A Complete Sudoku Game Based On Constraint Propagation Strategies Using MATLAB

Ankit Kumar Sinha, Nihit Raj, A.V Prabu ,Vishal Kumar Singh

Abstract--- Sudoku puzzles appear in many forms of digital and physical media. In fact, millions of people around the world know how to play Sudoku. All the same, the skill behind this game is a lot more complex than it seems. That is why many researchers have set a notable measure of exertion to generate efficient algorithms to figure out these puzzles. Some researchers might even propose that an algorithm to solve Sudoku games without trying a large amount of permutations. This report delineates the maturation and execution of a Sudoku solver using MATLAB. Furthermore, this text file includes detailed directions about the founding of a graphical user interface and the carrying out of a constraint-propagation algorithm, randomization and zero bias gerechte design of Sudoku. The mapping the dependent variable as the network reaches out to multiple variables is also described

Keywords: Constraint propagation algorithm, Gerechte Design, Randomization

INTRODUCTION

Puzzles and riddles have been a stage for utilization of arithmetic, counterfeit consciousness and other field strategies. The previous years have brought into consideration an extremely famous riddle called Sudoku. The run of the mill Sudoku riddle matrix is a 9-by-9 cells into nine 3-by-3 squares. The rules of the Sudoku are

Every row, column, and square (of 3-by-3) must be filled with each of the numbers 1 till 9 and that number cannot appear more than once in any of the row, column, or square. Sudoku has led other researchers to some advances in algorithm design and implementation. This study was mostly motivated by the interesting mathematical concepts behind it.

Sudoku is a logic based number placement puzzle. The aim is to populate a 9x9 grid in such a way that each column, row, and 3x3 box contains a single availability of the digits 1-9. The game goes with a partially completed grid, and the answer to the puzzle is the arrangement of digits that come across the single-instance criteria.

A minirow consists of three cells forming a row of a sub square, and a minicolumn consists of three cells forming a column of a sub square. We define a broken row to be the union of three minirow occurring in the same position in three sub squares in a column, and a broken column to be the union of three mini columns occurring in the same position in three sub squares in a row. A location is a set of nine cells occurring in a fixed position in all of the sub squares (for example, the center cells of each sub square).

Now a symmetric Sudoku solution is an arrangement of the symbols 1. . . 9 in a 9 × 9 grid On such an approach that every image happens once in each row, column, sub square, broken row, broken column or anyplace in the grid inside the bolder limits. In other words, it is a multiple gerechte design for the partitions into sub squares, broken rows, broken columns, and locations.

The strategy behind using a model checker to solve a Sudoku puzzle is this: formulate a logical proposition that suggests,

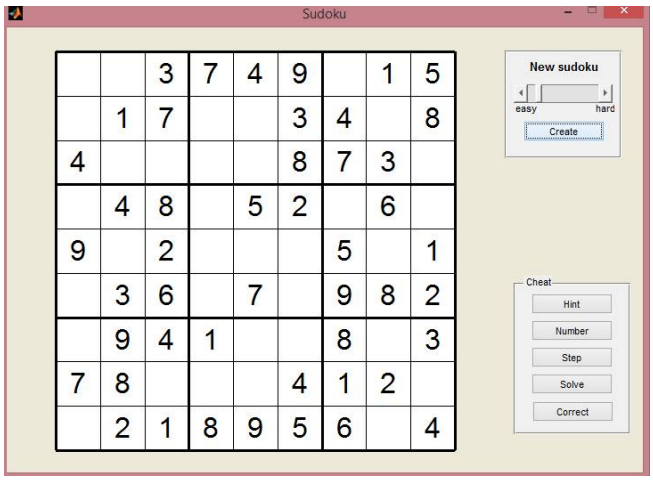

Fig 1: Null Sudoku Representation

given an initial state, no events exist that meet all Sudoku requirements. The resultant counter example is the answer to the puzzle.

.

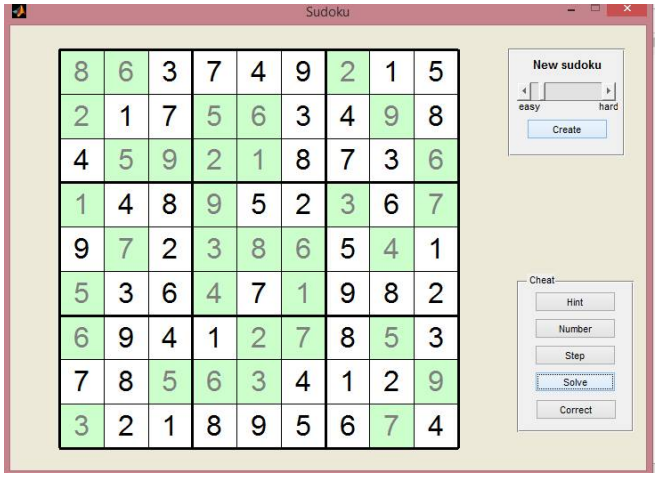

Fig2: Complete Sudoku Representation

ALGORITHM

Backtrack-Free Constraint Propagation Algorithm using Parallel Consistency Improvement Schemes [14]

Parallel consistency in CP means that several constraints will be evaluated in parallel.

