Solving Sudoku with Consistency: A Visual and Interactive Approach
Ian Howel, Robert J. Woodward, Berthe Y. Choueiry, Christian Bessiere

To cite this version:

HAL Id: lirmm-01897933
https://hal-lirmm.ccsd.cnrs.fr/lirmm-01897933
Submitted on 17 Oct 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Solving Sudoku with Consistency: A Visual and Interactive Approach

Ian Howell, Robert Woodward, Berthe Y. Choueiry, Christian Bessiere
1Constraint Systems Laboratory, University of Nebraska-Lincoln, USA
2CNRS, University of Montpellier, France
{ihowell|woodwar|choueiry}@cse.unl.edu bessiere@lirmm.fr

Abstract

We describe an online, interactive system with a graphical interface to illustrate the power and operation of consistency algorithms in a friendly and popular context, namely, solving Sudoku puzzles. Our tool implements algorithms for enforcing five (domain-based) consistency properties on binary and non-binary constraint models. Our tool is useful for research, education, and outreach. From a scientific standpoint, we propose a new consistency property that can solve the hardest known 9×9 Sudoku instances without search, but leave open the question of the lowest level of consistency needed to solve every 9×9 Sudoku puzzle. We have used the current tool in the classroom to introduce students to modeling problems with constraints, explain consistency properties, and illustrate the operations of constraint propagation and lookahead. Finally, we have also used this tool during outreach activities to demystify AI to children and the general public and show them how computers ‘think.’

1 Introduction

The popularity of the Sudoku puzzle stems perhaps from the simplicity of its rules and the wide range of difficulty levels in which it can found in magazines, books, and on the internet. This combinatorial decision problem is known to be NP-complete [Yato, 2003] and lends itself particularly well to introduce the general public and students of Computer Science to the area of Constraint Processing. The current state of the art in terms of the availability of online solvers (based on backtracking search or on convoluted patterns and rules defined by humans) has not changed much since 2007 when our lab introduced our first constraint-based Sudoku solver [Reeson et al., 2007].

2 Constraint Models and Consistency Properties

Constraint Satisfaction Problems (CSPs) are defined as a tuple (X, D, C), where X is a set of variables, D is the set of domains of the variables with D_i representing the domain of X_i, and C is a set of constraints, where a constraint C_i has scope \{x_{i1}, \ldots, x_{ik}\} and a set of acceptable assignments \{(v_{i1}, \ldots, v_{ik}) \in D_{i1} \times \ldots \times D_{ik}\} restricting the values assignable to the variables in C_i’s scope. In a binary CSP, a constraint has two variables in its scope. In non-binary CSPs, it has more. Solving a CSP requires assigning a value to each variable such that all constraints are satisfied.

The constraint model of the 9×9 Sudoku puzzle has 81 variables representing the cells of the 9×9 board. The domain of a variable is the set \{1, 2, \ldots, 9\}. The binary model has 810 binary DIFF constraints defined over every two variables in the same row, column, or block. The non-binary model has 27 non-binary ALLDIFFERENT constraints defined over the 9 variables appearing in a row, column, or block [Régin, 1994].

Consistency algorithms enforce a given (domain-based) consistency property by removing, from the domains, values that cannot appear in a solution to the CSP. Such algorithms are typically used before search and/or as lookahead during search to reduce the size of the search space. Below, we distinguish between the consistency properties of the binary model and those of the non-binary model by including the letter G for the latter (e.g., AC and GAC) and state the definitions in a way that is applicable to both models.

Definition 1 Arc Consistency (AC and GAC) [Waltz, 1975] [Mackworth, 1977]: A constraint network P = (X, D, C) is

\[http://www.mediafire.com/9ypndhalzadpwaw\]

\[http://sudoku.unl.edu/\]
AC iff, for every constraint \( C_i \in C \), and \( \forall x_j \in \text{scope}(C_i) \), every value \( v \in D(x_j) \) is consistent with \( C_i \) (i.e., appears in some support of \( C_i \)).

