A Scale to Measure the Difficulty of Sudoku Puzzles

José Silva Coelho
CESUR/IST; Universidade Aberta
jcoelho@univ-ab.pt

Abstract

In the last few years, Sudoku has become a popular game, deserving the attention of many researchers. Most Sudoku puzzles are classified as easy, average and hard according to their degree of difficulty. However when asked about the criteria used to classify them, there is no clear answer. This paper presents a PnP (paper-and-pencil) method to solve and measure the degree of difficulty of Sudoku puzzles. Contrary to other methods, this method can classify the puzzles automatically without human intervention. The scale used here was inspired on the type of scale used in the snow trails: green, blue, red and black. The Sudoku puzzle is transformed to SAT, and its level is set depending on the SAT problem being easy or hard to solve in SAT, which is equivalent to the puzzle being solved using an easy or hard PnP method. A classifier and a set of classified Sudoku problems are made available on the web.

Keywords: Sudoku, instance hardness, transformation to SAT

Resumo

Nos últimos anos, o Sudoku tornou-se um jogo muito popular, merecendo a atenção de muitos investigadores. A maior parte dos puzzles de Sudoku é classificada como simples, média ou difícil, consoante o seu grau de dificuldade. Contudo, quando questionados acerca do critério de classificação, não existe uma resposta clara. Este artigo apresenta um método manual para medir o grau de dificuldade dos puzzles de Sudoku. Ao contrário de outros métodos, este método classifica os puzzles automaticamente, sem intervenção humana. A escala utilizada é inspirada na escala utilizada nas pistas de neve: verde, azul, vermelho e negro. O puzzle de Sudoku é transformado num problema de SAT e o nível de dificuldade é dado pela resolução do problema de SAT, o qual é equivalente ao da resolução manual. O classificador de dificuldade e um conjunto de puzzles de Sudoku classificados estão disponíveis na Web.

Palavras-chave: Sudoku, nível de dificuldade dos exemplos, transformação em SAT

1-Introduction

Sudoku is a puzzle that consists in filling up a partial filled 9x9 grid, with numbers from 1 to 9, in such a way that no number appears twice in each column, row or 3x3 major sub-grid. Sudoku means “one solution” in Japanese so a valid puzzle only allows one solution. The problem can be generalized for other sizes. For an introduction on Sudoku and its history, we refer the reader to [Sudoku/Wikipedia 2007; Delahaye 2006; Hayes 2006].

Sudoku was analyzed by several researchers in different ways. In [Yato & Seta 2002], it was proved that a generalization of Sudoku is a NP-complete problem. In [Royle 2007], a set of instances of Sudoku with only 17 initial filled squares was built. In [SourceForge 2005], a solver and generator are produced that support different instance sizes, and a method to generate symmetric puzzles chosen among several strategies to solve the puzzle. In [Simonis 2005], the author converts the puzzle to Constraint Programming solve it with different models, describe a generator and give a measure of difficulty. In [Lynce & Ouakine 2006], the authors convert the puzzle to SAT, then solve it without search using SAT inference technique, and [Kwon & Jain 2006] propose an encoding that allows solving large instances. In [Della Croce 2006], the author formulates the puzzle as an ILP model to solve it, and describe a procedure to generate Sudoku puzzles. In [Lewis 2007], metaheuristics are used to solve Sudoku. In [Davis 2006], commonly used Sudoku strategies are described and ranked by difficulty of application and some examples of Sudoku puzzles given to show the applicability of each strategy. The author also describes a method to generate puzzles and measure their level of difficulty.

Note that even though the generic Sudoku is a NP-complete problem, solving 9x9 Sudoku puzzles with a computer program is a trivial task, either by conversion to another problem or by direct search on Sudoku. Of course, if these methods are applied to larger instances, they could prove inefficient since the computation time grows exponentially with the size of the instance. Since this problem is to be solved mainly by humans, our aim is to find an adequate scale to measure the difficulty of Sudoku puzzles solved with PnP methods.

Most of the scales used classify the level of difficulty of Sudoku Puzzles as easy, average and hard. However, when asked about the criteria used to archive the conclusion, there is no clear answer. This paper presents a PnP (paper-and-pencil) method to solve and measure the degree of difficulty of Sudoku puzzles. Contrary to other methods, this method can classify the puzzles automatically without human intervention. The scale was inspired on the type of scale used in the snow trails: green, blue, red and black.

The automatic measure of difficulty was carried out by converting the puzzle to SAT as in [Lynce & Ouakine 2006], using some SAT inference techniques. We establish then the correspondence between SAT inference techniques and the PnP method. In this way we can automatically know if a puzzle can be solved with easy inference techniques or if it requires more elaborate techniques. Also, our proposed PnP method covers all common Sudoku strategies described in [SourceForge 2005; Davis 2006] but using a simpler format. All instances analyzed are also available in the web, fully classified and ready to be solved[1]. Also available is a classifier tool that allows the classification of new Sudoku instances[2].

This paper is organized as follows: in section 2, we present the transformation to SAT and in section 3, the green puzzles with the proposed PnP method. In section 4, we start the review of the techniques available for solving the hard instances that were also applied in Sudoku, and the base of our decision for the PnP blue method and the corresponding blue puzzle definition. In section 5, we define the red and black puzzles and in section 6, we provide some computational

[1] http://jcoelho.m6.net/freeware/sudoku2/

[2] http://jcoelho.m6.net/edicao3.asp?pa=5056

studies. We end up with some conclusions in section 7. In the appendix, we show a detailed application of the PnP green method.

