# Classification results

The data and programs in this repository, together with the CSP-Rules-V2.1 software ( https://github.com/denis-berthier/CSP-Rules-V2.1) were the basis for the classification results below.

# 1) A controlled-bias generator of minimal puzles

The above classification results are relative to the output of classical top-down or bottom-up generators of minimal puzzles. There are now serious reasons to suspect that these generators are biased with respect to the number of clues of these puzzles: the relative proportions of puzzles in the collection with different numbers of clues do not reflect reality. (Red Ed, on the Sudoku Player's Forum, was the first to suggest that the classical generators had a bias.)
As, from the above tables, there appears to be a small upward trend of (SER or W) complexity with respect to the number of clues, the above results may also have some bias with respect to these variables.

As the correlation coefficient between the SER or W rating and the number of clues is very small (0.1), the potential bias with respect to SER or W is likely to be small also. Nevertheless, one may want statistics based on collections of minimal puzzles that are unbiased, perhaps not in the absolute but at least with respect to these variables.

Unfortunately, no generator of minimal puzzles is currently guaranteed to have no such bias and building such a generator with reasonable computation times seems out of reach.

I therefore devised another way of proceeding: taking the generators as they are and applying corrections for the bias, if we can estimate it.
The method is similar to what becomes now applied in cameras: instead of complex optimisations of the lenses to reduce typical anomalies (such as chromatic aberration, purple fringing, barrel or pincushion distortion, and so on) — optimisations that lead to large and expensive lenses —, some camera makers now accept a small amount of these in the lenses and they correct the result in real time with dedicated software before recording the photo.
The main question here is: can we determine the bias of these current generators? Unfortuntely, the answer is negative for the classical top-down or bottom-up generators.

But there appears to be a medium way between "improving the lens" and "correcting its small defects by software": I devised *a conceptually simple modification of the top-down generators such that it allows a precise mathematical computation of the bias and a simple correction procedure*.

***Acknowlegments:***
Thanks to Eleven for implementing the first modification of top-down suexg-x.x compliant with the specification of controlled-bias defined below and then several faster versions of it. This allowed to turn the whole idea into reality.
Thanks to Paul Isaacson for adapting Brian Turner's fast solver so that it could be used instead of that of suexg-cb, thus making it still faster.
Thanks to Glenn Fowler (gsf) for providing an a priori unbiased source of complete grids: the full (compressed) collection of their equivalence classes together with a fast decompressor.
Thanks also to Allan Barker, Coloin, David P. Bird, Mike Metcalf for discussions and/or various contributions.
This informal collaboration via the Sudoku Player's Forum was very productive: due to several independent optimisations, the last version of suexg-cb (cb-opt, which doesn't retain much of the original suexg code) is 200 times faster than the first.

## 1.1) A controlled-bias top-down generator

A standard top-down generator works as follows to produce ***one*** minimal puzzle (it has to be iterated n times to produce n minimal puzzles):

- 1) choose randomly a complete grid P
- 2a) choose one clue randomly from P and delete it, thus obtaining a puzzle P2
- 2b) if P2 has several solutions, GOTO 2a (i.e. reinsert the clue just deleted and try deleting another)
- 2c) if P2 is minimal, printout P2 and exit the whole procedure
- 2d) otherwise (the puzzle has more than one solution), set P=P2 and GOTO 2a

Clause 2b makes any analysis very difficlut. Moroever, it also causes the generator to go deeper, i.e. towards puzzles with fewer clues. It thus introduces a strong, uncontrolled bias.
Consider therefore the following, modified top-down generator of minimal puzzles, the ***controlled-bias generator***. The step described below produces ***one*** minimal puzzle (it has to be iterated n times to produce n minimal puzzles):

- 1) choose randomly a complete grid P
- 2a) choose one clue randomly from P and delete it, thus obtaining a puzzle P2
- 2b) if P2 has several solutions, GOTO 1 (i.e. restart with another complete grid)
- 2c) if P2 is minimal, printout P2 and exit the whole procedure
- 2d) otherwise (the puzzle has more than one solution), set P=P2 and GOTO 2a

