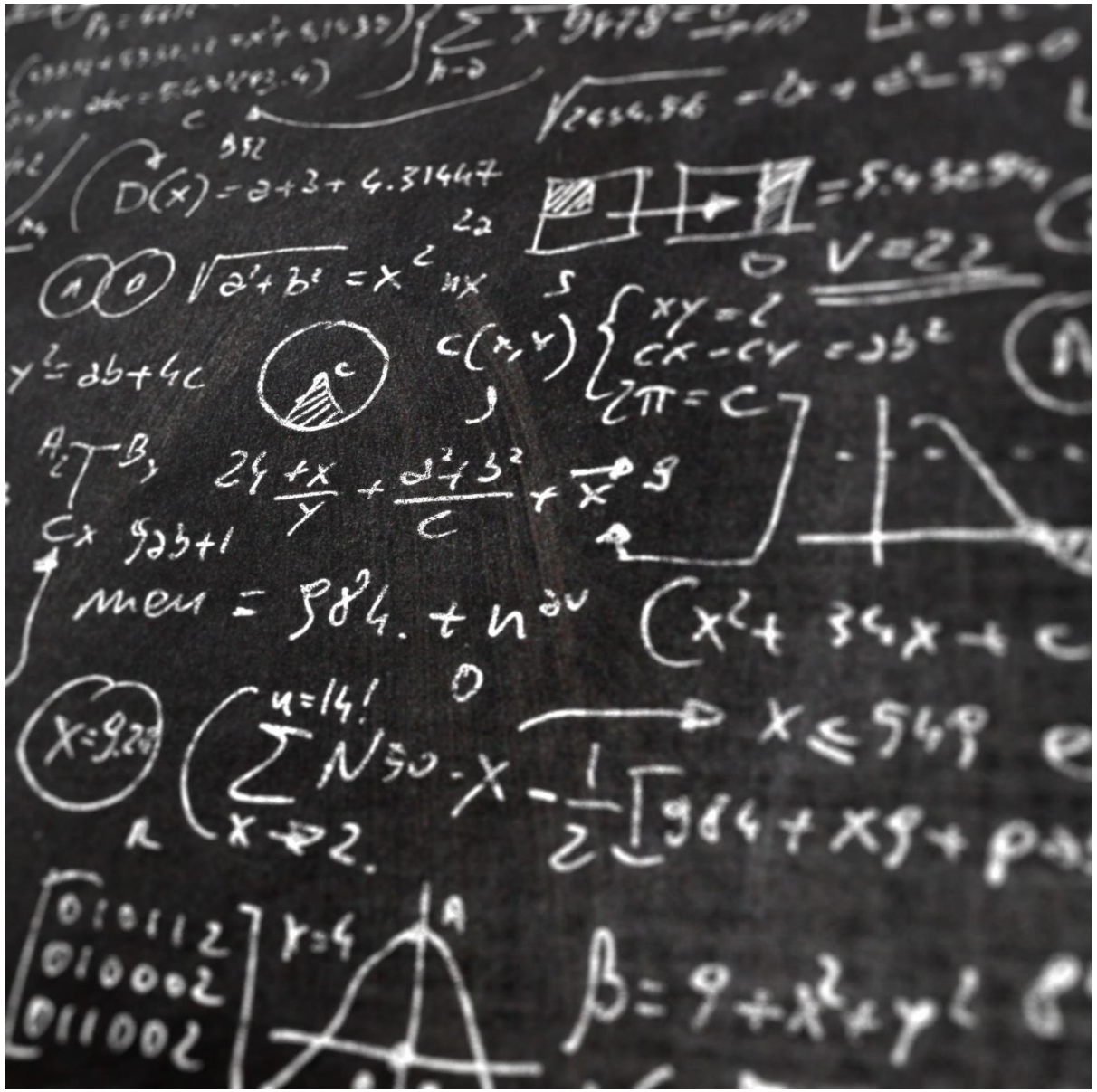
HEURISTIC FUNCTION



Heuristic Function

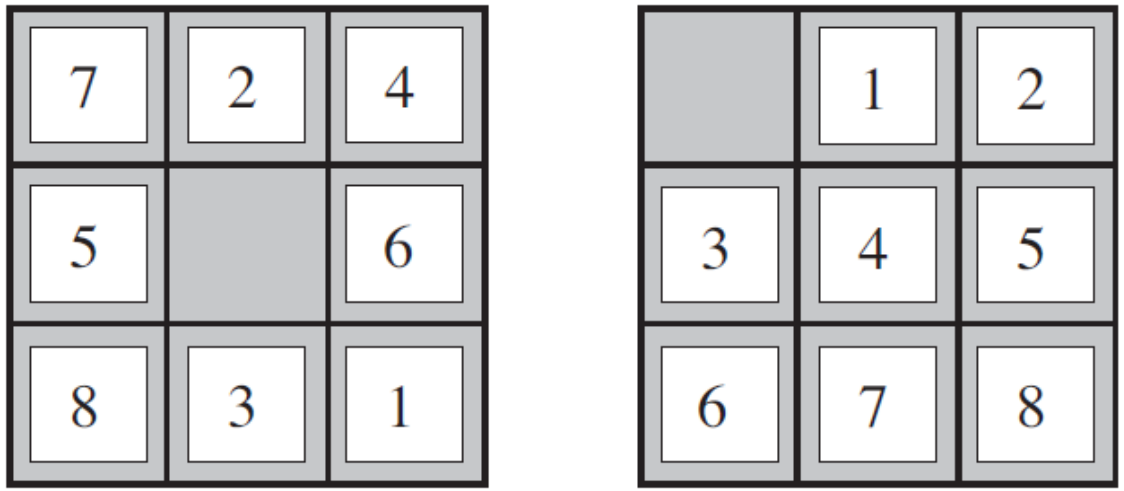


Figure 3.28 A typical instance of the 8-puzzle. The solution is 26 steps long.

