Cent. Eur. J. Comp. Sci. • 2(2) • 2012 • 143-151
DOI: 10.2478/s13537-012-0011-y

VERSITA

Central European Journal of **Computer Science**

Minimal number of clues for Sudokus

Research Article

Christoph Lass[1]*

1 *Ernst-Moritz-Arndt-Universität, Institut für Mathematik und Informatik, Walther-Rathenau-Straße 47, 17487 Greifswald, Germany*

Received 23 December 2011; accepted 23 May 2012

Abstract: In this article we will present a universal scheme for calculating the minimal number of clues needed for a generalized Sudoku to be uniquely solvable. By using equivalence partitioning and analyzing uniqueness properties of patterns, we will reduce the search space to a feasible size. As a result, we will present the minimal number for 2×4 Sudokus.

Keywords: Sudokus • Combinatorial optimization • Equivalence partitioning

Sudoku is a popular puzzle with simple rules. We are given a partially filled 9×9 matrix with 9 3×3 blocks.

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

All we have to do is add digits such that every row, column and block contains the numbers from 1 to 9. This yields interesting combinatorial problems such as given a blank matrix, how many ways are there to fill the matrix obeying the above rules? Felgenhauer and Jarvis [3] obtained $6.67 \cdot 10^{21}$ full Sudokus. We discuss a different popular problem. How many clues must a Sudoku have such that the completion is unique? The minimum for the standard 3×3 Sudoku is 17 which was proven by Gary McGuire [8] in 2012 and will be validated by Lin and Wu [6] by analyzing the essentially different solution grids.

Those papers solely focus on the 3×3 case while the minimum clues question is open for generalized cases such as

* *E-mail: christoph.lass@uni-greifswald.de*

2×4 Sudokus and as we will see this question is not trivial. Our goal is to answer the minimum clues question for arbitrary Sudoku sizes using a different approach to [6, 8] based on patterns. We will start by defining a generalized Sudoku and examine its uniqueness properties. Afterwards we introduce a canonical form to reduce the search space and show that this canonical form yields properties which result in efficient algorithms. We use these algorithms to calculate the minimum number of clues needed to ensure uniquess for a 2×4 Sudoku. Since our research does not focus on one Sudoku size we will also gain insight into the standard 3×3 case and our paper might help to answer the open question on how many essentially different, uniquely solvable 3×3 Sudokus with 17 clues there are.

1. Necessary conditions for uniqueness

Definition 1.1.

A $R \times C$ *Sudoku* is a matrix $S \in \mathbb{N}^{N \times N}$ with $N = R \cdot C$ and $s_{ij} \in \{0, \dots, N\}$. It consists of N disjoint matrices $B \in \mathbb{N}^{R \times C}$ called *blocks*. A *band* is the set of R rows which contains R blocks and a *stack* is the set of C columns which contains C blocks.
We call a Sudoku *full* if every element is positive, otherwise we call it *partial*. A full Sudoku is *valid* if each row, column and block consists of pairwise different elements. We call S^* a *solution* of S, if S^* is full, valid and $s_{ij} = s^*_{ij}$ for all $s_{ij} > 0$. We are only interested in Sudokus with a unique solution. If an element is 0, it will not be displayed in the matrix S. As of now we will call positive elements *clues*.

Russell and Jarvis [5] calculated the number of essentially different, valid 3×3 Sudokus with regards to transformations which preserve the validity. They are

(1) Permuting the numbers,

(2) Permuting the rows (columns) in a band (stack),

(3) Permuting the bands (stacks),

(4) If $R = C$: Transposing.

We refer to (1) as *number transformations* T_N and (2)-(4) as *pattern transformations* T_P. This gives us the total number of validity preserving transformations T

$$|T| = \begin{cases} N!(R!)^{C+1}(C!)^{R+1} & R \neq C \\ 2N!(R!)^{C+1}(C!)^{R+1} & R = C \end{cases}$$

We will use these transformations to reduce the search space and to obtain necessary conditions for a partial Sudoku having a unique solution.

Lemma 1.1.

Let S be Sudoku with a unique solution, then it holds:

(a) There are at least $N - 1$ pairwise different clues.

(b) Each band (stack) has at most one row (column) without a clue.

(c) There is at most one band (stack) without a clue.

(d) If $R = C$: Not all clues are located on the diagonal or the antidiagonal.

