Procedures to find solutions in "multiple-solution" Sudoku puzzles

Jacques Paris

30.03.2006, rev. 06.04.2006

Wishing to explore more fully the issue of multiple solutions Sudoku puzzles, I developed a procedure combining the use of my "Template for helping in solving Sudoku puzzle" (see related documents on www.paris-pc-gis.com/LUDIK/varia_main.htm under the Sudoku header) with that of a program available on the web for designing and solving those puzzles "SudokuJM.exe" (www.sudokujm.com).

"Template" and "Program" issues

SuddokuJM is a much more sophisticated and powerful tool than my template but both have some elements in common that allow some useful comparisons. Let us start with the rules on which those tools are built. The template uses what I call the basic rules (each integer present once in each row, each column and each group of 3*3 cells); these rules are implemented in the program by what is identified as "isolating singles" ; the program when controlled for doing only "singles" and the template yield the same results.

SudokuJM goes further by applying "logical" rules[1] of increasing complexity that may be sufficient for reaching a final solution. But sometimes the program stops when it has exhausted all its logical resources with a partially filled grid. This "puzzle cannot be solved" outcome is different from an "illegal grid" that happens when the program encounters some impossibility or some break in the basic rules while trying to solve a puzzle. A partially solved puzzle is a legal proposition, but it does not contain enough information for reaching a solution; some external input is required.

I have shown that automatic assignations with the template do not necessarily lead to a solution; in case of stoppage, I have recommended adding to the puzzle one of the possible values in a cell where two values could be assigned, hoping that this addition would be enough to start a new automatic assignation phase; and I proposed to name the combination choice + automatic assignations a step. I further remarked that puzzles that can be solved in one step have a single solution.

That notion of step can be applied also in the context of the SudokuJM program. If the program is limited to the use of the basic rules (isolate singles), the steps will be identical to those of the template. If the program is let free to go the further it can, the assignations in a given phase may proceed further than in the template as we shall see later in the P-92 example. It is even possible for a "template step" not to appear at all, and that is precisely the case for those puzzles that I have identified as single-step puzzles: SudokuJM solves them directly but after calling some "higher level rules"; that is perfectly consistent with author's statement that his program can solve any single solution puzzle and that my contention that direct and single-step puzzles (in the "template" perspective) have single solutions.

Identifying "multiple solution puzzles"

A multiple solution puzzle exists when more than one value can be assigned to each one of a set of cells. The minimum requirement is a "quad"[2] of 4 cells to which two values can be assigned, with the resulting two combinations ab/ba and ba/ab. It is only because the application of the built-in rules fails to solve a puzzle that such situation can occur but once it exists, it is hard to track all the consequences and to be sure that all the possible solutions are identified.

A multiple solution puzzle is thus one that cannot be solved directly or in one step when using the "Template" or automatically by the SudokuJM.exe program using basic and higher level rules.

---

[1] These logical rules are identified as "isolate doubles", "isolate triples", isolate quadruples", "clear square" and "clear impossible digit".

[2] "Quad" means that the 4 cells are the 4 corners of a "horizontal" rectangle that, if collapsed, would form a square.

Finding “multiple” solutions

The objective here is to find all possible solutions to a puzzle; it is different from that of finding a solution in the perspective of minimizing a criterion, as I have done in the article about measuring the difficulty of a puzzle by following one branch of the solution tree, selecting at each node the choice that would minimize the “degrees of freedom” resulting from the choice and the possible resulting automatic assignations.

In order to optimize a procedure, I have defined several elements of what can be viewed as a decision tree. These remarks may seem somewhat abstract but they have been validated by some serious practical experience.

General procedure

The first step is to treat the puzzle for automatic assignations; this is done once and the result of that treatment is input to the initial node. The standard loop starts next.

The candidates for the next step are identified; “candidates” are the partial puzzle input to this node plus one of all the possible assignations (a value for a given cell) that are detected in all the “binary” cells (cells with 2 potential values) of this partially solved puzzle. There are twice as many candidates than binary cells.

