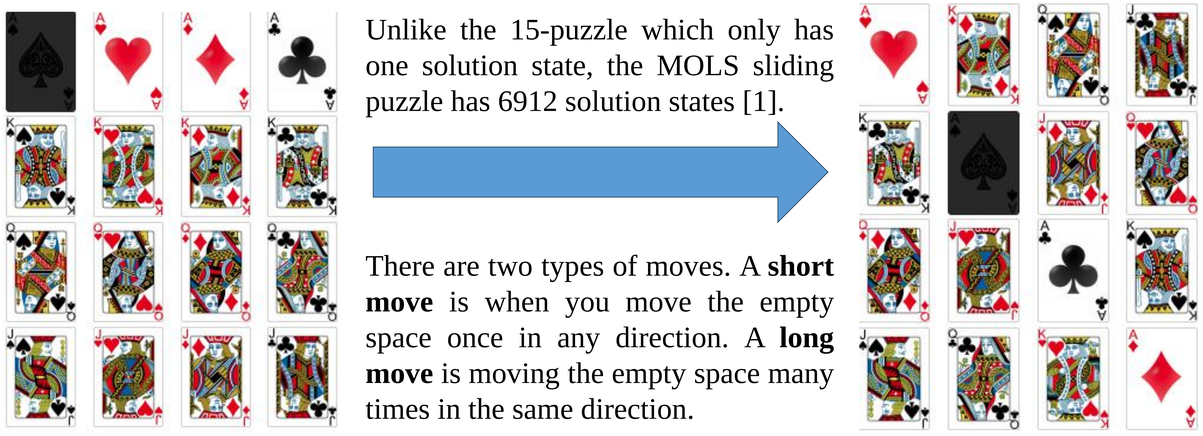
Algorithmic Approaches to MOLS Sliding Puzzle

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

Daniel Henderson, Maxim Chadaev

Introduction

The MOLS sliding puzzle is a type of sliding block puzzle similar to the famous 15-puzzle. The main difference is that instead of tiles numbered 1 through 15, it uses tiles with elements created from the cartesian product of two symbol sets. For our presentation, our two symbol sets will consist of four values and four suits of playing cards. The goal of the puzzle is to arrange the cards in such a way that there are no repetitions in suit or value across all rows and columns of the grid. An example of a solved and unsolved MOLS sliding puzzle is shown below.



Unlike the 15-puzzle which only has one solution state, the MOLS sliding puzzle has 6912 solution states [1].

There are two types of moves. A **short move** is when you move the empty space once in any direction. A **long move** is moving the empty space many times in the same direction.

Notation

Definition 1. *Define a grid* $\boldsymbol{G}$ *to be a* ***MOLS*** *if it is solved. Define a grid* $\boldsymbol{G}$ *to be an* ***MOS*** *if it is solved or unsolved.*

Definition 2. *Let* $\boldsymbol{A}$ *and* $\boldsymbol{B}$ *be symbol sets of size 4. Then,* $\mathcal{G}_{(A,B)} = \{G \mid G \text{ is a MOS over } A \times B\}$ *and*

$\mathcal{L}_{(A,B)} = \{G \mid G \text{ is a MOLS over } A \times B\}$. *Notice that for all symbol sets* $\boldsymbol{A}$, $\boldsymbol{B}$, *it follows that* $\mathcal{L}_{(A,B)} \subseteq \mathcal{G}_{(A,B)}$.

Definition 3. *Define* Γ_ℓ *to be the undirected graph with vertex set* $V(\Gamma_\ell) = \{G \mid G \in \mathcal{G}\}$ *and edge set* $E(\Gamma_\ell) = \left\{(G, G') \in V(\Gamma_\ell)^2 \mid G \in \mathcal{G} \text{ can be converted into } G' \in \mathcal{G} \text{ with exactly one long move}\right\}$. *Define* Γ_s *for short moves analogously.*

Definition 4. *Let* $G \in \mathcal{G}$. *Define the number of* **coincidences** *of* $\boldsymbol{G}$ *as the sum of the number of repetitions it has in each row and column for both symbol sets* $\boldsymbol{A}$ *and* $\boldsymbol{B}$.

