Acceleration Experiment of Genetic Computations for Sudoku Solution on Multi-core Processors

Mikiko Sato
Graduate School of Engineering
Tokyo University of Agriculture and Technology
Tokyo, Japan
mikiko@namikilab.tuat.ac.jp

Yuji Sato
Faculty of Computer and Information Sciences
Hosei University
Tokyo
Japan
yuji@k.hosei.ac.jp

Mitaro Namiki
Institute of Engineering
Tokyo University of Agriculture and Technology
Tokyo
Japan
namiki@cc.tuat.ac.jp

ABSTRACT

We focus on parallel-processing effect for Sudoku-solving and we show that diversifying initial values can reduce the Sudoku solution time. In an experiment using the commercially available Intel Corei7 multi-core processor, we show that a correct solution rate of 100% can be achieved with an average execution time of several tens of seconds even for super-difficult problems.

Categories and Subject Descriptors

J.0 [Computer Applications]: General

General Terms

Design, Experimentation, Performance, Verification

Keywords

Genetic Algorithms, Evolutionary Computation, Multi-core Processor, Parallel Computing

1. INTRODUCTION

Sudoku[5], a pencil and paper puzzle, is considered as a large combinatorial optimization problem. We have proposed genetic operations that consider effective building blocks for using genetic algorithms to solve Sudoku puzzles[3]. However the processing time was still very poor compared to the backtracking algorithm.

We aim to show that the parallelization of genetic computing in a many-core processor environment using a GPU or multi-core processor can be used for tackling practical problems in realistic times, even for problems for which the use of genetic computing has not been investigated previously because of the processing time problem. The use of multi-core processors has recently expanded to familiar environments like desktop PCs and laptop computers, and it has become easy to experiment with parallelization programs on multi-core processors through thread programming. Therefore, as one of the approaches to achieve our goal, we take the problem solving Sudoku puzzles and investigate acceleration of the processing by general-purpose thread programming conforming to POSIX specifications [1] on homogeneous multi-core processors.

GECCO'11, July 12–16, 2011, Dublin, Ireland.
ACM 978-1-4503-0690-4/11/07.

2. SUDOKU SOLUTIONS USING GENETIC OPERATION

To solve Sudoku problems by GA operations in realistic time, we have proposed a crossover operation that takes building-block linkage into account by defining 9×9 two-dimensional arrays as the GA chromosom[3]. Mutations are also performed for two numerals within a sub-block that are not given in the starting point and selected randomly and their positions are swapped. In addition to these operations, we have added only 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. These experiments use tournament selection. 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.

