



## Sudoku Players' Forums

[FAQ](#)
[Search](#)
[Memberlist](#)
[Usergroups](#)  
[Profile](#)
[You have no new messages](#)
[Log out \[ denis\\_berthier \]](#)

### THE REAL DISTRIBUTION OF MINIMAL PUZZLES

Goto page [Previous](#) [1](#), [2](#), [3](#) ... , [27](#), [28](#), [29](#) [Next](#)

[new topic](#)
[postreply](#)
[Sudoku Players' Forums Forum Index -> General/puzzle](#)

[View previous topic :: View next topic](#)

#### Author

#### Message

**David P Bird**

Posted: Fri Oct 02, 2009 11:35 am Post subject:

[quote](#)

Joined: 17 Sep 2008  
 Posts: 163  
 Location: Middle  
 England

A question about U4s if someone would be so kind to answer:

What is the probability of a solution grid containing N U4s? For example N = 9 as I recently found for one published puzzle.

Can we safely assume that as we randomly fill cells, the probability of a U4 is being made is independent of any previous ones that have already occurred? My instinct is that the answer to that should be no, because the options for either avoiding or hitting them must be steadily changing.

[Back to top](#)

[profile](#)
[pm](#)

**Allan Barker**

Posted: Fri Oct 02, 2009 12:21 pm Post subject:

[quote](#)

Joined: 21 Feb 2008  
 Posts: 348  
 Location: Bangkok

#### montecar.exe revision

##### Allan Barker wrote:

There is also a binary montecar.exe on the website so no need to compile.

Denis,

I have fixed a [bug](#) in the **montecar** program and posted a corrected version on my website. The problem was not in the algorithm, but in the command line program. The routine **run\_montecarlo()** returns 0 when it can't converge within a set time limit, in which case grids should be rejected. This happens about one in about  $10^5$  or  $10^6$  times depending on the set limit. When I made the command line version, I forgot to reject these grids which could print erroneous output.

This has now been fixed.

Allan

[profile](#)
[pm](#)
[www](#)

[Back to top](#)

**coloin**

Posted: Fri Oct 02, 2009 1:02 pm Post subject:

[quote](#)

Joined: 06 May 2005  
 Posts: 1079  
 Location: Devon UK

And there are no U4s in bands 1-30....[I am pretty sure]

If all bands are 1 - we can get the MC grid [Most Canonical]

This all begs the rather difficult question of the distribution and [total number] of minimal puzzles in any one grid.

The MC grid

**Code:**

```

+---+---+---+
| 123|456|789|
| 456|789|123|
| 789|123|456|
+---+---+---+
| 231|564|897|
| 564|897|231|
| 897|231|564|
+---+---+---+
| 312|645|978|
| 645|978|312|
| 978|312|645|
+---+---+---+

```

band 111,111. It has 648 automorphism. NO U4s and a lot of U6s

This has the highest average clue count with suexg of 25.7 - so the real average might be around 27.7 ?

Because of the degree of automorphism, it is just about possible to find most [? all] of the non-isomorphic puzzles at the extremes of size.

Because I could - here is the distributon up to now - still some 34s to find. I will update as and when !

**Code:**

```

19      0
20      1
21     595
22    17000
..
..
34     26000
35      289
36       11
37       0

```

Multiplying by 648 - I am not sure if there are more total puzzles in this grid than average or not.

C

[Back to top](#)

[profile](#) [pm](#)

**eleven**

☐ Posted: Fri Oct 02, 2009 2:24 pm Post subject:

[quote](#)

Joined: 10 Feb 2008  
Posts: 508

**coloin wrote:**

Multiplying by 648 - I am not sure if there are more total puzzles in this grid than average or not.

I guess much less minimals.

A quick test with 20 mio tries only gave 5 puzzles (1 26, 3 27, 1 28), i.e. 4 mio tries/puzzle!

[Back to top](#)

[profile](#) [pm](#)

**denis\_berthier**

☐ Posted: Sat Oct 03, 2009 3:47 am Post subject:

[quote](#) [edit](#)

Joined: 19 Jun 2007  
Posts: 883  
Location: Paris, France

**Allan,**

I compiled and ran the last version of montecar.c (using your compilation directives), but after running full speed (100% of CPU used) for 5 mn, there's no output. As I guess it shouldn't take so long get a grid, there must be some problem (in the output function? something specific to Windows?).

