

Disjoint Pattern Database Heuristics

by R.E. Korf and A. Felner

Tsan-sheng Hsu

徐讚昇

tshsu@iis.sinica.edu.tw

<http://www.iis.sinica.edu.tw/~tshsu>

Abstract

- Introducing a new form of heuristic called **pattern databases**.
 - Compute the cost of solving individual subgoals independently.
 - If the subgoals are disjoint, then we can use the sum of costs of subgoals as a new admissible cost function.
 - Make use of the fact that computer can memorize lots of patterns.
 - Solution to pre-stored patterns can be precomputed.
 - Speed up factor of over 2000 compared to previous results in 1985.

Definitions

- $n^2 - 1$ puzzle problem:
 - The numbers 1 through $n^2 - 1$ are arranged in a n by n square with one empty cell.
 - How to slide the tiles to a goal position.
- 15 puzzle:
 - Invented by Sam Loyd in the 1870s.
 - State space is divided into even and odd permutations.
 - ▷ *Number of inversions in a permutation $\pi_1\pi_2 \dots \pi_n$: the number of distinct pairs $\pi_i > \pi_j$.*
 - ▷ *Even permutation (parity): one with an even number of inversions.*
 - ▷ *Odd permutation (parity): one with an odd number of inversions.*
 - ▷ *Slide a tile never change the parity.*
 - The above argument may not be true for the 3*3 case.
 - Using DEC 2060 a 1 MIPS machine: solves random instances of the 15 puzzle problem within 30 CPU minutes in 1985.

Core of past algorithms

- Using Iterative-deepening A*.
- Using the Manhattan distance heuristic as the remaining cost estimation function.
 - Suppose a tile is currently at (i, j) and its goal is at (i', j') , then the Manhattan distance for this tile is $|i - i'| + |j - j'|$.
 - The Manhattan distance between a board and a goal board is the sum of the Manhattan distance of each tile.
- Manhattan distance is a lower bound on the number of slides needed to reach the goal board.
 - The tiles get in each other's way.
 - Sliding a tile to reach the destination may make the other tiles that are already in their destinations to move away.
 - A form of interaction is called **linear conflict**:
 - ▷ *To flip (2,1) needs more than 2 moves.*
 - ▷ *In addition, slides tiles other than 1 and 2 may be also needed to slip (2,1).*

Non-additive pattern databases

- Intuition: try not to measure the distance of one tile at a time.
 - Pattern database: try to measure the collective distance of a pattern, i.e., a group of tiles, at a time.

- The goal **fringe** pattern for the 15 puzzle:

*			3
			7
			11
12	13	14	15

- * means an empty cell.
- Blank means do care.
- There are $16!/8! = 518,918,400$ possible fringe arrangements.
- For each fringe arrangement, precompute the number of moves needed to make it into the goal fringe pattern.
 - ▷ *Using the original Manhattan distance heuristic to solve this smaller problem.*
 - ▷ *Can also solve this problem by BFS.*
 - ▷ *This is called the **fringe number**.*

Comments on pattern size

- Larger pattern is better in terms of a larger fringe number.
- Larger pattern means consuming lots of memory to memorize these patterns.
- Larger pattern also means consuming lots of time in constructing these patterns.
- Depends on your resource, pick the right pattern size.

Additional properties of fringes

- New heuristic function for IDA*: using the fringe number as the new lower bound estimation.
 - Reduce a 15-puzzle problem into a 8-puzzle one.
 - Solution =
 - ▷ *First reach the fringe pattern.*
 - ▷ *Then solve the 8-puzzle problem without using the fringe tiles.*
 - ▷ *Combining these two solutions to a form solution for the 15-puzzle problem.*
 - May not be optimal.

More than one patterns

- Can have many different patterns:

*		2	
		6	
8	9	10	11
		14	

1	7	2	3
4	*		
5			
6			

- Cannot use the divide and conquer approach anymore for some patterns.
- If you have many different pattern databases P_1, P_2, P_3, \dots
 - The heuristics are not disjoint.
 - ▷ *Solving tiles in one pattern may help solving tiles in another pattern.*
 - The heuristic function is $\max\{h(P_1), h(P_2), \dots\}$.

