Solving sudokus via metaheuristics and AC3

Conference Paper · October 2014

DOI: 10.1109/icastech.2014.7068127

5 authors, including:

Cristian Galleguillos Miccono
Pontificia Universidad Católica de Valparaíso
22 PUBLICATIONS **98** CITATIONS
SEE PROFILE

Ricardo Soto
Pontificia Universidad Católica de Valparaíso
267 PUBLICATIONS **1,488** CITATIONS
SEE PROFILE

Broderick Crawford
Pontificia Universidad Católica de Valparaíso
332 PUBLICATIONS **1,886** CITATIONS
SEE PROFILE

Sanjay Misra
Covenant University Ota Ogun State, Nigeria
528 PUBLICATIONS **2,709** CITATIONS
SEE PROFILE

Solving Sudokus via Metaheuristics and AC3

Ricardo Soto*†‡, Broderick Crawford*§¶, Cristian Galleguillos*, Sanjay Misra‖ and Eduardo Olguín¶
*Pontificia Universidad Católica de Valparaíso, 232807 Valparaíso, Chile.
Email: ricardo.soto@ucv.cl, broderick.crawford@ucv.cl, cgalleguillosm@ieee.org
†Universidad Autónoma de Chile, 7500138 Santiago, Chile.
‡Universidad Central de Chile, 8330601 Santiago, Chile.
§Universidad Finis Terrae, 7501015 Santiago, Chile.
¶Universidad San Sebastián, 8420524 Recoleta, Santiago, Chile.
Email: eduardo.olguin@uss.cl
‖Nigeria Covenant University, Ogun State, Nigeria.
Email: ssopam@gmail.com

***Abstract*—The Sudoku puzzle consists in filling a square matrix with 9 rows and 9 columns, divided into 9 3×3 regions, so that each column, row, and region contains different digits from 1 to 9. Such a puzzle belongs to the NP-complete class of problems, existing different exact and approximate methods able to solve it. This paper reports recent results for solving Sudokus achieved by combining metaheuristics and a filtering technique coming from the constraint programming domain named AC3.**

Keywords: **Constraint Satisfaction; Sudoku; Arc-consistency.**

I. Introduction

The Sudoku is a logic-based placement puzzle that consist in filling a square matrix with 9 rows and 9 columns, divided into 9 3×3 regions, so that each column, row, and region contains different digits from 1 to 9. The puzzle considers some cells already filled that cannot be replaced. The position of these pre-filled cells is relevant, having more impact on the difficulty of the instance than the total amount of them. An interesting three-level difficulty for Sudokus is given in [1].

During the last years, Sudoku puzzles have been successfully solved by different techniques ranging from classic exact methods such as constraint programming [2], [3], [4] and boolean satisfiability [5] to more modern metaheuristics such as genetic programming [1] simulated annealing [6], ant colony optimization [7], and particle swarm optimization [8].

This paper reports recent results for this problem achieved by combining metaheuristics and a powerful filtering technique from the constraint programming domain named AC3. The main purpose of the filtering techniques is to previously delete from the search space the values that do not conduct to any feasible solution. In this way, the work of the metaheuristic is alleviated leading to a faster solving process.

This short paper is structured as follows. The Combining Tabu Search and AC3 is described in Section II. Experiments are illustrated in Section III. Finally, we conclude and give some directions for future work.

II. Combining Metaheuristics and AC3

Algorithm 1 [9] illustrates the combination of a metaheuristic with the filtering technique. In practice, a classic tabu search is combined with AC3.

Tabu Search (TS) [10] is a widely used metaheuristic devoted to solve optimization problems, that has successfully been employed for solving various problems from real-world as well as problems from academic literature such as the knapsack problem, the traveling salesman problem, and the timetabling problem.

The main idea of TS relies in using a local search procedure that permits to iteratively move from one potential solution to another promising one until some stop criterion has been reached. This procedure is complemented with a memory structure called tabu list, which is perhaps a main feature that distinguishes TS from many incomplete methods. The goal of this memory structure is on one hand to help the TS to escape from poor-scoring areas, and on the other to avoid returning to recent visited states.

Here, the idea is to employ the filtering technique AC3 as a pre-processing phase (line 1) and at each iteration of the tabu search (line 9). In this way, we attempt to delete the unfeasible values in both stages of the process reducing the work of the metaheuristic.