[Back to top](#)

[profile](#) [pm](#) [www](#)

**PIsaacson**

☐ Posted: Sat Oct 03, 2009 4:37 am Post subject:

[quote](#)

Joined: 02 Jul 2008  
Posts: 200  
Location: Campbell, CA

Denis,

I'm late to this party, but perhaps I can make amends by speeding up the suexg/suexg-cb programs by a decent factor. In catching up, I downloaded the generators, compiled them and starting building Glen's band sudz files. I tested against the 300-416.sudz using suexg-cb as posted by Eleven here

<http://www.sudoku.com/boards/viewtopic.php?p=81638#81638>

The time to produce a controlled bias puzzle seemed excessive, so I recompiled with profiling to see what was taking so long. The low hanging fruit appears to be the function solve, so I replaced it with the brute-force solver from Brian Turner as posted on the Sudoku Programmers' Forum. His bb\_solver is wicked fast, so I modified the suexg and suexg-cb code to use his bb Solve function.

As an example run, using the expanded 300-416.sudz which contains 2097068 puzzles according to wc:

suexg 0 1 /temp/300-416.txt -- took ~ 148 minutes.

suexg.new 0 1 /temp/300-416.txt -- took ~ 30 minutes.

The following blocks contain tests using the suexg-cb and suexg-cb.new code and demonstrate similar speed improvements. What I'm less certain of is why there's such a large difference in the number of puzzles produced by the faster solver code. I believe the bb\_solver is exactly correct in terms of solving for 0/1/multiple solutions, but obviously it's not finding the exact same sequence of solutions. How important is that???

**Code:**

```
469> time suexg-cb 0 1 /temp/300_416.txt

1...56.8.457..9.....63.....2..9.847.....6....8....395....7.....426.....3..
573943
1.3.5..8.4.....6.....3..5..2..6.....5.3..91....9.3...9..6...7.729...3..6.5.....
6004
12...6...4..18..2.....75.....8.96..8.9...7....6.2....4....6..7..38..86..15...
63893
1.....9.5..8.6....9..31...3...5..851....32.....4...27.84.....6....94..2....
175224
...5.....1.9...6.9372..52.....4.9.....827.6.7..4.87.56....3.41.3....
226174
..3..67...5718.6....9.....5.384..6...79...4.8.....57.....1..67.....4..52.8
111111
12....7...57...6.26..3.....4..5.78..6..4....15.374.....5...9..8.....5682.3
3537
..3.5.7.9..71..3..8..23.....6....4.1.2..9..9...8...3.....6...5.4.2.3.....5.
27759
.23.5.78.4.....6...8.7....4.....1.....64..583..5.....49...6.....29.....81..
74888
12.45..8...7.8.....3.1424..83.....91....28.....1..6.6...2...8.25...9.
4435
1.34.6...5.1...2...2.7.4..39.....8.....5.2.....91..4..2..39...64.7.
258217
...5...94...9.3..98..71..2.....4...6.4..91.....5..33...6..2.....94..5...3..7.
202533

real    3m35.000s
user    3m34.858s
sys     0m0.156s

470> time suexg-cb 0 10
.....7.19..65....4.7.8.2...4..6....6.5..4.1...8...378..69.....45....9...3..
360464
..5.....28945..8.4..6.11...3..9.....3..71.6.....1983...2...25..413..
47301
51.7.....3..6.79..1.....9.....4....8..2.73..6...94.2.1.316...9..5.....
97919
.6.2..1.3.....3...198.6.538...9.4..7.....2.....4.5...51.3..6...2..7..5.
822478
...571..3.2.....8.56.....94.98.....8...5.....472.....58...3.....189.47
346388
3..7...9.6...8.31..7.9.64....3.4.7....2..39.....8...3571...1..6...5..4...2
1.23774e+06
.9..6.....2...5.....291.....4..1.2...56.7...53.18..3.....68..28..3...51..34.
237063
..6.4..5...2571.....376....4.....6.....21.....478.....281..3..81..49.1.....
173369
.6...4.9.25.8.34.....6.5.2.4.....1...6.8.2...73.....7.2..8.....14.3..4...7
183650
..3...4.246...51...2...7..94..7.....4...57..216..4..6...7.....8..1.935..
125450

real    10m26.703s
user    10m26.593s
sys     0m0.030s
```

