Sudoku Players' Forums

THE REAL DISTRIBUTION OF MINIMAL PUZZLES

Goto page Previous 1, 2, 3 ... , 38, 39, 40 Next

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

View previous topic :: View next topic

Author | **Message**

denis_berthier

Posted: Fri Oct 16, 2009 7:14 am Post subject: quote

Joined: 19 Jun 2007
Posts: 939
Location: Paris, France

> **denis_berthier wrote:**
> The problem I mentioned to gsf seems to be different: it happens even if memory is still available. I've launched new computations with the new gsf script, I have to wait until the end to see if everything is fine.

It works.

Back to top profile pm www

Red Ed

Posted: Fri Oct 16, 2009 10:19 am Post subject: quote

Joined: 06 Jun 2005
Posts: 810

On standard error of the mean-number-of-minimals-per-grid estimate in the supersets method: there's no need to use bootstrap resampling. What *was* I thinking?

For example, with the current jct[] figures for a run with SUBSZ=24 and TARGET=32, where jct[0,1,2,3,...]={114911,44,19,4,...}, we can use 'R' to get the estimates:

Code:

```
> rbind(x,ct)
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
x       0    1    2    3    4    5    6    7    9    14    15    20    25
ct 114911   44   19    4    5    1    4    1    1     1     1     1     1
> sum(x*ct)/sum(ct)*choose(81,24)/choose(32,24)
[1] 44411829518
> sqrt((sum(ct)*sum(x*x*ct)/sum(x*ct)^2-1)/(sum(ct)-1))
[1] 0.1910014
>
```

... which means we estimate there are 4.44e10 proper minimal 32s per grid, with relative standard error 19.1% (ouch) of that.

Back to top profile pm

PIsaacson

Posted: Fri Oct 16, 2009 9:58 pm Post subject: quote

Joined: 02 Jul 2008
Posts: 229
Location: Campbell, CA

> **Red Ed wrote:**
> On standard error of the mean-number-of-minimals-per-grid estimate in the supersets method: there's no need to use bootstrap resampling... we can use 'R' to get the estimates.

Is this the correct 'R' package: http://www.r-project.org/ ???

Back to top profile pm

denis_berthier

Posted: Fri Oct 16, 2009 10:16 pm Post subject: quote

Joined: 19 Jun 2007
Posts: 939
Location: Paris, France

Paul,

Yes, this is the correct R package.

How can the bb-solver be used as an independent solver:
- puzzles (file or stream) as input,

- stream of solution grids as output

Back to top profile pm www

Red Ed

Posted: Fri Oct 16, 2009 11:01 pm Post subject: quote

Joined: 06 Jun 2005
Posts: 810

> **PIsaacson wrote:**
> Is this the correct 'R' package: http://www.r-project.org/ ???

Yep. But don't bother; I'll whomp up a 'C' function for you.

Back to top profile pm

PIsaacson

Posted: Fri Oct 16, 2009 11:56 pm Post subject: quote

Joined: 02 Jul 2008
Posts: 229
Location: Campbell, CA

> **denis_berthier wrote:**
> How can the bb-solver be used as an independent solver:
> - puzzles (file or stream) as input,
> - stream of solution grids as output

You can download and compile the base source code from http://sites.google.com/site/bbsudokufiles
The zip file contains a pre-compiled binary that runs fine on XP. Just type "bb_sudouk -help" for the options.

For your specific request, I would probably just use the defaults as long as you don't mind having the solution tagged with " - solved". The input and output are stdin/stdout, so redirection and pipes are possible for both. If the output absolutely must be just the 81 char solution, then you can pipe the output to 'cut -b1-81". The final output line is a count of the puzzles input and solved, so you'll have to recompile to filter that out.

Back to top profile pm

PIsaacson

Posted: Sat Oct 17, 2009 12:04 am Post subject: quote

Joined: 02 Jul 2008
Posts: 229
Location: Campbell, CA

> **Red Ed wrote:**
> > **PIsaacson wrote:**
> > Is this the correct 'R' package: http://www.r-project.org/ ???
>
> Yep. But don't bother; I'll whomp up a 'C' function for you.

Thanks for the offer! I hate having to learn an entire package just to replicate your stats on other data sets. Then again, some of the example graphs looked very pretty!