The only difference is in clause 2b: if we find a multi-solution puzzle, instead of backtracking to the previous state, we merely discard the current complete grid and restart the search for a minimal puzzle with another complete grid.
Notice that, contrary to the standard top-down algorithm which produces one minimal puzzle per complete grid, the modified algorithm will generally use several complete grids before it outputs a minimal puzzle.
The efficiency question is: how many? Experimentations show that many complete grids (approximately 250,000 in the mean) are necessary before a minimal puzzle is reached. But this is a question of efficiency of the generator, not a conceptual problem.
Once this algorithm is defined, it can be implemented by a simple modification of the top-down sueg-x.x (the version of suexg used to build the sudogen0_1M collection), call it ***suexg-cb***. The modified generator is indeed much slower than the original one. The purpose here is not speed, but controlled bias and, as mentioned in the acknowlegments, drastic optimisations are possible.

The top-down controlled-bias generator has the same output as its following "virtual" counterpart. As a result, the top-down controlled-bias generator will output minimal puzzles according to the same probability as this virtual counterpart.

Repeat until a minimal puzzle has been printed

- 1) choose randomly a complete grid P
- 2) while P has at least one clue:
- 2a) choose one clue randomly from P and delete it, thus obtaining a puzzle P2
- 2b) if P2 is minimal, printout P2 (but do not exit the procedure)
- 2c) set P=P2

end while

This virtual generator does the same thing as the controlled-bias one, except that, once it has found a minimal puzzle or a multi-solution one, instead of stopping, it blindly continues along a useless path until it reaches the empty grid.
But this virtual generator is interesting theoretically because it works similarly to the random uniform search defined in section 3.2 below and according to the same transition probabilities and it outputs minimal puzzles according to the probability Pr on the set B of minimal puzzles defined below.

Let us now build our formal model of this generator.

# 1.2) A forest of paths from complete grids to puzzles

Let us introduce the notion of a *doubly indexed puzzle*. We consider only (single or multi solution) consistent puzzles P. The double index of a doubly indexed puzzle P has a clear intuitive meaning: the first index is one of its solution grids and the second index is a sequence (notice: not a set, but a sequence, i.e. an ordered set) of clue deletions leading from this solution to P. In a sense, the double index keeps track of the generation process.
Given a doubly indexed puzzle Q, there is an underlying singly-indexed puzzle: the ordinary puzzle obtained by forgetting the second index of Q, i.e. by remembering the solution grid from which it came and by forgetting from which sequence of deletions Q was reached from this solution.
Given a doubly indexed puzzle Q, there is also a non indexed puzzle, obtained by forgetting the two indices.
Notice that, for a single solution doubly indexed puzzle, the first index is useless as it can be computed from the puzzle; in this case singly indexed and non indexed are equivalent. (In terms of the generator, it could as well output minimal puzzles or couples minimal-puzzle-plus-solution.)

Consider now the following layered structure (a forest of trees with branches pointing downwards), the nodes being (single or multi solution) doubly indexed puzzles:
- floor 81 : the N different complete solution grids (considered as puzzles), each indexed by itself and by the empty sequence; notice that all the puzzles at floor 81 have 81 clues;
- floor 80: each doubly indexed puzzle Q at floor 81 sprouts 81 branches pointing to floor 80, one for each clue C in Q; the other end of this C branch will be the doubly indexed puzzle obtained from Q by removing clue C and indexed by the same complete grid as Q and by the 1-element sequence (C); notice that all the puzzles at floor 80 have 80 clues;
- recursive step: given floor n+1 (each doubly indexed puzzle of which has n+1 clues and is indexed by a complete grid that solves it and by a sequence of length 81-(n+1)), build floor n as follows:
each doubly indexed puzzle Q at floor n+1 sprouts n+1 branches; for each clue C in Q, there is a branch leading to a doubly indexed puzzle R at floor n: R is obtained from Q by removing clue C; its first index is identical to that of Q and its second index is the (81-n)-element sequence obtained by appending C to the end of the second index of Q; notice that all the doubly indexed puzzles at floor n have n clues and the length of their second index is equal to 1 + (81-(n+1)) = 81-n.

It is easy to see that, at floor n, each doubly indexed puzzle has an underlying singly indexed puzzle identical to that of (81 - n)! doubly indexed puzzles with the same first index at the same floor (including itself).
This is equivalent to saying that, at any floor n < 81, any singly-indexed puzzle Q can be reached by exactly (81 - n)! different paths from the top (all of which start necessarily from the complete grid defined as the first index of Q). These paths are the (81 - n)! different ways of deleting one by one its missing 81-n clues from its solution grid.
Notice that this would not be true for non indexed puzzles that have multiple solutions. This is where the first index is useful.