2-Transformation to SAT

In this section we present the transformation of Sudoku to SAT. In the Internet we can find several implementations of how to solve Sudoku directly (see [Sudoku/Wikipedia 2007; Sudoku-Goku 2005; SourceForge 2005]), as well as through conversions to other problems [Lynce & Ouakine 2006; Della Croce 2006; Simonis 2005]. We will transform the problem to SAT and define the hardness of the problem based on the complexity of the techniques used to solve the equivalent SAT problem. Other measures of difficulty were suggested that are also based on the hardness of the complexity of the techniques used [Simonis 2005; SourceForge 2005; Davis 2006], but unfortunately these measures cannot be easily used.

A SAT instance is a propositional formula in which a solution is an assignment of all variables that satisfies the formula. All propositional formulas can be transformed to conjunctive normal format, and in this way we can define a SAT instance as a set of clauses, and a clause as a set of literals. A literal is a variable or its negation. A solution is a set of literals that satisfies all clauses, in such a way that at least one literal in each clause is in the solution, and the solution does not contain a literal and its negation.

In less technical terms we can say that a clause is a constraint that contains a set of possible scenarios, the literals. A literal represents the scenario of a specific variable to be true or false. To obtain a valid solution, it is required to find a set of scenarios that satisfies all constraints. Some of the scenarios are incompatible: a variable cannot be true and false at same time. We will use our less technical terms in the rest of the paper.

In Sudoku, the scenarios considered will be that a specific number k can be or not in a specific square sx,y. That was done in [Lynce & Ouakine 2006], the authors considered two possibilities to add the constraints that force the Sudoku rules: a minimal codification and an extended codification. The names of the codifications are clear. In the first, the authors use the minimal number of clauses needed to force the Sudoku rules and in the second, include extra clauses to help SAT inference.

The minimal SAT codification:
a) In each square there is one number at most;
b) All numbers must appear in all regions (columns, rows and sub-grids).

These two rules are enough to verify all Sudoku rules.

The extended SAT codification:
c) In each square there is at least one number;
d) For each region (column, row and sub-grid), all numbers must appear at most once.

These two rules are normally declared in Sudoku explanation rules. In [Della Croce 2006], the ILP formulation is equivalent to the extended SAT codification, and also in [Lynce & Ouakine

2006], the authors found the extended codification, better. We use also the extended SAT codification.

Why are we also coding Sudoku in SAT? The main reason is for using the SAT solvers and measures the type and number of inferences in solving SAT and extrapolates the same to Sudoku in two ways: the scale of difficulty and the PnP method to apply the corresponding SAT inference rules in Sudoku. We start by the easiest problems in SAT, in the next section.

3-Green Puzzles

The easiest SAT instance is an instance that can be solved using only unit clauses and propagation rules (see [Davis & Putman 1960]). A constraint (clause) is a set of possible scenarios (literals) and if only one is possible then that scenario must be in the solution (unit clause rule). When a scenario is in the solution, all the constraints that contain that scenario as a possibility are automatically satisfied and the inverse scenario must be removed from all other constraints since it will not be in the solution. This is what the propagation rule does: since the knowledge that a scenario is in a solution is propagated to other constraints, and some constraints are satisfied, others will be with one scenario less, eventually some constraints will became with only one scenario allowing to apply the unit clause rule.

The Sudoku green puzzles can be solved by the unit clause and propagation rules only using extended SAT codification. We must build the PnP method equivalent to the unit clause and propagation rules, method that we call the green method. This method does not require space to write numbers outside the puzzle or more than one number in each square of the puzzle. So in the standard 9x9 Sudoku puzzle, it only requires a 9x9 grid, and the human capacity to mental mark the grid.

The green method:
1) Select a number k;
2) Mentally mark regions (columns, rows and sub-grids) where k is present;
3) Check if there are regions with only one empty square not mentally marked. Write k in those squares;
4) Return to step 1 or continue to step 5 after processing all numbers without adding any number;
5) For each empty square, if all numbers exist in the regions where the square is in, except one, write the missing number in the square;
6) If there are some changes return to step 1; otherwise the method stops.

In the appendix there is a small example of how to apply the green method to a small instance, step-by-step. Note that step 3 must check all regions, not only the sub-grids but also the columns and rows. In figure 1, we present just two steps, first selecting the number 5 and marking all regions. The columns and rows marked have only one square that is not empty and not marked so the number 5 must be inserted in those squares (in italic). In the second step, we select the regions with only one empty square (step 5). A column and a row have only one empty square, so the number of the square is the missing one in the region. Step 5 can be used in a more general way because all the other numbers can be in more than one region.