Cheers,
Paul

Back to top profile pm

Red Ed

Posted: Sat Oct 17, 2009 12:41 am Post subject: quote

Joined: 06 Jun 2005
Posts: 810

When I offered the 'C' function, I was thinking in terms of my supersets method. Is that what you're after - mean and relative error for the supersets method - or some other stats, e.g. significance intervals for the path-based estimator?

Back to top profile pm

PIsaacson

Posted: Sat Oct 17, 2009 1:11 am Post subject: quote

Joined: 02 Jul 2008
Posts: 229
Location: Campbell, CA

> **Red Ed wrote:**
> When I offered the 'C' function, I was thinking in terms of my supersets method. Is that what you're after - mean and relative error for the supersets method - or some other stats, e.g. significance intervals for the path-based estimator?

Just for your code containing the supersets method. I've been playing with your posted code to see what I can make of it. Still pretty tough going since I'm not sure that my setups are correct, but having automatic stats produced at the end of the run(s) would be helpful. I hate loading all the raw data into Excel.

Cheers,
Paul

Back to top profile pm

Red Ed

Joined: 06 Jun 2005
Posts: 810

Posted: Sat Oct 17, 2009 2:41 am Post subject: quote

OK, done. Hit "quote" on this message to see the code.
Any more mods needed -- just let me know.

```
/* 31a_posted2.c
*/

/* Compilation:
* \mingw\bin\gcc -o 31a 31a.c mtwist-0.6\mtwist.c mtwist-0.6\randistrs.c -Wall -O9
*/

#include "mtwist-0.6/mtwist.h"
#include "mtwist-0.6/randistrs.h"

#include <stdio.h>
#include <stdlib.h>
#include <mem.h>
#include <math.h>
#include <time.h>

#if DEBUG
#define SUBSZ (22)
#define TARGET (30)
#else
#define SUBSZ (24)
#define TARGET (30)
#endif
#define NTMPL (46656)

#define SAVE_N(D,L) old_n##D[L] = n##D;

#define COMPRESS(D,L) \
{ \
int x; \
n##D = old_n##D[L]; \
for (x=0; x<n##D; x++) { \
if ( (mask[0]&active.t[D][x][0]) \
|| (mask[1]&active.t[D][x][1]) \
|| (mask[2]&active.t[D][x][2]) ) { \
uint32 tmp = active.t[D][x][0]; \
active.t[D][x][0] = active.t[D][--n##D][0]; \
active.t[D][n##D][0] = tmp; \
tmp = active.t[D][x][1]; \
active.t[D][x][1] = active.t[D][n##D][1]; \
active.t[D][n##D][1] = tmp; \
tmp = active.t[D][x][2]; \
active.t[D][x--][2] = active.t[D][n##D][2]; \
active.t[D][n##D][2] = tmp; \
} \
} \
}

#define START_LOOP(D) \
for (i##D=0; i##D<n##D; i##D++) { \
uint32 t[3]; \
t[0] = active.t[D][i##D][0]; \
t[1] = active.t[D][i##D][1]; \
t[2] = active.t[D][i##D][2]; \
if ( 0==(t[0]&mask[0]) && 0==(t[1]&mask[1]) && 0==(t[2]&mask[2]) ) { \
mask[0] ^= t[0]; \
mask[1] ^= t[1]; \
mask[2] ^= t[2];

#define END_LOOP \
mask[0] ^= t[0]; \
mask[1] ^= t[1]; \
mask[2] ^= t[2]; \
} \
}

typedef unsigned int uint32;
```

```
typedef struct {
int notlast;
uint32 t[10][NTMPL][3];
uint32 nt[10];
int idx2digit[10];
} tlist_t;

static int slist_mem = 0, slist_len[1+TARGET-SUBSZ];
static uint32 (*slist)[3] = NULL;

static tlist_t all_tmpl;
static tlist_t active;
static tlist_t base_tmpl;
static tlist_t punctured[81];

static int jct[1001] = {0};

