Solving Sudoku Puzzle by Evolutionary Algorithm

Kedar Nath Das[1], Sumit Bhatia[2], Shubin Puri[3], Kusum Deep[4]
[1]kedar.iitr@gmail.com, [2]sumit@cse.psu.edu, [3]shubhin@gmail.com, [4]kusumdeep@gmail.com

[1]Department of Mathematics, NIT Silchar, Assam, India
[2]Department of Computer Science and Engineering, The Pennsylvania State University, USA
[3]Department of Chemical Engineering, [4]Department of Mathematics, IIT Roorkee, India

Abstract: *'**Sudoku**' means '**Single number**'. 'Sudoku Puzzle' is an interesting and popular Japanese game, where the non-givens need to be filled by a single number (from 1 to 9) provided no repetition occur in the corresponding rows, columns and sub-squares. Solving a Sudoku puzzle is challenging due to its easy rules and difficult inherent phenomenon. Although a number of approaches exist for solving a given Sudoku puzzle, it becomes a challenge among the researchers to solve it by using evolutionary algorithms. In this paper, a Retrievable Genetic Algorithm is proposed to solve a given Sudoku puzzle. A new fitness function is designed with puzzle-character-dependent constraints. The Genetic Algorithm is made "Retrievable", since the population is reinitialized after a certain number of generations in order to escape from the premature convergence or from being trapped in the local minima. A set of 9 sample puzzles of different difficulty label have been considered for comparison. The superiority of Ret-GA is ensured from the comparative results and discussions.*

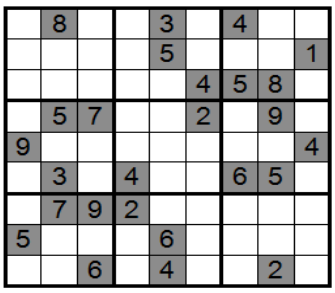
1. Introduction

'Sudoku Puzzle' is a number game pioneered by the great Swiss mathematician Leonhard Euler in the year 1783. The word 'Sudoku' comes from Japan and consists of the Japanese characters Su (meaning 'number') and Doku (meaning 'single'). Today, it is one of the most popular puzzles attracting young and old alike. Due to its addictive and challenging nature, it has spread like a wildfire throughout the globe and it has attracted the attention of many researchers who are trying to design algorithms to solve it by applying varied approaches. The traditional method (deterministic approaches) became popular to solve such puzzles. Recently researchers put their efforts to design robust heuristic approaches like simulated annealing [1] and Genetic Algorithms [2], hybrid Genetic Algorithm [3], geometry particle swam optimization [4] to solve Sudoku puzzles. Haynes et al. [5] defined a new mutation technique called 'exponential moving average' to solve Sudoku puzzles.

Sudoku is a 9 X 9 square that is divided into nine, 3 X 3 sub squares. In the beginning, there are some static numbers (called givens) in the puzzle. The game is to fill all non-givens such that each row, column and sub square contains each integer from 1 to 9 once and only once. The difficulty level of the Sudoku puzzle is determined not only by the number of givens [6], but also it dependents on about 20 factors [7].

Figure 1 is an example of a Sudoku puzzle and Figure 2 represents its solution. The static numbers given in Fig. 1 retain their positions and values in the solution. In the solution, each row, column and sub square of solution (Figure 2) contains integers from 1 to 9 once and only once. The Sudoku can be modeled as a combinatorial optimization problem. It is NP-hard since the total number of unique 9 X 9 Sudoko that can be generated are 6,670,903,752,021,072,936,960 (~6.67×1021) [8], where each has a unique solutions.

The objective of this paper is to present a new Retrievable Genetic Algorithm based on a new model of the fitness function for solving a Sudoku puzzle. Sudoku of varied difficulty levels are solved. The results are compared with the results given in [2].



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

Fig. 1 A Sudoku puzzle

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

Fig. 2 Solution

This paper is organized as follows. In section 2, a review of literature on solving the Sudoku using Genetic Algorithms is presented. In section 3, the proposed Retrievable Genetic Algorithm is stated. In section 4, the numerical results on various difficulty levels of the puzzle are discussed and analyzed. Finally, the conclusions are drawn in section 5.

