An Evaluation of Uninformed and Informed Search Algorithms on the K-puzzle Problem

Liow Jia Chen[A0184022R], Tianzhen Ni[A0164726Y], Tan Teik Jun[A0190118L], and Yan Zhiwen[A0164790X]

National University of Singapore

1 Problem Specification

1.1 Overview of the K-Puzzle Problem

The K-puzzle is defined as a problem where given n by n tiles where $n^2 = k + 1$ and each tile could possibly be a space or take on a value from the set $\{1..k\}$ where every tile has to be unique. We want to find an assignment to each tile such that if we start from the top and go left to right and top to bottom we get the tiles increasing order starting from 1 to k and the last tile being the space. To get to this solution, a tile is allowed to move from square A to square B if **(a) A and B are vertically or horizontally adjacent to one another** and **(b) B is blank.**

The 8-puzzle problem is a special case of this problem where $k = 8$.

1.2 Task environment

The task environment for the k-puzzle problem can be defined as a *fully observable*, *single agent*, *deterministic*, *sequential*, *static* and *discrete* environment.

1.3 Problem formulation

- **State:** A state description specifies an assignment of $\{1..k\}$ tiles on the n by n board with one space somewhere on the board.
- **Initial State:** Any valid state could potentially be an initial state although it might or might not have a solution or be solvable.
- **Actions:** Blank space can either move *UP*, *DOWN*, *LEFT* or *RIGHT* if valid depending on where the blank is.
- **Transition Model:** Given a state and an action, the transition model returns the resulting state or swapping the blank space with a neighbouring tile.
- **Goal Test:** Returns true if blank is at the bottom right and tiles are arranged in order left to right and top to bottom.
- **Path Cost:** Since we have uniform step cost = 1, total path cost is equal to the number of steps taken.

2 Technical Analysis

2.1 Uninformed Search: Bidirectional Search

For uninformed search, we implemented bi-directional search. We chose bi-directional search over IDS due to the fact that we wanted to leverage the unique goal state the k-puzzle possess to reduce the running time by a root factor. We acknowledge that by doing so we make a time-space trade off.

Bi-directional search works by searching forward by the initial state and search backwards from the goal state and looking for the point where these two search spaces meets yielding a path from the initial state to the goal state.

The time complexity of bidirectional search is $O(b^{d/2})$ and it's space complexity is also $O(b^{d/2})$. Please refer to AIMA for full correctness proofs.

2.2 Informed Search: A* Search

For informed search, implemented A* search using 3 different heuristics. We chose A* because of its optimal and complete nature given a finite space and an admissible/consistent heuristic.

The time complexity of A* is $O(b^{\Delta})$ where $\Delta = h^* - h$. Since the problem has constant step costs, we can rewrite the time complexity as $O(b^{\epsilon d})$ where relative error, $\epsilon = (h^* - h)/h^*$. The effective branching factor is given by b^{ϵ}. The values for effective branching factor differs across heuristics, they will be given in Section 4. The space complexity is $O(b^d)$. Please refer to AIMA for full correctness proofs.

Heuristic 1: Number of tiles out of row / column
*Definition.*We define a tile to be out of row / column if it is **not** in the same row / column (respectively) as its goal position. The heuristic, h_1, is given by:

$$h_1(n) = (\#\text{ of tiles out of row}) + (\#\text{ of tiles out of column})$$

Constraints relaxed From section 1.1, constraint (a) is relaxed, (b) is removed. Tiles can move from square A to square B if A and B are in the same row or column.

Proof for Consistency A heuristic h is said to consistent iff $h(n) \leq c(n, a, n\prime) + h(n\prime)$. Since all step cost are uniform, $c(n, a, n\prime) = 1$. Hence, h_1 is consistent if $h_1(n) - h_1(n\prime) \leq 1$. First, let us assume that the tile moves from one row to another (within a column). There are 3 possible values of $h_1(n)$ - $h_1(n\prime)$:

1. Tile moves from goal row to out of goal row. (-1)
2. Tile moves from out of goal row into goal row. (+1)
3. Tile moves from out of goal row to another out of goal row. (0)

In all cases, $h_1(n) - h_1(n\prime) \leq 1$. Due to symmetry, the same result applies to movement from one column to another. Therefore, h_1 is consistent.

