Sudoku solver using Genetic Algorithm and Backtracking

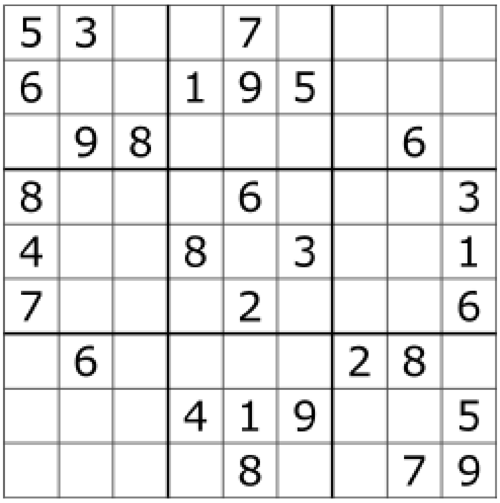
Sudarshan Rao Mundkur
Computer Science Department, Purdue University
Fort Wayne, IN, 46805
munds01@pfw.edu

Abstract. Sudoku is a number placement puzzle played all over the world and is a simple, yet interesting game. The objective of this project is to develop an optimal solution to the game. This project is a python implementation using Genetic Algorithm approach and Back-tracking approach. Objective is to compare both algorithms and choose a better fit for the problem.

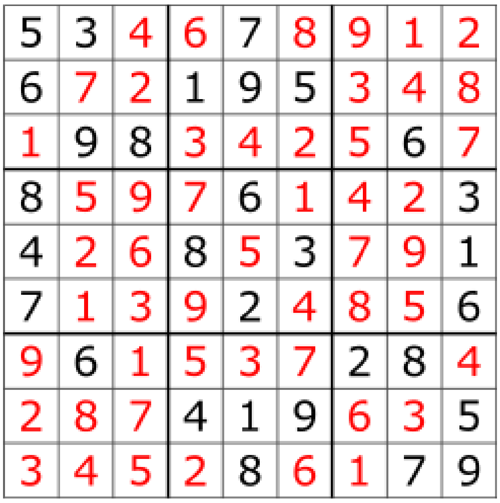
Keywords: Sudoku, Genetic Algorithm, Backtracking, Python, Optimal Solution.

1 Introduction

Sudoku [1] is a number placement puzzle that has achieved remarkable popularity in the past few years. In its classic form, the objective is to fill a 9x9 grid with the digits 1 to 9, subject to the following constraints: each row, each column, and each of the nine 3x3 sub grids must contain a permutation of the digits from 1 to 9. The initial puzzle configuration provides a partially completed grid, such as the example grid illustrated below left. The Sudoku solver must fill in the remaining grid elements to obtain a solution like the one given on the right.



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



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

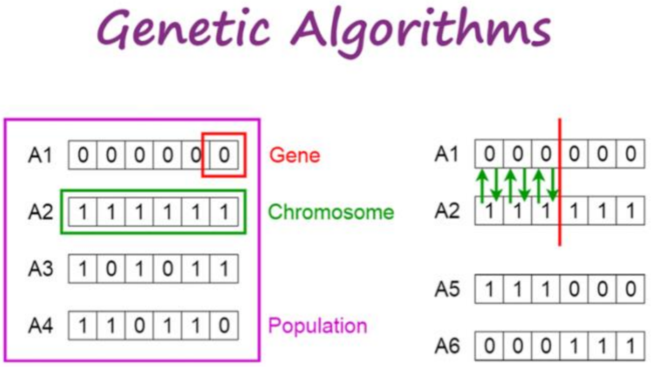
Sudoku is a puzzle based on a small number of very simple rules:

- Every square must contain a single number.
- Only the numbers from 1 through to 9 can be used.
- Each 3×3 box can only contain each number from 1 to 9 once.
- Each vertical column can only contain each number from 1 to 9 once.
- Each horizontal row can only contain each number from 1 to 9 once.

Genetic algorithm [2] is a search heuristic that is inspired by Charles Darwin's theory of natural evolution. This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation. It starts with the selection of fittest individuals from a population. They produce offspring which inherit the characteristics of the parents and will be added to the next generation. If parents have better fitness, their offspring will be better than parents and have a better chance at surviving. This process keeps on iterating and at the end, a generation with the fittest individuals will be found.

Five phases are considered in a genetic algorithm.

1. Initial population
2. Fitness function
3. Selection
4. Crossover
5. Mutation



2 Problem Statement

"To implement Genetic algorithm and Backtracking algorithm to solve Sudoku Puzzle."

1. Methodology

Genetic Algorithm:

Genetic algorithms are commonly used to generate high-quality solutions to optimization and search problems by relying on biologically inspired operators such as mutation, crossover and selection.

Genetic Algorithm Design to solve Sudoku:

Step 1: Initialization. Create an initial population of Sudoku boards using the helper array. The helper array contains a set of possible values for each position in the Sudoku board. Each board in the population is created using the helper array and made sure that all the rows in each board are unique.

Step 2: Calculate the fitness of each individual and rank the Population.

We made sure that all the numbers in each row of a Sudoku board are unique while creating the population, so fitness calculation for rows can be ignored.

How the fitness value is calculated?

Fitness function counts the number of unique values of numbers 1 through 9 in each column, each of these unique values in the subsets are summed and divided by 9. The sum of these values is divided by 9 resulting in a floating value between 0 and 1.

The same process is repeated for each individual 3*3 sub grids in the board.

The product of these floating-point values provides a total fitness value ranging between 0 and 1.

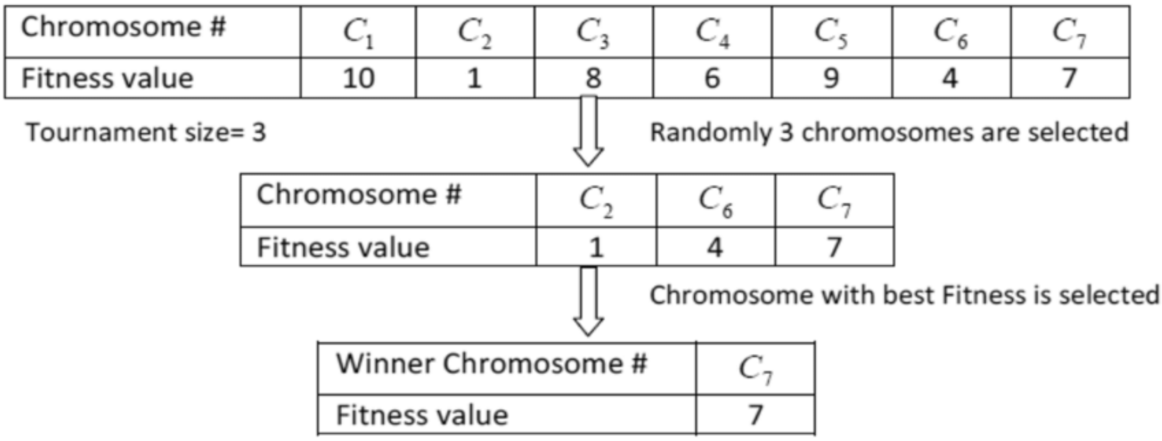
Fitness value equals to 1 indicates that the puzzle is solved.

