Sudoku Players' Forums

THE REAL DISTRIBUTION OF MINIMAL PUZZLES

Goto page Previous 1, 2, 3 ... , 29, 30, 31 Next

newtopic postreply **Sudoku Players' Forums Forum Index -> General/puzzle**

View previous topic :: View next topic

Author	Message
denis_berthier Joined: 19 Jun 2007 Posts: 894 Location: Paris, France	Posted: Sat Oct 03, 2009 4:01 pm Post subject: quote edit **Red Ed wrote:** The scores for my generator are:[list][*]**Ed's Generator 1** - 0 Liverpool FC I meant the 3322 tests. Back to top profile pm www
denis_berthier Joined: 19 Jun 2007 Posts: 894 Location: Paris, France	Posted: Sat Oct 03, 2009 4:10 pm Post subject: quote edit **eleven wrote:** I have tested with 10 mio puzzles, that the U4 test did not see a U4, when it was unique, but it is more effort to ensure, that it does not miss a U4. Then everything seems alright: as it is only there for optimisation, it isn't really important if it misses a few U4, is it? Back to top profile pm www
Red Ed Joined: 06 Jun 2005 Posts: 758	Posted: Sat Oct 03, 2009 4:24 pm Post subject: quote **denis_berthier wrote:** **Red Ed wrote:** The scores for my generator are:[list][*]**Ed's Generator 1** - 0 Liverpool FC I meant the 3322 tests. Then perhaps you should read the link posted previously **;-)** http://www.sudoku.com/boards/viewtopic.php?p=39771#39771 Back to top profile pm
denis_berthier Joined: 19 Jun 2007 Posts: 894 Location: Paris, France	Posted: Sat Oct 03, 2009 4:47 pm Post subject: quote edit **Red Ed wrote:** Then perhaps you should read the link posted previously **;-)** http://www.sudoku.com/boards/viewtopic.php?p=39771#39771 OK, I had missed the 1.2 for your generator. But this brings us back to an old, unanswered but central, question: **Red Ed wrote:** My testing currently works as follows. I've got a library of ~3000 single-band clue patterns

> chosen sort-of randomly. For each pattern, *p*, I know the mean, *mean(p)*, and an upper bound, *sd(p)*, on the standard deviation of the number of those patterns that exist in a perfectly randomly chosen solution grid.

How do you *know* the mean of these 3322 patterns "in a perfectly randomly chosen solution grid"? How did you choose a sample of grids "perfectly randomly"?

Back to top profile pm www

Red Ed

Posted: Sat Oct 03, 2009 6:27 pm Post subject: quote

Now answered on the relevant thread.

Joined: 06 Jun 2005
Posts: 758

Back to top profile pm

PIsaacson

Posted: Sat Oct 03, 2009 8:30 pm Post subject: quote

Denis,

Joined: 02 Jul 2008
Posts: 208
Location: Campbell, CA

I downloaded your specific version of suexg-cb and performed the following modifications:
1) I added a second MCWx + associated data elements to split the main and solve code usage per Red Ed's insight (thanks for the tip!!!).
2) In a copy of the above modified source, I replaced the solve code with a call to a library version of Brian Turner's bb_solve function Solver.

In numerous tests against various complete grids generated using gsf's bands and Allan Barker's montecar, both versions now generate exactly the same minimal controlled bias grids.

Using the 300-416 band decoded grids:

