# Parallelization of Genetic Operations that Takes Building-Block Linkage into Account

Yuji Sato*, Hazuki Inoue*, Mikiko Sato+

*Graduate School of Computer and Information Sciences, Hosei University,

3-7-2 Kajino-cho, Koganei-shi, Tokyo, 184-8584, Japan

Tel:+81-42-387-4533, Fax:+81-42-387-4560, E-mail: yuji@k.hosei.ac.jp

+The Graduate School of Engineering, Tokyo University of Agriculture and Technology,

2-24-16 Naka-cho, Koganei-shi, Tokyo, 184-8588, Japan

Tel:+81-42-388-7139, Fax:+81-42-388-7139, E-mail: mikiko@namikilab.tuat.ac.jp

**Abstract**- We propose a performance enhancement using parallelization of genetic operations that takes highly fit schemata (building-block) linkages into account. Previously, we used the problem of solving Sudoku puzzles to demonstrate the possibility of shortening processing times through the use of many-core processors for genetic computations. To increase accuracy, we proposed a genetic operation that takes building-block linkages into account. Here, in an evaluation using very difficult problems, we show that the proposed genetic operations are suited to fine-grained parallelization; processing performance increased by approximately 30% (four times) with fine-grained parallel processing of the proposed mutation and crossover methods on Intel Core i5 (NVIDIA GTX5800) compared with non-parallel processing on a CPU. Increasing GPU resources will diminish the conflicts with thread usage in coarse-grained parallelization of individuals and will enable faster processing.

**Key Words** - Genetic Algorithms, Linkage, Parallelization, Sudoku

## 1. INTRODUCTION

Sudoku pencil puzzles are a type of constraint satisfaction problem. There are several studies that have used genetic algorithms (GA) to solve Sudoku puzzles. However, the execution time and number of steps leading to the optimum solution for particularly the difficult problem is enormous [1]. The presumed cause is that there are many local solutions, and crossover, i.e., the main operation of GAs, is more likely to destroy some of the valid puzzle solutions. In many cases, the previous studies have tried to address the issue through the use of grammatical evolution [2] or the cultural algorithm [3]. Previously, we proposed genetic operations considering the linkage of loci and tried to improve the accuracy of the genetic operation [4]. As a result, we found that a solution could be arrived at in more than half the number of runs. However, Sudoku solving algorithms like the backtracking algorithm [5] can be executed faster than the proposed method. On the other hand, research has growing on using multi-core processors and GPUs as a means to speed up the GA [6-10]. Began to be popular in the general PC with multi-core processors and GPU, it became available relatively cheaply, was considered one of the reasons many examples of research.

In this study, we investigated the possibility of reducing processing time for genetic computations by parallelization of genetic operations that take building-block linkages into account. In section 2, we briefly describe the rules of Sudoku puzzles. In section 3, we show how the accuracy of Sudoku puzzle solution can be improved by using a genetic operation that takes building-block linkages into account. In section 4, we describe the method that parallelizes the proposed genetic operation. Section 5 describes a comparative evaluation of the solutions of a

difficult Sudoku puzzle executed on a CPU and on a many-core processor. Section 6 concludes this paper.

## 2. RULES OF SUDOKU

We shall explain the rules of Sudoku with the help of Fig. 1. Sudoku puzzles consist of a 9 x 9 matrix of square cells, some of which already contain a numeral from 1 to 9. 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. The arrangement of initially given numerals is called the starting point. The puzzle in Fig. 1 contains 24 non-symmetrically placed numbers at the starting point, and the goal is for the player to enter the correct number in the other 57 squares. The degree of difficulty varies with the number of given numerals and their placement. Basically, fewer numerals at the starting point means more combinations among which the solution must be found, so this raises the degree of difficulty. Moreover, there are 15 to 20 other factors that have an effect on the difficulty rating [11].

The above rules are good for basic puzzles, and methods such as back-tracking, which counts up all possible combinations in the solution, and the meta-heuristics approach [12] are effective for solving such puzzles. There are also certain algorithms that solve such Sudoku puzzles [13-15] faster than GAs can**.**

On the other hand, there are many variations of Sudoku. Some puzzles are of a 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. In particular, GAs and stochastic search methods are effective at solving 16 x 16 or 25 x 25 puzzles.