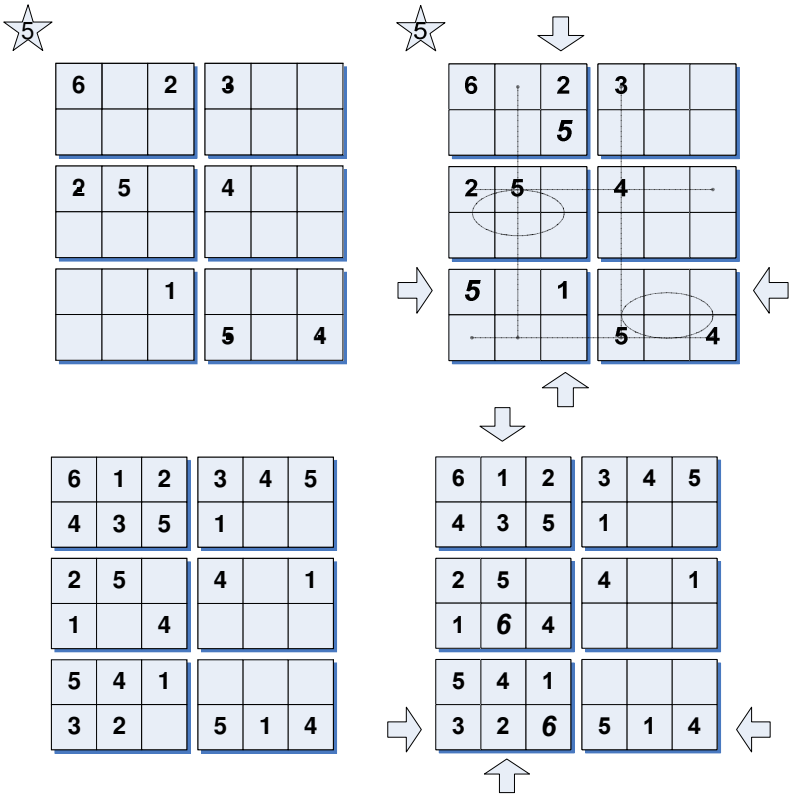


Figure 1 – Two steps in the green method: at top, selecting number 5
(steps 1 to 3), at bottom selecting a region with only one empty square (step 5)

This method is also described as the "Scanning method" in [Sudoku/Wikipedia 2007], as the "Obvious Strategies" in [Davis 2006], and as "Single Candidature" in [SourceForge 2005]. The selection of the number k is not important, and step 5 can be executed anytime, as long as all numbers are selected and no changes occurred before the method ends. Changes in the order of execution of the steps only affect the number of steps required to complete the method not the final result. We advise to select, in step 1 the high frequency numbers first, but there are no guaranties that the selection will reduce the number of steps required to solve the problem.

Now that we have described the method, we must show that it is equivalent to the use of unit clause and propagation rules in the extended SAT codification of Sudoku, in order to mark Sudoku instances as green if the SAT instance is solved with the inference rules specified.

Since in the extended SAT codification of Sudoku, for each region, all numbers must appear at most once (rule d), we can conclude that a number k cannot be in the regions of squares where k is already in. In step 2 of the green method, marking the squares in those regions is equivalent to propagate the negative scenarios associated with k in SAT. Also, we can conclude that k cannot be on filled squares, since in each square can be at most one number (rule a). Since all numbers appear in all regions (rule b), we can conclude, in step 3, that the regions were k is missing, if we have only one empty square not marked, then that square needs to be k (unit clause rule).

When we have a region with only one square empty (step 5), since all numbers appear in all regions (rule b), and in each square there is at most one number (rule a), in that region the missing number cannot be in the other squares, so it must be in the empty square (unit clause rule).

The green method does not solve all Sudoku puzzles. Sometimes is not possible to make more deductions using the green method. We define the green indicator as the number of empty squares that remain after applying the green method.

In [Sudoku-Goku 2005], it is a web page for 9x9 puzzles, which allows the introduction of puzzles and produces a step-by-step resolution. If no advanced techniques are selected, the page solves only the green puzzles. This page is ideal for a beginner, or for an advanced reader that wants to jump to the hardness of the problem.

This indicator gives a notion of the hardness of puzzles that are harder than green. It does not distinguish green puzzles. We can compute this indicator automatically using the SAT instance. In SAT it will be the number of squares that have some unsigned variables, after applying the unit clause and propagation rules.

4-Blue Puzzles

We stated earlier in the text that we will define the hardness of the problem based on the complexity of the techniques used to solve the equivalent SAT problem. The first technique is the unit clause and propagation rules. We will consider the following techniques for puzzles that are harder than green (see [Lynce & Ouakine 2006] and [Coelho 2004]):
1) Search;
2) Failed literal rule (FLR);
3) Metaheuristics;
4) Equivalency, clause subsumption, simple resolution (blue);
5) Equivalency, clause subsumption, simple resolution, ignore i (red).

The search involves guessing a value for an empty square, and reapplying the green method. If a contradiction is found, then undo the changes and try another value for the same square. If the guess is not enough to solve the entire problem or to find a contradiction, then a second guess is required and so on. The guesses are applied recursively, and the maximal number of pendant guesses (Search level) is the measure of the hardness of this method. This indicator can be easy calculated by solving the SAT instance, but is also dependent on the selected squares and values for guessing. Given a different guess, the level of a puzzle can be different. So this indicator must be used carefully. This method always solves the puzzle, but cannot be easily implemented as a PnP method since it may require undoing for an unspecified number of times that would eventually require an unspecified number of sheets of paper. This is not our choice for defining the blue puzzle.

The equivalent in Sudoku to the failed literal rule in SAT consists in guessing a value for an empty square, and then applying the green method. If a contradiction is found then undoing deduces that the guessed value cannot be in that square. If no contradiction is found, then undo and try the opposite, that is, the guessed number is not in that square. Again, if a contradiction is

found undo and then deduce the guessed number in that square. Otherwise just undo and guess another value for another square. The number of guesses here is the hardness indicator, since it is more or less correlated to the number of the undo operations. Again, the number of guesses can differ depending on the selected square. If the puzzle is not solved, then the number of empty squares (ESFLR) is used to measure the hardness of the puzzle. In [Lynce & Ouakine 2006], it was reported that all puzzles in [Royle 2007] are solved by this method. In [Sudoku/Wikipedia 2007; Simonis 2005; SourceForge 2005], several strategies that are equivalent to the failed literal rule in SAT: "what-if"; "guess-and-check"; "backtracking"; "Nishio"; "Shaving", are described. This method contains undoing operations that makes it hard for a PnP method. So this method is not also our choice for defining the blue puzzle.