As input, Algorithm 1 receives the size of the tabu list and the Sudoku problem to be solved, which is stated as a constraint network $\langle X, D, C\rangle$ where:

- $X = (x_{11}, \ldots, x_{nm})$ 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 $i = 1, \ldots, n$ and $i = 1, \ldots, m$.
- 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 guarantee 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 guarantee that values are different in sub-matrices: $x_{(k1-1)*3+k2,(j1-1)*3+j2} \neq$

978-1-4799-4998-4/14/$31.00 © 2014 IEEE

$x_{(k1-1)*3+k3,(j1-1)*3+j3}, \forall(k1, j1, k2, j2, k3, j3 \in [1,3] \mid k2 \neq k3 \wedge j2 \neq j3)$

The output of the procedure is S_{best}, which is the best solution reached. A solution is thus a complete variable-value assignment $\{x_{11} \rightarrow a_{11}, \ldots, x_{nm} \rightarrow a_{nm}\}$ such that $a_{ij} \in D(x_{ij})$ for $i = 1, \ldots, n$ and $i = 1, \ldots, m$. At line 1 of the procedure, the pre-processing phase is triggered.

The input of the AC3 filtering phase is the constraint network $\langle X, D, C \rangle$ representing the Sudoku, while the output is a new initial solution S_{best}. The set of domains D are also updated by deleting the unfeasible values. At the second line, the tabu list is initialized, followed by a while loop that controls the iterations of the process until the stop condition is met (a fixed number of iterations).

The candidate list is set to 0 at line 4. Next, a for loop handles the generation of candidates, which is performed via roulette-wheel [11], [12]. Then, the best candidate is located, which corresponds to the matrix with more filled cells. This candidate is then submitted to a new filtering step so as to attempt to reduce its set of domains $D_{S_{candidate}}$.

It is worth mentioning that domain filtering via AC3 is also capable to reduce a domain $D(x_{ij})$ to a single value. This permits one to fill the corresponding cell x_{ij} with the unique remaining value and thus to have a better quality $S_{candidate}$.

At line 10, the cost of the $S_{candidate}$ is compared to the best reached solution. If the cost of $S_{candidate}$ is better than the one of S_{best}, $S_{candidate}$ becomes the new S_{best}. The cost of a solution corresponds to the sum of empty cells of the matrix. Clearly, a solution with cost zero is optimal. Finally, the tabu list is updated according to a classic tabu search procedure.

Algorithm 1 - pre-filtered tabu search

Input: $TabuList_{size}, X, D, C$
Output: S_{best}
1 $S_{best}, D \leftarrow$ `AC3` (X, D, C)
2 `tabuList` $\leftarrow 0$
3 **While** $\neg$ `StopCondition` **do**
4 `CandidateList` $\leftarrow 0$
5 **For** $(S_{candidate} \in S_{best_{neighboorhood}})$ **do**
6 `CandidateList`$\leftarrow$ `CandidateGenerator()`
7 **End For**
8 $S_{candidate} \leftarrow$ `LocateBestCandidate(CandidateList)`
9 $S_{candidate}, D \leftarrow$ `AC3` $(X, D_{S_{candidate}}, C)$
10 **If** `cost`$(S_{candidate}) \leq$ `cost`(S_{best})
11 $TabuList \leftarrow$ `FeatureDifferences(`$S_{candidate}, S_{best}$`)`
12 $S_{best} \leftarrow S_{candidate}$
13 **While** $TabuList > TabuList_{size}$ **do**
14 `ExpireFeature(`$TabuList$`)`
15 **End While**
16 **End If**
17 **End While**
18 **Return** S_{best}

The integration of AC3 can be performed with any metaheuristic. Indeed, combinations of AC3 with cuckoo search [13] and with artificial fish swarm (there is no formal publication for this work yet) have also been implemented.

The filtering process is carried out by enforcing a consistency property on the problem. Here, we employ the arc-consistency [14], which is one of the most used consistency properties in constraint programming. The purpose is to revise the constraint relation between variables by removing the values from $D(x_{ij})$ that lead to inconsistencies w.r.t. a given constraint. Details about the employed filtering algorithms can be seen in [9], [13].

III. Experimental results

Table I depicts the results obtained by the hybrid AC3-tabu search [9], table II shows the results using hybrid AC3-cuckoo search [13], and table III by the genetic algorithm described in [1], which is the best approximate method reported.

We contrast the percentage of hard Sudoku problems(available at http://lipas.uwasa.fi/∼timan/sudoku/) successfully solved taking into account 30 tries (100% means that 30 out of 30 Sudokus were successfully solved) considering two different cases: 100,000 and unlimited iterations.

