A Filtering Technique for Helping to Solve Sudoku Problems

Ricardo Soto[1,2,3], Broderick Crawford[1,4,5], Cristian Galleguillos[1(✉)], Kathleen Crawford[1], and Fernando Paredes[6]

[1] Pontificia Universidad Católica de Valparaíso, Valparaíso, Chile
{ricardo.soto,broderick.crawford}@ucv.cl
{cristian.galleguillos.m,kathleen.crawford.a}@mail.ucv.cl
[2] Universidad Autónoma de Chile, Santiago, Chile
[3] Universidad Cientifica del Sur, Lima, Peru
[4] Universidad Central de Chile, Santiago, Chile
[5] Universidad San Sebastián, Santiago, Chile
[6] Escuela de Ingeniería Industrial, Universidad Diego Portales, Santiago, Chile
fernando.paredes@udp.cl

Abstract. This paper highlights the current usability issues when solving Sudoku problems. This problem is a well-known puzzle game which consists in assigning numbers in a game board, commonly of 9×9 size. The board of the game is composed of 9 columns, 9 rows and 9 3×3 sub-grids; each one containing 9 cells with distinct integers from 1 to 9. A game is completed when all cells have a value assigned, and the previous constraints are satisfied. Some instances are very difficult to solve, to tackle this issue, we have used a filtering technique named Arc Consistency 3 (AC3) from the Constraint Programming domain. This algorithm has revealed which is much related to the strategies employed by users in order to solve the Sudoku instances, but in contrast, this technique is executed in a short time, offering a good resolution guide to the users. In general, filtering techniques make easier solving Sudoku puzzles, providing good information to users for this.

Keywords: Sudoku · Constraint programming · Arc consistency

1 Introduction

Sudoku is a mathematician game, widely known for being a very entertaining pastime, which consists in assigning numbers in a game board. Traditionally this board is a grid of 9×9 size, then there exists 81 positions which are called cells. The grid is composed of 9 columns, 9 rows and 9 3×3 sub-grids; each of these structures containing 9 cells. The game begins with distinct integers from 1 to 9 sparsely assigned in the grid, these are called "givens". The givens must be at least 17 digits to ensure that the game will have a unique solution [7].

An instance of the game is completed, when all cells have a assigned value, but it is not so easy, the digit assignations must comply the problem constraints.

C. Stephanidis (Ed.): HCII 2015 Posters, Part I, CCIS 528, pp. 598–603, 2015.
DOI: 10.1007/978-3-319-21380-4_102

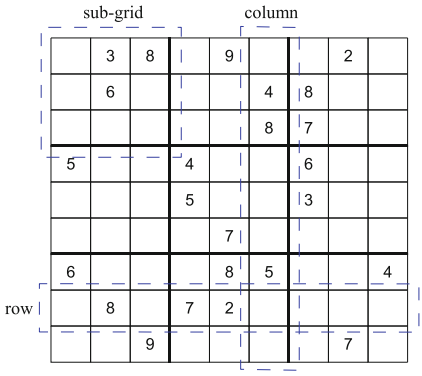

Fig. 1. Example of a Sudoku instance with 24 givens.

The constraints can be reduced to a single statement which says, every cell must have a number, it must be unique in the row, column and sub-grid which it belongs to. Then, if the statement is separated in single constraints, we have 3 constraints for every cell making this game very difficult to solve (Fig. 1).

Such a puzzle belongs to the NP-complete collection of problems [14], this problem has widely been studied, proposing several approaches during the last years to solve it, from complete search methods such as constraint programming [10,11] and boolean satisfiability [3] to incomplete search methods such as genetic programming [5], metaheuristics in general [1,2,6,8,9], and hybrid algorithms [12,13]. The main interest of the Sudoku problem is due to some instances are very difficult to solve.

To tackle this issue, we have used a filtering technique in order to validate the assignation from users, moreover to meet the possible value assignation to facilitate the Sudoku resolution and determining whether or not is a solution in an earlier stage of the resolution process by users, in relation to assigned and given values. The filtering technique chosen is named Arc Consistency 3 (AC3) from the Constraint Programming domain, which is an algorithm that enforces the arc consistency property on a constraint network. After reviewing its resolution process, we have realized that the resolution phase is much related to the strategies employed by users in order to solve the instances, but in contrast, this technique is executed in a short time, offering a good resolution guide to the users.

The effectiveness of filtering techniques was addressed performing an evaluation; it was realized in order to identify the quality of filtering over the different difficulties of the Sudoku puzzle. In general, filtering techniques make easier solving Sudoku puzzles, providing good information to users for this.