**Definition 2** Singleton Arc Consistency (SAC and SGAC) [Debruyne and Bessiere, 1997]: A constraint network \( P = (X, D, C) \) is singleton arc consistent iff \( \forall x_i \in X, \forall a \in D(x_i), \) the network \( P|_{x_i=a} = (X, D|_{x_i=a}, C) \) obtained by replacing \( D(x_i) \) by the singleton \( \{a\} \) is not arc inconsistent. If \( P|_{x_i=a} \) is arc inconsistent, we say that \( (x_i, a) \) is SAC inconsistent.

**Definition 3** Partition-One Arc Consistency (POAC and POGAC) [Bennaceur and Affane, 2001]: Given a constraint network \( P = (X, D, C) \), a variable \( x_i \) is partition-one-AC iff \( D(x_i) \neq \emptyset \), all values in \( D(x_i) \) are SAC, \( x_j \notin \{1, \ldots, n\}, j \neq i, \forall v_j \in D(x_j), \exists v_i \in D(x_i) \) such that \( v_j \in AC(P|_{x_i=a}) \). The constraint network \( P \) is POAC iff all its variables are POAC.

**Definition 4** Bidirectional Singleton Arc Consistency (BiSAC and BiSGAC) [Bessiere and Debruyne, 2008]: A constraint network \( P = (X, D, C) \) is bidirectional singleton arc consistent iff \( \forall x_i \in X, \forall a \in D(x_i), AC(T_{ia}) \neq \emptyset \), where \( T_{ia} = (X, D_{ia}, C) \), with \( D_{ia}(x_j) = \{b \in D(x_j)|(x_i, a) \in AC(P|_{x_i=a})\} \).

**Definition 5** Double Singleton Arc Consistency (SSAC and SSGAC): A constraint network \( P = (X, D, C) \) is double singleton arc consistent iff \( \forall x_i \in X, \forall a \in D(x_i) \) the network \( P|_{x_i=a} = (X, D|_{x_i=a}, C) \) obtained by replacing \( D(x_i) \) by the singleton \( \{a\} \) is not singleton arc inconsistent.

Of the above properties, AC is the weakest and SSGAC is the strongest. To enforce SAC, SGAC, POGAC, BiSGAC and SSGAC, we use the specialized algorithm for the ALLDIFFERENT global constraint [Régin, 1994]. We also implement a backtrack search for finding all solutions, in the case the puzzle entered has more than one solution (i.e., ill formed).

A 9×9 well-formed Sudoku puzzle has a fixed size and, by definition, a single solution. Thus, the question of the weakest consistency to solve every instance is legitimate but remains open. Interestingly, SSGAC solves all 375 hardest known Sudoku instances. In fact, 276 instances require at least SSGAC, 25 BiSGAC+SSAC, 7 BiSGAC, and 67 SSAC.

### 3 User Interface

The user interface has two halves: the control panel to the right (Figure 1) and the Sudoku board to the left (Figure 2).

The control panel has five tabs to access various functionalities: LOAD (an instance from the database), UPLOAD (a picture of an instance using OCR methods from Aruco and CV Javascript libraries), SOLVE (the current instance on the board), and SUBMIT (a new instance to the database).

The board shows a standard 9×9 Sudoku layout. It is surrounded on the top and left sides by characters designating columns and rows and on the bottom and right sides by buttons to enforce AC (single line) and GAC (multiple lines) on the variables of the corresponding row or column.

The board reflects the current state of the constraint model. Clues are displayed as blue characters in bold and are immutable but can be converted to standard assigned values.

![Figure 1: Enforcing the sequence AC, GAC, SAC, then GAC in the SOLVE tab of the control panel](image)

From the SOLVE tab, the user can apply AC or GAC on a block (ENFORCE), enforce one of the 10 consistencies shown in a Hasse diagram of their relative strength (PROPAGATE), maintain partial or realfull lookahead while interactively instantiating variables (FILTER), run search to find all the solutions of the configuration on the board (FIND). As the user enforces a consistency property (PROPAGATE), the board is dynamically updated to reflect the effects of this action. The user can step through the iterations of a given consistency algorithm. The board is animated by highlighting in grey the constraint under consideration, in red values removed at the current iteration, and bolded in black the values to be removed in future iterations of the same propagation.

### Acknowledgments

This research is supported by an NSF Grant No. RI-1619344 and a UNL UCARE grant.
References


