An Evaluation of Uninformed and Informed Search Algorithms on the k-puzzle Problem[⋆]

Dalis Chan *(A0187451Y)*, Johanna *(A0187536R)*, Sean Low Chen Yi *(A0183743Y)*, and Law Ann Liat, Larry *(A0189883A)*

National University of Singapore
Repository Link here

Abstract. K-puzzle is often used as test problems for new search algorithms in artificial intelligence [4, p71]. This paper evaluates the use of iterative deepening search (IDS) and A^* search. Since A^* search uses heuristic functions to guide its search, this paper also evalutes the heuristic functions euclidean distance, manhattan distance, and linear conflict.

1 Problem Specification

1. **State:** For $k \in \{3, 4, 5\}$, a $k \times k$ matrix M with each entry $m_{i,j}$ being a unique integer from $\{0, 1, \cdots, 8\}$ where 0 represents the blank tile.
2. **Initial State:** Puzzle can start in any state s.
3. **Actions or *Actions(s)*:** Let $m_{k,l} \in M$ denote the blank tile and $m_{i,j} \in M$ denote the tile **adjacent** to the blank tile $m_{k,l}$. Actions are movements of the adjacent tile $m_{i,j}$ towards the blank tile $m_{k,l}$. For example, the action *Left* moves the adjacent tile $m_{k,l+1} \in M$ to the blank tile $m_{k,l}$.
4. **Transition Model or *Result(s,a)*:** $Result(s, a)$ swaps the pair of tiles specified in action a in the current state s and returns this new state *s'*.
5. **Goal State:**

$$M_{goal} = \begin{bmatrix} 1 & 2 & \cdots & k \\ k+1 & k+2 & \cdots & 2k \\ \vdots & \vdots & \vdots & \\ k^2-k+1 & k^2-k & \cdots & 0 \end{bmatrix}$$

6. **Path Cost:** Every step cost *c(s, a, s') = 1*, and the path cost is the summation of the step costs from the initial state to the goal state.

2 Technical Analysis of the Selected Algorithms and Heuristics

2.1 Rule to Check if k-puzzle is Solvable

Definition. [1]. A pair of tiles form an *inversion* if the values on tiles are in the reverse order of their appearance in the goal state.

[⋆] Supported by Prof Yair Zick and Prof Daren Ler.

Rules [1]. Let M denote a $k \times k$ matrix, $m_{i,j}$ denote a blank tile in M, and n_i denote the number of inversions in the intial state $M_{initial}$. Puzzle is solvable if . . .

1. k is odd and n_i is even
2. k is even,
   (a) n_i is odd and $m_{i,j}$ is on an even row from the bottom (ie j is odd).
   (b) n_i is even and $m_{i,j}$ is on an odd row from the bottom (ie j is even).

2.2 Uninformed Search

1. **Implementation:** Graph-based IDS. Step costs are equal, thus it is optimal [4, p88]. Furthermore, since the search space is large and the depth of the solution is not known, IDS is preferred [4, p90].
2. **Correctness:** Branching factor $b = 4$ is finite, thus IDS is complete [4, p88-90].
3. **Time Complexity:** $O(b^d)$ [4, p88-90].
4. **Space Complexity:** $O(bd)$ [4, p88-90].

2.3 Informed Search

1. **Implementation:** Graph-based A^* search. It improves on greedy best first search (ie $f(n) = h(n)$) as it avoids expanding paths that are already expensive (ie $f(n) = g(n) + h(n)$).
2. **Correctness:** Since the search space is finite, A^* search will be complete.
3. **Time Complexity:** $O(b^{h^*(s_0)-h(s_0)})$ [4, p93-99].
4. **Space Complexity:** $O(b^m)$ [4, p93-99].

2.4 h_1 : Manhattan Distance

Definition. Manhattan Distance heuristic is defined as the sum of the distance of the tiles from their goal positions [4, p103] . Note that this sum only includes horizontal and vertical distances as *Actions* do not allow diagonal movements.
Proof for Consistency. Proof for consistency can be found in appendix A.

2.5 h_2 : Euclidean Distance

Definition. Euclidean Distance heuristic is defined as the straight line distance between the tiles from their goal position [3].
Proof for Consistency. Euclidean distance is a form general triangle inequality, given that the euclidean distance from start state S to end state G (1 side of the triangle) cannot be longer than the sum of the 2 sides (the actual distance from S to middle state N and the euclidean distance from N to G) as the euclidean distance from S to G is already the shortest path. Since general triangle inequality fulfills the definition of consistency [4, p95], euclidean distance is consistent.