Let N be the number of complete grids. At each floor n, there are:
N * 81! / n! doubly indexed puzzles,
N * 81! / (81-n)! / n! singly indexed puzzles.
For each n, there is therefore a uniform probability P(n) = 1/N * 1/81! * (81-n)! * n! that a singly indexed puzzle Q at floor n is reached by a random (uniform) search starting from the associated complete grid (its first index) at the top.

What is important here is the ratio: **P(n+1) / P(n) = (n + 1) / (81 - n**).
This formula is valid globally if we start from all the complete grids, as above, but it is also valid for all the single solution puzzles if we start from a single complete grid (just forget N in the proof above). (Notice however that it is not valid if we start with a subgrid instead of a complete grid.)

Now, call B the set of (non indexed) minimal puzzles. On B, all the puzzles are minimal. Any puzzle strictly above B has redundant clues and a single solution. Notice that, for all the puzzles on B and above B, singly indexed and non indexed puzzles are in one-to-one correspondence.
On the set B of minimal puzzles there is a probabily Pr naturally induced by the different Pn's (and renormalised to sum up to 1) and it is the probability that a minimal puzzle Q is output by our controlled-bias generator. It depends only on the

number of clues and it is defined, up to a multiplicative constant k, by Pr(Q) = k P(n), if Q has n clues. k must be chosen so that the probabilities of all the minimal puzzles sum up to 1.
But we need not know k. What is important here is that, by construction of Pr on B (a construction which models the workings of the controlled bias generator), the fundamental relation **Pr(n+1) / Pr(n) = (n + 1) / (81 - n**) holds for any two minimal puzzles, with respectively n+1 and n clues.

For n < 41, this relation means that a minimal puzzle with n clues is more likely to be reached from the top than a minimal puzzle with n+1 clues. More precisely, we have:
Pr(40) = Pr(41),
Pr(39) = 42/40 * Pr(40),
Pr(38) = 43/39 * Pr(39).

Repeated application of the formula gives Pr(24) = 61.11 Pr(30) : a puzzle with 24 clues has ~ 61 more chances of being output than a puzzle with 30 clues. This is indeed a strong bias.

A non-biased generator would give the same probability to all the minimal puzzles.
The above relation shows that **the controlled bias generator:**
**- is unbiased when restricted (by filtering its output) to n-clue puzzles, for any fixed n,**
**- is biased towards puzzles with fewer clues,**
**- this bias is well known.**

Moreover, **the puzzles produced by the controlled-bias generator are uncorrelated, provided that the complete grids are chosen in an uncorrelated way**.

**As we know precisely the bias with respect to uniformity, we can correct it easily by applying correction factors cf(n) to the probabilities on B**. Only the relative values of the cf(n) is important: they satisfy **cf(n+1) / cf(n) = (81 - n) / (n + 1)**. Mathematically, after normalisation, cf is just the relative density of the uniform distribution on B with respect to the probability distribution Pr.

[Notice that a classical top-down generator is still more biased in favour of puzzles with fewer clues because, instead of discarding the current path when it meets a multi-solution puzzle, it backtracks to the previous floor and tries again to go deeper.]

# 1.3) Computing unbiased means and standard deviations of a variable X using a controlled-bias generator

In practice, how can one compute statistics of minimal puzzles based on a (large) sample produced by a controlled-bias generator, using an uncorrelated source of complete grids?
If we consider any random variable X defined (at least) on minimal puzzles, let:
- on(n) be the observed number of puzzles with n clues in the sample,
- E(X, n) be the observed mean value of X for puzzles with n clues in the same sample,
- sd(X, n) be the observed standard deviation of X for puzzles with n clues in the same sample.