The metaheuristics approach was used by [Lewis 2007]. The author implemented a simulated annealing algorithm. These methods explore the space of complete solutions, which in Sudoku means a complete filled grid. The goal is to find the complete solution with the best value for a fitness function which, in this case, is a counter on the number of violations in the grid. Even a light metaheuristic represents a very hard task in a PnP method. First it must fill the empty squares at random and then count the number of violations. Then it must implement some type of neighborhood. So it would require a sheet of paper for each neighbor. We have also not chosen this method to define the blue puzzle.

The blue technique is stated as using more advanced SAT inference techniques, such as: equivalency, clause subsumption and simple resolution. If a problem can be solved with these techniques then it is a blue puzzle. The red technique adds the "ignore i" simplification that we will explain in the next section.

The most important simplification for Sudoku is the equivalency. If we have two scenarios A and B and two constraints, one specifying that scenario A must be in the solution, or that scenario B must not be in the solution, and the other specifying that scenario A must not be in the solution or that scenario B must be in the solution, then we can conclude that both scenarios A and B will be in the solution or neither one will be. This means that we can replace scenario B by scenario A in all constraints.

The clause subsumption states that if we have two constraints A and B and constraint A contains all scenarios of constraint B, then we can remove constraint A since we know that constraint B will guarantee a scenario to satisfy constraint A.

The simple resolution states that if we have two constraints with only two scenarios each and both contain scenario A and one contain scenario B and the other contain scenario not B, then we can conclude that scenario A must be in the solution since whether or not scenario B is in the solution, one of the constraints will force scenario A to be in the solution.

The PnP method that we present is equivalent to those techniques, method that we call the blue method. This method requires space in a square enough to write at most 9 numbers in each square, in a 9x9 puzzle. It is more complex than the green method. But considering the alternatives it is simple and we only use one sheet of paper.

The blue method:
1) Apply the green method;
2) Apply steps 1 to 3 using the green method again for all numbers, except in step 3 write the number k in small in all empty squares not marked;
3) Optional steps (if something is done, go to step 1):

a) If in a sub-grid, only one column (or row) has a small number y, exclude y from the squares in the rest of that column (or row);

b) If in a column (or row), only one sub-grid has a small number y, exclude y from the squares in the rest of that sub-grid;

c) Find a set of s empty squares in the same region (column, row or sub-grid) that has only s small numbers. In this case, it is possible to remove any of those small numbers in the rest of the region;

d) Find a set of s numbers in the same region that are only in s squares. In this case, it is possible to remove any small numbers that do not belong to s but are in the same squares as s numbers.

4) Select a small number k that: is in a square with only one other number, or in a region where only other number k exists. Write a new letter next to k (A).
5) Propagate the letter (A):

a) if in a square where the letter A (or /A) is next to a number there is only other number, write /A (or A) next to the other number;

b) if a number k with the letter A (or /A) is in a region with only other number k, write the letter /A (or A) next to the other number;

c) delete small numbers k in a region if there is the same letter A and /A next to two k small numbers;

d) if in a square two small numbers have a letter A and /A, delete the rest of the small numbers in that square;

e) repeat step 5 until the letter cannot be expanded.
6) Check for contradictions for A (or /A):

a) in the same square, A (or /A) appears more than once, then select all numbers with the value /A (or A);

b) in the same region, A (or /A) is next to two numbers k, then select all numbers with the value /A (or A).

7) If something is done since step 1, go to step 1, otherwise go to step 4. If there are no more small numbers k in the condition specified in step 4 and then stops.

An example of an application of the blue method makes the text too long, so we leave to the reader the application of this method.

In step 2, the small numbers mean the possible scenarios that are not yet discarded for that square. Since the green method stops, the empty squares will have at least two small numbers in each square, after step 2 is finished. Step 2 cannot be incomplete before preceding otherwise the wrong conclusions can be drawn.

Step 3 is optional but it is advisable to make it since it is normally easy to do most of the simplifications possible before entering in step 4 assigning letters. Steps 3.a) and 3.b) are valid due the subsume clause rule. Since number y must be in the sub-grid and also in the column, since that column is the only one in the sub-grid with a small number y, all other possible small

numbers y in the rest of the column not in the sub-grid must be deleted, since y must be set in the sub-grid and cannot be twice in that column. In SAT, the constraint inserted, forcing the number y in the sub-grid, as the only scenarios in the constraint inserted, forcing the number y in the column, so the later constraint can be removed by cause subsume.

For step 3.c) since those s squares require a number and there are only s numbers, no more than those numbers can be used anywhere in the region. The step 3.d) is the inverse since the s numbers must all be in that region and are only in s squares, so other small numbers in those squares can be deleted. This will result also from clause subsumption in conjunction with equivalency.

Steps 4 and 5 are the application of the equivalency simplification in a very direct way, but steps 5.c) and 5.d) require simple resolution to remove the other numbers from the squares or regions where A and /A are present at the same time.

The step 6 has deductions from the simple contradictions, forcing conclusions.

The number of small numbers that need to be written in step 2, can be seen as a hardness indicator for blue puzzles, since it is more precise than the green indicator. This indicator can also be automatically calculated, since it corresponds, in SAT, to the number of unsigned variables.

