Solving Sudoku with Genetic Operations that Preserve Building Blocks

Yuji Sato, *Member, IEEE*, and Hazuki Inoue

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 three puzzles ranked as super difficult compared with previously reported examples show that the rate of optimum solutions can be greatly improved. It is demonstrated that even further improvement in accuracy is expected from a correction technique based on disparity hypothesis.**

I. Introduction

Sudoku [1], a pencil and paper puzzle that was devised by an American architect in 1979, is considered as a large combinatorial optimization problem. Currently, Sudoku seems to be a major fad in England and is spreading in popularity around the world, including Japan. Genetic algorithms (GA) [2], one method of stochastic search, are effective for large combinatorial optimization problems, and can obtain solutions within realistic times for the optimum solution or an approximate solution for the traveling salesman problem (TSP) [3] and the *N*-queen problem and other such large-scale problems that are difficult to solve by conventional methods. That has motivated research on applying GA to solve Sudoku, and there are reports on success with relatively simple puzzles that have many given numbers in the initial placement [4-6]. There also exist research on applying GA to generate new puzzles efficiently. For difficult puzzles in which there are fewer given numbers, however, the number of generations needed to reach the optimum solution becomes enormous and the probability of obtaining a solution in realistic time becomes small [4-6]. That led to approaches in which GA was combined with Cultural Algorithms [7] and Grammatical Evolution [8] and other such methods to improve the accuracy and efficiency of solution search. We believe reason for the ineffectiveness of GA in solving difficult Sudoku is not simply the large combination search space, but that the existence of a large number of local solutions is also factor. The emphasis on maintenance of diversity in the definition of genetic operations in previous examples thus resulted in destruction of effective building blocks (BB) [9, 10] in the search. In this paper, we propose genetic operations that focus on averting the destruction of BB in an attempt to increase the accuracy of GA solutions of Sudoku puzzles. Section 2 gives a simple explanation of the Sudoku rules; section 3 presents our proposed genetic operations that emphasize preservation of BB; section 4 presents and discusses the experimental results and is followed by the conclusion.

Yuji Sato is with the Faculty of Computer and Information Sciences, Hosei University, Tokyo 184-8584, Japan (email: yuji@k.hosei.ac.jp).

Hazuki Inoue is with the Faculty of Computer and Information Sciences, Hosei University, Tokyo 184-8584, Japan (email: haduki.inoue.yw @gs-cis.hosei.ac.jp).

II. The Rules of Sudoku

The Sudoku rules are explained in Fig. 1. General Sudoku puzzles consist of a 9 x 9 matrix of square cells, some of which already contain a numeral from 1 to 9. The arrangement of given numerals when the puzzle is presented is called the starting point. In Fig. 1, it contains 24 nonsymmetrical given numbers, and the correct number for the other 57 points should be solved. The degree of difficulty varies with the number of given numerals and their placement. Basically, fewer given numerals means a higher number of combinations among which the solution must be found, and so raises the degree of difficulty. But, there are about 15 to 20 factors that have an effect on difficulty rating [1]. 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 (the sub-blocks are bound by heavy lines in Fig. 1) may contain more than one of any numeral. An example solution to the example Sudoku puzzle given in Fig. 1 is shown in Fig. 2. In this figure, the given numbers marked in bold-face.

For these basic puzzles, methods such as back-tracking, which counts up all possible combinations in the solution, and meta-heuristics approach [11] are effective. There also exist some effective algorithms to solve Sudoku puzzles [12-14] and faster than GA.

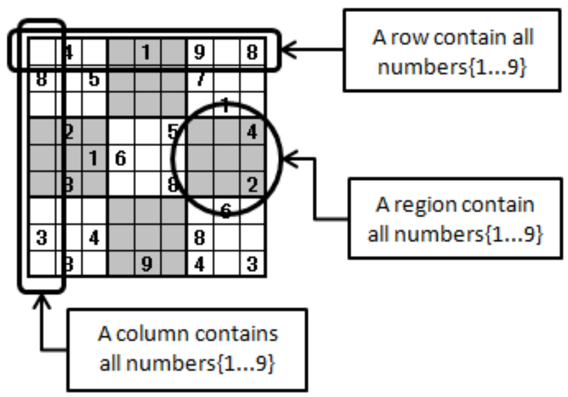
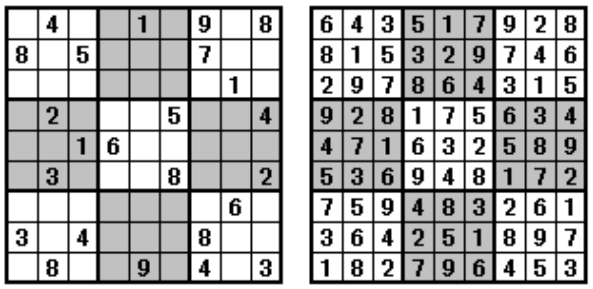