The raw (biased) mean of X is classically estimated as: sum[E(X, n) * on(n)] / sum[on(n)] (theorem on the additivity of the mean values).
The real, unbiased mean of X must be estimated as (this is a mere weighted average):

**real-mean(X) = sum[E(X, n) * on(n) * cf(n)] / sum[on(n) * cf(n)]**.

Similarly, the raw (biased) standard deviation of X is classically estimated as: sqrt{sum[sd(X, n)^2 * on(n)] / sum[on(n)]} (theorem on the additivity of the variances - beware, not the standard deviations!).
And the real, unbiased standard deviation of X must be estimated as (this is merely the standard deviation for a weighted average):

**real-sd(X) = sqrt{sum[sd(X, n)^2 * on(n) * cf(n)] / sum[on(n) * cf(n)]}**.

These formul¾ show that the cf(n) sequence needs be defined only modulo a multiplicative factor.
It is convenient to choose cf(26) =1. This gives the following sequence of correction factors (in the range 19-31, which includes all the puzzles of all the samples we have obtained with all the random generators considered here):

cf-sequence[19...31] = [0.00134 0.00415 0.0120 0.0329 0.0843 0.204 0.464 1 2.037 3.929 7.180 12.445 20.474]

It may be shocking to consider that a 30-clue puzzle in a sample must be given a weight 61 times greater than a 24-clue puzzle, but that's how it is.
A consequence of all this is that unbiased statistics for the mean number of clues of minimal puzzles must rely on extremely large samples with sufficiently many 29-clue and 30-clue puzzles. Practical computations below show that the interval of interest is [22, 30].

## 1.4) Very small sensitivity of the controlled-bias generator to the source of complete grids

The source of (sufficiently random) complete grids has a very limited impact on the output of the controlled-bias generator. This nice property has been tested with different sources of complete grids. See section 4.2.
It is easily understandable: as two thirds (in the mean) of a complete grid are deleted in the deletion phase, any structure that might have existed in the complete grid is washed away by this deletion phase.

## 1.5) A remark on bottom-up generators

A similar analysis for bottom-up generators is more difficult, perhaps even unfeasible, because these generators are not purely bottom-up. Starting with 0 clues, they add clues until they reach a single solution puzzle, but after that they delete clues until they reach a minimal puzzle. Contrary to top-down generators, there doesn't seem to be an easy way of modifying them to get some form of controlled bias.
Moreover, as bottom-up generators are more biased than top-down, it doesn't seem useful to start from them if one wants to build a modified algorithm with controlled bias: the correction factors would have to be very large.

# 2) Unbiased classification of minimal puzles

The controlled-bias method works in practice, although minimal puzzle generation was very slow before various optimisations of the generator were made. The algorithm can be accelerated by deleting the first 46 (or even 48) clues without doing any intermediate test, because the probability of obtaining an n-clue minimal puzzle with n > 35 (or even n > 33) is very small (as shown by the first 180,000 puzzles generated without this optimisation, in which the maximum number of clues was 31). The resulting accelerated algorithm is 6 times faster (20 times if we consider only the deletion part). After generating more than 6.5 millions of minimals, only two 32s and no minimal puzzle with more than 32 clues has been found.

I first used the method on a collection of 500,000 puzzles generated by the controlled-bias generator suexg-cb.
(The first 180,000 puzzles were generated before the algorithm was accelerated; the distributions before and after the above 46-clue acceleration have been checked to be the same).
Apart from the distribution of clues, which is more sensitive to fluctuations, the various unbiased averages obtained are very stable.

After the above computations were done, it was suggested that the source of complete grids I had used (the internal suexg generator) might be a source of bias (this will be discussed in section 3.2).
Alternative sources of complete grids were then tested.
A final solution to this problem was reached when Glenn Fowler (gsf) made available, in a compressed form of reasonable size, a full list of all the (equivalence classes of) complete grids and a fast decompressor. Not only did this anihilated any

doubts about the source of complete grids, it also allowed an important speed improvement. This is of course an uncorrelated source of complete grids (more precisely: essentially uncorrelated, i.e. uncorrelated modulo isomorphims).
Technically, Unix piping is used to input an integer number of full scans of gsf's collection into the (deletor part of the) suexg-cb controlled-bias generator.
The results reported in this section are therefore guaranteed to be unbiased.

Sub-section 2.2 gives the results when the suexg internal generator of complete grids is used instead of gsf's collection. It shows that the controlled-bias generator is relatively insensitive to the source of complete grids: even a strong bias in this source (suexg is known to generate complete grids with ~ 20% more minimals than an ubiased source) has almost no effect on the statistics of minimals.

