Sudoku solver using backtracking algorithm AND Differential Evolution

OVERVIEW

Firstly, let’s discuss what is SUDOKU game briefly.
Sudoku is a logic-based, combinatorial number-placement puzzle. The objective is to fill a 9×9 grid with digits so that each column, each row, and each of the nine 3×3 sub-grids that compose the grid (also called "boxes", "blocks", or "regions") contain all the digits from 1 to 9. The puzzle setter provides a partially completed grid, which for a well-posed puzzle has a single solution.
For example, the (left) figure demonstrates a typical Sudoku puzzle, and its solution (right).

This project deals with differences between two algorithms used for solving Sudoku games. The algorithms used are Differential evolution and Backtracking algorithm.
The goal of this project is to solve the game by the mentioned algorithms and then show the differences between those two algorithms.

PROJECT IDEA

Search algorithms can be compared with different parameters: speed, memory consumption, ability to work with heuristics and others. Comparison between two search algorithms will be shown.
For solving a Sudoku problem, we have investigated two search algorithms: **Differential evolution** and **Backtracking** algorithm.

SIMILAR APPLICATIONS

1- GIVE A TRY! (Sudoku online)

Features:

1. Solve the whole sudoku
2. Solve selected cell
3. Solving different sizes of the sudoku grid from 6*6 up to 25*25
4. Solving different types of sudoku i.e., irregular sudoku, hyper sudoku, extra region sudoku, and odd-even sudoku, etc.
5. Reset the whole state of sudoku board.

2- GIVE A TRY! (Classic Sudoku)

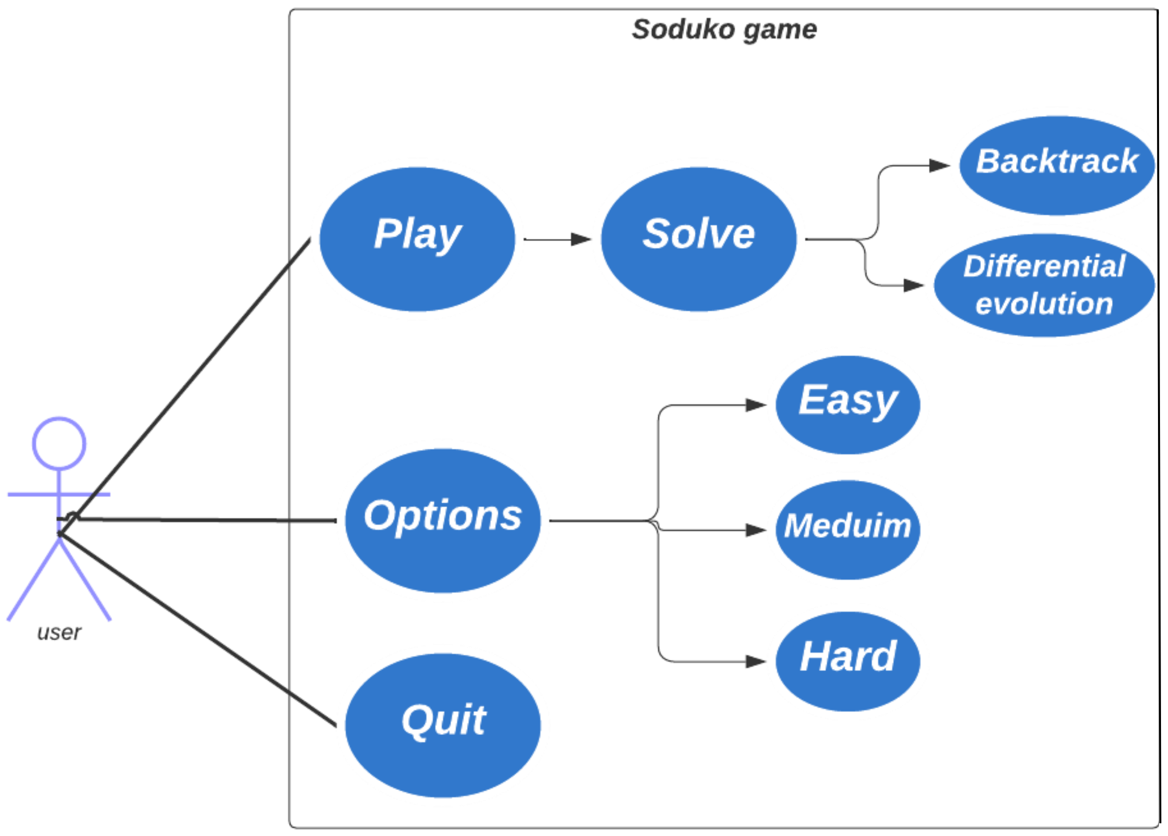
Game review:

