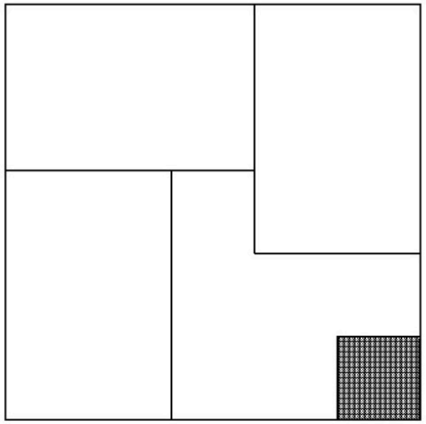
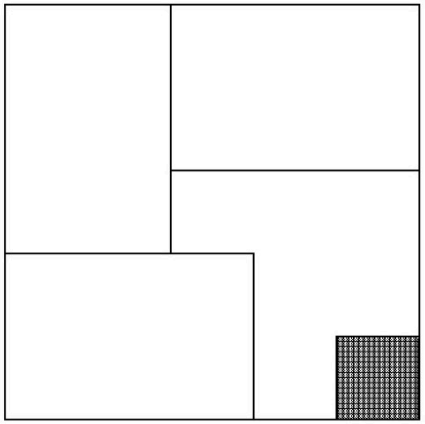
Solving N-Puzzle Using Pattern Database

