ISSN No. 0976-5697

Volume 1, No. 3, Sept-Oct 2010

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

SudokuBee: An Artificial Bee Colony-based Approach in Solving Sudoku puzzles

John Paul T. Yusiong*
Division of Natural Sciences and Mathematics
University of the Philippines Visayas Tacloban College
Tacloban City, Leyte, Philippines
jpyusiong@gmail.com

Glaiza Mae M. Seno
Division of Natural Sciences and Mathematics
University of the Philippines Visayas Tacloban College
Tacloban City, Leyte, Philippines
glaizaseno@gmail.com

Jaysonne A. Pacurib
Division of Natural Sciences and Mathematics
University of the Philippines Visayas Tacloban College
Tacloban City, Leyte, Philippines
jaysonne_pacurib@yahoo.com

Abstract: Sudoku is a popular newspaper puzzle that has become an international phenomenon. Sudoku which means "single digits" is a logic-based combinatorial puzzle with rules that are quite simple. Several algorithms have been used to solve this combinatorial optimization problem. This paper explores the possibility of using the Artificial Bee Colony (ABC) algorithm, a relatively new swarm-based optimization algorithm that mimics the foraging behaviour of bees, in solving Sudoku puzzles. The experiment results show that the ABC-based Sudoku solver has an excellent performance in solving the *6 x 6*, *9 x 9* and *12 x 12* Sudoku puzzles with varying levels of difficulty. Thus, the results reveal the potential of the ABC algorithm in solving Sudoku puzzles.

Keywords: Artificial Bee Colony, Sudoku Puzzles, Combinatorial Optimization Problem, Sudoku solver, NP-Complete Problem

I. INTRODUCTION

Puzzles are problems designed to be mentally challenging, and they require ingenuity and creativity to be solved. Sudoku is a popular newspaper puzzle that has become an international phenomenon [12]. Sudoku which means "single digits" is a logic-based combinatorial puzzle with rules that are quite simple. It is mathematically interesting given that simple logic can be applied to solve this puzzle. It can also be examined as a graph coloring problem. More importantly, it has interesting combinatorial properties. But aside from being a combinatorial optimization problem it is also a constraint satisfaction problem [4, 5, 11]. Moreover, Sudoku puzzles belong to a set of hard problems called NP-Complete problems [20], which means that to find the optimal solution one has to develop an efficient algorithm that can check all possible combinations. Even though most experts believe that no such algorithm exists, they still continue to search for improved algorithms that can offer a better way to obtain the best solution [1]. Thus, attempting to find improved algorithms to quickly solve Sudoku puzzles may lead to discovering better ways of solving NP-Complete problems.

The challenge in Sudoku is not just about finding a solution that satisfies the given constraints. It also requires the solution to be found as quickly as possible. For instance, given a *9 x 9* Sudoku, the objective is to put a number from 1 to 9 in each cell such that each row, column and block contains exactly one instance of each number. Several algorithms have been applied to solve this combinatorial optimization problem including Ant Colony Optimization (ACO) algorithm [16], Belief Propagation (BP) [15], Constraint Programming (CP) [5], Cultural Algorithms (CA) [14], Genetic Algorithms (GA) [13], Harmony Search (HS) algorithm [7], Particle Swarm Optimization algorithm (PSO) [8], Simulated Annealing (SA) [12] and hybrid algorithms [11, 18].

In this paper, we present a new method in solving Sudoku puzzles using the Artificial Bee Colony (ABC) algorithm, which is based on the foraging behavior of bees. This algorithm, which is designed for solving multi-dimensional and multi-modal optimization problems, is known to be simple, robust, efficient and flexible [9]. It has been shown to outperform other well-known algorithms [10]. The motivation of this work is to test the efficiency of the ABC algorithm when presented with a discrete multi-dimensional optimization problem and compare its performance with other optimization algorithms. That is, this paper aims to solve the Sudoku puzzles with the objective of finding the optimal solution in a reasonably short period of time.