Sudoku is also available in google play and Microsoft store for people to install and play with many different levels: I think it is implemented by backtracking because when I tried to play it, it counts the number I place in the grid as wrong number even if it satisfies all conditions but it's not in the ideal place, the program runs the grid to the end to see if it is ideal position or not and counts the move as a mistake if it's not or when it needs to backtrack

LITERATURE REVIEWS OF ACADEMIC PUBLICATIONS

- https://www.ijsrp.org/research-paper-0513/ijsrp-p1735.pdf
- https://sci-hub.se/10.1109/wict.2013.7113148
- https://sci-hub.se/https://ieeexplore.ieee.org/abstract/document/9234267
- https://sci-hub.se/10.1109/iadcc.2014.6779291
- https://sci-hub.se/10.1109/lisat.2014.6845190

MAIN FUNCTIONALITIES



APPLIED ALGORITHMS

Backtracking algorithm description.

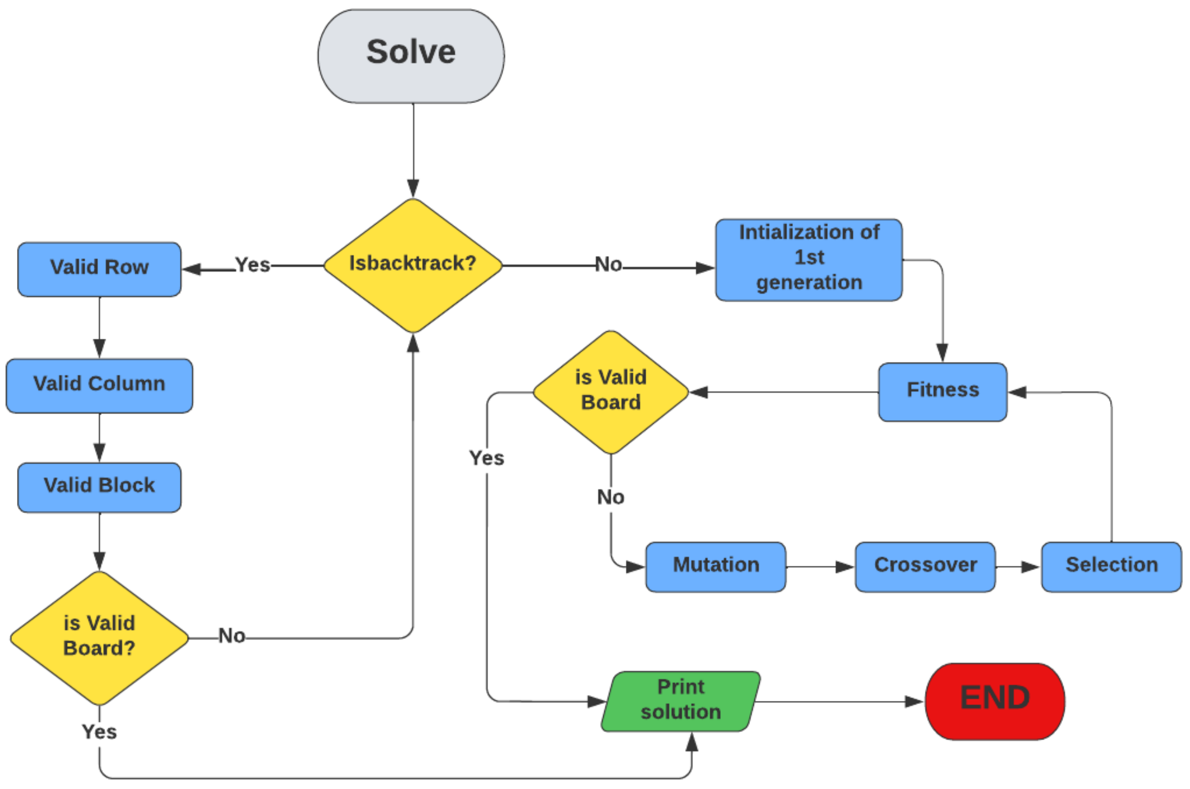
• Backtracking Aim: find the solution of Sudoku take index of column initialize zero, index of row initializes zero and the board.
How to work: make iteration from 1 to 9 and but the numbers in the empty cell then call function CheckValidCell, CheckValidRow, CheckValidColumn and CheckValidBlock (3x3) to check if valid number or not then call itself again by column+1 (recurse) if column reaches 9 (end of columns) or end row return True else if it is end of column increase row by one then make column = 0
• *Return: return true if you find answer else return Null