Some steps in the blue method include known simplifications. In [SourceForge 2005], 3.a) and 3.b) are called “Single Sector Candidates” and 3.d) is called “Disjoint Subsets”. In [Davis 2006], 3.a) and 3.b) are called “Locked Candidates”, 3.c) is called “Naked Pairs” in [Davis 2006], and 3.d) is called “Hidden Pairs”. Step 3 is optional. The steps 4-6 contain these simplifications, but applying step 3 can eventually reduce the number of steps. Other strategies reported in [SourceForge 2005; Davis 2006] are also covered by this method but the difficulty of applying those strategies is harder than applying the steps 4-6. The only exception is the unique solution constraints, described in [Davis 2006], which make use on the fact that only one solution exists in Sudoku. This is difficult to codify in SAT because we need to change the SAT solver to take advantage of this information. This was not done in this work. We will use the Sudoku puzzles given in [Davis 2006] to demonstrate the application of each simplification, and to verify if those instances are solvable with the blue method.

5-Red and Black Puzzles

For the puzzles that cannot be solved by the blue method, we add a simplification “ignore i” that was introduced in [Coelho 2004] and consists in applying general resolution in order to remove one selected variable i, that is the variable that will add less number of clauses. The method stops after no variable can be removed without increasing more than a specific limit number of clauses. We set the limit to 16 clauses.

The red method is quite similar to the blue method except that in step 5 we consider more than one variable. In the squares with several variables next to the small numbers in that square, for instance, A and /B, the clause (/A or B) must be added to a set of clauses needed to register dependencies between variables. The same must be done for variables next to the same number k in a region. This method requires some basic deductions:

1) if two clauses exist (A + B) and (A + /B), then we can deduce A;
2) if two clauses exist (A + /B) and (/A + B), then we can deduce A=B, and replace B by A;
3) if two clauses exist (A + B) and (/A + /B), then we can deduce A=/B, and replace B by /A;

Technique e is the simple resolution, and technique f and g is the equivalency (see the previous section). In an extreme situation, the all complex part of the puzzle is passed to clauses, and you will have a SAT instance to solve, but much smaller than the instance resulting of the conversion of Sudoku to SAT.

The blue and red indicators should be the number of empty squares after applying each of the methods. But because these methods do not guarantee to match the simplifications in the SAT equivalent problem, we have no way of automatically calculating those indicators. Instead, we use the number of empty squares after applying the blue and red simplifications in SAT. Since the red simplification uses "ignore i", which is an advanced SAT technique, we rely on that to assure that the red puzzles require more logical deductions than blue puzzles.

What should be done with puzzles that are harder than red? Our advice is to solve them with the "what-if" method. As we will see it works with all the puzzles collected in [Royle 2007], or use search. Those would be black puzzles and will be measured with Search red level but using all simplifications (red simplifications) between guesses. For each guess, we advise the duplication of the puzzle to allow "undo" if needed. A very hard alternative is to continue to add letters to numbers, and adding clauses to represent dependencies and, after the whole puzzle is represented in clauses, try to deduce something without guessing. We do not have a hardness indicator for this alternative.

6-Computational Studies

Now that we have defined the transformation to SAT, PnP techniques and hardness indicators, we have to test them in some set. In [Royle 2007], a set of Sudoku puzzles with 17 seed numbers each is collected. The number of distinct puzzles, at the time this paper was written, was 36628. No puzzle with 16 seed numbers was found to date, nor was a proof that such a puzzle does not exist, presented. These puzzles were used in [Lynce & Ouakine 2006] to test the solver using only SAT simplifications. The authors found that with only unit propagation, the minimal clause codification could not solve any puzzle but in the extended clause codification there were 47% of puzzles solved without searching.

We also ran the solver on all the puzzles and calculated the above hardness indicators. The results shown in
Table **1**, confirm the findings in [Lynce & Ouakine 2006], with a slight difference since, at that time, the instance set had only 24260 Sudoku puzzles. These results also confirm that these instances are not all very difficult, as sometimes is reported. Only 9% of them are classified as "black".

Table 1 – Distribution of Sudoku 17 instance set by puzzle type

Puzzle Type	Instances	Percentage
Green	16867	46%
Blue	14648	40%
Red	1995	5%
Black	3118	9%
Total	36628	100%

shows the mean and the maximum value of each indicator, for the Sudoku 17 instance set. The green instances are as expected with all indicators in zero. No difference can be found in red and black instance sets.

Table 2 – Mean and maximum values of indicators by puzzle type of Sudoku 17 instance set

Indicator		Green	Blue	Red	Black
Search level	Mean	0,0	2,5	3,0	3,3
	Maximum	0,0	18	15	15
Number of Guesses	Mean	0,0	13,2	21,5	24,6
	Maximum	0,0	205	228	225
Small Numbers	Mean	0,0	141,8	159,5	182,9
	Maximum	0,0	309	311	310
Green Indicator	Mean	0,0	39,8	42,9	46,8
	Maximum	0,0	64	64	64
Blue indicator	Mean	0,0	0,0	39,5	44,4
	Maximum	0,0	0,0	64	64
Red indicator	Mean	0,0	0,0	0,0	44,2
	Maximum	0,0	0,0	0,0	64
Search red level	Mean	0,0	0,0	0,0	2,4
	Maximum	0,0	0,0	0,0	9

In Table 3 is the absolute frequency of instances by the two black indicators, the Search level and Search red level. Applying the search, using the red method, requires on average a lower number of consecutive guesses, but there are cases where the Search level is 1 and Search red level is 8. This means that it is not always useful to simplify as much as possible before guessing.

