Work-In-Progress: Solving Sudoku Puzzles Using Hybrid Ant Colony Optimization Algorithm

Ibrahim Sabuncu, Yalova University
Engineering Faculty, Industrial Engineering Department
Yalova, Turkey
isabuncu@yalova.edu.tr

***Abstract*— Sudoku puzzle is a popular logic game since 2005. This puzzle is an NP-Complete problem which means that a very hard problem that is required deep and efficient algorithm to be solved. So it also draws attention of the scientists to develop methods and algorithm in order to solve Sudoku puzzles. In this study I tried to develop a hybrid algorithm which consist both analytical and heuristic steps to solve the Sudoku puzzle. The developed hybrid algorithm includes two analytical steps and one heuristically step. In the first analytical step, basic manual Sudoku solving methods are used to solve the puzzle. If puzzle is not solved completely than improvement analytical step is applied to solve the puzzle. If still puzzle is not solved completely then heuristic step applied to solve the puzzle completely. In the heuristic step ant colony optimization algorithm (ACO) will be applied to the puzzle. Experiments show that this hybrid ACO algorithm can solve the hardest Sudoku puzzles less than one second. As a result, this study shows that ACO is an effective method that can be applied to solve Sudoku puzzles.**

Keywords*— *Ant Colony Optimization, Hybrid Algorithm, Sudoku

I. INTRODUCTION

Sudoku is a number placement puzzle game. Unlike other strategy games such as chess, Sudoku is provably hard, and NP complete problem [1-2]. It is invented by Howard Garnds in 1979 and first published in Dell Magazine using "Number Place" game. In 1986, the game named "Sudoku" in japan, which means "the digits must occur only once". Wayne Gould wrote software to generate Sudoku puzzle and publish them daily in London Times, in 1997 [3]. In 2005, Sudoku puzzles become worldwide popular since 2005 [4]. Since 2005 Sudoku puzzles published a lot newspapers, magazines and web pages widely on the world.

Most general, Sudoku puzzles consist of 81 cells in a 9*9 grid form. Every cell must be filled with the numbers from one to nine. The main grid divided in nine 3*3 sub-groups. Some cells are pre-filled by puzzle developer such that there is only one puzzle, so only one solution that can meet puzzle constraints as follow: The numbers one to nine must appear only once in each row, each column and each sub group [5]. Some Sudoku examples are given at the appendix with their solution.

In the literature, there are a lot of different solving method and algorithm for solving Sudoku puzzles include brute force, constraint programming, binary integer linear program and exact covering implemented via Knuth's Dancing Links algorithm [6]. There are also Belief Propagation, Cultural Algorithms, Genetic Algorithms, Harmony Search algorithm, Particle Swarm Optimization algorithm, Simulated Annealing and Artificial Bee Colony algorithm for solving Sudoku puzzles [7]. We can also count stochastic algorithm [5] and convex optimization [6]. Besides all of these algorithms, Ant Colony Optimization (ACO) algorithm is tried to be applied to Sudoku puzzles [4, 9].

The ACO algorithm that is developed by Mullaney seem not an effective method to solve Sudoku puzzles, because of it only solves %20 of Sudoku puzzles and solving time is 3 to 7 minutes [8]. Muhammad Asif did a similar study. He tested his ACO algorithm with just single puzzle, the run time is only 1.4 seconds, but only 76 cells can be assigned, which should be 81, means that cannot find the solution of the puzzle [9]. So, as far as I can found, there are only two ACO applications to the Sudoku puzzle in the literature, and both cannot solve all type of Sudoku puzzle. However, in this study I had developed a hybrid ant colony optimization algorithm that can solve all type of Sudoku puzzles in less than one or two seconds.