Heuristic 2: Sum of Manhattan Distance
Definition. Manhattan Distance (*mdist*) between two points measures the shortest path from point x to point y given that the agent can only move up, down, left and right. Let $x = (a, b)$ and $y = (c, d)$ then *mdist* from x to y is defined as $|a - c| + |b - d|$.

$$h_2(n) = \sum mdist(curr\_position, goal\_position) \text{ over all nodes.}$$

This heuristic sums across all their tiles from their current location to to location in the goal state. Intuitively, this is admissible since all tiles will at least take this amount to steps to get into their correct position. This heuristic is also consistent. Since this is a well known and documented heuristic, please refer to Appendix A for the proof of consistency of h_2.

Heuristic 3: Linear Conflict
Definition. Two tiles P and Q are considered to be in linear conflict if P and Q are in the same line and the goal positions of P and Q are in the reverse order of their current positions. The Linear Conflict heuristic, h_3, is given by:

$h_3(n)$ = sum of distances of each tile from their goal position + 2 * (sum of # of tiles to remove from each row/column to resolve linear conflicts)

The first part of the expression is equivalent to the Sum of Manhattan Distance heuristic (h_2). We can simplify the expression as $h_3 = h_2 + h_L$, where h_L is the sum of number of tiles to remove from each row/column to resolve linear conflict. Note that constraint (b) from *section*1.1 is relaxed.

Proof of Consistency We can show that heuristic h_3 is consistent by proving that $h_3(n) - h_3(n\prime) \leq 1$. First, let us assume that tile P moves from row r_{old} to r_{new}, while remaining in the same column. There are 3 possible cases.

1. The goal position of P is neither in r_{old} or r_{new}
   $h_2(n\prime) = h_2(n) \pm 1$ depending on whether the tile moves towards or away from goal position. $h_L(n\prime) = h_L(n)$ because a tile movement cannot add/remove a row linear conflict if the tile does not move to/from its goal row. Summing the parts, we get $h_3(n\prime) = h_3(n) \pm 1$. Hence, $h_3(n) - h_3(n\prime) = \pm 1 \leq 1$.
2. The goal position of P is in r_{new}.
   $h_2(n\prime) = h_2(n) - 1$ because the tile moves 1 step closer to its goal position. $h_L(n\prime) - h_L(n)$ evaluates to 2 (if the move results in a linear conflict in the goal row) or 0 (otherwise). Summing the parts, we get $h_3(n\prime) = h_3(n) \pm 1$. Hence, $h_3(n) - h_3(n\prime) = \pm 1 \leq 1$.
3. The goal position of P is in r_{old}
   $h_2(n\prime) = h_2(n) + 1$ because the tile moves 1 step away from the goal position. $h_L(n\prime) - h_L(n)$ evaluates to -2 (if the move removes a linear conflict from the goal row) or 0 (otherwise). Summing the parts, we get $h_3(n\prime) = h_3(n) \pm 1$. Hence, $h_3(n) - h_3(n\prime) = \pm 1 \leq 1$.

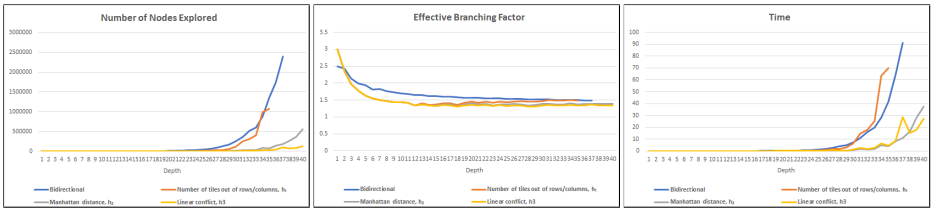
In all cases, $h_3(n) - h_3(n\prime) \leq 1$. Due to symmetry, the same result applies to movement within a row. Therefore, h_3 is consistent.

3 Experimental Setup

We tested the efficiency of our uninformed and informed search algorithms by applying these algorithms to 4x4 puzzles with increasing depth of the puzzle (from 1 to 40), which is defined as the minimum number of moves required to solve the puzzle. For each depth, we conduct 10 trials on different puzzles and take the average.

After the experiment, we tabulate the average time taken for one algorithm to solve a puzzle (t), the average number of nodes explored (n), as well as the estimated effective branching factor (β, for a varied depth of the puzzle). (Effective branching factor is estimated using $n^{1/d}$ since β^d is the dominating component in the series). A detailed explanation of the setup can be found in Appendix B.

4 Results and Discussion



The uninformed bidirectional search always has worse performance than the informed search, which means more nodes explored, larger effective branching factor and longer time consumed. Larger plots can be found in Appendix D.