2. Literature Review

Genetic algorithm (GA) has been one of the population based paradigms pioneered by John Holland in 1975. Based on genetic process of biological organisms, GA works surprising well in determining the global (near) optimal solution. GA has been successfully used in solving combinatorial problems. Sudoku puzzle is combinatorial optimization problem [9]. It is similar to the ancient magic square problem (Latin square), where different sizes of squares must be filled, so that the sum of each column and each row are equal. This magic square problem has been solved by GAs [10, 11]. Generating threshold matrix for halftoning [12] grayscale images is also a related problem. Threshold matrices have been optimized by GAs as in [11, 13-16]. In [17] GA is used to generate Sudokus internally. It is claimed that the generated Sudokus are very hard to solve by their GA. Unfortunately there is no details available saying how the GA works internally. Moraglio et. al. [18] designed a product Geometric Crossover incorporating the distance of the search space treated as metric space, to solve Sudoku puzzles and concluded that on Medium and Hard problems, the new geometric crossovers perform significantly better than hill-climbers and mutation alone. Timo Mantere and Janne Koljonen [16] worked on solving, rating and generating Sudoku puzzles.

Mantere and Koljonen [2] proposed a method to generate Sudoku puzzles and to solve a given puzzle. In their paper, Sudoku is treated as a constrained satisfaction problem. Mainly, there are three constrains as follows.

i. The sum of each row/column /sub-square entries must be 45.
ii. The product of each row/column/sub-square entries must be 9!.
iii. No entries should be repeated in each row / column / sub-square.

They modified the constrained optimization problem into an unconstrained one and used GA to solve it. Their proposed the objective function is to Minimize

$$f(x)=10*\left(\sum_i g_{i1}(x)+\sum_j g_{j1}(x)\right)+\left(\sum_i \sqrt{g_{i2}(x)}+\sum_j \sqrt{g_{j2}(x)}\right)+50*\left(\sum_i g_{i3}(x)+\sum_j g_{j3}(x)\right) \quad (1)$$

where, for row-wise operation i,

$$g_{i1}(x)=\left|45-\sum_{j=1}^{9} x_{i,j}\right|, \quad g_{i2}(x)=\left|9!-\prod_{j=1}^{9} x_{i,j}\right| \text{ and } g_{i3}(x)=\left|(\{1,2,3,4,5,6,7,8,9\}-x_i)\right|$$

Similarly, for column-wise operation j, the terms $g_{j1}(x)$, $g_{j2}(x)$ and $g_{j3}(x)$ are defined. It is worth to note here that the three terms present in the right of equation (1) involves the constraints mentioned above in (i), (ii) and (iii) respectively. The first term of equation (1) requires that each row and column sum should be equal to 45. The second term requires that each row and column product should be $9!$. The third term requires that each row and column must contain each integer from 1 to 9 exactly once. There all leads to minimize $f(x)$.

The static numbers (givens) are not changed throughout the entire computation. Authors claimed that their software can generate the Sudoku puzzles of different difficulty levels and can solve the puzzles up to some extent. The results reported by the authors clearly indicate the potential of Genetic Algorithms to solve Sudoku puzzles. However, their algorithm works very well for puzzles of lower difficulty level but its efficiency decreases rapidly with increasing difficulty level. Further, it was reported that this fitness function may not be the best for Sudoku puzzles.

In the present work, we have used an entirely new fitness function incorporating puzzle-character-dependent constraints. Further, as evolution often proceeds in *punctuated equilibrium*, the fitness value ceased to improve after certain number of generations. Also, it was observed that all the chromosomes tend to converge to a point very close to the desired solution. It may be possible that after a very large number of generations we may obtain the exact solution but in order to reduce computational time we introduced a random restart mechanism where after a certain number of generations (which depends on the difficulty level of the puzzle) the population is again reinitialized.

