A Novel Multistage Genetic Algorithm Approach for Solving Sudoku Puzzle

Haradhan chel[1], Deepak Mylavarapu[2] and Deepak Sharma[2]
[1] Central Institute of Technology Kokrajhar,Kokrajhar, BTAD, Assam, India, PIN-783370
[2] Indian Institute of Technology Guwahati, Guwahati, Assam, India , PIN-781039
Email : h.chel@cit.ac.in, dsharma@iitg.ernet.in

Abstract*: Sudoku is a NP complete combinatorial number placement puzzle which has been solved using various algorithms including evolutionary algorithms. In this paper, we propose a multistage genetic algorithm (GA) for solving Sudoku. In this algorithm, the group table concept has been incorporated. This work progresses with a couple of cycles. In every cycle GA works for finding better solution. The each elements of the best solution in any particular cycle undergo through a multidirectional crosscheck validation process and finally selected subject to a probability. After each cycle, group table is updated depending on the chosen elements of the best solution in the previous cycle. This algorithm also comprises of new population generation, fitness assignment with more penalization, crossover and mutation operators etc. The results show that multistage GA is competitive with good successful rate for solving various Sudoku puzzles.

Keywords—Multistage Genetic Algorithm, Group Table, Crossover, Sudoku.

I. INTRODUCTION

Sudoku is a numbering puzzle in which integers from 1 – 9 are placed in a grid of 9-by-9 grid. Some rules are defined for placing integers such as, any particular integer from 1 – 9 should appear only once in a row, in a column, and also in a 3-by-3 sub grids. An example can be seen in Fig. 1. The difficulty or complexity of Sudoku is categorized as NP-Complete problem [1],[2]. Sudoku puzzle solving is also attributed with the number of minimum clues so that it should have a unique solution. Such Sudoku are considered as well posed puzzles. The literature suggests that minimum of 17 clues are required to have a unique solution [3] [4]. An example can be seen in Fig. 1. A Sudoku with less than 17 clues does not exist. Several methods have been suggested in the literature for solving Sudoku. The main aim of these methods is to solve Sudoku in less time, and requires less computation. Backtracking and brute force algorithms with some other techniques like naked cell, hidden cell, naked pair and hidden pair etc., are quite popular which are used in many applications on mobile phone and internet [5]. However, these methods involves several complex logics [6],[7],[3], and sometime require human interventions for reduction the search space. Using naked cell, hidden cell, naked pair, hidden pair [5] [3] techniques the search space size is drastically reduced. Later brute force backtracking algorithm can be used for all possible combinations. EA are also popular for solving Sudoku. Nicolau and Ryan [8] proposed genetic operators and sequencing for genetic algorithm using grammatical evolution (GAuGE) system such as slice and dice, column fill, row fill and raising numbers. The method was found better than standard GA.Moraglioet.al [9] proposed a geometric crossover method in which swapping was done among integer cells such that its geometric feature

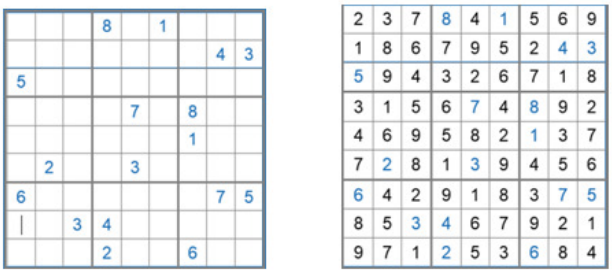

Fig. 1: A Sudoku with 17 clues and its unique solution.