Among the informed search, the Number of tiles out of rows/columns has the worst performance, which is expected since it is dominated by the other two heuristics. In terms of the number of nodes explored, effective branching factor and the time consumed, it does not beat the uninformed search by much especially in large cases. This is understandable since its estimation is far below the actual cost which makes it a poor heuristics.

The comparison between Manhattan distance and Linear conflict is interesting. We know that Linear conflict dominates Manhattan, which is reflected in the much lower number of nodes explored and lower effective branching factor. However, the Linear conflict shows similar time consumed as the Manhattan distance. This is because the calculation of Manhattan distance with the prior knowledge of the parent node is $O(1)$ time complexity, which the calculation of linear conflict is $O(n^3)$. This difference in calculation complexity compensates for the difference in their number of nodes explored. In the future, the calculation of Linear conflicts may be improved which can further lower its time consumed.

In conclusion, the performance of the informed search is better in all aspects. Among the heuristics analyzed, we would recommend Manhattan Distance or Linear Conflict which have a better performance. If memory is a constraint, Linear Conflict is recommended since it explores few nodes. However, if memory is very constraining, IDS or IDA* may be used although it is not discussed in our experiment.

Appendix A Proof for Manhattan Distance

Definition The Sum of Manhattan Distance heuristic, h_2, is given by:

$$h_2 = \text{sum of the distances of the tiles from their goal positions}$$

Constraints relaxed From Section 1.1, constraint (b) is removed.

Relaxed Problem Tile can move from square A to B if A is adjacent to B.

Proof of Consistency We can show that heuristic h_2 is consistent by proving that $h_2(n) - h_2(n\prime) \leq 1$. There are 2 possible values of $h_2(n)$ - $h_2(n\prime)$:

1. Tile moves horizontally or vertically away from goal position (-1)
2. Tile moves horizontally or vertically towards from goal position (+1)

In all cases, $h_2(n) - h_2(n\prime) \leq 1$. Therefore, h_2 is consistent.

Appendix B Additional Information on Experiment

The puzzles used in the test cases are generated by randomising the moves backwards from the goal state, with validations after puzzle generation to ensure that the puzzles have the correct depth as intended.

We use depth instead of board size to test the performance because the influence of depth on performance, when board size is kept constant, is much more significant than the board size, when depth is kept constant.

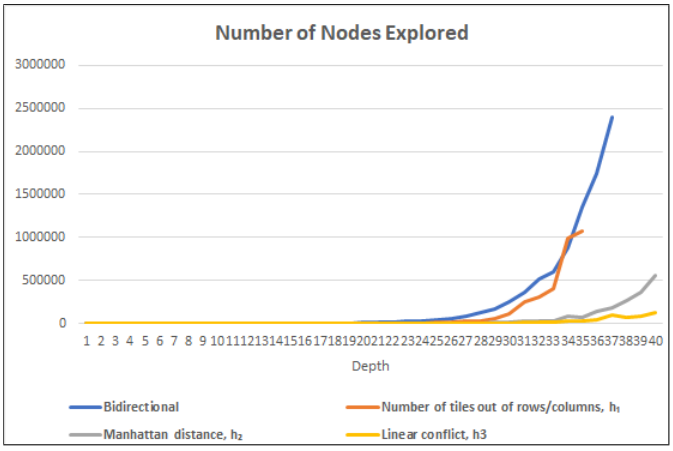
Constrained by the computational capacity, when the time taken for an algorithm to solve a puzzle exceeds 200 seconds, we recorded a timeout and the algorithm will not be used for subsequent trials with larger depth.

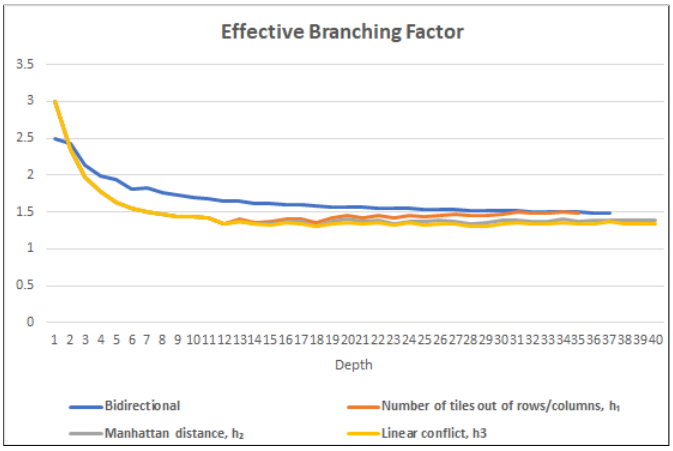
The number of nodes explored represents the number of steps taken by the algorithm to find the optimal solution. The number of nodes and effective branching factor are the direct indications of the efficiency of the algorithm since a better algorithm should take a more targeted approach and explore fewer nodes for each expansion. However, considering the computational overhead for algorithms like linear conflict, we also included the time to provide a more comprehensive assessment of the algorithms from the practical perspective.