Ant colony algorithms are popular methods for solving combinatorial optimization problems in the literature which is first introduced by Dorigo et al. [10]. The fundamental idea of ant heuristics is based on the behavior of natural ants. The ants leave pheromone during walking, and can smell this pheromone, and usually choose the path that has higher pheromone quantity. Because of ants has these three properties, they can find the shortest way from their nest to the food resources. For example, assume that there are two bridges; one is longer than the other, between ants nest and the food resources. The first ant departed from nest, can choose any of these bridge, probability of choosing these two bridges are currently same (%50). Assume that half of the ants had chosen short bridge and other half chosen long bridge. The ants that had chosen short bridge can turn to nest shorter, so the pheromone quantity on the short bridge will be higher than longer, because more ants passes there in the same time period. Because of higher pheromone quantity, more ants choose the short bridge, this cause more pheromone on the short bridge, after this cycle after a time period, short bridge pheromone quantity will be much higher than long bridge, and as a result nearly all ants chose short bridge. This behavioral mechanism is used to solve complex problems heuristically by

INISCom 2015, March 02-04, Tokyo, Japan

DOI 10.4108/icst.iniscom.2015.258984

using artificial ants. In the next section, the developed hybrid ant algorithm will be explained in detail.

II. The Hybrid Ant Colony Optimization Algorithm

In this study, I tried to develop a hybrid algorithm consist both heuristic ACO and analytical methods to solve Sudoku puzzle. Because of the reason that is explained at the below, sole ACO algorithm solution time will be high and sole analytical methods cannot solve hard Sudoku puzzles. So I mixed them to solve all Sudoku puzzles include hardest ones in very short time.

The hybrid algorithm is consist three steps. The first step is the basic assignment step. In this step, all unassigned cells available assignable numbers sets will be determined. And if there is only one available number that can be assigned a cell, which means there is only one number in this cell's available assignable numbers set, this number assigned to this cell. As an example, assume that in first row and first column, there is an empty cell and in the first column numbers 1,3,5, in the first row numbers 2,4,6, in the sub group numbers 8,9 is appear. These means cell's (1,1) available assignable numbers set consist only one number, 7. So number 7 is assigned cell(1,1). This first step will be repeated until all cells will be assigned or no cell will be left that has only one available assignable number. In the next step, the improvement assignment step, all cells' available number set will be checked, and if a number is appear only one cell's available number set of the cells that in the same row, column or sub group then this number will be assigned to this cell. This second step will be repeated until all cells will be assigned or no cell will be left that satisfy assignment criteria. For the easy and some of medium Sudoku puzzles, these two steps will be enough. But for hard ones, these are not enough. At this point our last step, ACO step, will solve the puzzle completely.

At the last step, for all remaining unassigned cells, a random number will be chosen from the available number set by using ant colony optimization algorithm. Assume that there are 20 cells remain unassigned and each of their available assignable number sets consist 3 numbers. Then there are 320 different solution paths for this problem. Assume that, at the beginning there are 60 unassigned cells, if there weren't first two steps, and ACO will be tried to be applied this problem directly, then there will be 360 different solution paths for this problem. As it can be seen this hybrid algorithm, both decrease solution time because of its analytical steps, and can solve all hard Sudoku puzzles because of it heuristic ACO step.

In the begging of the ACO step, the artificial ants randomly travel on the solution three. First they choose a cell then a number for this cell, from this cell's available assignable number set. Because of there is only one feasible solution path, if ants had chosen wrong path, then in forward steps, ant can be face dead end path. If that will be happen, then a negative pheromone will be leaved to this path. So, probably next ant will not choose the nodes on the solution three that leads to dead ends. As a result solution three will get smaller on every step and finally the feasible solution can be found. During ACO step, basic assignment step will be applied again in each iteration, so the solution speed will be increased too much. The algorithm details will be given at the below.

A. Basic assignment step

1. Load the puzzle from the file or the screen (Only at the start of the program)
2. Do following steps until there isn't any more improvement
    - 2.1.1. For all row (i=1 to 9) repeat following steps
        - 2.1.1.1. For all column (j=1 to 9) repeat following steps
            - 2.1.1.1.1. Check the cell (i,j) if it is empty (zero) then
                - 2.1.1.1.1.1. Find available number(s) that can be assigned to this cell
                - 2.1.1.1.1.2. Initially add all 9 numbers (1..9) to the available numbers set that can be assigned to this cell (i,j)
                - 2.1.1.1.1.3. Check all column on i.th row and all rows on j.th column and all cells which are in the same sub (3*3cells) group of (i,j) cell. If there is any cell that has any number, then remove this number from available numbers set of cell (i,j).
                - 2.1.1.1.1.4. If there is only one number left in the available numbers set of cell (i,j) then assign this number to the cell (i,j).
            - 2.1.1.1.2. End if
        - 2.1.1.2. End Repeat
    - 2.1.2. End Repeat
3. If any cell had assigned in current loop (means there is an improvement) then continue to the loop else exit loop.
4. Check all cells if there is any cell left that is not assigned then proceed to next step.

