IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 3, Ver. III (May-Jun. 2014), PP 01-07
www.iosrjournals.org

An Optimized Algorithm for Solving Combinatorial Problems using Reference Graph

Raktim Chakraborty[1],Saubhik Paladhi[2],Sankhadeep Chatterjee[3], Soumen Banerjee[4]

[1234]*Hooghly Engineering & Technology CollegeVivekananda Road, Pipulpati, Hooghly-712103, West Bengal, India*

***Abstract*:** *In this paper a novel optimized graph referencing method is proposed to tackle combinatorial problems with greater ease. The proposed algorithm is a modification of the existing backtracking algorithm. It takes into account the advantage of backtrack algorithm while eliminating its disadvantages. Special care has been taken to reduce the time complexity in avoiding frequent unsuccessful searches and in providing accurate and fast solution to the targeted sub problem. The theoretical analysis and experimental results presented reveal the superiority of proposed algorithm over the conventional algorithms.*

Keywords:*Backtracking, Combinatorial problem, Graph Referencing algorithm, Optimization, Sudoku.*

I. Introduction

A special branch of mathematics known as Combinatorics deals with the study of finite structure including counting the structure of a given kind and size, depending upon criteria to be met and analysis and constructs objectives meeting such criteria. Combinatorial problems find extensive application in fields of mathematics, machine learning, AI, Auction Theory, Software Engineering etc. Backtracking is a general algorithm for finding all or some solution to some combinatorial problems. However, Backtracking suffers inherent disadvantages in terms of conjunction of a large processing time and memory space. It is here that the proposed novel optimized graph referencing method established its superiority. The algorithm as proposed by the authors, which is a modification of conventional Backtracking algorithm, not only solves efficiently different combinatorial problems but does so in the least possible time.

Tackling various combinatorial problems is a subject of research worldwide. Different efficient algorithms are being developed to this purpose. Xiao et. al. [1] designed a stepwise enumerative algorithm while another algorithm based on GA was developed by Liu et. al. [2]. The Backtracking algorithm is reported to be used to solve Sudoku, a combinatorial problem, by Xu [3]. The solution to the Sudoku problem was proposed by Zhang [4] who intend to use the application development language AutoLISP embedded in AutoCAD for this purpose. An altogether different algorithm to solve Sudoku was proposed by Zhao et. al. [5] while the same was tackled with the help of cultural algorithm and GA optimization as reported by Mantere and Kolijonen [6-7]. The underlying science behind combinatorial problem was reported by Delahaye [8] while structural engineering problems were solved using an innovative yet powerful search and optimization method by Lee and Geem [9-10]. An improved and modified version of graph matching algorithm was developed by Cordella et. al. [11]. The DLX algorithm (using Dancing Link) implements natural algorithm for exact cover problems in an effective manner towards a faster solution as put forwarded by Knuth [12]. Several other combinatorial problems was solved by a novel technique proposed by Hitotumatu and Noshita [13]. Thus the conventional Backtracking algorithm finds extensive application in the field of science and technology. Thus the authors inspired by the Backtracking algorithm took a challenge to design an absolute new algorithm eliminating the disadvantages of the Backtracking algorithm for solving the various types of combinatorial problems, thereby opening an absolute new window to the science and engineering fraternity for extensive research in the domain. The present work will be very helpful in analysing and solving combinatorial problems and would be much beneficial to the research community.

II. Implementation of the Reference Graph

Several fields like Artificial Intelligence, Auction Theory, Software Engineering etc. have applications related to Combinatorics. The target is to find the best or optimized result from a finite set of results. The authors consider here a set of Sudoku problems as a typical example of combinatorial problems and aims to solve it efficiently keeping in mind space and time complexity.

In generalized format a Sudoku has N number of row and columns such that the N x N grid is partitioned into N number of $\sqrt{N}$ x $\sqrt{N}$ boxes or blocks. Each row, column and box has N number of cells with a total number of cells in the grid being N^2. As per rule of Sudoku, a digit cannot exist in any cell of same row or

same column or same box more than once. Fig.1depicts a 4 x 4 Sudoku cell's connection graph and it is observed that there are total 16 nodes. If any two node V_i and$V_j$are connected, there will be an edge between V_i and V_j. Thus any element of the set can influence the other elements of the same. For instance in Fig..1 node 1 is linked with the set A= {0,2,3,4,5,9,13} i.e. the digit in the cell 1 will not be placed in any node belongs to set A. Moreover a cell under consideration is not connected with itself. Fig. 2 depicts a typicalReference Graph Matrix for a Reference Graph depicted in Fig. 1.