remains unchanged. The problem was subjected to some hard constraints, few soft constraints, distance constraint, and feasibility on geometric crossover and geometric mutation operators. Mantere and Koljonen [10] proposed a mutation based genetic algorithm (GA) in which integer values within a grid were swapped [11]. But their efficiency was found to be poor. Cultural Algorithm (CA) [12] was also proposed in which a set of random integers were chosen from different blocks within a grid. These integers were then swapped randomly with integers from another grid. But, it showed slower convergence. Improved GA was then proposed [13][14] in which better selection, crossover and mutation strategies were discussed. Hybrid GA was proposed by Deng and Li [4] in which a solution was replaced by another solution, if it was preserved from last four generation. This strategy helped to maintain diversity in a population so that the algorithm should not trap in the local optima.GA was then coupled with particle swarm optimization (PSO) [15] at crossover level so that changes were made at the chromosomes. In mutation integers were swapped between cells within a sub-block. This method was faster and accurate than other GAs. Geem [16] proposed harmony search algorithm for solving Sudoku which mimic phenomena of musicians for their pitch correction when they play in a group. Deng. et.al. [17] introduced a crossover operator which had dual effects of self-experience and population experience. The self-experience was originated from two selected parental chromosomes, and the optimal chromosome in the current population represented the population experience. Result showed that the method was better than GA [14] and CA [12]. From the above studies it can be observed that Evolutionary Algorithms (EA) can be viable tool for solving Sudoku puzzle.

But, EAs need some structural changes so that EAs can perform well, and evolve feasible solution for Sudoku puzzle.

In this paper a new multistage technique is proposed which is different in several aspects as compared to other methods. It uses Sudoku group table for generation of population. In every stage the group table gets updated, and also reduces size of the population. It is found that in every cycle the fitness improves without being trapped into the local optima.

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

(a) Sudoku with 17 clues

[2,7]	[2,4,6]	1	[3,6,7,9]	8	5	[2,4,6,7,9]	[3,4,6,9]	[3,4,6,7]
[2,5,7,8]	[2,4,5,6]	3	[6,7,9]	[6,7,9]	1	[2,4,6,7,8,9]	[4,5,6,8,9]	[4,5,6,7,8]
[5,7,8]	[5,6]	9	4	[6,7]	2	[1,6,7,8]	[1,3,5,6,8]	[1,3,5,6,7,8]
6	7	[2,4]	[1,2,5,8,9]	[1,2,5,9]	[4,8,9]	3	[1,4,8,9]	[1,4,8]
[2,3]	[1,2,3,4]	8	[1,2,3,6,7,9]	[1,2,6,7,9]	[3,4,6,7,9]	5	[1,4,6,9]	[1,4,6]
9	[1,3,4,5]	4	[1,3,5,6,8]	[1,5,6]	[3,4,6,8]	[1,4,6,8]	7	2
1	[3,6,9]	5	[6,7,8,9]	4	[6,7,8,9]	[6,7,8]	2	[3,6,7,8]
4	8	[2,6,7]	[1,2,5,6,7]	[1,2,5,6,7]	[6,7]	[1,6,7]	[1,3,5,6]	9
	[2,7]	[2,6,9]	[2,6,7]	[1,2,5,6,7,8,9]	3	[6,7,8,9]	[1,4,6,7,8]	[1,4,5,6,8]

(b) Group table for Sudoku

Fig.2: Sudoku with clues and its group table.

This paper is organized in four sections. Section II discusses the proposed methodology. In this section, we discuss various operators of GA that include population generation, selection, fitness function, and crossover and mutation techniques. Section IIII presents results and discussion on various Sudoku puzzles. The paper is concluded in section IV.

II. PROPOSED METHODOLOGY

Solution procedure adopted in this paper for Sudoku puzzle is different than real parameter optimization because near optimal solution is actually infeasible. Moreover, penalty function approach cannot yield a feasible solution. Nevertheless, we need to choose a proper fitness function, and also devise selection, crossover and mutation operators for GA. In this paper we propose a multistage GA technique in which we prepare a group table from the given clues of Sudoku.

An example of Sudoku and its group table is shown in Fig. 2. It can be seen from Fig. 2(a) that empty top left side grid can take either 2 or 7, because other integers are already present either in its column or row, or in its 3-by-3 sub-grid. Similarly other possible integers for empty grids are identified and stored in the group table as shown in Fig. 2(b).