## 3. IMPROVING ACCURACY IN SUDOKU USING GENETIC OPERATION THAT TAKES LINKAGE INTO ACCOUNT

### 3.1 IDENTIFYING BUILDING-BLOCK LINKAGE THROUGH THE CONSTRAINTS

GAs are generally useful for solving problems with a wide search range of solutions. The GA implicitly manipulates a large number of highly fit schemata (building blocks) [16, 17] by mechanism of selection and recombination. References [1-3], for example, defines a one-dimensional chromosome that has a total length of 81 integer numbers and consists of linked sub-chromosomes for each 3 x 3 sub-block of the puzzle, and applies uniform crossover in which the crossover positions are limited to the links between sub-blocks. However, the conventional Sudoku applications require a huge number of generations to solve a hard problem with few initial placements [1]. The presumed causes are as follows. It is thought that crossover, the main operation of GAs, likely destroys building blocks and a proper growth and mixing of good building blocks were not achieved. Basically, most GAs employed in practice are unable to learn genetic linkage and the linkage learning genetic algorithm (LLGA) [18] was proposed to tackle the linkage problem with several specially designed mechanisms. LLGA is capable of learning genetic linkage in the evolutionary process, but it takes an extra time for linkage learning. To avoid that problem, we defined 9 x 9 two-dimensional arrays as the GA chromosome and devised

a genetic operation [4] that takes building-block linkages through the constraints of Sudoku.

### 3.2 SETTING INITIAL VALUES

The blanks of the puzzle are initially filled with random numbers. The numbers filling each of the 3 x 3 sub-blocks do not contain duplicates, i.e., each grid will run through the numbers from 1 to 9. By performing operations on each sub-block and continuing to meet the above criteria, the building blocks in sub-blocks will not be destroyed.

### 3.3 FITNESS FUNCTION

The evaluation value in Equation (1) is the sum of all evaluation values of rows and columns with no duplicate numbers. In addition, as indicated in equation (2) and (3), the numbers in each row and column are limited to 1-9. Therefore, the highest score of each row or column is 9, and hence, the maximum score of the fitness function $f(x)$ becomes 162.

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

$$g_i(x) = \left| x_i \right| \tag{2}$$

$$g_j(x) = \left| x_j \right| \tag{3}$$

### 3.4 CROSSOVER THAT PRESERVE BUILDING BLOCKS

The crossover operations emphasize averting destruction of building-blocks 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. 2. 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.

### 3.5 MUTATION

An example of mutation using the upper left sub-block of the puzzle is shown in Fig. 3. 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. Mutations are performed for each sub-block.

### 3.6 IMPROVED LOCAL SEARCH

We used the multiple offspring sampling (MOS) [19] method to generate the child individual. That is, we used a simple local search function in which multiple child candidates are generated when mutation occurs and the candidate that has the highest score is selected as the child.

## 4. PARALLELIZATION OF GENETIC OPERATION

There are various methods of parallelizing the GA on many-core processors [6-10]. In

particular, coarse-grained parallelization of individuals and has been found to be successful at speeding up GAs for Sudoku [10]. We also tried to implement parallel operations considering the linkage of genetic loci.

### 4.1 PARALLEL MUTATION

As shown in Fig. 4, to increase the parallel processing speed of mutation, nine threads were allocated to the processing of one individual. Two numbers were randomly chosen within a sub-block, and the swapping of the two within the nine region blocks was processed in parallel.

### 4.2 PARALLEL CROSSOVER

Crossover consists of six comparisons and six inheritance operations, as described in Section 3.1.4. As Fig. 5 shows, 3-parallel processing is done by performing the same thread row-wise and column-wise. Parallelization in one block thus includes the calculation of the evaluation value, comparison, and inheritance of the child individuals.

### 4.3 PARALLEL EVALUATION CALCULATION

The evaluation is done in the same thread as the genetic operations. As Fig. 6 shows, Sudoku puzzles are 9 x 9 matrixes, so the parallelization for this calculation is 9-parallel.