Step 3: Get the Best fitness, if fitness value is equal to 1, then Sudoku Solution is found. Exit code.

Step 4: Select the Elite candidates from population based on fitness values. The Elitism rate used in implementation is 5%. That is 5% of 1000 = 50 best candidates from the population are selected for next generation.

Step 5: Select random parents from the non-elite candidates list to perform crossover. The selection is done using Tournament selection method.

Tournament selection involves running several "tournaments" among a few individuals (or "chromosomes") chosen at random from the population. The winner of each tournament (the one with the best fitness) is selected for crossover.



Chromosome #	C_1	C_2	C_3	C_4	C_5	C_6	C_7
Fitness value	10	1	8	6	9	4	7

Tournament size= 3

Randomly 3 chromosomes are selected

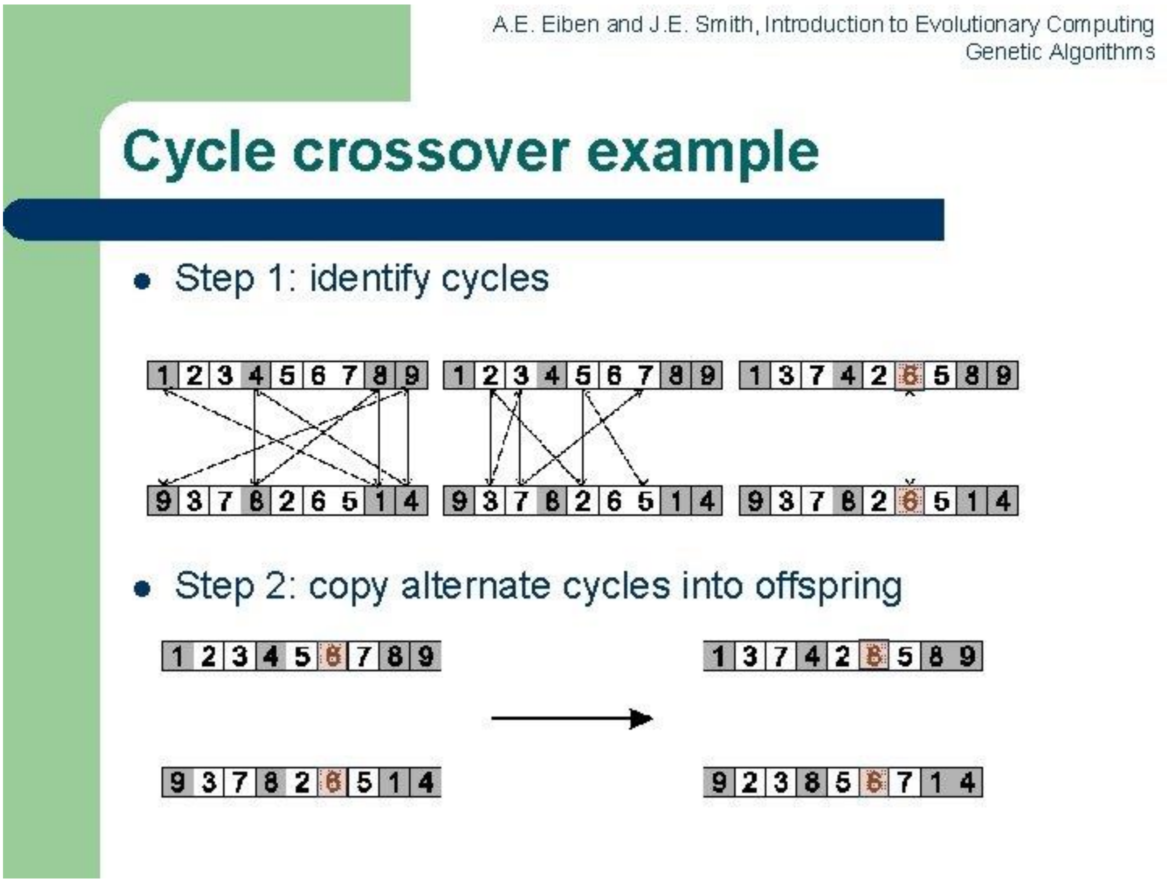
Chromosome #	C_2	C_6	C_7
Fitness value	1	4	7

Chromosome with best Fitness is selected

Winner Chromosome #	C_7
Fitness value	7

Step 6: Perform Crossover over the selected parents to create children for the next generation. The crossover method used is Cycle Crossover.

Cycle Crossover operator identifies cycles between two parent chromosomes. Then, to form the children, all the odd cycles values are swapped, and the even cycle values are kept as it is.



A.E. Eiben and J.E. Smith, Introduction to Evolutionary Computing
Genetic Algorithms

Cycle crossover example

- Step 1: identify cycles

- Step 2: copy alternate cycles into offspring

Step 7: Mutate the children and add it to the next population list.

The mutation method used is swap mutation.

Select two random values from the same corresponding rows of two candidate boards selected at random.

Check if the selected values are fixed elements such as the question points. If yes, change the selection and repeat until two points are selected which are not question points.

Check if swapping the values results in violation of the column rule or box rule in each of the candidate boards.

If yes, change the selection and repeat until a successful swap is done.

Step 8: Append the elite candidates to the next population list and pass it on as next generation.

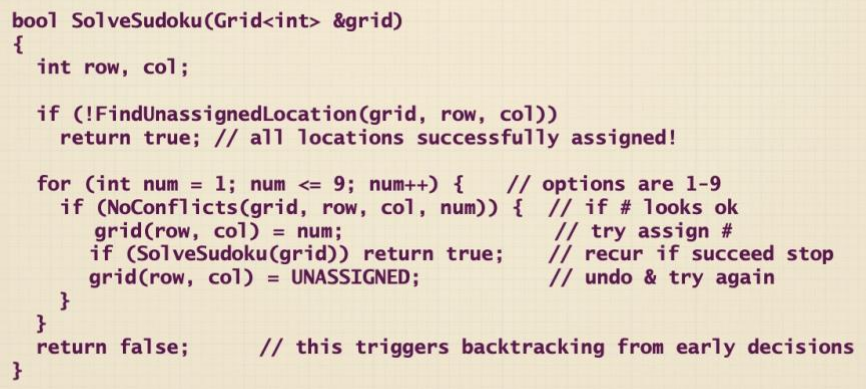
Step 9: Repeat steps 2 to 7 until Fitness equal to 1 or Number of generations reach's 10000.

