Enumerating possible Sudoku grids

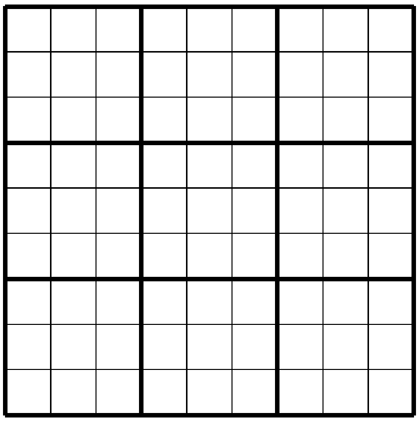
Bertram Felgenhauer
Department of Computer Science
TU Dresden
01069 Dresden
Germany
`bf3@mail.inf.tu-dresden.de`

Frazer Jarvis
Department of Pure Mathematics
University of Sheffield, Sheffield S3 7RH, U.K.
`a.f.jarvis@shef.ac.uk`

June 20, 2005

Introduction

Sudoku puzzles became extremely popular in Britain from late-2004. *Sudoku*, or *Su Doku*, is a Japanese word (or phrase) meaning something like *Number Place*. The idea of the puzzle is extremely simple; the solver is faced with a 9×9 grid, divided into nine 3×3 blocks:



In some of these boxes, the setter puts some of the digits 1–9; the aim of the solver is to complete the grid by filling in a digit in every box in such a way that each row, each column, and each 3×3 box contains each of the digits 1–9 exactly once.

In this note, we discuss the problem of enumerating all possible Sudoku grids. This is a very natural problem, but, perhaps surprisingly, it seems unlikely that the problem should have a simple combinatorial answer. Indeed, Sudoku grids are simply special cases of Latin squares, and the enumeration of Latin squares is itself a difficult problem, with no general combinatorial formulae known. Latin squares of sizes up to 11×11 have been enumerated, and the methods are broadly brute force calculations, much like the approach we sketch for Sudoku grids below. See [1], [2] and [3] for more details. It is known that the number of 9×9 Latin squares is $5524751496156892842531225600 \approx 5.525 \times 10^{27}$. Since this answer is enormous, we need to refine our search considerably in order to be able to get an answer in a sensible amount of computing time.

1 Initial observations

Our aim is to compute the number N_0 of valid Sudoku grids. In the discussion below, we will refer to the blocks as B1–B9, where these are labelled

	B1			B2			B3	
	B4			B5			B6	
	B7			B8			B9	

First note that we can assume, after relabelling, that the top left block (B1) is given by

1	2	3
4	5	6
7	8	9

This relabelling procedure reduces the number of grids by a factor of $9! = 362880$. We are reduced to counting the number $N_1 = \frac{N_0}{9!}$ of Sudoku grids whose top left-hand block is of this canonical form.

Broadly speaking, our strategy is to consider all possible ways to fill in blocks B2, B3, given that B1 is in the canonical form above, reducing the problem to a (much) smaller search space. This forms the outer loop of our brute force search. For the inner loop, we work out all possible ways to complete blocks B2 and B3 to a full grid.

2 Blocks B2 and B3

Here we try to catalogue efficiently the possibilities for blocks B2 and B3. We will give a lengthy discussion of how to find a comparatively small list of blocks B2 and B3 which will allow us to give the final answer. (Some of these reductions could also be applied to blocks B4 and B7 to speed up the search, although once B2/B3 is fixed, many of the reduction steps cannot be performed on B4/B7 in the same way).

