P versus NP

Nikos Dimitris Fakotakis

***Abstract*— The generalized Sudoku problem, with S as the set of the symbols, is known to be $\mathcal{NP}$-complete and it is equivalent to any other $\mathcal{NP}$-complete problem. In this paper, we present a method that solves the Sudoku Problem in polynomial time and proves that $\mathcal{P} = \mathcal{NP}$.**

I. Introduction

The generalized Sudoku, with S as the set of the symbols ($s_i \in S$, with $i = 1, 2, ..., Q(S)$), has a Latin square, of size $Q(S) \times Q(S)$, where $Q(S)$ is the cardinal number of the S set. If $Q(S)$ is a perfect square, then the Latin square (Sudoku board) can be divided into $Q(S)$ regions, of size $\sqrt{Q(S)} \times \sqrt{Q(S)}$, called blocks. Sudoku problem has fixed values, in some of the cells. If only a unique solution remains, the Sudoku Problem is well-formed. It is considered as a difficult problem in the $\mathcal{NP}$ set, for which a positive solution can be certified in polynomial time. We developed and present an algorithm that finds the solution of the Sudoku problem, in polynomial time.

II. Sudoku Solution

Sudoku has three sets of constraints, that have to be simultaneously satisfied:

- Each of the $Q(S)$ blocks must contain each symbol, from s_1 to $s_{Q(S)}$, precisely once.
- Each of the $Q(S)$ rows must contain each symbol, from s_1 to $s_{Q(S)}$, precisely once.
- Each of the $Q(S)$ columns must contain each symbol, from s_1 to $s_{Q(S)}$, precisely once.

We enumerate the steps of the algorithm that finds the solution of the problem:

1) We examine the blocks, in order to find in which of the $Q(S)$ blocks, the s_i symbol of the $Q(S)$ symbols, does not appear (with $i = 1, 2, \ldots, Q(S)$).
2) In the first block n_j, in which the s_i symbol does not appear (with $j = 1, 2, \ldots, Q(S)$), there exists $Q(S)$ cells. In each cell of this block, we examine if the row or the column of the cell c_k (with $k = 1, 2, \ldots, Q(S)$), have the s_i symbol. If they do not have it, then we place the s_i symbol as a note (a probably correct symbol), in the c_k cell. We can have multiple notes in each cell. If the row or the column have the symbol, then we move to the next cell c_{k+1}, of the n_j block, until we examine every cell of the block.
3) Then, we go to step 2, replace the block n_j with the next block (n_{j+1}) and check if the block has the symbol. If it does not have it, we repeat the procedure from step 2, until we examine every block (with $n_{j=Q(S)}$ as the final block).
4) Then, we go to step 1, we replace the symbol s_i with the next symbol (s_{i+1}) and repeat until $i = Q(S)$.
5) Then, we go to step 1 and repeat the procedure, until we fill every cell with at least one symbol as a note.
6) Initially, we have presupposed that the Sudoku Problem has one solution. This means that in this state, the Latin square has to have at least one cell with only one symbol as a note. In each of these cells, we convert the note as a fixed value. Then, we go to step 1 and start the algorithm, having the new added fixed values in the Latin square. We repeat the steps, until we find the solution.

III. Conclusions

The initial, 21 $\mathcal{NP}$-complete problems, described by Karp [1], can be solved in polynomial time, by converting the Sudoku Problem to them.

References

[1] R. M. Karp, "On the computational complexity of combinatorial problems," Networks, vol. 5, no. 1, pp. 45–68, 1975.