Parameters used in implementation:

- Number of candidates (Population Size): 1000
- Elitism rate: 5%
- Mutation rate: 0.06
- Number of Generations: 10000

Back Tracking Algorithm:

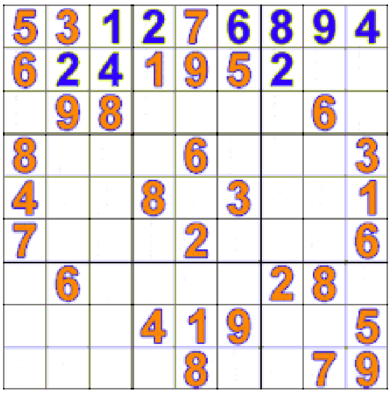
Sudoku [3] can be solved by one-by-one assigning numbers to empty cells. Before assigning a number, check whether it is safe to assign. Check that the same number is not present in the current row, current column and current 3X3 sub-grid. After checking for safety, assign the number, and recursively check whether this assignment leads to a solution or not. If the assignment doesn't lead to a solution, then try the next number for the current empty cell. And if none of the number (1 to 9) leads to a solution, return false and print no solution exists.



```
bool SolveSudoku(Grid<int> &grid)
{
  int row, col;

  if (!FindUnassignedLocation(grid, row, col))
    return true; // all locations successfully assigned!

  for (int num = 1; num <= 9; num++) {    // options are 1-9
    if (NoConflicts(grid, row, col, num)) {  // if # looks ok
       grid(row, col) = num;                 // try assign #
       if (SolveSudoku(grid)) return true;   // recur if succeed stop
       grid(row, col) = UNASSIGNED;          // undo & try again
    }
  }
  return false;      // this triggers backtracking from early decisions
}
```

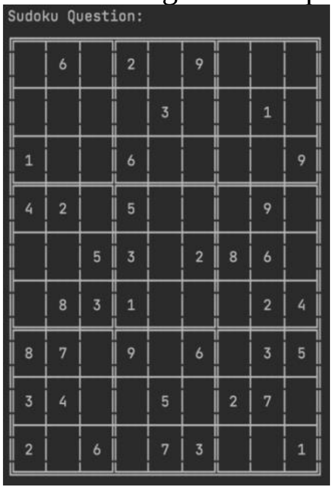


5	3	1	2	7	6	8	9	4
6	2	4	1	9	5	2		
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

4. Result and Comparison:

On executing the python file, the result obtained has been shown below. There are 4 levels of Sudoku considered for this experiment.

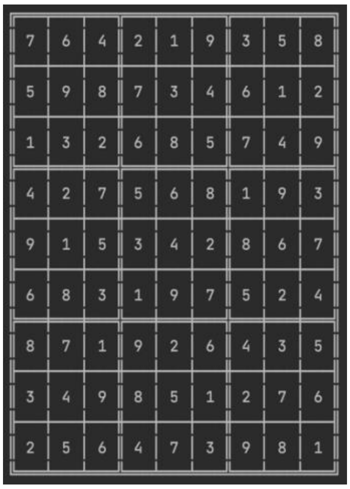
1. Easy Sudoku Puzzle with Number of givens is equal to 38.



Sudoku Question:

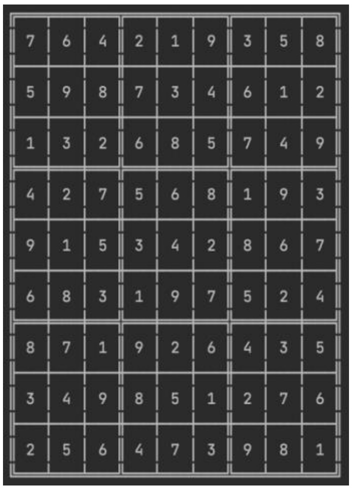
	6		2		9			
				3			1	
1			6					9
4	2		5				9	
		5	3		2	8	6	
	8	3	1				2	4
8	7		9		6		3	5
3	4			5		2	7	
2		6		7	3			1

```
Execution Started: Genetic Algorithm
Generation: 0  Best fitness: 0.9259259259259299
Generation: 1  Best fitness: 0.8802011888431678
Generation: 2  Best fitness: 0.9259259259259299
Generation: 3  Best fitness: 0.9036732205456524
Generation: 4  Best fitness: 0.9036732205456524
Generation: 5  Best fitness: 0.9506172839506214
Generation: 6  Best fitness: 0.9506172839506214
Generation: 7  Best fitness: 0.9506172839506214
Genetic Algorithm Solution found at generation: 8
Time elapsed:   6.73s
```



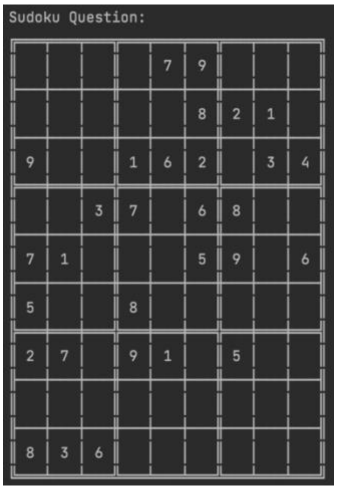
```
7 6 4 2 1 9 3 5 8
5 9 8 7 3 4 6 1 2
1 3 2 6 8 5 7 4 9
4 2 7 5 6 8 1 9 3
9 1 5 3 4 2 8 6 7
6 8 3 1 9 7 5 2 4
8 7 1 9 2 6 4 3 5
3 4 9 8 5 1 2 7 6
2 5 6 4 7 3 9 8 1
```

Execution Started: Backtracking
Backtracking Solution found:
Time elapsed: 0.00s



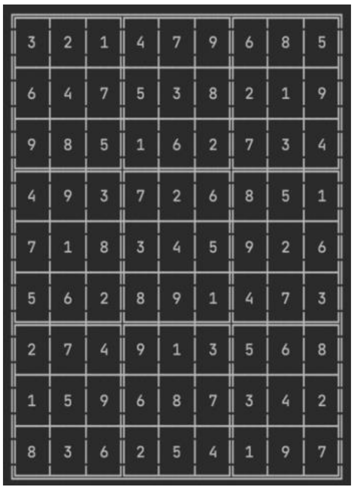
```
7 6 4 2 1 9 3 5 8
5 9 8 7 3 4 6 1 2
1 3 2 6 8 5 7 4 9
4 2 7 5 6 8 1 9 3
9 1 5 3 4 2 8 6 7
6 8 3 1 9 7 5 2 4
8 7 1 9 2 6 4 3 5
3 4 9 8 5 1 2 7 6
2 5 6 4 7 3 9 8 1
```

