Solving Sudoku Puzzles with Node Based Coincidence Algorithm

Kiatsopon Waiyapara
Department of Compute Engineering,
Faculty of Engineering,
Chulalongkorn University,
Bangkok, Thailand
kiatsopon.w@gmail.com

Warin Wattanapornprom
Department of Compute Engineering,
Faculty of Engineering,
Chulalongkorn University,
Bangkok, Thailand
yongkrub@gmail.com

Prabhas Chongstitvatana
Department of Computer Engineering,
Faculty of Engineering,
Chulalongkorn University,
Bangkok, Thailand
prabhas@chula.ac.th

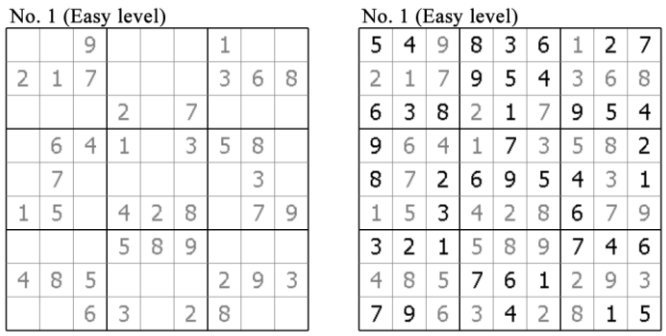
***Abstract*— In Evolutionary computation, Sudoku puzzles are categorized as hard combinatorial problems. It is almost impossible to solve these puzzles using only native operations of genetic algorithms. This article presents an application of Coincidence algorithm, which is an Estimation of distribution algorithms in the class of evolutionary computation that can outperform traditional algorithms on several combinatorial problems. It makes use of both positive and negative knowledge for solving problems. The proposed method is compared with the current best known method. It significantly outperforms problem-specific GA to solve easy, medium, and hard level of Sudoku puzzles.**

Keywords—Coincidence Algorithm; Sudoku Puzzle; Multimodal Optimization; Estimation of Distribution Algorithms; Genetic Algorithm;

I. INTRODUCTION

Sudoku [1], is a popular puzzle, considered as a hard combinatorial problem. There are various sizes of the Sudoku tables, but the size 9x9is more recognized standard. The problem determines the initial given numbers that suffice for only one global solution. An initial table of a size 9x9 puzzle and its result is illustrated in figure. 1. The purpose of the puzzle is to compose groups of permutation values in each row, column, and sub-block. There have been applications for solving the Sudoku derived from several group of algorithms such as binary integer linear programming (BILP) [2] for exact algorithm, as well as genetic algorithm (GA) [3][4][5] and estimation of distribution algorithm (EDA) for metaheuristics algorithms.

Compared to the exact algorithms, metaheuristics algorithms usually take longer time to solve the small combinatorial problems; however, for the larger size of problems, they can find good solutions with in polynomial time [6]. Implementations of GAs for combinatorial optimizations have some limitations. In particular, recombination operators can result in producing infeasible or invalid solutions [6]. A repair operator is required to fix these solutions. In addition, the algorithms provide poor maintenance of building blocks. The recombination operators cannot identify and compose the building blocks efficiently [3]. However, in recent years, there emerge new ways to find more competitive solutions called Estimation of Distribution Algorithms [7]. This class of algorithms makes use of probabilistic model for learning and generating solutions instead of using traditional recombination and mutation operators of GA. For combinatorial optimizations, the probabilistic model can identify building blocks and generate candidate feasible solutions that contain the building block efficiently. There have been several algorithms in this class such as Edge Histogram Based Sampling Algorithms (EHBSA) [8], Node Histogram Based Sampling Algorithms (NHBSA) [9], and Coincidence Algorithms (COIN) [10].



(a) The initial Sudoku problem (38 Givens)

(b) The result of the problem (38 Givens)

Fig. 1. Example of a sudoku puzzle and its solution.