Key observations I

- Divide the board into several disjoint regions.
- For each region, solve the problem optimally and then count only the moves **that are made by tiles in this region**.
 - The new “fringe” number is the least number of slides made on tiles in this region.
 - Find one with the least number of fringe number.

- **Examples:**

- | | | | |
|---|---|---|---|
| * | A | A | A |
| A | A | A | A |
| B | B | B | B |
| B | B | B | B |

- | | | | |
|---|---|---|---|
| * | A | B | B |
| A | A | B | B |
| A | A | B | B |
| A | A | B | B |

- Can also divide the board into more than 2 disjoint patterns.

Key observations II

- **For the Manhattan distance heuristic:**
 - Each pattern is a tile.
 - They are disjoint, thus can be added together to form a heuristic.
- **Comments:**
 - Some kinds of divide and conquer.
 - Spaces required by pattern must be within the main memory.
 - Each pattern must be able to be solved optimally by “primitive” methods.
 - It is better to put near-by tiles together to better deal with the conflicting problem.

Performance

- Running on a 440 MHZ Sun Ultra 10 workstation.
 - SPECint = 1.0 (1 MIPS) in 1985.
 - SPECint = 17.9 in 2002.
- Solves 15 puzzle problem in time that is more than 2,000 times faster than the previous Manhattan distance heuristic.
- Solves 24-puzzle problems
 - An average of two days per problem instance.
 - Generates 2,110,000 nodes per second.
 - The average solution length was 100.78 moves.
 - The maximum solution length was 114 moves.
 - Prediction: using the Manhattan distance heuristic, it would take an average of about 50,000 years to solve a problem instance.
 - ▷ *The average Manhattan distance is 76.078 moves.*
 - ▷ *The average value for the disjoint database heuristic is 81.607 moves, which gives a tighter bound.*

Other heuristics

- The main drawback of disjoint heuristics is that they do not capture interactions between tiles in different groups of the partition.
- 2-tile pattern database:
 - For each pair of tiles, and for each pair of possible locations, compute the optimal solution for this pair of tiles to move to their destinations.
 - ▷ This is called *pairwise distance*.
 - ▷ For an $n^2 - 1$ puzzle, we have $O(n^4)$ different combinations.
 - ▷ For $n = 4$, $n^4 = 256$.
 - ▷ For $n = 5$, $n^4 = 625$.
 - For a given board, partition the board into a collection of 2-tiles so that the sum of cost is maximized.
 - ▷ This can be done using a maximum weighted perfect matching.
 - ▷ Build a complete graph with the tiles being the vertices.
 - ▷ The edge cost is the pairwise distance between these two tiles.
 - ▷ Try to find a perfect matching with the sum of edge costs being the largest possible.
 - ▷ Algorithm runs in $O(n(m + n \log n))$ is known where n is the number of vertices and m is the number of edges.

Comments

- The Manhattan distance is a partition into 1-tile patterns.
- For 2-tile patterns:
 - Faster approximation algorithms for finding maximum perfect matchings on complete graphs are known.
 - The cost for exhaustive enumeration is



$$\binom{16}{2} \binom{14}{2} \cdots \binom{4}{2} \binom{2}{2}$$

▷ = $16!/2^8 = 81,729,648,000$

- Can also build 3-tile databases, but the corresponding 3-D matching problem is NP-C.
- Requires much less memory than the fringe method.
- Some kinds of bootstrapping: solving smaller problems using primitive methods, and then use these results to solve larger problems.

What else can be done?

- Looks like some kinds of bi-directional search.
 - Searching from one direction means building precomputed results, e.g., patterns.
 - Searching from the other direction meets the precomputed results.
- Better way of partitioning the patterns.
- Is it possible to generalize this result to other problem domains?
- How to decide the amount of time used in searching and the amount of time used in retrieving precomputed knowledge?

References and further readings

- R. E. Korf. Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence*, 27:97–109, 1985.
- J. Culberson and J. Schaeffer. Pattern databases. *Computational Intelligence*, 14(3):318–334, 1998.
- * R. E. Korf and A. Felner. Disjoint pattern database heuristics. *Artificial Intelligence*, 134:9–22, 2002.