Application of Hill Climbing algorithm in Artificial Intelligence

Sathisha.G[1], Sathisha.G[2], Pragathi S.B[3], Soujanya G.K[4]
Department of Computer Science Engineering
[1,2,3,4]Atria Institute of Technology, Bangalore

Abstract- *In this paper, ß-Hill Climbing algorithm, the recent local search-based meta-heuristic, are tailored for Sudoku puzzle. ß-Hill Climbing algorithm is a new extended version of hill climbing algorithm which has the capability to escape the local optima using a stochastic operator called ß-operator. The Sudoku puzzle is a popular game formulated as an optimization problem to come up with exact solution. Some Sudoku puzzle examples have been applied for evaluation process. The parameters of the ß-Hill Climbing is also studied to show the best configuration used for this game. ß-Hill Climbing in its best parameter configuration is able to find solution for Sudoku puzzle in 19 iterations and 2 seconds.*

Keywords- ß-Hill Climbing; Optimization; Sudoku puzzle; Artificial Intelligent; local search.

I. INTRODUCTION

Optimization is an important field in decision support systems where the problem of finding the maximum profit or minimum cost is considered. In a nutshell, optimization concerns with finding the optimal values of the decision variables evaluated against certain objective function. The solution, included the optimal values for the decision variables, is called the "optimal solution". That solution is not trivially found where a strong method with a maneuver movement shall be used. This kind of algorithms is called optimization algorithms [1].

Optimization algorithms are normally initiated with stochastic solution(s). Iteratively, the stochastic solution(s) will be improved using intelligent operators with a machine learning mechanism that is able to drive the search toward the optimality. These operators normally exploit the current solution using a heuristic movement and exploring new areas where other solution with better quality can be found. By means of this, two search concepts (exploration and exploitation) must be included in any optimization algorithm [2].

Over the years, several optimization algorithms are developed. The earliest were based on the calculus-based methods in which the exact solution of any optimization problem is found based on some mathematical steps exhaustively processed. Although this kind of methods is useful in finding exact solution, the time complexity cannot be easily bounded for the highly dimensional problems.

Quit recently, meta-heuristic methods have been presented to deal with optimization problems of high dimensions. Meta-heuristic methods are general template for many optimization problems. They are categorized into either local search-based methods or population-based algorithms (i.e., evolution-based algorithms and swarm-based algorithms) according to the number of solutions processed in every iteration. Population-based methods initiated with a group of solutions called population. At each generation, those solutions are recombined and mutated to generate a new improved population. In contrast, the local search-based algorithms start with a single stochastic solution. At each iteration, that solution is moved from one point to another searching for better neighbouring solution that could be adopted during the iterative processes.

The two search concepts of exploration and exploitation must be balanced in all successful meta-heuristic method [3]. The problem search space has all possible solutions distributed in regions where in each region; there is a connected graph that must be tracked using exploitation concept which normally trapped in local optima. The ultimate goal of any meta-heuristic methods is to find the global optimal solution by means of jumping from one region to another using exploration concept. The operators of meta-heuristic methods are normally concerned with either exploration or exploitation. Population-based algorithm is more concern with exploration where local search-based method is tends towards exploitation.

Hill climbing algorithm, the origin of any local search algorithm, starts with stochastic solution. It climbs the hill by means of only accepting a better neighboring solution than the current one. It considered being a pure exploiter algorithm with no exploration capability. Several extensions to hill climbing have been developed to include an exploration-oriented operators such as simulated annealing [4], Tabu search [5], variable neighbourhood search [6], iterated local search [7], GRASP [8]. Quit recently, ß-Hill Climbing algorithm have been proposed by Al-Betar [9] for tackling any

optimization problem. ß-Hill Climbing is a new extension of hill climbing utilized a premium operator called ß-operator that has capability of empowering the search with the exploration concept to escape trapping in local optima. The ß-operator is a constructed with a process in which the values of the decision variables in the current solution are either copied from the neighboring solution or stochastically reset. It is worth mentioning that this algorithm can be easily tailored to a wide range of optimization problems.