A row contain all numbers{1...9}
A region contain all numbers{1...9}
A column contains all numbers{1...9}

Fig. 1. An example of Sudoku puzzles, 24 positions contain a given number, the other position should be solved.



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

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

Fig. 2. A solution for the Sudoku puzzles given in fig 1. The given numbers marked in bold-face.

On the other hand, 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 larger puzzles such as 16 x 16 or 25 x 25, GA or other stochastic search method may be effective. Methods for speeding up evolutionary computations through implementations on graphics processing units (GPU) may be also effective.

III. Sudoku Solution Accuracy by GA

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 [5, 6], 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 [4] 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 Cultural Algorithms [7], and with Grammatical Evolution (GAuGE) [8]. GAuGE optimizes the sequence of logical operations that are then applied to find the solution. We believe the reason for the failure of this design is that the main GA operation, crossover, tends to destroy highly fits, schemata (BB) [9, 10]. We propose below a means of dealing with that problem.

A. Proposed method for dealing with the building block problem

1) Initial settings: 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. We define this 9 x 9 two-dimensional array as the GA chromosome.

2) Crossover: 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.

An example of crossover is shown in Fig. 3. In this figure, we assumed that the highest score of each row or column is 9. Therefore, the highest score of each row or column in sub-blocks becomes 27. Child 1 inherits the row information from parent 1, parent 2, and parent 1 in order from top to bottom. Child 2 inherits the column information from parent 1, parent 2, and parent 2 in order from left to right.

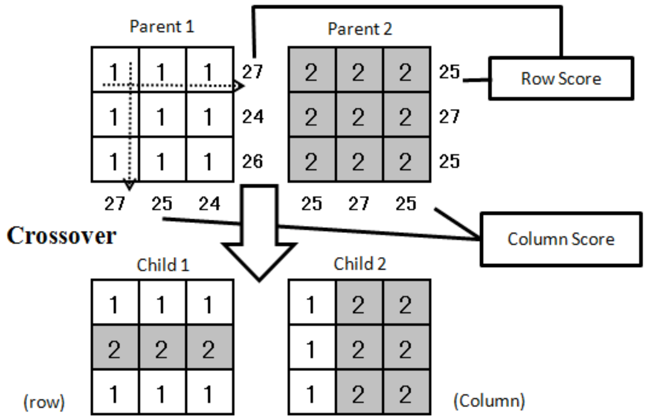


Fig. 3. An example of the crossover considered the rows or the columns that constitute the sub-blocks.

The reason for defining the two-dimensional array as the chromosome and considering the scores of the rows or columns in sub-blocks when generating child individuals is to avoid highly fit schema of long length and destruction of BB that comprises cell rows or columns in sub-blocks. An example of conventional design is shown in Fig. 4. In this figure, the chromosome is defined as one-dimensional array of 81 numbers that is divided into nine sub blocks and the crossovers points can only appear between sub blocks. From this figure, we can know that this design can avoid the crossover point be inside a sub-blocks. On the other hand, this design generate chromosomes comprises highly fit schema of long length that is constructed from cell rows or columns in sub-blocks, and this highly fit schemata (BBs) are easy to be destructed by the crossover operation. This design also cause a kind of hitchhiking [15] phenomenon. For example, sub-blocks that have a low score but neighbor a sub-block that has a highly score are inherited together with the high-score sub-block, 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

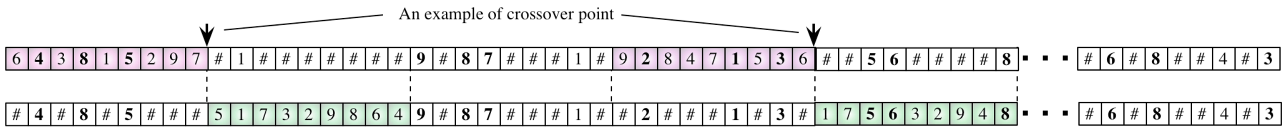


Fig. 4. An example of conventional design of the chromosome and the crossover operation. The chromosome is defined as one-dimensional array of 81 numbers that is divided into nine sub blocks and the crossovers points can only appear between sub blocks.