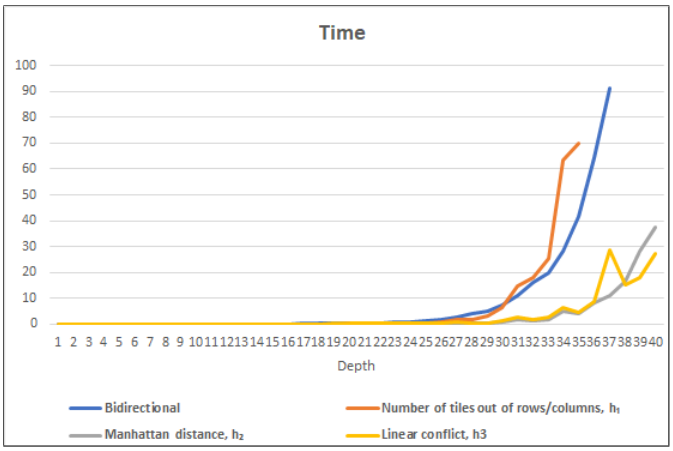
Appendix C Experiment Results

	Uninformed			Informed								
	Bidirectional			Number of tiles out of rows/columns			Manhattan distance			Linear conflict		
Depth	n	β	t	n	β	t	n	β	t	n	β	t
1	2.5	2.5	0.00017266	3	3	0.00014341	3	3	0.00014215	3	3	0.00095446
2	5.9	2.42899156	0.00022345	5.5	2.34520788	0.00020943	5.5	2.34520788	0.00022154	5.5	2.34520788	0.00143652
3	9.7	2.13267124	0.00033014	7.6	1.96609514	0.0002784	7.6	1.96609514	0.00030034	7.6	1.96609514	0.00197031
4	15.5	1.98418848	0.00040658	10.1	1.78270853	0.00034223	10.1	1.78270853	0.00037787	10.1	1.78270853	0.00249701
5	28	1.94729436	0.00074017	11.4	1.62697534	0.00038612	11.4	1.62697534	0.0004178	11.4	1.62697534	0.00270333
6	35.6	1.81373987	0.00072911	14.3	1.55795893	0.00054765	14.3	1.55795893	0.00057406	14.3	1.55795893	0.00328252
7	65.7	1.81824414	0.00137548	17.7	1.5075853	0.00083992	17.7	1.5075853	0.00070496	17.7	1.5075853	0.00438828
8	91.2	1.757921	0.00186009	20.8	1.46136236	0.00079219	20.8	1.46136236	0.00081503	20.8	1.46136236	0.00475705
9	137.4	1.72804115	0.00298357	27	1.44224957	0.000965	27	1.44224957	0.00110869	27	1.44224957	0.00614648
10	189.9	1.68986689	0.00387018	37.4	1.43643891	0.00136142	37.4	1.43643891	0.0015985	37.4	1.43643891	0.00845373
11	309.8	1.68446483	0.00670843	48.3	1.4226108	0.00166309	46.3	1.41715206	0.00188537	45.8	1.41575391	0.0100889
12	422.7	1.6551449	0.00862398	35.2	1.34548405	0.0012692	35	1.34484532	0.00125792	34.4	1.34290885	0.00729861
13	683.9	1.65225226	0.01554253	76.3	1.39575625	0.00267456	64.2	1.37733999	0.00253861	57	1.3647946	0.01227348
14	804.8	1.61268349	0.02018247	64.6	1.34679757	0.00368564	60.4	1.340346	0.00232911	56.6	1.33413929	0.01586018
15	1352.9	1.61715276	0.03332062	120.5	1.37636135	0.00497139	76.9	1.33575996	0.00385759	61.2	1.31557848	0.01546235
16	1828.8	1.59913596	0.0367918	239.4	1.40830026	0.00917628	155.2	1.37066314	0.00585592	131.5	1.3565408	0.0265867
17	2701.8	1.5917027	0.05658789	348.6	1.41107036	0.01273606	201.1	1.36613881	0.00764499	139.6	1.33711799	0.02844269
18	3674.3	1.57784834	0.07760355	224	1.35073245	0.0080111	134.4	1.31293856	0.00552649	113.6	1.30073158	0.02284167
19	5068.7	1.56673144	0.10839584	865.2	1.42752946	0.03331122	437.9	1.37727205	0.0174449	271.6	1.34307902	0.05523169
20	7978.5	1.56709766	0.17532771	1677.2	1.44953679	0.06654155	807.4	1.39750843	0.03310857	480.2	1.36166761	0.09689772
21	11258.3	1.55929138	0.2531245	1656.9	1.42331106	0.06439939	830.3	1.37724496	0.03541222	473.9	1.3409533	0.09409857