Each candidate is treated for automatic assignations. There are three possible outcomes:

- the puzzle is entirely solved: the final grid should be compared to other solutions (if any yet) and is retained only if different.

- the puzzle is partially solved: the new partial grid if compared to those already recorded for that node (if any yet) and is recorded as a new pattern only if different.

- the puzzle is an illegal (impossible) grid: that is the end for that candidate but its alternate (formed with the other value of the binary cell) should be acceptable.

  It would be totally illogical for both potential values that could be placed in a cell to lead to impossibilities; that would imply that the partial puzzle input in that node would be “impossible” and the procedure makes sure that such patterns are not retained for the next step.

From the initial node, branches are drawn to next level nodes, one for each new pattern recorded for the previous node. Impossible situations are simply cast off and eventual solutions recorded as “dead-end” branches.

The procedure loops around with the identification of candidates for a new node.

Implementation at any node

The input is a pattern for which we know the number of potential values for each of the non-assigned cells (Global Performance Table of the template, or the displayed partially solved puzzle in the SudokuJM window after isolating singles). All “binary” possibilities are recorded in a format such as 12.1, 12.6, 13.4, 13.6 … (12.1 = row 1 column 2 value 1); each cell identifier is repeated twice with two different values.

The candidates are formed by the pattern plus one of the “binary” possibilities. I create a txt file for input into SudokuJM by repeating the original pattern as many times as required and adding in each one the required “binary” value.

The file is imported in SudokuJM. Each grid is run in automatic mode and the outcome recorded in a table as solved, partially solved or impossible.

The candidates that are “possible” are then processed in the template and the outcome treated as indicated in the general procedure.

Comments about that mode of implementation

1 – SudokuJM is used essentially to identify the impossible patterns. This particular property can save time because in automatic mode, SudokuJM may go further than what would be allowed in a single step with the template;

impossibilities can be thus uncovered much earlier and that can reduce the number of candidates formed at one node.

2 – One can imagine easily an all SudokuJM procedure taking advantage of the program ability to store an intermediate grid and to print (to paper or to pdf) the corresponding grid with all the possible values. I have not gone that way yet because I wanted to make sure that the "solution" tree would be as detailed as possible in order to uncover all possible solutions. I intend to test the impact of the possibly shorter "branches" created in the automatic mode before making any recommendation on the subject. (see some preliminary results in Annex C)

3 – Using the template allows me to use the measure of difficulty I have proposed, i.e. the degrees of freedom of the Global Performance Table.

4 – I am planning to explore the use of the detected impossible grids in the perspective of the following reasoning: if a candidate based on a given pattern is an illegal grid, the alternate binary value for that candidate must be part of the final solution for the given pattern. As all alternate values to impossible candidates must also be part of the solution, the idea would be to add to all the candidates retained after that step of analysis the alternate values to impossible grids. I expect this addition to reduce the number of different "augmented" patterns but I have no idea of its impact on the overall performance of multiple solutions detection (see in Annex C their potential impact on the preliminary test I conducted).

A start towards the identification of multiple solutions in Gould_92

The puzzle I identified as P-92 (Fig. 1) is a mistaken transcription of the puzzle that bears the n^{o} 92 in a book written by Wayne Gould[3]. In Gould-92, the clue 35.6 (the value 6 in row 3 column 5) is indeed 34.6 (the 6 is in the column4). That error was pointed to me by the author of the SudokuJM program after many tries to solving it. What amazed me is that the P-92 was obtained accidentally by moving a clue and that move could have resulted in an illegal grid (it would have be illegal if the 6 was moved to column 6; there would be then a conflict with the 6 already present in that column in row 7).

		9						2
			3		1		8	
2		8			6			
	5				2			1
			4			8		
6					9			4
						6	4	3
		7			8	5		
1							2	

Fig. 1 Original P-92 (dof 150)

After processing with the template, that original becomes the initial pattern for the solution tree (Fig. 2).