This paper considers the extended results of the previous work presented in a conference [17] by adding further descriptions and experiments. This paper is organized as follows. A description about Sudoku puzzles is presented in Section 2. Section 3 introduces the Artificial Bee Colony (ABC) algorithm followed by a discussion on the proposed ABC-based Sudoku solver in Section 4. The experimental results are shown in Section 5 while Section 6 contains the conclusion.

II. SUDOKU PUZZLES

Sudoku, also known as "Number Place", has become an international phenomenon. Sudoku puzzles are featured regularly in many newspapers and the Internet contains thousands of references to these interesting puzzles [11]. These puzzles are available at different levels of difficulty. Nonetheless, Sudoku is drawing more attention than other types of puzzles.

The standard Sudoku has a dimension of *9 x 9* with nine *3 x 3* blocks and a total of 81 cells. But other Sudoku puzzles with different dimensions are also available. Table 1 lists several of these Sudoku puzzles with different dimensions.

However, regardless of dimension, a Sudoku puzzle can be defined as follows. Given a set of numbers from 1 to *n*, where *n* is a positive integer, an *n x n* Sudoku puzzle is to be filled such that:

1. Each row of cells contains numbers 1 to *n* exactly once.
2. Each column of cells contains numbers 1 to *n* exactly once.
3. Each block contains numbers 1 to *n* exactly once.

Initially, the puzzle has fixed values for certain cells called "start squares". The value on each "start square" cannot be modified when searching for an optimal solution. Fig. 1(a) shows an unsolved *9 x 9* Sudoku puzzle.

Table I. Different Sudoku Puzzle Sizes

Dimension	*Number of Blocks*	*Number of Cells*
4 x 4	four 2 x 2	16
6 x 6	six 2 x 3	36
8 x 8	eight 2 x 4	64
12 x 12	twelve 3 x 4	144
16 x 16	sixteen 4 x 4	256
25 x 25	twenty-five 5 x 5	625

To solve an *n x n* Sudoku puzzle, it has to be completely filled in such a way that each of the n rows, n columns and n blocks contains the numbers 1 to *n* exactly once, as illustrated in Fig. 1(b). This means that the objective is to completely fill the grid with numbers from 1 to *n* such that each row, column and block contains the numbers without any duplication [4]. Interestingly, the sum of the numbers in each row, column and block of the optimal solution should be equal.

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

(a)

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

(b)

Figure 1. A 9 x 9 Sudoku Puzzle.

The goal of finding a solution to a Sudoku puzzle may look trivial but it has been calculated that a *9 x 9* Sudoku puzzle has 6,670,903,752,021,072,936,960 different combinations [4, 6] or approximately 6.671×10^{21} valid Sudoku puzzles. Hence, applying the constraints would lessen the number of possible combinations. However, the resulting search space would still be too large for one optimal solution.

III. THE ARTIFICIAL BEE COLONY (ABC) ALGORITHM

The Artificial Bee Colony (ABC) algorithm is a swarm-based optimization algorithm designed by Dervis Karaboga in 2005 [9]. This is a relatively new population-based optimization algorithm that mimics the behavior of bees when foraging. A bee communicates with other bees through a waggle dance wherein that bee conveys the distance and direction of the food source, and the quality of the nectar found. This foraging behavior of bees has been expanded to build partial solutions in many problem domains including combinatorial optimization problems [2, 3, 19].

Given that this has been successfully applied to many problem domains, the researchers are exploring the idea of using ABC in solving the Sudoku puzzle, which can be formulated as a multi-dimensional optimization problem that takes into consideration the three constraints mentioned earlier. Table 2 shows the different parameters of the ABC algorithm followed by the pseudo-code of the ABC algorithm.

