**Project 2 Report**

**A Sudoku Solver using Differential Evolution AND the Backtracking Algorithm.**

|  |  |  |
| --- | --- | --- |
| **المستوى الدراسي** | **رقم الطالب** | **إسم الطالب** |
| **الثالث** | **202000524** | **عبدالرحمن محمد جمعه شافعى** |
| **الثالث** | **202000156** | **امنية فصيح محمد راغب** |
| **الثالث** | **202000526** | **عبدالرحمن محمد عبدالعزيز محمد** |
| **الثالث** | **202000521** | **عبدالرحمن علاء عبدالرحمن محمد** |
| **الثالث** | **202000639** | **فاطمه فتحى محمد مختار** |
| **الثالث** | **202000372** | **ساره عبدالحفيظ على عبدالحفيظ** |

1. **project idea in details**

Definition of a Sudoku Puzzle

Sudoku is a puzzle game which is made up of a board of n² × n² cells, divided in(n×n )boxes, containing whole numbers in the range 1 to n² (n is called the order of a sudoku puzzle) . The goal of the game is to fill the cells so that each row, column and box contain all the numbers ranging from 1 to n² . An important consequence of this is that there must be exactly one number each for each row, column, and box. Every sudoku puzzle start with some cells already containing a number. This is known as a clue, which is a part of the solution. Using the clues as well as the definitions for a solution, one is expected to find a solution.

A Sudoku puzzle’s complexity is defined by the order of the puzzle as well as the number of clues given; as the order increase and the number of clues decrease, the complexity of the puzzle increase.

As the complexity of the puzzle increase, the likelihood of finding a solution within a reasonable time decrease. The hardest Sudoku puzzles have 17 clues, as puzzles with fewer clues will not have unique solutions.

Text

Description automatically generated

Different constraints

Constraints as inhibitory effects of cells the most important feature of a sudoku puzzle is that every cell contains a unique number rowwise, columnwise and boxwise. One interpretation of this is that, given that a cell contains a number, each cell associated to that cell rowwise, columnwise or boxwise is inhibited from adopting the same number.

The method for checking this constraint depends on how the Sudoku puzzle is represented in memory.

Representations of Sudoku puzzles

This chapter will describe different ways in which a Sudoku puzzle can be represented in memory. Depending on representation, common operations for modifying the data set and retrieve information for analysis will vary in time complexity. Be wary that n in the following sections is the order of a Sudoku puzzle.

**Standard Representation**

The standard representation of a sudoku puzzle is a n² × n² board, divided into n boxes. In working memory, this representation can be stored as an n² × n² integer matrix. The memory complexity is n4

Calculating the inhibitory effect for a number on a cell

The inhibitory effect on a cell has to be calculated by checking the numbers of every associated cell rowwise, columnwise and boxwise. The operation for checking if a cell is inhibited from adopting a given number will have a worst case time complexity of 3 n² .

**Definitions**

Table 1 depicts the meaning of different terms used in this paper is given in is this paper

|  |  |
| --- | --- |
| Term | Definition |
| Cell | A single square in the puzzle |
| Box | A group of 3X3 cells |
| Grid | A group of 3X3 boxes |
| Column | A column of 9 cells |
| Row | A row of 9 cells |
| Given value | A value (between 1-9) that was already assigned to cell at the start of a game, which cannot be changed |
| Valid input | A value that can be inserted into a cell without violating the rule of the game at time when the value is inserted. Does not mean the value inserted is the correct value for the game |
| Possible values | Lists of valid inputs for a cell |

**2- Main functionalities**

Diagram

Description automatically generated

**3- Similar applications in market**

There are a lot of applications that solve the sudoku problem like:

https://[www.sudoku-solutions.com](http://www.sudoku-solutions.com)

<https://www.sudokuspoiler.com/sudoku/sudoku9>

<https://www.sudokuwiki.org/sudoku.htm>

These applications use different algorithms and techniques to solve

The sudoku problem online and by just using your browser

**Literature review of academic publications**

**Abstract**

Our project, Artificial Intelligence based real time Computer Vision and puzzle solver using web-cam allows to provide the computer system with the Sudoku-puzzle in real time and give the optimum solution to it. This project not only involves use of Artificial Intelligence, but also makes use of Computer vision, thus combining the two major concepts and increasing the future scope. In this project the computer tries to analyze the environment by capturing the multiple image bursts from the real time and from those images it would detect the Sudoku grid. For the detection of the grid the use of Hough Transform technique has been made. Then the numbers are detected using OCR i.e. Optical Character Recognition. Thus, the system gets the total knowledge of the puzzle and then computes the final solution by making the use of Artificial Intelligence based strategies for getting the optimal solution to the Sudoku puzzle problem.

