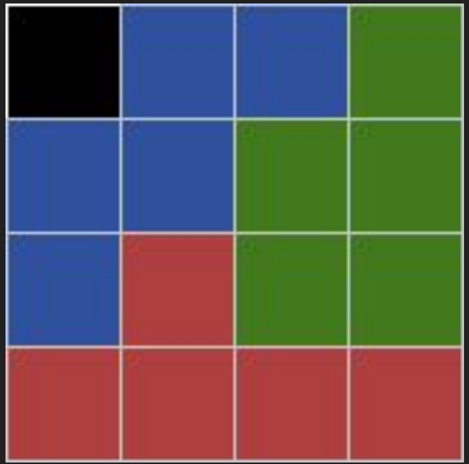
Pattern Databases



Plain Additive Pattern Databases







Post-Hoc Optimization

$$\begin{array}{llcll}
\text{minimize:} & \displaystyle\sum_{t=1}^{15} X_t & & & \\
\text{subject to:} & \displaystyle\sum_{t \in P_i} X_t & \geq & h^{P_i}(s) & \text{for every pattern } P_i \in \{P_1, \ldots, P_n\} \\
& X_t & \geq & 0 & \text{for every tile } t \in \{1, \ldots, 15\}
\end{array}$$

Pattern Sizes

Size	Memory	Number
1	0.22 KB	15
2	3.13 KB	102
3	40.67 KB	455
4	488.65 KB	1’365
5	5.54 MB	3’003
6	55.37 MB	5’005
7	480.32 MB	6’435

14

13

12

11

10

9

Pattern Sizes: Results

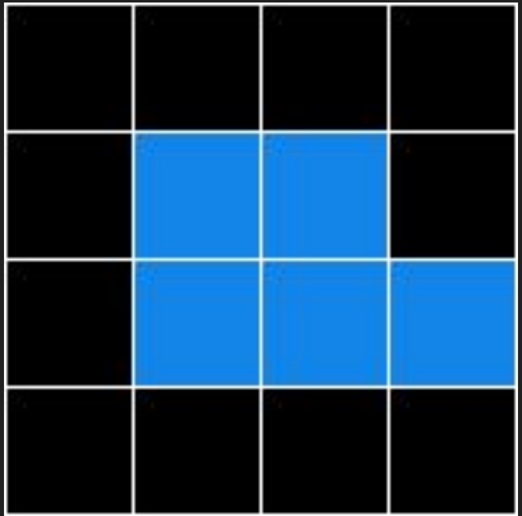
Experiment:

- 200 collections using uniformly sampled:
  - 5 x size 6
  - 50 x size 5
  - 550 x size 4
- 100 benchmark instances

Results:

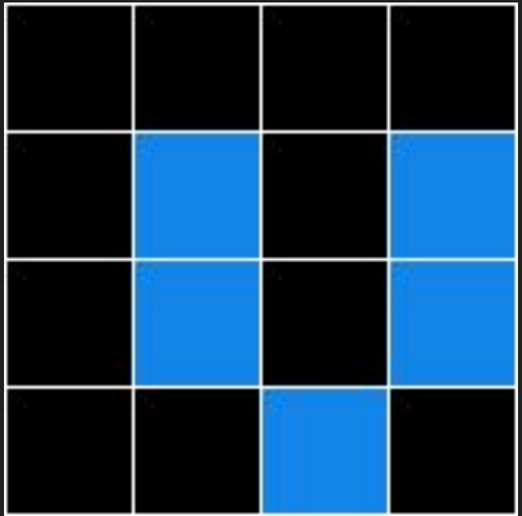
- Fewer total expansions for smaller PDBs
- Fewer generated nodes per second for smaller PDBs