Table II. Different Parameters of the ABC algorithm

Parameter	*Symbol*
Number of Employed Bees	EBee
Number of Onlooker Bees	OBee
Number of Scout Bees	SBee
Food Source Dimension	Decision variables, D
Random Function: rand()	[-1,1]
Search Steps/Cycles	maxCycles

Begin
[01] Initialize Population
[02] Evaluate Population
[03] Set cycle=1
[04] Repeat
[05] Produce new solution V_i in the neighborhood of X_i
[06] Evaluate Population
[07] Apply Greedy Selection process between X_i and V_i
[08] Calculate the probability values P_i
[09] Normalize Pi values into [0,1]
[10] Produce new solutions V_i from X_i depending on the value of P_i
[11] Evaluate Population
[12] Apply Greedy Selection process between X_i and V_i
[13] If abandoned solution exists
[14] Replace abandoned solution with new solution
[15] Memorize best solution
[16] cycle=cycle+1
[17] Until (termination condition is met)
End

Fig. 2 illustrates the colony of the artificial bee, composed of three groups: employed bees, onlooker bees and scout bees, with their corresponding tasks.

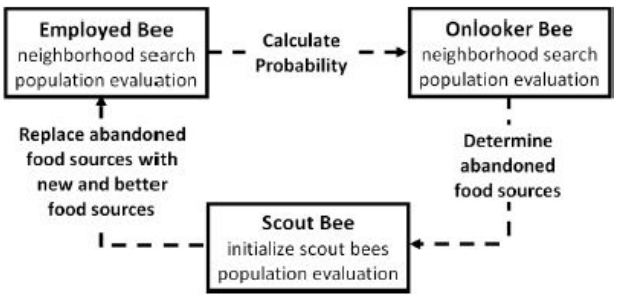

Figure 2. The ABC algorithm.

IV. SUDOKUBEE: AN ABC-BASED SUDOKU SOLVER

The research of Quan and Shi in [19] presented an improved version of the artificial bee colony, which was first developed by Karaboga [9].

In this section, a discrete case of the improved ABC algorithm is presented to solve Sudoku puzzles.

In the proposed approach, the Sudoku puzzle is formulated as a discrete multi-dimensional optimization problem. Furthermore, the puzzle is represented as an instance of combinatorial optimization problems. That is, it is

characterized by intrinsically ensuring that the third constraint is not violated while leaving the task of satisfying the other two constraints to the algorithm being employed [18]. In other words, the optimization problem is to minimize the number of missing digits in each row and column.

The optimization process carried out using the artificial bee colony in solving the Sudoku puzzle problem can be summarized as follows:

1. Each employed bee randomly develops a food source, x_i and remembers the nectar amount of the food source. The food source is a *D*-dimensional vector where *D* is the number of cells in a Sudoku puzzle. Fig. 3(a) is an example of a food source while Fig. 3(b) shows a food source developed by the employed bee. In developing a food source, the employed bee randomly assigns digits to all nonstart squares.

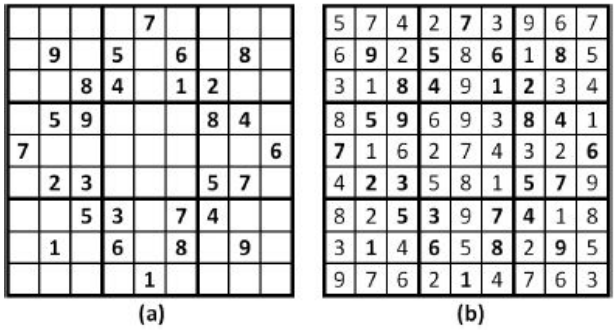

Figure 3. The Food Source.

2. The probability P_i represents the waggle dance of a bee and it is simply an indicator of the attractiveness of a food source to the onlooker bees. Hence, with respect to the nectar amount fit_i, of every food source, the probability P_i is computed using (1):

$$P_i = \frac{fit_i}{\sum_{i=1}^{N} fit_i} \quad (1)$$

where *N* represents the total number of food sources. The nectar amount fit_i is computed using (2).

$$fit_i = \frac{1}{1+f_i} \quad (2)$$

where the food source x_i has a function value f_i, which is the total number of missing digits in each row and column of the Sudoku puzzle. In Fig. 4, the function value of the food source developed by an employed bee x_i is 40.