[3] "100 Su Doku no. 1" by Wayne Gould, (Édition Générales First, Paris) originally published as "The Times Su Doku Book 1" (Harper Collins Publishers)

	9	8						2
		3	1	2	8			
2	8		6	9				
	5	6	2					1
	1	4		8		2		
6			9	1		4		
			7	6	4			3
	7		8	5				
1		9			2			

4	5	0	0	2	2	5	5	0
3	3	3	0	0	0	0	4	5
0	4	0	2	0	0	4	4	3
5	0	2	0	0	2	3	4	0
3	3	0	0	2	0	5	0	4
0	4	2	2	0	0	3	0	3
3	3	2	2	0	0	0	4	0
3	5	0	2	0	0	3	3	2
0	4	4	0	2	2	0	4	4

Fig. 2 Initial pattern (dof 109) and its General Performance Table

Initial node **A**

The identification of the new patterns to be used as initial patterns for the next level nodes is summarized in Fig. 3.

"Binary" values	Su_JM outcome	Template pattern	dof	Su_JM pattern	"Binary" values	Su_JM outcome	Template pattern	dof	Su_JM pattern
15.4		AA	8	AA	64.5		AB	80	AB+
15.5		AB	80	AB+	64.7		AA	8	AA
16.4		AB	80	AB+	73.2		AE	83	AE
16.7		AA	8	AA	73.5	imposs			
34.5		AA	8	AA	74.1		AF	105	AF
34.7		AB	80	AB+	74.2	imposs			
43.3	imposs				84.1	imposs			
43.4		AE	105	AE	84.2		AF	105	AF
46.3		AA	8	AA	89.6		AG	96	JMA
46.7		AB	80	AB+	89.9		AH	97	JMB
55.3		AB	80	AB+	95.3		AA	8	AA
55.5		AA	8	AA	95.4		AB	80	AB+
63.2	imposs				96.3		AB	80	AB+
63.3		AD	98	AE	96.4		AA	8	AA

Fig. 3 Identification of patterns formed on the initial node.

The program outcome is recorded as impossible or as a solution; blank means a partial valid pattern. Template patterns are displayed in annex A. Su_JM patterns will be discussed in Annex C.

The overall “performance” of a node can be described as in the table of Fig 4.

# of candidates	32	
yielding a solution		
different solutions		
giving a partial pattern	27	
different patterns		8
leading to an impossibility	5	

Fig. 4 Summary for initial node

First level node **AA**

As the grid is almost filled up, the number of binary cells is limited and some solutions begin to appear.

"Binary" values	Su_JM outcome	Template pattern	dof
12.1	solved	S01	
12.3		A	4
17.1	solved	S02	
17.6		B	4
32.1		A	4
32.3	solved	S01	
38.1	solved	S01	
38.3	solved	S02	
87.1		B	4
87.6	solved	S02	
88.1	solved	S02	
88.6		B	4

Fig. 5 First level node AA performance

# of candidates	12	
yielding a solution	7	
different solutions		2
giving a partial pattern	5	
different patterns		2
leading to an impossibility		

Fig. 6 Summary for node AA

Second level nodes **AAA and AAB**

At each node, there are only 4 cells not assigned and they contain only 2 different values. All candidates lead to solutions, a third one is registered, different from the two identified at the first level node AA. (Summary of solutions in Annex B)

AAA		AAB	
17.1	S02	12.1	S01
17.6	S03	12.3	S03
18.1	S03	18.1	S03
18.6	S02	18.3	S01
87.1	S03	32.1	S03
87.6	S02	32.3	S01
88.1	S02	38.1	S01
88.6	S03	38.3	S03

Fig. 6 Second level nodes AAA and AAB performance

The branches A/AA/AAA and A/AA/AAB are complete. To continue with the example, we will deal with the first level node AB.

First level node **AB**

As the initial pattern AB is much less defined than AA, it is not surprising that there are more possible partial patterns at the end of the automatic assignations. One should also noticed the relatively high number of impossibilities.