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

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

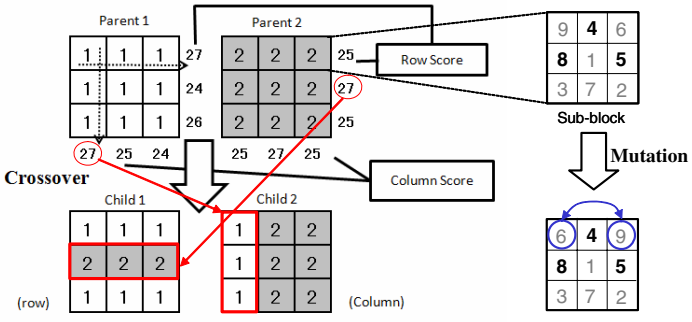

Figure 1: The GA operation that takes linkage into account

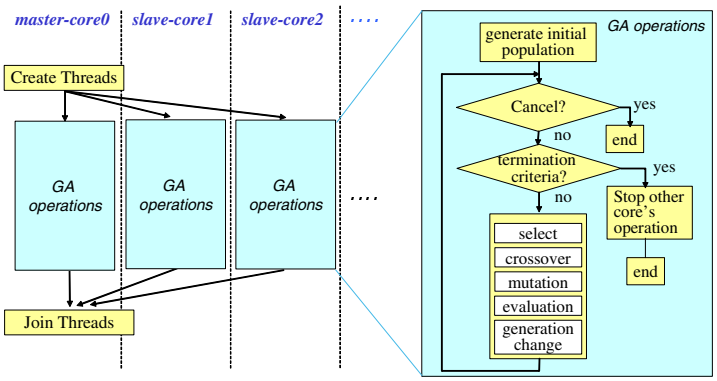

Figure 2: Parallel GA model for homogeneous multi-core processors

Table 1: The specifications of execution environment

CPU	Intel Corei7 920 (2.67GHz, 4cores)
OS	Ubuntu 10.04
C compiler	gcc 4.4.3 (optimization "-O3")

3. PARALLEL GA MODEL AND IMPLEMENTATION ON MULTI-CORE PROCESSORS

From the consideration of our previous study[3], a Sudoku solution using genetic computations is dependent on the initial value when a sufficient number of individuals cannot be set due to insufficient memory capacity or other constraints. With this in mind, we generate the same number of threads as cores in the target processor and propose a method that executes genetic operations with each core having a different initial value. We also adopt the value of the core that finds a Sudoku solution first. Fig. 2 shows this parallel GA model for multi-core processor.

The procedure of the parallel GA model at each core processor is as follows.

(a) All individuals are randomly generated on each core.

(b) The GA operation is repeated until the termination criteria are satisfied.

(c) The core that finds a Sudoku solution first cancels the operations on the other cores.

4. EVALUATION EXPERIMENTS

We varied the number of threads to be executed in parallel from 1 to 8 and evaluated (1) solution rate, (2) average number of generations until the correct solution was obtained, and (3) average execution time. The environment shown in Table 1 is used as an execution platform. We performed the experiment by executing a maximum of 8 threads in parallel using Intel hyper-threading technology[2].

We used the particularly difficult Sudoku puzzles SD1, SD2 and SD3 introduced by Timo Mantere in reference [4]. The results are presented in Tables 2. The values shown in the results are the averages for 100 experiments that were conducted with 150 individuals and a cut-off of 100,000 generations. Our experimental parameters are as follows: population size is 150, crossover rate is 0.3, mutation rate is 0.3 and tournament size is 3.

Table 2: The rate of correct answers, the number of average generations, and the average execution time

	Number of Threads	Count [%]	Average Generation	Average Exec. time
SD1	1	94	32858	22s 19
	2	100	15268	13s 87
	4	100	7694	13s 11
	8	100	3527	7s 39
SD2	1	82	42276	28s 41
	2	98	25580	22s 48
	4	100	13261	21s 47
	8	100	5992	12s 12
SD3	1	69	60157	39s 88
	2	93	46999	40s 43
	4	100	19982	30s 79
	8	100	8795	17s 13

From Table 2, we can see that increasing the number of threads reduces the execution time and increases the correct solution rate. In other words, the problem of initial value dependence tends to be eliminated as the number of threads is increased for both the processing time and the correct solution rate.

5. CONCLUSIONS

We proposed a parallel-processing effect for solving of Sudoku puzzles by genetic computation with the aim of using a many-core processor. Evaluation results showed that a correct solution rate of 100% can be achieved with an average execution time of several tens of seconds even for super-difficult problems on Intel Corei7, a commercially-available multi-core processor. In future research, we apply the proposed technique to problems of even larger scale and to experiment on many-core processors incorporating even more cores to shorten execution time.

6. REFERENCES

[1] IEEE. ISO/IEC 9945-1 ANSI/IEEE Std 1003.1, 1996.

[2] Intel Technology Journal(Hyper-Threading Technology). Hyper-threading technology architecture and microarchitecture. Available via WWW: http://download.intel.com/technology/itj/2002/volume06issue01/art01_hyper/vol6iss1_art01.pdf.

[3] Y. Sato and H. Inoue. Solving sudoku with genetic operations that preserve building blocks. In Proceedings of the IEEE COnference on Computational Intelligence in Game, pages 23–29, 2010.

[4] 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.

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