The onlooker bees, $obee_i$, are then assigned to every food source x_i to exploit the neighborhood of the food sources according to the probability P_i in (1). Thus, the number of onlooker bees, $obee_i$ in every food source is given by (3):

$$obee_i = P_i * OBee \quad (3)$$

where $obee_i$ is the number of onlooker bees assigned to a food source x_i, P_i is defined in (1) and *OBee* is the total number of onlooker bees in the colony.

5	7	4	2	**7**	3	9	6	7	**2**
6	**9**	2	**5**	8	**6**	1	**8**	5	**3**
3	1	**8**	**4**	9	**1**	**2**	3	4	**3**
8	**5**	**9**	6	9	3	**8**	**4**	1	**2**
7	1	6	2	7	4	3	2	**6**	**3**
4	**2**	**3**	5	8	1	**5**	**7**	9	**1**
8	2	**5**	**3**	9	**7**	**4**	1	8	**1**
3	**1**	4	**6**	5	**8**	2	**9**	5	**1**
9	7	6	2	**1**	4	7	6	3	**2**
2	**4**	**2**	**4**	**4**	**3**	**1**	**1**	**1**	**40**

Figure 4. The Function Value, f_i, of the Food Source.

3. Employed bees in every food source and onlooker bees assigned to this food source x_i perform the neighborhood search process using (4):

$$v_{ij}(s+1) = ceil(x_{ij}(s) + \phi_{ij} * abs(x_{ij}(s) - x_{kj}(s))) \quad (4)$$

where i, k ∈ {1, 2,…, N} are indexes of the food sources and j ∈ {1, 2,.., D} is an index of the *D*-dimensional vector of food sources, which corresponds to a nonstart square of the Sudoku puzzle, *s* indicates the search step order of the neighborhood search process. $v_{ij}(s+1)$ is the position that the employed bee and onlooker bees and at search step $(s+1)$. $x_{ij}(s)$ is the optimal position found by the employed bee and onlooker bees in the same food source at each search step. $x_{kj}(s)$ is a food source that is randomly chosen from the set of food sources. Φ_{ij} is a random number between [-1,1], *ceil()* is a function that rounds a number up to the nearest integer and *abs()* returns the absolute value of a number.

If v_{ij} is greater than *n* then (5) below is used to recompute for v_{ij}.

$$v_{ij}(s+1) = \left(v_{ij}(s+1) mod\ n\right) + 1 \quad (5)$$

In each search step, the employed bee and each onlooker bee assigned to a food source randomly select another food source x_k from the set of food sources and produce a new position v_{ij} by (4). If the new position v_{ij} yields a better fitness, that is, $fit_i(v) \geq fit_i(x)$, then the new position v_{ij} will replace the current x_{ij}, as the optimal position of the food source x_i. Otherwise, the current food source position x_{ij} is kept.

However, prior to comparing the fitness of v_i and x_i, it has to be ascertained that when a number at a certain food source is changed, the third constraint of the Sudoku puzzle is not violated. Otherwise, a swap operation is performed wherein the previous position of v_{ij} is replaced with the value at position x_{ij}. Fig. 5 to Fig. 7 illustrate this step. For example, Fig. 5(a) is the food source x_i which is to be optimized and Fig. 5(b) is x_k which is a randomly chosen food source from the set of food sources. For this case the random value generated for *j* is 41.

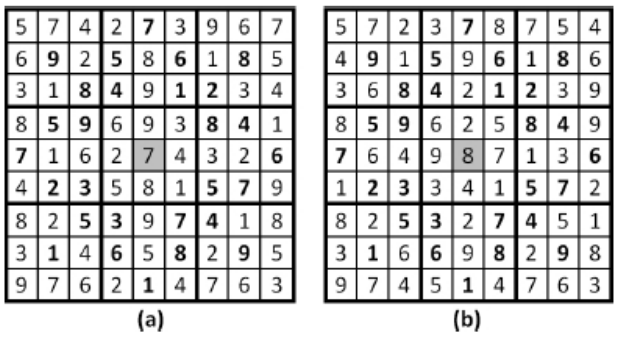

Figure 5. (a) is x_i and (b) is x_k.

