A Real-Time Algorithm for the $(n^2 - 1)$-Puzzle

Ian Parberry

Department of Computer Sciences, University of North Texas, P.O. Box 13886, Denton, TX 76203–6886, U.S.A. Email: ian@cs.unt.edu. *URL:* http://hercule.csci.unt.edu/ian.

Abstract

A real-time algorithm for the (n^2-1)-puzzle is designed using greedy and divide-and-conquer techniques. It is proved that (ignoring lower order terms) the new algorithm uses at most $5n^3$ moves, and that any such algorithm must make at least n^3 moves in the worst case, at least $2n^3/3$ moves on average, and with probability one, at least $0.264n^3$ moves on random configurations.

Keywords: Analysis of algorithms, 15-puzzle, $(n^2 - 1)$-puzzle, greedy algorithm, divide and conquer, real-time algorithm, lower bound, Manhattan distance.

1 Introduction

The 15-puzzle is defined as follows. We are given 15 numbered tiles arranged in a 4×4 grid (leaving one tile missing). The aim is to scramble the puzzle and return it to the target configuration shown in Figure 1 by repeatedly sliding an adjacent tile into the blank location. Horden [3] gives an interesting history of this puzzle and its variants. It has long been known that exactly half of the permutations of tiles can be solved (see, for example, Gardner [2]). When the blank is in the lower right-hand corner of the board, the legal configurations of the puzzle are exactly those that can be obtained by performing an even number of transpositions of tiles. Early work on the 15-puzzle includes Johnson [4] and Storey [10].

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

Fig. 1. Target configuration of the 15-puzzle. The blank is black.

The obvious generalization of this puzzle to an $n \times n$ board is called the $(n^2 - 1)$-puzzle. The minimum number of moves to solve each legal permutation of the 8-puzzle has been exhaustively enumerated by Schofield [9]. The minimum number of moves needed to solve the 15-puzzle in the worst case is unknown, but has been the subject of various papers in the Artificial Intelligence literature including Michie, Fleming and Oldfield [7] and Korf [5]. Ratner and Warmuth [8] have proved that the problem of determining the minimum number of moves for any given legal configuration of the $(n^2 - 1)$-puzzle is NP-complete, and they demonstrate an approximation algorithm that makes no more than a (fairly large) constant factor number of moves than necessary for any given legal configuration. Kornhauser, Miller, and Spirakis [6] have shown an algorithm for the $(n^2 - 1)$-puzzle and its generalizations that always runs in time $O(n^3)$.

We present a new algorithm for the $(n^2 - 1)$-puzzle that is based on two standard algorithm design techniques: divide-and-conquer, and greedy algorithms. The new algorithm has the following features:

- it is a real-time algorithm, that is, it generates a series of moves with $O(1)$ computation time required before the first move, and between each successive move,
- it makes no more than 5 times more moves than necessary on the worst-case configuration.
- it makes no more than 7.5 times more moves than necessary on average configurations,
- it makes no more than 19 times more moves than necessary on random configurations, with probability one. (We follow the popular shorthand of writing *with probability one* for a probability that approaches 1 as n increases. In our case, the probability approaches 1 exponentially fast.)

The remainder of the paper is divided into two major sections, the first of which describes the new algorithm and proves the upper bound on the number of moves, and the second of which proves Manhattan distance lower bounds on the number of moves required to solve worst-case, average, and random configurations.

2 The Algorithm

The divide-and-conquer algorithm is described in high-level terms in Figure 2. We will refer to the grid position in row i and column j as *location* (i, j), and refer to the tile that belongs there as *tile* (i, j). The greedy algorithm alluded to in line 2 of Figure 2 simply places tiles $(1, k)$ for $1 \leq k \leq n$ into place, and then tiles $(k, 1)$ for $2 \leq k \leq n$ into place, one at a time. In general, for

```
procedure puzzle(n)
1.   if n = 3 then solve by brute force else
2.      use a greedy algorithm to put the first row and column into place
3.      call puzzle(n − 1) to solve the remaining rows and columns
```

Fig. 2. Procedure for solving an $n \times n$ puzzle.