COIN is an algorithm in the class of EDA. It uses a model called a joint probability matrix to identify and compose good building blocks to form the solutions. The special characteristic of COIN is that it does not only learn from the good solutions, but also, learns from the poor solutions in order to prevent the composition of bad building blocks found in the poor solutions. This method is called Negative correlation learning (NCL). The performances of COIN have been test in many applications including chess puzzles[11], path finding[10], production line sequencing [12] and production line balancing [10] for both single and multiple objectives. From preliminarily empirical studies[13], COINs have shown the capability to maintain building blocks found in diverse solutions which is a

fundamental capability to solve multimodal and multi-objective problems.

There has been the group of problems that contain different solutions of similar qualities, called multimodal problems [6]. Some of them can contain a lot of global optimum solutions. However, in some problems, the solutions found can have lots of global and local optimum solutions. EC cannot solve these problems easily because, sometimes, some of the solutions with high fitness values cannot guide the algorithm to find better or global optimum solutions. That is, in each run, the guide mechanism of EC mostly trapped in one or more local optimum. This problem can be found in Sudoku as well. The puzzles, especially at the difficult level, contain many near-optimal solutions that are not similar to the global optimum solution. These near-optimal solutions usually mislead the algorithms to get stuck at a local optimum in which local search operators cannot improve further. Figure 2 shows the example of three Sudoku solutions with high fitness scores equaled to 160 including (a),(b) and (c) and the optimal solution with the fitness score equaled to 162 (d). The darker numbers indicate the solution filled by the algorithms while the brighter numbers indicate the given initial numbers. The fitness scores are based on the function evaluation of GA [5].

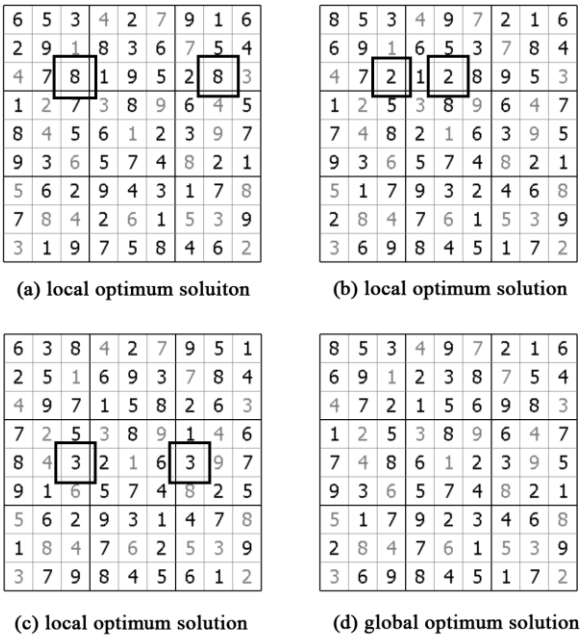


Fig. 2. Example of the three near-optimal solutions (a) ,(b), (c) and the only one global optimal solution (d)

COIN is in some way similar to EHBSA proposed by Shigeyoshi Tsutsui [9]. Both COIN and EHBSA learn to generate the population using the edge based probabilistic models. Such models provide flexible structures where the remaining of the generate sequences significantly depend on the previous generated sequences. Figure 3 shows that one edge based model can represent several solutions. The main difference of COIN and EHBSA is the learning method. EHBSA uses histogram to learn the building blocks from the populations which is an ad-hoc learning model while COIN uses the incremental learning model which allows the probabilistic model to learn from the negative samples. Such incremental learning model enables the algorithm to produce more diverse solution [14].

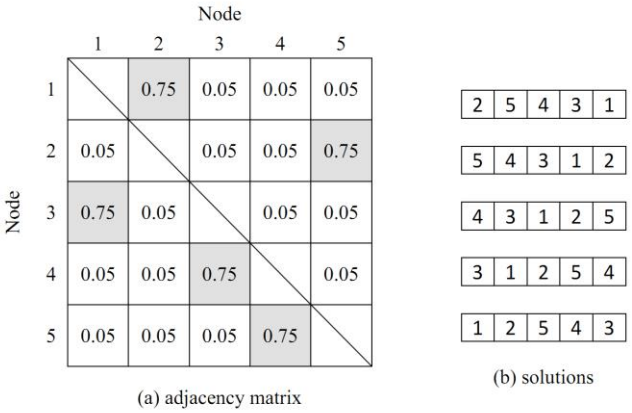


Fig. 3. Example of five solutions (on the right) generated from the probabilistic model of COIN.