**Code:**

```

467> time suexg-cb.new 0 1 /temp/300_416.txt

12...6..94....96...6..274.....3.....89....5...5...72....612.....4....8....3..7
25   194679
..3..6.8.4....62.9..3.74....4....91..6.152.....43..6.58.....7....2.....9....
25   9624
.23....94.7..9..3..3.....78.4.....2...8165.42...8.....3....8.2..28..136
27   47016
..3.....18.6....8.27.1....8...4...45..891.3.....3..9.....1...47..6452..
25   1805
.2....89.5.1..6..9.6.....3.....5.8.1..7.4.2...513..7...9...45..3....56.1...
26   98119
...4..7...5..8....986.....7...86...3.2..7.1.6...5.1...8.975...3.1.....36
25   96607
..3.56..9..718...39.6..7.....4...9.319...2.....2..1.6.....57.....3..8.25....
26   72010
.2.4..78...71.9..3.....23..45...9..73..1.7...8.35..58.....6....9..8....4...
26   40264
..34...9..718..2..8.....3.....4.1..32.....9..2.5.6.7...8.....5.84...39..
24   76009
1...5.7.....18..3..8..7..5.9.3...1..1..5248.....9..5..9.2.....3...6...6.1..7
26   164828
...5...9..71..6.2...2..1...3.....97.7.9.2.....8.4.2..3.....95..8.1.8..6.7...
25   20417
.2.4.6...7...2...73.5.2..7...63..6...8.....95...18.....57...3...62..147.
26   20083
.2..56...4...9.3..8.....4..6.4..55317..4.8..8.....3.....8.1.....9...5.6..2.
25   133676
.2.4..78.....896.....3.....84.3..872.56..4..5.....4..32683.....5.1.....
26   162721
.....4.718.....9.23...2...7..1.....58..9.8.....3.425...8....9.3.....6.25.
24   69540
...45..89...1.....8.7.2...2..5.14.6..4....13..1.....346..5.....289.....9...3..
26   112134
.2...6.894...8....6..7..4.....139...4.....2746...8..1..7.1..3.....3..6
24   24193
.....8.4.7..9...68.7.2..42....7..8..5...9...63.5.7.3...1..9.5...3.26.....4...
26   20837
.....678.4.7.....8..3..14.758.1.6..94...1.....5.31.7...6.....3...682...5.
27   90795
...5..8.4..1.....89.32...26..9.1..3...1.9..7.5..3...9.6.78...5...63.....4
26   36332
1.3.5.....8.....14528.....1.....7.92..1.36.4....7.489...59...4.6...3
26   65932
.2.....1.....69.....4.4..9.3.6..9745..1.....9.57..63.....29..5.3.....24.
25   173967
12..56...57...63.....6...3.51...68..4..85..2...82..1.....953..73.6....
27   208440

real    1m36.359s
user    1m36.093s
sys     0m0.156s

468> time suexg-cb.new 0 10

.25..6...4.....6.3...7.1.5...8..3...1.4.7.58.....4.9.....2..852.13....6..9....
26   105650
..53.....6.1.5..3.7...8...9.6.....78.....2...3.8.1..4.4..1..1.256...8...4....
25   25409
.....8.45...9..1.2...7..67..64.12...5.....41.....7.16.395...9.876.....
26   210023
.....4.7.....329.3...83..7...6.....68...7.62...84..2.1..75.....42.7...6.5
26   1208
...4.21.....17358.....9...6...31.....2...98...53.8.6.9...7...8...8.7...32
26   565902
1....4.....56813.8.3.....7.4...5.2..1..8...5...9.....5.84...8.572...7...
25   172336
.5.184.7.2...7..5.....2.68.4...91..6...7...7.1..2.6...3...4.....2.9.2...8
26   24896
8.3..724.....6..814.5.....62...3...8...1..9.....76...462...6..7.9..5.92.....
27   209275
6..5.827...3.9.....2.....4.....3..97...26.1.3.4..9.527.....9.6.8.....
24   166981
..8.....91..26.....32...1...5.73..4.36...1.56.7.....2...85.1.....4...63.7
26   207929

```

```
real    2m4.859s
user    2m4.765s
sys     0m0.031s
```

I was going to insert Allan's monte-carlo code in suexg/suexg-cb to replace the code to internally generate a valid grid, but with the code change to use stdin and pipes, there was no need. I simply downloaded his latest montecar.c & montecar.h file and built an executable using gcc on mingw. I didn't have any problems, so perhaps you just need to download the updated code???

Anyway, the bb\_solver looks promising in terms of speeding up what appears to be extremely long runs against extremely large data bases in order to produce a set of controlled biased puzzles.