/* ------------------------------------------------------
* find all 46656 unconstrained templates
* ------------------------------------------------------
*/

static void get_unconstrained_templates(int box) {
static int grid[9][9];
int i0, j0;
int i, j, k;

if (box == 0) {
for (i=0; i<81; i++) grid[i/9][i%9] = 0;
all_tmpl.nt[1] = 0;
}

if (box == 9) {
uint32 t[3] = {0};
if (all_tmpl.nt[1] == NTMPL) {
fprintf(stderr,"template overflow!\n");
exit(1);
}
for (i=0; i<81; i++) if (grid[i/9][i%9]) t[i/32] |= 1<<(i%32);
all_tmpl.t[1][all_tmpl.nt[1]][0] = t[0];
all_tmpl.t[1][all_tmpl.nt[1]][1] = t[1];
all_tmpl.t[1][all_tmpl.nt[1]][2] = t[2];
all_tmpl.nt[1]++;
return;
}

i0 = 3 * (box/3);
j0 = 3 * (box%3);
for (i=i0; i<i0+3; i++) {
for (j=j0; j<j0+3; j++) {
for (k=0; k<9; k++) if (grid[k][j] == 1) break; if (k<9) continue;
for (k=0; k<9; k++) if (grid[i][k] == 1) break; if (k<9) continue;
grid[i][j] = 1;
get_unconstrained_templates(box+1);
grid[i][j] = 0;
}
}

if (box) return;

if (all_tmpl.nt[1] != NTMPL) {
fprintf(stderr,"template underflow! (all_tmpl.nt[1]=%d)\n",all_tmpl.nt[1]);
exit(1);
}

for (i=2; i<=9; i++) {
for (j=0; j<NTMPL; j++) {
all_tmpl.t[i][j][0] = all_tmpl.t[1][j][0];
all_tmpl.t[i][j][1] = all_tmpl.t[1][j][1];
all_tmpl.t[i][j][2] = all_tmpl.t[1][j][2];
}
all_tmpl.nt[i] = NTMPL;
}

for (i=1; i<=9; i++) all_tmpl.idx2digit[i] = i;
all_tmpl.notlast = 0;

return;
```

```
}

/* ------------------------------------------------------
* filter templates for a given puzzle
* - if hole is specified then introduce a puncture
* ------------------------------------------------------
*/

static void filter_templates(tlist_t *src, tlist_t *dst, int (*puz)[9], int hole) {
int pass, order[10];

dst->notlast = src->notlast;
if (0<=hole && hole<81) {
if (src->notlast) {
fprintf(stderr,"PANIC: attempt to double-puncture\n");
exit(-1);
}
dst->notlast = puz[hole/9][hole%9];
if (!dst->notlast) {
fprintf(stderr,"PANIC: attempt to puncture with a blank\n");
exit(-1);
}
}

for (pass=0; pass<2; pass++) {
int dstidx;

for (dstidx=1; dstidx<=9; dstidx++) {
uint32 hit[3]={0}, miss[3]={0};
int srcidx = pass ? order[dstidx] : dstidx;
int srcdigit = src->idx2digit[srcidx];
int i;

for (i=0; i<81; i++) {
int zap = (i==hole && srcdigit==puz[i/9][i%9]);
if (!zap && puz[i/9][i%9]==srcdigit) {
hit[i/32] |= 1u << (i%32);
} else if (zap || (i!=hole && puz[i/9][i%9])) {
miss[i/32] |= 1u << (i%32);
}
}
dst->nt[dstidx] = 0;
for (i=0; i<src->nt[srcidx]; i++) {
if ( (miss[0]&src->t[srcidx][i][0])==0 && (hit[0]&~src->t[srcidx][i][0])==0 \
&& (miss[1]&src->t[srcidx][i][1])==0 && (hit[1]&~src->t[srcidx][i][1])==0 \
&& (miss[2]&src->t[srcidx][i][2])==0 && (hit[2]&~src->t[srcidx][i][2])==0 ) {
if (pass == 1) {
dst->t[dstidx][dst->nt[dstidx]][0] = src->t[srcidx][i][0];
dst->t[dstidx][dst->nt[dstidx]][1] = src->t[srcidx][i][1];
dst->t[dstidx][dst->nt[dstidx]][2] = src->t[srcidx][i][2];
}
dst->nt[dstidx]++;
}
}
}

if (pass == 0) {
for (dstidx=1; dstidx<=9; dstidx++) {
int mind=0, d;