## 5. EVALUATION

### 5.1 EXPERIMENTAL METHOD

For the puzzles used to investigate the effectiveness of the genetic operations proposed in [4], we selected two puzzles from each level of difficulty in the puzzle set from a book [20]: 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. We also used the particularly super difficult Sudoku puzzles introduced in reference [21] for the parallel processing with the Intel Core i5 and NVIDIA GTX580. An example of the puzzles used in the experiment is shown in Fig. 7 and Fig. 8 respectively. We used tournament selection and the experimental parameters [4] are described in Table 1.

5.2 EXPERIMENTAL ACCURACY

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 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.

On the other hand, Fig. 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.

### 5.3 EXPERIMENTS ON INTEL CORE i5 AND OpenMP

#### 5.3.1 Specifications of computer

Table 2 lists the specifications of the PC used in these experiments.

#### 5.3.2 Results

Table 3 shows the relationship between the number of givens in the starting point and the execution time to reach the optimal solution. On the other hand, Figure 11 shows the relationship between the number of givens in the starting point and the execution time required to search the 10,000 generations. Both of the case, the tests were run 50 times and the averages of the results were compared.

From Table 3 and Fig. 11, we can see that the parallelized crossover was slower than no-parallelized crossover, whereas the parallelized evaluation calculation and mutation were faster. The mutation parallelization was more effective when there were more initial placements.

### 5.4 EXPERIMENTS ON GTX580 AND CUDA

#### 5.4.1 Specifications of computer

The parameters for the genetic algorithm were the same as those of the previous experiment, with the level of difficulty set at Super Difficult 3 [21]. Table 4 lists the specifications of the machine used for the experiment.

5.4.2 Results

Table 5 shows the fine-grained parallel processing results. Processing time equals the average generation number required to obtain the answer divided by the execution time (sec).

In addition, the degree of increase in processing performance is a value relative to the execution without parallel processing of individuals on a CPU.

5.5 DISCUSSION

In the accuracy experiments, the proposed method finds the optimum solution within 20,000 generations more than half of the time and had significantly improved accuracy compared with the conventional method that took 100,000 generations to solve the puzzle. However, the proposed method had a high initial value dependence and great variation in the number of generations needed to reach the optimum solution. In the performance experiments, parallelization of the evaluation calculation and the mutation led to a speed-up from 20 to 30% on the Intel Core i5 and approximately four times on the NVIDIA GTX5800. This is attributed to that each operation takes up a small ratio in the overall program and that creating threads and communication between threads costs too much. Another reason is that the performance assessment was done on an individual without parallel processing. We already implemented coarse-grained parallel genetic computing on the NVIDIA GTX 460, and showed that execution

acceleration factors of from 10 to 25 relative to execution of a C program on a CPU are attained [10]. The proposed method involves parallelization of the genetic operation, and it can be used in conjunction with parallelization of individuals. Consequently, the parallelization would multiply in their effect.

## 6. CONCLUSION

Parallelization of genetic operations based on genetic operations that consider effective building blocks can speed up Sudoku calculations. The parallelization described here resulted in a constant speedup of approximately 30% on an Intel Core i5 and four times on an NVIDIA GTX5800, compared with execution without parallel processing on a CPU. The parallelization of genetic operations would work even better in conjunction with parallelization of individuals since parallelization of genetic operations does not compete with parallelization of individuals.

## ACKNOWLEDGMENTS

This research is partly supported by the collaborative research program 2012, Information Initiative Center, Hokkaido University, and a grant from the Institute for Sustainability Research and Education of Hosei University 2012.

## REFERENCES

[1] Mantere, T., Koljonen, J., Solving and Rating Sudoku Puzzles with Genetic Algorithms, *In Proceedings of the 12th Finnish Articial Conference STeP 2006*, Espoo, Finland, October 26-27, 2006, pp.86–92.

[2] Nicolau, M., Ryan, C., Genetic Operators and Sequencing in the GAuGE System, *IEEE Congress on Evolutionary Computation 2006 (CEC 2006),* Vancouver, BC, July 16-21, 2006, pp. 1561–1568.

[3] Mantere, T., Koljnen, J., Solving and Analyzing Sudokus with Cultural Algorithms, *IEEE Congress on Evolutionary Computation 2008 (CEC 2008)*, Hong Kong, China, 1-6 June, 2008, pp.4053–4060.