# 2.1) Final results using an *a priori* unbiased uncorrelated source of complete grids (with the final version of the controlled-bias generator)

The reults below were obtained with a sample of 5,926,343 uncorrelated minimal puzzles, corresponding to 279 full scans of gsf's collection.
Remark: there is some (very small: 0.03% in the mean) discrepancy in the number of tries, whether one counts them directly in the algorithm (as I did when I needed this datum, i.e. only for the estimated number of minimals, section 3.1.4) or one multiplies the number of gsf scans with the number of gsf grids. This discrepancy is so small that it would have been harmless even if it had not been random.
But, for some time, I have been wondering about the cause. I have found.
UNIX piping is far from perfect: there is some (very small) random data leak. In the present case, a very small percentage of the complete gsf grids are lost before reaching the deletor part of suexg-cb. This has no impact on the following results, as one can consider this data leak as a (real, not pseudo) random sampling of the input complete grids, with high probability of acceptance.
The leak seems to be different on different machines. It is very low on my Mac (0.01%) and a little higher on other Unix machines I've been able to use. The positive aspect of this difference is, I could check that the results don't depend on the leak.

## 2.1.1) Global results

At the risk of some redundancy with the results in the "Comparisons" file, it is interesting to compare the global results for the (now three) main kinds of generators with the real (estimated) values.

| generator | bottom-up generator (suexg-bu) | top-down generator (suexg-td) | controlled-bias generator (suexg-cb) | real (estimated) value (suexg-cb with correction factors) |
|---|---|---|---|---|
| mean(#clues) | 23.87 | 24.38 | 25.667 | 26.577 |
| standard-deviation(#clues) | 1.08 | 1.12 | 1.116 | 1.116 |
| skewness(#clues) | 0.11 | 0.08 | 0.087 | ~ 0 |
| kurtosis(#clues) | 0.026 | 0.007 | 0.024 | ~ 0 |
| mean(W) | 1.80 | 1.94 | 2.217 | 2.449 |
| standard-deviation(W) | 1.24 | 1.29 | 1.35 | 1.39 |
| mean(SER) | 3.50 | 3.77 | 4.29 | 4.73 |
| standard-deviation(SER) | 2.33 | 2.42 | 2.54 | 2.49 |
| correlation coeff (W, SER) | 0.898 | 0.895 | 0.897 | 0.90 |
| correlation coeff (#clues, W) | 0.096 | 0.115 | 0.192 | 0.19 |
| correlation coeff (#clues, SER) | 0.11 | 0.120 | 0.199 | 0.20 |

| | | | | |
|---|---|---|---|---|
| max(W) (*)<br>(size of sub-sample used) | 11<br>(10,000) | 13<br>(1,000,000) | 16<br>(5,926,343) | 16 |
| max(SER) (*)<br>(size of sub-sample used) | 9.2<br>(1,000,000) | 9.3<br>(1,000,000) | 9.3<br>3,037,717 | 9.3 |

(*) comparison of maximum values for samples of different sizes is not meaningful

It can be seen that, in all these cases, the number-of-clues skewness and kurtosis are close to 0, which means that the general shape of each of these distributions is close to that of a Normal (but, of course, they are not continuous).

## 2.1.2) The real number of clues of minimal puzzles

controlled-bias mean = 25.667 controlled-bias standard-deviation = 1.116

**real (estimated) mean = 26.577** real (estimated) standard-deviation = 1.116

The real, unbiased value for the mean number of clues of minimal puzzles is 0.9 more than the raw mean number given by the controlled-bias generator.

The controlled-bias and real (estimated) number-of-clues distributions are given by the following table:

| #clues | #instances<br>in cb sample | %<br>in cb sample | real %<br>(estimated) | real %<br>standard-deviation |
|---|---|---|---|---|
| 19 | 0 | 0.0 | 0.0 | |
| 20 | 2 | 0.000037 | 1.32e-07 | 0.93e-07 |
| 21 | 164 | 0.00277 | 3.14e-05 | 0.25e-05 |
| 22 | 6,651 | 0.112 | 3.48e-03 | 0.43e-03 |
| 23 | 110,103 | 1.858 | 0.148 | 0.00045 |
| 24 | 704,089 | 11.88 | 2.285 | 0.0027 |
| 25 | 1,814,413 | 30.62 | 13.425 | 0.010 |
| 26 | 2,002,349 | 33.79 | 31.909 | 0.023 |
| 27 | 1,007,700 | 17.00 | 32.712 | 0.033 |
| 28 | 247,259 | 4.172 | 15.480 | 0.031 |
| 29 | 31,449 | 0.531 | 3.598 | 0.020 |
| 30 | 2,088 | 0.0352 | 0.414 | 0.009 |
| 31 | 74 | 1.25e-03 | 0.0241 | 0.0028 |
| 32 | 2 | 3.37e-05 | 1.02e-03 | 0.7e-03 |
| total | 5,926,343 | 100 | 100 | |

