International Journal for Modern Trends in Science and Technology, 6(12): 165-170, 2020
Copyright © 2020 International Journal for Modern Trends in Science and Technology
ISSN: 2455-3778 online
DOI: https://doi.org/10.46501/IJMTST061232
Available online at: http://www.ijmtst.com/vol6issue12.html

Sliding Puzzle CAPTCHA Analysis

Saurabh Chauhan[1] | Shreya Kapoor[2]

[1]B.Tech Scholar, Department of IT, Maharaja Agrasen Institute of Technology, Delhi, India
[2]Assistant Professor, Department of IT, Maharaja Agrasen Institute of Technology, Delhi, India

To Cite this Article
Saurabh Chauhan and Shreya Kapoor, "Sliding Puzzle CAPTCHA Analysis", *International Journal for Modern Trends in Science and Technology*, 6(12): 165-170, 2020.

Article Info
Received on 08-November-2020, Revised on 28-November-2020, Accepted on 02-December-2020, Published on 05-December-2020.

ABSTRACT

Today a number of everyday activities are done through the Internet. To perform such web services users must register in relation to websites or fill some form. In such websites, some hackers write malicious programs called bots that destroy website resources by creating fake registrations or form submissions. This false registration may adversely affect the performance of websites. Therefore, it is necessary to distinguish between actual human users and Web bots (or computer programs) via tests known as CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart). Most of the conventional CAPTCHA challenges have become an easy nut to crack for bots, hence calling a need for better CAPTCHA alternatives. This paper studies the infamous 'Sliding Puzzle' as an alternate CAPTCHA challenge and analysing the complexity to solve it via bot script.

KEYWORDS:CAPTCHA, Web Accessibility, Sliding Puzzle, Hill Climbing, Web Security*

I. Introduction

CAPTCHA ("Completely Automated Public Turing test to tell Computers and Humans Apart") is a type of challenge-response test used by many web applications to ensure that the response is not generated by a computer. Visual CAPTCHA implementations are often vulnerable to various kinds of attacks.

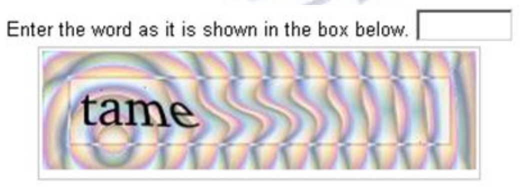


Fig 1.1: Text Based CAPTCHA

As per numerous reports [1], fraudsters find sophisticated ways to get past Captchas. For example, some bots that practice Captcha-solving depend on the artificial intelligence of the auto image or sound recognition. But another tactic, which is becoming more and more common, is relying on a time-tested problem-solving device i.e. the human brain. Surprisingly, Captcha farms are not about people assisted by bots but about human-assisted bots. Such huge database of solved generic CAPTCHAs is then used by spam bots.

ONLINE CAPTCHA SOLVING JOBS
unto nations
trieste
modernday
Captcha Work: 8 Best Captcha Entry Jobs Sites...
Images may be subject to copyright. Learn More
Related images
See more
ONLINE CAPTCHA SOLVING
online captcha solving jobs

Fig 1.2: CAPCTHA solving jobs

There are plenty of job postings online for captcha solving that pay people for bulk solving. This practice is majorly reported in developing countries where people are ready to solve at cheaper rates.

Greg Mori and Jitendra Malik of UC Berkeley Computer Vision Group, Simon Fraser University, in their research paper 'Breaking a Visual CAPTCHA' [2], write about how their method can successfully pass Yahoo's 'gimpy' CAPTCHA test 92% of the time using object recognition algorithm. Gimpy is a more difficult variant of a word-based CAPTCHA that generates a picture of distorted and skewed alphanumeric string. As a challenge the user is supposed to recognize the word embedded in the picture. Such vulnerability of generic CAPTCHAs has erupted a demand for better alternatives that are based on user interactions rather than mere text recognition.