Code:

```
132> time suexg-cb 0 1 /temp/300-416.txt
123..........896........1.4.1.....7..4...2538..9.....2....6...75.47.3...6..2..3..
1916
..3....8.4...8.623....271...95.7.3.6..4.6...1..6....7.5...4....6..5..2.....6.....
131643
12.4.6.8.4..1..6...8...7...21.5.837.5....2..8...7....23...91......8.......5......
344171
.2..5...94.718...3......5...3...8157............9..368.......9.56.7.....8..23.4..
102628
..34....9.57..........27514...8..9.5.9...4.72.7.........2....6..4...5....3..6...8
15466
.23...7....71.96...8.3...1.....7...5..8..4....9..1....3.5.....1.1.9..3.....5....2
34
.2..........18....6...73....34...9.5.76....2.9.8........56...717..8...468...4.29.
224940
..34.6.8...7..96.......3.5....31.4..5......6.9.8........2.4......18.5.4.8.......1
54471
.2....7....71.96...8...3...2..64..9.36......59.5.......3.7...........2......249.3
74783
...4.67.94..1...3.68...2..42....7.........32.8.....9.5.....42......3...7.1..68...
160677
..34.6.....7.8...2..9372...2.....3.....8...1..31...426.72...56..........9..5....7
50259
....5..8.4.7...6..6.9......29.6......7....1.65..821........8..1...5.4..79.23..84.
300064
.....67..4...1.9.3.698..7..52............9...8.85...4.3..2..5.1....2....68.......4
229633
..3......45.....3...62..1..2..3...74..4.7.5..97...532....7...9.....9....7..864...
212260
....5...94..1..632.9..2.4.1.3..9.....8.6..9.....74....3......26.78...1....2..5.7.
61109
.....6...4.71...3.....2.45...8.....7....62918.....4...5..29..7..395.....71...8...
7089
.2..5.....57.891..6...1.2.4........8...1.....91....5....1..2..7..2..483...43.....
75790

real    4m14.907s
user    4m10.780s
```

```
sys     0m0.218s

132> time suexg-cb.new 0 1 /temp/300-416.txt
123.........896........1.4.1.....7..4...2538..9.....2....6...75.47.3...6..2..3..
1916
..3....8.4...8.623....271...95.7.3.6..4.6...1..6....7.5...4....6..5..2.....6.....
131643
12.4.6.8.4..1..6...8...7...21.5.837.5....2..8...7....23...91......8.......5......
344171
.2..5...94.718...3......5...3...8157............9..368.......9.56.7.....8..23.4..
102628
..34....9.57..........27514...8..9.5.9...4.72.7.........2....6..4...5....3..6...8
15466
.23...7....71.96...8.3...1.....7...5..8..4....9..1....3.5.....1.1.9..3.....5....2
34
.2..........18....6...73....34...9.5.76....2.9.8........56...717..8...468...4.29.
224940
..34.6.8...7..96.......3.5....31.4..5......6.9.8........2.4......18.5.4.8.......1
54471
.2....7....71.96...8...3...2..64..9.36......59.5.......3.7...........2......249.3
74783
...4.67.94..1...3.68...2..42....7.........32.8.....9.5.....42......3...7.1..68...
160677
..34.6.....7.8...2..9372...2.....3.....8...1..31...426.72...56..........9..5....7
50259
....5..8.4.7...6..6.9......29.6......7....1.65..821........8..1...5.4..79.23..84.
300064
.....67..4..1.9.3.698..7..52............9...8.85...4.3..2..5.1....2....68.......4
229633
..3......45.....3...62..1..2..3...74..4.7.5..97...532....7...9.....9....7..864...
212260
....5...94..1..632.9..2.4.1.3..9.....8.6..9.....74....3......26.78...1....2..5.7.
61109
.....6...4.71...3.....2.45...8.....7....62918.....4...5..29..7..395.....71...8...
7089
.2..5.....57.891..6...1.2.4........8...1.....91....5....1..2..7..2..483...43.....
75790

real    1m45.515s
user    1m44.671s
sys     0m0.171s
```

The bb_solver is about 2x faster in all cases. I compiled with the gcc 4.4.1 compiler for the latest/greatest possible optimized code generation. Using gcc 3.4.5, the bb_solver was about 3x faster in all cases, so the GCC group has made some really great improvements in the optimizer with the 4.x.x releases.

I've sent PM and e-mail to Brian Turner requesting his permission to post the modified code. He has a Copyright clause in the header, but no GPL license, so I'm in legal limbo pending his approval. Until then, if there are any other specific tests/comparisons you desire, please let me know.

Cheers,
Paul

Back to top profile pm

eleven

Posted: Sat Oct 03, 2009 8:48 pm Post subject: quote

Joined: 10 Feb 2008
Posts: 515

Nice results.