Both hybrids outperform by far the performance of the genetic algorithm which is able to success in only 1 out of 3 hard Sudoku instances. An extended result list can be seen in [9], [13].

TABLE I. Percentage of Sudokus successfully solved by AC3-tabu search considering 100,000 and unlimited iterations

	AC3-tabu search	
Problem	Unlimited	100,000
Hard a	100%	100%
Hard b	100%	100%
Hard c	100%	100%

TABLE II. Percentage of Sudokus successfully solved by AC3-cuckoo search considering 100,000 and unlimited iterations

	AC3-cuckoo search	
Problem	Unlimited	100,000
Hard a	100%	100%
Hard b	100%	100%
Hard c	100%	100%

TABLE III. Percentage of Sudokus successfully solved by GA considering 100,000 and unlimited iterations

	GA	
Problem	Unlimited	100,000
Hard a	100%	6.66%
Hard b	0%	0%
Hard c	0%	0%

IV. Conclusion

In this paper we have reported recent results for Sudokus achieved by combining metaheuristics and AC3, which is a

powerful filtering technique from the constraint programming domain. The goal of the AC3 here is to previously delete from the search space the values that do not conduct to any feasible solution.

In this way, the work of the metaheuristic is alleviated, and as a consequence the solving process is accelerated. Experimental results validates this approach, where the best reported solutions are outperformed by both the hybrid AC3-tabu search and the hybrid AC3-cuckoo search.

A clear direction for future work is the introduction of pre-filtering phases to additional metaheuristics such as particle swarm optimization, ant, or bee colony algorithms to solve Sudokus or any combinatorial problem.

REFERENCES

[1] T. Mantere and J. Koljonen, "Solving, rating and generating Sudoku puzzles with GA," in *IEEE Congress on Evolutionary Computation*. IEEE Computer Society, 2007, pp. 1382–1389.

[2] F. Rossi, P. van Beek, and T. Walsh, *Handbook of Constraint Programming*. Elsevier, 2006.

[3] T. K. Moon and J. H. Gunther, "Multiple constraint satisfaction by belief propagation: an example using Sudoku," in *IEEE Mountain Workshop on Adaptive and Learning Systems*, 2006, pp. 122–126.

[4] H. Simonis, "Sudoku as a constraint problem," in *4th International Workshop on Modelling and Reformulating Constraint Satisfaction Problems*, 2005, pp. 13–27.

[5] I. Lynce and J. Ouaknine, "Sudoku as a SAT problem," in *International Symposium on Artificial Intelligence and Mathematics (ISAIM)*, 2006.

[6] R. Lewis, "Metaheuristics can solve sudoku puzzles," *J. Heuristics*, vol. 13, no. 4, pp. 387–401, 2007.

[7] M. Asif, "Solving NP-complete problem using ACO algorithm," in *International Conference on Emerging Technologies*. IEEE Computer Society, 2009, pp. 13–16.

[8] A. Moraglio and J. Togelius, " Geometric particle swarm optimization for the Sudoku puzzle," in *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM Press, 2007, pp. 118–125.

[9] R. Soto, B. Crawford, C. Galleguillos, E. Monfroy, and F. Paredes, "A hybrid ac3-tabu search algorithm for solving sudoku puzzles," *Expert Syst. Appl.*, vol. 40, no. 15, pp. 5817–5821, 2013.

[10] F. Glover and M. Laguna, *Tabu Search*. Kluwer Academics Publishers, 1997.

[11] F. Xhafa, J. Carretero, B. Dorronsoro, and E. Alba, "A tabu search algorithm for scheduling independent jobs in computational grids," *Computing and Informatics*, vol. 28, no. 2, pp. 237–250, 2009.

[12] N. M. Razali and J. Geraghty, "Genetic algorithm performance with different selection strategies in solving TSP," in *Proceedings of the World Congress on Engineering 2011 Vol II (WCE)*, 2011, pp. 1134–1139.

[13] R. Soto, B. Crawford, C. Galleguillos, E. Monfroy, and F. Paredes, "A Pre-filtered Cuckoo Search Algorithm with Geometric Operators for Solving Sudoku Problems," *The Scientific World Journal*, vol. Article ID 465359, 2014.

[14] A. Mackworth, "Consistency in networks of relations," *Artificial Intelligence*, vol. 8, no. 1, pp. 99–118, 1977.