Constraints that contain the same variables have data dependencies, and therefore their pruning must be synchronized. However, since the pruning is monotonic, the order in which the data is modified does not affect the correctness. This follows from the property that well-behaved constraint propagators must be both decreasing and monotonic [8][3]

Parallel Consistency will help deduce the leftover grid elements and create a time sensitive computational series for the algorithm. It will reduce the performance load on P1 (given P1, P2, P3 are the three major processors under consideration in this example) [9]. The overhead time for the communication will reduced by the implementation of a reservoir memory which will be used in a quick succession for on-going processes between the grid completions. The 3x3 grid will then be verified against the respective row and column.

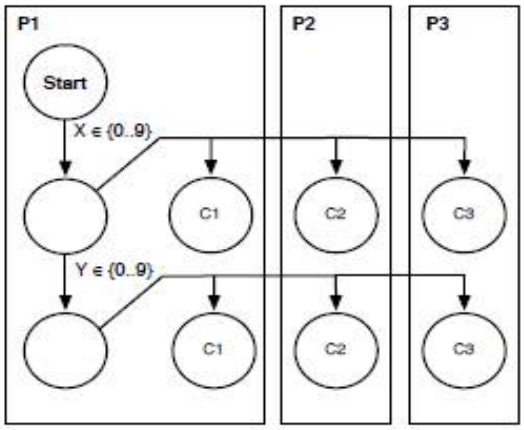

Fig. 3 Parallel consistency in constraint programming[11]

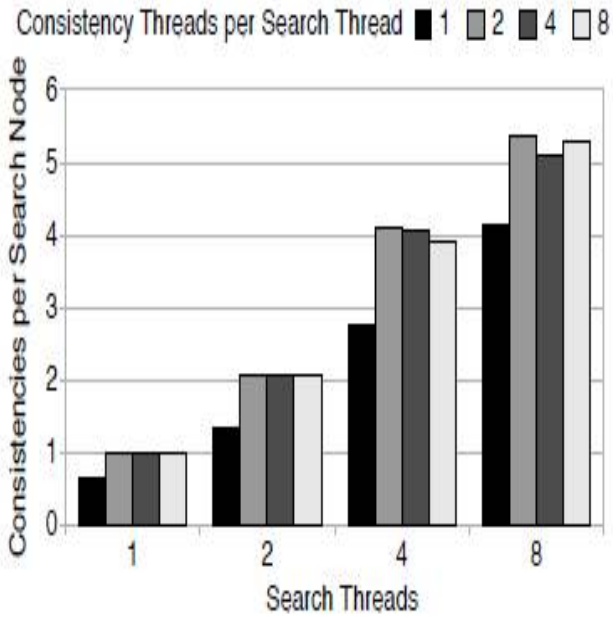

Fig. 3. Consistency enforcements per search Node when searching for 200 solutions

To Sudoku.

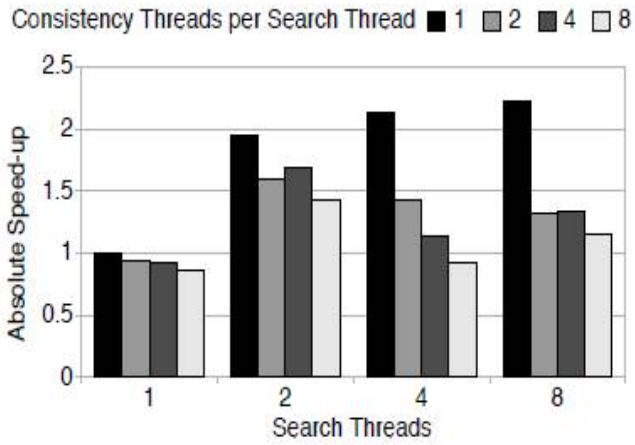

Fig.4. Consistency enforcements per node when searching for 5000 solutions to Sudoku

Redundant Constraints

Regardless of we use hyper arc-consistency as the system to every one distinctive imperatives we won't generally attain worldwide consistency. The imperatives inter- act clinched alongside various routes and the nearby thinking with respect to every demand alone can't misuse these associations [1]. A normal technique should help with this issue will be the utilization about excess imperatives which could reinforce those findings.