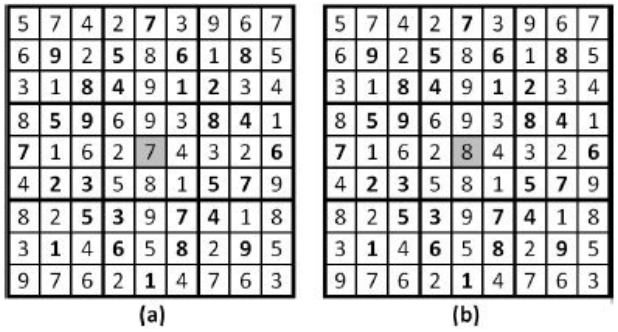

Figure 6. (a) is x_i and (b) is v_i.

In Fig. 6(b), v_{ij} is represented by the shaded cell. Also, in Fig. 6(b) it can be observed that the third constraint of the Sudoku puzzle is violated because of the two 8's in the middle block. Thus a swap operation is performed, that is, the previous position of v_{ij} is replaced with the value at position x_{ij}. Fig. 7 shows the food source v_i after performing the swap operation.

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

Figure 7. v_i after the swap operation was performed.

4. The number of abandoned food sources x_i is determined using (6).

$$SBee = 0.10 * EBee \quad (6)$$

where *SBee* is the total number of scout bees in the colony and *EBee* is the total number of employed bees in the colony.

The abandoned food sources are those with the worst fitness. A scout bee is assigned to each abandoned food source x_i. The scout bees perform an exploration step to discover a new food source that will replace the abandoned food source. In the exploration step, each scout bee randomly develops a new food source. If the new food source has a better fitness than the abandoned food source x_i, then the new food source becomes the new x_i. Otherwise, the abandoned food source is kept.

5. Repeat steps (2) to (4) until the search step is equal to a pre-determined number of cycles, MaxCycles or when the fitness of a certain food source is equal to 1. If the first termination condition is satisfied this means that an optimal solution to the puzzle was not found but if the second condition is satisfied then an optimal solution is found, hence the Sudoku puzzle is solved.

V. EXPERIMENT RESULTS AND DISCUSSION

In order to evaluate the performance of the ABC-based Sudoku solver, several *6 x 6*, *9 x 9* and *12 x 12* Sudoku puzzles with varying levels of difficulty were used as benchmarks to determine the effectiveness of the Sudoku solver. For each puzzle, the experiments were repeated thirty (30) times and each experiment uses a different randomly generated initial population. In the experimental phase, the following parameters were defined:

- Number of Employed Bees, *EBee* = 100
- Number of Onlooker Bees, *OBee* = 200
- Search steps/Cycles, *maxCycles* = 100,000
- Number of Scout Bees, *SBee* = 0:10 * *EBee*

Fig. 8 to Fig. 10 are the Sudoku puzzles that were used in the experiments. Tables 3-8 show the results obtained from our experiments.

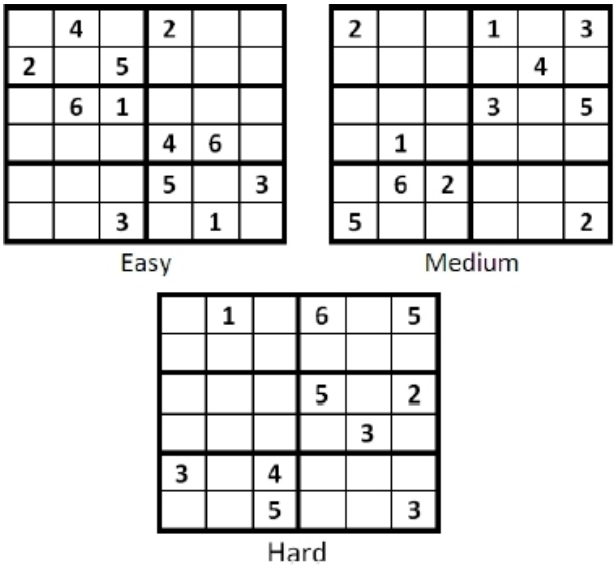

Figure 8. The *6 x 6* Sudoku Puzzles.