for (d=1; d<=9; d++) if (mind==0 || dst->nt[d]<dst->nt[mind]) mind=d;
order[dstidx] = mind;
dst->nt[mind] = NTMPL+1;
}
if (src->idx2digit[order[9]] == dst->notlast) {
int tmp = order[9];
order[9] = order[8];
order[8] = tmp;
}
}
}

for (pass=1; pass<=9; pass++) dst->idx2digit[pass] = src->idx2digit[order[pass]];

return;
}

/* ------------------------------------------------------
* copy templates
* ------------------------------------------------------
```

```
*/

static void copy_templates(tlist_t *src, tlist_t *dst) {
int i, j;

dst->notlast = src->notlast;
for (i=1; i<=9; i++) {
for (j=0; j<src->nt[i]; j++) {
dst->t[i][j][0] = src->t[i][j][0];
dst->t[i][j][1] = src->t[i][j][1];
dst->t[i][j][2] = src->t[i][j][2];
}
dst->nt[i] = src->nt[i];
dst->idx2digit[i] = src->idx2digit[i];
}

return;
}

/* ------------------------------------------------------
* count
* ------------------------------------------------------
*/

static int count(tlist_t *tlp, int maxct, int (*solution)[9], int (*base)[9]) {
static uint32 mask[3], soltmpl[10][3];
int n1, n2, n3, n4, n5, n6, n7, n8;
int old_n3[2];
int old_n4[3];
int old_n5[4];
int old_n6[5];
int old_n7[6];
int old_n8[7];
int i1, i2, i3, i4, i5, i6, i7, i8;
int ct;

if (tlp != &active) copy_templates(tlp,&active);

if (maxct < 0) {
int i, j;
if (maxct!=-1 || !solution || !base) {
fprintf(stderr,"PANIC: bad parameters to count()\n");
exit(-1);
}
for (i=1; i<=9; i++) {
soltmpl[i][0] = soltmpl[i][1] = soltmpl[i][2] = 0;
for (j=0; j<81; j++) if (solution[j/9][j%9] == active.idx2digit[i]) soltmpl[i][j/32] |= 1<<(j%32);
}
}

n1 = active.nt[1];
n2 = active.nt[2];
n3 = active.nt[3];
n4 = active.nt[4];
n5 = active.nt[5];
n6 = active.nt[6];
n7 = active.nt[7];
n8 = active.nt[8];

SAVE_N(3,1)
SAVE_N(4,1)
SAVE_N(5,1)
SAVE_N(6,1)
SAVE_N(7,1)
SAVE_N(8,1)
for (i1=ct=0; i1<n1; i1++) {
mask[0] = active.t[1][i1][0];
mask[1] = active.t[1][i1][1];
mask[2] = active.t[1][i1][2];
COMPRESS(3,1)
COMPRESS(4,1) SAVE_N(4,2)
COMPRESS(5,1) SAVE_N(5,2)
COMPRESS(6,1) SAVE_N(6,2)
COMPRESS(7,1) SAVE_N(7,2)
COMPRESS(8,1) SAVE_N(8,2)
START_LOOP(2)
COMPRESS(4,2)
COMPRESS(5,2) SAVE_N(5,3)
COMPRESS(6,2) SAVE_N(6,3)
```

```
COMPRESS(7,2) SAVE_N(7,3)
COMPRESS(8,2) SAVE_N(8,3)
START_LOOP(3)
COMPRESS(5,3)
COMPRESS(6,3) SAVE_N(6,4)
COMPRESS(7,3) SAVE_N(7,4)
COMPRESS(8,3) SAVE_N(8,4)
START_LOOP(4)
COMPRESS(6,4)
COMPRESS(7,4) SAVE_N(7,5)
COMPRESS(8,4) SAVE_N(8,5)
START_LOOP(5)
COMPRESS(7,5)
COMPRESS(8,5) SAVE_N(8,6)
START_LOOP(6)
COMPRESS(8,6)
START_LOOP(7)
for (i8=0; i8<n8; i8++) {
if ( 0 == (active.t[8][i8][0]&mask[0]) \
&& 0 == (active.t[8][i8][1]&mask[1]) \
&& 0 == (active.t[8][i8][2]&mask[2]) ) {
if (maxct == -1) {
uint32 soldigits[3] = {0};
uint32 nines[3] = {~0,~0,~0};

/* this part for the solutions list
*/
#define GETSOL(D) \
soldigits[0] |= soltmpl[D][0] & active.t[D][i##D][0]; \
soldigits[1] |= soltmpl[D][1] & active.t[D][i##D][1]; \
soldigits[2] |= soltmpl[D][2] & active.t[D][i##D][2];
GETSOL(1)
GETSOL(2)
GETSOL(3)
GETSOL(4)
GETSOL(5)
GETSOL(6)
GETSOL(7)
GETSOL(8)
#define ZAPSOL9(D) \