In the searching area of artificial intelligence, the game is the most popular search problems that can be impressively used to clarify the behavior of the searching algorithm such as ß-Hill Climbing. One popular game in Arab countries and Europe countries are Sudoku puzzle. The Sudoku puzzle consists of 9 × 9 grid and 3 × 3 blocks for all the 81 cells. The solution of this are normally initiated with few discrete numbers from the range {1, 2,…, 9} randomly distributed. The dilemma is how to distribute the rest of numbers in the blank cells in the conditions that the numbers in the rows, columns and blocks shall be a complete set of the numbers {1, 2,…,9} without redundancy. Several algorithms that tackled different Sudoku games as a form of optimization problem are harmony search [10], genetic algorithm [11, 12], simulated annealing [13], artificial bee colony [14], tabu search [15], and particle swarm optimization [16].

The main objective of this paper is to tailor the ß-Hill Climbing algorithm for Sudoku puzzle game. The Sudoku puzzle game is initially formulated as an optimization problem. This problem is tackled using some real world examples with a small number of violations for hard problems and exact solution for easy one. The parameters of the ß-Hill Climbing algorithm are also studied using various values of ß and bw parameters. To the best of our knowledge, this is the first investigation of using ß-Hill Climbing algorithm for Sudoku puzzle.

The rest of the paper is as follows: Sect. II provides the description and formulation of Sudoku puzzle game, whereas, the description of the ß-Hill Climbing algorithm is provided in Sect. III. Next, the experimental results are presented in Sect. IV. Finally, conclusions and future directions are mentioned in Sect. V.

II. SUDOKU PUZZLE GAME

A. Game definition

The Sudoku puzzle is a popular game across the globe which is a fixed part of most newspapers and magazines. The Sudoku puzzle is represented in a board of size 9X9. That board is subdivided into sub-grids (or blocks) of size 3×3 as shown in Fig. 2. The cells in the board are either pre-fulfilled by fixed digits that cannot be changed or empty to be fulfilled by the solver. The solution of the Sudoku puzzle is reached when the empty cells are fulfilled by a suitable numbers where the complete solution must satisfy the following constraints:

1) Each row must contain digits from 1 9 exactly once.

2) Each column must contain digits from 1 9 exactly once.

3) Each 3 x 3 grid must contain digits 1 9 exactly once.

Indeed, the Sudoku puzzle can be classified in terms of difficulty into three categories: easy, medium, and hard. This is due to the number of given digits in the initial board. As the number of given digits increases, the difficulty of the

Sudoku puzzle is decreases. In terms of optimization, Sudoku puzzle is a combinatorial optimization problem and it is NP-complete in almost all its variations [17]. The main objective is how to find the unique solution for the Sudoku puzzle of the given digits.

B. Solution representation

The solution of Sudoku puzzle is represented as a matrix of size 9X9 as follows:

$$X = \begin{matrix} x_{11} & x_{12} & x_{19} \\ x_{21} & x_{22} & x_{29} \\ x_{91} & x_{92} & x_{99} \end{matrix}$$

Where the cell have initially either given digit or value of 0. It should be noted that in the final solution all cells are fulfilled by integer numbers from the range of {1,2, ..., 9}.

C. Objective function

The objective function is formulated in a way that the three constraints of the Sudoku puzzle are manipulated through the process of searching for the solution. The equation bellow is a cumulative formation of three parts; the first part calculates the violation in the vertical rows; the second in the horizontal columns; while the third in the blocks ( ). The objective function is formulated as follows:

$$\text{Min } Z = \sum_{i=1}^{9}\left|\sum_{j=1}^{9} x_{ij} - 45\right| + \sum_{j=1}^{9}\left|\sum_{i=1}^{9} x_{ij} - 45\right| + \sum_{k=1}^{9}\left|\sum_{(l,m)\in B_k} x_{lm} - 45\right| \quad (2)$$

where is the digit in the cell occurs at raw i and column

j. is the set of coordinates for block k. Apparently the summation of the digits in each row, column, and block is equal to 45 which is the summation of the integer numbers from 1 to 9. The solution is reached when the value of Z=0. By means of this, the solution is guaranteed to have the digits from 1 to 9 without repeating any in each raw, column and grid.