The particular cell of the Sudoku matrix is generated by the relation:-
*Index = (row * order of Sudoku grid + column)*

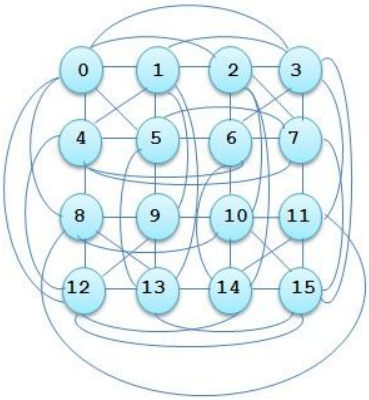

Fig. 1. Reference Graph for 4 x 4

column index

node index	0	1	2	3	4	5	6
0	1	2	3	4	8	12	5
1	0	2	3	5	9	13	4
2	0	1	3	6	10	14	7
3	0	1	2	7	11	15	6
4	5	6	7	0	8	12	1
5	4	6	7	1	9	13	0
6	4	5	7	2	10	14	3
7	4	5	6	3	11	15	2
8	9	10	11	0	4	12	13
9	8	10	11	1	5	13	12
10	8	9	11	2	6	14	15
11	8	9	10	3	7	15	14
12	13	14	15	0	4	8	9
13	12	14	15	1	5	9	8
14	12	13	15	2	6	10	11
15	12	13	14	3	7	11	10

Fig. 2. Reference Graph Matrix

The main key factors for time management and efficiency of the solution are the Reference Graph & the Referenced Graph matrix. The pre-produced Reference Graph matrix has been proposed in anefficient way such that the problem solving time will be reduced as it places the appropriate digit in a cell by searching its absence in all other cells which are connected with each other.

III. The Proposed Algorithm

In this section an algorithm has been proposed that solves combinatorial problems. At first a Referenced Graph matrix for this problem is to be produced. As the size of the matrix varies with the order of the problem (problem with higher order causes higher order matrix production), the authors have decided to propose a generalized algorithm which produces the connection matrix for a specific problem.

The generalized algorithm to produce Referenced Graph matrix is:-
* In algorithm order is the grid order of the problem. Graph [ ] [ ] is the matrix which is representing the connection between the nodes and box no is the index of box under consideration. *

```
Algorithm: Graph Referenced Matrix ()
1)      repeat until k: = order² -1
do row ← k / order
        col ← k mod order
        count ← 0
2)      repeat for j: = 0 to order -1
do
        p ← row * order + j
        check if p = row*order + col then
        continue
        endif
        Graph[k][count]←p
        count←count+1
Next j
3)      repeat for i: = 0 to order -1
do
        p←i*order+col
        check if p = row * order + col then
continueend if
```

```
Graph[k][count]←p
        count←count+1
Next i
4)      box no ← ((row / box order) * box order) + (col / box order)
5)      rowinitial ← (box no / √(order)) * √ (order)
6)      rowfinal ← rowinitial + (box order -1)
7)      columninitial ← (box no mod box order) * box order
8)      columnfinal ← columninitial + (box order -1)
9)      repeat for i: = rowinitial to rowfinal
10)     repeat for j: = columninitial to columnfinal
do p ← i * order + j
check if  p = row * order + col or (i = row or j = col) then
continue
end if
do Graph [k][count]←p
count← count + 1
Next j
Next i
11)     k ← k+1
wend
```

The steps for solving are as follows:-

1) Problem [x] [y] contains the total problem.
Problem [i] [j] = value (i^{th} row and j^{th} column cell)
Problem [i] [j] = 0 if i^{th}row andjth column cell is empty

2) Empty [z] stores the indexes of empty cells. Index = (row * order of Sudoku grid + column) .

3) Find the location of empty cell in Problem [ ] [ ] by following
s = Empty[m]
row = s / order
column = s % order where m $\in$ [0,z-1]
So Problem [row] [column] is empty.

4) Search from Graph [s] [0] to Graph [s] [count -1] to discard the elements which have already been assigned .