Apropos desire

PIsaacson,
would you be so kind also to try the suexg U4 version with different test grids.
And as noted, i would be interested, if a U4 test before the solver call would fasten or slow down the program with the bb_solver.

gsf,
would you be so kind to integrate a cb-random puzzle generator in your program ? From what i read, your solver should like 34 clue puzzles

Back to top profile pm

PIsaacson

Joined: 02 Jul 2008
Posts: 208
Location: Campbell, CA

Posted: Sat Oct 03, 2009 10:24 pm Post subject: quote

Eleven,

I copy/pasted what I think is your latest code from http://www.sudoku.com/boards/viewtopic.php?p=81638#81638

Code:

```
commented out lines for the UR checking
1) call to calc4unavoid()
2) test in solve for empty4unavoid()
3) changed the output to include a count of the givens
81> 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    1m44.250s
user    1m43.858s
sys     0m0.109s

82> vi suexg-cb.new.cpp - restored commented out UR checking
83> r for - recompiled
for i in *.mk; do make -f $i; done
g++ -c -g -O3 -MD -enable-auto-import -march=k8-sse3 -mtune=k8-sse3 -DWIN32 -I. -
I/usr/include/boost-1
g++ -g -O3 -MD -enable-auto-import -march=k8-sse3 -mtune=k8-sse3 -DWIN32 -I. -
I/usr/include/boost-1_33
g++ -c -g -O3 -MD -enable-auto-import -march=k8-sse3 -mtune=k8-sse3 -DWIN32 -I. -
I/usr/include/boost-1
g++ -g -O3 -MD -enable-auto-import -march=k8-sse3 -mtune=k8-sse3 -DWIN32 -I. -
I/usr/include/boost-1_33
84> r time - rerun time test
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    1m36.781s
user    1m36.515s
sys     0m0.233s
```

So, 1:44 vs 1:36 and the winner is (tada) using the unavoidable checking!!!

Next stop - seeing if the boost::lagged_fibonacci607 is faster than the included Mersenne Twister. [edit] Just tested and the results are within a fraction of a second so I guess I'd stay with the MT rather than forcing a dependency on the boost headers/library.

Cheers,
Paul

Last edited by PIsaacson on Sat Oct 03, 2009 10:56 pm; edited 1 time in total

Back to top profile pm

eleven

Posted: Sat Oct 03, 2009 10:42 pm Post subject: quote

Many thanks for this test.

Joined: 10 Feb 2008
Posts: 515

What exactly means it now ? That suexg with U4 test is faster ? If so, what about bb_solver with U4 test ?

Back to top profile pm

PIsaacson

Posted: Sat Oct 03, 2009 11:05 pm Post subject: quote

Eleven,

Joined: 02 Jul 2008
Posts: 208
Location: Campbell, CA

Sorry, I should have made it clearer that I was using the suexg-cb with the bb_solver calls and was only comparing with/without the U4 testing. So far, that's the fastest version of the suexg-cb that I have. I'm currently working on comparing the internal generator against piping in grids from Allan Barker's montecar generator, but it's kind of an apples/oranges situation other than comparing speed for producing a controlled bias puzzle.

Cheers,
Paul

Back to top profile pm

eleven Posted: Sat Oct 03, 2009 11:32 pm Post subject: quote

I see, so tributes to Coloin and Mike, the U4 idea was from.

Joined: 10 Feb 2008
Posts: 515

Back to top profile pm

denis_berthier Posted: Sun Oct 04, 2009 6:59 am Post subject: quote edit

Joined: 19 Jun 2007
Posts: 894
Location: Paris, France

CONTROLLED-BIAS GENERATORS: WHERE WE ARE

As the idea of a controlled-bias generator is becoming reality, with various implementations being developed and/or optimised, I needed to have a summary of the situation, as a necessary first step to my roadmap for future computations and analysis. It was only intended for me, because I needed a clear view of which (long) computations I should run in the near future (when my CPU is freed from gsf's generation and compression job), but perhaps it will be helpful to others as well.

First, the separation between the complete grid generation phase and the deletion phase has been accomplished, via piping. It is thus easier to deal with each phase separately and to make comparisons for the different solutions available for each of them. I'll be concerned only with publicly available solutions.

Complete grid generation

Why do we need different generators?
1) To check that the distribution of the controlled-bias minimal puzzles we obtain does not depend on the source of complete grids or, if it does (we already know that it can only be very slightly), to quantify it.
2) To select the best one(s) wrt to speed/disk space/bias/...
3) To compare the SER and NRCZT distributions of the minimals; I've shown that they don't depend only on the number of clues (in particular, for each fixed number of clues, suexg-cb and suexg have different mean NRCZT or SER ratings - see the "Rating..." thread)

Which generators do we have?
1) the suexg internal generator, the original version;
2) Allan's montecarlo generator, interesting because based on very different generation principles; moreover, I've already shown that, when combined with the original (pre controlled-bias) suexg deletor, its statistics (based on a collection of 1 million minimals for each) are very close to those of suexg; there remains to be checked that the controlled-bias statistics are close also;
3) gsf's list of all (equivalence classes of) complete grids, by far the fastest (once all the bands have been generated), a priori the least biased *if the whole set of bands is used* and not only a few bands, but requires a pre-generation of all the bands (15 days of CPU time) and 5.7 Gb of disk space; the must if speed is the main criterion and a large collection of minimals is needed.
It's a pity we haven't found a way of avoiding or sharing the 15-day generation and compression work.