3. The proposed Retrievable Genetic Algorithm

In this section a new GA algorithm is proposed and is called as "Retrievable GA (Ret-GA)". The motivation behind this proposal is, in Mantere and Koljonen [2] there are mainly three instances those need to be improved. First, while solving Sudoku game, authors started with an initial population, where they take care of non-repetition of the numbers in each sub-square. But this mechanism hinders the randomness concept of GA mechanism. Thus, the initial population needs to be generated randomly to reduce the time. Of course, there may be some repetitions initially (refer the next paragraph). Secondly, as the crossover operator is being used with probability 1 in the population, it may not able to maintain the diversity in the population and the number of function evaluations increases. Therefore, in the proposed method, few individuals are not allowed to participate in the crossover process. This can be controlled with considering a high probability of crossover (refer the 'crossover' section). Thirdly, while using the mutation operator, authors use the swap mutation, 3-swap mutation and insertion mutation with a probability ratio of 0.5:0.3:0.2. It kills time but in return, it has no much effect in the solution quality. To overcome this, a bit-wise mutation is being applied with a small probability (refer the 'mutation' section).

Initial Population and Selection

The algorithm proceeds with the generation of initial population which consists of 10*N individuals, where N is the order of the given Sudoku puzzle. The population size 10*N is recommended after observing the performance of Ret-GA by varying it value in the range 2*N to 30*N. Each individual is an N X N array. The entries corresponding to the non-givens in the Sudoku puzzle are assigned randomly generated values from 1 to N. This step is important because GA yields robust optimal solution due to its randomness characteristics and it approaches the solution by excluding the unfit individuals. Hence, the random characteristic is being utilized starting from the initial population itself unlike [2]. As a result the proposed algorithm randomly explores the larger search space and selects the better individuals for each consecutive generation.

Crossover

Select 2 conjugative individuals from the population with a crossover probability 0.8 (experimentally verified). Apply Uniform Crossover to them. The mechanism for such Crossover is to choose the crossover sites as much as possible, in between each pair of rows. Now interchange the rows alternatively throughout. In Figure 3, an example of crossover is given between two individuals. The dotted lines represent the possible crossover sites. The bi-headed arrows indicate the rows to interchange, which need to be performed alternatively.

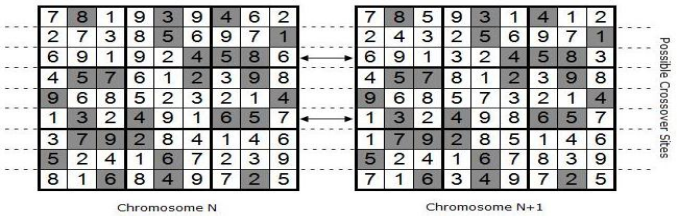


Fig. 3 Row-wise Uniform Crossover.

Mutation

In mutation, a new approach is followed where the non-givens flip their positions bit wise. Each of the non-given entry is replaced by a randomly generated number (from 1 to N) with a probability of 0.2, which has been fine-tuned and recommended after an extensive experimentation. For example, in Figure 4, to mutate the position '9', we need to discard it first and then will be filled by a number randomly generated from 1 to 9. It may be noted that just after the mutation few repeated entries may be appeared in the corresponding row.

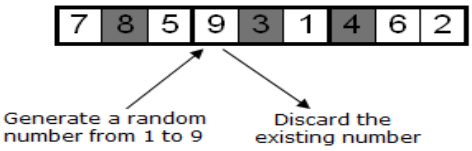


Fig. 4 Bit-wise Mutation

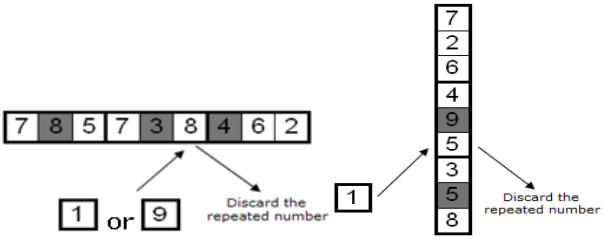
Elitism

While conducting the experiments, neither complete nor partial elitism was found to be efficient. Hence, we followed an approach in between these two. The population before crossover and after mutation were combined together to form a population of size 20*N (Double Size). Arrange the individuals in the ascending order of their fitness values. To maintain the diversity in the population, the alternate individuals are then selected for the next generation and the cycle continues with the population of the original size 10*N. This step is called as 'alternate elitism'.