5) First valid digit is assigned to Problem [row] [column] .
If no valid digit is found then the above steps are repeated for m= m-1

6) For successful placement of element the steps will be repeated for m=m+1 until all the empty cells are filled with appropriate element to yield the solution.
7)

IV. Implementation of Proposed Algorithm

To implement the algorithm the authors have decided to choose a 4 x 4 Sudoku problem with 5 clues in Fig. 3 on which the operations will be done. Referenced Graph and Graph referenced matrix for a problem with 4 x 4 order have beenshown in Fig. 1 and Fig. 2. Graph referenced matrix can be derived from the above algorithm .
After deriving the matrix the algorithm enter into the next stage where valid digits are assigned to cells. The steps according to the algorithm are :

1) Problem [4] [4] will store the content of the problem .

```
Problem [ 4 ] [ 4 ] =   { 1, 0, 0, 4,
                          0, 4, 0, 0,
                          0, 0, 3, 1,
                          0, 0, 0, 0}
```

2) Empty [ ] will store only the index of an empty cell . Relation for giving index in particular cell is Index = (row * order of Sudoku grid + column) .

Here order of Sudoku grid is 4.
So Empty [ ] = {1,2,4,6,7,8,9,12,13,14,15} .

3) Here Empty[0] = 1 .
So row = 1 / 4 = 0 and column = 1% 4 = 1 . From this we can say the 0^{th} row and 1^{st} column element of the problem is empty.

4) Algorithm will search from Graph [1] [0] to Graph [1] [6] for the cell indexes which contain digits. Assigned digits will be discarded for the cell under consideration. 1^{st} row of the matrix is
|0 2 3 5 9 13 4|

The list of row and column element

Row	0/4 = 0	2 / 4 = 0	3 / 4 = 0	5 / 4 = 1	9 / 4 = 2	13 / 4 = 3	4 / 4 = 1
Column	0%4=0	2 % 4 = 2	3 % 4 =3	5 % 4 =1	9 % 4 =1	13 % 4 =1	4 % 4 =0
Value (Problem[row] [column])	1	0	4	0	0	0	0

5) So 1 and 4 will be discarded from the list of valid number for Problem [0] [1] . So set of valid digit for that cell is {2,3} .
6) 2 will be assigned to Problem [0] [1] i.e. Problem [0] [1] ← 2 which has been shown in Fig.4.
7) Algorithm will now go for Empty [1] element and like the above steps it will try to find invalid digits for Problem [Empty [1] / 4] [Empty [1] % 4] i.e. Problem [0] [2] and in this case it will be found that no valid digit exist for that cell as all the digits are already in the cells which are connected with it.
8) At this point backtrack procedure will take place by moving back to Problem [0] [1] and place next valid digit for that cell .In this case Problem [0] [1] ← 3 i.e. 3 will be assigned to that cell (shown in Fig.5)
9) Thus the algorithm will fill all the cells with valid digits. Solved problem has been shown in Fig.ure 6.

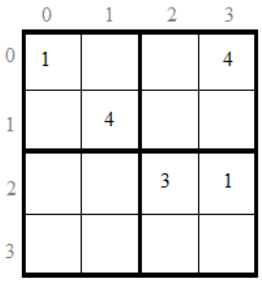

Fig. 3 : Main Problem assigned

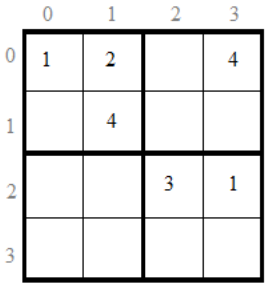

Fig. 4 : 2 is assigned

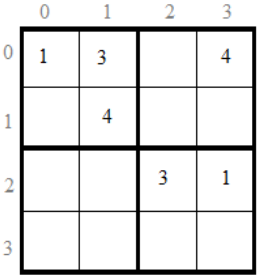

Fig. 5 : After backtracking 3 is

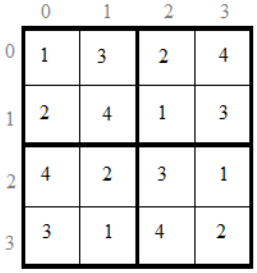

Fig. 6 : Complete solution

V. Results and Discussion

The proposed Graph Referencing algorithm is tested upon randomly selected combinatorial problems of various difficulty levels. The same set problems have also been solved using the conventional Backtracking algorithm and the various comparative results are depicted in Fig. 7 to Fig. 12.