However, the traditional edge based model of COIN is rather suitable for solving problems where the fitness scores depend on the relative sequences of elements. Consequently, the edge based model is not appropriate for solving Sudoku problems because the building blocks of Sudoku problems rather stick with the absolute positions. NHBSA [10] is another algorithm in the same family of EHBSA. The node based model is more appropriate for Sudoku problems.

This work presents a new algorithm called Node Based Coincidence Algorithm (NB-COIN) which adopts the node based representation from NHBSA and adopts the incremental learning method and negative correlation learning method from COIN in order to solve the Sudoku problems. The related algorithms are reviewed in section II. Section III presents the basic idea of COIN. Section IV introduces the adaptation of NB-COIN for solving Sudoku problems. Section V compares the performance of NB-COIN and GA [5]. Finally, Section VI concludes the work.

II. RELATED WORK

There are few evolutionary algorithms proposed for solving Sudoku puzzles. Including algorithms based on GA and EDA. These algorithms contain some problem specific encodings and operators that are useful to guide the search. Some of these ideas from the previous works can be applied to enhance the performance. The first implementation of GA for solving Sudoku was proposed by Timo Mantere and Janne Koljonen [3] in 2003. Each candidate solution is encoded as an integer string of size 81. The solution composes of 9 integer strings of sub-block concatenated in a sequence. The recombination operator was design to allow the exchange between sub-blocks of the parents. Consequently, the offspring are always feasible permutation strings in which the positions of the given numbers are all maintained. There are three mutation operators to apply for each sub-block including swap, 3-swap and insertion mutation as shown in Figure 4. These mutation

operators make use of an array to detect the illegal attempt of swapping, in order to maintain the entire initial given numbers. However, this work cannot solve the difficult Sudoku problems.

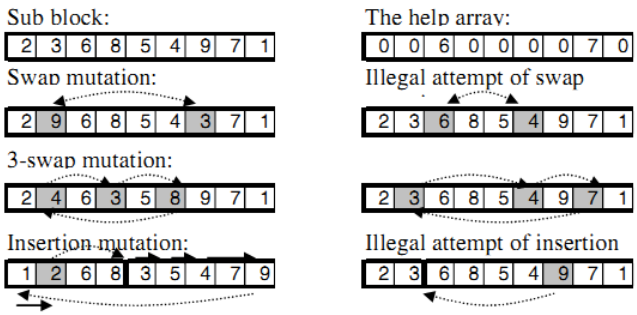


Fig. 4. The mutation technique using in proposed GA [3].

Later in 2007, the succeeding improvement of [3] is [4], a work proposed by the same group of researchers. The new mutation operators have been proposed. The result is significantly improved from the previous work.

Yuji Sato, and Hazuki Inoue published a better work based on GA in 2010 [5]. They proposed a new encoding scheme for solving the Sudoku problems where each solution is encoded as a 2-dimensional matrix of integer similar to the Sudoku patterns. A new fitness function is also proposed. The new fitness function (eq. 1) has the maximum value of 162. $g_i(x)$ and $h_i(x)$ represent the amount of different integer integers range from 1 to 9 in each row and column respectively. In other words, $g_i(x)$ refer to the amount of unique number in horizontal line as well as $h_i(x)$ refer to the amount of unique number in vertical line. In addition, $|\ .\ |$ represents the amount of unique number in each row or column [5]. That is, it examines the set and returns the number of unique digit. The problem specific recombination operator defines the cutting point on each vertical and horizontal border lines and on each sub-block. The mutation operator is a simple 2-swap operation. This work also applies a local search technique to improve the performance of the algorithm.