Remove repetition

In order to ensure faster convergence, we employ a strategy to utilize the givens of the Sudoku puzzle. We call this operation as "Remove Repetition", in which any repetition of a given is replaced by a randomly selected number from the set ($\{1, 2, 3, 4, 5, 6, 7, 8, 9\} - G_i$), where G_i is the set of numbers present in the ith row or column under consideration (see Figure 5). The operation is first performed on each row and then on each column. During column wise operation some of the givens may be repeated in a row but such instances were found to be very few, hence the overall fitness of the individuals improves. It should be noted that repetition of non-givens is not removed

by this process. This technique is applied after generating initial population, after crossover and after mutation.



(a) Row-wise (b) Column-wise

Fig. 5 Remove repetition of givens

Fitness function

It is often difficult to design a fitness function in combinatorial problems [19]. However, in this paper, an attempt is made to design a simple fitness function for a generalized $N \times N$ Sudoku puzzle. A simple *uniqueness* technique is being used to design the fitness function. It consists of three different fitness terms, namely row-fitness, column-fitness and sub-square-fitness. These are the only constraints taken with equal penalty, 1 each. Hence the overall fitness function is defined as:

Fitness function = Row fitness + Column fitness + Sub-square fitness (2)

Each of the above three functions attains maximum value only when the solution is reached. In the following lines we derive the expression for maximum overall fitness value. Each row entry is compared with all the remaining entries to its right. If the two entries are not equal, row-fitness value is incremented by 1 otherwise it remains same. Thus for the solution the contribution from each row is $N(N-1)/2$ (sum of first $N-1$ natural numbers). Hence for N rows, it will be $N^2(N-1)/2$. Similar results hold for column and sub-square. Hence for an $N \times N$ Sudoku, the maximum fitness value is $3N^2(N-1)/2$ (which comes out to be 972 for a 9 X 9 Sudoku puzzle). To derive the fitness function for a Sudoku puzzle at any intermediate point during simulation, we define a function

$$f(i,j,k,l) = \begin{cases} 0, & if\ (i,j) = (k,l) \\ 1, & otherwise \end{cases} \quad (3)$$

where, (i, j) and (k, l) refer to two positions of an $N \times N$ Sudoku puzzle.

The fitness function for the rows is defined as follows,

$$Row\ fitness = \sum_{i=1}^{N} \sum_{j=1}^{N-1} \sum_{l=j+1}^{N} f(i,j,i,l) \quad (4)$$

Equation (4) indicates that comparison starts from the cell (1, 1) with (1, 2), (1, 1) with (1,3),….. up to (1,1) with (1, N); (1,2) with (1,3), (1,2) with (1,4),……. up to (1,2) with (1,N);……..; finally (1, N-1) with (1, N). This completes the first row, i.e. for $i = 1$. Similarly, it moves up to N rows.

The fitness function for the columns is defined as follows,

$$Column\ fitness = \sum_{j=1}^{N} \sum_{i=1}^{N-1} \sum_{l=i+1}^{N} f(i, j, l, j) \quad (5)$$

The similar arguments hold as described above, starting from column 1 to N, by using equation (5). The fitness function for sub-square is defined as follows,

$$Sub-square-fitness = \sum_{i=1}^{N} \left[\sum_{q=1}^{\sqrt{N}} \left\{ \sum_{j=1+(q-1)\sqrt{N}}^{q\sqrt{N}-1} \sum_{l=j+1}^{q\sqrt{N}} f(i, j, i, l) + \sum_{\substack{r=1+(q-1)\sqrt{N} \\ i \neq t\sqrt{N}, t \in z^{+}}}^{q\sqrt{N}} \sum_{k=i+1}^{i+\sqrt{N}-i\left(\mathrm{mod}\sqrt{N}\right)} \sum_{s=1+(q-1)\sqrt{N}}^{q\sqrt{N}} f(i, r, k, s) \right\} \right] \quad (6)$$

where, Z^{+} is the set of all positive integers.

Equation (6) represents that first the comparison completes in the first sub-square and then it jumps to the 2nd sub-square on right to it. Gradually it covers all the N number of sub-squares. For a clear understanding it can be exercised for a 9X9 Sudoku puzzle, where $N = 9$.