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 and also expect less destruction of BB that comprises cell rows or columns than occurs with the crossover method used in the conventional example [4-7].

3) Mutation: 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.

An example of mutation using the upper left sub-block of the puzzle is shown in Fig. 5. The numbers displayed in gray are given in the starting point. In this example, the starting point of the sub-block includes the three given numbers {4, 5, 8}. Therefore, two numbers are selected randomly from the set {1, 2, 3, 6, 7, 9}, which excludes the three given numbers. The figure shows an example of swapping in which the two numerals 6 and 7 are selected and swapped.

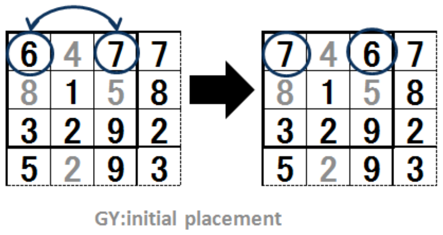


Fig. 5. An example of the swap mutation. Two numbers inside the sub block are selected randomly if the numbers are free to change.

B. Proposal for a more powerful local search

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. We then selecting from them a equal to the number of parents on order of highest score. The concept for improving the local search capability is illustrated in Fig. 6. The figure depicts the case in which a very small space around the parent is extracted from a large search space when the child is generated. If one parent generates one child, then a child that is even further from the optimum solution than the parent may be generated, even when the parent is already near the optimum solution. Thus, the path leading to the optimum solution may be highly circuitous. If one parent generates multiple child individuals, on the other hand, the dispersion in direction of the child individuals relative to the parent increases the probability of a child that is closer to the optimum solution than the parent being generated. Selecting the high-scored individuals from among the generated child candidates allows good efficiency in the search for the optimum solution.

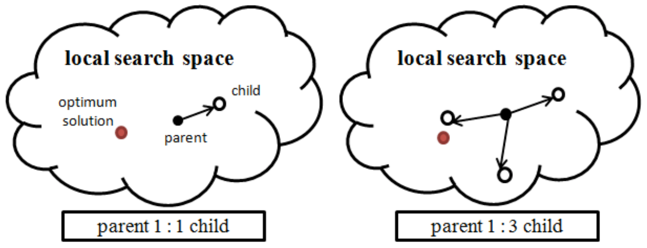


Fig. 6. A concept for improving the local search capability.

IV. EXPERIMENT

A. Experimental method

These experiments use tournament selection [9]. The fitness function, Eq. (1), 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 (g_i) or column (h_j), and the sum of the row and column scores is the score for the individual.

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

$$g_i(x) = |x_i|,\quad h_j(x) = |x_j|$$

Here, | . | indicates the number of different numerals in a particular row or column. Therefore, maximum score of the fitness function $f(x)$ becomes 162.

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 a book [16]: puzzles 1 and 11 from the easy level, 29 and 27 from the intermediate level, and 77 and 106 from the difficult level, for a total of six puzzles. An example of the puzzles used in the experiment is shown in Fig. 7.

No. 1

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

(a) Easy level Sudoku (Givens: 38)

No. 11

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

(b) Easy level Sudoku (Givens: 34)

No. 27

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

(c) Medium level Sudoku (Givens: 30)

No. 29

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

(d) Medium level Sudoku (Givens: 29)

No. 77

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

(e) Difficult level Sudoku (Givens: 28)

No. 106

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

(f) Difficult level Sudoku (Givens: 24)

Fig. 7. The puzzles used to investigate the effectiveness of the genetic operations proposed in this paper.