Algorithm 1 presents MGA in which group table is constructed to create an initial random population. This group table also gets updated in each cycle which is discussed later in section E. In the following sections, steps of algorithm 1 are discussed.

Algorithm 1: Multistage GA

Define: Fitness function, Crossover type, Mutation type, Group Table updating method.
Initialization: Parent Population size (μ), Offspring generation factor (k), number of cycle(C), Number of iterations over each cycle (N), crossover probability (p_c), mutation probability (p_m), group table updating parameters and Group Table (GT),
***Output*:** Desired solution
GT = Create group table.
While fitness>=0
While cycle<=C
1. If cycle>=1
   GT=update group table, discussed in section E.
2. Define: Pop_size=μ /cycle;
   Define: Offspring_size= Pop_size*k;
3. Generate population of size Pop_size
4. For each iteration ($i \leq N$) perform crossover, evaluate population, and choose best μ solutions.
5. Perform mutation over few good solutions and evaluate them.
6. Find the best solution for updating group table.
7. cycle=cycle+1.
8. **goto** step 1.

end while cycle
end while fitness

A) POPULATION GENERATION

Population is generated using group table in each cycle as presented in Step 3 of algorithm 1.The size of population and

its offspring are evaluated as shown in Step 2 of algorithms 1. The 9-by-9 grid is converted into one dimensional 1-by-81 grids. Thereafter, we assign integers cell by cell from group table. It can be seen from Fig. 2(a) that, the top left corner grid is empty, and possible integer is either 2 or 7 from group table shown in Fig. 2(b). We assign an integer from the group of {2,7} at random at the first position of one dimensional 1-by-81 grids. In case a grid is filled with a clue as shown in Fig. 2(a), we keep the same value of integer. This procedure is followed to fill 81 grids to generate one member of population. Similarly, we can generate a complete population from group table. The same has been shown pictorially in Fig. 3 by showing three members of a population.

Group Table	[2,7]	[2,4,6]	1	[3,6,7,9]	8	5	[2,4,6,7,9]	[3,4,6,9]	[3,4,6,7]	[2,5,7,8]
S1	2	4	1	9	8	5	7	4	6	5
S2	2	2	1	7	8	4	6	3	4	8
S3	7	4	1	3	8	5	4	9	3	2

Fig. 3: Generation of three member population from group table of Sudoku.S1, S2 and S3 stand for three members of initial population.

B) CROSSOVER OPERATOR

We adopt multipoint product geometric crossover operator [18] for performing crossover mentioned in Step 4 of algorithm 1. The main feature of geometric crossover operator is that the hamming distance between two parent solutions is same as hamming distance between offspring and two parent solutions. It is done by swapping integers of every grid by satisfying probability of crossover. If probability is satisfied,

Group Table	[2,7]	[2,4,6]	1	[3,6,7,9]	8	5	[2,4,6,7,9]	[3,4,6,9]	[3,4,6,7]	[2,5,7,8]
Probability	1	1	0	1	0	1	1	0	1	0
P1	2	4	1	6	8	5	9	3	7	7
P2	7	6	1	3	8	5	2	3	4	8
C1	7	6	1	3	8	5	2	3	4	7
C2	2	4	1	6	8	5	9	3	7	8

Fig. 4: Geometric crossover operator. P1 and P2 represent parent solutions, and C1 and C2 represent offspring solutions.

integers are swapped; otherwise we keep unchanged integer value in the grid. An example is shown in Fig. 4, where probability 1 indicates that it got satisfied.