The moments a solution is tapped, it needs to check whether the maximum fitness value is achieved or not (Figure 6). This process is applied separately after generating the initial population, after crossover and after mutation. Attaining the maximum fitness plays an important role as it is being used as one of the stopping criteria in the algorithm. However, it is observed that as the fitness value of an individual moves towards the maximum value, many a time, it gets trapped at some particular value. It becomes very difficult to get out of it. As a result, it provides a premature convergence. Hence to overcome this shortcoming, the population is to be reinitialized after a *reset point* in the same run and thus try to approach the solution through a different path. The *reset point* for a particular type of problem is defined as the number of generations needed to wait with a repeated fitness values during the simulation. After the reset point attained, proposed method attempts to regenerate the initial population, only if the number of generation is not attaining the maximum of 50000. We set different reset points for different difficulty level Sudoku puzzles. Higher the difficulty level (or, less is the number of givens, in general), greater is the reset point. After a series of experiments and observation, the reset point is set to 2000 if there are 27 givens or less, 350 if there are exactly 28 or 29 givens, 300 if there are exactly 30 or 31 givens and 200 if there are 32 givens or more. Thus our algorithm tries to retrieve the solution if it gets stuck somewhere. Hence we call it as a Retrievable Genetic Algorithm *(Ret-GA)*. The mechanism of Ret-GA is depicted in Figure 6.

4. Computational Experiment

Experimental Setup

The proposed Ret-GA algorithm starts with generating a "Blueprint Matrix". This matrix contains 0 or 1 at the place of non-given and given positions respectively. Thus Blue print matrix just acts as a reference matrix in the background of the simulation with a warning that not to change the positions where it contains '1s'. Only changes possible where there are '0s'.

In the Ret-GA, the population size is kept fixed to 90 (i.e. 10*N, where N=9). Uniform crossover and bitwise mutation have been incorporated at a probability of 0.8 and 0.2, respectively.

A test bed of 9 Sudoku puzzles are picked up from [2] except their 1st (new) puzzle, where there are no givens. Out of these 9, there are 5, which they considered from the newspaper [20], marked with difficulty rating *1-5 star*, where there are symmetric givens. Rest 4 problems are taken from newspaper [21], marked with difficulty rating: *Easy, Challenging, Difficult and Super difficult*. These puzzles contain 23 to 26 nonsymmetrical givens. These are called as *Sample Sudoku*

Puzzles. Each problem undergoes 100 runs. The stopping criterion for a run is either the optimum value (972) is reached or the maximum generation (fixed to 50,000) is attained.

Results and Discussion

In 100 different runs, a run is said to be success, if it finds the optimum value 972. In table-1, GA represents the GA used in [2]. The number of times the run is a success one, is reported in Table-1. Similarly, the minimum, maximum and average number of function evaluations from both GA and Ret-GA are recorded in table-1. The last two columns represent the median and standard deviation of the required function evaluations of the successful runs only.

By Table 1 it can be noted that in all the sample problems success rate is better in Ret-GA than that of GA, where in the puzzles *1-star* and E*asy*, they are equal to 100%. Out of all successful runs in 100 complete runs, the minimum and maximum generations obtained for each puzzle is reported.