"Binary" values	Su_JM outcome	Template pattern	dof	"Binary" values	Su_JM outcome	Template pattern	dof
11.3		ABA	46	67.7		ABE	57
11.7		ABB	23	69.7	Imposs		
39.4		ABC	77	69.8		ABE	57
39.5		ABD	11	73.2		ABE	57
43.3	Imposs			73.8	Imposs		
43.4		ABE	57	74.1		ABJ	76
47.3		ABF	20	74.2	Imposs		
47.9		ABG	74	84.1	Imposs		
51.7		ABH	21	84.2		ABJ	76
51.9		ABI	68	89.6		ABK	55
52.7		ABI	68	89.9		ABL	68
52.9		ABH	21	92.6	Imposs		
63.2	Imposs			92.8		ABM	35
63.3		ABE	57	93.5		ABM	35
67.3	Imposs			93.6		ABM	35

Fig. 7 First level node AB performance

# of candidates	30	
yielding a solution		
	different solutions	
giving a partial pattern	22	
	different patterns	13
leading to an impossibility	8	

Fig. 8 Summary for node AB

Second level node **ABA**

"Binary" values	Su_JM outcome	Template pattern	dof	"Binary" values	Su_JM outcome	Template pattern	dof
17.1		ABAA	33	51.7		ABAC	21
17.6		ABAB	21	51.9		ABAD	20
21.5		ABAC	21	52.7		ABAD	20
21.7		ABAD	20	52.9		ABAC	21
23.5		ABAD	20	71.5		ABAD	20
23.7	Imposs			71.9		ABAD	20
32.1		ABAE	38	72.8		ABAC	21
32.4		ABAF	11	72.9		ABAD	20
39.4		ABAE	38	89.6		ABAG	13
39.5		ABAF	11	89.9		ABAH	27
47.3		ABAG	13	92.6		ABAC	21
47.9		ABAH	27	92.8		ABAD	20
48.3		ABAH	27	93.5		ABAC	21
48.9		ABAG	13	93.6		ABAD	20

Fig. 9 Second level node ABA performance

# of candidates	28	
yielding a solution		
different solutions		
giving a partial pattern	27	
different patterns		8
leading to an impossibility	1	

Fig. 10 Summary for node ABA

Third level node **ABAA**

"Binary" values	Su_JM outcome	Template pattern	dof	"Binary" values	Su_JM outcome	Template pattern	dof
12.6	solved	S04		51.7	solved	S05	
12.7	solved	S05		51.9	solved	S04	
18.6	solved	S05		52.7	solved	S04	
18.7	solved	S04		52.9	solved	S05	
21.5	solved	S05		71.5	Imposs		
21.7	solved	S04		71.9	solved	S05	
23.5	solved	S04		72.8	solved	S05	
23.6	solved	S05		72.9	solved	S04	
37.3	Imposs			87.6	Imposs		
37.5		13	A	87.9		13	A
38.3		13	A	89.6		13	A
38.5	Imposs			89.9	Imposs		
47.3		13	A	92.6	solved	S05	
47.9	Imposs			92.8	solved	S04	
48.3	Imposs			93.5	solved	S05	
48.9		13	A	93.6	solved	S04	

Fig. 11 Third level node ABAA performance

# of candidates	32	
yielding a solution	19	
different solutions		2
giving a partial pattern	6	
different patterns		1
leading to an impossibility	7	

Fig. 12 Summary for node ABA

Fourth level node **ABAAA**

The 13 not assigned cells of the unique pattern ABAAA are all binary and the values are interwoven. If one value is assigned to a cell, all the others are determined. We are left with all the candidates leading to one of two solutions (the details would be fastidious to transcribe and read). Both solutions are different from those identified in the branch AA(A/B). (Summary of solutions in Annex B)

"Binary" values	Su_JM outcome	Template pattern
12.6	solved	S04
12.7	solved	S05

Fig. 13 Fourth level node ABAAA performance
(only value assigned to first binary cell is shown)

Interruption of the procedure

We have just completed 3 branches: AA(A/B) and ABAAA. It is just a sliver of all possible branches and we have collected already 5 different solutions. It is quite difficult to imagine how many more are hidden there, but unearthing them without some automated tool would require becoming a medieval monk in his *transcriptorium*, a feat way beyond the years I have still to live.