2.1 Top row of blocks

We want to list all the possible configurations for the top three blocks, given that the first block is of the canonical form. Let's think about the top row of the second block. Either it consists (in some order) of the three numbers on row 2 or row 3 of block B1, or a mixture of the two. We'll call these the "pure" and "mixed" situation. The possible top rows of blocks B2 and B3 are given by:

$\{4,5,6\}$	$\{7,8,9\}$
$\{4,5,7\}$	$\{6,8,9\}$
$\{4,5,8\}$	$\{6,7,9\}$
$\{4,5,9\}$	$\{6,7,8\}$
$\{4,6,7\}$	$\{5,8,9\}$
$\{4,6,8\}$	$\{5,7,9\}$
$\{4,6,9\}$	$\{5,7,8\}$
$\{5,6,7\}$	$\{4,8,9\}$
$\{5,6,8\}$	$\{4,7,9\}$
$\{5,6,9\}$	$\{4,7,8\}$

$\{7,8,9\}$	$\{4,5,6\}$
$\{6,8,9\}$	$\{4,5,7\}$
$\{6,7,9\}$	$\{4,5,8\}$
$\{6,7,8\}$	$\{4,5,9\}$
$\{5,8,9\}$	$\{4,6,7\}$
$\{5,7,9\}$	$\{4,6,8\}$
$\{5,7,8\}$	$\{4,6,9\}$
$\{4,8,9\}$	$\{5,6,7\}$
$\{4,7,9\}$	$\{5,6,8\}$
$\{4,7,8\}$	$\{5,6,9\}$

(where $\{a, b, c\}$ indicates the numbers a, b and c *in any order*).

The pure top row $\{4,5,6\}|\{7,8,9\}$ can be completed as follows:

1	2	3	$\{4,$	$5,$	$6\}$	$\{7,$	$8,$	$9\}$
4	5	6	$\{7,$	$8,$	$9\}$	$\{1,$	$2,$	$3\}$
7	8	9	$\{1,$	$2,$	$3\}$	$\{4,$	$5,$	$6\}$

giving $(3!)^6$ possible configurations.

However, a mixed top row can be completed in more ways; for example, the top row $\{4,5,7\}|\{6,8,9\}$ can be completed as:

1	2	3	$\{4,$	$5,$	$7\}$	$\{6,$	$8,$	$9\}$
4	5	6	$\{8,$	$9,$	$a\}$	$\{7,$	$b,$	$c\}$
7	8	9	$\{6,$	$b,$	$c\}$	$\{4,$	$5,$	$a\}$

where a, b and c stand for 1, 2 and 3, in some order, giving $3 \times (3!)^6$ possible configurations (b and c are interchangeable).

We have 2 possible pure top rows (the one above and its reversal), and 18 mixed top rows (the 9 above and their reversals). In total, we therefore have

$$2 \times (3!)^6 + 18 \times 3 \times (3!)^6 = 56 \times (3!)^6 = 2612736$$

possible completions to the top three rows.

2.2 Reduction

At this stage, we have a list of all possibilities for blocks B2 and B3. We will loop over all these possibilities, and for each, fill in the remaining blocks to form valid Sudoku grids. The outer loop will run over possible blocks B2 and B3. However, to run through all 2612736 possibilities for B2 and B3 would be prohibitively time-consuming. We need some way to reduce the number of possibilities over which we have to loop. We will identify configurations of numbers in these blocks which give the same number of ways of completing to a full grid. Effectively, we define equivalence relations on the set of B2/B3 configurations in such a way that any two elements in the same class can be completed in the same number of ways.

Luckily, there are a lot of operations we can apply which leave the number of Sudoku grids invariant. We have already seen the relabelling operation. But there are others; for example, if we exchange B2 and B3, then every way of completing B2-B3 to a complete grid gives us a unique way to complete B3-B2 to a complete grid (just exchange B5 and B6, and B8 and B9). Indeed, we can permute B1, B2 and B3 in any way we choose (although this changes B1, we can then relabel to put B1 back into canonical form).

Furthermore, we can permute the columns within any block in any way we wish, performing the same operation to the columns in a completed grid. We see that there are many operations on B1-B3 which give other possible top blocks which complete to full grids in the same number of ways.

Lexicographical reduction

Take all of the 2612736 possibilities mentioned above. We catalogue them first as follows:

1. We begin by permuting the columns within B2 and B3 so that the first entries are in increasing order.
2. We then exchange B2 and B3 if necessary, so that B2 would come before B3 in a dictionary ("lexicographic order").