SD1

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

(a) GA generated Super difficult Sudoku (Givens: 24)

SD2

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

(b)AI Escarcot - Claim to be the most difficult Sudoku (Givens: 23)

SD3

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

(c) Super difficult Sudoku from www.sudoku.com (Givens: 22)

Fig. 8. The puzzles used for comparison with the conventional example. These three Sudoku puzzles were taken from http://lipas.uwasa.fi/~timan/sudoku/EA_ht_2008.pdf#search='CT20A6300%20Alternative%20Project%20work%202008'

For comparison with the conventional examples, we also used the particularly difficult Sudoku puzzles introduced by Timo Mantere in reference [17]. The puzzle used for comparison with the conventional example is shown in Fig. 8.

The experimental parameters are described below.

[Population size] 150

[Number of child candidates/Parents] 2

[Crossover rate] 0. 3

[Mutation rate] 0. 3

[Tournament size] 3

B. *Experimental results*

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 and Fig. 9. 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. The termination point for the search was 100,000 generations. 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 [5].

Givens	36	33	30	28	25	23	22
Our methods	100	100	100	100	100	100	93
Mantere-2006 [5]	100	100	69	46	30	4	6

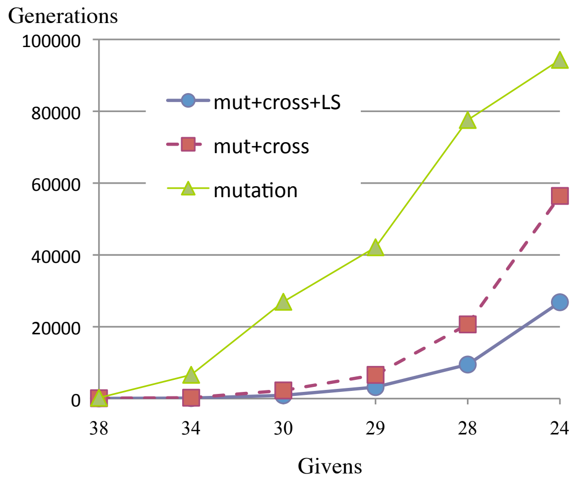


Fig. 9. Relationship between givens and the average of how many GA generations were need to find the solution. (100,000 trials, 100 test runs).

2) Comparison with previous research: The results are compared with previous research in Table 2 and Table 3. We decided to compare our results with the study in ref. [5, 7]. Because, they had already compared their result with other studies [4, 8]. 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 [5]. We selected our benchmark Sudoku puzzles from the book [16]. 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. In ref. [6] they made significantly improved the solving rate of Sudoku compared to their previous version [5], and they also reduced their population size to 21. 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.

In Table 3, the number of times the optimum solution was obtained in 100 test runs using AI Escarcot [18], said to be one of the most difficult Sudoku puzzles, are compared with the results of reference [7]. 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”.

Table. 3 Our result and the result represented in [7].

Sudoku puzzle	Our proposed GA 100, 000 trials	Mantere-2008 [7] 100, 000 trials
AI Escarcot	83	5

The numbers represents how many times out of 100 test runs each method reached the optimum.

V. Discussion

From Table 1 and Fig. 9, we can see that the proposed crossover method and local search strengthening measure are effective. For example, when the starting point provides 30 numbers, addition of the proposed crossover method reduced the number of generations required to obtain the optimum solution to about 10% in comparison with using mutation alone. By adding improved local search, it was further reduced to less than 5%. The difference in the results of the three methods decreased as the number of givens in the starting point decreased, but the reason for that tendency is that the search was terminated after only 100,000 generations to reduce the time required for the experiment. The reasons why this is a good crossover are because (i) it preserves feasibility of the regions, so it searches the smaller space of solutions with feasible regions (ii) it uses partial evaluations of row and columns to build good offspring greedily, so it uses extra problem knowledge in the search operator (iii) child 1 and child 2 are constructed complementarily exploiting separately rows and columns, so it uses two types of complementary greedy constructions.