C) FITNESS FUNCTION AND SELECTION

Fitness assigned in Improved GA [13] and Hybrid GA [4] methods consider repetitions of any integer over rows and columns. But we target fitness assignment by considering four cases which as follows; (i) repetition of integer over its row, (ii) repetition of integer over its column, (iii) repetition of integer over its 3-by-3 sub-block, and (iv) total number of any particular integer over the 9-by-9 grid should count 9. The fitness function is given as,

$$F = \sum_{j=1}^{9}\sum_{i=1}^{9} n_r^i(j) + \sum_{j=1}^{9}\sum_{i=1}^{9} n_c^i(j) + \sum_{j=1}^{9}\sum_{i=1}^{9} n_b^i(j) + \sum_{j=1}^{9} abs(N(j) - 9) - 27 \quad (1)$$

In the above formulation, first, second and third summation captures repetition over rows, columns and sub-blocks respectively. The fourth summation captures the amount how far that number deviates from 9. N stands for number of a particular integer$(j : 1 \leq j \leq 9)$ over the whole 9 x 9 sudoku. Equation (1) is used to calculate fitness of each solution mentioned in Step 4 of algorithm 1.

Selection presented in Step 4 of algorithm 1 is done by choosing the better members based on their fitness value from the combined population of parent and offspring. Here, μ and λ are the sizes of parent and offspring populations, respectively.

D) MUTATION

Mutation is similar to crossover operator. However, the swapping of integer for every grid is done with respect to group table. Swapping is done when mutation probability gets satisfied. We keep this probability low as suggested in the literature of GA for Step 5 of algorithm 1. The fitness is then evaluated for new solutions, and the best solution is chosen for updating group table.

E) UPDATE GROUP TABLE

Group table has been used for creating initial population, and for performing mutation. Therefore, we update group table after every cycle as mentioned in Step 1 of algorithm 1. The purpose of it is to fix integers in the grids of group table. For Sudoku puzzle, which is shown in Fig. 1, initial group table is shown in Fig. 2(b). The updated group table after 20 cycles is shown in Fig. 5. It can be observed that most grids have fixed integers as compared to initial group table.

1	7	5	6	3	4	9	8	2
8	2	4	7	1	9	3	6	5
[3,9]	9	6	[2,7,8]	[1,2,8]	5	1	[1,2,4]	[2,7]
2	[5,8]	9	1	5	6	7	3	4
6	3	[1,5]	[2,5]	4	7	[1,5]	9	8
4	[4,5]	[1,5,7]	[3,5]	[5,9]	8	2	[1,3]	6
5	6	2	9	8	1	4	7	3
7	1	3	4	6	2	8	5	9
[4,9]	[4,9]	8	[3,5]	7	3	6	2	1

Fig. 5: Group table after 20 cycles.

For updating group table we need initial group table, the best solution found in the last cycle, and group table of the last cycle. We introduce many search directions that can help in updating group tables. The obtained solution from the present iteration is used for updating the group table for next cycle. The best solution is passed through multidirectional validation process. It means all the cell elements of the best solution is

crosschecked for a invalid repetition. These search directions successive and it means the best solution is passed through validation of multidirectional search one after another. In this work, four different directions are chosen and they are (1) row-wise starting from left corner grid, (2) row-wise starting from right corner grid, (3) column-wise starting from top to bottom, (4) column-wise starting from bottom to top, and (5) diagonally as shown in Fig. 6.

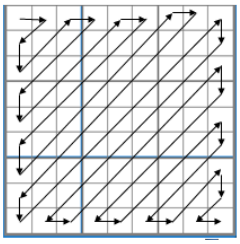

Fig. 6: Diagonal search direction

During the validation of the best solution and during any direction, if any value is found to be repeated, the value is ignored and that cell remains vacant and the algorithm proceeds to the next cell. In such a way during every search direction, different solutions are obtained which all are subsets of the last best solution. Finally after multidirectional validation, among different solutions, some cells out of 81 cells matches (clues in original Sudoku does not undergo for validation). If all the solutions are compared cell wise, for a particular cell where all values are found equal, it is accepted subject to a random variable. We generate a random variable between 0 and 1 with uniform probability density function and if the threshold is greater than that, the value is accepted. This is logical AND operation between i) the certainty of all the solution values of particular cell to be equal and ii) the occurrence of the random variable greater than the threshold. The value of that random threshold is chosen empirically and it is found that .3 is the best value for the threshold. Insertion of this randomness helps the algorithm to avoid the trapping in a local minimum during run due to a wrongly chosen value at any particular cell.