[4] Sato, Y., Inoue, H., Solving Sudoku with Genetic Operations that Preserve Building Blocks, *IEEE Symposium on Computational Intelligence and Games (CIG) 2010*, Copenhagen, Denmark, Aug. 18-21, 2010, pp.23–29.

[5] Wikipedia, Backtracking, http://en.wikipedia.org/wiki/Backtracking (cited 1.11.2011).

[6] Byun, J.H., Datta, K., Ravindran, A., Mukherjee, A., Joshi, B., Performance analysis of coarse-grained parallel genetic algorithms on the multi-core sun UltraSPARC T1, *IEEE Southeastcon, SOUTHEASTCON'09*, March 5-8, 2009, pp. 301–306.

[7] Serrano, R., Tapia, J., Montiel, O., Sep'ulveda, R., Melin, P., High performance parallel programming of a GA using multi-core technology, *Soft Computing for Hybrid Intelligent Systems*, Studies in Coputational Intelligence, Vol. 154, Springer Berlin Heidelberg, 2008, pp. 307–314.

[8] Tsutsui, S., Fujimoto, N., Solving quadratic assignment problems by genetic algorithms with GPU computation: a case study, *In Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers (GECCO '09)*,

Montreal, Canada, July8-12, 2009, pp. 2523–2530.

[9] Munawar, A., Wahib, M., Munetomo, M., Akama, K., Theoretical and Empirical Analysis of a GPU Based Parallel Bayesian Optimization Algorithm, *In Proceedings of the International Conference on Parallel and Distributed Computing, Applications and Technologies* (PDCAT '09), IEEE, Hiroshima, Japan, December 8-11, 2009, pp. 457–462.

[10] Sato, Y., Hasegawa, N., Sato, M., Acceleration of Genetic Algorithms for Sudoku Solution on Many-core Processors. *In Proceedings of the 2011 ACM/SIGEVO GECCO Workshop on Computational Intelligence on Consumer Games and Graphics Hardware CIGPU-2011*, Dublin, Ireland, July 12-16, 2011, pp. 407–414.

[11] Wikipedia, Sudoku. http://en.wikipedia.org/wiki/Sudoku (cited 8.3.2010).

[12] Lewis, R., Metaheuristics can Solve Sudoku Puzzles, *Journal of Heuristics*, Vol. 13 (4), August, 2007, pp. 387–401.

[13] Simonis, H., Sudoku as a constrain problem, *In Proceedings of 4th International Workshop Modeling and Reformulating Constraint Satisfaction Problem*, 2005, pp. 13–27.

[14] Lynce, I., Ouaknine, J., Sudoku as a SAT problem, *In 9th International Symposium on Artificial Intelligence and Mathematics* (AIMATH'06), Fort Lauderdale, January 4-6, 2006.

[15] Moon, T.K., Gunther, J.H., Multiple constrain satisfaction by belief propagation: An example using Sudoku, *In 2006 IEEE Mountain Workshop on Adaptive and Learning Systems*, July 24-26, 2006, pp. 122–126.

[16] Goldberg, D. E., Genetic algorithms in search optimization and machine learning (1st ed.),

Boston, MA, USA, Addison-Wesley Longman Publishing Co., Inc., 1989.

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

[18] Harik, G. R., Goldberg, D. E., Learning linkage, *In Foundations of Genetic Algorithms* 4, 1996, pp. 247–262.

[19] LaTorre, A., Peña, J. M., Robles, V., De Miguel, P., Supercomputer Scheduling with Combined Evolutionary Techniques, *In Metaheuristics for Scheduling in Distributed Computing Environments, Studies in Computational Intelligence*, Vol. 146, Springer Berlin Heidelberg, 2008, pp. 95–120.

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

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

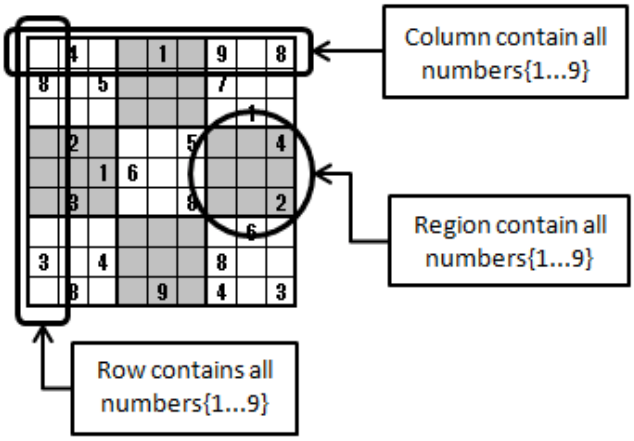


Fig.1 Example Sudoku puzzle.

