# Solving Logic Puzzles with Genetic Algorithm

**Darunee Bunma, Prabhas Chongstitvatana**

Department of Computer Engineering, Faculty of Engineering,
Chulalongkorn University, Bangkok, Thailand
Darunee.b@student.chula.ac.th, Prabhas.c@chula.ac.th

**ABSTRACT**

This work proposes a genetic algorithm with special encoding and operators to solve Sudoku puzzles. The proposed method can also be applied to a wide range of logic puzzles. Sudoku puzzle is very complex, hence solving it successfully is a challenge. The proposed algorithm is tested on four hardness levels of the puzzles: easy, medium, hard and expert. The result from the experiment shows that the proposed algorithm works very well, with the success rate 100% and the run-time is competitive with the existing methods.

**Keywords:** Genetic Algorithm, Sudoku Puzzle, Mutation operator

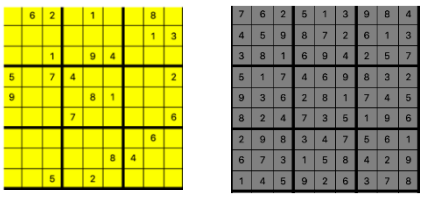
## 1. INTRODUCTION

The purpose of this puzzle is to compose groups of permutation value in each row, column, and sub-block. The applications for solving these puzzles derived from several group of genetic algorithms (GA) and estimation of distribution algorithm (EDA) variety of optimization for metaheuristics algorithms.

Sudoku puzzles are composed of $n^2 x\, n^2$ grid/board and divided into $n^2$ different $n\, x\, n$ sub grids (Fig.1). To solve Sudoku puzzles of $3^2$ x $3^2$ grid must be followed:

1. Each row and column has all integers from the set 1-9 without repetition.
2. Each sub-grid 3x3 has all integers from the set 1-9 without repetition.

**No.257 Expert Level (20 Givens)**



(a) The initial problem (b) The result

***Fig. 1:*** *Sudoku puzzles*

This work proposes a genetic algorithm with special encoding and operators to solve Sudoku puzzles. The proposed method can also be applied to a wide range of logic puzzles.

## 2. RELATED WORK

Meta-heuristics shows how different type of evolutionary techniques such as GA, PSO and ACO have been efficiently used to solve Sudoku puzzles [1]. An implementation of GA for solving Sudoku was proposed by Mantere and Koljonen [2]. They presented the mutation technique that was designed to be used for sub-blocks (3x3). Sato, and Hazuki Inoue [3] proposed a local search technique with a fitness function (Eq.1) where $g_i\,(x)$ refers to the amount of unique number in horizontal line and $h_j(x)$ refers to the amount of unique number in vertical line, the mutation operator is a simple 2 swap operation.

$$f(x) = \sum_{i=1}^{9} g_i\,(x) + \sum_{j=1}^{9} h_j\,(x) \qquad (1)$$

$$where\ g_i(x) = |x_i|\ and\ h_j(x) = |x_j|$$

The Restart Estimation of Distribution algorithm (RESEDA) [4] this algorithm is initial numbers in order to reduce the search spaces by not allowing number to be changed as those numbers already have been used. The function probabilistic model shown in (Eq.2)

$$p_{i,j,k}^{(n)} = ap_{i,j,k}^{(n)} + (1-\alpha) r_{i,j,k}^{(n+1)} \qquad (2)$$

where $0 \le \alpha \le 1$. $r_{i,j,k}^{(n+1)}$ is a probability distribution of the $k$ number of candidates found in population and the parent $\alpha$ represent a faster convergence process of learning rate function in specifies with larger value. If the algorithm getting trapped in a local optimum and then stuck condition restart the optimization process.

## 3. THE PROPOSED ALGORITHM

### 3.1 Encoding

A constraint array determined the free position where "0" mean the position can be changed and "1" means the position is fixed. (Fig.2)

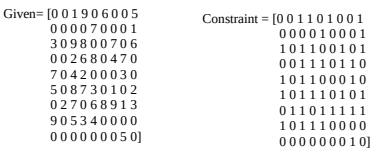



***Fig. 2****: Encoding of a solution and the constraint*

### 3.2 Initial Population

A GA starts with a set of candidates are also known as the chromosomes in the population. It is not allowed duplicates in the same sub-block. However, it is possible for duplicates to occur in rows and/or columns.