Deletion phase
Why do we need different "deletors"?
1) To select the best one(s) wrt to speed - very important as good statistics require large collections of puzzles
2) I can't see any other reason

Which "deletors" do we have?
Here, I don't count as different deletors those that differ only by:
- their RNG,
- whether they output sequences of puzzles or of (puzzle, #complete grids consumed) pairs

1) suexg-cb: the original version of the suexg-top-down "deletor", modified for controlled-bias;
2) suexg-cb-optim46: suexg-cb where the deletion of the first 46 clues is done without any testing;
3) suexg-cb-optim46-U4: suexg-cb-optim46 where the solve function starts with a U4 test (an idea of Coloin, Mike and eleven);
4) suexg-cb-optim46-Turner: suexg-cb-optim46 with solve() replaced by Brian Turner's solving procedure (an idea of Paul);
5) suexg-cb-optim46-U4-Turner: suexg-cb-optim46-U4 where the solve function (i.e. Turner's) starts with a U4 test.

Current results Eleven, Paul, please correct me if I've misinterpreted your results in 3, 4, 5 below or if I've forgotten any.
First a convention: if I consider generator G with deletor D, I'll name it G|D (in reference to piping)

1) suexg|suexg-cb-optim46 is 6 times faster than suexg|suexg-cb (my estimations, on a total of 500,000 puzzles generated);
2) suexg-cb-optim46 vs suexg-cb. I don't have separate comparisons for the deletors part, but considering that the generator part in suexg-cb takes at least 50% of the time, this should make suexg-cb-optim46 at least 12 times faster than suexg-cb; it'd be interesting if anyone is able to make this analysis more precise, e.g. with gsf|suexg-cb-optim46 vs gsf|suexg-cb;
3) suexg-cb-optim46-U4 is 1.5 to 2 times faster than suexg-cb-optim46 - eleven;
4) suexg-cb-optim46-U4-Turner is 2.2 times faster than suexg-cb-optim46-U4 - Paul, based on gsf|suexg-cb-optim46-U4(-Turner) and using the grids in the gsf 300-416 band;
5) suexg-cb-optim46-Turner-U4 is 1.44/1.36 = 1.06 times faster than suexg-cb-optim46-Turner - Paul, based on gsf|suexg-cb-optim46-Turner(-U4).
[There seems to be some contradiction between 3, 4 and 5; Paul and eleven, what do you think?, am I still misinterpreting one of your posts?]

It seems that U4-Turner and U4 provide ~ equivalent gains in speed, the combination providing only a modest gain (6%).
Another (possibly noticeable) gain one can imagine is replacing optim46 with optim48 (delete the first 48 clues with no check). After all, the largest #clues I've got in 500,000 controlled-bias minimals is 31; choosing a max of 33 instead of the 35 I allowed with the optim46 idea should be harmless unless we want to generate billions of minimals. I wish I had 100 Macs to test all that I need to test.

Before I run a new series of long gsf|suexg-cb-something computations, I want to choose a fixed deletor. Starting from my current reference suexg-cb-optim46 (from which I've already generated 500,000 different minimals, consuming ~ 112 billions of complete grids), here is what I'm tempted to do (but I still have time to think of it before my CPUs are freed), in descending order of priority:
- replace optim46 by optim48 (even with my limited understanding of C, I could do it myself) and estimate the speed gain gsf|suexg-cb-optim48 vs gsf|suexg-cb-optim46;
- adopt U4 (simpler than Turner);
- adopt Turner (Paul's modified version, with no non portable inline code), when copyright problems are settled.

Comments on this policy and any participation to it are welcome.

I may update this from time to time, when new results are obtained.

Last edited by denis_berthier on Sun Oct 04, 2009 11:06 am; edited 4 times in total

Back to top profile pm www

denis_berthier

Posted: Sun Oct 04, 2009 8:16 am Post subject: quote edit

Joined: 19 Jun 2007
Posts: 894
Location: Paris, France

gsf,
I've answered your PM but it seems it takes some time to be transmitted. I've corrected the "compression" in my post.