5.1 Row/Column interaction

This demand handles those collaboration of rows and columns in the grid. The grid demand [5], also known as cardinality grid for [6], communicates demand on the amount of occurrences of the qualities over rows. Furthermore columns of a grid this diminishes should a situated from claiming basic matching issues to a permutation grid. Each quality must happen precisely once for every column Furthermore column, this corresponds with a matching between column and columns (the two sets of nodes), and edges which join a column and a section on the provided for esteem will be in the space of the comparing grid component. By discovering a maximal matching et cetera distinguishing determinedly joined parts clinched alongside a re-oriented chart we might dispense with the individuals values starting with the sum domains which don't have a place with whatever maximal matching.

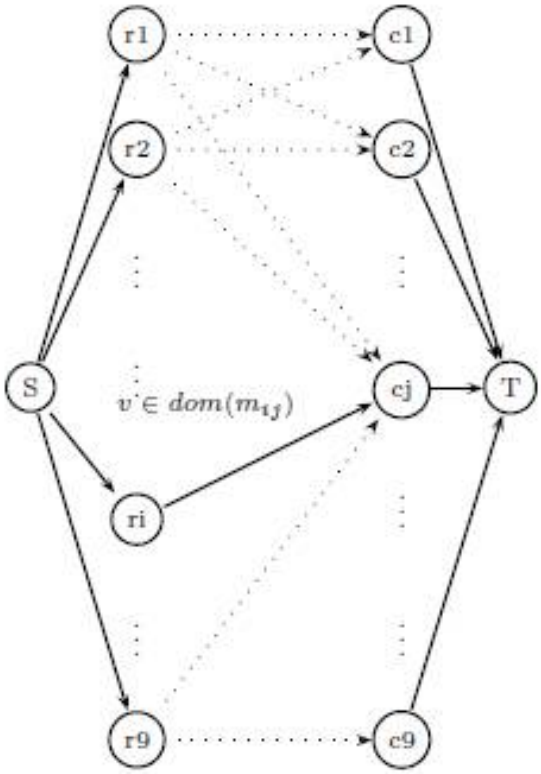

Fig 5: Grid implementation

5.2 *Row/Block interaction [12]]*

We now look at the interaction of a single row (column) with a single major block as shown in figure 6. Those column the greater part distinctive demand and the sum diverse demand. To the significant piece correspond previously, three variables. One vital condition will be that the set from 1111111111claiming variables b = {x14, x15, x16, x17, x18, x19} utilization the same situated from claiming qualities Similarly fas those situated c = {x21, x22, x23, x31, x32, x33} of variables. We could misuse this condition Eventually Tom's perusing evacuating from the variables for set b every

last bit qualities which need aid not underpinned via qualities previously, set c's also the other way around. This demand will be an uncommon instance of the same with cardinality from claiming [7], which accomplishes hyper arc-consistency Eventually Tom's perusing comprehending a stream issue.

5.3 Rows/Blocks interaction

We can additionally think as of those communication of at significant squares in a accordance (column) with the three meeting rows (columns), as indicated to figure 7. Every esteem must happen precisely once done each row, as well as for each piece. To each value, we might express this concerning illustration An matching issue between rows and blocks, which need aid those hubs in the bipartite chart indicated done figure 7. Square On the provided for quality is in the Web-domain for whatever of the three covering variables [2][3]. On that edge doesn't have a place with any matching, we could uproot the quality starting with the area of the sum three variables. The algorithm meets expectations in the common way, we principal Figure a maximal matching, reorientation the edges not in the matching Furthermore look for determinedly joined parts (SCC). Edges which are not in the matching and whose winds don't have a place with the ssame SCC could make uprooted.

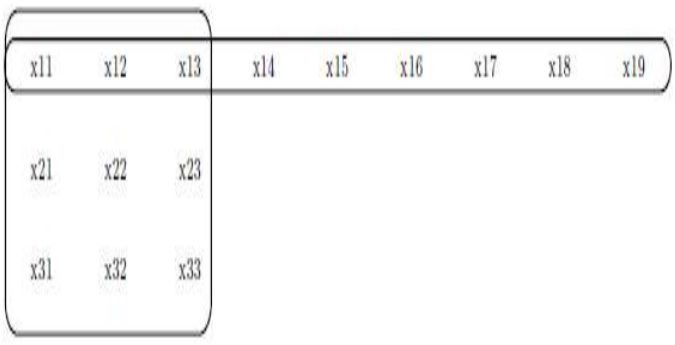

Fig 6: Row/Blocks Implementation

5.3 *Rows/Columns/Blocks interaction*

Ideally, we might want with catch the interactions about row, section Also piece imperatives together. In any case we at that point need three sets of hubs to consider, something like that that a straightforward bipartite matching appears to be no more workable. We might require an All the more perplexing stream model such as those one viewed as over [13] should catch those interactional of the sum diverse imperatives