Pattern Connectedness

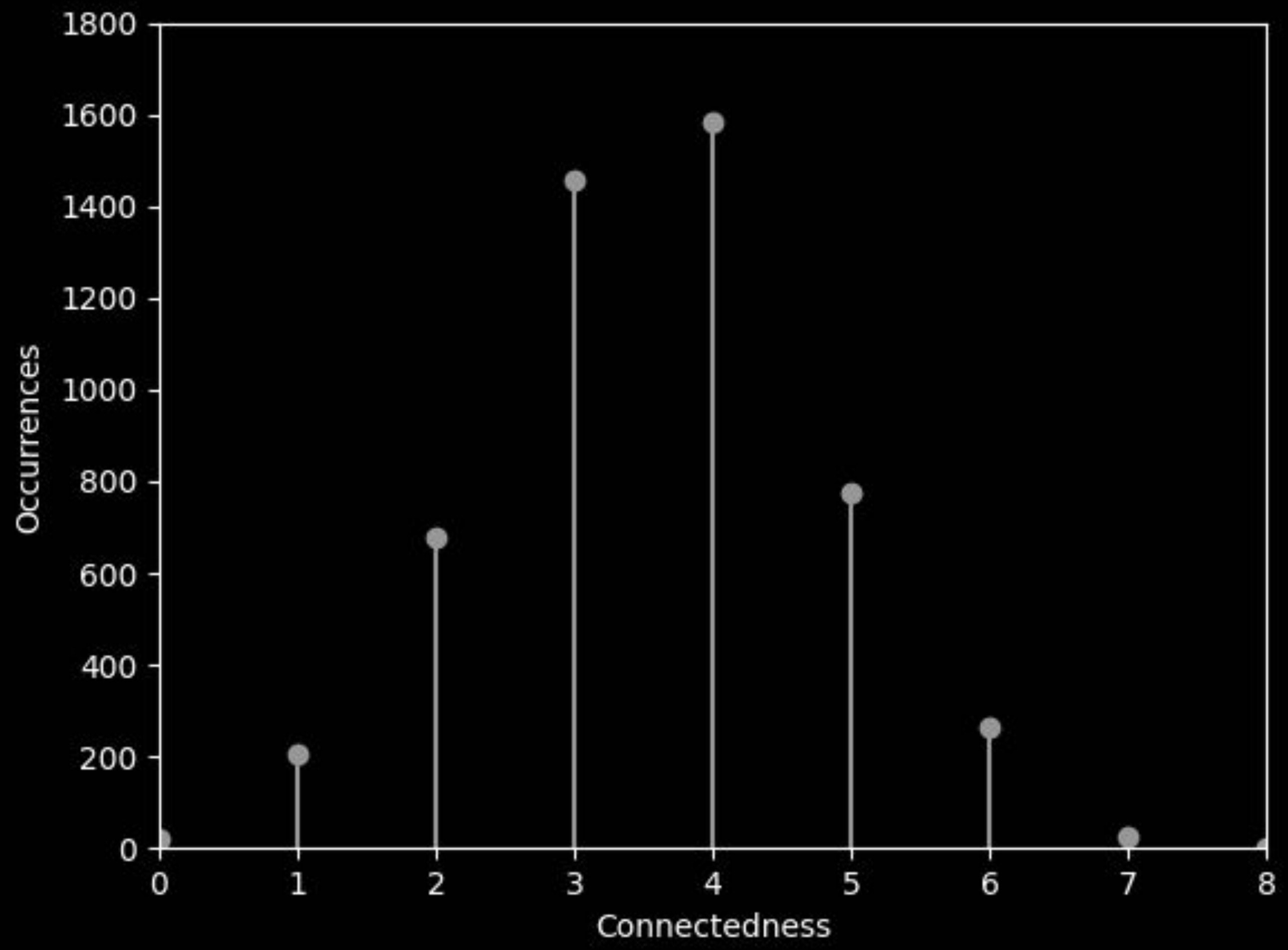
Connectedness () = 5

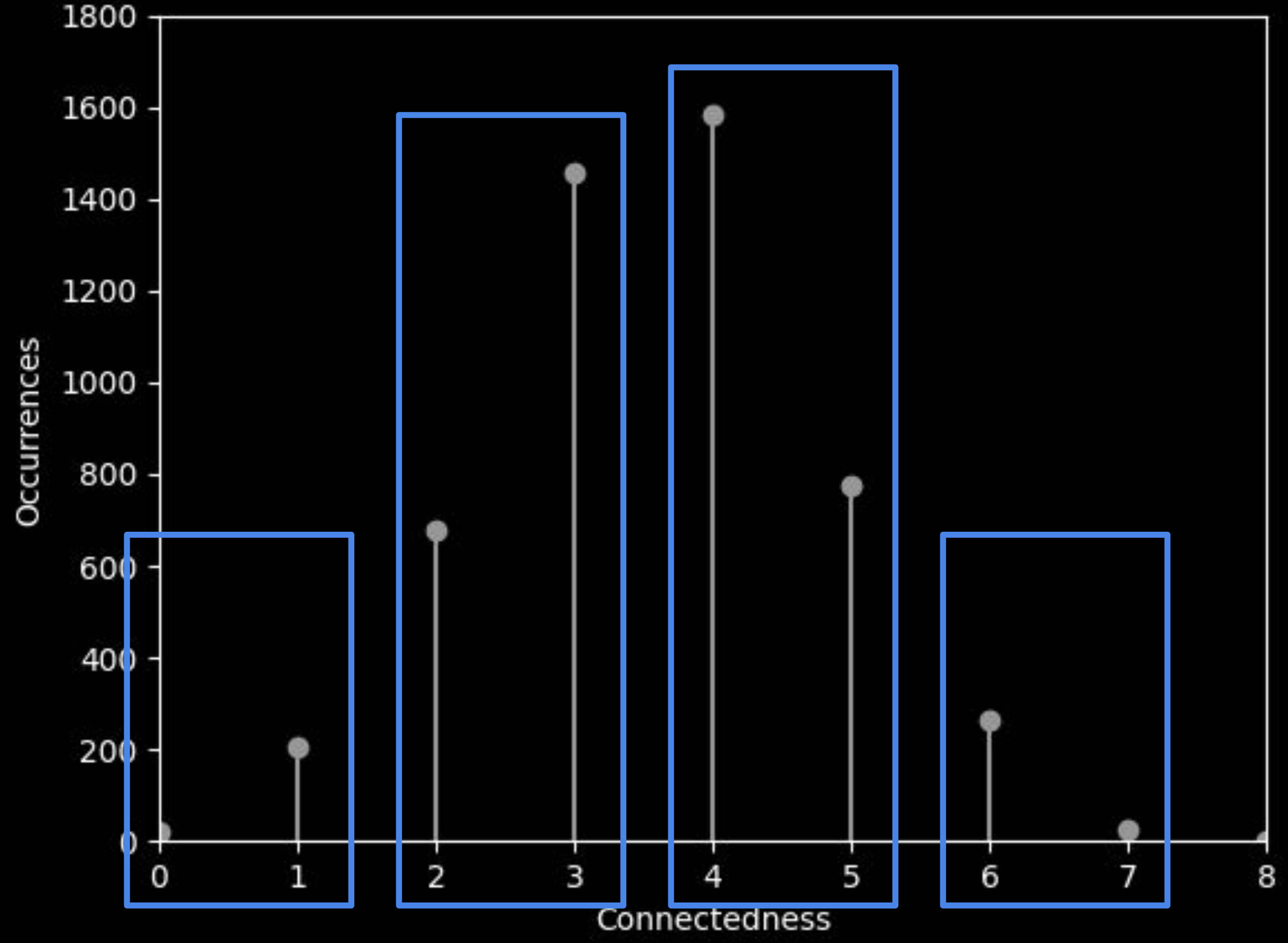
Pattern Connectedness

Connectedness (



) = 2


1800
1600
1400
1200
1000
800
600
400
200
0
Occurrences
0
1
2
3
4
5
6
7
8
Connectedness


1800
1600
1400
1200
1000
800
600
400
200
0
Occurrences
0
1
2
3
4
5
6
7
8
Connectedness

Pattern Connectedness: Results

Experiment:

- All patterns of size 6 divided into 4 levels of connectedness
- Randomly sample 200 collections per level using 20 PDBs
- 100 benchmark instances

Results:

- Fewer total expansions for connected nodes
- Fewer generated nodes per second for connected nodes