Table 4 presents the results of the instances in figures of [Davis 2006]. Those instances are used in the paper to demonstrate the application of each simplification. All the instances with complex simplifications are solved with the blue method, except the instance of figure 7-right and figure 23 that are black, and figure 19 that was solved with the red method. Figure 19 demonstrates the uniqueness constraint that is not incorporated in SAT, and figure 23 is an example of a very hard instance that does not apply any simplifications. Our blue and red method cannot do anything either. The figure 7-right demonstrates the application of swordfish, but the application of this simplification in this puzzle does not solve the puzzle. These results confirm that our automatic

classification can distinguish the puzzles that can be solved without any type of guessing, the non black types, from the ones that probably cannot be solved without some type of guessing, the black ones.

Table 3 – Frequency of Search level vs Search red level of Sudoku 17 instance set

	Search red level									
Search level	0	2	3	4	5	6	7	8	9	Total
0	16867									16867
1	6370	605	89	26	5	2	1	2		7100
2	3843	519	112	35	13	5	1		2	4530
3	2424	410	92	27	16	7		1		2977
4	1659	286	72	27	13	3	1	1		2062
5	980	174	66	19	11	2				1252
6	573	99	51	18	5	3	3		1	753
7	371	77	19	12	8		2	1		490
8	193	35	16	10	3					257
9	115	21	10	6	4	1				157
10	55	12	5	3	2	2	1		1	81
11	33	10	7	1	2		1			54
12	6	7	2	1						16
13	10		2							12
14	6	1	2							9
15	3	1								4
16	6									6
18	1									1
Total	33515	2257	545	185	82	25	10	5	4	36628

Table 4 – Indicators results in Tom Davis instances

Figure	Green indicator	Small Numbers	Blue indicator	Red indicator	Search level	Search red level	Number of Guesses	ESFL	Type
1	0	0	0	0	0	0	0	0	Green
2	0	0	0	0	0	0	0	0	Green
3	29	74	0	0	1	0	2	0	Blue
7 left	28	78	0	0	1	0	6	0	Blue
7 right	44	145	44	44	2	2	12	0	Black
11 left	34	91	0	0	1	0	1	0	Blue
11 right	41	108	0	0	1	0	2	0	Blue
13	39	117	0	0	1	0	1	0	Blue
14	19	39	0	0	1	0	1	0	Blue
16	29	73	0	0	1	0	1	0	Blue
17	31	75	0	0	1	0	1	0	Blue
19	33	94	33	0	2	0	1	0	Red
22	28	78	0	0	1	0	6	0	Blue
23	53	203	53	53	8	3	117	0	Black

In all the instances tested, no instances were found that cannot be solved with the failed literal rule. That does not mean that they do not exist in a 9x9 Sudoku. It just means that the probability is very low.

7-Conclusions

A PnP method was presented to solve a generic Sudoku puzzle which contains all common strategies used and a measure of difficulty of a puzzle was proposed. Several hardness indicators were also calculated giving the human solver more information about the hardness of the puzzles that are being solved. In the web there are published 146,628 puzzles classified and ready to be printed and solved[3], and is also provided a classifier tool that allows the classification of new Sudoku instances[4].

As a future work we can point the need of identifying puzzles easier than the green puzzles, since the green method could still be hard for some people to use. It is also missing a proposal for a black PnP method. In the green method, is missing a rule to determine in the step 2, the order of the numbers to select, in order to minimize the total number of steps required to solve a green puzzle.

References

Coelho, J., 2004,"Modelo Genérico para Gestão de Projectos: SATPSP", tese de doutoramento, UTL/IST, http://jcoelho.m6.net/freeware/tesedoc.pdf

Davis, M. and H. Putman, 1960, "A computing procedure for quantification theory", Journal of the Association for Computing Machinery, 7:201-215

Davis, T. 2006, "The Mathematics of Sudoku", http://www.geometer.org/mathcircles

Delahaye, J.P., 2006, "The science behind Sudoku", Scientific American, pp 80-87, http://www.cs.virginia.edu/~robins/The_Science_Behind_SudoKu.pdf

Della Croce, F., 2006, "Sudoku and OR", Euro Newsletter

Hayes, B., 2006, "Unwed Numbers", American Scientist, pp 12-16, Vol. 94, http://www.americanscientist.org/content/AMSCI/AMSCI/ArticleAltFormat/2005125111619_866.pdf

Kwon, G. and H. Jain, 2006, "Optimized CNF Encoding for Sudoku Puzzles", Proceedings of LPAR

[3] http://jcoelho.m6.net/freeware/sudoku2/

[4] http://jcoelho.m6.net/edicao3.asp?pa=5056

Lynce, I. and J. Ouakine, 2006, "Sudoku as a SAT Problem", Proceedings of AIMATH, http://citeseer.ist.psu.edu/744249.html

Lewis, R. 2007, "Metaheuristics can solve sudoku puzzles", Journal of Heuristics, 13:387-401

Royle, G., 2007, "Minimum Sudoku", http://people.csse.uwa.edu.au/gordon/sudokumin.php

Simonis, H. 2005, "Sudoku as a Constraint Problem", Workshop on Modeling and Reformulating Constraint Satisfaction Problems,
http://www.inf.tu-dresden.de/content/institutes/ki/cl/study/winter06/fcp/fcp/sudoku.pdf

