CS3243 Project 1

Chan Tjun Chuang Alex, Calvin Chen Xingzhu, Lim Jun Hup, and Ritwik Jha

NUS School of Computing, COM1, 13 Computing Drive, Singapore 117417.

Abstract. This paper details the implementation of uninformed and informed searches to solve a k-puzzle of arbitrary size. Iterative Deepening Search (IDS) and A* Search with three heuristics are implemented. Results of running the algorithms on 3-puzzle to 5-puzzle grids are shown and discussed.

Keywords: Informed search · Search heuristics · K-puzzle.

1 Problem Specification

The k-puzzle can be solved by performing a search through all possible states of the puzzle, until the goal state is reached. The possible states from the present state can be modelled as a tree, with the nodes representing the states, and the branches representing the transition from one state to next [1]. Such a tree can be searched using a search function.

2 Technical Analysis

Definitions: d - depth of goal node; b - branching factor, m - maximum depth of search tree, $h^*(s_0)$ - actual distance to goal node, $h(s_0)$ - heuristic distance to goal node, $C*$ - Cost of optimal solution path.

2.1 Correctness & complexity

Iterative Deepening Search

Completeness Complete if there exists a goal node at a finite depth d and branching factor is finite [1].

Optimality Optimal when the path cost is a non decreasing function of the depth of the node [1].

Space Complexity $O(bd)$

Time Complexity $O(b^d)$

A* Graph Search

Completeness Complete if there are finitely many nodes with cost $\leq C*$, which is true if all step costs $>$some finite ϵ and if b is finite [1].

Optimality Optimal provided heuristic is consistent–and hence admissible for graph search [1]. For tree search, A* is optimal provided heuristic is admissible[1]. (proof of admissibility and consistency shown in Section 2.2).

Space Complexity $O(b^m)$

Time Complexity $O(b^{h^*(s_0)-h(s_0)})$

2.2 Admissibility & consistency

Linear Conflict (Relaxed version)
The version used is a relaxed version of the Linear Conflict (LC) heuristic described in [2]. Two tiles t_j and t_k are in a linear conflict if t_j and t_k are the same line, the goal positions of t_j and t_k are both in that line, t_j is to the right of t_k, and goal position of t_j is to the left of the goal position of t_k.

When there exists 1 or more linear conflict within a row/col between the tiles, we consider this row/col to only have 1 linear conflict.

Our heuristic value is calculated as such: $h(n) = 2\times$ No. of linear conflict + Manhattan Distance(MD)

Lemma 1. *When moving a tile involved in a row/col conflict away to another row/col, MD increases by 1.*

At the row where there exists a row conflict, MD $\neq 0$ (since state is not equal to goal state whenever there exists a row conflict). When moving a tile involved in the row conflict away to another row, the tile is now one more position away from it's goal position, as it now has to move back to the goal row. As such, MD increases by 1. This applies for column conflict as well.

Theorem 1. *Proof of consistency for Linear Conflict*

Definition 1. *Take MD(h(n)) as the Manhattan Distance in h(n).*

Base Case
Take t as the goal node, with node n as a predecessor of t such that t is a successor of n.

$$
\begin{aligned}
h(n) &= \text{there exists no row/col conflict} + \text{Manhattan distance} \\
h(n) &= 0 + 1c(n,t) + h(t) \\
&= 1 + 0(h(t) = 0(\text{since t is the goal node}) \\
\therefore h(n) &= 1 <= c(n,t) + h(t) = 1
\end{aligned}
$$

Induction Step *For any n and any successor n' of n, there exists a path (n, n_1, .. n_m, n').*

- ***Case 1*** *(Transiting from j row/col conflict to j + 1 row/col conflicts, j >0)* n_i *has j row/col conflict and* n_{i+1} *has j + 1 has row/col conflict.*

$$\begin{aligned} h(n_i) &= 2j + MD(h(n_i)) \\ h(n_{i+1}) &= 2(j+1) + MD(h(n_{i+1})) - 1 \textbf{\textit{(Lemma 1)}} \\ \therefore h(n_i) &= 2j + MD(h(n_i)) \leq c(n_i, n_{i+1}) + h(n_{i+1}) \\ &= 1 + 2(j+1) + MD(h(n_i)) - 1 \\ &= 2j + MD(h(n_i)) + 2 \end{aligned}$$

- ***Case 2*** *(Transiting from j row/col conflicts to j - 1 row/col conflict, j >0)* n_i *has j row/col conflict and* n_{i+1} *has j - 1 has row/col conflict*

$$\begin{aligned} h(n_i) &= 2j + MD(h(n_i)) \\ h(n_{i+1}) &= 2(j+1) + MD(h(n_{i+1})) + 1 \textbf{\textit{(Lemma 1)}} \\ \therefore h(n_i) &= 2j + MD(h(n_i))) \leq c(n_i, n_{i+1}) + h(n_{i+1}) \\ &= 1 + 2(j-1) + MD(h(n_i)) + 1 \\ &= 2j + MD(h(n_i)). \end{aligned}$$

- ***Case 3*** *(Transiting from j row/col conflicts to j row/col conflicts, j >0)*

$$\begin{aligned} h(n_i) &= 2j + MD(h(n_i)) \\ h(n_{i+1}) &= 2j + MD(h(n_{i+1})) \end{aligned}$$

Since MD is a consistent heuristic,

$$\begin{aligned} MD(h(n_i)) &\leq c(n_i, n_{i+1}) + MD(h(n_{i+1})) \\ h(n_i) &\leq c(n_i, n_{i+1}) + h(n_{i+1}) = 1 + h(n_{i+1}) \end{aligned}$$

$\therefore$*for any* n_i *on this path,*

$$\begin{aligned} h(n_i) &\leq c(n_i, n_{i+1}) + h(n_{i+1}) \\ Soh(n) &\leq c(n, n_1) + h(n_1) \leq c(n, n_1) + c(n_1, n_2) + h(n_2) \\ &\leq \cdots \leq c(n, n_1) + c(n_1, n_2) + \cdots + c(n_m, n') + h(n') \\ &= c(n, n') + h(n') \end{aligned}$$

Hence, heuristic 3 is consistent.