Proof. (a) is obvious for $N = 1$. Let $N \geq 2$ and without loss of generality let no clue of S be 1 or 2. If S^*_1 is a solution of S, we can obtain a new solution S^*_2 of S by permuting 1 and 2 in S^*_1. (b) and (c) can be proven in a similar way by permuting the free rows (columns) or bands (stacks) in the solution. (d) follows from transposing the solution. □

If $R, C \geq 2$ (c) follows from (b). (a) gives us a lower bound which is sharp for latin squares (R or $C = 1$) of size $N \leq 3$. We use (a) and (b) as simple tests for uniqueness.

2. Canonical minlex form

We want to define a canonical form for Sudokus which allows us to only analyze Sudokus which are essentially different, i.e. S_1 is essentially different from S_2 if there exists no validity preserving transformation $\sigma \in T$ such that $\sigma(S_1) = S_2$, since Sudokus which are essentially the same have the same properties such as number of solutions.

Definition 2.1.
The pattern $p(S) \in \mathbb{N}^{N^2}$ of a Sudoku S is defined as

$$p_{(i-1)N+j}(S) := \begin{cases} 1 & s_{ij} > 0 \\ 0 & s_{ij} = 0 \end{cases} \qquad i, j = 1, \ldots, N.$$

The first idea is to look at the pattern of a Sudoku to determine its canonical form.

Definition 2.2.
Let $v, w \in \mathbb{R}^n$. v is *lexicographically smaller* than w ($v \prec w$) if

$$\exists i \in \{1, \ldots, n\} : v_i < w_i \ \wedge \ \forall j < i : \ v_j = w_j.$$

Let $V = \{v^i \in \mathbb{R}^n \mid v^i \neq v^j \Leftrightarrow i \neq j, \ i, j = 1, \ldots, m\}$. The lexicographically smallest vector $v^* \in V$ is the *minlex vector* of V ($v^* = \operatorname{minlex} V$) if

$$\forall v \in V : v^* \prec v \ \vee \ v^* = v$$

The lexicographically smallest pattern of S

$$p_{\min}(S) := \operatorname*{minlex}_{\sigma \in T_p} p(\sigma(S))$$

is the *minlex pattern* of S. A Sudoku S is in *pattern minlex form* if $p(S) = p_{\min}(S)$.
There can be multiple transformations which lead to the same minlex pattern. We call those transformations *(pattern) automorphisms*. This term is justified by only considering Sudokus in pattern minlex form. In this case the identity is a trivial automorphism.

The following lemma shows that we do not have to consider all pattern transformations to check if a Sudoku is in pattern minlex form.

Lemma 2.1.
Let S be a Sudoku in pattern minlex form, then it holds:

(a) The rows and columns (considered in $\{0,1\}^N$ pattern form) in each band (stack) are lexicographically ordered with the first row (column) being the lexicographically smallest.

(b) The bands (considered in $\{0,1\}^{RN}$ pattern form) are lexicographically ordered with the first band being the lexicographically smallest.

(c) The stacks (considered in $\{0,1\}^{CN}$ pattern form, row by row) are lexicographically ordered with the first stack being the lexicographically smallest.

Proof. (a) is trivial for $R = 1$. Let S_1 be a Sudoku in pattern minlex form with $R \geq 2$ and without loss of generality let row 1 be lexicographically larger than row 2. By permuting those rows we get S_2 which is essentially the same Sudoku as S_1. But $p(S_2) \prec p(S_1)$, hence S_1 cannot be in pattern minlex form. The rest of (a),(b) and (c) can be proven in the same way by permuting the columns, bands and stacks. □

Lemma 2.1 gives us necessary conditions of the pattern minlex form and allows us to propose the following algorithm which examines sufficient conditions.

Algorithm 1. Checking pattern minlex form of S_1

1. For all permutations of the stacks and for all permutations of the columns in each stack:
   (a) Transform $S_2 := S_1$ according to the permutations.
   (b) Sort the rows in each band of S_2 in lexicographically ascending order.
   (c) Sort the bands of S_2 in lexicographically ascending order.
   (d) If $p(S_2) \prec p(S_1)$, S_1 is not in pattern minlex form and the algorithm stops.
2. If $R = C$: Do the same for $S_2 := S_1^\top$.

Having fixed the pattern, we now regard the number transformations to define a canonical form.

Definition 2.3.

The *representation* $r(S) \in \mathbb{N}^{N^2}$ of a Sudoku S is defined as

$$r_{(i-1)N+j}(S) := s_{ij} \qquad i, j = 1, \ldots, N.$$

The idea of our canonical form is to choose the Sudoku with the lexicographically smallest representation which is in pattern minlex form.