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

$$where\ g_i(x) = |x_i|\ and\ h_j(x) = |x_j| \quad (2)$$

Recently, there has been an algorithm designed for solving Sudoku based on EDA called restart estimation of distribution algorithm (RESEDA) [15]. The algorithm is originated by Sylvain Maire, and Cyril Prossette. The model can be described in term of $p_{i,j,k}^{(n)}$. The algorithm makes use of given initial numbers in order to reduce the search space by not allowing such numbers to be changed as those numbers already have been used. The update function of the probabilistic model is shown in eq. (3), where $0 \leq \alpha \leq 1$. $r_{i,j,k}^{(n+1)}$ is an empirical probability distribution [15] of the k numbers among some of the better candidate solutions found in the population. The parameter α represents the learning rate of the update function in which the larger value specifies the faster convergence of the process. This algorithm also determines the getting stuck conditions where the populations are trapped in a local optimum and then restarts the optimization process in order to jump out of the local traps.

$$p_{i,j,k}^{(n+1)} = \alpha p_{i,j,k}^{(n)} + (1-\alpha) r_{i,j,k}^{(n+1)} \quad (3)$$

III. BASIC IDEA OF COINCIDENCE ALGORITHM

The operational process of COIN [11] composes of several steps including (i) initializing the joint probability matrix, (ii) generating candidate solutions, (iii) evaluating the population for fitness value, (iv) selecting the good and the poor solutions from the population, and (v) updating the model. The flowchart of COIN is shown in Figure 5. The algorithm is terminated when a satisfactory solution is found or a maximum generation number is reached.

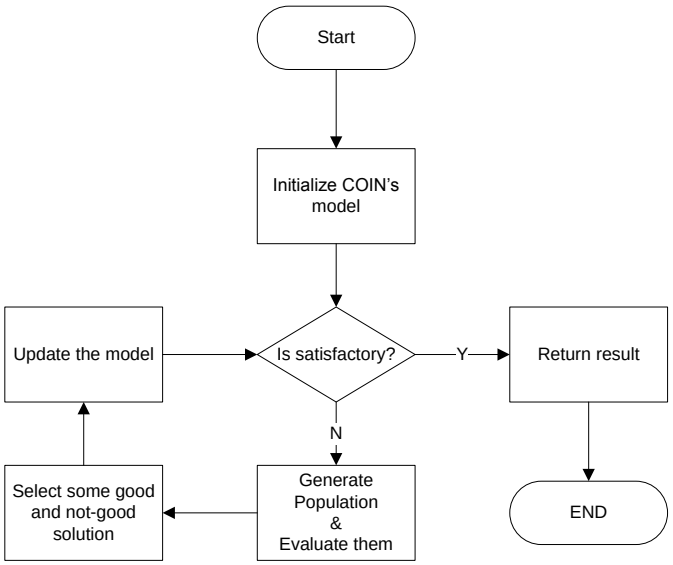


Fig. 5. Flowchart represented overall process of COIN.

$\alpha + \beta = \chi.$ (1) (1)

The initialization of the model is determined by $\frac{1}{n-1}$ in edge based representation and $\frac{1}{n}$ in node based representation. n denotes the problem size. Figure 6 shows the example of COIN with node based representation for a problem size 5. In node based COIN, the initial value of all possible numbers in the matrix is equaled to 0.2. The solutions are generated from this model. Each column denotes each position and each number in a column denotes the probability that it takes a particular value. Suppose the darker blocks designate the higher probability then one potential solution is 2-1-4-5-3.

After the initialization state, the candidate solutions are generated from the probabilistic model and then the population is evaluated. Both good and poor solutions from the population are selected for updating the matrix. The selected population size denoted by C%, is called the selection pressure.

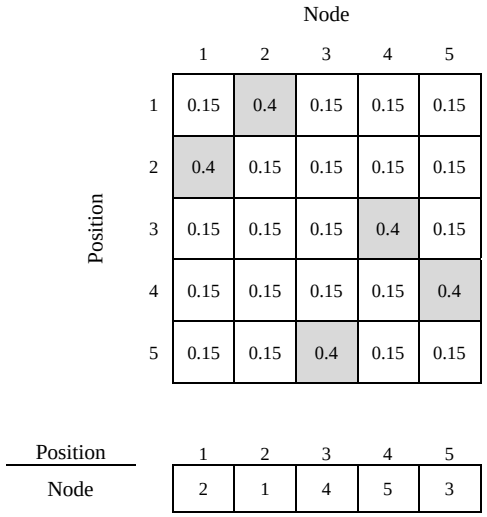


Fig. 6. The probabilistic model of NB-COIN based on a problem size equal to 5! and the potential solution generated from the model.