Differential evolution algorithm description

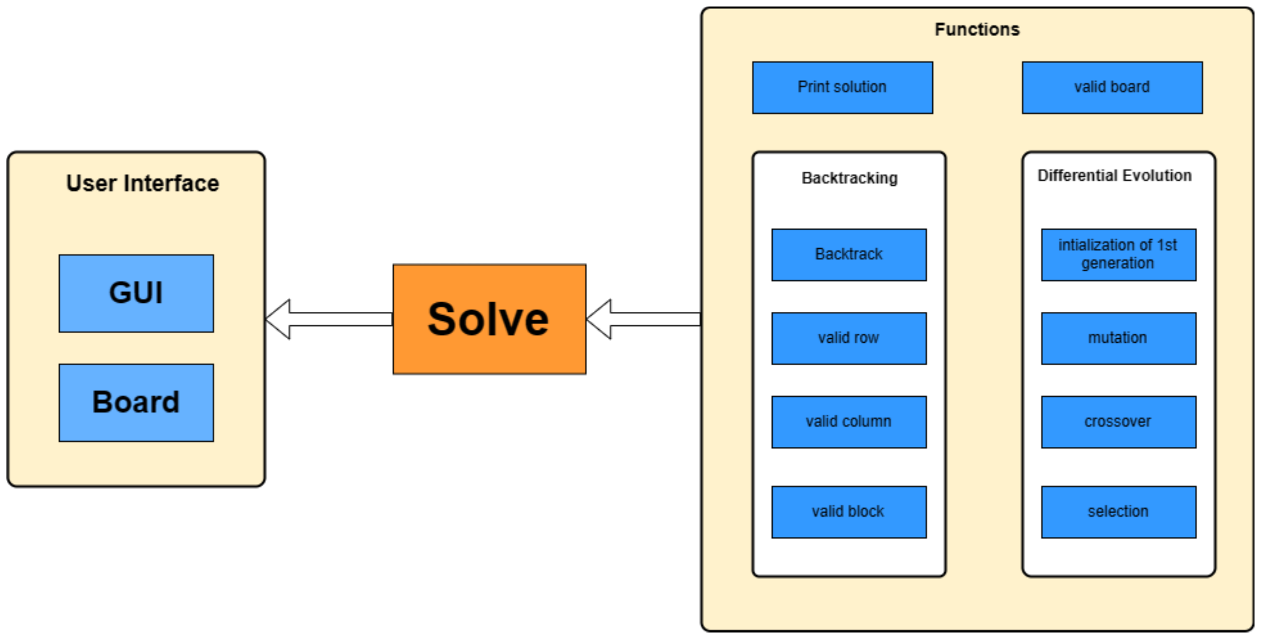
• "DE" Aim: Population need to evolve to advance the next generation
*How to work: For each individual in population do that:
1) Select three unique parents from population and work on them mutation then it returns new individual(female)
2) Make crossover by CrossOverRate and generate new child caused by mixing the parent(male) and individual(female) generated by mutate function.
3) Selection: if fitness of the child better than the fitness of parent then replace the parent with the child.