### 3.3 Selection

The tournament selection is employed. The result is two individuals designated as parents.

### 3.4 Crossover

The crossover function requires parents to be selected from the population by the selection method above two rows from parents are randomly selected. The free positions of two rows are exchanged.

### 3.5 Mutation

The mutation is applied between possible candidates that resulted in not worsening the fitness value. For example, the candidates and their positions are shown in red (Fig. 3a). Let's assume the positions m1 and m2 are selected for swap mutation. Two positions are considered, swap 5 with 1, or swap 5 with 9 (Fig. 3b). The selected one will be the one that does not lower the fitness value of the solution. This way is a normal re-production process offspring candidate by mutate parents.

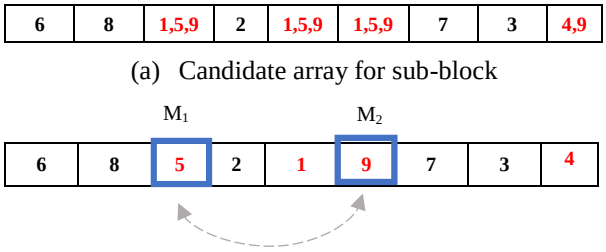
### 3.6 Restart

If the evolution get stuck (the fitness does not improve within 100 generations) then re-seed the initial population.

The parameters for running genetic algorithm is shown in ***Table 1:***

***Table 1:*** *GA Parameters*

| Description | Value |
|---|---|
| population size | 1,000 |
| maximum number of generations | 10,000 |
| mutation rate | 0.06 |
| number of elites | 50 |
| Tournament selection (select the better one with 0.8 probability) | 0.8 |




(a) Candidate array for sub-block

(b) Allowed and the lower case when the swap mutation is prohibited

***Fig. 3.*** *Mutation operation*

## 4. RESULT

The results are shown in Table 2 and 3. The proposed algorithm compared very well with the competing one in both criteria. In terms of run-time performance, the proposed algorithm is faster in all puzzles.

***Table 2:*** *Success rates*

| Difficulty | Success rate | Success rate (our approach) |
|---|---|---|
| Easy1 & 2 | 100 | 100 |
| Medium1&2 | 100 | 100 |
| Hard1&2 | 100 | 100 |
| Expert1&2 | Cannot be Solved | 100 |

***Table 3:*** *Speed of calculation*

| Difficulty | Duration (ms) | Duration (our approach) (ms) |
|---|---|---|
| Easy | 8 | 1 |
| Medium | 44 | 20 |
| Hard | 87 | 44 |
| Expert | Cannot be Solved | 257 |

## 5. CONCLUSION

In this work we have shown that the proposed genetic algorithm is effective in solving Sudoku puzzles. It demonstrates good quality in search with success rate always reaching 100%. The run-time is also competitive. The algorithm can be applied to solve a wide variety of optimization problems

## 6. REFERENCES


[1] Waiyapara, K., Wattanapornprom, W. and Chong stitvatana, P. (2013) "Solving Sudoku Puzzles with Node Based Coincidence Algorithm," Int. Joint Conf. on Computer Science and Software Engineering, pp.11-16

[2] Mantere, T. and Koljonen, J. (2007) "Solving, rating and generating Sudoku puzzles with GA," IEEE Congress on Evolutionary Computation.

[3] Sato, Y. and Inoue, H. (2010) "Solving Sudoku with genetic operations that preserve building blocks," IEEE Conference on Computational Intelligence and Games, pp. 23-29, August 2010.

[4] Maire, S. and Prissette, C. (2012) "A restarted estimation of distribution algorithm for solving sudoku puzzles," in Monte Carlo Methods and Applications, Sabelfeld, K. (eds), vol.18, issue 2.