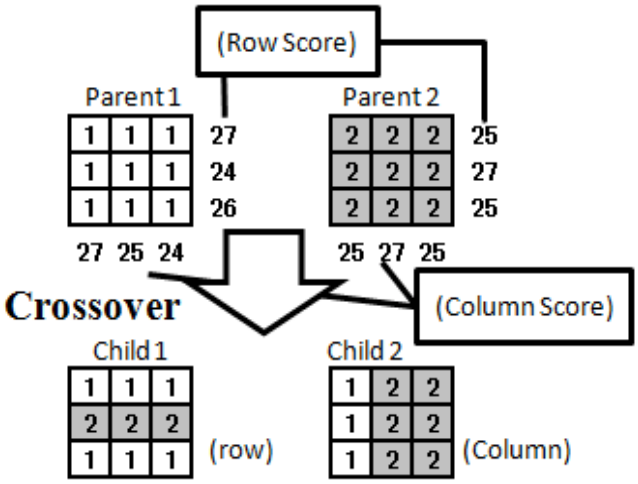


Fig.2 Example of crossover considered preserve building blocks

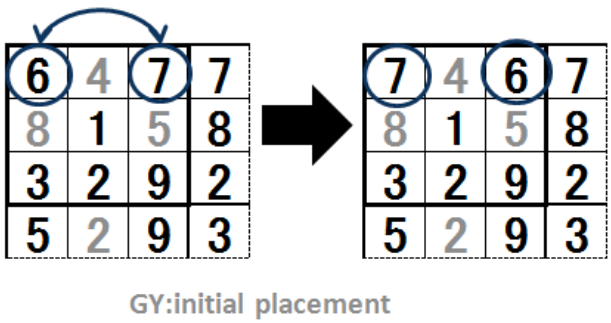


Fig.3 Example of mutation. Two numbers inside the sub block are selected randomly if the numbers are free to change puzzles.