R04922034 吳軒衡

5x5 disjoint patterns

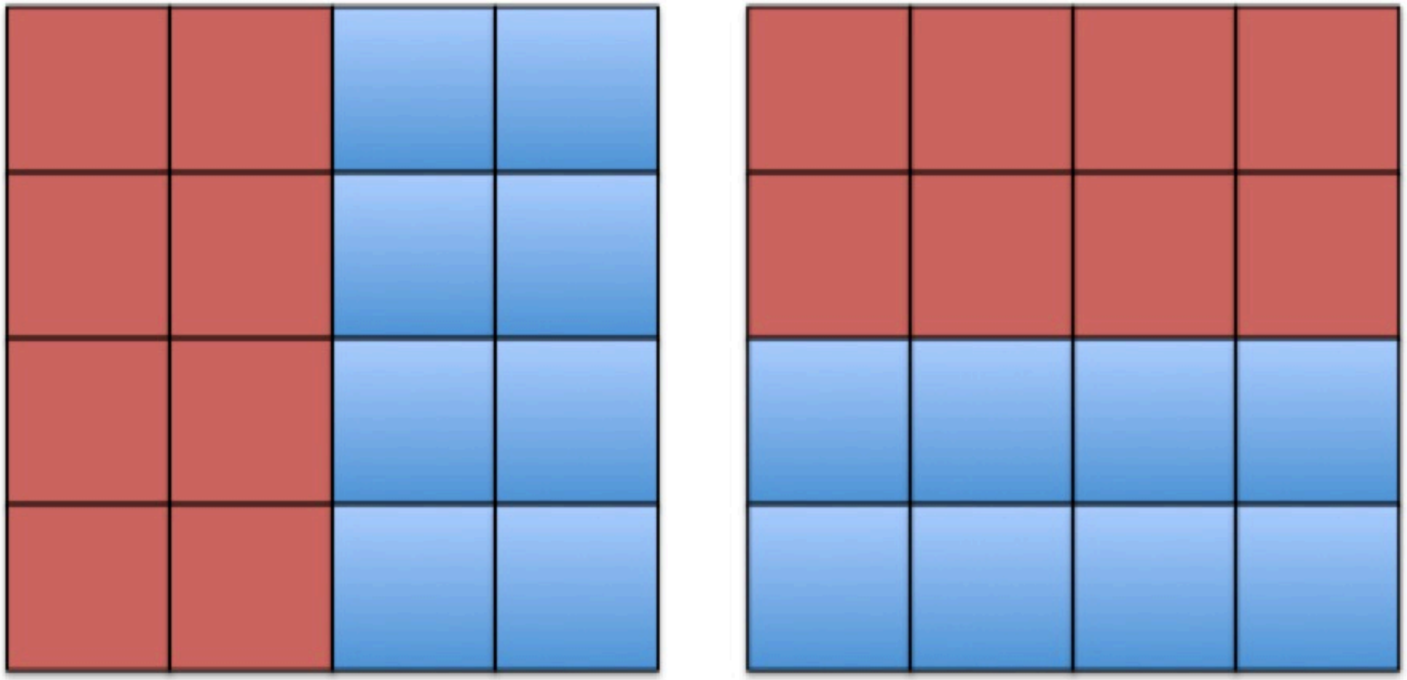




Number of pattterns

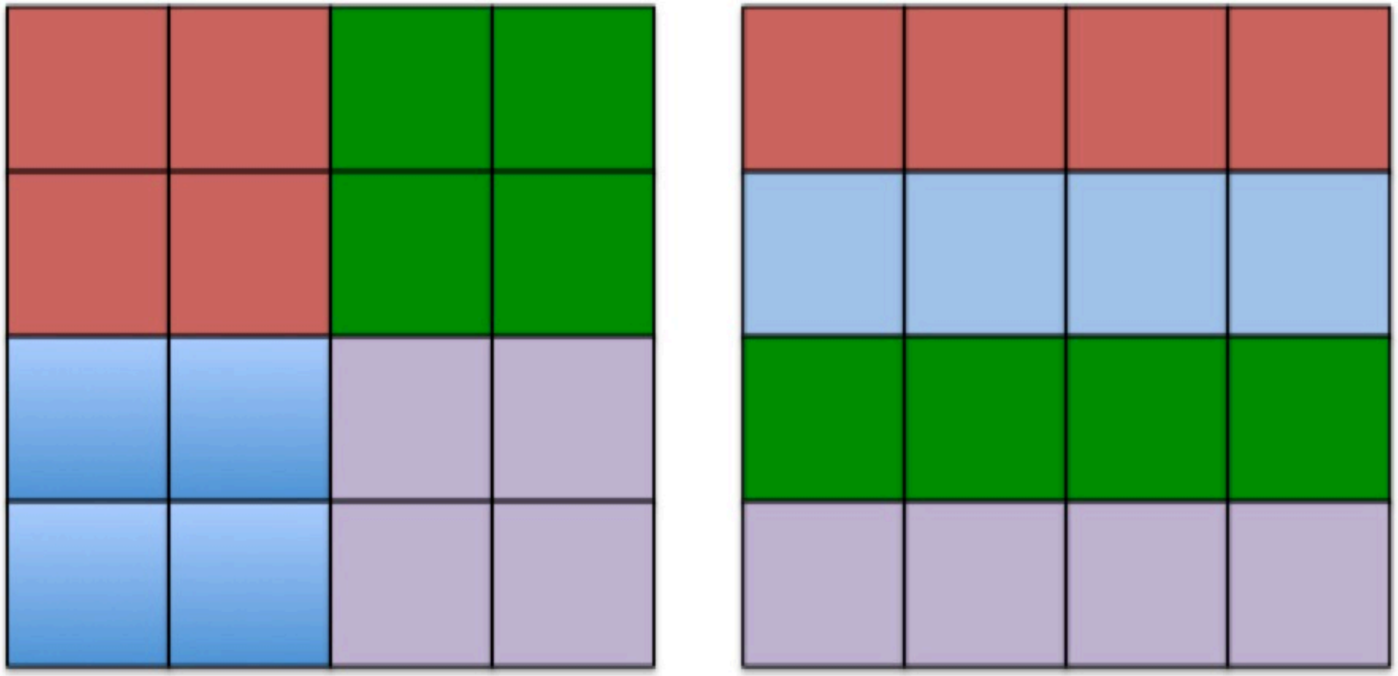
- C(25,6) * 6! = 127,512,000 for each of 8 patterns of size 6
- Requires 8 * C(25,6) * 6! * 4 byes =>
- 4 GB of storage

4x4 using 8/8 patterns



- C(16,8)*8! = 518918400 for each of the 4 patterns of size 8
- Requires 4*4*C(16,8)*8! Bytes=>
- 8GB of storage

4x4 using 4/4/4/4 patterns



- C(16,4)*4! = 43680 for each of the 8 patterns of size of 4.
- Requires 8*4*C(16,4)*4! = 1397760 Bytes =>
- 1MB Storage

Method of Pre-computing Database

- Generate all sequences C(n,k) * k! and perform A* with each of the starting position (pattern)
- A* uses Manhattan Distance as heuristic and outperforms BFS at least 10000 times .
- Only the moves of the tiles in the pattern are used, but the move of the empty tile should not be counted(in order not to overestimate the possible conflict).

Method of Problem Solving

- Preload Database into Memory (For performance issue).
- Perform Iterative Deepening A* (Iterative on the heuristic function)
- Update cost-limit to the smallest cut-off heuristic value
- For each node, the child are expanded through a fix sequence of direction

Experiment

- Conducted on size 4 pattern database
- Comparison between
- a. using single heuristic (2x2)
- b. using single heuristic (1x4)
- c using the max{a,b} as heuristic

Experiment Result

- C yields better result as expected

	max(h1,h2)	h1	h2
1	0.0006	0.00083	0.000669
2	2.311	2.07	12.416741
3	0.000137	0.000142	0.000119
4	0.000079	0.000098	0.00013
5	0.052651	0.100884	0.586012

Future Work

- Time permitting , conduct experiment on a 5x5 or larger puzzle.
- Conduct experiment using more examples.
- Automation on finding the best pattern to be use in this additive pattern database.
- Code Refactoring