The vast majority of minimal puzzles produced by the controlled-bias algorithm is still in the range [23 - 28] clues, but the real distribution one can deduce from it is notably different from its raw distribution. For $n < 26$, it has fewer occurrences, for $n \geq 26$ it has more occurrences.

## 2.1.3) The real W rating (mean, standard deviation, skewness and kurtosis) as a function of the number of clues

Remember that, for any fixed number of clues, the controlled-bias generator is completely unbiased. As a result, each row of the the table below gives both the controlled-bias and the real values for the n-clue W rating. Only the global mean value, standard deviation, skewness and kurtosis have to be computed differently.

| #clues | #instances in cb sample | real mean(W) | real standard-deviation(W) | real skewness(W) | real kurtosis(W) |
|---|---|---|---|---|---|
| 19 | 0 | | | | |
| 20 | 2 | 1.95 (*) | 1.05 (*) | -1.94e-22 (*) | -2.0 (*) |
| 21 | 164 | 1.52 | 0.93 | 3.7e-05 | 0.60 |
| 22 | 6,651 | 1.64 | 1.12 | 0.0016 | 1.46 |
| 23 | 110,103 | 1.73 | 1.18 | 0.025 | 1.16 |
| 24 | 704,089 | 1.86 | 1.25 | 0.13 | 0.41 |
| 25 | 1,814,413 | 2.04 | 1.32 | 0.26 | -0.16 |
| 26 | 2,002,349 | 2.27 | 1.38 | 0.20 | -0.53 |
| 27 | 1,007,700 | 2.55 | 1.42 | 0.055 | -0.67 |
| 28 | 247,259 | 2.85 | 1.42 | 0.0045 | -0.52 |
| 29 | 31,449 | 3.16 | 1.37 | -0.00038 | -0.11 |
| 30 | 2088 | 3.47 | 1.29 | -6.8e-05 | 0.51 |
| 31 | 74 | 3.57 | 1.23 | 3.5e-06 | 0.99 |
| 32 | 2 | 4.5 (*) | 0.5 (*) | 0.0 (*) | 0.99 (*) |
| total | 5,926,343 | 2.449 | 1.39 | | |

(*) values based on small samples are not meaningful
(**) although the standard deviation depends on the number of clues, this dependency is small and we have used the formula in section 3.3.

Which gives:
controlled-bias mean(W) = 2.217 controlled-bias standard-deviation(W) = 1.35
**real mean(W) = 2.449 real standard-deviation(SER) = 1.39**
These values are identical to those obtained in case the source of complete grids was the suexg internal generator.

**Conclusion:**
**there seems to be a (non absolute) barrier of complexity such that, when the number of clues (n) increases:**
**- the n-clue mean complexity increases;**
**- the proportion of puzzles away from the n-clue mean increases;**
**but**
**- the proportion of puzzles far below the n-clue mean increases;**
**- the proportion of puzzles far above the n-clue mean decreases.**
**Graphically, the n-clue distribution looks like a wave; when n increases, the wave moves to the right, with a longer left tail and a steeper right front.**

## 2.1.4) The real SER rating (mean, standard deviation, skewness and kurtosis) as a function of the number of clues

Here again, we use the fact that, for any fixed number of clues, the controlled-bias generator is completely unbiased. As a result, each row of the the table below gives both the controlled-bias and the real values for the n-clue SER. Only the global mean value, standard deviation, skewness and kurtosis have to be computed differently.
Computations for the SER were done on a sub-sample of only 1,380,962 puzzles (corresponding to 65 full scans of gsf's

collection).