Next, the results presented in Table 2 and Table 3 show the optimum solution rate is greatly improved relative to the previous method, particularly for difficult puzzles such as AI Escargot. With the previous method, the optimum solution was obtained five times in 100 trials when the search termination point was set to 100,000 generations. With the proposed method, the optimum solution was obtained 83 times under the same conditions. The reason for the difference is considered to be that the conventional method emphasizes diversity in the crossover method, which results in destruction of building blocks in the search process. While GA generally place importance on maintaining diversity in the search process, our approach in applying GA to solving Sudoku places more importance on preserving BB in the crossover method. We can thus consider the proposed method to be effective for Sudoku puzzles. The population size of [6, 7] was 21 and 11 respectively. On the other hand, our size was 150. Basically, we had to compare our algorithm with others using their population size. Therefore we cannot directly compare our results with their method. But, this is because that they are trying to speed up through small population size and improve efficiency through hybrid of GA with cultural algorithms [7]. On the other hand, we are thinking to improve efficiency through properly GA design and then speed up through implementations on GPU.

Figure 10 shows the relation of the average number of generations and dispersion. For puzzles that have the same number of initial givens, there is a dependence on the locations of the givens, and a large variance is seen in the mean number of generations needed to obtain the optimum solution. Furthermore, for difficult puzzles that provide few initial givens, there were cases in which a solution was not obtained even when the search termination point was set to 100,000 generations. The reason for that result is considered to be that the search scope for the solution to a difficult puzzle is large and there exist many high-scored local solutions that are far from the optimum solution. Another possibility is that there are puzzles for which the search scope is too broad and there is a dependence on the initial values.

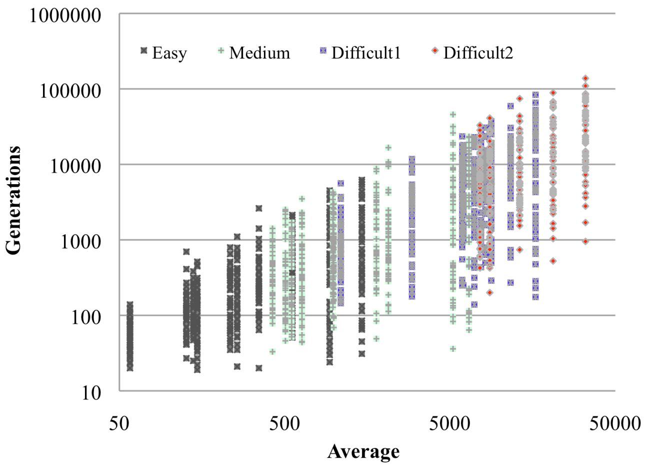


Fig. 10. The difficulty order of tested Sudoku. The minimum and maximum generations needed to solve each Sudoku from 100 test runs as a function of generations needed.

We did a comparative study of based on "Disparity Hypothesis and Gene Duplication [19]" using child individuals generated with a high mutation rate in addition to the child individuals generated with the low mutation rate used in the described above experiment as a way to solve such puzzles. Table 4 compares using the genetic operations proposed in section 3 with the method that adds correction based on disparity hypothesis. Table 4 presents the numbers of times the optimum solution is obtained in 100 test runs for the three puzzles shown in Fig. 8, which are ranked as "super difficult [17]".

Table. 4 Our result using the genetic operations proposed in section 3 and the method that adds correction based on disparity hypothesis. (100,000 trials, 100 test runs)
The numbers represents how many times out of 100 test runs each method reached the optimum with each puzzle.

Sudoku puzzles	Givens	mut+cross+LS		Added Disparity hypothesis	
		Count	Average	Count	Average
SD1	24	98	27171	98	25257
SD2	23	83	44170	90	40365
SD3	22	58	64259	62	62283

The solution rates increased and the average generation to find the solution reduced for all three of the puzzles, demonstrating the effectiveness of the proposed correction. Other ideas for adding a function for escaping local solution, or a function for increasing diversity in the search process function to the genetic operations proposed in section 3 can also be considered. Continuing research on escaping local solution is needed. Study of reducing the number of generations and parallel computing to reduce the actual processing time required to reach the optimum solution is also needed. At present, there exist more efficient algorithms [12-14] to solve Sudoku puzzles consist of a 9 x 9 cells, and faster than our method. However, in the results reported, these methods fail to solve some Sudoku puzzles they tested. Now, we are trying to speeding up our GA through implementations on GPU and the results may be reported later.