This paper is organized as follows. In Sect. 2, the employed filtering technique is explained. We describe the game prototype in Sect. 3, and the corresponding discussions. Finally, our conclusions are presented.

2 Filtering Technique: Arc Consistency

The Arc Consistency property comes from the Constraint Programming (CP) domain. CP is a programming paradigm where the mathematical problems are stated as variables and constraints, and then solved by a CP solver. A problem which is stated in terms of variables and constraints is called Constraint Satisfaction Problem (CSP). In order to resolve a CSP, the state of the variable are searched and the restrictions must be satisfied simultaneously. This property establishes a relation between two variables and maintain only feasible values in each variable domains, permitting to reduce the combinatorial search space, deleting the unfeasible values.

To ensure that a problem is arc consistent, it is needed to apply a filter technique, existing different algorithms to ensure the property, the algorithm used in this research was Arc Consistency Algorithm #3, defined by Alan Mackworth [4].

To apply the mentioned algorithm, the problem must be stated as a CSP, then a 9×9 Sudoku can be modeled as:

- $X = (x_{11}, \ldots, x_{99})$ is the sequence of variables, and $x_{ij} \in X$ identifies the cell placed in the i^{th} row and j^{th} column of the Sudoku matrix, for $i = 1, \ldots, 9$ and $i = 1, \ldots, 9$.
- D is the corresponding set of domains, where $D(x_{ij}) \in D$ is the domain of the variable x_{ij};
- C is the set of constraints defined as follows:
  - To ensure that values are different in rows and columns: $x_{k,i} \neq x_{k,j} \wedge x_{i,k} \neq x_{j,k}, \forall(k \in [1,9], i \in [1,9], j \in [i+1,9])$
  - To ensure that values are different in sub-squares: $x_{(k1-1)*3+k2,(j1-1)*3+j2} \neq x_{(k1-1)*3+k3,(j1-1)*3+j3}, \forall(k1, j1, k2, j2, k3, j3 \in [1,3] | k2 \neq k3 \wedge j2 \neq j3)$

Algorithm 1. Revise3
Input: x_i, c
Output: $CHANGE$
1 $CHANGE \leftarrow$ false
2 **Foreach** $v_i \in D(x_i)$ **do**
3 **If** $\nexists \tau \in c \cap \pi_{X(c)}(D)$ *with* $\tau[x_i] = v_i$ **do**
4 remove v_i from $D(x_i)$
5 $CHANGE \leftarrow$ true
6 **End If**
7 **End Foreach**
8 **Return** $CHANGE$

The AC-3 algorithm if composed from the Algorithm 1 and 2. The main idea of these algorithms are the examination of value pairs and remove every value in $D(X_i)$ that is inconsistent with the problem constraints, this is performed by the function `Revise3`. If the pair of the value v_i is eliminated from $D(x_i)$, then it is necessary to verify the arc from the other direction. Every domain must be

consistent, doing a detailed revision of all the domains. This is done by using a loop that verifies arcs until no change happens. The algorithm ends when all arcs are revised and no more changes exists (Q is empty), and it returns true when all arcs have been verified and remaining values of domains are arc consistent for all the problem constraints.

Algorithm 2. AC3/GAC3
Input: X, D, C
Output: `Boolean`
1 $Q \leftarrow (x_i, c) | c \in C, x_i \in X(c)$
2 **while** $Q \neq \emptyset$ **do**
3 select and remove (x_i, c) from Q
4 **If** `Revise3`(x_i, c) **then**
5 **If** $D(x_i) = \emptyset$ **then**
6 **return false**
7 **Else**
8 $Q \leftarrow Q \cup \{(x_j, c') | c' \in C \wedge c' \neq c \wedge x_i, x_j \in X(c') \wedge j \neq i \}$
9 **End If**
10 **End If**
11 **End While**
12 **Return** true

3 Experiments and Discussions

In Fig. 2, it is possible to see the gameboard with a Sudoku instance at the left. When a user selects a cell, if the checkbox "Help" is checked on the right side, an assistance for solving is given. Before and after typing an digit on the board, the AC3 algorithm is executed and the constraints are checked, delivering the possible assignations. It is possible to recognize that the algorithm is very effective for helping to solve, because a guidance through all the resolution process is given, making more easy to solve the game instances.

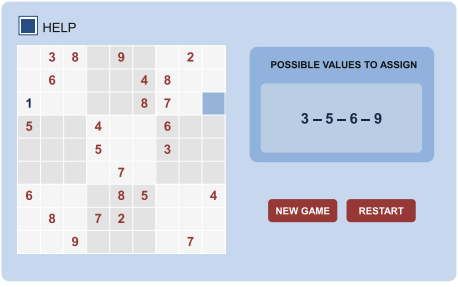

Fig. 2. Showing help solving an instance to a user.