Puzzles like jigsaw, image rotation are getting very common these days as CAPTCHA challenge, and one such puzzle challenge is the 'n-Sliding Puzzle', as shown in Fig 1.3. The paper aims to analyze sliding puzzle challenge by visualizing the AI algorithms used to solve it and the feasibility of making a spam bot for the same. The effectiveness of different algorithms in solving the puzzle is hence analyzed in section III.

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

Fig 1.3: 15-Sliding Puzzle (Solved)

II. Methodology

This section consists of brief introduction to the problem and the algorithm's working used for solving. For the working project used for observations, Hill climbing algorithm has been used which will be briefly introduced along with the slight variation adopted to design the solver-bot. All the insights and observations that follow in section III are hence compiled after visualizing the same variant of algorithm.

N-Sliding Puzzle:

The 15-Sliding Puzzle was created by Samuel Loyd in the 1870's [3]. In this game, 15 tiles are arranged in a 4 × 4 grid with one empty space. Tiles are rated from 1 to 15. Figure 2.1-left shows a possible initial configuration of the 8-tile puzzle while Figure 2.1-right shows the solved configuration of the same.

The state of the puzzle can be changed by sliding one of the numbered tiles - next to the empty space - into the empty space. The action is indicated in the direction, in which the tile with the number is moved. In each state, a set of possible actions is therefore a subset of {up, down, left, right}. The goal is to get the puzzle converted into the final state shown in Fig. 2.1 right through the sequence of actions. Configuration of the puzzle is considered as solvable if there is a sequence of actions that leads to the goal state.

The 15-sliding-puzzle problem is well researched in the optimization community. Search algorithms, such as A ∗ and mountain climbing, can be used to find potential solutions. The biggest difficulty of the game is the size of the state space. 15 -sliding-puzzles with (16)! different states. Due to the large size of the state space, complete search is difficult and hence the 15-puzzle problem is considered as one of the most popular benchmarks for heuristic search algorithms.

6		5
4	1	8
7	2	3

1	2	3
4	5	6
7	8	

Fig 2.1: A possible start state (left) and the goal state (right)

The empty space cannot move diagonally and can take only one step at a time.All possible moves of an Empty tile are as follows:

O – position: total possible moves are 2,
x – position: total possible moves are 3
- position: total possible moves are 4

O	X	O
X	#	X
O	X	O

Fig 2.2: All possible moves of empty tile

Hill Climbing Algorithm (with depth variation): Hill Climbing is a heuristic search used for mathematical optimization problems in the field of Artificial Intelligence [4]. Given a large set of inputs and a good heuristic function, it tries to find a sufficiently good solution to the problem. This solution may not be the global optimal maximum.

- 'Heuristic search' means that this search algorithm may not find the optimal solution to the problem. However, it will give a good solution in reasonable time.

- A heuristic function is a function that will rank all the possible alternatives at any branching step in search algorithm based on the available information. It helps the algorithm to select the best route out of possible routes.

Features of Hill Climbing:
A.Variant of generate and test algorithm:
It is a variant of generate and test algorithm. The generate and test algorithm is as follows:

i. Generate possible solutions.
ii. Test to see if this is the expected solution.
iii. If the solution has been found, quit; else go to step i.

B. Uses the Greedy approach:
At any point in state space, the search moves in that direction only which optimizes the cost of function with the hope of finding the optimal solution at the end.

