# Genetic Operations to Solve Sudoku Puzzles

Yuji Sato
Hosei University
3-7-2 Kajino-cho Koganei-shi
Tokyo 184-8584 JAPAN
+81-42-387-4533
yuji@k.hosei.ac.jp

Hazuki Inoue
Hosei University
3-7-2 Kajino-cho Koganei-shi
Tokyo 184-8584 JAPAN
+81-42-387-4533
haduki.inoue.ec @cis.hosei.ac.jp

## ABSTRACT
Genetic operations that consider effective building blocks are proposed for using genetic algorithms to solve Sudoku puzzles. A stronger local search function is also proposed. Evaluation of the proposed techniques using commercial Sudoku puzzle sets and puzzles ranked as super difficult compared with previously reported examples show that the rate of optimum solutions can be greatly improved.

## Categories and Subject Descriptors
D.0 [**Computer Applications**]: General

## General Terms
Algorithms, Performance, Design, Experimentation, Verification.

## Keywords
Genetic Algorithms, Genetic Operation, Building Blocks, Sudoku.

## 1. INTRODUCTION
Sudoku, a pencil and paper puzzle, is considered as a large combinatorial optimization problem. A Sudoku puzzle is completed by filling in all of the empty cells with numerals 1 to 9, but no row or column and no 3 x 3 sub-block may contain more than one of any numeral. There are also many variations of Sudoku. Some are puzzles of larger size, such as 16 x 16 or 25 x 25. Others impose additional constraints, such as not permitting the same numeral to appear more than once in diagonals or in special sets of 9 cells that have the same color, etc. For these basic puzzles, methods such as back-tracking, which counts up all possible combinations in the solution, are effective, but for larger puzzles such as 16 x 16 or 25 x 25, GA or other stochastic search method may be effective.

A number of studies on application of GA to solving Sudoku have already been made. On the other hand, there seems to be relatively few scientific papers. Reference [1], for example, defines a one-dimensional chromosome that has a total length of 81 integer numbers and consists of linked sub-chromosomes for each 3x3 sub-block of the puzzle, and applies uniform crossover in which the crossover positions are limited to the links between sub-blocks. Reference [2] uses the same chromosome definition, but compares the effectiveness for different crossover methods, including one-point crossover that limit crossover points to links between sub-blocks, two-point crossover, and crossover in units of row or column. In these examples, optimum solutions to simple puzzles are easily found, but the optimum solutions for difficult puzzles in which the starting point has few givens are often not obtainable in realistic time. That led to approaches in which GA was combined with Grammatical Evolution (GAuGE) [3]. We believe the reason for the failure of this design is that the main GA operation, crossover, tends to destroy effective partial solutions (BB). We propose below a means of dealing with that problem.


*GECCO'10*, July 7–11, 2010, Portland, Oregon, USA.
ACM 978-1-4503-0073-5/10/07.

## 2. METHOD FOR DEARING WITH THE BUILDING BLOCK PROBLEM
Fill in the cells that do not contain given values in the starting point with random numerals such that there are no more than one of any numeral in any sub-block. By applying this operation to all sub-blocks, the Sudoku rule that "no numeral is duplicated within any sub-block" is satisfied.

The crossover operations emphasize averting destruction of BB over creating diversity in the search process. When two child individuals are generated from two parents, scores are obtained for each of the three rows that constitute the sub-blocks of the parents, and a child inherits the ones with the highest scores. Then the columns are compared in the same way and the other child inherits the ones with the highest scores. The reason for considering the scores of the rows or columns in sub-blocks when generating child individuals is to avoid a kind of hitchhiking phenomenon. Columns that have a low score but neighbor a column that has a high score are inherited together with the high-score column, so the overall score of the child does not improve. That is to say, checking the scores for each of the three rows that constitute a sub-block and passing the those of the highest score on to the child reduces the probability of a high score BB being reduced to a low score BB by hitchhiking. We can also expect less destruction of BB that comprises cell rows or columns than occurs with the crossover method used in the conventional example [1, 2, 4].

Mutations are performed for each sub-block according to the mutation rate. Two numerals within a sub-block that are not given in the starting point are selected randomly and their positions are swapped. This operation avoids duplication of numerals within a sub-block and inclusion of initially given numerals in the randomly selected numerals.

Generally, GA does not perform as well in local search as in global search. Thus, we tried to strengthen the local search function by using mutation to generate a number of children that is an integer multiple of the number of parents. Selecting the high-scored individuals from among the generated child candidates allows good efficiency in the search for the optimum solution.

# 3. EXPERIMENT

## 3.1 Experimental method

These experiments use tournament selection. The fitness function is based on the rule that there can be no more than one of any numeral in a row or column. The score is the number of different elements in a row or column, and the sum of the row and column scores is the score for the individual.

For the puzzles used to investigate the effectiveness of the genetic operations proposed in section 3, we selected two puzzles from each level of difficulty in the puzzle set from the book [5]. For comparison with the conventional examples, we also used the particularly difficult Sudoku puzzles introduced by Timo Mantere in reference [6]. The experimental parameters are population size: 150, number of child candidates/parents: 2, crossover rate: 0.3, mutation rate: 0.3, tournament size: 3.