III. ß-HILL CLIMBING ALGORITHM FOR SUDOKU PUZZLE

ß-Hill Climbing is the recent metaheuristic algorithm. It is an extended version of hill climbing (i.e., the base version of the local search algorithm). In this section, ß-Hill Climbing is pseudo-coded in Algorithm 1 and the folwchart that shows how ß-Hill Climbing algorithm is processed through the search is given in Fig. 1.

As shown in ß-Hill Climbing pseudo-code at Algorithm 1 Step1, the search is initiated with a totally random solution. Initially, the algorithm generates integer numbers in the range of 1 - 9 and fulfills the empty cells by these generated digits. Note that the initial solution may have replicate digits in each row, columns and block. However, the given digits that are predefined in advance cannot be changed or moved. The initial solution is evaluated using the objective function discussed in Eq. (2). Thereafter, the algorithm will process the improvement loop.

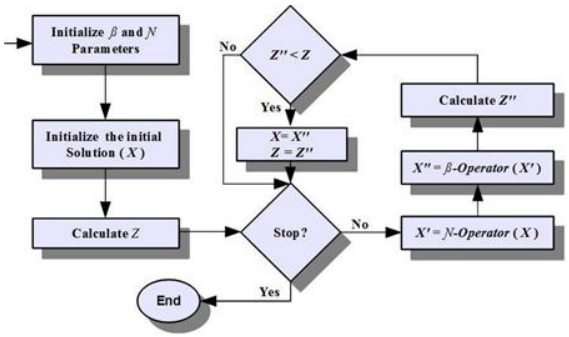


Fig. 1. ß-Hill Climbing algorithm flowchart.

Algorithm 1 –hill climbing pseudo-code

1. {Initialize X}
2. Calculate
3. itr =0
4. while (itr ≤ Max _Iterations) do
5. if (rnd)then
6. = -operator ( )
7. if (rnd ß)then
8. = ß--operator ( )
9. end if
10. end if
11. Calculate
12. if () then
13. =
14. =
15. end if
16. itr= itr+1
17. end while

In the Improvement loop of ß-Hill Climbing algorithm, two operators are executed: Neighboring Operator (i.e., - is improved by navigating one neighboring solution .operator)andß-operator.In-operator,thecurrentsolution

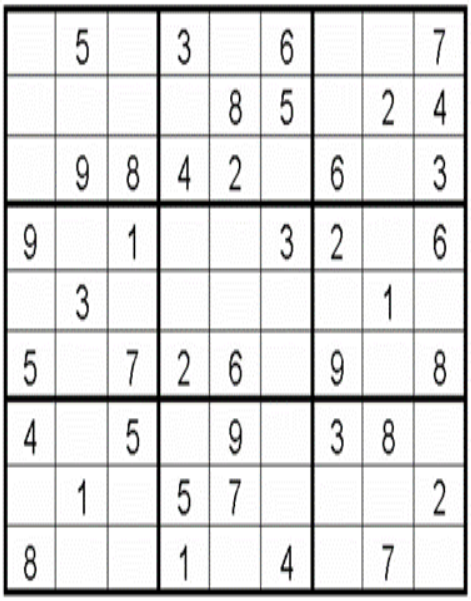
The improvement is done by randomly selecting set cells that met the probability of where from the board then by adding one to or subtracting one from the digits in these cells in the condition the digits do not exceed the limit. This operator can be considered the source of exploitation in the algorithm where the current values are used in the process of generating the neighboring value.

In ß-operator, all cells in the improved Sudoku board is scanned for weather or not to be regenerated with a probability range of ß where ß (0,1). In case the probability range of ß is reached, the cell will take new randomly generated integer number from the range 1- 9. This operator can be considered as the source of exploration where the current selected digits are randomly regenerated from the scratch without relying on the existing digits. The resulting solution from this operator is .