III. Result and Discussion

The difficulty level of a Sudoku puzzle is defined by the number of clues given and its unique solution. Sudoku having less clues but unique solution is considered as hard Sudoku. We consider Sudoku problem given in Fig. 1 in which 17 clues are given with unique solution, and it is solve using multistage GA. The parameters of GA are given in Table 1.

Table 1: GA parameters	
Population size	500
Offspring population size	500
Crossover probability	0.7
Mutation probability	0.4
Number of runs per cycle	15

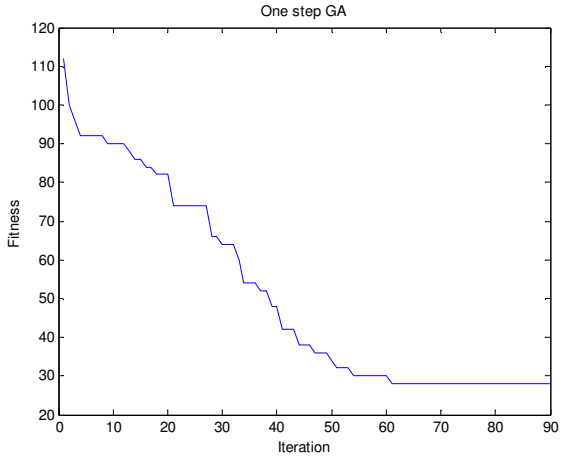

Fig. 7. Convergence plot of single-stage GA fro run

We solve the given Sudoku of difficult label using single stage and multistage GA. In single-stage GA, we consider one run per cycle. However in multistage GA, 15 runs per cycle is considered as shown in Table 1.Figure 7 shows convergence of single-stage GA for hard Sudoku puzzle given in Fig. 1. It can be observed that the single-stage GA gets stuck after 60 generations with very high fitness value. It is noted that the optimum solution is generated when the fitness becomes zero. The run shown in this figure corresponds to the best fitness achieved among 10 independent runs of single-stage GA.

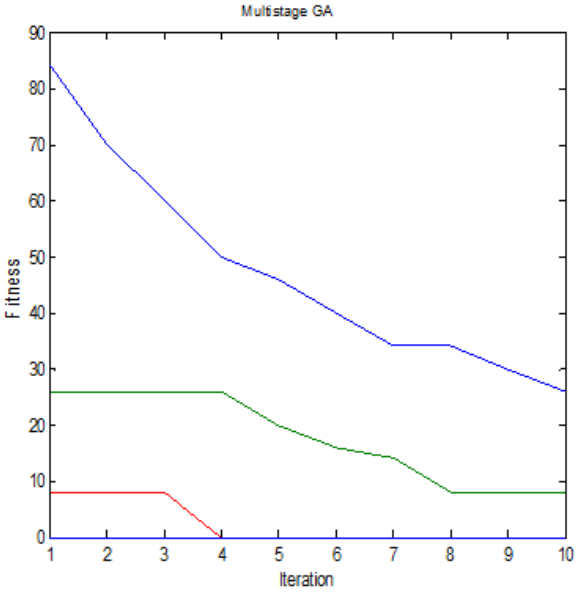

Fig. 8. Convergence plot of multistage GA for best run

The same Sudoku problem is solved using proposed multistage GA, and the convergence plot is shown in Fig. 8. Here, it can be observed that using MGA, the fitness curve gets converged to the optimal solution where the zero fitness. The solution is found in three cycle as described in

the run shown in this figure correspond to the best run among 10 independent runs of multistage GA.