2.6 h_3 : Linear Conflict

Definition. Two tiles t_j and t_k are in a linear conflict if t_j and t_k are in 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 the goal position of t_j is to the left of the goal position of t_k [2, p13].
Derivation. For any state s,

1. For each tile t_j in r_i, let $C(t_j, r_i)$ denote the number of tiles conflicting with t_j in row r_i.
2. While there is a non-zero $C(t_j, r_i)$ value,
   (a) Move out the tile with the most conflicts from r_i. Let this tile be t_k.
   (b) Set $C(t_k, r_i) = 0$.
   (c) For every tile t_j in conflict with t_k, decrement $C(t_j, r_i)$ by 1.
   (d) Let $lc(s, r_i)$ denote the number of tiles that must be removed from row r_i in order to resolve the linear conflicts in r_i. Increment $lc(s, r_i)$ by 1.
3. Repeat Step 1 and 2 for other rows and columns and sum the values of all $lc(s, r_i)$ and $lc(s, c_i)$.
4. Let $LinearConflict(s)$ denote the minimum number of additional moves necessary to resolve the linear conflicts in state s. $LinearConflict(s)$ = 2 x result from Step 3.

$$h_3(s) = ManhattanDistance(s) + LinearConflict(s)$$

Prove for Consistency. To prove consistency, we must prove that for all s and s', $f(s') \geq f(s)$, where s' is the successor of s.

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

where $g(s') = g(s) + 1$ and $h(s) = ManhattanDistance(s) + LinearConflict(s)$. Assume that tile t_k moves from row r_i to row r_j and stays in the same column. Let $ManhattanDistance(s)$ be $MD(s)$ and $LinearConflict(s)$ be $LC(s)$.

1. **Condition 1:** Both r_i and r_j are not the goal row of t_j.
   $MD(s') = MD(s) \pm 1$. $LC(s)$ is unchanged. Thus, $h(s') = h(s) \pm 1$ and $f(s') = f(s) + 1 \pm 1 \geq f(s)$.
2. **Condition 2:** r_j is the goal row of t_j.
   As t_j moves to its goal row, $MD(s') = MD(s) - 1$. Since r_i is not the goal row of t_j, $lc(s', r_i) = lc(s, r_i)$. As r_j is the goal row, the conflicts in row r_j may or may not increase; so it is either $lc(s', r_j) = lc(s, r_j)$ or $lc(s', r_j) = lc(s, r_j) + 2$. Hence, $h(s') = h(s) \pm 1$ and $f(s') = f(s) + 1 \pm 1 \geq f(s)$.
3. **Condition 3:** r_i is the goal row of t_j.
   As t_j moves away from its goal row, $MD(s') = MD(s) + 1$. As r_i is the goal row, the conflicts in row r_i may or may not decrease; so it is either $lc(s', r_i) = lc(s, r_i)$ or $lc(s', r_i) = lc(s, r_i) - 2$. Since r_j is not the goal row of t_j, $lc(s', r_j) = lc(s, r_j)$. Therefore, $h(s') = h(s) \pm 1$ and $f(s') = f(s) + 1 \pm 1 \geq f(s)$.

All 3 cases show that $f(s') \geq f(s)$. Thus, for any tile which moves from column c_i to c_j while remaining in the same column, $f(s') \geq f(s)$ will still hold by the symmetry of the puzzle.

3 Experimental Setup

3.1 Experiment Goals

The goal of the experiments are to measure the time and space complexity of the search algorithms. The metrics used to measure these complexities are justified below. The rationale for the chosen algorithms and heuristics are elaborated in section 2.
Time Complexity: Number of nodes generated during the search [4, p80]. This is measured by number of the explored states.
Space Complexity: Maximum number of nodes stored in memory during the search [4, p80]. This is measured by the largest summation of number of nodes in the explored set and frontier size during the search. However, note that maximum number of nodes $\neq$ maximum number of nodes in the explored set + maximum size of frontier. This is because the maximum sizes of explored states and frontier size might not occur in the same step; the frontier size is always changing.

3.2 Experiment Implementation Description

For n = $\{1 \dots 25\}$

1. Generate 3×3 matrix M that has n number of steps to reach the goal state.
2. Perform search alogrithm on matrix M
3. Plot the number of nodes generated and maximum nodes in memory against n.
4. To minimise bias, for each iteration, the puzzle solves 30 different puzzles of n steps and the average number of nodes and maximum nodes in memory are taken.

3.3 Results and Discussion

For simple problems (number of steps is less than 15), the differences in the three heuristics' performance are negligible. However, as the puzzle becomes more complicated there is a clear divergence in the time and space complexity as can be seen in Fig. 1 above.

4 First Section

4.1 A Subsection Sample

Please note that the first paragraph of a section or subsection is not indented. The first paragraph that follows a table, figure, equation etc. does not need an indent, either.

Subsequent paragraphs, however, are indented.

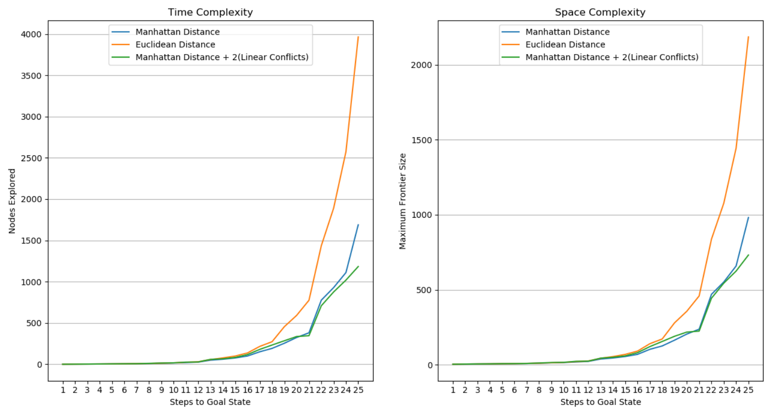

Fig. 1. Performance of the three different heuristics under varying levels of k-puzzle difficulty.