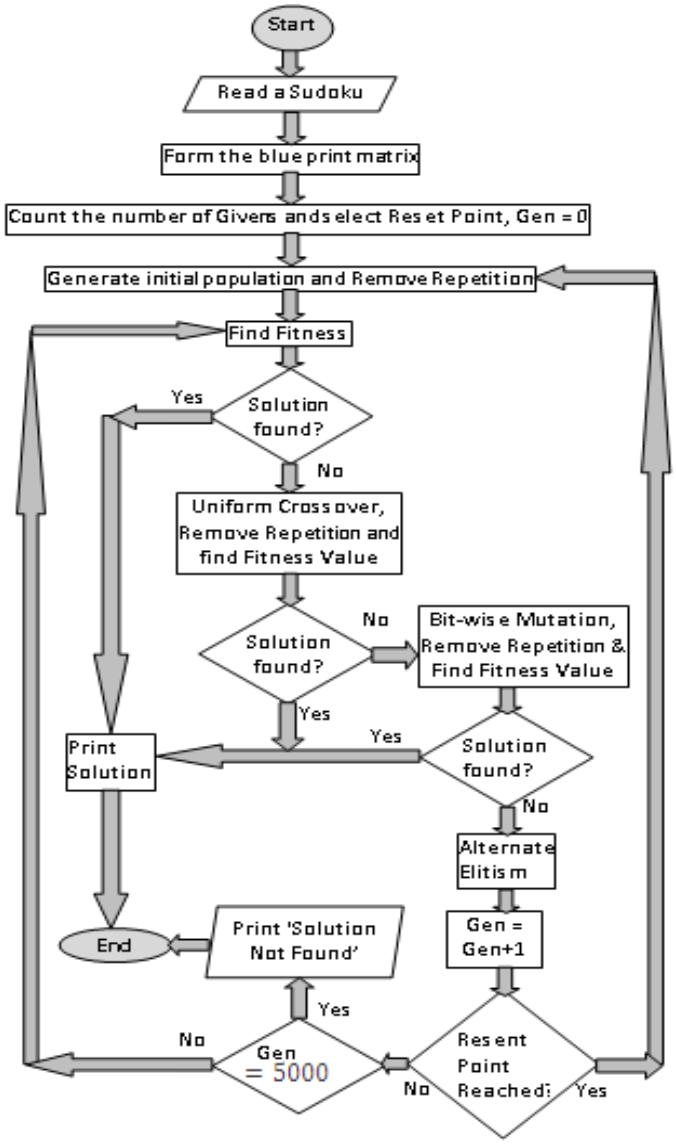


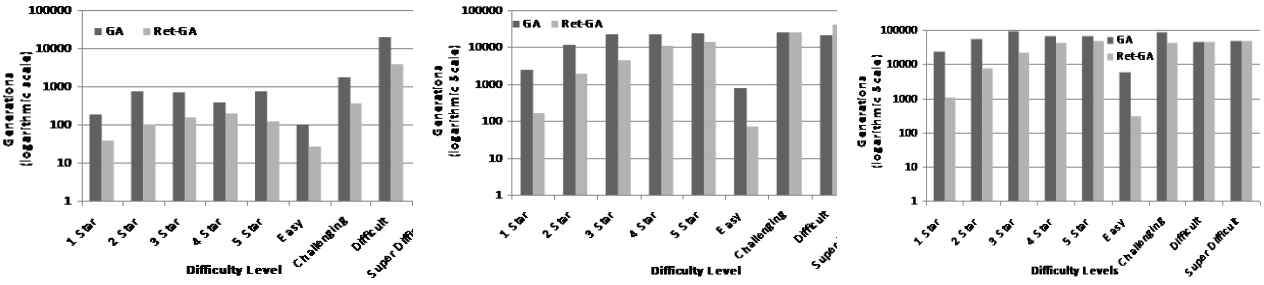
Fig. 6 Schematic representation of the proposed Ret-GA

The average number of generations required for the successful runs are also mentioned. To better understand the comparison for minimum, maximum and average number of generations, we plot graphs in Figure 7 (a), (b) and (c) respectively, for all 9 puzzles. From Figure 7 it is clear that in all the first 6 puzzles Ret-GA is better than GA with respect to the minimum, maximum and average number of iterations. Ret-GA takes less number of generations to solve *Challenging* and *Difficult* problems in the sense of minimum and maximum number of generations, but GA beats Ret-GA in average. Only in the super difficult problem, Ret-GA takes less number of iterations with respect to minimum, maximum and average. However, it should be noted that even Ret-GA takes more number of generations to solve *Supper difficult* problems, but in return it provides higher success rate and lower SD than GA. Moreover, corresponding to each and all nine sample Sudoku puzzles, the SD by Ret-GA is less than that of GA. Therefore as a whole, Ret-GA performs better than GA in most of the cases.

It is found that the average number of function evaluations for the puzzles rating *Challenging, Difficult and Super Difficult*; the minimum number of function evaluations for *Super Difficult puzzles*, are smaller in case of GA. Thus it is a difficult task to arrive to a concrete conclusion. Hence, a second method of analysis is considered as described below.