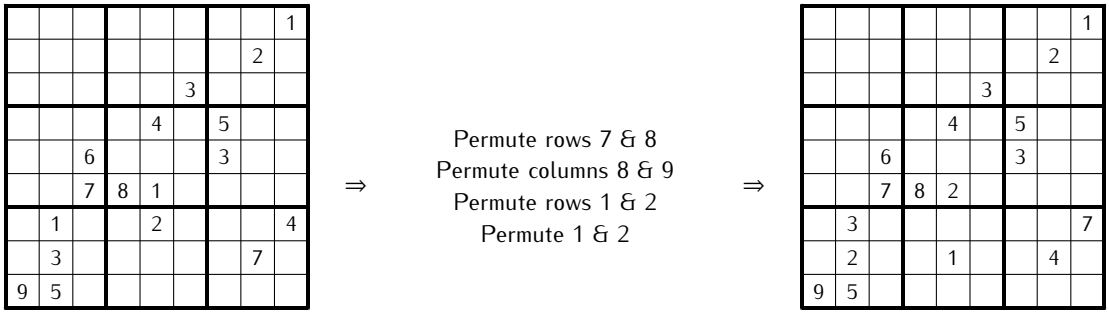
Definition 2.4.

The *minlex representation* of S is

$$r_{\min}(S) := \operatorname*{minlex}_{\substack{\sigma \in T \\ p(\sigma(S))=p(S)}} r(\sigma(S)).$$

A Sudoku S is in *canonical minlex form* if $p(S) = p_{\min}(S)$ and $r(S) = r_{\min}(S)$.

Other popular canonical forms are the lexicographically smallest representation with no restriction on the pattern (*row-minlex form*) and the lexicographically smallest representation of the solution [1] whose transformation is applied to the partial Sudoku. The latter is of no use to us since it can only be applied to Sudokus with a unique solution. The first is not necessarily the same canonical form as ours as the example below shows (first 17 clue 3×3 Sudoku of [9]).



Row-minlex form

Canonical minlex form

In both canonical forms the value of the clues are ordered in a fixed way. The first clue is 1, afterwards the first clue different from 1 is 2 and so on. The following lemma shows that this property holds for our canonical form and we actually do not have to consider all $N!$ number transformations because only one permutation is possible.

Lemma 2.2.
Let S be a Sudoku in canonical minlex form with at least $N-1$ pairwise different clues and $n(S,i)$ be defined as follows

$$n(S,i) := \min\left(\{j \mid r_j(S) = i\} \cup \{N^2+1\}\right) \qquad i = 1,\ldots,N,$$

then it holds $n(S,1) < n(S,2) < \cdots < n(S,N)$.

Proof. Let S be a Sudoku in canonical minlex form and without loss of generality let $2 = \arg\min_i n(S,i) < n(S,1)$. Then we can get a new Sudoku S^* in pattern minlex form with a lexicographically smaller representation than S by permuting 1 and 2. It follows $n(S^*,1) = n(S,2) < n(S,1) = n(S^*,2)$ and S cannot be in canonical minlex form. The other relations can be proven in a similar way. □

Together with Lemma 1.1 this implies that only N does not have to be a clue in a partial Sudoku with a unique solution in canonical minlex form. As an example we look at the above Sudokus which have the property $(n(S,1), n(S,2), \ldots, n(S,9)) = (9, 17, 24, 32, 34, 39, 48, 49, 73)$.
Therefore all we have to do is determine all automorphisms of S in pattern minlex form and apply the according number transformation such that Lemma 2.2 holds which will validate if S in canonical minlex form.

3. Minirow approach

Before we describe the whole procedure, we discuss a more sophisticated algorithm for checking the pattern minlex form which was developed by Deverin [2]. Let us take a look at the following example:

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

S

⇒ Transpose, Row order: 897 132 546, Column order: 645 213 897 ⇒

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

S in pattern minlex form

Column 8 is the only column with one clue and since every other column and row has more clues S must be transformed in such a way that column 8 becomes row 1, hence stack 3 becomes band 1. Furthermore column 9 is transformed to be row 2 and band 2 becomes stack 1 since it has no clue in the intersection of column 9 and band 2. So in this case there is only possible permutation of the rows in the new band 1 and the permutation of the stacks is also unique. Another observation is that in each intersection of row 1 and 2 with the stacks the clues are positioned in the lexicographically smallest way because otherwise a permutation of the columns would lead to a smaller pattern. So if we only regard the first two rows of S in pattern minlex form there are only 4 possible permutations of the columns in stack 2 and 3 instead of 36.
The main idea of the algorithm is to look at each row and stack permutation in each band and choose the band and the according transformations with the least amount of clues in the intersections. We introduce some definitions before we describe the algorithm in detail.