I don't know if this post will have any merit but for me. At least it has led me to a question for you:
in this generation/compression process, which part of the time is taken by generation only?
Said otherwise: if this part is small, couldn't you also have an (almost) unbiased generator of complete grids we could use in the same manner as the existing ones (no disk space required)?

gsf wrote: what band are you up to?

I've finally used 1 to 3 of the 4 cores of my Mac in parallel but interrupted their jobs several times - so that I'll have no estimation of the generation/compression time.
I now have a total of 2.8 Gb, bands:
1 to 38
61 to 73
101 to 121
201 to 299
and your 300-416

Back to top profile pm www

PIsaacson

Posted: Sun Oct 04, 2009 8:50 am Post subject: quote

Joined: 02 Jul 2008
Posts: 208
Location: Campbell, CA

denis_berthier wrote:

> It seems that Turner and U4 provide ~ equivalent gains in speed, but the combination provides only marginal gain (6%).

Not true. The Turner bb_solver was compared with the U4 version in my posting
http://www.sudoku.com/boards/viewtopic.php?p=81825#81825.
It wasn't explicitly stated, but the suexg-cb source that I used at that time was Eleven's latest with the U4 + Mersenne Twister changes. Those results showed that replacing the suexg-cb solve function with bb_solver produced about a 2.2x reduction in run time using the 300-416 band gsf grids. So, both executables contained the U4 code and bb_solver was significantly faster.

Eleven then asked me to test with/without the U4 code which proved that the U4 code did provide a modest (but not insignificant) gain of 6%. But that was also based on the latest suexg-cb source code plus the Turner bb_solver.

The modifications I made to convert the bb_solver to a callable library contain no machine inline code or anything specific to Windows or Linux. It compiles with cygwin gcc, mingw gcc, VC++... so I think it shoud be fine on a Mac. There were some OS specific timers in the main line code, but the solver itself is now a separate file plus header(s).

If I were Brian, I'd be honored to have such an august assembly of Sudoku afficionados hack my code to pieces like a pack of hungry wolves. Oops... it's late and my mind is drifting...

Cheers,
Paul

Back to top profile pm

denis_berthier

Posted: Sun Oct 04, 2009 9:01 am Post subject: quote edit

PIsaacson wrote:

Joined: 19 Jun 2007

Joined: 19 Jun 2007
Posts: 894
Location: Paris, France

denis_berthier wrote:

> It seems that Turner and U4 provide ~ equivalent gains in speed, but the combination provides only marginal gain (6%).

Not true. The Turner bb_solver was compared with the U4 version in my posting http://www.sudoku.com/boards/viewtopic.php?p=81825#81825.
It wasn't explicitly stated, but the suexg-cb source that I used at that time was Eleven's latest with the U4 + Mersenne Twister changes. Those results showed that replacing the suexg-cb solve function with bb_solver produced about a 2.2x reduction in run time using the 300-416 band gsf grids. So, both executables contained the U4 code and bb_solver was significantly faster.

OK, thanks for these precisions. I've corrected my post. Can you check that it is correct now?

PIsaacson wrote:

> Eleven then asked me to test with/without the U4 code which proved that the U4 code did provide a modest (but not insignificant) gain of 6%. But that was also based on the latest suexg-cb source code plus the Turner bb_solver.

Well, I'm lost. Doesn't that mean that
"suexg-cb-optim46-Turner-U4 is 1.44/1.36 = 1.06 times faster than suexg-cb-optim46-Turner"
as I said in my post?

PIsaacson wrote:

> The modifications I made to convert the bb_solver to a callable library contain no machine inline code or anything specific to Windows or Linux. It compiles with cygwin gcc, mingw gcc, VC++... so I think it shoud be fine on a Mac. There were some OS specific timers in the main line code, but the solver itself is now a separate file plus header(s).

Very good! I've modified my post.
Maybe the Mersenne Twister in suexg-cb should also be kept in a separate file.

Back to top profile pm www

Display posts from previous: All Posts Oldest First Go

newtopic postreply **Sudoku Players' Forums Forum Index -> General/puzzle**

All times are GMT
Goto page Previous 1, 2, 3 ... , 29, 30, 31 Next

Page 30 of 31

Stop watching this topic

Jump to: General/puzzle Go