Sample Heading (Third Level) Only two levels of headings should be numbered. Lower level headings remain unnumbered; they are formatted as run-in headings.

Sample Heading (Fourth Level) The contribution should contain no more than four levels of headings. Table 1 gives a summary of all heading levels.

Table 1. Table captions should be placed above the tables.

Heading level	Example	Font size and style
Title (centered)	**Lecture Notes**	14 point, bold
1st-level heading	**1 Introduction**	12 point, bold
2nd-level heading	**2.1 Printing Area**	10 point, bold
3rd-level heading	**Run-in Heading in Bold.** Text follows	10 point, bold
4th-level heading	*Lowest Level Heading.* Text follows	10 point, italic

Displayed equations are centered and set on a separate line.

$$x + y = z \tag{1}$$

Please try to avoid rasterized images for line-art diagrams and schemas. Whenever possible, use vector graphics instead (see Fig. 1).

Theorem 1. *This is a sample theorem. The run-in heading is set in bold, while the following text appears in italics. Definitions, lemmas, propositions, and corollaries are styled the same way.*

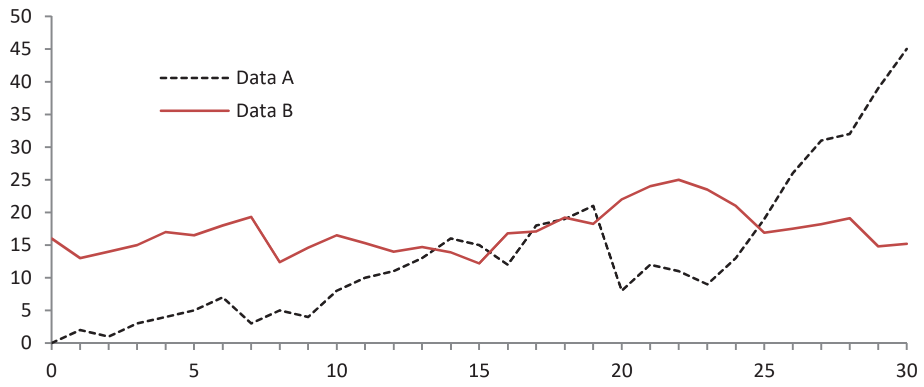

Fig. 2. A figure caption is always placed below the illustration. Please note that short captions are centered, while long ones are justified by the macro package automatically.

Proof. Proofs, examples, and remarks have the initial word in italics, while the following text appears in normal font.

For citations of references, we prefer the use of square brackets and consecutive numbers. Citations using labels or the author/year convention are also acceptable. The following bibliography provides a sample reference list with entries for journal articles [1], an LNCS chapter [2], a book [3], proceedings without editors [4], and a homepage [5]. Multiple citations are grouped [1,2,3], [1,3,4,5].

References

1. Goel, A.: How to check if an instance of 15 puzzle is solvable?
2. Othar Hansson, Andrew E. Mayer, Mordechai M. Yung: Generating admissible heuristics by criticizing solutions to relaxed models
3. Rosalind: Euclidean distance, `http://rosalind.info/glossary/euclidean-distance/`
4. Stuart Russell, Peter Norvig: Artifical Intelligence: A Modern Approach. Pearson Educaiton, Inc., 3 edn.

References

1. Author, F.: Article title. Journal **2**(5), 99–110 (2016)
2. Author, F., Author, S.: Title of a proceedings paper. In: Editor, F., Editor, S. (eds.) CONFERENCE 2016, LNCS, vol. 9999, pp. 1–13. Springer, Heidelberg (2016). https://doi.org/10.10007/1234567890
3. Author, F., Author, S., Author, T.: Book title. 2nd edn. Publisher, Location (1999)
4. Author, A.-B.: Contribution title. In: 9th International Proceedings on Proceedings, pp. 1–2. Publisher, Location (2010)
5. LNCS Homepage, `http://www.springer.com/lncs`. Last accessed 4 Oct 2017

A Proof for Manhattan Distance Consistency

To prove that Manhattan Distance is consistent.

Proof. Proof by Cases

1. $|h(n') - h(n) = 1|$ ($\because c(n, a, n') = 1$, any node n' is 1 step away from node n)
2. Case 1: $h(n') = h(n) + 1$
   (a) $h(n) \leq h(n) + 1 + 1 \implies h(n) \leq h(n') + c(n, a, n')$
3. Case 2: $h(n') = h(n) - 1$
   (a) $h(n) \leq h(n) - 1 + 1 \implies h(n) \leq h(n') + c(n, a, n')$
4. For both cases of $h(n')$, $h(n)$ is consistent. (•)