2. Medium Sudoku Puzzle with Number of givens is equal to 30.



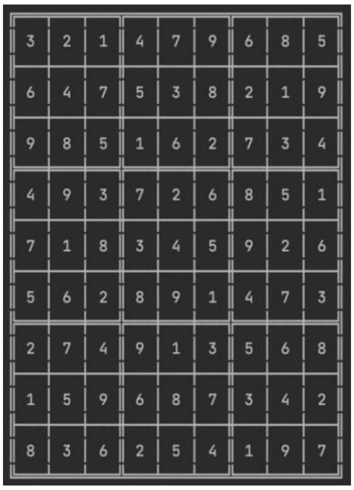
```
Sudoku Question:
        7 9      
          8 2 1  
9     1 6 2   3 4
    3 7   6 8    
7 1       5 9   6
5     8          
2 7   9 1   5    
                 
8 3 6            
```

```
Execution Started: Genetic Algorithm
Generation: 0  Best fitness: 0.8332571254381987
Generation: 1  Best fitness: 0.7511050144795028
Generation: 2  Best fitness: 0.7863130620332297
Generation: 3  Best fitness: 0.7628410303307451
Generation: 4  Best fitness: 0.7863130620332297
Generation: 5  Best fitness: 0.8332571254381987
Generation: 6  Best fitness: 0.8802011888431678
Generation: 7  Best fitness: 0.8567291571406833
Generation: 8  Best fitness: 0.8332571254381987
Generation: 9  Best fitness: 0.8332571254381987
Generation: 10  Best fitness: 0.8332571254381987
Generation: 11  Best fitness: 0.8567291571406833
Generation: 12  Best fitness: 0.8518518518518553
Generation: 13  Best fitness: 0.8518518518518553
Generation: 14  Best fitness: 0.8567291571406833
Generation: 15  Best fitness: 0.9012345679012383
Generation: 16  Best fitness: 0.9012345679012383
Generation: 17  Best fitness: 0.8567291571406833
Generation: 18  Best fitness: 0.8518518518518553
Generation: 19  Best fitness: 0.9012345679012383
Generation: 20  Best fitness: 0.9012345679012383
Generation: 21  Best fitness: 0.9012345679012383
Generation: 22  Best fitness: 0.9012345679012383
Generation: 23  Best fitness: 0.9012345679012383
Generation: 24  Best fitness: 0.9012345679012383
Generation: 25  Best fitness: 0.9012345679012383
Generation: 26  Best fitness: 0.9012345679012383
Generation: 27  Best fitness: 0.9012345679012383
Generation: 28  Best fitness: 0.9036732205456524
Generation: 29  Best fitness: 0.9506172839506214
Generation: 30  Best fitness: 0.9036732205456524
Genetic Algorithm Solution found at generation: 31
Time elapsed:  31.69s
```



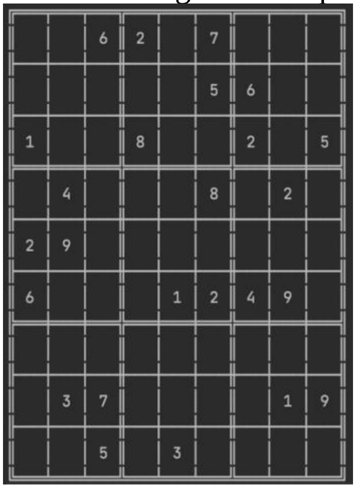
3	2	1	4	7	9	6	8	5
6	4	7	5	3	8	2	1	9
9	8	5	1	6	2	7	3	4
4	9	3	7	2	6	8	5	1
7	1	8	3	4	5	9	2	6
5	6	2	8	9	1	4	7	3
2	7	4	9	1	3	5	6	8
1	5	9	6	8	7	3	4	2
8	3	6	2	5	4	1	9	7

```
Execution Started: Backtracking
Backtracking Solution found:
Time elapsed:   0.00s
```



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

3. Hard Sudoku Puzzle with Number of givens is equal to 25.

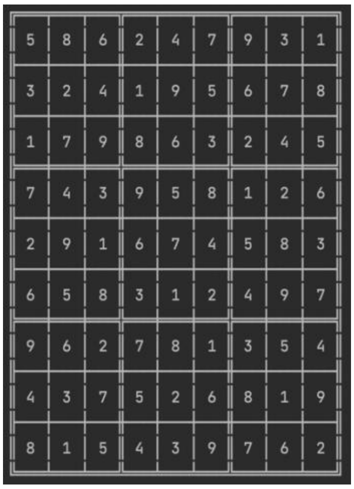


		6	2		7			
					5	6		
1			8			2		5
	4				8		2	
2	9							
6				1	2	4	9	
	3	7					1	9
		5		3				

```
Execution Started: Genetic Algorithm
Generation: 0  Best fitness: 0.5820759030635592
Generation: 1  Best fitness: 0.5820759030635592
Generation: 2  Best fitness: 0.5820759030635592
Generation: 3  Best fitness: 0.5857338820301802
Generation: 4  Best fitness: 0.5820759030635592
Generation: 5  Best fitness: 0.5922877610120427
Generation: 6  Best fitness: 0.6127114769090097
Generation: 7  Best fitness: 0.6241426611797003
Generation: 8  Best fitness: 0.634049687547632
Generation: 9  Best fitness: 0.6310013717421146
Generation: 10  Best fitness: 0.6415180612711499
Generation: 11  Best fitness: 0.6538637402834956
Generation: 12  Best fitness: 0.6342021033379078
Generation: 13  Best fitness: 0.6415180612711499
Generation: 14  Best fitness: 0.6817558299039804
```

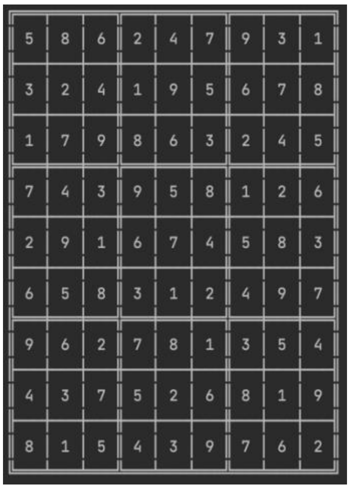
```
Generation: 15  Best fitness: 0.6425849718030809
Generation: 16  Best fitness: 0.6925773510135674
Generation: 17  Best fitness: 0.6835848193872909
Generation: 18  Best fitness: 0.6817558299039804
Generation: 19  Best fitness: 0.6898338667886017
Generation: 20  Best fitness: 0.7201646090535005
Generation: 21  Best fitness: 0.723212924859018
Generation: 22  Best fitness: 0.7009602194787405
Generation: 23  Best fitness: 0.7009602194787405
Generation: 24  Best fitness: 0.6898338667886017
Generation: 25  Best fitness: 0.7575064776710896
Generation: 26  Best fitness: 0.7343392775491568
Generation: 27  Best fitness: 0.723212924859018
Generation: 28  Best fitness: 0.7454656302392956
Generation: 29  Best fitness: 0.7565919829294343
Generation: 30  Best fitness: 0.8097850937357142
Generation: 31  Best fitness: 0.8567291571406833
Generation: 32  Best fitness: 0.8097850937357142
Generation: 33  Best fitness: 0.7863130620332297
Generation: 34  Best fitness: 0.8097850937357142
Generation: 35  Best fitness: 0.8122237463801282
Generation: 36  Best fitness: 0.9012345679012383
Generation: 37  Best fitness: 0.9012345679012383
Generation: 38  Best fitness: 0.9012345679012383
Generation: 39  Best fitness: 0.9506172839506214
Generation: 40  Best fitness: 0.864197530864201
Generation: 41  Best fitness: 0.8215211095869565
Generation: 42  Best fitness: 0.8567291571406833
Generation: 43  Best fitness: 0.8567291571406833
Generation: 44  Best fitness: 0.8567291571406833
Generation: 45  Best fitness: 0.8802011888431678
Generation: 46  Best fitness: 0.8332571254381987
Generation: 47  Best fitness: 0.8518518518518553
Generation: 48  Best fitness: 0.8802011888431678
Generation: 49  Best fitness: 0.8567291571406833
Generation: 50  Best fitness: 0.8802011888431678
Generation: 51  Best fitness: 0.8567291571406833
Generation: 52  Best fitness: 0.8567291571406833
Generation: 53  Best fitness: 0.844993141289441
Generation: 54  Best fitness: 0.844993141289441
Generation: 55  Best fitness: 0.9012345679012383
Generation: 56  Best fitness: 0.9506172839506214
Generation: 57  Best fitness: 0.8567291571406833
Generation: 58  Best fitness: 0.9012345679012383
Generation: 59  Best fitness: 0.8802011888431678
Generation: 60  Best fitness: 0.9506172839506214
Generation: 61  Best fitness: 0.9506172839506214
Generation: 62  Best fitness: 0.9506172839506214
```

```
Generation: 63  Best fitness: 0.8573388203017868
Generation: 64  Best fitness: 0.8567291571406833
Generation: 65  Best fitness: 0.9259259259259299
Generation: 66  Best fitness: 0.9259259259259299
Generation: 67  Best fitness: 0.8802011888431678
Generation: 68  Best fitness: 0.9036732205456524
Generation: 69  Best fitness: 0.9012345679012383
Generation: 70  Best fitness: 0.9012345679012383
Generation: 71  Best fitness: 0.9259259259259299
Generation: 72  Best fitness: 0.9259259259259299
Generation: 73  Best fitness: 0.9506172839506214
Generation: 74  Best fitness: 0.9506172839506214
Generation: 75  Best fitness: 0.9506172839506214
Generation: 76  Best fitness: 0.9135802469135841
Generation: 77  Best fitness: 0.9506172839506214
Generation: 78  Best fitness: 0.9506172839506214
Generation: 79  Best fitness: 0.9506172839506214
Generation: 80  Best fitness: 0.9506172839506214
Generation: 81  Best fitness: 0.9259259259259299
Generation: 82  Best fitness: 0.9506172839506214
Genetic Algorithm Solution found at generation: 82
Time elapsed: 72.90s
```