The first step gives 6 ways to permute the columns in each block, so that, given any grid, there are $6^2 = 36$ grids derived from it with the same number of ways of completing; then the third essentially doubles this number. Overall, we are reducing the number of possibilities we need to consider by a factor of 72, giving 36288 possibilities for our catalogue. This reduces our search to 36288, which is becoming practicable, although more reductions are desirable.

Nevertheless, it gives us a good start with our reduction, reducing our catalogue to a size which allows a finer control over the individual entries.

Refined permutation and relabelling

In fact, we haven't really made full use of all of the permutation and relabelling possibilities. The idea is to go through all permutations of the 36288 possibilities of the *three* blocks B1-B3, and all permutations of columns within these three blocks, making $6^4 = 1296$ possibilities for each block. We then look at our new first block, and relabel it so that it is in canonical form, relabelling B2 and B3 similarly, and then using the lexicographical reduction on the result. This provides a huge improvement again, reducing the size of the catalogue to just 2051 possible B2/B3 pairs. (The huge majority of these 2051 pairs arise from exactly $18 = 6^4/72$ of the 36288 possibilities. Some, however, arise from fewer, so it is necessary to store exactly how many of the 36288 possibilities give rise to the given blocks.)

But this is not all – we can do the same for the 6 permutations of the three *rows* of the configuration. That is, we can choose any permutation of these rows, and then relabel to put B1 back into canonical form. This gives a further reduction to testing just 416 possibilities for blocks B2 and B3.

Duplication

While this improvement is extremely useful, the more reductions we can make, the faster the program will finish. Indeed, there are still more possibilities for improving our outer loop. Here is a possible arrangement for the top three rows in a Sudoku grid:

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

Consider the positions of the numbers 1 and 4:

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

Let's relabel to interchange these two numbers:

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

It is easy to see that any grid completing these three blocks also completes the same three blocks with the 1s and 4s reversed in B1/B2:

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

Consequently, the number of ways to extend the original three rows is the same as the number of ways of extending these three rows, and so we should only compute this once. Note that the same can be done for the pair of numbers 5 and 8 in columns 2 and 9, and also for the pair 6 and 9 in columns 3 and 6. We require that two numbers in one column of B1 also occur in the same positions of a column of B2/B3 (but in the opposite order, of course). Then we may interchange the other two occurrences of these numbers. The following arrangement permits no fewer than six pairs of numbers:

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

An extension of this method allows us to identify any two configurations with a subrectangle of size $2 \times k$ (respectively, $k \times 2$) whose entries consist of two columns (respectively, rows) with the same numbers.

Using this trick with just the 2×2 subrectangles reduced the 416 equivalence classes to 174. Using 2×3, 3×2 and 4×2 rectangles as well reduces this list to just 71 classes.

This completes our discussion of blocks B2 and B3.

2.3 Summary of results

A C++-program was written that generates all 36288 lexicographically reduced configurations, and then builds an equivalence relation based on the equivalences pointed out above, that is, it determines the reflexive, symmetric, transitive hull of the relation generated by those equivalences. The program then generates a list of a representatives and the size of each equivalence class. In fact, the representative chosen is the lexicographically smallest member of the corresponding equivalence class.

In the original calculation, not all of these equivalences were implemented; we reduced the outer loop to run over 306 classes. Subsequently, the programs were run again with the set of 71 representatives, and the same answer was returned.

3 The inner loop

3.1 Left column of blocks: B4 and B7

Exactly the same analysis of the left columns gives the number of possible completions of the left three columns as $56 \times (3!)^6 = 2612736$ (again assuming the top left block to be in canonical form). Again, we may permute the rows of B4, or permute the rows of B7, or exchange B4 and B7, reducing by a factor of 72 to 36288, using our lexicographical reduction method. Some of the above reduction methods could be used to reduce the size of the B4/B7 catalogue further.

However, at this stage, we have already reduced the size of the B2/B3 catalogue sufficiently that a complete optimisation of the B4/B7 catalogue is not required. Indeed, the first author, who wrote the programs, decided that it was simpler to run the loop over just the 720 possible first columns of B4 and B7 (all permutations of the remaining numbers $\{2, 3, 5, 6, 8, 9\}$ in the first column). Again, by re-ordering the rows of B4 and B7, and exchanging B4 and B7 if necessary, we are reduced to just 10 possibilities for these first columns, without storing the remainder of the data. As already remarked, the predicted running time at this point was sufficiently low – a few hours on a single PC – that the possible speed-up gained by looping over a catalogue of possibilities for all of blocks B4 and B7, reduced by some of the methods listed above, was hardly worth implementing.