B. Improved assignment step

5. 'Control row of the cell(i,j) if there is a sole number only in the cell(i,j) available number set but not in other's.
6. For all row and column (j=1 to 9 ; i=1 to 9) repeat following steps
    - 6.1.1. Check the cell (i,j) if it is empty (zero) then
        - 6.1.1.1. For all numbers that is in the available numbers set of cell (i,j) do the following steps
            - 6.1.1.1.1. Let X is the current number in the available numbers set of cell (i,j)
            - 6.1.1.1.2. Check all other cells' available numbers sets that is in the same row with cell(i,j)
                - 6.1.1.1.2.1. If there is a number that is equal to X in the available numbers set of current checked cell than update X

(next number in the set) exit loop

6.1.1.1.2.2. End if

6.1.1.1.3. Continue to the loop (until all cells checked in the row)

6.1.1.1.4. If there isn't any number founded equal to X, then assign X to the cell(i,j)

6.1.1.2. Continue to the loop (until all numbers in the set will controlled)

6.1.2. End if

7. End Repeat

8. 'Control column of the cell(i,j) if there is a sole number only in the cell(i,j) available number set but not in other's.
9. For all row and column (j=1 to 9 ; i=1 to 9) repeat following steps

9.1.1. Check the cell (i,j) if it is empty (zero) then

9.1.1.1. For all numbers that is in the available numbers set of cell (i,j) do the following steps

9.1.1.1.1. Let X is the current number in the available numbers set of cell (i,j)

9.1.1.1.2. Check all other cells' available numbers sets that is in the same column with cell(i,j)

9.1.1.1.2.1. If there is a number that is equal to X in the available numbers set of current checked cell than update X (next number in the set) exit loop

9.1.1.1.2.2. End if

9.1.1.1.3. Continue to the loop (until all cells checked in the column)

9.1.1.1.4. If there isn't any number founded equal to X, then assign X to the cell(i,j)

9.1.1.2. Continue to the loop (until all numbers in the set will controlled)

9.1.2. End if

10. End Repeat

11. 'Control sub group of the cell(i,j) if there is a sole number only in the cell(i,j) available number set but not in other's.
12. For all row and column (j=1 to 9 ; i=1 to 9) repeat following steps

12.1. Check the cell (i,j) if it is empty (zero) then

12.1.1. For all numbers that is in the available numbers set of cell (i,j) do the following steps

12.1.1.1. Let X is the current number in the available numbers set of cell (i,j)

12.1.1.2. Check all other cells' available numbers sets that is in the same sub group with cell(i,j)

12.1.1.2.1. If there is a number that is equal to X in the available numbers set of current checked cell than update X (next number in the set) exit loop

12.1.1.2.2. End if

12.1.1.3. Continue to the loop (until all cells checked in the sub group)

12.1.1.4. If there isn't any number founded equal to X, then assign X to the cell(i,j)

12.1.2. Continue to the loop (until all numbers in the set will controlled)

12.2. End if

13. End Repeat
14. Check all cells, if there is any cell left that is not assigned then at first repeat basic assignment process to update available number sets of the cells then proceed to next step.

C. Heuristic Ant Colony Optimization assignment step

15. Do following steps for all unassigned cells

15.1. Randomly choose one unassigned cell

15.2. Calculate chosen probability of the one numbers in the available number set of chosen cell, according to their weight (pheromone value). 'At the beginning all weights are equal in the set, during the process weights will be updated

15.3. Randomly choose one number in the available number set of chosen cell

15.4. Repeat basic assignment process to update available number sets of the cells

15.5. If there is a cell that has an empty available number set (because of random assignment, the assignment can be wrong and cause unfeasible solution) this mean currently solution of the puzzle unfeasible. Then:

15.5.1. Decrease the weight of the number that is chosen randomly for the chosen cell (so next time this number probably will not assign to this cell)

