N-Puzzle Problem

Practical analysis of the problem

Luigi Seminara

Dipartimento di
Matematica e Informatica
Università degli studi di Catania, Italia

Academic Year 2022/2023

Problem

Description of the problem

The N-Puzzle, also known as the sliding tile puzzle, is a classic problem in artificial intelligence and cognitive psychology. The problem consists of a square board with $N^2 - 1$ numbered tiles and one blank space, where N is the dimension of the board.

Example with $N = 3$:

1	2	3
4	5	6
7	8	

Goal

Description of the goal

The goal of the problem is to rearrange the numbered tiles in ascending order using a series of sliding moves on the blank space.

Example with $N = 3$:

1	2	3
4		6
7	5	8

1	2	3
4	5	6
7	8	

Table of contents

DFS

1. Define the initial state of the puzzle and the goal state of the puzzle.
2. Create a stack to store the puzzle states that need to be explored. Initially, the stack should contain the initial puzzle state.
3. While the stack is not empty, pop the top state from the stack and check if it is the goal state.

DFS

1. If the current state is not the goal state, generate all possible next states by sliding the blank tile in different directions.
2. Repeat until the goal state is found or the stack is empty.

DFS

Problem: The time complexity of depth-first search depends on the depth of the deepest solution, as it expands each node at least once. Spatial complexity depends on the depth of the deepest solution, since all expanded nodes must be stored in memory. Depth-first search isn't optimal, as it doesn't take stock cost into account.

1	2	3
4	5	6
7		8

1	2	3
4	5	6
7	8	

BFS

- The breadth-first search is complete, as it explores all states of the search space until finding the solution.
- The time complexity of the breadth-first search is $O(b^m)$, where b is the branching factor and m is the maximum depth.
- The space complexity of the breadth-first search is $O(b^m)$, as it maintains all states at the maximum depth in the frontier queue.
- The breadth-first search does not guarantee optimality, as it explores states based on their depth, regardless of the path cost to reach them.

Informed Research

Definition

Informed research is a problem solving strategy that uses additional information about the environment and the problem. This information, called heuristics, is used to guide the search towards the regions of the search space most likely to contain a solution.

A* search

A* search is an informed search strategy that uses an evaluation function to determine the distance of the current node from the goal and the estimated cost to reach the goal through that node. The evaluation function of A* is defined as:

$$f(n) = g(n) + h(n)$$

where:

- $g(n)$ is the actual cost to reach node n (i.e. the cost of the solution so far);
- $h(n)$ is the estimated cost to reach the goal from node n (i.e. the heuristic).

A* search expands the node with the lowest $f(n)$ value first. This means that A* aims to find the optimal solution, i.e. the solution with the lowest cost. A* uses an admissible heuristic $h(n)$ that never overestimates the actual distance.

IDA* search

IDA* provides the benefits of A* without the need to maintain all the reached states in memory, at the cost of visiting some states multiple times. It is an important and commonly used algorithm for problems that do not fit in memory. In standard iterative deepening, the cut-off is the depth, which is increased by one in each iteration. In IDA*, the cut-off is the cost $f(g + h)$, where g is the cost to reach the current state and h is the heuristic to estimate the cost to reach the final state; in each iteration, the cut-off value is the smallest f cost of any node that exceeded the cut-off in the previous iteration.

Heuristics

Manhattan distance

This heuristic calculates the total Manhattan distance between each tile's current position and its goal position. This is done by summing the absolute differences of the current and goal row and column indices of each tile.

Heuristics

Euclidean distance

This heuristic calculates the total Euclidean distance between each tile's current position and its goal position. This is done by summing the square root of the sum of the squared differences of the current and goal row and column indices of each tile.

Heuristics

Misplaced tiles

This heuristic simply counts the number of tiles that are not in their goal position on the puzzle.

Heuristics

Linear conflict

This heuristic counts the number of pairs of tiles that are in the same row or column as their goal position but are in the wrong order.

Example:

X	X	X
X	6	5
X		X

Heuristics

Linear conflict + Manhattan distance

This heuristic is a combination of the Linear conflict and Manhattan distance heuristics. It calculates the sum of both the Manhattan distance and the Linear conflict for the puzzle.

8-Puzzle

$N = 3$

8	6	4
5	3	
2	7	1

A* search

Heuristic	**Cost path**	**Nodes expanded**	**Search depth**	**Max depth**	**Time**
MD	27	5138	27	27	2s
ED	27	13500	27	27	11s
LC + MD	27	**3716**	27	27	2s

15-Puzzle

$N = 4$

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

A* search

Heuristic	**Cost path**	**Nodes expanded**	**Search depth**	**Max depth**	**Time**
MD	31	26268	31	31	**79s**
ED	29	29440	29	29	130s
LC + MD	33	35275	33	34	185s

24-Puzzle

$N = 5$

1	3	4	14	5
6	7		8	9
11	18	2	23	13
16	12	19	24	17
21	22	20	15	10

A* search

Heuristic	**Cost path**	**Nodes expanded**	**Search depth**	**Max depth**	**Time**
MD	39	40950	39	39	232s
ED	-	-	-	-	+1440s
LC + MD	41	11755	41	42	23s

N-Puzzle Problem

Practical analysis of the problem

Luigi Seminara

Dipartimento di
Matematica e Informatica
Università degli studi di Catania, Italia

Academic Year 2022/2023