4) If fitness function for individual equal zero (the optimal solution) then sort the population by fitness function and return it
*Take: 1-population 2-original board 3-mutateRate 4- CrossOverRate
*Return: population after advance the next generation

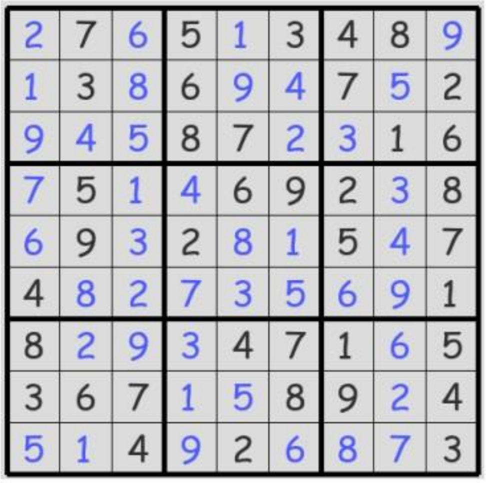
Solution algorithm flow chart



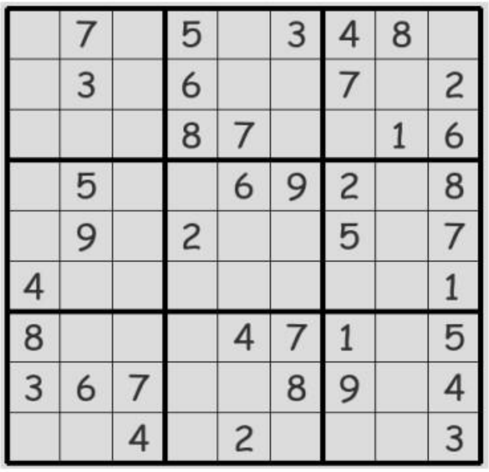
Solve block diagram


User Interface
GUI
Board
Solve
Functions
Print solution
valid board
Backtracking
Backtrack
valid row
valid column
valid block
Differential Evolution
intialization of 1st generation
mutation
crossover
selection