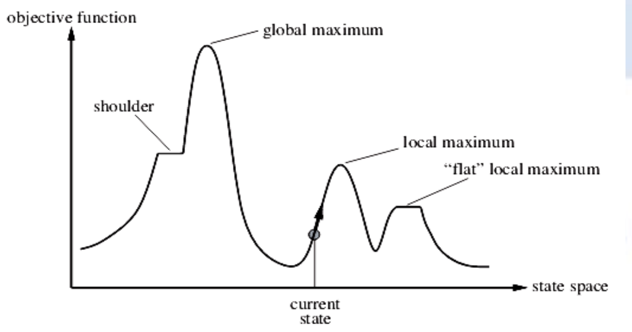


Fig 2.3: Various regions in Hill-Climbing algo

Hill climbing is an iterative heuristic based search algorithm and is widely used in optimisation of search problems. In 'An Introduction to Machine Learning' book by Miroslav Kuba [5], it is defined as follows:

i. Create two lists, L and L_{seen}. At the beginning, L contains only the initial state, and L L_{seen} is empty.
ii. Let n be the first element of L. Compare this state with the final state. If they are identical, stop with success.
iii. Apply to n all available search operators, thus obtaining a set of new states. Discard those states that already exist in $L_{seen.}$ As for the rest, sort them by the evaluation function and place them at the front of L.
iv. Transfer n from L into the list $L_{seen,}$ of the states that have been investigated.
v. If L = 0, stop and report failure. Otherwise, go to ii.

Evaluation function at step iii basically just calculates the distance of the current state from the final state (Manhattan Distance [6]). For example:

Current state

1	2	3
	4	6
7	5	8

Goal state

1	2	3
4	5	6
7	8	

In the current state all values except (4,5 and 8) are at their respective places, so, the heuristic value is 3.(Total of three values are misplaced to reach the goal by distance of 1 each).

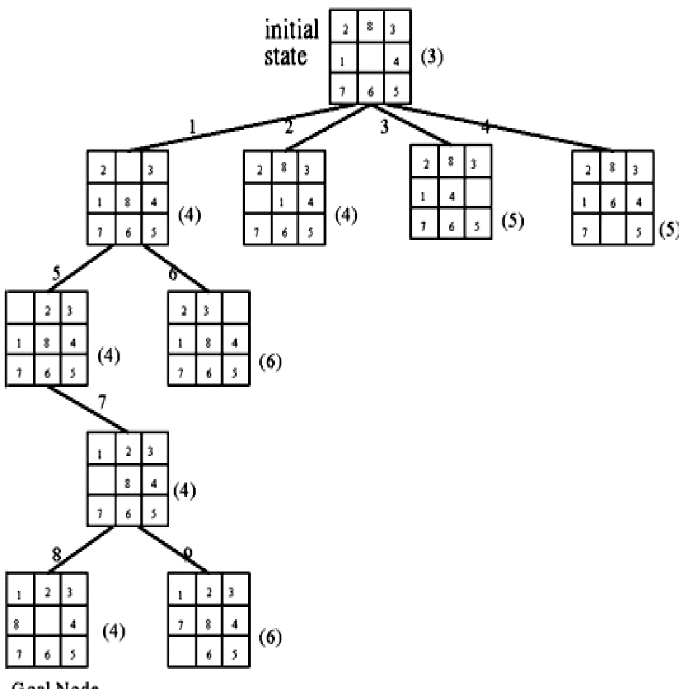


Fig 2.4: Sliding Puzzle state space (while using Hill-Climbing

Solving Sliding Puzzle with Hill-Climbing (with variable Depth):

Instead of BFS or other non-heuristic searches, that are slower, the puzzle solver bot tries to solve the problem with Heuristic Search that is also known as Informed Search (Hill Climbing/A*)
To solve the problem with Heuristic search or informed search we have to calculate Heuristic values of each node to calculate cost function as discussed in previous part. Moreover, to effectively chose the right path, Hill Climbing with the depth-first approach is put into use. The algorithm's idea revolves around traversal of a path for a defined number of steps(depth) to confirm that it's the best move.

i. Iterate over all the possible next moves/states for the current state.

ii. Redo step i until depth 'd' has been reached and thus generating a tree of height d.

iii. Pick the move/state with minimum cost(dF)

iv. Return dF to parent node so that evaluation can be done at depth-1 level.