The example will stop here. I hope that it will prove several things such as procedure automation is a reasonable task to farm out the proposed algorithm being rather straightforward, and there will be room to improve its efficiency once some working hypotheses have been tested (e.g. the insertion of impossibilities, the complete use of SudokuJM logical rules)

Annex A

The 8 patterns formed on the initial node

AA

5		9	8	4	7			2
4	7	6	3	1	2	8	9	5
2		8	5	6	9	7		4
8	5	4	6	2	3	9	7	1
7	9	1	4	5	8	3	2	6
6	2	3	7	9	1	5	4	8
9	8	2	1	7	6	4	5	3
3	4	7	2	8	5			9
1	6	5	9	3	4	2	8	7

AB

		9	8	5	4			2
			3	1	2	8		
2		8	7	6	9			
	5		6	2	7			1
		1	4	3	8		2	
6			5	9	1		4	
				7	6	4		3
		7		8	5			
1			9	4	3	2		

AC

		9	8					2
			3	1	2	8		
2		8		6	9			
	5	4	6	2				1
		1	4		8		2	
6				9	1		4	
				7	6	4		3
		7		8	5			
1			9			2		

AD

		9	8					2
			3	1	2	8		
2		8		6	9			
	5	4	6	2				1
		1	4		8		2	
6		2		9	1		4	
		3		7	6	4		3
		7		8	5			
1			9			2		

AE

		9	8					2
			3	1	2	8		
2		8		6	9			
8	5	4	6	2				1
		1	4		8		2	
6	2	3		9	1		4	8
		2	1	7	6	4		3
		7	2	8	5			
1			9			2		

AF

		9	8					2
			3	1	2	8		
2		8		6	9			
	5		6	2				1
		1	4		8		2	
6				9	1		4	
			1	7	6	4		3
		7	2	8	5			
1			9			2		

AG

		9	8					2
			3	1	2	8		
2		8		6	9			
	5		6	2				1
		1	4		8	6	2	
6				9	1		4	
				7	6	4		3
		7		8	5			6
1			9			2		

AH

		9	8					2
			3	1	2	8	9	
2		8		6	9			
	5		6	2				1
		1	4		8		2	
6				9	1		4	
				7	6	4		3
		7		8	5			9
1			9			2		

Annex B

The first 5 uncovered solutions

S01

5	1	9	8	4	7	6	3	2
4	7	6	3	1	2	8	9	5
2	3	8	5	6	9	7	1	4
8	5	4	6	2	3	9	7	1
7	9	1	4	5	8	3	2	6
6	2	3	7	9	1	5	4	8
9	8	2	1	7	6	4	5	3
3	4	7	2	8	5	1	6	9
1	6	5	9	3	4	2	8	7

S02

5	3	9	8	4	7	1	6	2
4	7	6	3	1	2	8	9	5
2	1	8	5	6	9	7	3	4
8	5	4	6	2	3	9	7	1
7	9	1	4	5	8	3	2	6
6	2	3	7	9	1	5	4	8
9	8	2	1	7	6	4	5	3
3	4	7	2	8	5	6	1	9
1	6	5	9	3	4	2	8	7

S03

5	3	9	8	4	7	6	1	2
4	7	6	3	1	2	8	9	5
2	1	8	5	6	9	7	3	4
8	5	4	6	2	3	9	7	1
7	9	1	4	5	8	3	2	6
6	2	3	7	9	1	5	4	8
9	8	2	1	7	6	4	5	3
3	4	7	2	8	5	1	6	9
1	6	5	9	3	4	2	8	7