Table 1. Performance of GA Vs. Ret-GA for different difficulty levels of sample Sudoku Puzzles

Level	Givens	Count		Minimum		Maximum		Average		Median		Standard Deviation	
		GA	Ret-GA	GA	Ret-GA	GA	Ret-GA	GA	Ret-GA	GA	Ret-GA	GA	Ret-GA
1 Star	33	100	**100**	184	**39**	23993	**1100**	2466.60	**159.24**	917	**98**	3500.98	**159.95**
2 Star	30	69	**100**	733	**99**	56484	**7459**	11226.80	**1864.94**	7034	**1441**	11834.68	**1728.25**
3 Star	28	46	**100**	678	**152**	94792	**21595**	22346.40	**4338.30**	14827	**2755**	24846.46	**3906.40**
4 Star	28	26	**100**	381	**198**	68253	**42382**	22611.30	**10716.73**	22297	**6966**	22429.12	**11285.80**
5 Star	30	23	**100**	756	**117**	68991	**48336**	23288.00	**13569.76**	17365	**9393**	22732.25	**12832.50**
Easy	36	100	**100**	101	**27**	6035	**305**	768.60	**70.34**	417	**52**	942.23	**55.70**
Challenging	25	30	**94**	1771	**357**	89070	**41769**	**25333.30**	25932.87	**17755**	27786	23058.94	**19153.54**
Difficult	23	4	**16**	18999	**3699**	46814	**45676**	**20534.30**	40466.13	**26162**	33760	12506.72	**11940.66**
Super Difficult	22	6	**9**	**3022**	21424	**47352**	49832	**14392.00**	42354.86	**6722**	36318	17053.33	**11198.42**



(a) Minimum number of generations required. **(b) Maximum number of generations required.** **(c) Average number of generations required.**

Fig. 7 Comparison of GA and Ret-GA on Sample Sudoku Puzzles.

5. Conclusion

This paper exhibits a specialized heuristics, called "Retrievable Genetic Algorithms" for solving Sudoku puzzle. The Retrievable GA is designed to keep in mind a new way of designing the fitness function and the puzzle-characteristics dependent constraints. It is name as "retrievable" since it involves a single mechanism for escaping from the premature convergence, by reinitializing the population after a specified number of generations. Thus is helpful in maintaining the diversity in search domain. The results obtained by the experiment lead to draw the conclusion that the performance of the proposed Ret-GA is better in comparison to the results of existing GA. In solving the sample puzzles, it is observed that the success rate obtained by Ret-GA is higher than that of GA and the SD due to Ret-GA is lesser than that of GA in each problem. Hence we conclude that Ret-GA is more reliable and more stable than GA in solving Sudoku puzzles.

Further it is worthy to note that Ret-GA could solve 1-5 star and Easy rating puzzles with 100% success. It requires lesser number of generations as compared to existing GA. Although, Ret-GA is not able to solve Challenging, Difficult and Supper difficult problems with 100% success, but the success rate of Ret-GA is always higher as compared to GA. Moreover, the average number of generations required to solve Challenging, Difficult and Supper difficult problems by Ret-GA are lesser than that of GA.

After all, Ret-GA is a unique and novel approach to solve Sudoku puzzles. The approach may be extended to a larger order (size) Sudoku and our next paper may find to use Ret-GA effectively for higher dimensional puzzles. Researchers are encouraged to develop new fitness functions provides better success rate in fewer generations for a general Sudoku puzzle.

References

[1] Lewis, R. (2007). *Metaheuristics can solve Sudoku puzzles*. Journal of Heuristics, Springer, vol. 13, no. 4, pp. 387-401.

[2] Mantere, T. and Koljonen, J. (2006). *Solving and rating Sudoku puzzles with Genetic Algorithms*. New Development of Artificial Intelligence and the Semantic Web, Proceeding of the 12th Finnish Artificial Intelligence Conferences, STeP.

[3] Mantere, T. and Koljonen, J. (2007). *Solving, rating and generating Sudoku puzzles with GA*. CEC 2007, IEEE Congress on Evolutionary Computation, pp. 1382 – 1389.

[4] Moraglio, A. and Togelius, J. (2007). *Geometric particle swarm optimization for the sudoku puzzle*. GECCO '07 Proceedings of the 9th annual conference on Genetic and evolutionary computation, pp. 118-125.