| #clues | #instances in cb sample | real mean(SER) | real standard-deviation(SER) | real skewness(SER) | real kurtosis(SER) |
|---|---|---|---|---|---|
| 19 | 0 | | | | |
| 20 | 0 | | | | |
| 21 | 41 | 3.56 (*) | 2.01 (*) | | - 0.65 (*) |
| 22 | 1,526 | 3.15 | 2.16 | | - 0.24 |
| 23 | 25,884 | 3.35 | 2.24 | | - 0.72 |
| 24 | 163,694 | 3.61 | 2.36 | | - 1.18 |
| 25 | 422,451 | 3.96 | 2.47 | | - 1.53 |
| 26 | 467,047 | 4.40 | 2.54 | | - 1.73 |
| 27 | 234,963 | 4.93 | 2.53 | | - 1.68 |
| 28 | 57,615 | 5.47 | 2.44 | | - 1.31 |
| 29 | 7,243 | 6.07 | 2.19 | | - 0.32 |
| 30 | 481 | 6.76 | 1.71 | | + 2.61 |
| 31 | 16 | 5.79 (*) | 2.34 (*) | | + 2.34 (*) |
| 32 | 1 | 7.3 (*) | (*) | | (*) |
| all | 1,380,962 | **4.73** | **2.49** | | **- 0.41 (**)** |

(*) values based on small samples are not meaningful
(**) although the standard deviation depends on the number of clues, this dependency is small and we have used the formula in section 1.3.

Which gives:
controlled-bias mean(SER) = 4.29 controlled-bias standard-deviation(SER) = 2.48 controlled-bias kurtosis(SER) = - 1.70
**real mean(SER) = 4.73 real standard-deviation(SER) = 2.49**
These values are identical to those obtained in case the source of complete grids was the suexg internal generator.

The conclusions obtained for the W rating could be repeated unchanged for the SER rating.

## 2.1.5) The estimated mean number of minimal n-clue puzzles per complete grid

Finally, we can estimate the mean number of n-clue minimal puzzles per complete grid. This is not useful for our complexity computations, but it has been a longstanding open question in the Sudoku world.

| #clues | number of n-clue minimal puzzles per complete grid mean | number of n-clue minimal puzzles per complete grid error (1 standard deviation) |
|---|---|---|
| 19 | | |
| 20 | 6.152e+6 | 70.7% |
| 21 | 1.4654e+9 | 7.81% |
| 22 | 1.6208e+11 | 1.23% |
| 23 | 6.8827e+12 | 0.30% |
| 24 | 1.0637e+14 | 0.12% |
| | | |

| | | |
|---|---|---|
| 25 | 6.2495e+14 | 0.074% |
| 26 | 1.4855e+15 | 0.071% |
| 27 | 1.5228e+15 | 0.10% |
| 28 | 7.2063e+14 | 0.20% |
| 29 | 1.6751e+14 | 0.56% |
| 30 | 1.9277e+13 | 2.2% |
| 31 | 1.1240e+12 | 11.6% |
| 32 | 4.7465e+10 | 70.7% |
| all | 4.6563e+15 | 0.065% |

which (still with 0.065% relative error):
- multiplied by the number of complete grids (6,670,903,752,021,072,936,960) gives an estimated total of **3.1055e+37 minimal puzzles**
- multiplied by the number of non isomorphic grids (5,472,730,538) gives "only" an estimated total of **2.5477e+25 non equivalent minimal puzzles.**

# 2.2) Relative insensitivity of the controlled-bias generator to the source of complete grids

This sub-section gives the results when suexg internal generator of complete grids is used instead of gsf's collection. It shows that the controlled-bias generator is relatively insensitive to the source of complete grids: even a strong bias in this source (suexg is known to generate complete grids with 20% more minimals than an ubiased source) has almost no effect on the collection of cb minimals.

## 2.2.1) The number of clues of minimal puzzles

controlled-bias average = 25.65 controlled-bias standard-deviation = 1.113
**real (estimated) average = 26.56** real (estimated) standard-deviation = 1.113

The real, unbiased value for the mean number of clues of minimal puzzles is 0.9 more than the raw mean number given by the controlled-bias generator.

