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)*

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 evaluates the heuristic functions Manhattan Distance, Euclidean Distance, and Linear Conflict. We use these heuristics as our $h_1(n) \geq h_2(n), h_2(n) \geq h_{misplacedtiles}(n)$, and $h_3(n) \geq h_1(n)$.

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 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].

* Supported by Prof Yair Zick and Prof Daren Ler.

2. **Correctness:** Branching factor $b \leq 4$ 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].

2.2 Informed Search

1. **Implementation:** Graph-based A^* search. It improves on greedy best first search (i.e. $f(n) = h(n)$) as it avoids expanding paths that are already expensive (i.e. $f(n) = g(n) + h(n)$).
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].

2.3 h_1 : Manhattan Distance

Justification: Manhattan Distance (1) is consistent and (2) it dominates Euclidean Distance heuristic, thus theoretically more efficient [4, 104].
Definition: Manhattan Distance heuristic is defined as the sum of the horizontal distance and vertical distance from their goal positions [4, p103]. More formally, $h_1(s) = D_v(s) + D_h(s)$.
Proof for Consistency: Proof in appendix A.
Proof for Dominance: $\forall s \in S, h_2(s) \leq D_h(s) + D_v(s)(\because$ triangle inequality), where S denotes the set of possible states.

2.4 h_2 : Euclidean Distance

Justification: Euclidean Distance (1) is consistent and (2) it dominates Misplace Tiles heuristic, thus theoretically more efficient [4, 104].
Definition: Euclidean Distance heuristic is defined as the straight line distance between the tiles from their goal position [3].
Proof for Consistency: Prove by Construction.
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 ($\bullet$).
Proof for Dominance:

$$M(m_{i,j}) = \begin{cases} 0 & \text{if } m_{i,j} \text{ is not misplaced.} \\ 1 & \text{if } m_{i,j} \text{ is misplaced.} \end{cases}$$

Thus, $h_{misplacedtiles}(s) = \sum_{i=1}^{k} \sum_{j=1}^{k} M(m_{i,j})$. Let $D(m_{i,j})$ be the Euclidean distance of tile $m_{i,j}$ to its goal position. Hence,

$$D(m_{i,j}) = \begin{cases} 0 & \text{if } m_{i,j} \text{ is not misplaced.} \\ [1, \sqrt{2}(k-1)] & \text{if } m_{i,j} \text{ is misplaced.} \end{cases}$$

Since $h_2(s) = \sum_{i=1}^{k} \sum_{j=1}^{k} D(m_{i,j})$, $h_2(s) \geq h_{misplacedtiles}(s)$.

2.5 h_3 : Linear Conflict

Justification: Linear Conflict (1) is consistent and (2) it dominates Manhattan Distance heuristic, thus theoretically more efficient [4, 104].
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 [1, 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. Thus, $h_3(s) = ManhattanDistance(s) + LinearConflict(s)$.

Proof for Consistency: Proof By Cases.
To prove consistency, we must prove that for all state 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_j 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. **Case 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. **Case 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. **Case 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 row, $f(s') \geq f(s)$ will still hold by the symmetry of the puzzle.
Proof for Dominance: $\forall s \in S : (h_3(s) = h_1(s) + \text{LinearConflict(s)}) \geq h_1(s) (\because LinearConflict(s) \geq 0)$, where S denotes the set of possible states.

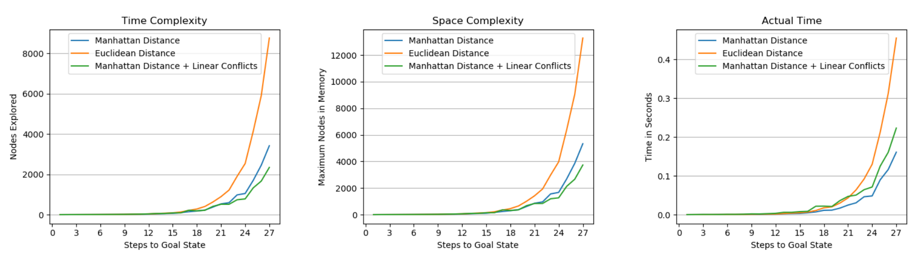
3 Experimental Setup

Experiment Goals: Our experiment aims to measure (1) **time complexity**, (2) **space complexity**, and (3) **actual run time**. (1) shows the theoretical time efficiency in terms of the number of nodes generated. (2) shows the maximum memory required for the search in terms of the max number of nodes stored. (3) shows the real time needed for the algorithm to reach the goal state.
Experiment Implementation Details: For n = $\{1 \dots 27\}$

1. Generate 3×3 matrix M that has n number of steps to reach the goal state.
2. Perform search algorithm on matrix M.
3. Plot the number of nodes generated, maximum nodes in memory, and the actual run time.
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.

4 Results and Discussion

Fig. 1. Performance of the 3 heuristics under 27 levels of k-puzzle difficulty.



In the analysis we denote the Euclidean, Manhattan, and Linear Conflicts heuristics as E, M and L respectively.
From the Time and Space Complexity graphs in Fig 1 above, for step values above 15, we observe that M and L outperform E. This is in line with the theoretical explanation of how M and L dominate E, accounting for the performance difference. In addition, for step values higher than 22, we observe that L outperforms M in terms of both time and space complexity. This is also in line with the theoretical explanation of how L dominates M which translates directly into time and space efficiency [4, p104].
However in terms of actual running time, L takes longer to solve the puzzle compared to M. This is contrary to our preliminary expectation that L will be more time efficient than M. This contradiction can be explained by the extra time required to calculate the heuristic for each node before it is put in the frontier.

References

1. Othar Hansson, Andrew E. Mayer, Mordechai M. Yung: Generating admissible heuristics by criticizing solutions to relaxed models
2. Princeton Computer Science: 8-puzzle, https://www.cs.princeton.edu/courses/archive/spring18/cos226/assignments/8puzzle/index.html
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.

A Rule to Check if k-puzzle is Solvable

Definition: [2]. A pair of tiles form an *inversion* if the values on tiles are in the reverse order of their appearance in the goal state.
Rules [2]. 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 or (2) k is even, the sum of n_i and i is odd.

A Proof for Manhattan Distance Consistency

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. (•)