S04

3	6	9	8	5	4	1	7	2
7	4	5	3	1	2	8	6	9
2	1	8	7	6	9	5	3	4
8	5	4	6	2	7	3	9	1
9	7	1	4	3	8	6	2	5
6	2	3	5	9	1	7	4	8
5	9	2	1	7	6	4	8	3
4	3	7	2	8	5	9	1	6
1	8	6	9	4	3	2	5	7

S05

3	7	9	8	5	4	1	6	2
5	4	6	3	1	2	8	7	9
2	1	8	7	6	9	5	3	4
8	5	4	6	2	7	3	9	1
7	9	1	4	3	8	6	2	5
6	2	3	5	9	1	7	4	8
9	8	2	1	7	6	4	5	3
4	3	7	2	8	5	9	1	6
1	6	5	9	4	3	2	8	7

Annex C

An optimized test

In that test, I tried to combine the use of the information given by the candidates that are found illegal and the powers of SudokuJM. But first the results obtained by using the template must be compared to those produced by the program.

The table in Fig. 3 shows some differences in the resulting patterns but these differences occur essentially through the use of the higher level logical rules (doubles, triples, etc.). If the program is limited to isolating singles (that can be done only by operating the program manually), results are identical. If the program does not find a way to use the higher level rules, “template” and SudokuJM patterns are identical.

As I expected it, the number of different patterns generated by a “free” SudokuJM” is smaller than those formed with the template. There are 6 (instead of the template 8), 3 being identical (A, E, F) one obtained by adding two values to B (74.1, 84.2) and two with too many differences to be described succinctly.

This reduction in the number of patterns becomes then a valuable feature, but it is only a start as the next topic will show.

It is easy to identify in Fig. 3 the “alternate-to-impossible” values that we want to include in the test (e.g. 43.3 is impossible, thus 43.4 is the alternate to use). All solutions to the proposed puzzle must respect that distribution of values; it is worth noting that all the solutions that have been uncovered without the specific use of that feature verify that rule (S01-S03).

4 3		
2	1 2	

Fig C-1 Alternate-to-impossible assignations

The next step is to add those values to the 6 patterns that do not contain them already. It is important to verify if these new grids are not candidate for automatic assignations; if that is the case (~1/2 in this example) assignations must be made and resulting patterns compared. If there are fewer patterns than before the addition of the alternates-to-impossible, this would be an added bonus, the solution tree having fewer branches and probably shorter ones. In this example, there are only 5 distinct patterns left compared to the original 8.

As pattern AA is identical to the corresponding one in the original example and would not most probably reveal any difference as it is almost filled up, I have chosen to begin with AB (original AB augmented by alternates-to-impossible and after automatic assignations).

First-level node **AB**

There are 24 candidates leaving 8 patterns to be tested

Second-level node **ABA**

The 26 candidates lead to 8 patterns

Third-level node **ABAA**

> All candidates produce only 2 solutions that are identical to those found with the template procedure (S04, S05).

Some comments on this "optimized" approach

Even if this test covers only a small part of the entire solution tree, I am tempted to draw the conclusions that follow, hoping that more extensive testing will corroborate them

.

1 – Introducing the "alternates-to-impossible" detected on the initial grid has an unforeseen impact; no impossible solution appears further down the tree, limiting the number of potential candidates to be tested.

> That is not exactly what turned out in subsequent tests. In one puzzle, I found some impossible situations almost at some branch ends. I did not see any added value in treating them as at the first node because alternates were leading directly to solutions, but had they happened earlier in the tree, I would have done it. I have also found no case of double impossibility which is validating the use of alternates-to-impossible as described for the first node, eliminating troubles later in the tree.

2 – The overall length of the branches seems to be reduced when combining the use of alternates and the higher level rules of SudokuJM. The template procedure identifies the solutions one step further down.

3 – Apparently no possible solution seems to be missed.

As a conclusion, I would recommend to use this kind of more tool-supported approach when investigating puzzles offering multiple solutions.