Above all, it is instructive to compare the mean values obtained for different generators with the estimated real value:

| generators | bottom-up generator (suexg-bu) | top-down generator (suexg-td) | controlled-bias generator (suexg-cb) | real value (suexg-cb with correction factors) |
|---|---|---|---|---|
| mean number of clues | 23.87 | 24.38 | 25.65 | 26.56 |

The following estimates are merely the product of the observed distribution and the correction factors, namely on(n) * cf(n) (normalised, of course, by sum(on(n) * cf(n)).
Figures in this section, especially for the tail of the distribution, should therefore be taken with care, due to the relatively small sample size.

| #clues | controlled-bias instances (for the 500,000 suexg-cb sample) | real instances (estimated) (normalised to 1,000,000 puzzles) ** |
|---|---|---|

| | | |
|---|---|---|
| 19 | 0 | 0.0 (*) |
| 20 | 2 | 1.6e-06 (*) |
| 21 | 4 | 9.2e-06 (*) |
| 22 | 615 | 3.9e-03 |
| 23 | 9,848 | 0.159 |
| 24 | 60,576 | 2.356 |
| 25 | 154,024 | 13.66 |
| 26 | 168,070 | 32.10 |
| 27 | 83,911 | 32.65 |
| 28 | 20,234 | 15.18 |
| 29 | 2,566 | 3.53 |
| 30 | 147 | 0.35 |
| 31 | 3 | 0.0117 (*) |
| 32 | 0 | 0 (*) |

* values based on few data are not reliable.
** the number of digits given here is obviously above the precision allowed by the sample.

### 2.2.2) Correlation coefficients

#clues vs SER = 0.20
#clues vs W = 0.19
SER vs W = 0.90

### 2.2.3) Conclusions

Form the above results, we can conclude that, when we use a controlled-bias genereator of minimal puzzles, the source of complete grids leads to very small differences in the controlled-bias and the unbiased statistics.
This can be understood on the basis of the results in section 2 relative to:
- the very weak correlation between the number of clues and the SER or W+S ratings,
- the small trend for increasing SER or W with increasing number of clues.

## 2.3) The real W distribution of minimal puzzles

Using the results obtained with the controlled-bias generator, we can now give the real (W) complexity distribution of the minimal puzzles, which may be considered the ultimate goal of this section.
It is interesting to compare it with the distributions obtained with different kinds of generators (bottom-up, top-down, controllled-bias).

| generator (sample size) | L1_0 | L1 | L2 | L3 | L4 | L5 | L6 | L7 | L8 | L9 | L10 | L11 | L12 | L13 | L14 | L15 | L16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| bottom-up (10,000) | 46.27 | 13.32 | 12.36 | 15.17 | 10.18 | 1.98 | 0.49 | 0.19 | 0.020 | 0.010 * | 0 * | 0.01 * | 0 * | 0 * | 0 * | 0 * | 0 * |
| top-down | 41.76 | 12.06 | 13.84 | 16.86 | 12.29 | 2.42 | 0.55 | 0.15 | 0.047 | 0.013 | 3.8e- | 1.5e- | 9.0e- | 2.0e- | 0 * | 0 * | 0 * |

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| (1,000,000) | | | | | | | | | | | 03 | 03 | 04 | 04 * | | | |
| controlled-bias (5,926,343) | 35.08 | 9.82 | 13.05 | 20.03 | 17.37 | 3.56 | 0.79 | 0.21 | 0.055 | 0.015 | 4.4e-03 | 1.2e-03 | 3.2e-04 | 1.0e-04 * | 6.75e-05 * | 1.7e-05 * | 1.7e-05 * |
| **real (computed from controlled-bias)** | **29.17** | **8.44** | **12.61** | **22.26** | **21.39** | **4.67** | **1.07** | **0.29** | **0.072** | **0.020** | **5.5e-03** | **1.5e-03** | **3.4e-04** | **0.7e-04 *** | **7.17e-05 *** | **6.3e-05 *** | **0.32e-05 *** |

* values based on a small sub-sample are not reliable.

These distributions show very clearly the complexity bias of the three kinds of generators.
All these distributions have the same two modes, at L1_0 and at L3, as the real distribution.
It can be seen that when one moves from bottom-up to top-down to controlled-bias to real, the distribution moves progressively to the right.
This displacement towards higher complexity occurs mainly at the first W-levels, after which it is only very slight.

**In any cases:**
**- more than 99% of the puzzles can be solved with whips of maximal length 7,**
**- more than 99.9% of the puzzles can be solved with whips of maximal length 9.**