Definition 3.1.
The intersection $m_{rs} \in \mathbb{N}^C$ of row r and stack s is called *minirow*. $a(m_{rs})$ is the amount of clues in minirow m_{rs}.

The *distribution* $d(B^i) \in \mathbb{N}^{R^2}$ of band B^i is defined as follows

$$d_j(B^i) := a(m_{rs}) \qquad r = (i-1)R+1,\dots,iR \qquad s = 1,\dots,R$$
$$j = (r-1)R - (i-1)R^2 + s \qquad j = 1,\dots,R^2 \qquad i = 1,\dots,C.$$

The *modified distribution* $d^*(B^i) \in \mathbb{N}^{R^2}$ of band B^i is

$$d^*_j(B^i) := \begin{cases} d_j(B^i) & j < \min\left\{\{k \mid d_k(B^i) > 0\} \cup \{R^2\}\right\} + R \\ 0 & \text{otherwise} \end{cases} \qquad j = 1,\dots,R^2 \qquad i = 1,\dots,C$$

Note that both distributions only depend on row and stack permutations. For the above example in pattern minlex form $d(B_1)$ is $(0,0,1,0,2,3,1,2,1)^\top$ and $d^*(B_1) = (0,0,1,0,2,0,0,0,0)^\top$.

Lemma 3.1.

Let S_1 and S_2 be $R \times C$ Sudokus with S_2 being in pattern minlex form and let $B^1_{S_1}$ and $B^1_{S_2}$ be the first bands of S_1 and S_2. If $d^*(B^1_{S_2}) \prec d^*(B^1_{S_1})$, then it holds $p(S_2) \prec p(S_1)$.

Proof. The modified distribution of the first band is defined in such a way that a positive element corresponds to the first minirow having clues in the according stack. Hence, in S_2 the clues in those minirows are ordered in such a way that they have the lexicographically smallest pattern possible since otherwise column permutation would lead to a smaller pattern of S_2. $d^*(B^1_{S_2}) \prec d^*(B^1_{S_1})$ implies there existing an index k such that $d^*_k(B^1_{S_2}) < d^*_k(B^1_{S_1})$ and $\forall j < k\ d^*_j(B^1_{S_2}) = d^*_j(B^1_{S_1})$. If $d^*_j(B^1_{S_2}) = d^*_j(B^1_{S_1})$, then at best the minirow of S_1 is ordered in the lexicographically smallest way, hence the patterns of both minirows are identical and we can examine the next minirow. Otherwise the pattern of S_2 is lexicographically smaller. For index k the pattern of the corresponding minirow of S_2 is smaller in all cases since it has less clues and S_2 is in pattern minlex form. It follows $p(S_2) \prec p(S_1)$. □

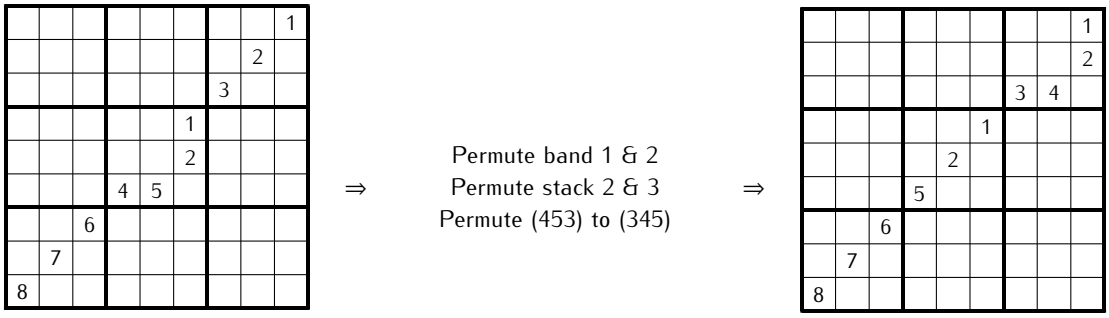
Using this property we can formulate our improved algorithm for checking the pattern minlex form.