3.2 The loop

At this stage, the top three blocks B1–B3, and also the first column of blocks B4 and B7 are filled. A backtracking algorithm was programmed by the first author, running over the posibilities for entering numbers in the following order:

X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X	X
X	6	12	18	19	20	21	22	23
X	5	11	17	28	29	30	31	32
X	4	10	16	27	36	37	38	39
X	3	9	15	26	35	42	43	44
X	2	8	14	25	34	41	46	47
X	1	7	13	24	33	40	45	48

This order is based on the observation that every Sudoku grid is also a Latin square. To keep the branching factor low, it is a good idea to start with an entry from the shortest remaining column or row.

This proved to be an extremely efficient method, exhausting the possibilities for a given configuration of blocks B2 and B3 to just under 2 minutes on a single PC.

The result

There are $N_0 = 6670903752021072936960 \approx 6.671 \times 10^{21}$ valid Sudoku grids. Taking out the factors of 9! and 72^2 coming from relabelling and the lexicographical reduction of the top row of blocks B2 and B3, and of the left column of blocks B4 and B7, this leaves $3546146300288 = 2^7 \times 27704267971$ arrangements, the last factor being prime.

Verification

Computer calculations are always met with some suspicion, so an effort was made to increase the confidence in our results. As there are two steps in the calculation, it makes sense to try to verify them independently.

For the outer loop, the program that generates the equivalence classes was modified such that it can also output a forest where each tree corresponds to an equivalence class. The tree is rooted at the representative of the equivalence class and each edge connects two nodes of the same equivalence class and is labelled with a reason why the two connected nodes are equivalent. In practice the reason denotes one of a few rules to apply to one of the nodes to get to the other. The rules that are used are swapping rows, columns, boxes, or relabelling $2 \times k$ or $k \times 2$ subrectangles as explained above, each followed by relabelling the first block and then lexicographically reducing the result. Each tree completely covers one equivalence class. This output was then verified by another program that was written from scratch for this purpose. We are, therefore, very confident that this reduction to 71 equivalence classes is indeed correct.

The inner loop is harder to verify, but, having made the method public, the authors were sent an independently developed search program, written by Ed Russell, that uses a completely different approach for the enumeration – it places all 1 digits at once, then all 2 digits, and so on – which the first author modified to work with initial placements of the first three blocks to profit from the outer loop reduction. This program turned out to be faster than our program by a factor of 3, and, most importantly, also returned the same results. We would like to thank Ed Russell for sending us his program and for interesting correspondence.

In fact, having made an exhaustive search of ways to complete our 71 representatives, we found that there were 44 distinct answers. This suggested that there were further equivalences which we had not implemented, and that our 71 equivalence classes can really be reduced to 44 distinct classes. Indeed, by storing the columns of B2/B3 suitably, Ed Russell's program also permitted the discovery of these equivalences, and reduced the B2/B3 catalogue to just 44 equivalence classes, as we had anticipated.

Finally, the difference between our exact value and an approximate value predicted using a very nice (and simple) heuristic argument by Kevin Kilfoil (see `http://www.sudoku.com/forums/viewtopic?t=44`), is just 0.2%.

The programs and data are stored at `http://www.inf.tu-dresden.de/~bf3/sudoku/` and also at `http://www.shef.ac.uk/~pm1afj/sudoku/`.

References

[1] S.F.Bammel, J.Rothstein, *The number of* 9×9 *Latin squares*, Discrete Mathematics 11 (1975) 93–95

[2] B.D.McKay, E.Rogoyski, *Latin squares of order 10*, Electronic J. Combin. 2 (1995), Note 3, approx 4pp. (electronic)

[3] B.D.McKay, I.M.Wanless, *The number of Latin squares of order eleven*, Ann. Combin., to appear