The objective function of the solution is calculated. Then the original value of the objective function for the current solution is compared with that of . In case the is better than , the improved solution replaces the current one and its objective function will replace . This process is keep repeated until the Max_iteration value is reached.

IV. EXPERIMENTAL RESULTS

In this section, the performance of the ß-Hill Climbing algorithm for Sudoku puzzle is experimentally evaluated. Fig. 2 shows the benchmark of the Sudoku which is taken from [18]. Again all cases of Sudoku puzzle have a unique solution.



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

Fig. 2. Example of easy sudoku puzzle problem.

The two parameters of the proposed algorithm are studied using various values as shown in Table I. This table includes the values of the two parameters (i.e., ß, and), the minimum time consumed to solve the problem, the minimum number of iterations to reach the solution, and rate of success to solve the Sudoku game over 10 runs.

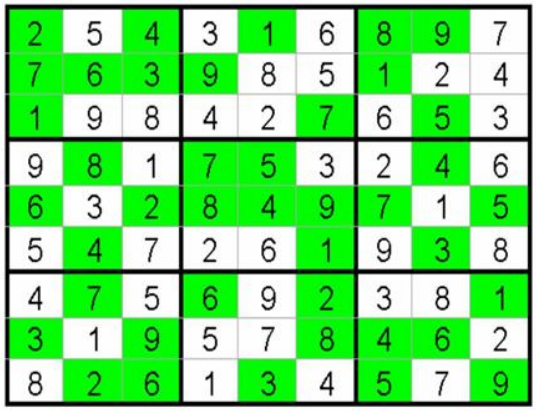
TABLE I. RESULTS OF ß-HILL CLIMBING WITH ITS

COLLABORATIVE PARAMETERS

Scenario	Parameter values		Results		
		ß	*Iterations*	*Time(s)*	*Success Rate (%)*
Sen#1	0.01	0.5	19	2	100
Sen#2	0.1		35	2	100
Sen#3	0.3		25	2	100
Sen#4	0.5		38	6	100
Sen#5	0.7		155	3	100
Sen#6	0.9		1289	10	90
Sen#7	0.3	0.01	5299	39	10
Sen#8		0.1	91	9	40
Sen#9		0.3	14	4	100
Sen#10		0.5	25	2	100
Sen#11		0.7	145	6	100
Sen#12		0.9	163	3	100

The parameter is studied with various values in six experimental scenarios (i.e., Sen#1 – Sen#6). The value of the ß parameter is fixed to 0.5. The higher value of parameter leads to higher rate of exploitation. It can be observed from the results, the parameter with lower values leads to better results, where the first three scenarios solve the game within two seconds and minimum number of iterations. The first experimental scenario (i.e., Sen#1) solves the problem within two seconds and 19 iterations. Furthermore, the third scenario (i.e., Sen#3) solves the game within two seconds and the best average of iterations over 10 runs. The parameter is set to 0.3 in the following experimental scenarios.

Similarity, the ß parameter is studied with different values using six experimental scenarios (i.e., Sen#7 – Sen#12). It should be noted that the higher the value of ß parameter the higher of exploration will be. Apparently, the experimental scenario (Sen#10) solves the game within two seconds and with the minimum number of iterations. Fig. 3 provides a solution for Sudoku puzzle given in Fig 2.



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

Fig. 3. A solution for Sudoku puzzle given in Fig. 2.

V. CONCLUTION AND FUTURE WORK

In this paper, ß-Hill Climbing, a recent meta-heuristic local search algorithm, is investigated for Sudoku puzzle. ß-Hill Climbing algorithm is an extended version of hill climbing algorithm in which a new operator called ß-operator is introduced. This operator empowers the algorithm to escape the local optima by means of adding exploration capability to the basic hill climbing. Sudoku puzzle is a popular game in the artificial intelligent field where the player attempt to fulfill the empty cells of a board of size 9X9 by a digits from the range 1-9 in the condition that these digits are distributed in each row, columns, and block of size 3X3 without repeating any.