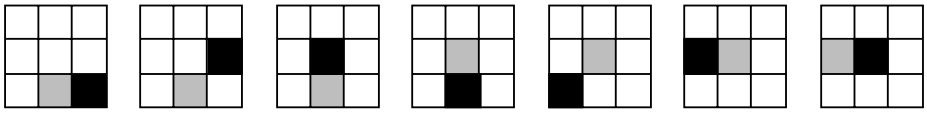

Fig. 3. Moving the shaded tile one place diagonally in six moves.

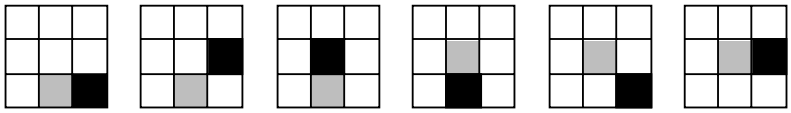

Fig. 4. Moving the shaded tile one place vertically in five moves.

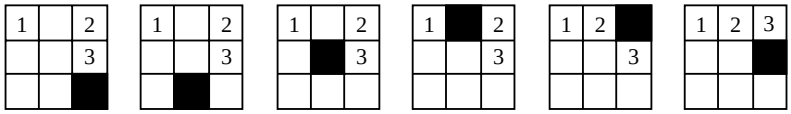

Fig. 5. Completing the end of a row.

$2 \leq k \leq n-1$, this is achieved as follows. We will first describe the procedure for tiles $(1, k)$ where $k \leq n/2$. For convenience we will call tile $(1, k)$ the *target tile*. Without loss of generality, we will assume that the target tile is initially located to the right of the board, that is, it is in location (i, j) for some $i > 1$, $j > n/2$. The case $j \leq n/2$ is similar and is left to the reader.

(i) Move the blank from location $(2, k-1)$ to the location immediately to the right of the target tile. (Note that the algorithm will ensure that the blank is in this location for all but the first tile in each row.)
(ii) Move the target tile to its home location $(1, k)$.
   (a) First move it diagonally to location the correct row or column (whichever it meets first) by repeating the sequence of moves shown in Figure 3.
   (b) Move the target tile vertically or horizontally to its home location $(1, k)$ by repeating the sequence of moves shown in Figure 4 for vertical motion, or its transpose for horizontal motion.

Theorem 1 *Procedure puzzle(n) solves the* $(n^2 - 1)$*-puzzle in at most* $5n^3 + O(n^2)$ *moves.*

Proof. Suppose the blank is in location $(1, k)$ for some $2 \leq k \leq n/2$, and that tiles $(1, j)$ are already home, for all $1 \leq j < k$. The number of moves required to move tile $(1, k)$ home is as follows:

(i) In the worst case the target tile will be in location (n, n), in which case the blank can be moved to the last row in $n - 1$ moves, and thence to the last column in $n - k$ moves, a total of $2n - k - 1$ moves in all. If the target tile is in a different location, fewer moves will suffice.

(ii) In the worst case the target tile will be in location $(n - 1, n)$, in which case:
   (a) it must be moved $n - k - 1$ places diagonally to location $(k + 1, k)$, which takes $6n - 6k - 6$ moves, and then
   (b) it must be moved k places vertically, which will take $5k$ moves.
   Hence, this takes a total of $6n - k - 6$ moves.

Therefore, tile $(1, k)$ can be moved home in at most $8n - 2k - 7$ moves, for $2 \leq k \leq n/2$. A similar argument will show that the same bound holds for $k = 1$. Hence, the leftmost half of the first row can be completed in

$$\sum_{k=1}^{n/2}(8n - 2k - 7) = 15n^2/4 - 4n$$

moves. The rightmost half of the first row can be completed in a similar fashion, using the vertical reflection of the operations shown in Figures 3 and 4. The only difficulty is that of placing tile $(1, n)$ correctly without disturbing the rest of the first row. This is achieved by routing tile $(1, n - 1)$ to place $(1, n)$ instead of place $(1, n - 1)$, and routing tile $(1, n)$ to place $(2, n)$ instead of place $(1, n - 1)$. This takes no more moves than is required to put them in their correct places. The row can then be completed in 5 moves as shown in Figure 5. The total number of moves required to solve the first row is therefore $2(15n^2/4 - 4n) + 5 = 15n^2/2 - 8n + 5$. The first column requires the same number of moves minus $8n-9$ for tile $(1, 1)$ which is already in place. The total for line 2 of the greedy algorithm is therefore $2(15n^2/2 - 8n + 5) - (8n - 9) = 15n^2 - 24n + 19$ moves.