15.5.2. Unassigned chosen cell (so another number can be assigned to this cell and puzzle solution can be still feasible)

15.6. End if

16. Loop

III. EXPERIMENTS

The developed algorithm has programmed at Visual Basic 5.0, and algorithm is tested by using this computer program. The algorithm tested with a lot of Sudoku Puzzles, but only ten of them are given in the paper at appendix. These ten Sudoku puzzles are the hardest ones that I can found. The first 5 puzzles are taken from a game web page that are highest level Sudoku puzzles. The 6th puzzle is taken from Daily Mirror's web page [11] which is a British national daily newspaper, this puzzle called World's hardest Sudoku. The 7th and 8th puzzles are taken from a web page that publish daily Sudoku puzzles [12], these are called hard and absurd puzzles of the day 31 August 2014. The 9th puzzle is used by Daniel D.

Friesen in his article [3] which is named evil puzzle that means very hard. The 10th puzzle, is obtain from Muhammad Asif's article [9]. The solution table shows that the algorithm can solve the hardest Sudoku puzzles in a few seconds or less than a second.

TABLE I. SOLUTION TIMES OF THE SUDOKU PUZZLES BY HYBRID ACO ALGORITHM.

Sudoku Puzzle Number	Time in Seconds
Puzzle 1	<1
Puzzle 2	1
Puzzle 3	3
Puzzle 4	<1
Puzzle 5	<1
Puzzle 6	2
Puzzle 7	1
Puzzle 8	1
Puzzle 9	1
Puzzle 10	<1

IV. CONCLUSION

In this paper a new hybrid algorithm is proposed to solve even hardest Sudoku Puzzles in a few seconds. Results show that the algorithm is a very powerful and efficient method to solve these puzzles. Although previous sole ant colony optimization algorithms implementation to the Sudoku puzzles weren't successful, this study shows that with an analytical part, a hybrid ACO algorithm can be a very successful method to solve this type of puzzles. The algorithm hybrid structure consist both analytical and heuristic parts so I hope this algorithm may help to Scientist to develop new hybrid algorithms that has similar properties. Additionally, I had used negative pheromone which is also a new aspect for the Ant colony optimization algorithms. I hope this study will be help implementation of the new ACO algorithm to other problems and development of the new efficient algorithms.

REFERENCES

[1] Sullivan, F.: Whip until solved. Computing in Science & Engineering. 12(1), (2010).

[2] Seta, T. Yato and T.: Complexity and Completeness of Finding another Solution and Its Application to Puzzles. IEICE Trans. Fundamentals Electronic. E86-A(5), (2003).

[3] Friesen, D., Patterson, M., Harmel, B.: A Spreadsheet Optimization Model for Solving Sudoku Problems. Business Management Dynamics. 2(9). (2013).

[4] Aaronson, L.: Sudoku Science. IEEE Spectrum. 43(2), p16-17, (2006).

[5] Maire,S., Prissette, M.: A restarted estimation of distribution algorithm for solving sudoku puzzles. Monte Carlo Methods Appl. 18, 147–160, (2012).

[6] Gunther, J., Moon, T.: Entropy Minimization for Solving Sudoku. IEEE transactions on signal processing 60(1), (2012).

[7] Yusiong,J. P., Seno, G.M.: SudokuBee: An Artificial Bee Colony-based Approach in Solving Sudoku puzzles. International Journal of Advance Research in Computer Science. 1(13), (2010).

[8] Mullaney,D.: Using Ant Systems to Solve Sudoku Problems. http://ncra.ucd.ie/COMP30290/crc2006/mullaney.pdf

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

[10] Baykasoglu, A., Dereli, T., Sabuncu, I., An ant colony based algorithm for solving budget constrained and unconstrained dynamic facility layout problems, Omega: International Journal of Management Science, 34(4), 385-396, 2006.