Fig. 7 depicts a comparative study on problem solution time of easy level combinatorial problems. A plot of solution time vs. corresponding puzzles (Typical combinatorial problems under easy category) has been depicted in the graph and the same has been re-arranged in ascending order of solution time. Analysis reveals that much lesser solution time is required in the case of proposed Graph Referencing algorithm (Refer blue line of the graph). Several corresponding points are marked with alphabets in the Fig.7 which indicates the superiority of the proposed Graph Referencing algorithm over Backtracking algorithm. For example, for the 2^{nd} combinatorial problem the point 'B' depicts a solution time of 0.001s using Graph Referencing algorithm whereas solution time at point 'A' is 0.002s using Backtrack algorithm. The point 'C' for problem number '3' depicts a solution time of 0.006s whereas point 'D' depicts a solution time of 0.004s, thereby clearly a gain in solution time of approx. 0.002s using Graph Referencing algorithm is achieved. In the same way point 'E' depicts a solution time of 0.009s of 5^{th} problem using Backtracking algorithm, which precedes the solution time by 0.002s for solving the same by using Graph Referencing algorithm as depicted in point 'F'.

Fig. 8 depicts similar analysis which has been done on a set of randomly chosen medium level combinatorial problems. An analysis reveals that time requirement in solving a problem is pretty much lesser in case of use of Graph Referencing algorithm than that for Backtracking algorithm. For instance point 'A' depicts

solution time 0.028s for 5th problem using backtracking algorithm while point 'B' depicts solution time of 0.022s for the same using graph Referencing algorithm. Hence the later one takes 0.006s lesser time in solving the problem than the former one, clearly indicating the ingenuity in case of solving medium level combinatorial problems.

Fig. 9 depicts the same analysis for a set of randomly chosen hard level combinatorial problems. The point marked 'B', depicts solution time of 0.020s for the 4th problem of hard category using Graph Referencing algorithm, which takes 0.004s less than the solution time of the same depicted in the graph as 'A' using Backtracking algorithm. The analysis clearly indicates the superiority of the proposed Graph Referencing algorithm over all three categories of combinatorial problems.

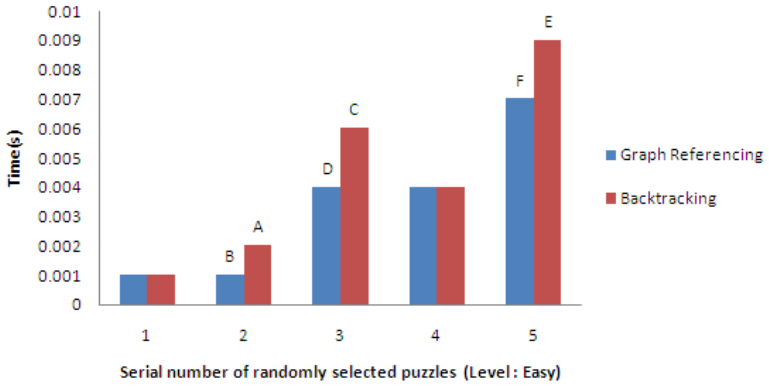

Fig. 7. Graph depicting time taken by randomly selected problmes of Easy category by Graph referencing algorithm & conventional Backtracking algorithm.

A set of 30 combinatorial problems comprising of various difficulty categories viz. easy, medium and hard are chosen randomly and the solution time using proposed Graph Referencing algorithm of the 30 randomly chosen problems set is depicted in Fig. 10 using scattered points in blue colour. A trend line (Straight line in blue) plotted from the scattered points is also depicted to indicate the overall nature of the time required to solve these problems using Graph Referencing algorithm.