Let $T(n)$ be the number of moves required by procedure puzzle(n). Schofield [9] studies configurations of the 8-puzzle that have the blank in the center, and has shown by exhaustive search that 30 moves are necessary and sufficient to move from one configuration of this form to another (provided, of course, that it is possible to do so at all). Therefore, we can conclude that $T(3) = 34$. The remaining part of the recurrence relation is given by $T(n) = T(n-1)+15n^2 - 24n + 19$ for all $n > 3$. One may very easily verify by induction on n that

$$T(n) = 5n^3 - 9n^2/2 + 19n/2 - 89.$$

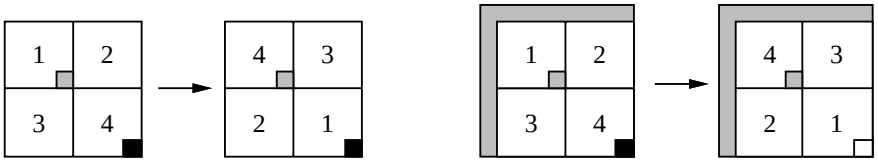

Fig. 6. A configuration with Manhattan distance $n^3 - O(n^2)$, for n even (left) and n odd (right). The blank is black, and shaded tiles are invariant.

□

It can be proved by induction on n that our algorithm operates in *real-time*, that is, it takes $O(1)$ time between moves.

3 Lower Bounds

Define the *Manhattan distance* of tile (i, j) in location (k, ℓ) to be $|i-k|+|j-\ell|$, the minimum number moves required to move to location (i, j). Further define the *Manhattan distance* of a puzzle configuration to be the sum of the Manhattan distances of all of its tiles. Since a single move changes the Manhattan distance of a configuration by unity, it follows that the maximum Manhattan distance is a lower bound on the number of moves necessary to solve the $(n^2 - 1)$-puzzle.

Theorem 2 *There is a configuration of the $(n^2 - 1)$-puzzle that requires at least $n^3 - O(n^2)$ moves to solve.*

Proof. The following construction, illustrated in Figure 6, results in a configuration with Manhattan distance $n^3 - O(n^2)$. Suppose n is even. Divide the (n^2-1)-puzzle into four quadrants numbered 1 through 4 in row-major order. Leaving the blank alone, swap the tiles in quadrant 4 with the corresponding tiles in quadrant 1. Similarly, swap all of the corresponding tiles in quadrants 2 and 3. The resulting configuration is legal since the blank is in the lower right-hand corner and it was obtained using an even number of transpositions, and it has Manhattan distance $n(n^2-2)$ since there are n^2-2 tiles that each have Manhattan distance n. If n is odd, perform the above construction on the lower right $(n-1) \times (n-1)$ sub-board, leaving the first row and column invariant. The resulting configuration is once again legal, and has Manhattan distance $(n-1)((n-1)^2-2) = n^3 - 3n^2 + n + 1$. This can be increased by $2n(n-1)$ by swapping tiles $(1, i)$ with $(n-i+2, 1)$ for $2 \leq i \leq n$, giving a configuration with Manhattan distance $n^3 - n^2 + n + 1$. □

Hence, we can conclude from Theorems 1 and 2 that the algorithm of Section 2 uses a number of moves within a factor of 5 of optimal on worst-case configurations. Next we will consider average-case configurations.

First, let us consider the problem of legal versus nonlegal configurations. Suppose the blank is in its home location, in the lower right-hand corner of the board. Then the legal configurations of the puzzle are exactly those that can be obtained by performing an even number of transpositions of tiles. There is a one-to-one correspondence between legal and nonlegal configurations (obtained by swapping tiles $(1,1)$ and $(1,2)$) that increases the Manhattan distance by at most 1. Therefore, we can conclude that worst-case, average-case and random-case Mahnattan distance over all configurations differs from the worst-case, average-case and random-case Manhattan distance for legal configurations (respectively) by at most unity. If the blank is not constrained to remain in its home location, the same statement holds with a difference of at most a term that is linear in n. Therefore, we will without loss of generality reason about average and random configurations without consideration of whether they are legal.