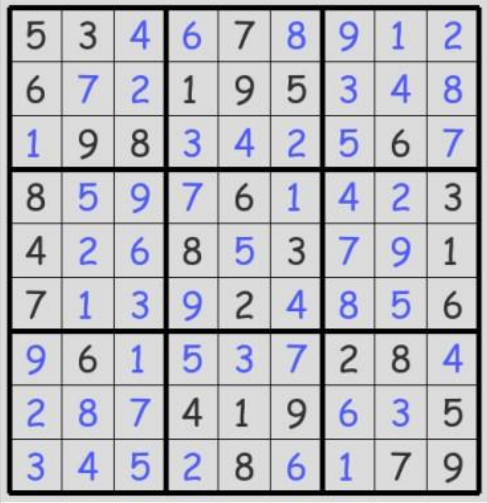
SAMPLES OF INPUT AND OUTPUT



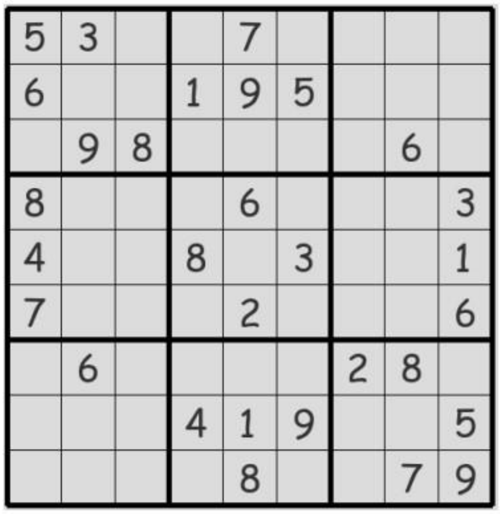
2	7	6	5	1	3	4	8	9
1	3	8	6	9	4	7	5	2
9	4	5	8	7	2	3	1	6
7	5	1	4	6	9	2	3	8
6	9	3	2	8	1	5	4	7
4	8	2	7	3	5	6	9	1
8	2	9	3	4	7	1	6	5
3	6	7	1	5	8	9	2	4
5	1	4	9	2	6	8	7	3



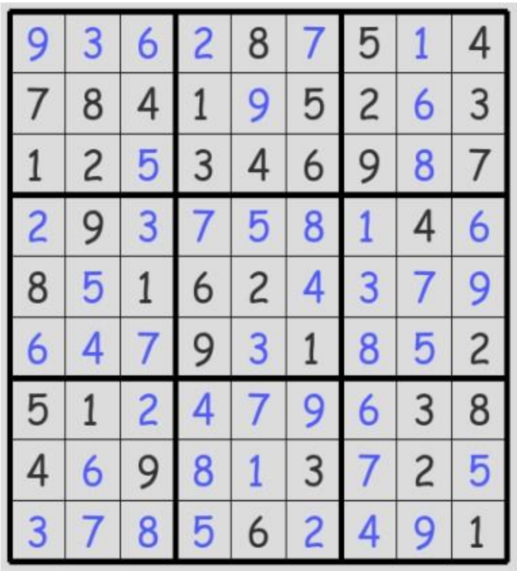
	7		5		3	4	8	
	3		6			7		2
			8	7			1	6
	5			6	9	2		8
	9		2			5		7
4								1
8				4	7	1		5
3	6	7			8	9		4
		4		2				3



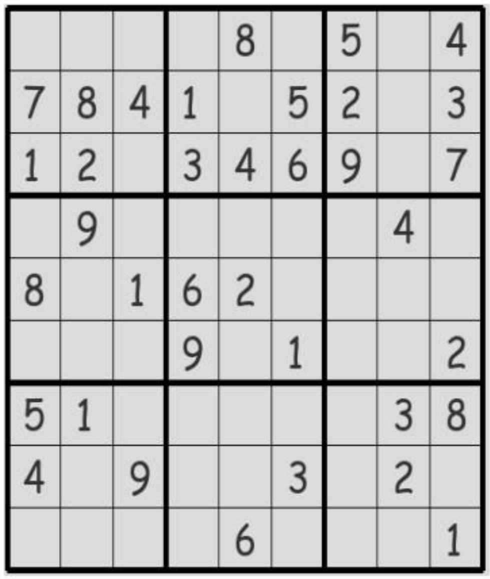
5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9