ß-Hill Climbing is evaluated using a Sudoku puzzle example taken from Geem [18]. The algorithm parameters (i.e., ß and) are carefully studied to show their effect in the convergence behavior of ß-Hill Climbing. Note that the ß parameter controls the size of exploration in the search while the parameter controls the exploitation. In a nutshell, the algorithm efficiently works when the values of and ß are low. It is noted that the exploration is increased when the value of

is increased where the number of local changes in the current solution is increased without guide form the objective function, therefore the exploration will increase.

As shown by the results, the convergence capability of ß-Hill Climbing algorithm is tested using easy example of Sudoku puzzle. In the future, ß-Hill Climbing can be experimented with different versions of different difficulty level of Sudoku puzzle (i.e., easy, medium, and hard).

REFERENCES

[1] BoussaïD, J. Lepagnot, and P. Siarry, "A survey on optimization metaheuristics," Information Sciences, vol. 237, pp. 82-117, 2013.

[2] K. Sörensen, "Metaheuristics—the metaphor exposed," International Transactions in Operational Research, vol. 22, pp. 3-18, 2015.

[3] C. Blum, J. Puchinger, G. R. Raidl, and A. Roli, "Hybrid metaheuristics in combinatorial optimization: A survey," Applied Soft Computing, vol. 11, pp. 4135-4151, 2011.

[4] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simmulated annealing," science, vol. 220, pp. 671-680, 1983.

[5] F. Glover, "Future paths for integer programming and links to artificial intelligence," Computers & operations research, vol. 13, pp. 533-549, 1986.

[6] P. Hansen and N. Mladenovi , "Variable neighborhood search: Principles and applications," European journal of operational research, vol. 130, pp. 449-467, 2001.

[7] H. R. Lourenço, O. C. Martin, and T. Stützle, "Iterated local search," in Handbook of metaheuristics, ed: Springer, 2003, pp. 320-353.

[8] T. A. Feo and M. G. Resende, "Greedy randomized adaptive search procedures," Journal of global optimization, vol. 6, pp. 109-133, 1995.

[9] M. A. Al-Betar, "\ beta-Hill climbing: an exploratory local search," Neural Computing and Applications, pp. 1-16. doi:10.1007/s00521-016-2328-2

[10] Z. W. Geem, "Harmony search algorithm for solving sudoku," in International Conference on Knowledge-Based and Intelligent Information and Engineering Systems, 2007, pp. 371-378.

[11] X. Q. Deng and Y. Da Li, "A novel hybrid genetic algorithm for solving Sudoku puzzles," Optimization Letters, vol. 7, pp. 241-257, 2013.

[12] T. Mantere and J. Koljonen, "Solving and rating sudoku puzzles with genetic algorithms," in New Developments in Artificial Intelligence and the Semantic Web, Proceedings of the 12th Finnish Artificial Intelligence Conference STeP, 2006, pp. 86-92.

[13] R. Lewis, "Metaheuristics can solve sudoku puzzles," Journal of heuristics, vol. 13, pp. 387-401, 2007.

[14] J. A. Pacurib, G. M. M. Seno, and J. P. T. Yusiong, "Solving sudoku puzzles using improved artificial bee colony algorithm," in Innovative Computing, Information and Control (ICICIC), 2009 Fourth International Conference on, 2009, pp. 885-888.

[15] R. Soto, B. Crawford, C. Galleguillos, E. Monfroy, and F. Paredes, "A hybrid ac3-tabu search algorithm for solving sudoku puzzles," Expert Systems with Applications, vol. 40, pp. 5817-5821, 2013.

[16] A. Moraglio and J. Togelius, "Geometric particle swarm optimization for the sudoku puzzle," in Proceedings of the 9th annual conference on Genetic and evolutionary computation, 2007, pp. 118-125.

[17] Y. Takayuki and S. Takahiro, "Complexity and completeness of finding another solution and its application to puzzles," IEICE transactions on fundamentals of electronics, communications and computer sciences, vol. 86, pp. 1052-1060, 2003.

[18] M. Nicolau and C. Ryan, "Solving sudoku with the gAuGE system," in European Conference on Genetic Programming, 2006, pp. 213-224.