Time vs. Quality

Offline Post-Hoc Optimization

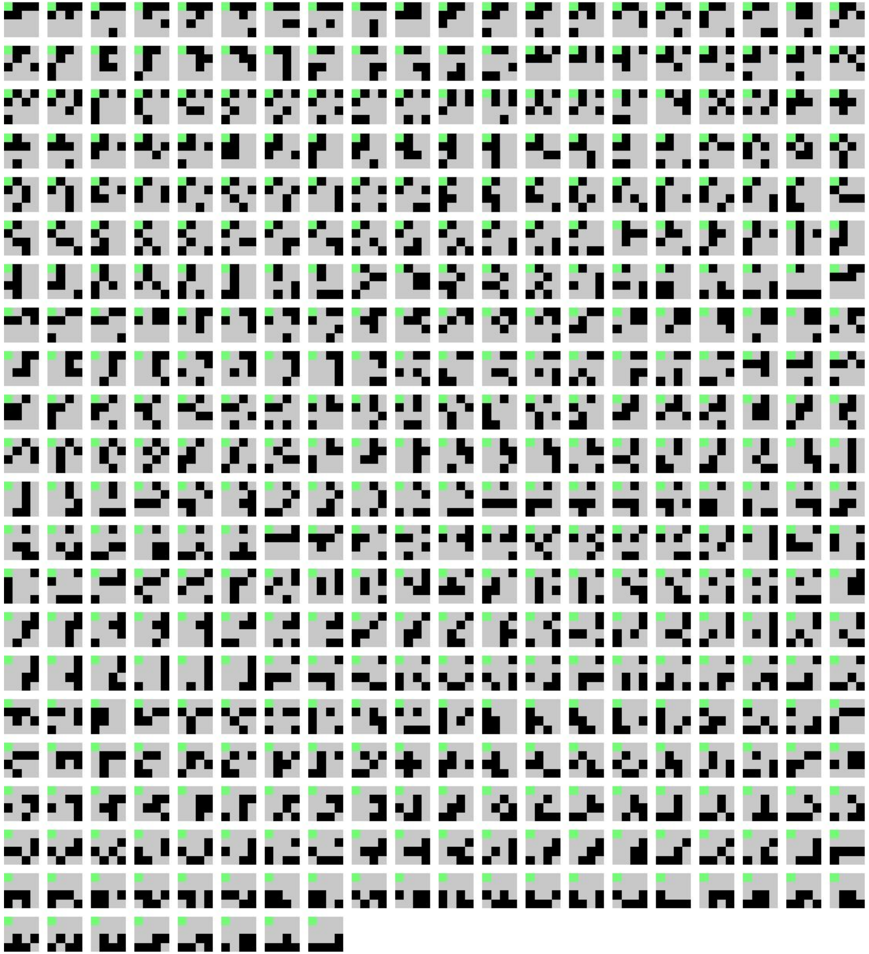
$$
\begin{array}{llcll}
\text{minimize:} & \displaystyle\sum_{t=1}^{15} X_t & & & \\
\text{subject to:} & \displaystyle\sum_{t \in P_i} X_t & \geq & h^{P_i}(s) & \text{for every pattern } P_i \in \{P_1, \ldots, P_n\} \\
& X_t & \geq & 0 & \text{for every tile } t \in \{1, \ldots, 15\}
\end{array}
$$

$$
\begin{array}{llcll}
\text{maximize:} & \displaystyle\sum_{i=1}^{n} Y_{P_i} h^{P_i}(s) & & & \\
\text{subject to:} & \displaystyle\sum_{P_i \ni t} Y_{P_i} & \leq & 1 & \text{for every tile } t \in \{1, \ldots, 15\} \\
& Y_{P_i} & \geq & 0 & \text{for every pattern } P_i \in \{P_1, \ldots, P_n\}
\end{array}
$$

Offline Post-Hoc Optimization

1. Calculate weights for N sample states
2. (Optimize weights)
3. During search:
    a. Calculate weighted sum of weights with PDB heuristics
    b. Use the maximum

- Input of 3’003 patterns of size 5
- 100 sample states
- 421 patterns
- Average of 3.6 non-zero weights
- 80% used exactly three PDBs with a weight of one



Offline Post-Hoc Optimization: Results

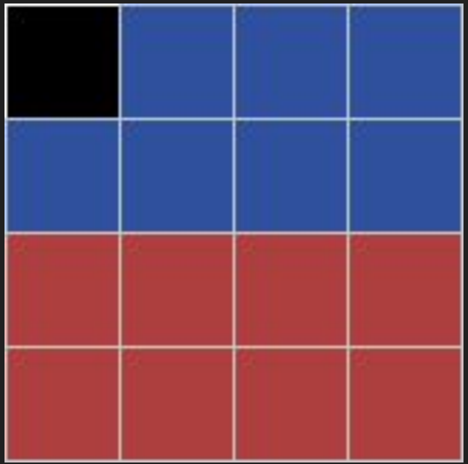
Experiment:

- OPHO vs. PHO
- (421) PDBs from before
- 100 benchmark instances

Results:

- OPHO generates more nodes per second (up to 1’000 sample states)
- Reduced heuristic quality leads to more expansions

Comparison with PA



Collection PA-8-7

Requires 5.66 GB of memory

Comparison with PA: Results

Algorithm	Expansions	Run Time (s)	Memory (GB)	Gen. per second
PA-8-7	3’744’197	.04	5.6	2’418’129
PHO-8-7	3’744’197	3.60	5.6	32’932
PHO-9x7	1’282’083’367	1’533.88	5.1	26’337
PHO-81x6	698’209’996	2’153.28	5.6	1’013’895
PHO-405x5	9’825’454	11’002.28	2.4	3’134

Summary

In our experiments:

- Many small PDBs provided a better heuristic, but generated fewer nodes per second.
- Connected patterns provided a better heuristic, but generated fewer nodes per second.
- OPHO can achieve lower run times, but requires more expansions.
- For many states, OPHO used additive collections.
- For memory limited to 5.6 GB, plain additive PDBs caused fewer expansions than PHO.