5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

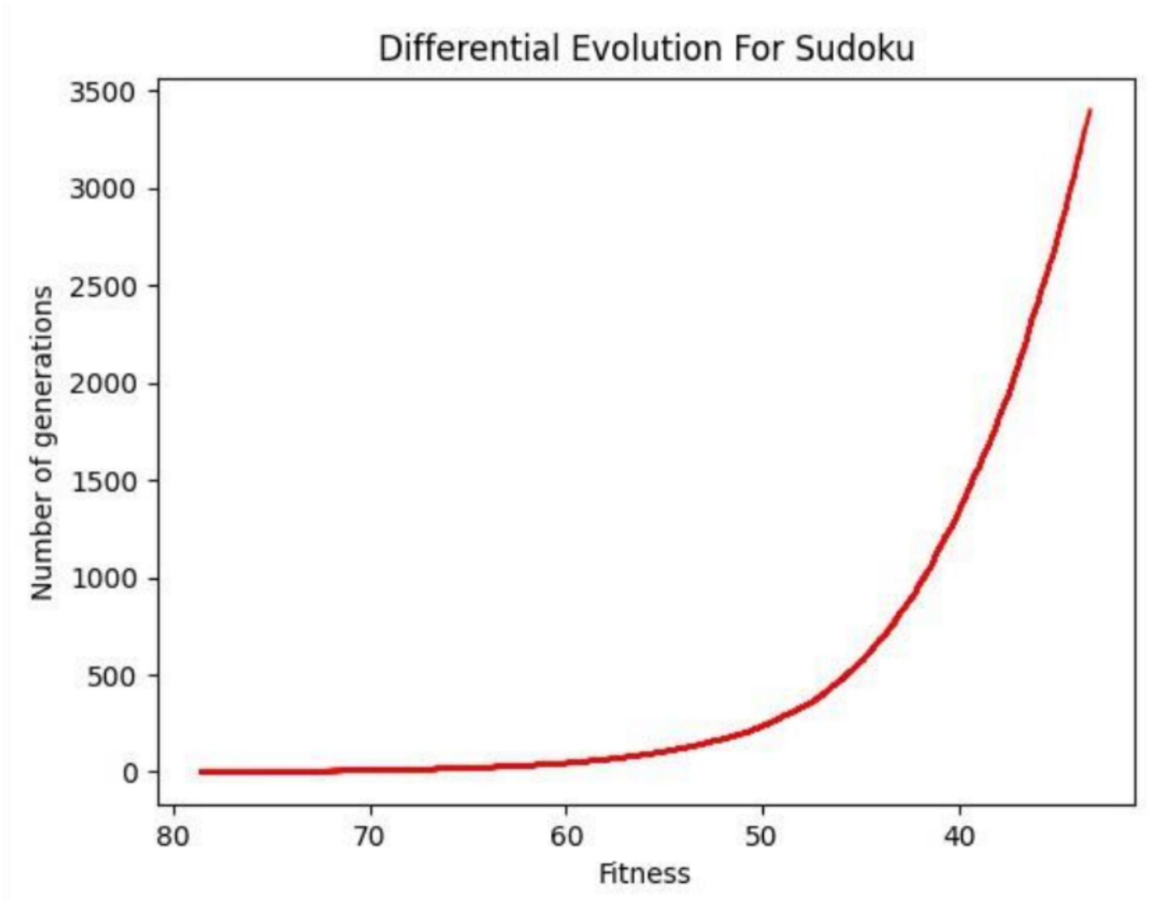


9	3	6	2	8	7	5	1	4
7	8	4	1	9	5	2	6	3
1	2	5	3	4	6	9	8	7
2	9	3	7	5	8	1	4	6
8	5	1	6	2	4	3	7	9
6	4	7	9	3	1	8	5	2
5	1	2	4	7	9	6	3	8
4	6	9	8	1	3	7	2	5
3	7	8	5	6	2	4	9	1



				8		5		4
7	8	4	1		5	2		3
1	2		3	4	6	9		7
	9						4	
8		1	6	2				
			9		1			2
5	1						3	8
4		9			3		2	
				6				1

EVOLUTION PLOT



ANALYSIS, DISCUSSION, AND FUTURE WORK

• **Analysis of the results, what are the insights?**
The graph increases by decreasing the grade of the populations.

• **What are the advantages?**
When grade of the populations decreases, Then the evolution is better, so we are closer to the target

• **What are the disadvantages?**
When grade of the populations is stopped decrease or slow in decreasing,
Then the evolution is not growing enough to the target

• **Why did the algorithm behave in such a way?**
The Differential Evolution is finding the global max, where every member is in local max. In initialization, it randomly creates the number of boards as population, and we try to make mutation, crossover, and selection to evolve the population. As it takes time to reach the global max

• **What might be the future modifications?**
By tuning hyperparameters, Like Mutant Factor, Crossover Rate, population quantity, optimization for fitness function.

SHARED PROJECT (INCLUDING IMPLEMENTATION)

Click here for our GitHub repository.