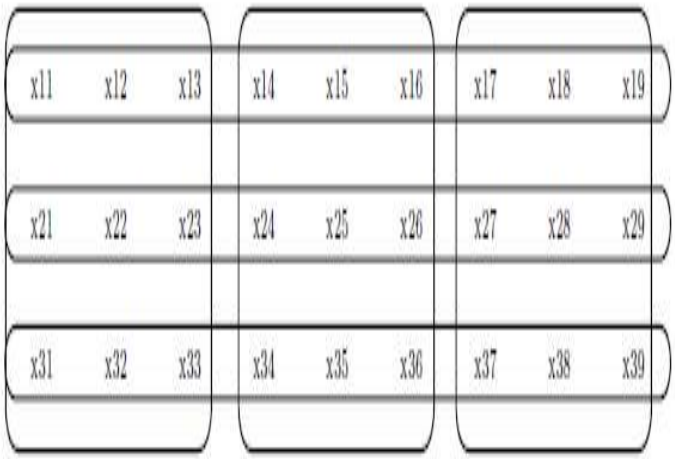

Fig 7: Rows/Columns/Blocks interaction

CONCLUSIONS

This allows us to generate a constraints based Sudoku grid solution with parallel consistency improvement techniques to deliver useful performance. The row and column interactions has been kept backtracking free with increased approach to redundancy and column perception matching using existing information from the reservoir memory available with the processor[10].

With the help of GUI implementation we successfully deliver the MATLAB model for the SUDOKU solution.

REFERENCES

1. A.C. Bartlett and A.N. Langville. An integer programming model for the
2. Sudoku problem. Preprint, available at http://www. Cofc. edu/~ langvil-

3. JF Crook. A pencil-and-paper algorithm for solving Sudoku puzzles. Notices of the AMS, 56(4):460{468, 2009.

4. Z. Geem. Harmony search applications in industry. Soft Computing Applications in Industry, pages 117{134, 2008.

5. R.C. Green II. Survey of the Applications of Artificial Intelligence Techniques to the Sudoku Puzzle. 2009.

6. SK Jones, PA Roach, and S. Perkins. Construction of heuristics for a Search based approach to solving Sudoku. Research and Development in Intelligent Systems XXIV

7. Rituparna Datta, Rommel G. Regis. (2016) A surrogate-assisted evolution strategy for constrained multi-objective optimization. *Expert Systems with Applications* **57**270-284. Online publication date: 1-Sep-2016. CrossRef

8. Sun, X.H., Chen, and Y.: Reevaluating amdahl's law in the multicore era. J. Parallel Distrib.Comput. 70(2) (2010) 183–188

9. Nguyen, T., Deville, Y.: A distributed arc-consistency algorithm. Sci. Comput. Program. 30(1-2) (1998) 227–250

10. Ruiz-Andino, A., Araujo, L., Sáenz, F., Ruz, J.J.: Parallel arc-consistency for functional

11. The 2009 International Conference on Parallel and Distributed Processing Techniques and Applications 2 (2009) 638–644

12. Schulte, C., Carlsson, M.: Finite domain constraint programming systems. In Rossi, F., van Beek, P., Walsh, T., eds.: Handbook of Constraint Programming. Foundations of Artificial Intelligence. Elsevier Science Publishers, Amsterdam, the Netherlands (2006) 495–526

13. Puget, J.F.: A fast algorithm for the bound consistency of alldiff constraints. In: AAAI/IAAI.(1998) 359–366

14. Oussama Ait Sahed, Kamel Kara, Abousoufyane Benyoucef, Mohamed Laid Hadjili. (2016) an efficient artificial bee colony algorithm with application to nonlinear predictive control. *International Journal of General Systems*1-25. Online publication date: 29-Feb-2016. CrossRef

Ankit Kumar Sinha- Student of Gandhi Institute of Engineering and Technology, Gunupur, Rayagada 4th Year, AE/IE Dept.

Nihit Raj- Student of Gandhi Institute of Engineering and Technology, Gunupur, Rayagada, 4th Year, AE/IE Dept.

A.V Prabu- Asst. Professor, Dept. Electronics, Gandhi Institute of Engineering and Technology, Gunupur, Rayagada

Vishal Kumar Singh- Student of Gandhi Institute of Engineering and technology, 4th Year, AE/IE Dept.