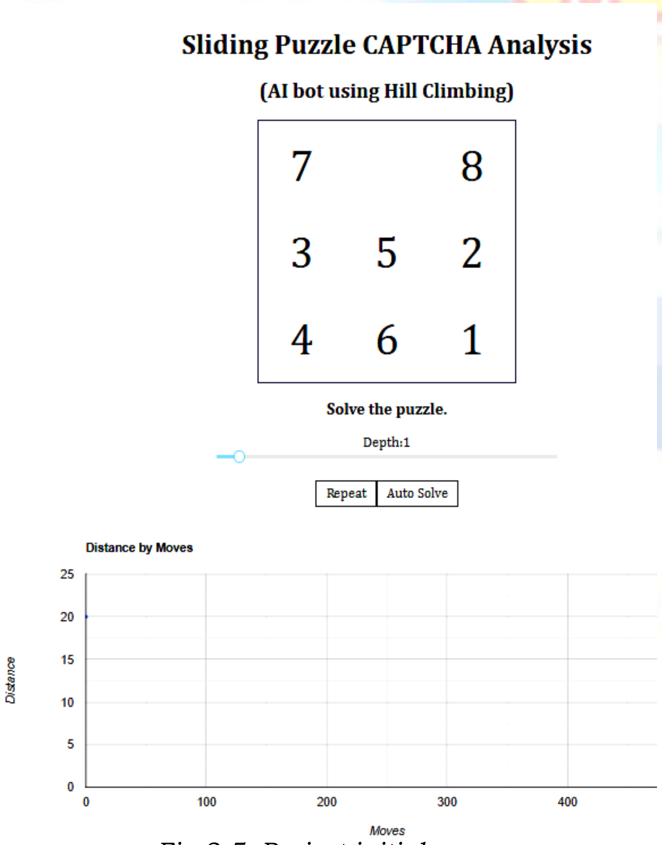


Fig 2.5: Project initial screen

III. OUTPUT AND OBSERVATIONS

Figure 2.5 shows the initial screen consisting of a visualizer graph. Y axis represents Cost/Heuristic function (i.e. Manhattan Distance here) and X axis represents number of moves (or states searched). Upon clicking Auto Solve the bot starts solving the puzzle using Hill Climbing with the depth set by the user.

Hill climbing in our algorithm evaluates the possible next states and picks the one which has the least distance. It also checks if the new state after the move was already encountered before, if yes, then it skips the move and evaluates the next one, this is to prevent an infinite loop. Drawbacks observed are as follows:

- As the empty tile can only be filled by its neighbours, Hill climbing sometimes gets locked and can't reach solution state. It's one of the major drawbacks of this algorithm and any other algorithm using Greedy approach.

- Another drawback is local optima. The algorithm decides the next move(state) based on immediate distance (i.e. of current state), assuming that the small improvement right now is the best way to reach the goal state. However, the path chosen by this greedy approach may lead to higher cost (more steps) later.

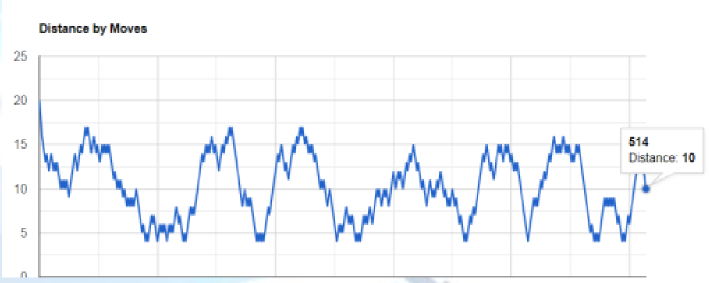


Fig 3.1: Unsolved puzzle even after 500+ moves on depth=0

The drawback of local optima can be reduced to an extent by using depth variation of Hill-climbing approach, where the algorithm traverses the state space tree up to 'd' depth and then out of all those states it finally choses the one that optimizes the heuristic function the most.

However, on the down side, it is memory intensive, proportional to the depth value used. This is due to the system keeping track of future states as per the depth level set.