Algorithm 2. Checking pattern minlex form of S using minirow approach

1. Let d^*_1 be the modified distribution of the first band of S.
2. For all bands B^i of S:
   (a) Compute the modified distribution $d^*(B^i)$ for each stack and row permutation.
   (b) If $d^*(B^i) \prec d^*_1$: S is not in pattern minlex form and the algorithm stops.
   (c) If $d^*(B^i) = d^*_1$:
   - Apply the row and stack permutations of step (2a) and permute band B^i and B^1.
   - Use Algorithm 1 for the modified Sudoku but fix the stack permutation and only allow column permutations which produce the lexicographically smallest pattern of the minirows in the first band where the according elements of the modified distribution are positive.
3. If $R = C$: Repeat the second step for $S^\top$.

Step 2 requires the calculation of up to $2C(R!)^2$ modified distributions which is more feasible if $R \le C$. For the above example this reduces the number of permutations in step 1 of Algorithm 1 from $2(3!)^4 = 2592$ to $(2!)^2 3! = 24$.

Deverin's original algorithm used the distribution to compare the bands and did not combine it with Algorithm 1. Though this approach might further reduce the number of permutations, it does not lead to our canonical minlex form as the following example shows.



Original algorithm

Canonical minlex form

Note that our canonical minlex form has a lexicographically smaller pattern and representation though the distribution of the first band is lexicographically larger. The modified distribution of band 1 is the same in both forms.

With some slight modifications we can use Algorithm 2 to determine the automorphisms of a Sudoku and its pattern. We further use the fact that the number of clues in each row, column, band and stack must be same in the transformed Sudoku.

Algorithm 3. Determine automorphisms of S

1. Let d_1 be the distribution of the first band of S.
2. For all bands B^i of S:
   (a) Compute the distribution $d(B^i)$ for each stack and row permutation.
   (b) If $d(B^i) = d_1$:
   - Apply the row and stack permutations of step (2a) and permute band B^i and B^1.
   - For all row, column and band permutations which do not change the first band and which preserve the number of clues in each band and the distribution in each row and column:
     – Apply the permutations to get the transformed Sudoku S^*.
     – If $p(S) = p(S^*)$: Save the automorphism.
3. If $R = C$: Repeat the second step for $S^\top$.

The first example only has the trivial automorphism with the final pattern check being reached twice. The second example has 576 automorphisms and the final pattern check is reached 20736 times. This is a worst case scenario, e.g. the algorithm detects 36 possible permutations for rows 4,5,6 and columns 4,5,6 but since they depend on each other there are only 6 permutations which do not change the pattern.

4. Results

With the above results, we can formulate a procedure to compute all essentially different Sudoku of a fixed size and with a fixed number of clues.

Algorithm 4. Computing all essentially different $R \times C$ Sudokus with a fixed number of clues

1. For all pattern p:
   (a) Check necessary pattern uniqueness conditions of Lemma 1.1.
   (b) Check necessary pattern minlex form conditions of Lemma 2.1.
   (c) Check sufficient pattern minlex form conditions using Algorithm 2.
   (d) If all conditions are met, go to step 2.

2. Determine all automorphisms of S with $p_{min}(S) = p$ using Algorithm 3 and all pairs of clues which are in the same row, column or block.

3. For all representations $r(S)$ with $p_{min}(S) = p$:

   (a) Check necessary conditions of Lemma 2.2.

   (b) Check if elements of pairs computed in step 2 have different values.

   (c) Check if no automorphism of S leads to a lexicographically smaller representation. Use Lemma 2.2 to determine the matching number transformation for each automorphism.

   (d) If all checks were positive, determine number of solutions of S and save S if it has a unique solution.

4. Compute next pattern and goto to step 1.

We determine the number of solutions of a Sudoku by using JSolve [7] which is a solver for 3x3 Sudokus based on bit operations. It can be modified to solve 2x4 Sudokus and it stops when a second solution occurs. In step (3b) we ensure the necessary uniqueness condition (a) of Lemma 1.1.
Before we present the 2×4 case, we take a look at the known results [4] for smaller cases which were verified by our algorithm.

Table 1. Minimum number of clues for $R \times C$ Sudokus

$R \times C$	Minimum number of clues	Equivalence classes
1×1	0	1
1×2	1	1
1×3	2	1
1×4	4	1
2×2	4	13
2×3	8	537

To get a initial guess for the minimum number of clues for 2×4 Sudokus we wrote a program which creates random valid Sudokus which were minimized, i.e. a Sudoku is minimal if every clue is needed to ensure uniqueness. We got 2 Sudokus with 15 clues from 500,000 randomly generated puzzles. Afterwards Algorithm 4 was used to find a Sudoku with 14 clues. The 243th pattern in pattern minlex form provided the first puzzle with a unique solution.