Definition 5. *Define* $\boldsymbol{M}$ *as the average shortest amount of moves that it takes to get from a random* $G \in \mathcal{G}$ *to any* $L \in \mathcal{L}$.

Introduction (cont.)

In this poster, we are only concerned with the symbol sets $\boldsymbol{A} = \{\spadesuit, \clubsuit, \diamondsuit, \heartsuit\}$ and $\boldsymbol{B} = \{A, K, Q, J\}$. For brevity, we will denote $\mathcal{G}_{(A,B)}$ and $\mathcal{L}_{(A,B)}$ as $\mathcal{G}$ and $\mathcal{L}$ respectively.

Our goals are to find $\boldsymbol{M}$ and develop an algorithm to solve any configuration of the puzzle in a reasonably short, if not the shortest, number of long and short moves possible.

Breadth-First Search (BFS)

One foundational method for determining the shortest path from any arbitrary configuration $G \in \mathcal{G}$ to the closest MOLS $L \in \mathcal{L}$ is through the application of a Breadth-First Search (BFS) over the graphs Γ_s and/or Γ_ℓ, terminating on the first encounter of an element in $\mathcal{L}$. The benefit of this approach is that it ensures that, upon reaching L, the calculated distance between G and L corresponds to the minimal number of moves required to solve the puzzle. Furthermore, this approach will always find a solution given enough runtime and computational power, provided a path to a solution exists.

Using this method, we found that solutions to the puzzle occur very rarely within 14 long moves (around 22 short moves) of a random configuration. Unfortunately, simulating any further down the graph costs too much computationally to be worthwhile.

Still, this algorithm can be used to help confirm the validity of a formula that estimates the average shortest path from $G \in \mathcal{G}$ to a $G' \in \mathcal{G}$ by seeing if the formula's predictions match the algorithm's results.

Average Number of Moves Model

First, let us assume that MOLS are uniformly distributed in Γ_ℓ. Next, we calculate the branching factor of Γ_ℓ. We use our BFS to count the number of branches at each depth. We then use the Nelder-Mead method to minimize the mean squared error (MSE) and find the best fit b for $y = b\hat{}x$, resulting in $b \approx 3.05$. Given that the graph is tree-like, we estimate the average number of moves as $M \sim \log_{3.05}\left(\frac{16!}{6912}\right) \approx 20$ long moves.

Best-First Search (BeFS)

A method to find a short path from any configuration $G \in \mathcal{G}$ to the nearest MOLS $L \in \mathcal{L}$ is to use a Best First Search (BeFS) algorithm. Although BeFS does not ensure the shortest path, it offers a sufficiently short one efficiently. Like BFS, BeFS always finds a solution. It prioritizes states based on the number of coincidences and path length, favoring states closer to MOLS for child generation, thus increasing the likelihood of reaching one. This method was tested 382 times with random starts; a solution was reached every time.

Table 2: Statistics for Short Moves

Statistic	Value
Mean	34.88
Median	35.00
Standard Deviation	4.92
Lower Bound	14
Upper Bound	47

Table 1: Statistics for Long Moves

Statistic	Value
Mean	27.67
Median	27.00
Standard Deviation	4.35
Lower Bound	13
Upper Bound	40

Conclusion

We believe this game offers significant potential for advancements in AI, machine learning (ML), and deep learning (DL). The simplicity of the game's design contrasts with the complexity required for our algorithms, including the BeFS algorithm. The game's multiple solution states add complexity to assessing position quality. We invite the community to engage with our findings, as insights and discussions will enhance our understanding of ML's capabilities in game theory and decision-making.

[1] OEIS Foundation Inc. Sequence A072377, numbers n such that n = (product of digits of n^3). *Online Encyclopedia of Integer Sequences, n.d.* Retrieved April 21, 2024, from https://oeis.org/A072377.