Cheers,  
Paul

[Back to top](#)



**Allan Barker**

Posted: Sat Oct 03, 2009 4:50 am Post subject:



Joined: 21 Feb 2008  
Posts: 348  
Location: Bangkok

**denis\_berthier wrote:**

**Allan,**  
I compiled and ran the last version of montecar.c (using your compilation directives), but after running full speed (100% of CPU used) for 5 mn, there's no output. As I guess it shouldn't take so long get a grid, there must be some problem (in the output function? something specific to Windows?).

You should not need to wait for more than 3ms for the first output as the program does not buffer any output. I downloaded the version on the website and recompiled. Both **montecar 0 100** and redirection with **montecar 0 100 > myfile** work fine.

Q. what command line are you using?  
Q. What OS and compiler?

Edit: Just realized, it could be that the pipe is being buffered, depending on OS and how your compiler is terminating a line (DOS 0D, 0A) or (unix 0A). This would look like a delay for a large number of grids. The fix is to use fflush(stdout), which is in the other program you are using.

Replace the print\_grid() routing with the following.

**Code:**

```
void print_grid(void)
{
    int i,j;

    for(i=0;i<9;i++)
        for(j=0;j<9;j++)
            printf("%c",SUD[i][j]+'0');
    printf("\n");
    fflush(stdout);
}
```

[Back to top](#)



**denis\_berthier**

Posted: Sat Oct 03, 2009 5:39 am Post subject:



**Allan Barker wrote:**

Q. What do you see if you run in the console without redirecting the outout?

Nothing, but the program runs.

**Allan Barker wrote:**

Q. What command line are you using for redirection?

None. This is my command line:  
./montecar.exe 0 5

**Allan Barker wrote:**

Q. what OS and compiler are you using?

Mac OSX 10.6

gcc

gcc -O3 -Wall montecar.c -o montecar.exe

I also tried without the compilation options (-O3 and -Wall), but it doesn't change anything.

**Allan Barker wrote:**

Try. Change ... In theory, these should all work.

But they don't.

The program runs for ever but outputs nothing.

I've dowloaded your last version, but now I get a compilation error:

**Code:**

```
montecar.c:30:17: error: mem.h: No such file or directory
montecar.c: In function 'mc_populate':
montecar.c:156: warning: implicit declaration of function 'memset'
montecar.c:156: warning: incompatible implicit declaration of built-in function 'memset'
montecar.c: In function 'mc_place':
montecar.c:179: warning: control reaches end of non-void function
```

**Allan Barker wrote:**

Edit: Just realized, it could be that the pipe is being buffered, depending on OS and how your compiler is terminating a line (DOS 0D, 0A) or (unix 0A). This could delay output for a large number of grids. The fix is fflush(stdout), which is in the other program you are using.

No, the problem isn't there. In suexg, printf works fine.

Last edited by denis\_berthier on Sat Oct 03, 2009 5:49 am; edited 1 time in total

[Back to top](#)

[profile](#) [pm](#) [www](#)

**denis\_berthier**

Posted: Sat Oct 03, 2009 5:48 am Post subject:

[quote](#) [edit](#)

Joined: 19 Jun 2007  
Posts: 883  
Location: Paris, France

**Pisaacson wrote:**

The low hanging fruit appears to be the function solve, so I replaced it with the brute-force solver from Brian Turner as posted on the Sudoku Programmers' Forum. His bb\_solver is wicked fast, so I modified the suexg and suexg-cb code to use his bb Solve function.

Hi Paul,

Where exactly is the code for this brute-force solver?

[Edit: Found it <http://www.setbb.com/sudoku/viewtopic.php?t=1663&mforum=sudoku>. It is said it has a lot of Windows specific code.

The main "idea" is that it computes singles. Doesn't the solve function in suexg use singles information?]

**Pisaacson wrote:**

What I'm less certain of is why there's such a large difference in the number of puzzles produced by the faster solver code. I believe the bb\_solver is exactly correct in terms of solving for 0/1/multiple solutions, but obviously it's not finding the exact same sequence of solutions. How important is that???

As the new-solve function is used only as a test and as it should always give the same 0/1/2 value as solve, it should give exactly the same puzzles with exactly the same numbers of grids used.

So - if you've been using the same seed each time - there must be a bug in one of these functions.

Given the potential improvement in speed, it's worth investigating.