In each updating process, COIN uses two sub-populations, the better groups for rewarding and worse groups for punishment. The reward function is equation (4). The punishment function is equation (5). $M_{i,j}^{t}$ $(i,j = 0,1,\dots,L-1)$ denotes the variable of probabilistic model of COIN at the position *i*, and the node *j*. *k* denotes the learning rate, and *L* denotes the problem size. s_m^t stands for each candidate solution at g in the *t-th* generation. It composes of *L* permutation elements showing in $\{\pi_m^t(0), \pi_m^t(1), \dots, \pi_m^t(L-1)\}$. $\delta_{i,j}(s_m^t)$, called delta function which is a selector function defined in equation (6). Both equations (4) and (5) are used to increase and decrease the probability value as well as to control the sum of probability in each column (position) to always equal to 1.0. The overall processes are repeated until the number of generation is equaled to the maximum generation.

$$M_{i,j}^{t+1} = M_{i,j}^{t} + \frac{k}{L}\sum_g \delta_{i,j}(s_g^t) - \frac{k}{L^2}\sum_{q=0}^{L-1}\sum_g \delta_{i,q}(s_g^t) \quad (4)$$

$$M_{i,j}^{t+1} = M_{i,j}^{t} - \frac{k}{L}\sum_b \delta_{i,j}(s_b^t) + \frac{k}{L^2}\sum_{q=0}^{L-1}\sum_b \delta_{i,q}(s_b^t) \quad (5)$$

$$\delta_{i,j}(s_m^t) = \begin{cases} 1, & if\ \pi_m^t(i) = j \\ 0, & otherwise \end{cases} \quad (6)$$

IV. Representation of NB-COIN for solving Sudoku

NB-COIN for solving the Sudoku puzzle makes use of several ideas from the previous works. To represent the solutions, the node based matrix is suitable as Sudoku problem has some predefined positions (Fig. 7 a). The fitness function (eq. 1) is adopted from GA[5].

Each 9x9 solution is represented as a 3x3 sub-blocks, each sub-block is a 3x3 matrix (Fig. 7 b). NB-COIN represents one probabilistic model for each sub-block. Therefore, NB-COIN uses totally 9 probabilistic models, each of size 9x9 for each sub-block to evolve the solutions. Figure 7 (c) shows an example of the probabilistic model for the sub-block number 3. These 9 probabilistic models are co-evolved, that is, they are independently evolved, and at the same time, they are partially influenced by their neighborhood models.

The initial given numbers are used for search space reduction. The predetermined of positions in each sub-block constraints its neighbor sub-block. For example, the sub-block no.3 (Fig.7 b), the number 7 and 3 are predetermined, therefore the NB-COIN matrix (Fig.7 c) (the row is the possible value in the table, the column is the position in the table, numbering the top-left as the position 1, and continue to the right and down, the bottom-right is the position 9) in the row 7 and 3 are fixed to zeros. According to the top-left position of the sub-block no. 3 (Fig. 7 a), along the vertical direction (column), the neighbor sub-block has a number 4, so this constrains the sub-block no. 3 at the position 1, 2 and 3 not to be the number 4 (the probabilities of each number in the row number 4 are all zeros). The predetermined process is applied for all the sub-blocks on the left and the right, the top and the bottom sub-blocks. It is determined that the only possible number at the position 6 of the sub-block no. 3 is the number 4 (shown in black, with the possibility 1.0) (Fig. 7 c). All these initial given numbers constraint the matrix in such a way that they reduce the search space enormously.