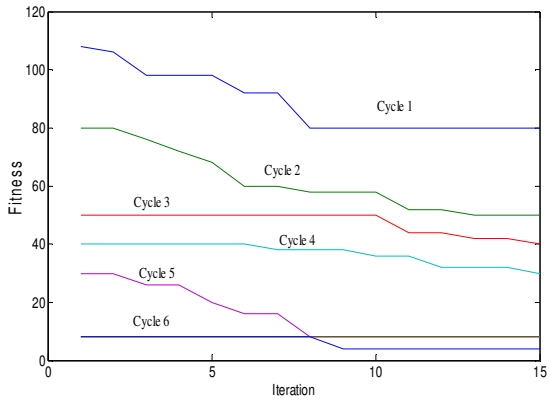

Fig. 9. Convergence plot of multistage GA for worst run

Fig. 9 shows the worst run among ten independent runs of MGA in which the algorithm is unable to converge to the optimal solution, but the solution is quite closer to the optimal solution. We found only one repetition of a number in the Sudoku solution which makes the fitness of four after six cycles. It can be concluded here that the proposed multistage GA can generate the optimal or near optimal solution for hard Sudoku puzzles. We test our algorithm on different Sudoku puzzles given in [8]. The puzzles are categorized in simple, easy, moderate, hard Sudoku's. We consider 10 different Sudoku for each category and run our algorithm on each Sudoku ten times. The parameters of GA given in Table 1 are considered. The obtained results using MGA are presented in Table 2. Here, the success rate is defined as number of runs generating the optimal solution vs. total number of runs of multistage GA. Average cycle is defined as the average number of cycles required to get the optimal or near optimal solutions over 10 runs. Average generation is defined by number of generations required per cycle to get the optimal or near optimal solution. Average fitness of unsuccessful events represents average fitness of multistage GA over 10 runs when it is found near optimal solution.

Table 2. Performance of multistage GA on various Sudoku puzzles

Difficulty	Success Rate (%)	Average cycle	Average Generation (rounded)	Average fitness of unsuccessful events
Simple	58	6.4	63	3.2
Easy	65	6.0	63	4.6
Moderate	48	8.2	8	4.8
Hard	12	16.6	166	6.7

It can be seen that the proposed MGA has good success rate for simple, easy and moderate puzzles with marginal average fitness of unsuccessful events. It is found that only one number gets repeated mostly in solved unsuccessful runs/event. It's performance for hard Sudoku puzzle is also competitive as the average fitness of unsuccessful runs is quite less. This indicates that when MGA cannot generate the optimal solution, it generates near optimal solution with repetition of one number in Sudoku puzzle

Table 3. Performance of multistage GA for different clue sizes of Sudoku (Out of total 15 runs)

Clue Size	15	17	19	21	23	25	27	29	31	33	35
Success Rate	15	15	14	14	12	14	14	15	15	15	15
Success percentage	100	100	93	93	80	93	93	100	100	100	100
Average cycle	5.2	5.3	10.4	12.6	18.5	10.4	10.8	8.2	5.6	5.6	3.0
Average generation (rounded)	55	55	142	128	188	108	102	85	52	55	32

Another experiment is done with multistage GA by creating Sudoku puzzles randomly from the solution of Sudoku. This procedure of generating Sudoku can generate any category of puzzle. The performance of multistage GA is shown in Table 3 for different clue sizes. We can observe from Table 3 that MGA shows good success rate for different clue size puzzles with less number of cycles required to get the optimal or near optimal solution.

IV. Conclusion

In this paper, we targeted solving all kinds of Sudoku puzzles which has both with unique solutions and multiple solutions. The results showed that multistage GA was quite successful and competitive in solving hard Sudoku puzzle. Average unsuccessful rate suggested that multistage GA was able to generate the optimal or near optimal solution for different categories of Sudoku puzzles. It was found that upgrading group table and multidirectional crosscheck validation were the main reason behind the success of this algorithm. A new type of technique of generation of population from group table is implemented. Fitness assignment scheme imposes more penalization, and crossover and mutation operators are carefully applied for designing a competitive algorithm. We realized that the algorithm can be further improved by incorporating some rules. These rules can improve the convergence of algorithm when near optimal solutions are generated.