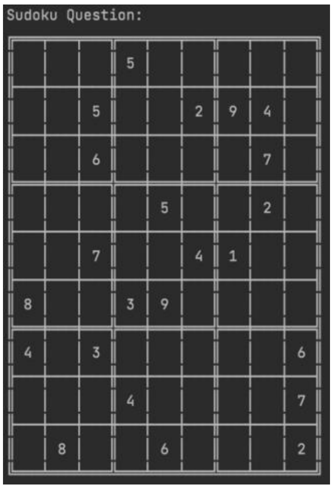
5	8	6	2	4	7	9	3	1
3	2	4	1	9	5	6	7	8
1	7	9	8	6	3	2	4	5
7	4	3	9	5	8	1	2	6
2	9	1	6	7	4	5	8	3
6	5	8	3	1	2	4	9	7
9	6	2	7	8	1	3	5	4
4	3	7	5	2	6	8	1	9
8	1	5	4	3	9	7	6	2

```
Execution Started: Backtracking
Backtracking Solution found:
Time elapsed:   0.02s
```



```
║ 5 │ 8 │ 6 ║ 2 │ 4 │ 7 ║ 9 │ 3 │ 1 ║
║ 3 │ 2 │ 4 ║ 1 │ 9 │ 5 ║ 6 │ 7 │ 8 ║
║ 1 │ 7 │ 9 ║ 8 │ 6 │ 3 ║ 2 │ 4 │ 5 ║
║ 7 │ 4 │ 3 ║ 9 │ 5 │ 8 ║ 1 │ 2 │ 6 ║
║ 2 │ 9 │ 1 ║ 6 │ 7 │ 4 ║ 5 │ 8 │ 3 ║
║ 6 │ 5 │ 8 ║ 3 │ 1 │ 2 ║ 4 │ 9 │ 7 ║
║ 9 │ 6 │ 2 ║ 7 │ 8 │ 1 ║ 3 │ 5 │ 4 ║
║ 4 │ 3 │ 7 ║ 5 │ 2 │ 6 ║ 8 │ 1 │ 9 ║
║ 8 │ 1 │ 5 ║ 4 │ 3 │ 9 ║ 7 │ 6 │ 2 ║
```