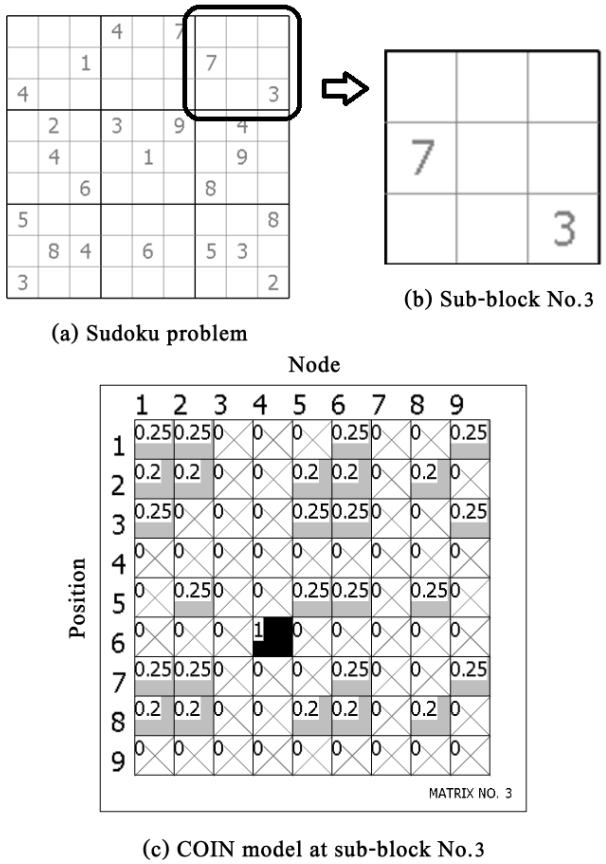


Fig. 7. Flowchart represented overall process of COIN.

V. EXPERIMENTAL RESULT

In the experiment, we compare the NB-COIN implemented in C++ with the previous work called GA preserving building blocks [5]. There are 6 benchmarks obtained from [3][4][5] which compose of three levels including easy, medium, and difficult. All of the parameters setting can be seen in the Table I. In order to compare the results, the population size and the number of maximum generation are set such that they are equal to the GA [5].

TABLE I. EXPERIMENTAL SETTING

Parameters	**NB-COIN**	**GA**
Population size	150	150
Selection pressure (*C*%)	25	2
Step size (*k*)	0.4	N/A
Upper bound	0.99	N/A
Maximum generation	100000	100000
Independent Run	100	100
Crossover rate	N/A	0.3
Mutation rate	N/A	0.3

Table II shows the experiment results. The column "Count" shows successful run in which the algorithm found the optimal solution out of 100 independent runs. The column "Nfe" shows the average number of function evaluation in term of number of generation used to find the best solution. The benchmarks (Fig. 1 and 8) compose of several levels from Sudoku problems such as easy (No.1), medium (No.27), and difficult (No. 106).

TABLE II. PERFORMANCE OF COIN AND GA FOR SOLVING SUDOKU

Difficulty rating	**Givens**	**NB-COIN**		**GA (mut+cross+LS)**	
		Count	***Nfe***	***Count***	***Nfe***
Easy (No. 1)	38	100	2	100	62
Easy (No. 11)	34	100	4	100	137
Medium (No. 27)	30	100	130	100	910
Medium (No. 29)	29	100	1196	100	3193
Difficult (No. 77)	28	100	2710	100	9482
Difficult (No. 106)	24	100	2341	96	26825

The result is shown in Table II, NB-COIN found the solution of Sudoku in every benchmark. But the result of GA in Difficult (No. 106) misses the optimal solution 4 times. Comparing the number of function evaluation, COIN uses several times fewer number of function evaluation in all benchmarks. It uses 3 times to 10 times less function evaluation than the competing algorithm.

VI. CONCLUSION

Though COIN is recently designed, it has been applied to a number of applications. The performances of COIN prove that it can work better than several traditional evolutionary algorithms. Particularly, COIN can maintain a large number of diverse solutions. Consequently, COIN is suitable for solving multimodal combinatorial optimization problems and thus is suitable for solving Sudoku problems which have several diverse near-optimal solutions. The experimental results demonstrate that node based implementation of COIN can outperform a problem specific GA.