[11] World's Hardest Sudoku. Daily Mirror: http://www.mirror.co.uk/news/weird-news/worlds-hardest-sudoku-can-you-242294

[12] Sudoku of the Day: www.sudokuoftheday.co.uk

APPENDIX : SUDOKU PUZZLES

Puzzle 1

1	2	7	6	0	0	4	8	5
8	4	0	1	0	5	0	0	7
0	9	5	7	4	0	3	0	2
2	6	9	0	0	0	5	0	0
0	0	0	8	5	0	6	4	0
0	5	0	0	7	0	2	0	1
3	1	4	0	0	0	0	2	0
0	0	6	2	3	7	0	0	0
0	0	0	0	6	0	8	5	0

Puzzle 6

0	0	5	3	0	0	0	0	0
8	0	0	0	0	0	0	2	0
0	7	0	0	1	0	5	0	0
4	0	0	0	0	5	3	0	0
0	1	0	0	7	0	0	0	6
0	0	3	2	0	0	0	8	0
0	6	0	5	0	0	0	0	9
0	0	4	0	0	0	0	3	0
0	0	0	0	0	9	7	0	0

Puzzle 2

7	5	0	9	8	0	0	0	0
6	0	0	0	5	0	0	0	8
0	0	0	0	0	0	4	2	0
0	0	0	3	9	5	0	0	0
0	2	3	0	0	0	0	8	1
0	0	0	8	0	0	0	5	0
0	0	4	0	0	0	3	0	0
0	0	0	0	7	9	0	0	0
0	0	8	0	0	0	0	1	2

Puzzle 7

0	0	0	0	6	8	0	0	0
0	2	0	7	0	0	5	0	0
0	0	0	0	4	0	0	2	0
8	0	0	4	0	0	0	0	3
3	0	0	0	8	9	0	7	0
4	6	1	0	0	0	0	0	0
0	7	6	0	0	0	9	0	0
0	0	0	0	0	0	0	0	8
0	0	0	0	0	1	6	0	0

Puzzle 3

8	4	2	0	0	0	0	0	0
0	0	0	5	0	1	7	0	0
0	0	0	0	0	0	3	8	0
9	5	0	0	0	0	0	0	2
0	0	0	0	5	0	0	0	0
0	0	0	9	0	0	0	4	6
1	0	0	0	0	7	4	0	0
0	0	8	0	6	0	0	0	0
0	0	4	0	0	0	0	3	8

Puzzle 8

2	0	0	0	9	3	7	0	0
5	0	8	0	0	0	0	0	0
0	6	7	0	0	0	0	0	0
0	9	0	0	0	4	0	2	5
0	0	0	0	0	0	9	0	7
0	0	0	0	8	0	0	0	0
0	0	0	0	0	5	4	0	0
0	0	0	3	0	1	0	5	0
0	7	0	8	0	0	0	6	0

Puzzle 4

5	0	9	6	0	0	0	0	0
0	3	0	8	0	7	9	2	0
0	0	0	3	0	0	8	0	0
0	0	0	0	1	6	0	8	0
0	5	0	0	0	0	0	1	0
0	0	0	0	0	0	0	3	2
1	0	4	0	3	0	0	0	0
0	0	6	7	0	9	0	0	0
0	0	0	0	0	0	0	0	3

Puzzle 9

6	0	0	0	4	0	0	1	0
0	1	0	0	0	0	0	0	3
0	0	2	0	0	8	0	4	0
0	2	0	0	0	0	0	0	4
0	0	7	3	8	2	6	0	0
5	0	0	0	0	0	0	2	0
0	9	0	5	0	0	1	0	0
4	0	0	0	0	0	0	7	0
0	5	0	0	9	0	0	0	2

Puzzle 5

0	0	4	0	8	6	0	0	0
9	0	3	4	7	0	1	0	0
0	8	2	5	0	0	0	6	7
0	9	0	8	0	0	3	0	2
0	5	0	0	0	0	0	4	0
2	0	6	0	0	1	0	5	0
3	4	0	0	0	9	2	7	0
0	0	5	0	3	4	8	0	6
0	0	0	7	1	0	5	0	0

Puzzle 10

0	3	0	0	4	6	2	0	0
8	0	0	3	1	0	7	4	0
0	2	0	0	0	8	0	0	0
4	1	0	0	0	0	6	0	0
0	0	0	0	7	1	8	5	2
5	8	2	0	3	0	0	7	4
3	0	1	5	0	4	9	2	0
0	0	5	0	6	7	0	3	0
0	4	8	2	9	0	5	0	7