22	15102.5	1.54866244	0.34840608	3505.6	1.44919078	0.14811342	1479.2	1.39345238	0.06327138	712.6	1.34795319	0.14315593
23	23440.3	1.54881091	0.55834689	3203.3	1.42042166	0.1328486	829.4	1.33937684	0.03444269	587.9	1.31948512	0.1180397
24	34497	1.54551855	0.87520156	6883.4	1.44513513	0.30554602	2081	1.37486905	0.09127817	1314.4	1.34879821	0.26480227
25	44409.4	1.53426012	1.18268025	8350.9	1.43505759	0.36929877	2438.3	1.36610313	0.11069975	1226.3	1.3290578	0.24489472
26	61640.2	1.52835957	1.68620622	18685.4	1.45978385	0.95529077	4805.1	1.38549174	0.2257272	2129.8	1.34280525	0.43960879
27	89603.1	1.52552515	2.52511387	30165.1	1.46523601	1.53103306	4767.6	1.36846384	0.22015183	2677.6	1.33953339	0.53526523
28	129803.3	1.52271052	3.77162232	32349.4	1.44899421	1.61169751	4120	1.34618105	0.192981	1960	1.31093324	0.38341522
29	170480.3	1.51496486	4.98031225	54827.9	1.45684668	2.90231881	7709.4	1.36155501	0.38284607	2335.9	1.30663292	0.45523593
30	246068.5	1.51252258	7.42963521	113929.5	1.47419366	6.39620473	16358.4	1.38184085	0.84006791	5563.6	1.33304619	1.11417422
31	361813.7	1.51114441	10.9622483	255342.5	1.49425021	14.8866294	28962.7	1.39293321	1.50597353	13909	1.36036267	2.82427881
32	522193.8	1.50897568	16.1190623	304040.2	1.48368462	18.1402569	21339.9	1.36548603	1.11137767	9278	1.33040218	1.88635073
33	602448.8	1.49674941	19.9693201	405158.9	1.47886334	25.2069891	33612	1.37140742	1.7572257	12470.4	1.33081469	2.58147514
34	881905.9	1.49577187	28.0767048	N/A	N/A	Timeout	85174.6	1.39639769	4.75443196	30670.6	1.35507215	6.45566361
35	1355795.3	1.49694397	41.6467373	N/A	N/A	Timeout	74202.1	1.37770029	3.99209573	22906.1	1.33220185	4.65066226
36	1745340.7	1.4906839	64.5562408	N/A	N/A	Timeout	142688.9	1.39052085	7.96470573	42018.3	1.34409154	8.74640939
37	2399758.2	1.48743145	91.2364209	N/A	N/A	Timeout	184336.5	1.38775826	11.155994	95964.3	1.36348892	28.5768299
38	N/A	N/A	Timeout	N/A	N/A	Timeout	271156.3	1.38988691	16.348049	73871	1.34312903	15.3101541
39	N/A	N/A	Timeout	N/A	N/A	Timeout	356396.5	1.3878971	27.9687646	88069.9	1.33903044	18.1366485
40	N/A	N/A	Timeout	N/A	N/A	Timeout	564381.1	1.39248111	37.5519925	127762.6	1.34171444	27.18561

Appendix D Experiment Graphs







References

1. Hansson O., Mayer A., Yung M. (1992). Criticizing Solutions to Relaxed Models Yields Powerful Admissible Heuristics. Retrieved from: https://cse.sc.edu/ñgv/csce580sp15/gradPres/HanssonMayerYung1992.pdf

2. Russell, S. J., Norvig, P. (2009). Artificial intelligence: A modern approach.