Heuristic Function

- Average solution cost is about 22 steps.
- Branching factor is about 3.
  - Empty tile is in the middle then four moves are possible.
  - Empty tile is on the edge then three moves are possible.
  - Empty tile is in the corner then two moves are possible.
- Exhaustive tree search of depth 22 will have about 3^{22} (b^d) nodes.
- Graph search will have about 9!/2 nodes.

Good heuristic function?

- A function that never overestimates the number of steps to the goal. This function can be used if we want to find shortest solution by applying A* search.
- h_1 = number of misplaced tiles.
- h_2 = the sum of the distances of the tiles from their goal positions.

- $h_1 = 8$
- $h_2 = 18$
- True solution cost = 26

Effect of heuristic accuracy on performance

- Effective branching factor b^*
- If the total number of nodes generated by A* for a particular problem is N
- Solution depth is d
- b* is the branching factor that a uniform tree of depth d would have to have in order to contain N+1 nodes.

$$N + 1 = 1 + b^* + (b^*)^2 + \ldots + (b^*)^d$$

- Consider d = 5 & N = 52 then b^* = ?
- b^* = 1.92
- Well defined heuristic would have b^* close to 1.

Effect of heuristic accuracy on performance

	Search Cost (nodes generated)			Effective Branching Factor		
d	IDS	A*(h_1)	A*(h_2)	IDS	A*(h_1)	A*(h_2)
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30
8	6384	39	25	2.80	1.33	1.24
10	47127	93	39	2.79	1.38	1.22
12	3644035	227	73	2.78	1.42	1.24
14	–	539	113	–	1.44	1.23
16	–	1301	211	–	1.45	1.25
18	–	3056	363	–	1.46	1.26
20	–	7276	676	–	1.47	1.27
22	–	18094	1219	–	1.48	1.28
24	–	39135	1641	–	1.48	1.26

Figure 3.29 Comparison of the search costs and effective branching factors for the ITERATIVE-DEEPENING-SEARCH and A* algorithms with h_1, h_2. Data are averaged over 100 instances of the 8-puzzle for each of various solution lengths d.

How to generate heuristic function?

1. Admissible heuristics from relaxed problems
2. Admissible heuristics from subproblems
3. Learning heuristics from experience

Admissible heuristics from relaxed problems

- A tile can move from square A to square B if
  - A is horizontally or vertically adjacent to B and B is blank.

- Relaxed problems
  - A tile can move from square A to square B if A is adjacent to B.
  - A tile can move from square A to square B if B is blank.
  - A tile can move from square A to square B.

  - $h(n) = \max\{h_1(n), \ldots, h_m(n)\}$

Admissible heuristics from subproblems: Pattern database

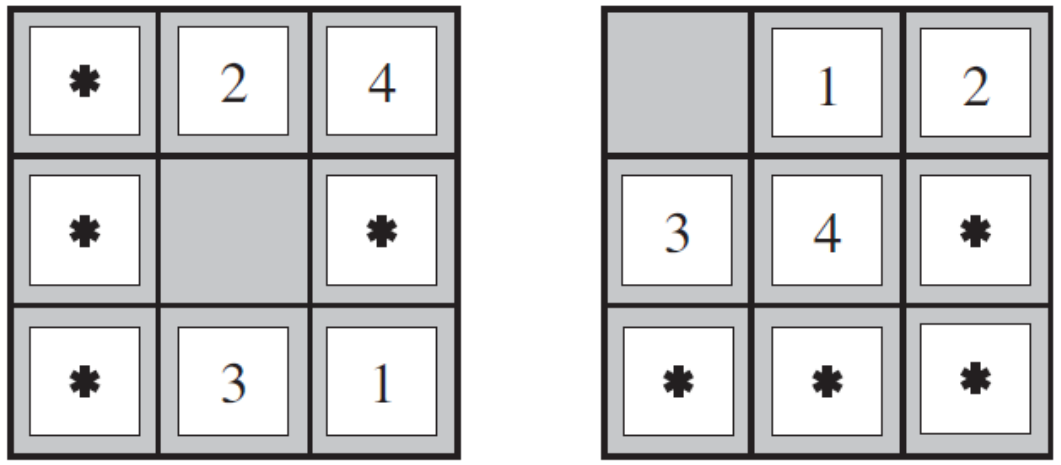


Figure 3.30 A subproblem of the 8-puzzle instance given in Figure 3.28. The task is to get tiles 1, 2, 3, and 4 into their correct positions, without worrying about what happens to the other tiles.

Learning heuristics from experience

- Solve lots of problems and learn from experience.
- Identify features
  - One feature learning: Number of misplaced tiles!
  - Two features learning:
    - Number of misplaced tiles!
    - Number of pairs of adjacent tiles that are nor adjacent in the goal state

$$h(n) = c_1x_1(n) + c_2x_2(n)$$