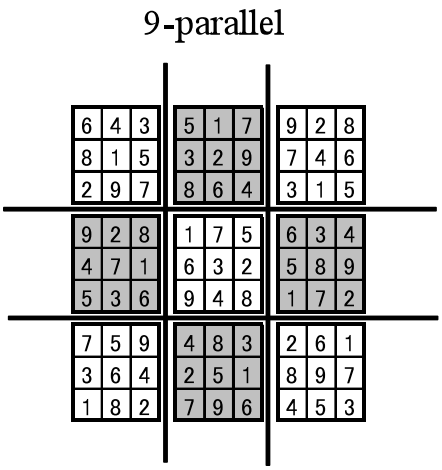


Fig.4 Parallelization of mutation

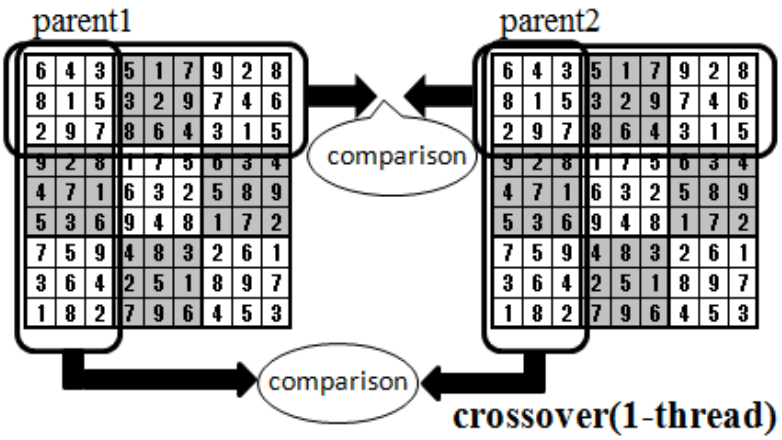


Fig.5 Parallelization of crossover

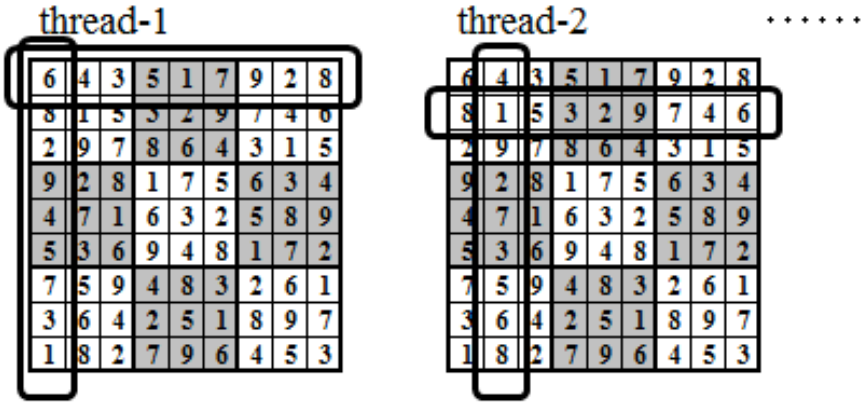


Fig.6 Parallelization of Evaluation Calculation

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 proposed genetic operations.

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 the particularly super difficult Sudoku puzzles.

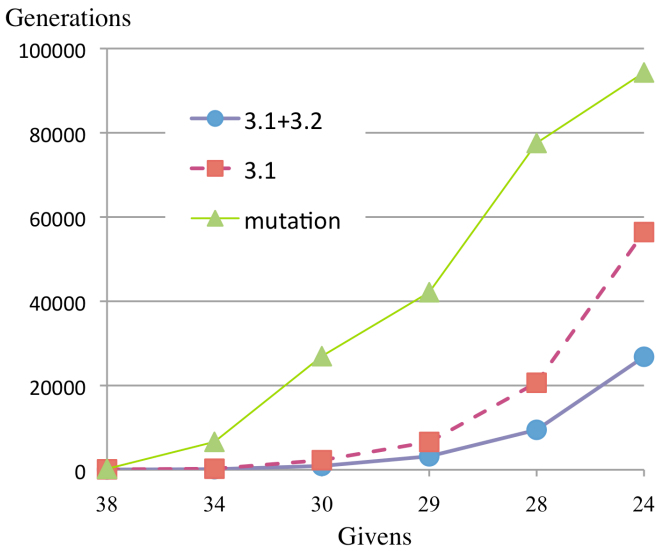


Fig.9 Relationship between number of numerals placed at the start (givens) and the average of each generation

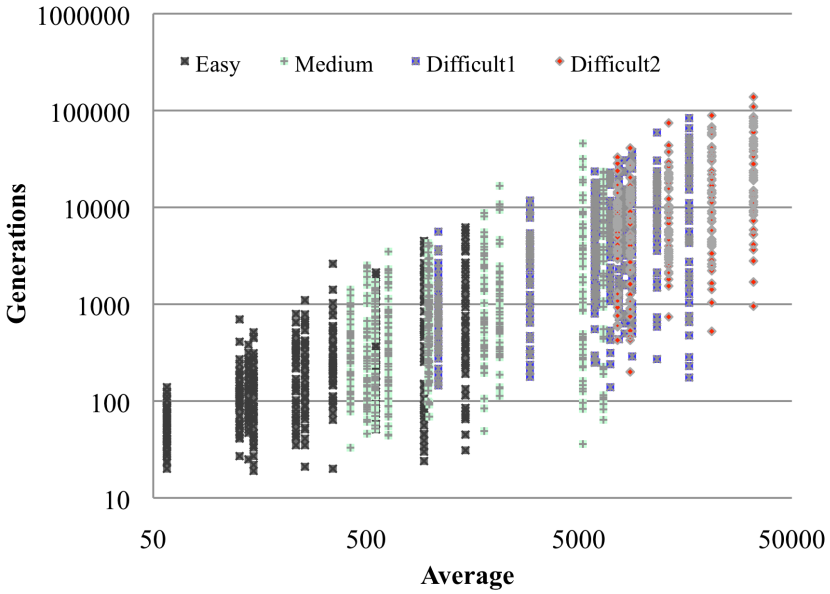


Fig.10 The difficulty order of tested Sudoku. The minimum and maximum generations needed to solve each Sudoku from 100 test runs as a function of generations needed.