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

Theorem 4.1.
A 2×4 Sudoku must have at least 14 clues to be uniquely solvable. This boundary is strict.

The main task was to prove that there exists no uniquely solvable 2×4 Sudoku with 13 clues. To speed up the calculation we parallelized the algorithm which can be done by using one selected core to determine the patterns in pattern minlex form and the other cores analyze the patterns. Hence, every core except the selected one solves the essentially different Sudokus of a given pattern. The results of the calculation are as follows:

Table 2. Computational results for 2×4 Sudokus with 13 clues

Category	Results
Patterns in pattern minlex form / Iterations of step 1	32,597,396 / 80,287,800
Automorphisms per pattern	2.049
Essentially different Sudokus per pattern	489,807
Essentially different Sudokus / Iterations of step 3	15,966,439,906,845 / 17,006,634,594,811
Sudokus solved per second per core	109,749
Sudokus with a unique solution	0

It takes 7 minutes to determine all patterns in pattern minlex form. We included skipping routines to further reduce the iterations of step 1 and step 3 leading to invalid patterns or representations. The number of Sudokus solved per second per core was estimated using a cluster with a total of 28 cores (7 Core i7 870, 4x2.93 GHz, 8 GB RAM). The test problem for this estimation were the first 1000 minlex patterns with JSolve taking up 97.8 % of the running time. In this case the speedup was 23.815 and the efficiency was 0.851. Note that both values have an upper bound of 27 respectively $\frac{27}{28}$ since one core only calculates the patterns and does not solve Sudokus. Both values can be improved by sending multiple patterns to one core to reduce the influence of latency caused by message passing.
Our algorithm is suited for a parallel environment since the serial part of the computation only takes up a fraction of the running time due to the nice properties of our canonical form. This becomes more apparent for larger cases, e.g. a pattern of a 16 clues 3×3 Sudoku can have up to 698 million essentially different puzzles with JSolve solving about 60,000 Sudokus per second.

5. Conclusion and future work

We introduced a canonical form, examined its properties and developed an algorithm to determine the essentially different $R \times C$ Sudokus for a fixed number of clues which can be used to calculate the minimum number of clues needed to ensure uniqueness. In contrast to [6, 8] we used an approach based on patterns. It was proven that the minimum for 2×4 Sudokus is 14. A future project is to determine all equivalency classes of the 14 clues case. So far, we have investigated 533,000 patterns which resulted in 2,642 essentially different 2×4 Sudokus with a unique solution.
Another open question is the minimum number of clues for 3×3 X-Sudokus, which are Sudokus with no repeats on the diagonal and the antidiagonal. This restriction reduces the number of valid field transformations to 96 with the currently known [10] minimum number of clues being 12. We are interested in verifying this number and investigating the field transformations for the generalized case.

References

[1] Barker M., http://forum.enjoysudoku.com/canonical-form-t5215.html, 2007
[2] Deverin M., http://www.setbb.com/sudoku/viewtopic.php?p=7528&mforum=sudoku#7528, 2007
[3] Felgenhauer B., Jarvis A.F.: Mathematics of Sudoku I, Mathematical Spectrum 39, 15-22, 2006
[4] Gupta S., Russell E., http://forum.enjoysudoku.com/sudoclues-max-min-forest-leaves-t3351.html, 2006
[5] Jarvis A.F., Russell E., Mathematics of Sudoku II, Mathematical Spectrum 39, 54-58, 2006
[6] Lin H., Wu I., Solving the Minimum Sudoku Problem, International Conference on Technologies and Applications of Artificial Intelligence, 456-461, 2010
[7] Linhart J., http://www.setbb.com/phpbb/viewtopic.php?p=11913&sid=19a58b319044bc49573398e0ebe7a3f4&mforum=sudoku#11913, 2010
[8] McGuire G., Tugemann B., Civario G., There is no 16-Clue Sudoku: Solving the Sudoku Minimum Number of Clues Problem, eprint arXiv:1201.0749, 2012
[9] Royle G., http://mapleta.maths.uwa.edu.au/~gordon/sudokumin.php, 2010
[10] van der Werf R., http://www.sudocue.net/minx.php, 2007