Experimental report for the 2021 COM1005 Assignment: The 8-puzzle Problem*

Muhammad Kamaludin

April 24, 2022

1 Descriptions of my breadth-first and A* implementations

1.1 Breadth-first implementation

Breadth-first search is essentially a type of uninformed search algorithm which equally searches every nodes at each depth before proceeding to the subsequent depth until the goal is reached. This algorithm works by traversing the higher depth's nodes first and add those nodes to the open list.

Open list stores the unexplored nodes and involves in the node selection. In this part, the open list will be implemented as a queue which acts on the first-in-first-out basis to achieve the intended behaviour.

This experiment uses three fixed puzzle patterns as shown in Figure 1, and additional three randomly generated puzzle patterns of 2022 seed; with three different difficulties which are 6, 9, and 12 as shown in Figure 2. The efficiencies of this algorithm towards each puzzle patterns are recorded and the average efficiency is calculated.

1.2 A* implementation

A* is a search algorithm which includes heuristics to help determines the next nodes. This implementation requires the following variables to execute:

- local cost
- total global cost

*Link to personal GitHub *private* https://github.com/aimanarifi/COM1005.git

- estimated remaining cost
- estimated total cost

In the 8-puzzle problem, the local cost is the number of tile movement at each node which is 1 since only 1 tile can move to the empty space at a time. The total global cost is the accumulated cost from the starting node to the current node. The estimated total cost is the sum of estimated remaining cost and total global cost.

In this implementation, the open list will not act like any elementary data structure and does not heavily affecting the algorithm. However, the algorithm will select a node with the least estimated total costs of all nodes in the open list.

In addition, the experiment will use two approaches of calculating the estimated remaining cost namely *Manhattan* and *Hamming*. From this, the experiment can show whether or not a determination of heuristic affects the efficiency of an algorithm.

- *Manhattan* - Sum of distance of individual number to their correct place
- *Hamming* - Number of tiles that are out of place

This approach also uses the same puzzle patterns as the Breadth-first implementation which are shown in Figure 1 and Figure 2. The efficiencies of this algorithm towards each puzzle patterns are recorded and the average efficiency is calculated.

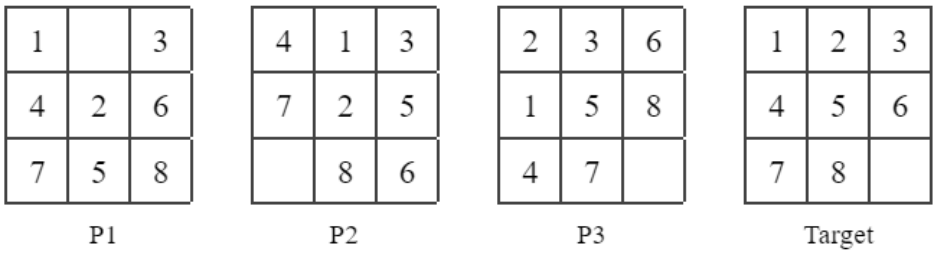

Figure 1: Fixed puzzle patterns as in assignment sheet

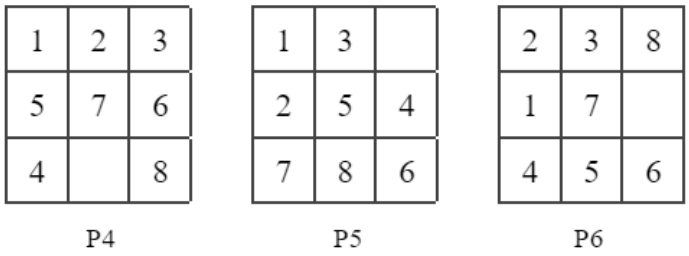

Figure 2: Randomly generated puzzle patterns of seed 2022, and difficulty 6, 9, and 12

2 Results of assessing efficiency for the two search algorithms

Efficiency of the three algorithms on each puzzle patterns are recorded and tabulated in Table 1.

Puzzle	Breadt-first	A* Manhattan	A* Hamming
P1	36.36%	100.00%	100.00%
P2	12.28%	87.50%	100.00%
P3	5.45%	75.00%	90.00%
P4	14.63%	75.00%	75.00%
P5	0.16%	4.87%	2.42%
P6	0.24%	10.81%	3.64%
Average	11.52%	58.86%	61.84%

Table 1: Efficiencies of algorithm on each puzzle patterns

3 Conclusions

Here I need to write what conclusions I can draw from my experimental work.