**Keywords** – Sudoku puzzle, Artificial Intelligence

**INTRODUCTION**

In this project, on the high-level three steps are performed: 1) See/Capture the image from the real time. 2) Identify/Detect whether the captured image is the same object or the puzzle query. 3) Solving the puzzle by making use of Artificial intelligence based strategies. Computer vision is a field that includes methods for acquiring, processing, analyzing, and understanding images and, in general, high-dimensional data from the real world in order to produce numerical or symbolic information, e.g., in the forms of decisions. A theme in the development of this field has been to duplicate the abilities of human vision by electronically perceiving and understanding an image. This image understanding can be seen as the disentangling of symbolic information from image data using models constructed with the aid of geometry, physics, statistics, and learning theory. Computer vision has also been described JOURNAL OF ENGINEERING, COMPUTING & ARCHITECTURE Volume 11, Issue 4, APRIL - 2021 ISSN NO:1934-71971 http://www.journaleca.com/ Page No: 11 as the enterprise of automating and integrating a wide range of processes and representations for vision perception.. The image data can take many forms, such as video sequences, views from multiple cameras, or multi-dimensional data from a medical scanner. As a technological discipline, computer vision seeks to apply its theories and models to the construction of computer vision systems.

Artificial intelligence (AI) is the intelligence exhibited by machines or software. It is also the name of the academic field of study which studies how to create computers and computer software that are capable of intelligent behavior. Major AI researchers and textbooks define this field as "the study and design of intelligent agents", in which an intelligent agent is a system that perceives its environment and takes actions that maximize its chances of success. John McCarthy, who coined the term in 1955, defines it as "the science and engineering of making intelligent machines". The central problems (or goals) ofAI research include reasoning, knowledge, planning, learning, natural language processing (communication), perception and the ability to move and manipulate objects. General intelligence is still among the field's long-term goals. Currently popular approaches include statistical methods, computational intelligence and traditional symbolic AI. There are a large number of tools used in AI, including versions of search and mathematical optimization, logic, methods based on probability and economics, and many others. The AI field is interdisciplinary, in which a number of sciences and professions converge, including computer science, mathematics, psychology, linguistics, philosophy and neuroscience, as well as other specialized fields such as artificial psychology. Here, in the project AI strategies are used to solve the puzzle acquired using computer vision. For solving the puzzle, the brute force attack strategy, standard Sudoku puzzle solving algorithms can be used. After performing the OCR the following data is then provided to the brute force attack or the problem solving engine. Thus, we get the final output here, i.e. the solved Sudoku puzzle. In our project, use of computer vision is been made for detecting the puzzle itself from the real time environment. It plays a key role as the Sudoku puzzle i.e. the query or input for processing is been captured and provided to the system. The system should get the appropriate information for further processing and to maintain optimality while producing the output. According to the steps mentioned above, we need to see and detect the puzzle from the real time environment and this is possible by computer vision. For recognition of the object in real time the system actually needs to capture the time-burst when the video is been taken. Now from around 30-50 images the system compares them and processes them to extract the required object (here, Sudoku puzzle grid) from them. After the detection of puzzle grid, the characters i.e. the numbers are identified from the grid. These numbers are identified using Optical Character Recognition. OCR makes use of Principle Component Analysis(PCA). This was the role of Computer vision in this project. Now the next task comes up of Artificial intelligence.

I. EXISTING SYSTEM The existing system i.e. the solvation of Sudoku puzzle needs huge mathematical computations as it contains 81 cells, in a 9 by 9 grid, and has 9 zones, each zone being the intersection of 3 rows and 3 columns. Each cell may contain a number from one to nine; each number can only occur once in each zone, row, and column of the grid no digit can appear twice in a unit. This implies that each square must have a different value. At the beginning of the game, many cells begin with numbers in them, and the goal is to fill in the remaining cells. If solved manually there is a wide range of strategies to solve Sudoku puzzles and will need huge efforts to do so and compute the optimal solution of the puzzle. Furthermore, several researches have been made to solve Sudoku problems in a more efficient way like using backtracking algorithm. It has conclusively been shown that solving the puzzle, by using different algorithms, is definitely possible but most developers seek for optimizations techniques such as genetic algorithms, simulated annealing. Different authors have made relative works already. Nelishia Pillaygives a solution for solving Sudoku by combining human intuition and optimization. However, we seek a solution to solve Sudoku puzzle based on human strategies, which uses techniques such as: naked single method, hidden single method etc. J.F. Crook have also discussed about solving Sudoku and presented an algorithm on how to solve the puzzles of differing difficulty with pencil-and-paper algorithm.

Flaws of the current system: 1) If solved manually players needs to check different alternatives and place the numbers in the empty squares by guessing as many options are valid. 2) Needs feeding of random numbers by player and then it is checked whether valid or not for all possible solutions to the puzzle until a valid solution is found which is a time consuming procedure resulting an inefficient solver. 3) As compared to real time it is time consuming and needs human efforts too which is not the case in real time puzzle solver. Example of Sudoku puzzle is been shown below. It shows the initial and the goal state that should be.