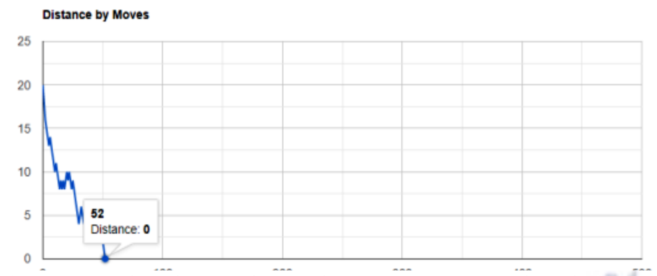


Fig 3.2: Solved puzzle with 52 moves on depth=4 (Note how there are lesser hills when higher value of depth 'd' is set)

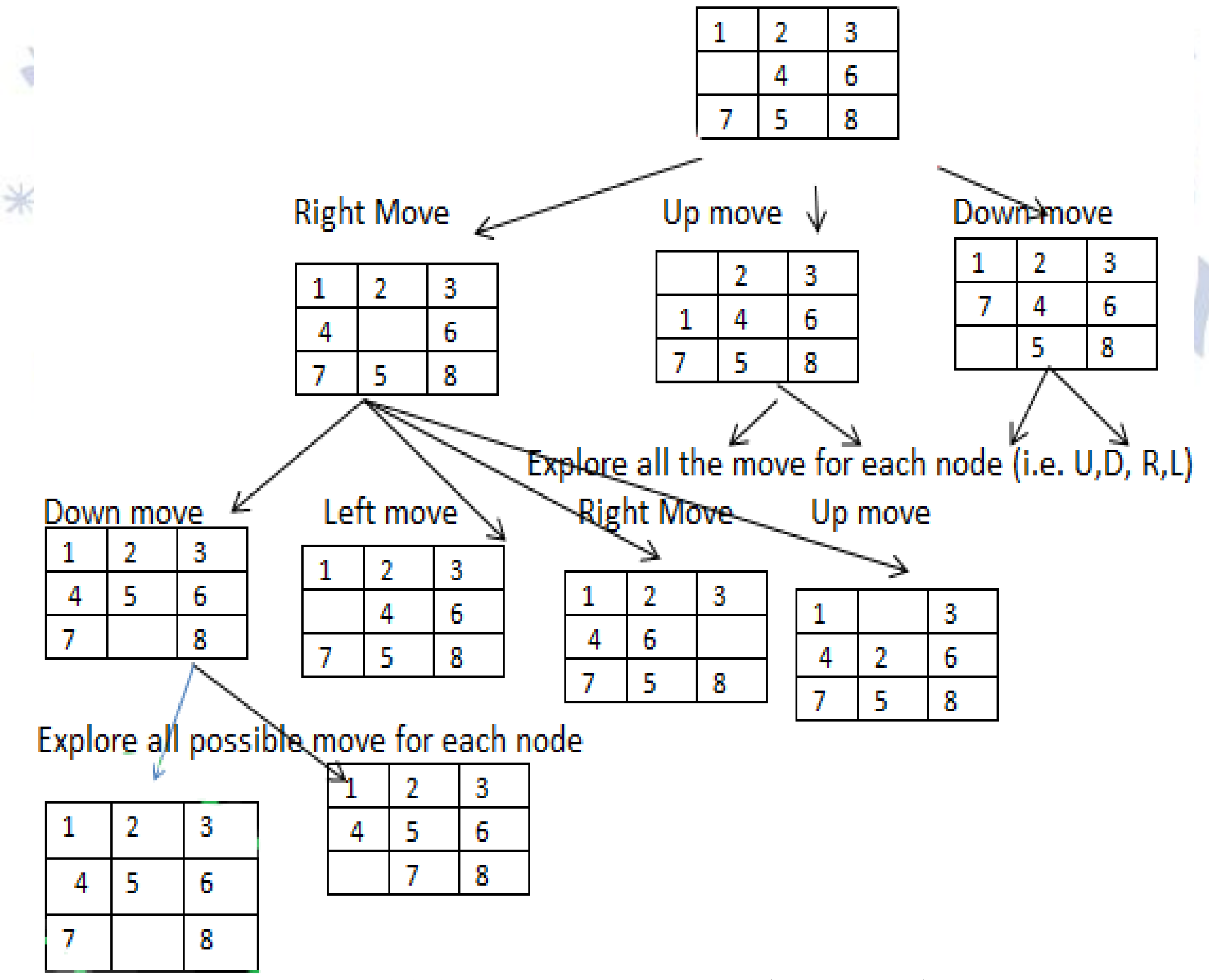
Comparison with BFS/A*:

Merits of using Hill Climbing Algo for solver-bot (compared to BFS,DFS, A*) are as follows:

- Uses Greedy Approach, hence lesser moves might solve the problem.
- Search space is reduced as the search only moves in direction of neighbour that optimises the cost function the most at that level.
- It is good in solving the optimization problem while using only limited computation power.

Demerits of using Hill Climbing Algo for solver-bot (compared to BFS,DFS, A*) are as follows:

- Whole state space is not explored; hence a solution is not always guaranteed, unlike A* etc.
- Algorithm might terminate on reaching local optima even if a better solution exists in global space.



Right Move
Up move
Down move
Explore all the move for each node (i.e. U,D, R,L)
Down move
Left move
Right Move
Up move
Explore all possible move for each node

FIG 3.3: STATE SPACE WHEN USING BFS (NO HEURISTIC)

IV. CONCLUSION

Interactive Challenge based CAPTCHA like Sliding puzzle provide following benefits over generic text-based captchas:

- Better user experience (UX) by eliminating the need to read distorted text. More engaging to users.
- No language barrier as challenge instructions (and question, if any) can be easily translated using browser tools/screen reader and the challenge itself utilizes numbers set that is universal.
- Hard to write mass spam bots for challenges that aren't text recognition based. Exploring whole state space to solve such puzzle requires computational as well as time resource which is not feasible for mass spam bots targeting multiple websites at once.

On the downside, sliding puzzles are definitely time taking and even hard for users to solve as a CAPTCHA challenge. There is a need to strike a right balance between security and challenge complexity when it comes to choosing CAPTCHA for some website. There are 3 basic properties that CAPTCHAs must ideally satisfy:

- easy for human users to pass.
- easy for a tester machine to generate and check.
- hard for a software robot to pass.

REFERENCES

[1] DataDome,"I'm (really) not a robot: Captcha farms and the challenges of Captcha bot detection"2020, accessed 15 November 2020, *https://datadome.co/bot-detection/how-to-detect-captcha-farms-and-block-captcha-bots/*

[2] Greg Mori, Jitendra Malik , UC Berkeley Computer Vision Group, Simon Fraser University,"Breaking a Visual CAPTCHA", accessed 15 November 2020, *https://www2.cs.sfu.ca/~mori/research/gimpy/*

[3] Wikipedia,"Sliding Puzzle"2020, accessed 15 November 2020, *https://en.wikipedia.org/wiki/Sliding_puzzle*

[4] GeeksForGeeks,"Introduction to Hill Climbing | Artificial Intelligence"2019, accessed 15 November 2020, *https://www.geeksforgeeks.org/introduction-hill-climbing-artificial-intelligence/*

[5] Kubat, Miroslav,"An Introduction to Machine Learning" - Hill climbing search algorithm Table 1.22015, accessed 15 November 2020, *https://www.springer.com/gp/book/9783319348865*

[6] Fred E. Szabo PhD, in "The Linear Algebra Survival Guide", 2015, ScienceDirect- "Manhattan Distance", accessed 15 November 2020, *https://www.sciencedirect.com/topics/mathematics/manhattan-distance*