Theorem 3 *The average configuration of the (n^2-1)-puzzle requires at least $2n^3/3 - O(n^2)$ moves to solve.*

Proof. Let $M_{i,j}$ denote the total Manhattan distance from every location in the board to location (i,j), then the average Manhattan distance for the (n^2-1)-puzzle will be $\sum_{i=1}^{n}\sum_{j=1}^{n} M_{i,j}/n^2$. $M_{i,j}$ can easily be determined as follows.

The Manhattan distance from each tile in column ℓ to location (i,ℓ) is

$$\sum_{k=1}^{i-1} k + \sum_{k=1}^{n-i} k = i^2 - (n+1)i + n(n+1)/2.$$

For a total of n columns, this gives a Manhattan distance of $ni^2 - n(n+1)i + n^2(n+1)/2$ to place all tiles in row i. The Manhattan distance from there to location (i,j) is (remembering that there are now n tiles in each location of row i) similarly $nj^2 - n(n+1)j + n^2(n+1)/2$. The total Manhattan distance is therefore

$$\sum_{i=1}^{n}\sum_{j=1}^{n} ni^2 - n(n+1)i + n^2(n+1) + nj^2 - n(n+1)j = 2n^3(n^2-1)/3.$$

The astute reader will notice that we have overcounted by $n(n-1)/2$ for the blank, giving a total Manhattan distance of $n(2n^4 - 2n^2 - 3n + 1)/3$. The average Manhattan distance is therefore $2n^3/3 - O(n^2)$. □

Hence, we can conclude from Theorems 1 and 3 that the algorithm of Section 2 uses a number of moves within a factor of 7.5 of optimal on average configurations. Finally we consider the Manhattan distance on random configurations.

Theorem 4 *For n sufficiently large, a random configuration of the (n^2-1)-puzzle has a Manhattan distance of at least $0.264n^3$ with probability at least $1-0.9998^n$.*

Proof. Let $B(m,N,p)$ be the probability of obtaining at least m successes out of N Bernoulli trials, each with probability p of success. The following result is a well-known consequence of the Chernoff bounds (see, for example, Valiant and Angluin [1] and Valiant and Brebner [11]): Let $\beta = m/Np - 1$. If $0 \leq \beta \leq 1$, then $B(m,N,p) \leq e^{-\beta^2 Np/2}$.

Consider a random configuration of the (n^2-1)-puzzle. Let p be the probability that a single tile lands within Manhattan distance dn of its home location. Suppose $p \leq \epsilon \leq 2p$. Then, the probability that at least ϵn^2 of the n^2 tiles land within Manhattan distance dn of their respective home locations is bounded above by $B(\epsilon n^2, n^2, p)$. That is, the probability that the total Manhattan distance is less than $d(1-\epsilon)n^3$ is bounded above by $B(\epsilon n^2, n^2, p)$.

Since there are at most $2d^2n^2 + 2dn + 1$ locations at Manhattan distance at most dn from any particular location of the board, $p = 2d^2 + 2d/n + 1/n^2$, which in the limit approaches $2d^2$. Suppose we choose $d = 0.4$, which implies that $p < 0.33$. We can then choose $\epsilon = 0.34$. (The optimum choice for d is actually $1/\sqrt{6}$ and ϵ may be chosen arbitrarily close but not equal to d, but these values are convenient and sufficient for our purposes.) From the above, the probability that the total Manhattan distance is less than $0.264n^3$ is bounded above, for large enough n, by $B(0.34n^2, n^2, 0.33) \leq e^{-n^2/6600} \approx 0.9998^{n^2}$. □

Hence, we can conclude from Theorems 1 and 4 that the algorithm of Section 2 uses a number of moves within a factor of $5/0.264 < 19$ of optimal on random configurations, with probability one.