Table

Description automatically generated

**4. Experiments & Results**

**1-The Backtracking algorithm**

The backtracking algorithm is a memory efficient brute force algorithm, making it a suitable implementation for embedded devices with limited memory. The concept of the algorithm is the following: Start from any cell and do the following steps recursively:

1. If the cell is empty, add a number that is not constrained.

a) If it is impossible to add a number due to constraints, report failure

b) Else start a new thread on a new cell, starting from step 2.

i. If this thread reports a failure, repeat step 2 with a new number (and exhaust the old number)

ii. If this thread reports success, report success, since we can assume that the last cell has been successfully filled.

2. If the cell is filled, then skip this cell and start a new thread on a new cell, starting from step 2. 3. If the algorithm has managed to move beyond the bounds of the board, report success. Before one can begin to implement this algorithm, it is crucial to define the con-straints. Another important factor is how a new cell is chosen. The next sections will discuss these topics.

A backtracking algorithm uses the [depth-first search](https://www.simplilearn.com/tutorials/data-structure-tutorial/dfs-algorithm) method. When the algorithm begins to explore the solutions, the abounding function is applied so that the algorithm can determine whether the proposed solution satisfies the constraints. If it does, it will keep looking. If it does not, the branch is removed, and the algorithm returns to the previous level. Consider the following scenario:

Diagram

Description automatically generated with medium confidence

1. In this case, S represents the problem's starting point. You start at S and work your way to solution S1 via the midway point M1. However, you discovered that solution S1 is not a viable solution to our problem. As a result, you backtrack (return) from S1, return to M1, return to S, and then look for the feasible solution S2. This process is repeated until you arrive at a workable solution.
2. S1 and S2 are not viable options in this case. According to this example, only S3 is a viable solution. When you look at this example, you can see that we go through all possible combinations until you find a viable solution. As a result, you refer to backtracking as a brute-force algorithmic technique.
3. A "space state tree" is the above tree representation of a problem. It represents all possible states of a given problem (solution or non-solution).

The final algorithm is as follows:

* Step 1: Return success if the current point is a viable solution.
* Step 2: Otherwise, if all paths have been exhausted (i.e., the current point is an endpoint), return failure because there is no feasible solution.
* Step 3: If the current point is not an endpoint, backtrack and explore other points, then repeat the preceding steps.

**Flowchart of Backtracking**

**Diagram

Description automatically generated**

**2-** **The** **Differential Evolution Algorithm**

The Differential evolution is a stochastic technique which was developed by K. Price and R. Storn in 1995 [17]. It is a population–based optimization method which can be used for example to numerical optimization [19], neural network training [9], filter design [18] or image analysis [5]. DE is conceptually like the evolutionary algorithm, but there are also quite big differences. First, the mutation is the major genetic operator. It is not a trivial process, and it also provides the algorithm’s convergence. Moreover, the mutation is performed before the crossover process. The pseudocode of the general DE algorithm:

1. Create the initial population of genotypes P0 = {X1,0,X2,0,...,Xn,0},

2. Set the generation number g = 0

3. Until the stop criterion is not met:

(a) Compute the fitness function for every genotype in the population { f(X1,g), f(X2,g),..., f(Xn,g)}

(b) Create the population of trial genotypes Vg based on Pg

(c) Make crossover of genotypes from the population Pg and Vg to create population Ug

(d) Choose the genotypes with the highest fitness function from the population Ug and Pg for the next population

(e) generation = generation+1, go to step a.

The DE algorithm begins with the initialization of the population P(0) which consist of nX individuals. Mutation is a process that adds randomly-generated values to the selected genes. Each increment moves selected individuals towards the global optimum. Note that the mutation is used to every individual in the population. Trial individual ui(t) is created as follows:

ui(t) = xi1(t) +F . ( xi2(t) − xi3(t))

Individual xi(t) is called the target vector. (xi2 (t)−xi3 (t)) is a differential vector created from the two random individuals xi2 (t) and xi3 (t).

Crossover uses both the genotype from the population Pg and the trial genotype (population Vg). This operator will be described deeply in next section. After the crossover process the offspring is compared with its parent. Next, the better one of these individuals is added to the new population. The last step of the algorithm is the increment of the generation counter t. The best individual from the last generation is the result of the DE algorithm.

Then we compare target vector to trial vector & which has best fitness score is added to the new population .

**Flowchart of Differential Evolution**

**Diagram

Description automatically generated**

**5.Development platform**

**Tools: Visual studio code (or pycharm).**

**Programming Languages : Python .**

**Python Libraries : : PyGame.**