In Tables 3 to 5, the tables show the average number of cycles it took the ABC-based Sudoku solver to solve the puzzle while Tables 6 to 8 show the average time in seconds it took the Sudoku solver to find an optimal solution. The values in the tables are averaged over 30 trials.

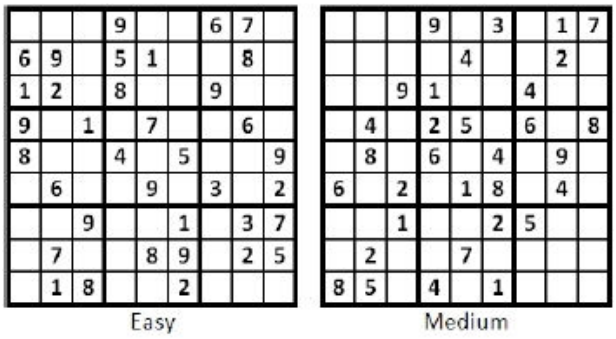

Figure 9. The *9 x 9* Sudoku Puzzles.

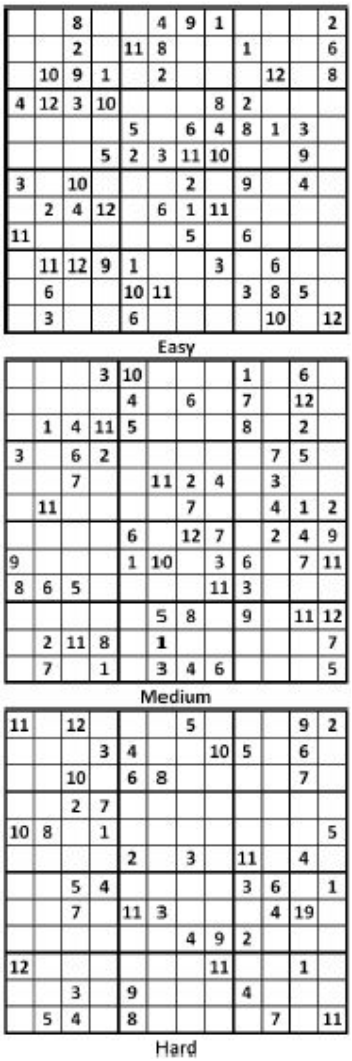

Figure 10. The *12 x 12* Sudoku Puzzles.

The experiment results indicate that the dimension of the Sudoku puzzle affects the number of search steps and time it takes the ABC-based Sudoku solver to find the optimal solution, that is, the higher the dimension of the puzzle the longer it takes the Sudoku solver to find the optimal solution. This is due to the fact that increasing the dimension of the Sudoku puzzle increases the search space.

Table III. The results obtained for the Easy Sudoku Puzzles (Average Cycles)

Easy	***6 x 6***	***9 x 9***	***12 x 12***
Mean	25.93	95.97	478.73
Median	25.00	104.50	475.00
Maximum	45.00	145.00	658.00
Minimum	6.00	33.00	235.00
Std. Dev.	9.64	28.41	118.58

Table IV. The results obtained for the Medium Sudoku Puzzles (Average Cycles)

Medium	***6 x 6***	***9 x 9***	***12 x 12***
Mean	29.87	374.30	514.00
Median	29.00	358.00	546.50
Maximum	45.00	771.00	799.00
Minimum	16.00	160.00	250.00
Std. Dev.	8.30	153.60	139.24

Table V. The results obtained for the Hard Sudoku Puzzles (Average Cycles)

Hard	***6 x 6***	***9 x 9***	***12 x 12***
Mean	49.53	554.10	12,333.00
Median	46.50	529.00	13,254.00
Maximum	98.00	937.00	21,537.00
Minimum	16.00	138.00	1,746.00
Std. Dev.	20.53	248.66	5,298.08

Also, the difficulty level of the Sudoku puzzle affects the ability of the ABC-based Sudoku solver to solve the puzzles. It can be observed that as the difficulty level increases, the number of search steps needed to find the optimal solution of the puzzle increases as well. Nonetheless, results show that the Sudoku solver has always found the optimal solution for all the Sudoku puzzles used in the experiments.