4. Expert Sudoku Puzzle with Number of givens is equal to 23.



```
Sudoku Question:
║   │   │   ║ 5 │   │   ║   │   │   ║
║   │   │ 5 ║   │   │ 2 ║ 9 │ 4 │   ║
║   │   │ 6 ║   │   │   ║   │ 7 │   ║
║   │   │   ║   │ 5 │   ║   │ 2 │   ║
║   │   │ 7 ║   │   │ 4 ║ 1 │   │   ║
║ 8 │   │   ║ 3 │ 9 │   ║   │   │   ║
║ 4 │   │ 3 ║   │   │   ║   │   │ 6 ║
║   │   │   ║ 4 │   │   ║   │   │ 7 ║
║   │ 8 │   ║   │ 6 │   ║   │   │ 2 ║
```

```
Execution Started: Genetic Algorithm
Generation: 0  Best fitness: 0.5761316872428001
Generation: 1  Best fitness: 0.5852766346593525
Generation: 2  Best fitness: 0.6127114769090097
Generation: 3  Best fitness: 0.6081390032007336
Generation: 4  Best fitness: 0.6601127876848064
Generation: 5  Best fitness: 0.678707514098463
Generation: 6  Best fitness: 0.6520347508001851
Generation: 7  Best fitness: 0.6415180612711499
Generation: 8  Best fitness: 0.7158969669257761
Generation: 9  Best fitness: 0.6835848193872909
Generation: 10  Best fitness: 0.6492912665752194
Generation: 11  Best fitness: 0.6492912665752194
Generation: 12  Best fitness: 0.6675811614083241
Generation: 13  Best fitness: 0.6675811614083241
Generation: 14  Best fitness: 0.7009602194787405
Generation: 15  Best fitness: 0.6492912665752194
Generation: 16  Best fitness: 0.7276329827770184
Generation: 17  Best fitness: 0.7628410303307451
Generation: 18  Best fitness: 0.7530864197530892
```

```
Generation: 19  Best fitness: 0.6898338667886017
Generation: 20  Best fitness: 0.7046181984453614
Generation: 21  Best fitness: 0.6828227404359115
Generation: 22  Best fitness: 0.7009602194787405
Generation: 23  Best fitness: 0.7046181984453614
Generation: 24  Best fitness: 0.6898338667886017
Generation: 25  Best fitness: 0.7201646090535005
Generation: 26  Best fitness: 0.7041609510745338
Generation: 27  Best fitness: 0.7046181984453614
Generation: 28  Best fitness: 0.678707514098463
Generation: 29  Best fitness: 0.6880048773052913
Generation: 30  Best fitness: 0.723212924859018
Generation: 31  Best fitness: 0.7158969669257761
Generation: 32  Best fitness: 0.7745770461819874
Generation: 33  Best fitness: 0.7745770461819874
Generation: 34  Best fitness: 0.7276329827770184
Generation: 35  Best fitness: 0.7393689986282607
Generation: 36  Best fitness: 0.7677183356195731
Generation: 37  Best fitness: 0.7628410303307451
Generation: 38  Best fitness: 0.7628410303307451
Generation: 39  Best fitness: 0.7628410303307451
Generation: 40  Best fitness: 0.7430269775948816
Generation: 41  Best fitness: 0.7628410303307451
Generation: 42  Best fitness: 0.7511050144795028
Generation: 43  Best fitness: 0.7343392775491568
Generation: 44  Best fitness: 0.7120865721688793
Generation: 45  Best fitness: 0.7628410303307451
Generation: 46  Best fitness: 0.7565919829294343
Generation: 47  Best fitness: 0.7343392775491568
Generation: 48  Best fitness: 0.8024691358024723
Generation: 49  Best fitness: 0.7511050144795028
Generation: 50  Best fitness: 0.8332571254381987
Generation: 51  Best fitness: 0.7788446883097119
Generation: 52  Best fitness: 0.7511050144795028
Generation: 53  Best fitness: 0.8332571254381987
Generation: 54  Best fitness: 0.7628410303307451
Generation: 55  Best fitness: 0.7454656302392956
Generation: 56  Best fitness: 0.7777777777777808
Generation: 57  Best fitness: 0.7511050144795028
Generation: 58  Best fitness: 0.7777777777777808
Generation: 59  Best fitness: 0.7777777777777808
Generation: 60  Best fitness: 0.7777777777777808
Generation: 61  Best fitness: 0.8684651729919255
Generation: 62  Best fitness: 0.7863130620332297
Generation: 63  Best fitness: 0.7863130620332297
Generation: 64  Best fitness: 0.8097850937357142
Generation: 65  Best fitness: 0.8271604938271638
Generation: 66  Best fitness: 0.8271604938271638
```

```
Generation: 67  Best fitness: 0.7901234567901265
Generation: 68  Best fitness: 0.8271604938271638
Generation: 69  Best fitness: 0.8271604938271638
Generation: 70  Best fitness: 0.8097850937357142
Generation: 71  Best fitness: 0.8116140832190247
Generation: 72  Best fitness: 0.8097850937357142
Generation: 73  Best fitness: 0.8271604938271638
Generation: 74  Best fitness: 0.8271604938271638
Generation: 75  Best fitness: 0.8215211095869565
Generation: 76  Best fitness: 0.8332571254381987
Generation: 77  Best fitness: 0.8518518518518553
Generation: 78  Best fitness: 0.8271604938271638
Generation: 79  Best fitness: 0.8024691358024723
Generation: 80  Best fitness: 0.798049077884472
Generation: 81  Best fitness: 0.8518518518518553
Generation: 82  Best fitness: 0.8116140832190247
Generation: 83  Best fitness: 0.8271604938271638
Generation: 84  Best fitness: 0.8332571254381987
Generation: 85  Best fitness: 0.8332571254381987
Generation: 86  Best fitness: 0.844993141289441
Generation: 87  Best fitness: 0.8097850937357142
Generation: 88  Best fitness: 0.8271604938271638
Generation: 89  Best fitness: 0.8097850937357142
Generation: 90  Best fitness: 0.8271604938271638
Generation: 91  Best fitness: 0.8567291571406833
Generation: 92  Best fitness: 0.8567291571406833
Generation: 93  Best fitness: 0.8567291571406833
Generation: 94  Best fitness: 0.8097850937357142
Generation: 95  Best fitness: 0.8332571254381987
Generation: 96  Best fitness: 0.814814814814818
Generation: 97  Best fitness: 0.8230452674897152
Generation: 98  Best fitness: 0.8097850937357142
Generation: 99  Best fitness: 0.8271604938271638
Generation: 100  Best fitness: 0.7887517146776437
Generation: 101  Best fitness: 0.8122237463801282
Generation: 102  Best fitness: 0.8097850937357142
Generation: 103  Best fitness: 0.8097850937357142
Generation: 104  Best fitness: 0.8215211095869565
Generation: 105  Best fitness: 0.8097850937357142
Generation: 106  Best fitness: 0.8518518518518553
Generation: 107  Best fitness: 0.8332571254381987
Generation: 108  Best fitness: 0.8518518518518553
Generation: 109  Best fitness: 0.8518518518518553
Generation: 110  Best fitness: 0.8802011888431678
Generation: 111  Best fitness: 0.814814814814818
Generation: 112  Best fitness: 0.8518518518518553
Generation: 113  Best fitness: 0.8567291571406833
Generation: 114  Best fitness: 0.8332571254381987
```