## 3.2 Experimental results

### *3.2.1 Benchmark test*

The relation between the number of givens in the starting point and the number of generations required to reach the optimum solution is shown in Table 1. For the three cases in which only mutation is applied (a kind of random search), when mutation and the proposed crossover method are applied (mut+cross), and when the local search improvement measure is applied in addition to mutation and crossover (mut+cross+LS), the tests were run 100 times and the averages of the results were compared. If a solution was not obtained before 100,000 generations, the result was displayed as 100,000 generations. When the search is terminated at 100,000 generations, the proportion of obtaining an optimum solution for a difficult puzzle was clearly improved by adding the proposed crossover technique to the mutation, and improved even further by adding the local search function. The mean number of generations until a solution is obtained is also reduced.

**Table. 1 The comparison of how effectively GA finds solutions for the Sudoku puzzles with different difficulty ratings. Count shows how many of the 100 GA test runs found the solution and Average shows how many GA generations was need to find the solution with 100,000 trials.**

| Difficulty rating | Givens | mut+cross+LS | | mut+cross | | Swap mutation | |
|---|---|---|---|---|---|---|---|
| | | Count | Average | Count | Average | Count | Average |
| Easy (No. 1) | 38 | 100 | 62 | 100 | 105 | 100 | 223 |
| Easy (No. 11) | 34 | 100 | 137 | 100 | 247 | 96 | 6627 |
| Medium (No. 27) | 30 | 100 | 910 | 100 | 2274 | 86 | 26961 |
| Medium (No. 29) | 29 | 100 | 3193 | 100 | 6609 | 66 | 42141 |
| Difficult (No. 77) | 28 | 100 | 9482 | 100 | 20658 | 35 | 77573 |
| Difficult (No. 106) | 24 | 96 | 26825 | 74 | 56428 | 9 | 94314 |

**Table. 2 Our result and the result represented in [1]**

| Givens | 36 | 33 | 30 | 28 | 25 | 23 | 22 |
|---|---|---|---|---|---|---|---|
| Our methods | 100 | 100 | 100 | 100 | 100 | 100 | 93 |
| Mantere-2006 [1] | 100 | 100 | 69 | 46 | 30 | 4 | 6 |

### *3.2.2 Comparison with previous research*

The results are compared with previous research in Table 2. In Table 2, for termination of search after 100,000 generations, the numbers of optimum solutions obtained with our method within 100 test runs for various starting points are compared with the results of reference [1]. We selected our benchmark Sudoku puzzles from the book [5]. On the other hand, they had taken their benchmark puzzles from the newspaper Helsingin Sanomat (2006) and newspaper Aamulehti (2006) that was unavailable for us. Moreover, their population size was 100 and our size was 150. Therefore we cannot directly compare our results with their method. But the proposed method clearly increased the solution rate for medium and difficult Sudoku puzzles.

Next, the number of times the optimum solution was obtained in 100 test runs using AI Escarcot [6], said to be one of the most difficult Sudoku puzzles, are compared with the results of reference [4]. Without any trial limitation both method was solved every time. On the other hand, when using a limit of 100,000 trials, their method was solved only 5 times out of 100 test runs and our method was solved 83 times. Their population size was 11, therefore we cannot directly compare our results with their method, but our method clearly increased the solution rate for super difficult Sudoku "AI Escarcot".

# 4. CONCLUSIONS

We proposed GA operations that cope with the building block destruction that is problematic when GA is applied to solving Sudoku puzzles. We also attempted to improve the accuracy of puzzle solution by adding a local search function to GA. The result is that solutions could be found with high probability, even for the puzzles that are ranked as super difficult and are difficult to solve by the conventional method.

# 5. REFERENCES


[1] Mantere, T., and Koljonen, J. Solving and Ranking Sudoku Puzzles with Genetic Algorithms. In *Proceedings of the 12th Finnish Artificial Conference STeP 2006*, 2006, 86-92.

[2] Moraglio, A., Togelius, J., and Lucas, S. Product Geometric Crossover for the Sudoku Puzzle. In *Proceedings of the IEEE Congress on Evolutionary Computation*, 2006, 470-476.

[3] Nicolau, M. and Ryan, C. Genetic Operators and Sequencing in the GAuGE System. In *Proceedings of the IEEE Congress on Evolutionary Computation*, 2006, 5710-5717.

[4] Mantere, T. and Koljonen, J. Analyzing and Solving Sudokus with Cultural Algorithms. In *Proceedings of the IEEE Congress on Evolutionary Computation*, 2008, 4054-4061.

[5] Number Place Plaza (eds.). *Number Place Best Selection 110*. Vol.15, ISBN-13: 978-4774752112, Cosmic mook, 2008.

[6] Super difficult Sudoku's. Available via WWW: http://lipas.uwasa.fi/~timan/sudoku/EA_ht_2008.pdf#search='CT20A6300%20Alternative%20Project%20work%202008' (cited 8.3.2010).