If a wrong digit is entered, the prototype will show an error message on top, and the possible assignations will be shown, as appear in Fig. 3. If there are not possible assignations, clearly the arc consistency has detected an unfeasible assignation, and the previously entered digits were wrongly entered to the gameboard.

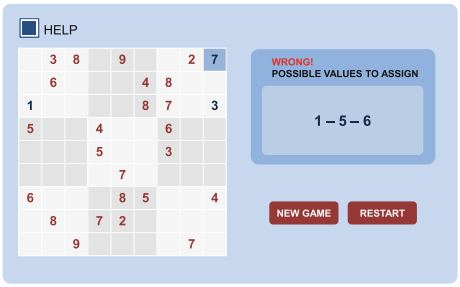

Fig. 3. Correcting an error, in order to help solving an instance to a user.

4 Conclusions

In this paper, we have presented a very useful technique in order to facilitate the resolution process of the Sudoku game. This game is known to be a NP-Complete problem, being a popular game in research fields like Constraint Programming and Metaheuristics, involving instances very difficult to solve. We have used the AC3 algorithm in order to offer a resolution guide to the users. This technique is similar to the strategies employed by users at the resolution process, but this technique is executed in a short time, making more easier and faster the solving process of the Sudoku puzzles.

Acknowledgments. Cristian Galleguillos is supported by Postgraduate Grant Pontificia Universidad Católica de Valparaíso 2015. Ricardo Soto is supported by Grant CONICYT / FONDECYT / INICIACION / 11130459. Broderick Crawford is supported by Grant CONICYT / FONDECYT / REGULAR / 1140897. Fernando Paredes is supported by Grant CONICYT / FONDECYT / REGULAR / 1130455.

References

1. Asif, M.: Solving NP-complete problem using ACO algorithm. In: International Conference on Emerging Technologies, pp. 13–16. IEEE Computer Society (2009)
2. Lewis, R.: Metaheuristics can solve sudoku puzzles. J. Heuristics **13**(4), 387–401 (2007)
3. Lynce, I., Ouaknine, J.: Sudoku as a sat problem. In: International Symposium on Artificial Intelligence and Mathematics (ISAIM) (2006)

4. Mackworth, A.: Consistency in networks of relations. Artif. Intell. **8**(1), 99–118 (1977)
5. Mantere, T., Koljonen, J.: Solving, rating and generating sudoku puzzles with ga. In: IEEE Congress on Evolutionary Computation, pp. 1382–1389. IEEE (2007)
6. Mantere, T., Koljonen, J.: Solving and analyzing sudokus with cultural algorithms. In: 2008 IEEE Congress on Evolutionary Computation. CEC 2008. (IEEE World Congress on Computational Intelligence), pp. 4053–4060, June 2008
7. McGuire, G., Tugemann, B., Civario, G.: There is no 16-clue sudoku: Solving the sudoku minimum number of clues problem. CoRR, abs/1201.0749 (2012)
8. Moraglio, A., Togelius, J.: Geometric particle swarm optimization for the sudoku puzzle. In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, GECCO 2007, pp. 118–125. ACM, New York, NY, USA (2007)
9. Moraglio, A., Togelius, J., Lucas, S.: Product geometric crossover for the sudoku puzzle. In: IEEE Congress on Evolutionary Computation, pp. 470–476. IEEE Computer Society (2006)
10. Rossi, F., van Beek, P., Walsh, T.: Handbook of Constraint Programming. Elsevier, New York (2006)
11. Simonis, H.: Sudoku as a Constraint Problem. In: Hnich, B., Prosser, P., Smith, B. (eds.) Proceedings 4th International Workshop on Modelling and Reformulating Constraint Satisfaction Problems, pp. 13–27 (2005)
12. Soto, R., Crawford, B., Galleguillos, C., Monfroy, E., Paredes, F.: A hybrid ac3-tabu search algorithm for solving sudoku puzzles. Expert Syst. Appl. **40**(15), 5817–5821 (2013)
13. Soto, R., Crawford, B., Galleguillos, C., Monfroy, E., Paredes, F.: A prefiltered cuckoo search algorithm with geometric operators for solving sudoku problems. Sci. World J. **2014**, 12 (2014). cited By (since 1996)2
14. Yato, T., Seta, T.: Complexity and completeness of finding another solution and its application to puzzles. IEICE Trans. **E86–A**(5), 1052–1060 (2003)