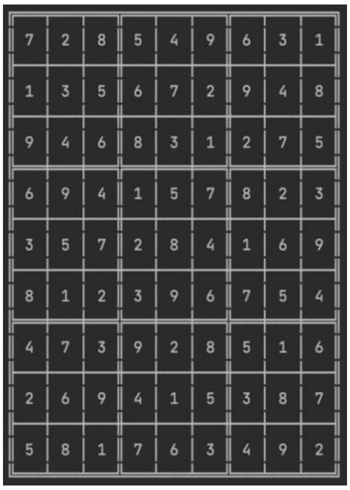
```
Generation: 115  Best fitness: 0.8332571254381987
Generation: 116  Best fitness: 0.8518518518518553
Generation: 117  Best fitness: 0.8518518518518553
Generation: 118  Best fitness: 0.8271604938271638
Generation: 119  Best fitness: 0.8567291571406833
Generation: 120  Best fitness: 0.8518518518518553
Generation: 121  Best fitness: 0.8518518518518553
Generation: 122  Best fitness: 0.8567291571406833
Generation: 123  Best fitness: 0.8518518518518553
Generation: 124  Best fitness: 0.8567291571406833
Generation: 125  Best fitness: 0.8567291571406833
Generation: 126  Best fitness: 0.9012345679012383
Generation: 127  Best fitness: 0.8567291571406833
Generation: 128  Best fitness: 0.8332571254381987
Generation: 129  Best fitness: 0.8332571254381987
Generation: 130  Best fitness: 0.8567291571406833
Generation: 131  Best fitness: 0.8518518518518553
Generation: 132  Best fitness: 0.8518518518518553
Generation: 133  Best fitness: 0.8271604938271638
Generation: 134  Best fitness: 0.8518518518518553
Generation: 135  Best fitness: 0.8518518518518553
Generation: 136  Best fitness: 0.8518518518518553
Generation: 137  Best fitness: 0.8518518518518553
Generation: 138  Best fitness: 0.8765432098765469
Generation: 139  Best fitness: 0.8332571254381987
Generation: 140  Best fitness: 0.8518518518518553
Generation: 141  Best fitness: 0.8518518518518553
Generation: 142  Best fitness: 0.8518518518518553
Generation: 143  Best fitness: 0.8567291571406833
Generation: 144  Best fitness: 0.8518518518518553
Generation: 145  Best fitness: 0.8567291571406833
Generation: 146  Best fitness: 0.9012345679012383
Generation: 147  Best fitness: 0.8802011888431678
Generation: 148  Best fitness: 0.8765432098765469
Generation: 149  Best fitness: 0.8518518518518553
Generation: 150  Best fitness: 0.8518518518518553
Generation: 151  Best fitness: 0.8518518518518553
Generation: 152  Best fitness: 0.8518518518518553
Generation: 153  Best fitness: 0.8395061728395096
Generation: 154  Best fitness: 0.8518518518518553
Generation: 155  Best fitness: 0.8518518518518553
Generation: 156  Best fitness: 0.8518518518518553
Generation: 157  Best fitness: 0.8518518518518553
Generation: 158  Best fitness: 0.8518518518518553
Generation: 159  Best fitness: 0.864197530864201
Generation: 160  Best fitness: 0.9012345679012383
Generation: 161  Best fitness: 0.9012345679012383
Generation: 162  Best fitness: 0.9012345679012383
```

```
Generation: 163  Best fitness: 0.9012345679012383
Generation: 164  Best fitness: 0.9012345679012383
Generation: 165  Best fitness: 0.9012345679012383
Generation: 166  Best fitness: 0.9012345679012383
Generation: 167  Best fitness: 0.8518518518518553
Generation: 168  Best fitness: 0.8518518518518553
Generation: 169  Best fitness: 0.8765432098765469
Generation: 170  Best fitness: 0.8518518518518553
Generation: 171  Best fitness: 0.9012345679012383
Generation: 172  Best fitness: 0.9012345679012383
Generation: 173  Best fitness: 0.8765432098765469
Generation: 174  Best fitness: 0.9012345679012383
Generation: 175  Best fitness: 0.8802011888431678
Generation: 176  Best fitness: 0.9012345679012383
Generation: 177  Best fitness: 0.9012345679012383
Generation: 178  Best fitness: 0.8332571254381987
Generation: 179  Best fitness: 0.8271604938271638
Generation: 180  Best fitness: 0.8518518518518553
Generation: 181  Best fitness: 0.8518518518518553
Generation: 182  Best fitness: 0.8518518518518553
Generation: 183  Best fitness: 0.8765432098765469
Generation: 184  Best fitness: 0.9012345679012383
Generation: 185  Best fitness: 0.9012345679012383
Generation: 186  Best fitness: 0.9506172839506214
Generation: 187  Best fitness: 0.8567291571406833
Generation: 188  Best fitness: 0.8518518518518553
Generation: 189  Best fitness: 0.8518518518518553
Generation: 190  Best fitness: 0.8684651729919255
Generation: 191  Best fitness: 0.8567291571406833
Generation: 192  Best fitness: 0.8567291571406833
Generation: 193  Best fitness: 0.9012345679012383
Generation: 194  Best fitness: 0.8567291571406833
Generation: 195  Best fitness: 0.8518518518518553
Generation: 196  Best fitness: 0.8518518518518553
Generation: 197  Best fitness: 0.8518518518518553
Generation: 198  Best fitness: 0.8518518518518553
Generation: 199  Best fitness: 0.8518518518518553
Generation: 200  Best fitness: 0.8518518518518553
Generation: 201  Best fitness: 0.8518518518518553
Generation: 202  Best fitness: 0.8518518518518553
Generation: 203  Best fitness: 0.8765432098765469
Generation: 204  Best fitness: 0.8765432098765469
Generation: 205  Best fitness: 0.8765432098765469
Generation: 206  Best fitness: 0.8518518518518553
Generation: 207  Best fitness: 0.9012345679012383
Generation: 208  Best fitness: 0.9012345679012383
Generation: 209  Best fitness: 0.9012345679012383
Generation: 210  Best fitness: 0.9012345679012383
```

```
Generation: 211  Best fitness: 0.9012345679012383
Generation: 212  Best fitness: 0.9012345679012383
Generation: 213  Best fitness: 0.9012345679012383
Generation: 214  Best fitness: 0.9012345679012383
Generation: 215  Best fitness: 0.9012345679012383
Generation: 216  Best fitness: 0.9012345679012383
Generation: 217  Best fitness: 0.9012345679012383
Generation: 218  Best fitness: 0.9012345679012383
Generation: 219  Best fitness: 0.8765432098765469
Generation: 220  Best fitness: 0.8765432098765469
Generation: 221  Best fitness: 0.8765432098765469
Generation: 222  Best fitness: 0.8765432098765469
Generation: 223  Best fitness: 0.8518518518518553
Generation: 224  Best fitness: 0.8518518518518553
Generation: 225  Best fitness: 0.8518518518518553
Generation: 226  Best fitness: 0.8518518518518553
Generation: 227  Best fitness: 0.8518518518518553
Generation: 228  Best fitness: 0.8518518518518553
Generation: 229  Best fitness: 0.8518518518518553
Generation: 230  Best fitness: 0.8518518518518553
Generation: 231  Best fitness: 0.8567291571406833
Generation: 232  Best fitness: 0.8567291571406833
Generation: 233  Best fitness: 0.8518518518518553
Generation: 234  Best fitness: 0.864197530864201
Generation: 235  Best fitness: 0.8518518518518553
Generation: 236  Best fitness: 0.8518518518518553
Generation: 237  Best fitness: 0.8567291571406833
Generation: 238  Best fitness: 0.9135802469135841
Generation: 239  Best fitness: 0.8567291571406833
Generation: 240  Best fitness: 0.8518518518518553
Generation: 241  Best fitness: 0.8518518518518553
Generation: 242  Best fitness: 0.8518518518518553
Generation: 243  Best fitness: 0.8518518518518553
Generation: 244  Best fitness: 0.9012345679012383
Generation: 245  Best fitness: 0.9259259259259299
Generation: 246  Best fitness: 0.9012345679012383
Generation: 247  Best fitness: 0.9012345679012383
Generation: 248  Best fitness: 0.9012345679012383
Generation: 249  Best fitness: 0.8765432098765469
Generation: 250  Best fitness: 0.9012345679012383
Generation: 251  Best fitness: 0.9012345679012383
Generation: 252  Best fitness: 0.9012345679012383
Generation: 253  Best fitness: 0.9012345679012383
Generation: 254  Best fitness: 0.9012345679012383
Generation: 255  Best fitness: 0.9012345679012383
Generation: 256  Best fitness: 0.9012345679012383
Generation: 257  Best fitness: 0.9506172839506214
Generation: 258  Best fitness: 0.9012345679012383
```