4 Conclusion and Open Problems

We have demonstrated a real-time algorithm for the (n^2-1)-puzzle that generates a solution that is no more than 5 times longer than necessary in the worst case, no more than 7.5 times longer than necessary on average configurations, and with probability one no more than 19 times longer than necessary on random configurations. It would be interesting to find a faster algorithm.

On the other hand, all three of our lower bounds seem weak. Is there some better lower bound technique than the Manhattan distance? A more feasible open problem would be to obtain a lower bound that is larger than that of Theorem 4 (which is proved by a straightforward method) for the Manhattan distance of a random configuration.

Our algorithm is not an approximation algorithm since there exist configurations reachable by a sequence moves of length $O(n^2)$ for which the algorithm takes $\Omega(n^3)$ moves to solve (for example, move the blank n revolutions around the perimeter of the board). However, we have demonstrated a constant-multiple that is superior to any that have been proved for approximation algorithms in both the worst case and on average. It has been conjectured that various approximation algorithms exhibit a better constant multiple in practice than the theoretical results might lead us to believe (see, for example, Ratner and Warmuth [8]). We cannot comment on that, but have observed that our algorithm may be superior in practice since it works in real time (and hence the user does not have to wait for tiles to move at any point), and empirical observations seem to indicate that random configurations are closer to the worst case than the lower bound of Theorem 4 might lead us to expect.

Horden [3] suggests that one should be allowed to slide a contiguous part of a row or column together instead of just a single tile, and still count it as one move. Our algorithm makes at most $11n^3/3 + O(n^2)$ moves in the worst case under this measure. However, there is no known matching asymptotic lower bound. Both the Manhattan distance and decision tree arguments give lower bounds of $\Omega(n^2)$ only. It is interesting to ask whether this gap between the upper and lower bound can be tightened.

5 Acknowledgements

The author is grateful to the following people for helping with references on the 15-puzzle: Lloyd Allison, David Grabiner, Othar Hansson, Dan Hoey, Jim Holloway, Richard Korf, Victor Miller, Alan Mackworth, and Manfred Warmuth; and also to Martin Huehne and Bettina Sucrow for stimulating email conversations. The author is also grateful to Steve Tate for the idea behind an earlier draft of Theorem 4, and the anonymous referees for several useful suggestions that improved the style and content of this manuscript.

References

[1] D. Angluin and L. Valiant. Fast probabilistic algorithms for Hamiltonian circuits and matchings. In *Proceedings of the Ninth Annual ACM Symposium on Theory of Computing*. ACM Press, 1977.

[2] M. Gardner. *The Mathematical Puzzles of Sam Loyd*. Dover, 1959.

[3] L. E. Horden. *Sliding Piece Puzzles*. Oxford University Press, 1986.

[4] W. A. Johnson. Notes on the 15 puzzle 1. *American Journal of Mathematics*, 2(4):397–399, 1879.

[5] R. E. Korf. Depth-first iterative deepening: An optimal admissible tree search. *Artificial Intelligence*, 27(1):97–109, 1985.

[6] D. Kornhauser, G. Miller, and P. Spirakis. Coordinating pebble motion on graphs, the diameter of permutation groups, and applications. In *25th Annual Symposium on Foundations of Computer Science*, pages 241–250. IEEE Computer Society Press, 1984.

[7] D. Michie, J. G. Fleming, and J. V. Oldfield. A comparison of heuristic, interactive, and unaided methods of solving a shortest-route problem. In D. Michie, editor, *Machine Intelligence 3*, pages 245–255. American Elsevier, 1968.

[8] D. Ratner and M. K. Warmuth. The $(n^2 - 1)$-puzzle and related relocation problems. *Journal for Symbolic Computation*, 10:11–137, 1990.

[9] P. D. A. Schofield. Complete solution of the eight puzzle. In N. L. Collins and D. Michie, editors, *Machine Intelligence 1*, pages 125–133. American Elsevier, 1967.

[10] W. E. Storey. Notes on the 15 puzzle 2. *American Journal of Mathematics*, 2(4):399–404, 1879.

[11] L. G. Valiant and G. J. Brebner. A scheme for fast parallel communication. *SIAM Journal on Computing*, 11(2):350–361, 1982.