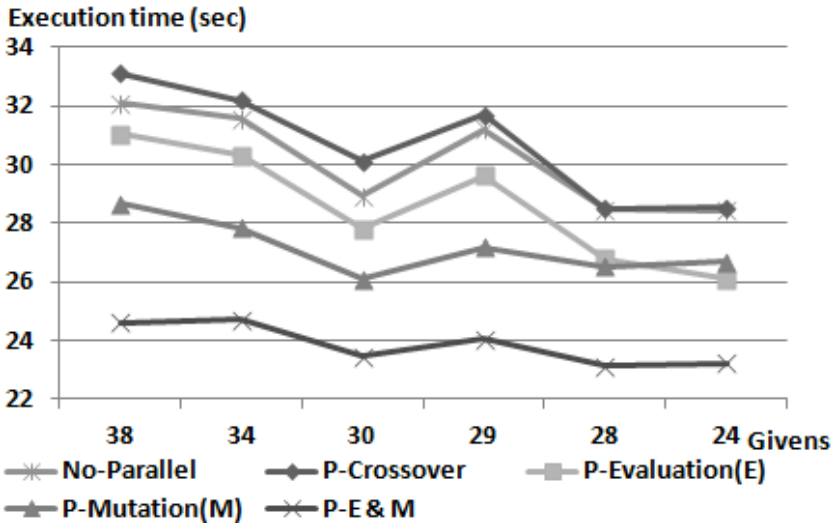


Fig.11 Relationship between givens and the execution time for 10,000 generations

Table 1 The experimental parameters of genetic operations

| Item | Parameter |
|---|---|
| Population size | 150 |
| Number of child candidates/Parents | 2 |
| Crossover rate | 0. 3 |
| Mutation rate | 0. 3 |
| Tournament size | 3 |

Table 2 Intel Core i5 execution environment

| | |
|---|---|
| OS | Microsoft Windows 7 Enterprise |
| Processor | Intel Core i5 M520 @ 2.40GHz |
| Core | 2/4 (logical / physical) |
| Memory | 4.00GB |

Table 3 Execution time to reach the optimal

| No. | 1 | 11 | 29 | 27 | 77 | 106 |
|---|---|---|---|---|---|---|
| Givens | 38 | 34 | 30 | 29 | 28 | 24 |
| Generations | 129 | 193 | 1169 | 1333 | 13786 | 14652 |
| Original[1)] | 0.50 | 0.72 | 4.17 | 4.84 | 44.63 | 47.00 |
| P-Crossover[2)] | 0.51 | 0.73 | 3.97 | 4.88 | 45.11 | 47.50 |
| P-Evaluation(E)[3)] | 0.49 | 0.67 | 3.51 | 4.37 | 39.95 | 41.74 |
| P-Mutation(M)[4)] | 0.42 | 0.62 | 3.49 | 4.22 | 40.21 | 42.97 |
| P-E&M[5)] | 0.36 | 0.53 | 3.03 | 3.58 | 34.46 | 36.93 |

1) Execution of the original method (not parallelized).
2) Execution of parallel crossover.
3) Execution of parallel evaluation calculation.
4) Execution of parallel mutation.
5) Execution of parallel evaluation calculation and mutation.

Table 4 GTX580 execution environment

| | |
|---|---|
| OS | Ubuntu 10.04 |
| CUDA | 4.0 |
| GPU | GeForce GTX580 |
| CPU | Phenom II X4 945 (3.0 GHz, 4 Core) |
| Memory | DDR2-800 4GB |

Table 5 Comparison of processing times in fine-grained parallel processing

| | Processing time (sec) | Processing performance (Generation/sec) | Increase in processing performance |
|---|---|---|---|
| No parallelon CPU[1)] | 42.6 | 1282 | 1.00 |
| No parallel on GPU[2)] | 17.1 | 3766 | 2.93 |
| Parallel on GPU[3)] | 14.3 | 4054 | 3.16 |
| Parallel on GPU[4)] | 10.7 | 5018 | 3.91 |

1) No parallel processing of individuals (CPU).
2) No parallel processing of individuals (GPU).
3) Parallel processing of crossovers (GPU).
4) Parallel processing of mutations and crossovers (GPU).