Fig. 11 depicts the solution time using scattered points in colour red of the same set of 30 randomly selected combinatorial problems using Backtracking algorithm. The overall nature of the solution time has been depicted using a red trend line using plotted scattered points depicting solution time of different combinatorial problems. The equations of the trend lines are calculated to be

$$y = 0.001x - 0.0023 \quad \text{... (1)}$$

$$y = 0.001x - 0.0052 \quad \text{... (2)}$$

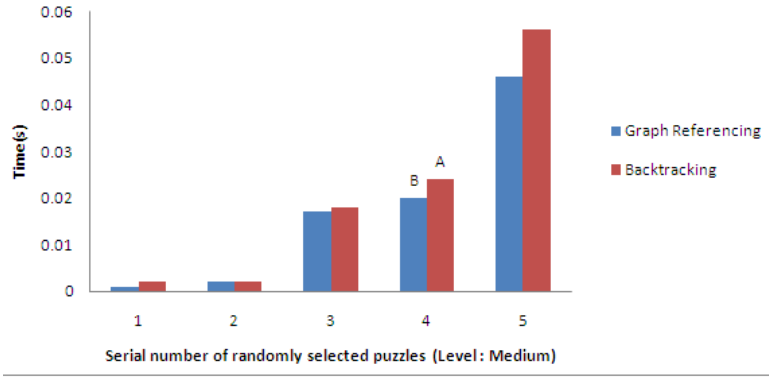

Fig. 8. Graph depicting time taken by randomly selected problmes of Medium category by Graph referencing algorithm & conventional Backtracking algorithm.

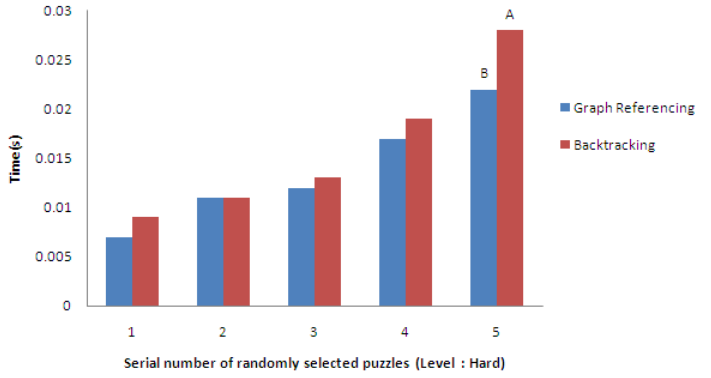

Fig. 9. Graph depicting time taken by randomly selected problmes of Hard category by Graph referencing algorithm & conventional Backtracking algorithm.

Further analysis of the graphs reveals that the straight lines are representing the overall nature of the solution time using two different algorithms. A detail meaning of the two straight lines are given by equation (1) & (2) what is actually depicted in Fig. 12.

A plot of the solution time for 30 different and randomly chosen problems of different standards (viz. easy, medium, & hard) is shown in Fig. 12 where the problems are solved using both Graph Referencing algorithm and Backtracking algorithm. The trend lines (in red and blue dashed lines) formed out of the scattered points are also depicted in the same graph. The equations of the trend lines are also shown in the graph. Thus it is evident that both straight lines are parallel to each other owing to identical slope values m = 0.001. However they differ in their intercept values c_1 = -0.0023 and c_2 = -0.0052. The graph clearly reveals that in all the cases the trend line in red always remains below the trend line in blue which is a vivid indication of the fact that lesser solution time is consumed for solving any type of combinatorial problems using Graph Referencing algorithm. It thus establishes the superiority in performance of the proposed Graph Referencing algorithm in comparison to the backtracking algorithm.

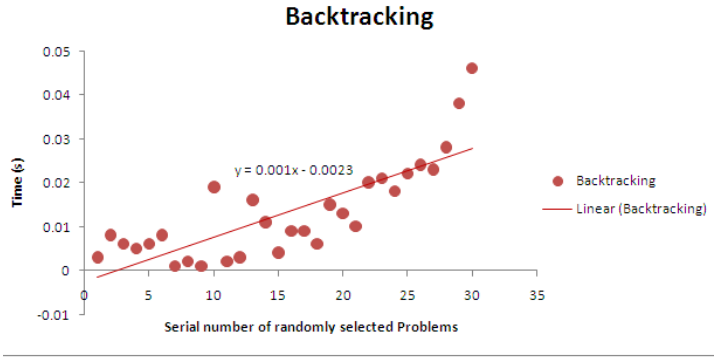

Fig. 10. Graph depicting time taken by randomly selected puzzles by Graph referencing algorithm.