[5] Haynes, D., Corns, S. and Venayagamoorthy, G. K. (2012). *An exponential moving average algorithms applied to Sudoku puzzle*. IEEE Congress on Evolutionary Computation, Brisbane, Australia, June 9-14.

[6] Semeniuk, I. (2005). *Stuck on you*. In *NewScientist 24/31*. pp. 45-47.

[7] Wikipedia. (2006) Sudoku. Available via WWW: http://en.wikipidia.org/wiki/Sudoku (cited 11.09.2006).

[8] Frazer Jarvis. (2006). *Sudoku enumeration problems*. Frazer Jarvis's home page, Retrieved on September 16.

[9] Lawler, E. L., Lentra, J. K., Rinnooy, A. H. G. and Shmoys D. B. (eds.). (1985). *The traveling salesman problem – A guided tour of combinatorial optimization*. John Wiley & Sons, New York.

[10] Ardel, D. H. (1994). *TOPE and magic squares, a simple GA approach to combinatorial optimization*. In J. R. Koza (ed.). Genetic Algorithms in Stanford, Stanford bookstore, Stanford, CA.
[11] Alander, J. T., Mantere, T. and Pyylampi, T. (1999). *Digital halftoning optimization via genetic algorithms for ink jet machine*. In B.H. V. Topping (ed.), Developments in Computational mechanics with high performance computing, CIVIL-COMP Press, Edinburg, UK, pp. 211-216.
[12] Kang, H. R. (1999). *Digital color halftoning*. SPIE Optical Engineering Press, Bellingham, Washington, & IEEE Press, New York.
[13] Alander, J. T., Mantere, T. and Pyylampi, T. (1998). *Threshold matrix generation for digital halftoning by genetic algorithm optimization*. In D. P. Casasent (ed.), Intelligent Systems and Advanced Manufacturing: Intelligent robots and Computer Vision XVII: Algorithms, Techniques, and Active Vision, volume SPIE-3522, Boston, MA, 1-6. SPIE, Bellingham, Washington, USA, pp.. 204-212.
[14] Kobayashi, N. and Saito, H. (1993). *Halftone algorithm using genetic algorithm*. In proceeding of 4th Int. conf. on Signal Processing Applications and Technology, Vol. 1, Newton, MA 28. Sept.-1. Oct. 1993, DSP Associates, Santa Clara, CA, Pages. 727-731.
[15] Newbern, J. and Bowe, Jr. M. (1997). *Generation of Blue Noise Arrays by Genetic Algorithms*. In B.E.Rogowitz and T. N. Pappas (eds.), Human Vision and Electronic Imaging II, San Jose, CA, 10-13 Feb, Vol. SPIE-3016, SPIE – Int. society of Optical Engineering, Bellingham, WA, pp. 441-450.
[16] Wiley, K. (1998*). Pattern evolver, an evolutionary algorithm that solves the non-intuitive problem of black and white pixel distribution to produce tiled patterns that appear grey*. The Handbook of Genetic Algorithms, CRC Press.
[17] Gold, M. (2006). *Using Genetic algorithms to come up with Sudoku puzzles*. Available via WWW: http://www.c-sharpcorner.com/UploadFile/mgold/Sudoku0923005003323AM/Sudoku.aspx?ArticleID-fba36449-ccf3-444f-a435-a812535c45e5 (cited 11.09.2006).
[18] Moraglio, A., Toqelius, J. and Lucas, S. (2006). *Product geometric Crossover for the Sudoku puzzle*. Evolutionary Computation, ECE 2006, IEEE cogress, pp. 470-476.
[19] Koljonen, J. and Alander, J. T. (2004). *Solvoing the urban horse problem by backtracking and genetic algorithm a comparison*. In Proceedings of the 11th Finnish Artificial Intelligence Conference (STeP 2004), Jarmo T. Alander, Pekka Ala-Siuru and Heikki Hyötyniemi (eds.), Vantaa (Finland), 1–3rd Sep. 2004, vol. 3, pages 127–136.
[20] Helsingin, S. (2006). Sudoku available via WWW: http://www2. hs.fi/extrat/Sudoku/Sudoku.html (cited 11.01.2006).
[21] Aamulehti,(2006). Sudoku online via WWW: http://www. aamulehti.fi/Sudoku/ (cited 11.01.2006).