Table VI. The results obtained for the Easy Sudoku Puzzles (Time in seconds)

Easy	***6 x 6***	***9 x 9***	***12 x 12***
Mean	0.29	1.90	25.33
Median	0.30	2.00	20.00
Maximum	0.407	3.00	65.00
Minimum	0.093	0.00	13.00
Std. Dev.	0.08	0.76	14.44

Table VII. The results obtained for the Medium Sudoku Puzzles (Time in seconds)

Medium	***6 x 6***	***9 x 9***	***12 x 12***
Mean	0.35	8.43	35.30
Median	0.31	7.00	22.00
Maximum	0.53	46.00	69.00
Minimum	0.19	3.00	14.00
Std. Dev.	0.10	7.71	21.34

Table VIII. The results obtained for the Hard Sudoku Puzzles (Time in seconds)

Hard	*6 x 6*	*9 x 9*	*12 x 12*
Mean	0.54	10.50	1,265.87
Median	0.51	10.50	899.00
Maximum	1.02	18.00	5,249.00
Minimum	0.203	3.00	92.00
Std. Dev.	0.20	4.80	1,553.00

Furthermore, the experiment results obtained for the Hard Sudoku puzzle used in [14] as shown in Fig. 11 were compared with the results presented in their paper. Table 9 shows the comparison between the two Sudoku solvers. Results indicate that the ABC-based Sudoku solver performs much better than the GA-based Sudoku solver.

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

Figure 11. The Al Escargot Hard Sudoku Puzzle.

Table IX. Performance of ABC-based Sudoku solver and GA-based Sudoku solver.

Cycle	*ABC*	*GA*
Mean	6,243.87	1,238,749.00
Minimum	784.00	7,220.00
Maximum	22,145.00	5,901,680.00

Moreover, in [18], several approaches were tested to solve a particular Sudoku Puzzle (see Fig. 12). These approaches included Cultural Genetic Algorithm (CGA), Repulsive Particle Swarm Optimization (RPSO), Quantum Simulated Annealing (QSA) and the Hybrid method with Genetic Algorithm and Simulated Annealing (HGASA). Based on their results RPSO was unable to solve the Sudoku puzzle.

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

Figure 12. The Sudoku Puzzle used in [18].

Table 10 shows the results obtained using the ABC-based Sudoku solver with the results presented in [18]. Experiment results also show that the ABC-based Sudoku solver performs much better than other stochastic-based Sudoku solvers.

Table X. Performance of ABC-based Sudoku solver andStochastic-based Sudoku solvers.

	ABC	*CGA*	*RPSO*	*QSA*	*HGASA*
Cycle	56	208	n/a	42,700	435
Time (s)	0.82	28.00	n/a	65.00	1.45

VI. CONCLUSIONS

This paper presented a new Sudoku solver based on the Artificial Bee Colony algorithm and its performance has been analyzed. Based on the experiment results, it can be seen clearly that the search steps required to solve a Sudoku puzzle is directly proportional to its dimension. The higher the dimension of the puzzle, the larger the number of search steps needed to find the optimal solution of the puzzle. Another factor affecting the performance of the Sudoku solver is the difficulty level of the Sudoku puzzle. This factor affects the ability of the ABC-based Sudoku solver to solve puzzles, that is, as the difficulty level increases, the number of search steps required to find the optimal solution of the puzzle increases as well. Nonetheless, results show that the proposed ABC-based Sudoku solver has always found the optimal solution for all the Sudoku puzzles used in the experiments. In addition, the ABC-based Sudoku solver performs much better when compared to other optimization algorithms like genetic algorithms and stochastic techniques. Thus, a Sudoku solver based on the ABC algorithm can be used to efficiently solve the Sudoku puzzle problem, a discrete multi-dimensional optimization problem.