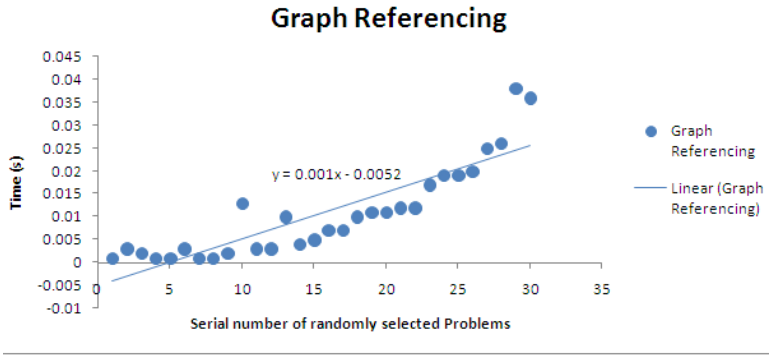

Fig. 11. Graph depicting time taken by randomly selected puzzles by Backtracking algorithm.

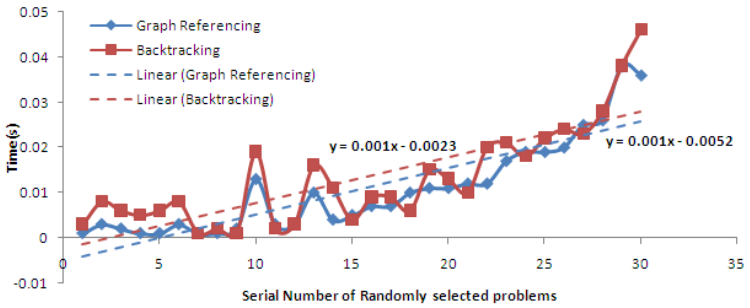

Fig. 12. Graph depicting time taken by randomly selected puzzles by both Backtracking algorithm and Graph Referencing algorithm.

VI. Conclusion

The present research work carried out by the authors in proposing the novel optimized Graph Referencing algorithm is a modification over the common Backtracking algorithm in efficiently solving any type of combinatorial problems using the least solution time possible. The results depicted in the paper will be very useful for the researchers and mathematicians in handling and solving combinatorial problems with ease in near future.

References

[1]. H. Xiao, Z. Tian and L. Ma, Design of stepwise enumerative algorithm based on rule about Sudoku, Computer Engineering and Design, 31(5), 2010, pp. 1035-1037.
[2]. Y. Liu and S. Liu, Algorithm based on genetic algorithm for Sudoku puzzles, Computer Science, 37(3), 2010, pp. 225-226.
[3]. J. Xu, Using backtracking method to solve Sudoku puzzle, Computer Programming Skills & Maintenance, 5, 2009, pp. 17-21.
[4]. Z. Zhang, Solve & generate Sudoku puzzle by programming in AutoCAD, Computer Programming Skills & Maintenance, 17, 2008 pp. 16-18.
[5]. Z. Zhao, J. Guo and L. Yang, Study of algorithm about Sudoku generation, Neijiang Science and Technology, 7, 2008, pp. 22-23.
[6]. T. Mantere and J. Koljonen, Solving and analyzingSudokus with cultural algorithms, Proc. of IEEE Congress on Evolutionary Computation (CEC 2008), IEEE, New York, 2008, pp. 4053-4060.
[7]. T. Mantere and J. Koljonen, Solving, rating and generating Sudoku puzzles with GA, Proc. of IEEE Congress on Evolutionary Computation (CEC 2008), IEEE, New York, 2007, pp. 1382-1389.
[8]. J. Delahaye, The science behind Sudoku, Scientific American, 294(6), 2006, pp. 80-87.
[9]. K. S. Lee, and Z. W. Geem, A new meta-heuristic algorithm for continuous engineering Optimization, Comput. Methods Appl. Mech. Engrg, 194, 2005, pp.3902–3933.
[10].] K. S. Lee, and Z. W. Geem, A new structural optimization method based on the harmony search algorithm, Computers and Structures, 82, 2004, pp. 781–798.
[11]. L. P. Cordella, P. Foggia, C. Sansone, M. Vento, An improved algorithm for matching large graphs. Proc. of the 3rd IAPR-TC-15 International Workshop on Graph based Representations, , Ischia, Italy, 2001, pp. 149–15.
[12]. Donald Knuth, Dancing links, Millennial Perspectives in Computer Science, 187, 2000, pp. 159.
[13]. Hitotumatu, Hirosi; Noshita, Kohei, A Technique for Implementing Backtrack Algorithms and its Application, Information Processing Letters, 8 (4), 1979, pp. 174–175.