[Back to top](#)

[profile](#) [pm](#) [www](#)

**Allan Barker**

Posted: Sat Oct 03, 2009 6:38 am Post subject:

[quote](#)

Joined: 21 Feb 2008  
 Posts: 348  
 Location: Bangkok

**denis\_berthier wrote:**

No, the problem isn't there. In *suexg*, *printf* works fine.

I think old and new were getting mixed up. As a last shot try the renamed *montercarlo.exe*, *.c*, and *.h* now on the website. It's got *fflush()* and pipes OK for me here. I don't have a mac, but admire them.

[Back to top](#)

 [profile](#)  [pm](#)  [www](#)

**denis\_berthier**

Posted: Sat Oct 03, 2009 7:04 am Post subject:

 [quote](#)  [edit](#)

Joined: 19 Jun 2007  
 Posts: 883  
 Location: Paris, France

For those interested in fast sudoku solvers: the free Sage 4.1.1 mathematical software (<http://www.sagemath.org/>) includes one, based on DLX and exact covers. It is said to be the fastest in the world.

The source code should be somewhere, but I'ven't found it (I didn't spend much time looking for it).

From the Ref Manual, section 6.1:

6.1 Sudoku Puzzles

This module provides algorithms to solve Sudoku puzzles, plus tools for inputting, converting and displaying various

ways of writing a puzzle or its solution(s). Primarily this is accomplished with the

`sage.games.sudoku.Sudoku`

class, though the legacy top-level `sage.games.sudoku.sudoku()` function is also available.

AUTHORS:

- Tom Boothby (2008/05/02): Exact Cover, Dancing Links algorithm
- Robert Beezer (2009/05/29): Backtracking algorithm, Sudoku class

From 4.1.1 Release notes / Miscellaneous:

An optimized Sudoku solver (Rob Beezer, Tom Boothby) — Support two algorithms for efficiently solving a Sudoku puzzle: a backtrack algorithm and the DLX algorithm. Generally, the DLX algorithm is very fast and very consistent. The backtrack algorithm is very variable in its performance, on some occasions markedly faster than DLX but usually slower by a similar factor, with the potential to be orders of magnitude slower.

...

We also compare the performance between the backtrack algorithm and the DLX algorithm.

[Back to top](#)

 [profile](#)  [pm](#)  [www](#)

**Red Ed**

Posted: Sat Oct 03, 2009 7:33 am Post subject:

 [quote](#)

Joined: 06 Jun 2005  
 Posts: 753

**PIsaacson wrote:**

What I'm less certain of is why there's such a large difference in the number of puzzles produced by the faster solver code. I believe the *bb\_solver* is exactly correct in terms of solving for 0/1/multiple solutions, but obviously it's not finding the exact same sequence of solutions. How important is that???

You shouldn't expect the exact same sequence because *suexg* uses the same RNG for puzzle creation as it does for solving (i.e. shared internal RNG state). So switching out *suexg*'s solver for another means that the puzzle creator will have different RNG state available to it (because the solver is no longer advancing the state in between times).

[Back to top](#)

 [profile](#)  [pm](#)

**denis\_berthier**

Posted: Sat Oct 03, 2009 8:47 am Post subject:

 [quote](#)  [edit](#)

**Allan Barker wrote:**

**denis\_berthier wrote:**

No, the problem isn't there. In *suexg*, *printf* works fine.

Joined: 19 Jun 2007  
 Posts: 883  
 Location: Paris, France

I think old and new were getting mixed up. As a last shot try the renamed montercarlo.exe, .c, and .h now on the website. It's got fflush() and pipes OK for me here. I don't have a mac, but admire them.

I tried again but still no output.  
I therefore tried on a distant Unix machine and it works.

I finally found the solution: in your code, there must be something incompatible with 64-bit mode. The default behaviour on Mac OS X 10.6 is 64-bit mode and the -O3 option includes the -m64. If I compile with only  
gcc -m32 -Wall montecar.c -o montecar.exe  
it works.