Yato, T. and T. Seta, 2002, "Complexity and completeness of finding another solution and its application to puzzles", Proceedings of the National Meeting of the Information Processing Society of Japan, http://www-imai.is.s.u-tokyo.ac.jp/~yato/data2/signal87-2.pdf

"A Sudoku Solver", SourceForge.net, 2005, http://act365.com/sudoku/

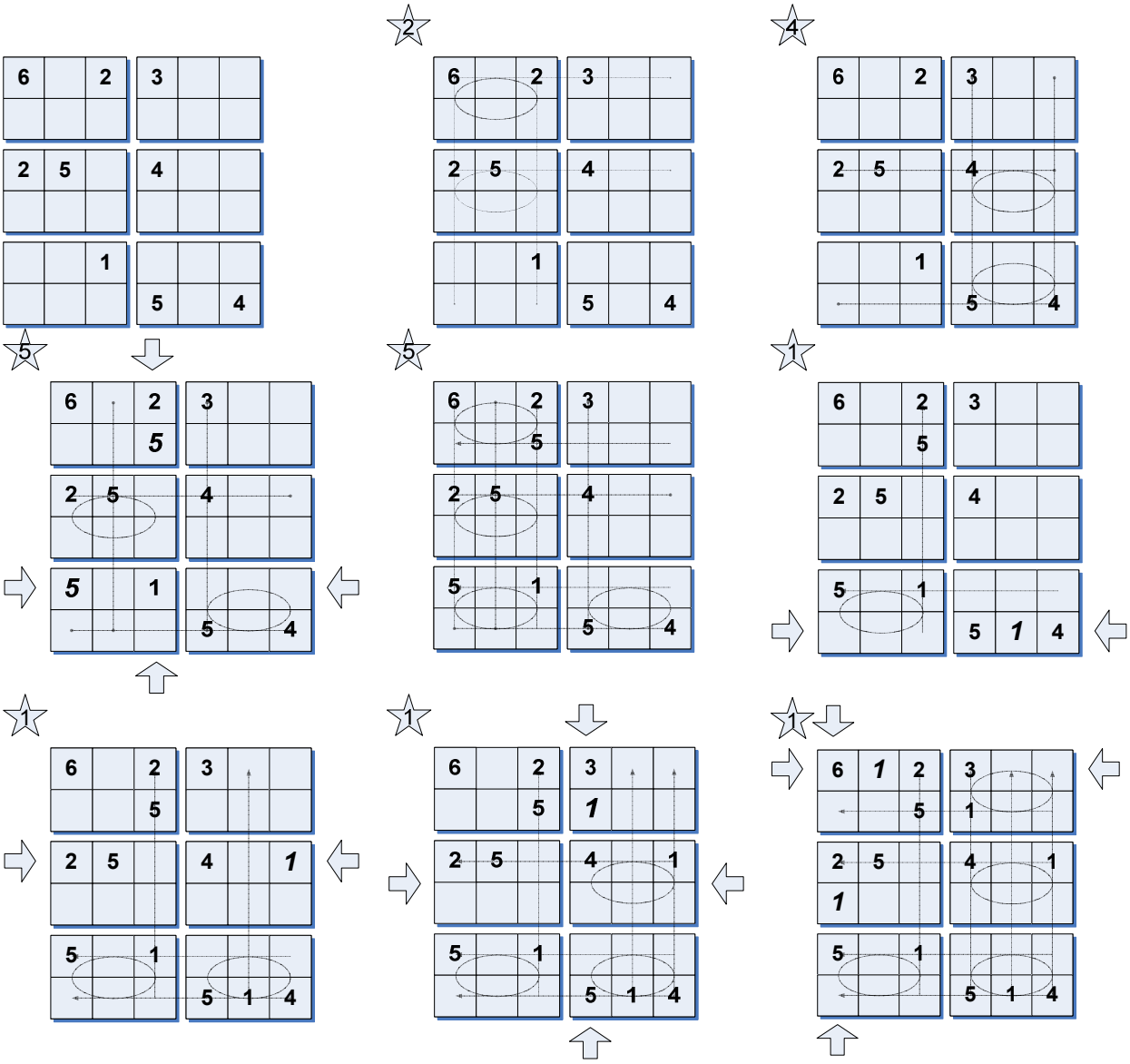
"Sudoku", Wikipedia, 2007, http://en.wikipedia.org/wiki/Sudoku