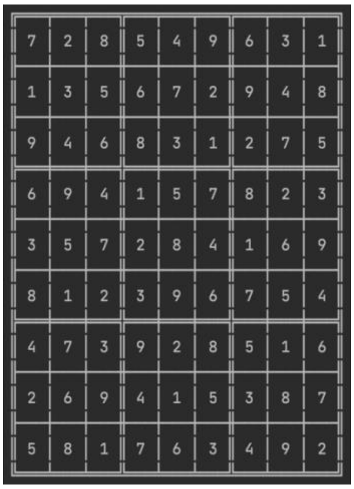
```
Generation: 259  Best fitness: 0.9012345679012383
Generation: 260  Best fitness: 0.9012345679012383
Generation: 261  Best fitness: 0.9012345679012383
Generation: 262  Best fitness: 0.9012345679012383
Generation: 263  Best fitness: 0.9012345679012383
Generation: 264  Best fitness: 0.9012345679012383
Generation: 265  Best fitness: 0.9012345679012383
Generation: 266  Best fitness: 0.9012345679012383
Generation: 267  Best fitness: 0.9012345679012383
Generation: 268  Best fitness: 0.9012345679012383
Generation: 269  Best fitness: 0.9012345679012383
Generation: 270  Best fitness: 0.9012345679012383
Generation: 271  Best fitness: 0.9012345679012383
Generation: 272  Best fitness: 0.9012345679012383
Generation: 273  Best fitness: 0.9012345679012383
Generation: 274  Best fitness: 0.9012345679012383
Generation: 275  Best fitness: 0.9012345679012383
Generation: 276  Best fitness: 0.9012345679012383
Generation: 277  Best fitness: 0.9012345679012383
Generation: 278  Best fitness: 0.9012345679012383
Generation: 279  Best fitness: 0.9012345679012383
Generation: 280  Best fitness: 0.9012345679012383
Generation: 281  Best fitness: 0.9012345679012383
Generation: 282  Best fitness: 0.9506172839506214
Generation: 283  Best fitness: 0.9012345679012383
Generation: 284  Best fitness: 0.9012345679012383
Generation: 285  Best fitness: 0.9506172839506214
Generation: 286  Best fitness: 0.9012345679012383
Generation: 287  Best fitness: 0.9012345679012383
Generation: 288  Best fitness: 0.9012345679012383
Generation: 289  Best fitness: 0.9259259259259299
Generation: 290  Best fitness: 0.9259259259259299
Generation: 291  Best fitness: 0.9506172839506214
Generation: 292  Best fitness: 0.9506172839506214
Generation: 293  Best fitness: 0.9012345679012383
Generation: 294  Best fitness: 0.9012345679012383
Generation: 295  Best fitness: 0.9012345679012383
Generation: 296  Best fitness: 0.9012345679012383
Generation: 297  Best fitness: 0.9506172839506214
Generation: 298  Best fitness: 0.9012345679012383
Generation: 299  Best fitness: 0.9012345679012383
Generation: 300  Best fitness: 0.9012345679012383
Generation: 301  Best fitness: 0.9012345679012383
Generation: 302  Best fitness: 0.9506172839506214
Generation: 303  Best fitness: 0.9506172839506214
Generation: 304  Best fitness: 0.9012345679012383
Generation: 305  Best fitness: 0.9259259259259299
Generation: 306  Best fitness: 0.9259259259259299
```

```
Generation: 307  Best fitness: 0.9506172839506214
Generation: 308  Best fitness: 0.9012345679012383
Generation: 309  Best fitness: 0.9012345679012383
Generation: 310  Best fitness: 0.9012345679012383
Generation: 311  Best fitness: 0.9506172839506214
Generation: 312  Best fitness: 0.9012345679012383
Generation: 313  Best fitness: 0.9012345679012383
Generation: 314  Best fitness: 0.9506172839506214
Generation: 315  Best fitness: 0.9012345679012383
Generation: 316  Best fitness: 0.9506172839506214
Generation: 317  Best fitness: 0.9506172839506214
Generation: 318  Best fitness: 0.9012345679012383
Generation: 319  Best fitness: 0.9259259259259299
Generation: 320  Best fitness: 0.9506172839506214
Generation: 321  Best fitness: 0.9012345679012383
Generation: 322  Best fitness: 0.9259259259259299
Generation: 323  Best fitness: 0.9259259259259299
Generation: 324  Best fitness: 0.9012345679012383
Generation: 325  Best fitness: 0.9012345679012383
Generation: 326  Best fitness: 0.9012345679012383
Generation: 327  Best fitness: 0.9135802469135841
Generation: 328  Best fitness: 0.9135802469135841
Generation: 329  Best fitness: 0.9506172839506214
Generation: 330  Best fitness: 0.9506172839506214
Generation: 331  Best fitness: 0.9506172839506214
Generation: 332  Best fitness: 0.9506172839506214
Generation: 333  Best fitness: 0.9259259259259299
Generation: 334  Best fitness: 0.9506172839506214
Generation: 335  Best fitness: 0.9506172839506214
Genetic Algorithm Solution found at generation: 336
Time elapsed: 170.45s
```



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

```
Execution Started: Backtracking
Backtracking Solution found:
Time elapsed:   0.52s
```



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

Comparison of Genetic algorithm and Back tracking algorithm to solve sudoku.

Puzzle Level	Number of Givens	Time Elapsed (in Seconds) -Genetic Algorithm	Time Elapsed (in Seconds) -Backtracking	Number of Generations – Genetic Algorithm
Easy	38	6.73s	0.00s	8
Medium	30	40.16s	0.00s	49
Hard	25	72.90s	0.02s	82
Expert	23	170.45s	0.52s	336

5. Conclusion and Future Enhancements:

This project has been successful in solving the Sudoku game and demonstrates two separate implementations for solving Sudoku using Genetic algorithm and Backtracking algorithm. From the experiments conducted, analyzing the above comparison table, genetic algorithm approach performs poorly in relation to standard backtracking algorithms, at least for a 9x9 grid. The Sudoku solver often gets stuck in local minima and requires restarts to successfully find the problem solution, especially for hard and expert level problems. To solve Sudoku, back-tracking method is more efficient and computationally feasible than Genetic algorithm. I gained a lot of knowledge on how to encode various optimization problems to Genetic Algorithm.

The system can be further extended, and a User Interface can be added to visualize the population creation, cross-over and mutation. Different methods of Selection, Cross-over, and mutation can be implemented and compared.

References:

[1] https://en.wikipedia.org/wiki/Sudoku

[2] https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3

[3] https://www.geeksforgeeks.org/sudoku-backtracking-7/

[4] Feigenbaum, Edward (1988). The Rise of the Expert Company. Times Books. p. 318. ISBN 0-8129-1731-6

[5] https://www.iitg.ac.in/dsharma/papers/Conference/2016_Suduko.pdf

[6] https://www.iosrjournals.org/iosr-jce/papers/Vol16-issue5/Version-6/K016566977.pdf

[7] https://www.human-competitive.org/sites/default/files/sato-chapter-19-paper.pdf

[8] https://stackoverflow.com/questions/13327819/solving-sudoku-using-a-genetic-algorithm

[9] https://stackoverflow.com/questions/9639110/genetic-algorithm-sudoku-optimizing-mutation?rq=1

[10] http://micsymposium.org/mics_2009_proceedings/mics2009_submission_66.pdf

[11] https://stackoverflow.com/questions/45471152/how-to-create-a-sudoku-puzzle-in-python