Piping works also. Using  
./montecar.exe 0 1000000 | ./suexg-cb-count-optim46-stream.exe 0 10 -

I got the first minimal:

```
.4..3...2.....5....896..7....3..2.8.....4.6...8.2....5..96211.....5.9....7.8.. 665090
..48.1.....6.593..3..6.....8..7.....377.9.6...1.9..1.5.3..1...9.....7.51.4 138788
41...7..6.9....53...3..9..4..4...6..1.8.75.....4.1.3....142..72....3.....2....5 324863
.....3...6..48..7.9.7.5...4.....2.3.....5..7.34...6.3.62.81.56....3.....7....5 319723
9...4..1...1..29...32.....1.8.9..7..6.....49.23.6...1...62..3...78....45.... 94452
..3.....8.1.....4.8..71.91..4....3...1..2.6...9....6.8....2.5...2..4...74..35 346644
..7...2...3.8.....86.47..41.9...9..651..1.4.....3.6....56...4...8....23... 44825
.1....7.3.....25...9.8.....7.....8.9...1....51.32.6...25.6.7.....7.....2.39... 94360
```

[Back to top](#)

 [profile](#)  [pm](#)  [www](#)

**denis\_berthier**

☐ Posted: Sat Oct 03, 2009 9:38 am Post subject:

 [quote](#)  [edit](#)

Joined: 19 Jun 2007  
Posts: 883  
Location: Paris, France

**denis\_berthier wrote:**

**Pisaacson wrote:**

What I'm less certain of is why there's such a large difference in the number of puzzles produced by the faster solver code. I believe the `bb_solver` is exactly correct in terms of solving for 0/1/multiple solutions, but obviously it's not finding the exact same sequence of solutions. How important is that???

As the new-solve function is used only as a test and as it should always give the same 0/1/2 value as solve, it should give exactly the same puzzles with exactly the same numbers of grids used.

So - if you've been using the same seed each time - there must be a bug in one of these functions.

Given the potential improvement in speed, it's worth investigating.

**Red Ed wrote:**

You shouldn't expect the exact same sequence because `suexg` uses the same RNG for puzzle creation as it does for solving (i.e. shared internal RNG state). So switching out `suexg`'s solver for another means that the puzzle creator will have different RNG state available to it (because the solver is no longer advancing the state in between times).

I had forgotten that solve() also uses the RNG.  
But, as you have another generator in your code, you could use MRW inside the solve function and the other one outside (or vice-versa)  
Then you should find exactly the same sequence of minimal and the same numbers of grids used.

[Back to top](#)

 [profile](#)  [pm](#)  [www](#)

**Pisaacson**

☐ Posted: Sat Oct 03, 2009 10:05 am Post subject:

 [quote](#)



Denis,

Joined: 02 Jul 2008

Posts: 200

Location: Campbell, CA

I downloaded the sage source distribution and found the sudoku and back-tracking source in the compressed file sage-4.1.1/spkg/standard/sage-4.1.1.spkg. Upon expanding with 7Zip, the actual source code is in sage/games/sudoku.py and sudoku\_backtrack.pyx. These are coded using python and cython, so I'm pretty skeptical of the "fastest" claim. I couldn't get the sage package to install or compile cleanly, so I can't run any actual timing tests against other solvers. If someone else can get it installed on their system, try timing against the tarek\_pearly6000 puzzles.

The bb\_solver whips through them all in 1.273 seconds, or about 212 usec/puzzle. My C++ DLX method takes 6.916 seconds total which works out to about 1153 usec/puzzle.

As for the bb\_solver code... I had to make lots of changes to his v06 code to get it to work as a direct replacement for the suexg/suexg-cb solve function, so I'd like to e-mail Brian Turner asking permission to release the modified solver code as a separate library. Plus I need to stress test it and get more profiling info.

I should have the entire first 100 bands finished in a few hours, so I've got lots of grids to test. Is there anything in particular you would like to see compared/testged using the new suexg-cb with the bb\_solver vs. the prior version?

Regarding the random number generators: I prefer the boost::lagged fibonacci 607 to the newly implemented Mersenne Twister code in the latest version of suexg-cb.c, so I may use that throughout the code to separate the solver usage from the main generator logic.

Cheers,

Paul

Last edited by PIsaacson on Sat Oct 03, 2009 10:14 am; edited 1 time in total

[Back to top](#)



Display posts from previous:



[Sudoku Players' Forums Forum Index](#) ->  
[General/puzzle](#)

All times are GMT

[Goto page](#) [Previous](#) [1](#), [2](#), [3](#) ... , [27](#), [28](#), [29](#) [Next](#)

Page 28 of 29

[Stop watching this topic](#)

Jump to:

You **can** post new topics in this forum  
You **can** reply to topics in this forum  
You **can** edit your posts in this forum  
You **can** delete your posts in this forum  
You **can** vote in polls in this forum