VI. 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 three puzzles that are ranked as super difficult and are difficult to solve by the conventional method. We also demonstrated that even higher accuracy can be expected by adding correction based on disparity hypothesis.

Acknowledgments

The authors would like to express their gratitude to Dr. Mikiko Sato of the Graduate School of Engineering, Tokyo University of Agriculture and Technology for her valuable support and comments on experiments. We also would like to thank to reviewers for their valuable comments for discussion.

References

[1] Wikipedia. Sudoku. Available via WWW: http://en.wikipedia.org/wiki/Sudoku (cited 8.3.2010).

[2] J. Holland, Adaptation in Natural and Artificial Systems, The MIT Press, Cambridge, MA, 1992.

[3] E.L. Lawler, J.K. Lentra, A.H.G. Rinnooy, and D.B. Shmoys (eds.), “The Traveling Salesman Problem – A Guided Tour of Combinational Optimization,” John Wiley & Sons, New York, 1985.

[4] A. Moraglio, J. Togelius, and S. Lucas, “Product Geometric Crossover for the Sudoku Puzzle,” in Proceedings of the IEEE Congress on Evolutionary Computation, July, 2006, pp. 470-476.

[5] T. Mantere and J. Koljonen, “Solving and Ranking Sudoku Puzzles with Genetic Algorithms,” in Proceedings of the 12th Finnish Artificial Conference STeP 2006, October, 2006, pp. 86-92.

[6] T. Mantere and J. Koljonen, “Solving, Rating and Generating Sudoku Puzzles with GA,” in Proceedings of the IEEE Congress on Evolutionary Computation, September, 2007, pp. 1382-1389.

[7] T. Mantere and J. Koljonen, “Analyzing and Solving Sudokus with Cultural Algorithms,” in Proceedings of the IEEE Congress on Evolutionary Computation, June, 2008, pp. 4054-4061.

[8] M. Nicolau and C. Ryan, “Genetic Operators and Sequencing in the GAuGE System,” in Proceedings of the IEEE Congress on Evolutionary Computation, July, 2006, pp. 5710-5717.

[9] D.E. Goldberg, Genetic algorithms in search optimization and machine learning, Reading, MA: Addison-Wesley, 1989.

[10] D.E. Goldberg, and K. Sastry, “A practical schema theorem for genetic algorithm design and tuning,” In Proceedings of the 2001 Genetic and Evolutionary Computation Conference, Morgan Kaufmann Publishers, pp. 328-335, 2001.

[11] R. Lewis, “Metaheuristics can Solve Sudoku Puzzles.” Journal of Heuristics, vol. 13 (4), 2007, pp. 387-401.

[12] H. Simonis, “Sudoku as a constrain problem,” B. Hnich, P. Prosser, B. Smith (eds.), in Proceedings of 4th Int. Workshop Modeling and Reformulating Constraint Satisfaction Problem, 2005, pp. 13-27.

[13] I. Lynce, J. Ouaknine, “Sudoku as a SAT problem,” In 9th Int. Symposium on Artificial Intelligence and Mathematics AIMATH’06, January, 2006.

[14] K. Moon, J. Gunther,” Multiple constrain satisfaction by belief propagation, “An example using Sudoku,” In 2006 IEEE Mountain Workshop on Adaptive and Learning Systems, July, 2006, pp. 122-126.

[15] M. Mitchell, Introduction to Genetic Algorithms, The MIT Press, Cambridge, MA, 1986.

[16] Number Place Plaza (eds.), “Number Place Best Selection 110,” vol. 15, ISBN-13: 978-4774752112, Cosmic mook, December, 2008.

[17] Super difficult Sudoku’s. Available via WWW: http://lipas.uwasa.fi/~timan/sudoku/EA_ht_2008.pdf#search='CT20A 6300%20Alternative%20Project%20work%202008' (cited 8.3.2010).

[18] A. Inkala, AI Sudoku 1002 Vaikeaa Tehtavaa, Pressman Oy, 2006.

[19] K. Wada, Y. Wada, H. Doi, S. Tanaka, and M. Furusawa, “Evolutionary System: Structures and Functions, Disparity Hypothesis and Gene Duplication,” in Proceedings of the IEEE International Conference on Evolutionary Computation, July, 1994, pp. 796-801.