VII. REFERENCES

[1] L. Aaronson, "Sudoku Science," IEEE Spectrum, vol. 43, Issue 2, 2006, pp. 16-17.

[2] B. Basturk and D. Karaboga, "An Artificial Bee Colony (ABC) Algorithm for Numeric Function Optimization," IEEE Swarm Intelligence Symposium 2006, vol. 8, Issue 1, 2006, pp. 687-697.

[3] A. Baykasoglu, L. Ozbakor and P. Tapkan, "Artificial Bee Colony Algorithm and Its Application to Generalized Assignment Problem," I-Tech Education and Publishing, Vienna, Austria, 2007, pp. 113-144.

[4] J-M. Chauvet, "Combinatorial Explorations in Su-Doku," Computing Research Repository, abs/0803.4253, 2008.

[5] B. Crawford, M. Aranda, C. Castro and E. Monfroy, "Using Constraint Programming to solve Sudoku Puzzles," ICCIT '08: Proceedings of the 2008 Third International Conference on Convergence and Hybrid Information Technology, 2008, pp. 926-931.

[6] B. Felgenhauer and F. Jarvis, "Mathematics of Sudoku I," Mathematical Spectrum 39, 2006, pp. 15-22.

[7] Z.W. Geem, "Harmony Search Algorithm for Solving Sudoku," Lecture Notes in Artificial Intelligence, vol. 4692, 2007, pp. 371-378.

[8] J.M. Hereford and H. Gerlach, "Integer-valued Particle Swarm Optimization applied to Sudoku puzzles," IEEE Swarm Intelligence Symposium (SIS 2008), 2008.

[9] D. Karaboga, "An Idea Based on Honey Bee Swarm for Numerical Optimization," Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, Turkey, 2005.

[10] D. Karaboga and B. Basturk, "A Powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm," Journal of Global Optimization, vol. 39, no. 3, 2007, pp. 459-471.

[11] R. Lewis, "On the Combination of Constraint Programming and Stochastic Search: The Sudoku Case, 4th International Workshop on Hybrid Metaheuristics, Lecture Notes in Computer Science, vol. 477, 2007, pp. 96-107.

[12] R. Lewis, "Metaheuristics Can Solve Sudoku Puzzles," Journal of Heuristics, vol. 13, 2007, pp. 387-401.

[13] T. Mantere and J. Koljonen, "Solving, Rating and Generating Sudoku puzzles with GA," IEEE Congress on Evolutionary Computation, 2007, pp. 1382-1389.

[14] T. Mantere and J. Koljonen, "Solving and Analyzing Sudokus with Cultural Algorithms," IEEE Congress on Evolutionary Computation, 2008, pp. 4053-4060.

[15] T.K. Moon and J.H. Gunther, "Multiple Constraint Satisfaction by Belief Propagation: An Example of Using Sudoku," IEEE Mountain Workshop on Adaptive and Learning Systems, 2006, pp. 122-126.

[16] D. Mullaney, "Using Ant Systems to Solve Sudoku Problems," School of Computer Science and Informatics, University College Dublin, 2006.

[17] J. Pacurib, G.M. Seno and J.P. Yusiong, "Solving Sudoku Puzzles using Improved Artificial Bee Colony Algorithm," Proceedings of the Fourth International Conference on Innovative Computing, Information and Control (ICICIC 2009), 2009, pp. 885-888.

[18] M. Perez and T. Marwala, "Stochastic Optimization Approaches for Solving Sudoku," Computing Research Repository, abs/0805.0697, 2008.

[19] H. Quan and X. Shi, :On the Analysis of Performance of the Improved Artificial-Bee-Colony Algorithm," Proceedings of the 2008 Fourth International Conference on Natural Computation, vol. 7, 2007, pp. 654-658.

[20] T. Yato and T. Seta, "Complexity and Completeness of Finding another Solution and Its Application to Puzzles," IEICE Trans. Fundamentals, vol. E86-A, no. 5, 2003, pp. 1052-1060.