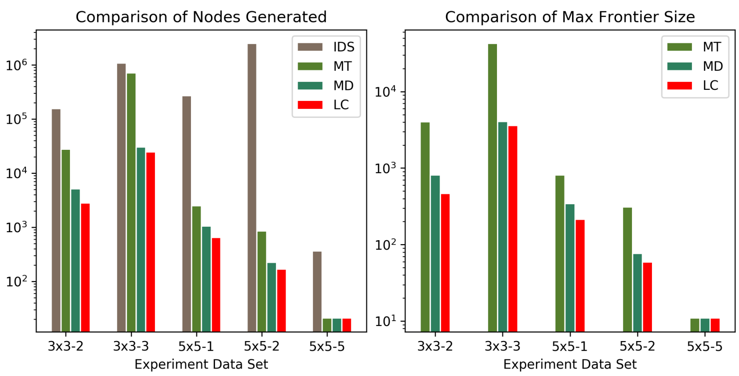
3 Results and Discussion

3.1 Experimental results

For all 4 methods, we noted down the number of steps required to get to the (optimum) solution, the number of nodes generated, max number of nodes in the frontier, and the runtime.

3.2 Iterative deepening search (IDS)

The team chose to use IDS as our algorithm for uninformed search since IDS combines space complexity $O(bd)$ of DFS with the completeness of BFS. Although it incurs additional overheads due to revisiting of earlier states, its impact is insignificant given a large number of states.



(a) Nodes Generated (b) Max Frontier Size[1]

Fig. 1: Comparison of nodes generated and max frontier size for IDS and A* with Misplaced Tiles(MT), Manhattan Distance (MD) and Linear Conflict (LC) heuristics.

3.3 A* graph search

The team chose to implement A* graph search as the informed search algorithm since it is optimal and complete. A consistent heuristic is required to ensure optimality. Memory requirement is also large as A* requires storage of all generated nodes.

3.4 Comparing IDS and A* Search

Generally, A* performs better than IDS given an admissible or consistent heuristic. Uninformed search algorithms have no knowledge of states unlike A* which uses heuristics to guide it to the goal. This can be seen from Figure 1 where all the three A* algorithms generated significantly lesser nodes as compared to IDS.

3.5 A* heuristics dominance

LC heuristic dominates the **Manhattan Distance** heuristic which in turn dominates the **Misplaced Tiles** heuristic. Given the time complexity of A* search is dependent on the heuristic, linear conflict has the best runtime performance. This is seen from Figure 1 where **Linear Conflict** heuristic generated the least nodes as compared to the other two heuristics.

References

1. Russell, Stuart J., Norvig Peter: Artificial Intelligence A Modern Approach. Prentice Hall **3**, (2010).
2. Hanson, Othar, Mayer, Andrew, Yung, Moti: Criticizing Solutions to Relaxed Models Yields Powerful Admissible Heuristics. Elsevier Science Publishing Co., Inc. 1992. Informational Sciences **63**, 207–227 (1992).

[1] IDS not shown because it was implemented recursively, so frontier was not generated

Appendix A

Table 1: Experimental results for each algorithm (and heuristic) implemented.

Input	Puzzle Size	Path Len., Nodes Gen., Max Frontier, Time			
		IDS	Misplaced Tile	Manhattan Distance	Linear Conflict
2	3×3	22, 156174, 0.88212	22, 27793, 4008, 0.28260	22, 5109, 812, 0.062760	22, 2801, 465, 0.052006
3	3×3	31, 1081374, 6.23765	31, 716725, 42510, 8.86442	31, 30313, 4051, 0.36118	31, 24465, 3608, 0.40544
2	4×4	Timeout	Timeout	50, 420749, 90989, 8.037749	50, 89445, 20222, 2.60211
3	4×4	Timeout	Timeout	34, 61221, 14873, 1.045083	34, 23517, 5488, 0.68001
4	4×4	Timeout	Timeout	43, 1657653, 402793, 34.88568	43, 588793, 138706, 21.52088
1	5×5	14, 271787, 1.52848	14, 2493, 812, 0.040834	14, 1057, 343, 0.021554	14, 645, 214, 0.038963
2	5×5	18, 2506523, 14.53066	18, 853, 311, 0.016602	18, 225, 77, 0.0057358	18, 169, 59, 0.010432
3	5×5	Timeout	33, 2348505, 708794, 54.67928	33, 40809, 12802, 0.91187	33, 6097, 1944, 0.28561
4	5×5	Timeout	30, 187629, 59736, 3.92041	30, 6021, 1932, 0.12593	30, 2281, 753, 0.11107
5	5×5	5, 367, 0.0027508	5, 21, 11, 0.00048494	5, 21, 11, 0.00051188	5, 21, 11, 0.0015499