REFERENCES

[1] Yato, T., and Seta, T.: Complexity and Completeness of FindingAnother Solution and its Application to Puzzles. IEICE Transactions onFundamentals of Electronics, Communications and Computer Sciences,Vol. **86**, No. 5. (2003).

[2] Asif, M.,Baig, R.:Solving NP-Complete Problem Using ACO Algorithm. International Conference on Emerging Technologies, pp-13-16(2009)

[3] McGuire, G.,Tugemanny, B.,Civarioz,G.: There is no 16-Clue Sudoku: Solving the Sudoku Minimum Number of Clues Problem via Hitting Set Enumeration. arXiv:1201.0749v2 [cs.DS] 1 Sep 2013.

[4] Deng, X. Q., Li, Y.D.: A novel hybrid genetic algorithm for solving Sudoku Puzzles. Optimization Letters**7**, pp. 241–257(2013).

[5] http://www.sudoku-solutions.com/index.php?page= solvingNakedSubsets #nakedSingle, dated: 20 December 2015.

[6] Li, H.: Algorithm and implementation for Sudoku puzzle based on graph search algorithm. J. TonghuaNomal Technol. **30**(10), 43–45 (2009)

[7] htttp://en.wikipedia.org/wiki/Mathematics_of_Sudoku

[8] Nicolau, M., Ryan, C.: Genetic operators and sequencing in the GAuGEsystem. In: IEEE Congress on Evolutionary Computation, pp. 1561–1568 (2006)

[9] Moraglio, A., Togelius, J.: Geometric Particle Swarm Optimization for the Sudoku Puzzle. In: Geneticand Evolutionary Computation Conference London, pp. 118–125 (2007)

[10] Mantere,T and Koljonen, J., Solving and Rating Sudoku Puzzles withGenetic Algorithms.In :New Developments in Artificial Intelligence and the Semantic Web Proceedings of the 12th Finnish Artificial Intelligence Conference, 2006, pp. 86-92.

[11] Mantere, T., Koljonen, J.: Solving, Rating and Generating Sudoku Puzzles with GA. In: 2007 IEEE Congresson Evolutionary Computation, pp. 1382–1389 (2007)

[12] Mantere, T., Koljonen, J.: Solving and Analyzing Sudoku with Cultural Algorithms. In: 2008 IEEE Congress on Evolutionary Computation, pp. (2008) pp. 4053-4060.

[13] Li, Y.D., Deng, X.Q.: Solving Sudoku puzzles base on improved genetic algorithm. Computer Applications and Softwares,issue- 3,pp-68–70 (2011)

[14] Deng, D., Li, Y.,Cai, R.: Solving Sudoku with New Genetic Algorithm, In: 2012 International conference of Artificial Intelligence and Soft computing, Lecture Notes of Information Technology, Vol 12

[15] Kennedy, J., Eberhart, R.: Particle Swarm Optimization. In: proceedings of IEEE International Conference on Neural Networks. Perth, pp. 1942–1948 (1995)..

[16] Geem, Z.W.: Harmony search algorithm for solving Sudoku. Lecture Notes in Computational Science,**4692**,pp. 371–378 (2007

[17] Deng, X., Li X., Li, G.:Research on Sudoku Puzzles Based on Metaheuristics Algorithm.Journal of Modern Mathematics Frontier.**2(1)**, pp. 2013, pp. 25-32 .

[18] Moraglio, A., Togelius, J., Lucas, S.: Product geometric crossover for the Sudoku puzzle. In: 2006 IEEE Congress on Evolutionary Computation, pp. 470–476 (2006)