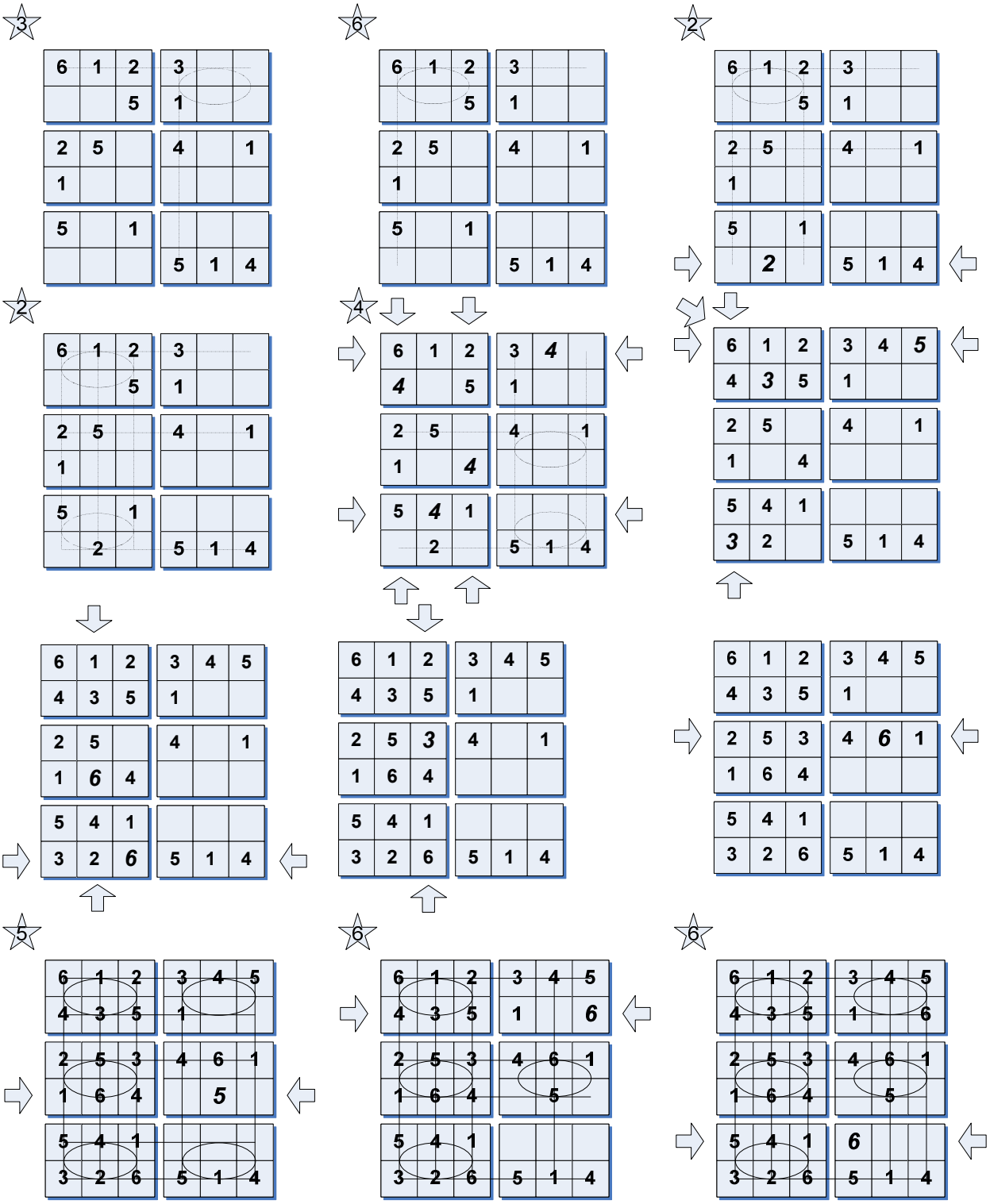
"Sudoku-Goku", 2005, http://www.sudoku-goku.com

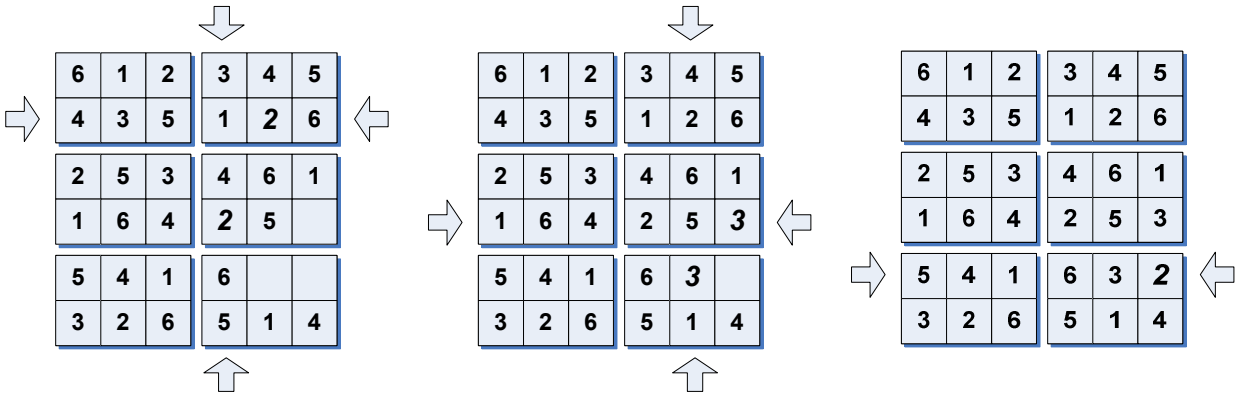
Appendix – Green method example

In this appendix we give a step-by-step example of application of the green method, to a puzzle 3x2.

The green method:
Select a number k (high frequency first);
1) Exclude columns, lines and boxes where k is in;
2) Check if exist columns, lines or boxes with only one square not excluded. Write k in those squares;
3) Return to step 1 or continue to step 5 after processing all numbers;
4) For each empty square, if all numbers exist in the column, row or box, but one, write the missing number in the square;
5) If there is some changes return to step 1, otherwise stop.




3
6
2
2
4
6
5
6
6


6 1 2 3 4 5
4 3 5 1 2 6
2 5 3 4 6 1
1 6 4 2 5
5 4 1 6
3 2 6 5 1 4
6 1 2 3 4 5
4 3 5 1 2 6
2 5 3 4 6 1
1 6 4 2 5 3
5 4 1 6 3
3 2 6 5 1 4
6 1 2 3 4 5
4 3 5 1 2 6
2 5 3 4 6 1
1 6 4 2 5 3
5 4 1 6 3 2
3 2 6 5 1 4