No. 11 (Easy level)

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			

(a) The initial Sudoku problem (34 Givens)

No. 11 (Easy level)

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

(b) The result of the problem (34 Givens)

No. 27 (Medium level)

	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	

(a) The initial Sudoku problem (30 Givens)

No. 27 (Medium level)

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

(b) The result of the problem (30 Givens)

No. 106 (Difficult level)

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

(a) The initial Sudoku problem (24 Givens)

No. 106 (Difficult level)

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

(b) The result of the problem (24 Givens)

No. 29 (Medium level)

		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				

(a) The initial Sudoku problem (29 Givens)

No. 29 (Medium level)

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

(b) The result of the problem (29 Givens)

No. 77 (Difficult level)

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		

(a) The initial Sudoku problem (28 Givens)

No. 77 (Difficult level)

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

(b) The result of the problem (28 Givens)

Fig. 8. Benchmarks [3][4][5] and their Solutions.

REFERENCES

[1] Wikipedia. Sudoku. Available via WWW: https://en.wikipedia.org/wiki/Sudoku (cited 20.2.2012)

[2] A. Bartlett, and A. Langville, "An Integer Programming Model for the Sudoku Problem," Journal online of Mathematics and Applications, vol. 8, May 2008.

[3] T. Mantere, and J. Koljonen, "Solving and Rating Sudoku Puzzles with Genetic Algorithms," Finnish Artificial Intelligence Society FAIS, Espoo, Finland, pp. 86-92, October 2006.

[4] T, Mantere, and J. Koljonen, "Solving, Rating and Generating Sudoku Puzzles with GA," IEEE Congress on Evolutoinary Computation, pp. 1382-1389, September 2007.

[5] Y. Sato, and H. Inoue, "Solving Sudoku with Genetic Operations that Preserve Building Blocks," IEEE Conference on Computational Intelligence in Game. IEEE, pp. 23-29, August 2010.

[6] X. Yu, and M. Gen, "Introduction to Evolutionary Alogorithms," Springer, pp. 267, 2010.

[7] L. Bi, and S. Zhang, "Analysis and Research Models of the Estimation of Distribution Algorithms," International Conference on Computer Science and Network Technology, 2011.

[8] S. Tsutsui, M. Pelikan, and D. E. Goldberg, "Using Edge Histogram Models to Solve Permutation Problems with Probabilistic Model Building Genetic Algorithms, " IlliGAL Report No. 2004022, University of Illinois, 2003.

[9] S. Tsutsui, M. Pelikan, and D.E. Goldberg, "Node Histogram vs. Edge Histogram: A Comparison of PMBGAs in Permutation Domains," MEDAL Report No. 2006009, July 2006.

[10] W. Wattanapornprom, P. Olanviwitchai, P. Chutima, and P. Chongstitvatana, "Multi-objective Combinatorial Optimization with Coincidence Algorithm," IEEE Congress on Evolutionary Computation, Norway, May 18-21, 2009.

[11] R. Sirovetnukul, P. Chutima, W. Wattanapornprom, and P. Chongstitvatana, "The Effectiveness of Hybrid Negative Correlation Learning in Evolutionary Algorithm for Combinatorial Optimization Problems," IEEE Int. Conf. on Industrial Engineering and Engineering Management, Singapore, 6-9 Dec 2011.

[12] P. Chutima, and N. Kampirom, "A multi-objective coincidence memetic algorithm for a mixed-model U-line sequencing problem," International Journal of Advanced Operations Management, vol. 2, no. 3/4, pp. 201-48, 2010.

[13] K. Waiyapara, and P. Chongstitwattana, "Solving Multimodal Problems by Coincidence Algorithm," Int. Joint Conf. on Computer Science and Software Engineering (JCSSE), Thailand, pp. 45-48, 30 May -1 June 2012.

[14] Y. Liu., X. Yao., and T. Higuchi. "Evolutionary Ensembles with Negative Correlation Learning" IEEE Transaction on Evolutionary Computation. 4 (September 2000) : 380-387.

[15] S. Maire, and C. Prissette, "A Restarted Estimation of Distribution Algorithm for Solving Sudoku Puzzles," Statistics and Computing, 2012.