nines[0] &= ~active.t[D][i##D][0]; \
nines[1] &= ~active.t[D][i##D][1]; \
nines[2] &= ~active.t[D][i##D][2];
ZAPSOL9(1)
ZAPSOL9(2)
ZAPSOL9(3)
ZAPSOL9(4)
ZAPSOL9(5)
ZAPSOL9(6)
ZAPSOL9(7)
ZAPSOL9(8)
soldigits[0] |= soltmpl[9][0] & nines[0];
soldigits[1] |= soltmpl[9][1] & nines[1];
soldigits[2] |= soltmpl[9][2] & nines[2];
if (slist_mem == ct) {
slist = realloc(slist,3*sizeof(uint32)*++slist_mem);
if (!slist) {
perror("slist");
exit(-1);
}
}
slist[ct][0] = soldigits[0];
slist[ct][1] = soldigits[1];
slist[ct][2] = soldigits[2];
}
if (++ct>maxct && maxct>=0) return ct;
}
}
END_LOOP /* 7 */
END_LOOP /* 6 */
END_LOOP /* 5 */
END_LOOP /* 4 */
END_LOOP /* 3 */
END_LOOP /* 2 */
} /* i2 */

return ct;
}
```

```
/* ------------------------------------------------------
* is it safe to add a single clue?
* ------------------------------------------------------
*/

static int fast_ok_to_add1(int (*base)[9], int idx, int value) {
int i, j, k;

for (i=0; i<81; i++) if (base[i/9][i%9]) {
for (j=1; j<=9; j++) {
int digit = punctured[i].idx2digit[j];
for (k=active.nt[j]=0; k<punctured[i].nt[j]; k++) {
if ( (digit==value) == ((punctured[i].t[j][k][idx/32]&(1<<(idx%32)))!=0) ) {
active.t[j][active.nt[j]][0] = punctured[i].t[j][k][0];
active.t[j][active.nt[j]][1] = punctured[i].t[j][k][1];
active.t[j][active.nt[j]][2] = punctured[i].t[j][k][2];
active.nt[j]++;
}
}
}
if (count(&active,0,NULL,NULL) == 0) return 0;
}

return 1;
}

/* ------------------------------------------------------
* is it safe to add a pair of clues?
* ------------------------------------------------------
*/

static int fast_ok_to_add2(int (*base)[9], int idx1, int idx2, int value1, int value2) {
int i, j;

base[idx1/9][idx1%9] = value1;
base[idx2/9][idx2%9] = value2;
for (i=0; i<81; i++) if (base[i/9][i%9]) {
j = base[i/9][i%9];
base[i/9][i%9] = 0;
filter_templates(&punctured[i],&active,base,999);
base[i/9][i%9] = j;
if (count(&active,0,NULL,NULL) == 0) break;
}
base[idx1/9][idx1%9] = 0;
base[idx2/9][idx2%9] = 0;

return (i == 81);
}

/* ------------------------------------------------------
* do those TARGET-clue solves
* ------------------------------------------------------
*/

static int do_jobs(int clues, int ptr, int (*safe2)[81], int (*base)[9], int (*solution)[9]) {
static int nr_tgts=0, nr_proper=0, nr_minchecks=0;
static int history[TARGET-SUBSZ];
uint32 intersection[3];
int jackpots = 0;
int i, j, k;

if (clues == SUBSZ+2) {
i = history[0]; base[i/9][i%9] = solution[i/9][i%9];
i = history[1]; base[i/9][i%9] = solution[i/9][i%9];
filter_templates(&base_tmpl,&active,base,999);
i = history[0]; base[i/9][i%9] = 0;
i = history[1]; base[i/9][i%9] = 0;

slist_len[clues-SUBSZ] = count(&active,-1,solution,base);
#if 0
printf("count(%2d,%2d) = ",history[0],history[1]);
printf("%6d\n",slist_len[clues-SUBSZ]);
#endif
intersection[0] = intersection[1] = intersection[2] = ~0;
for (i=0; i<slist_len[clues-SUBSZ]; i++) {
intersection[0] &= slist[i][0];
intersection[1] &= slist[i][1];
intersection[2] &= slist[i][2];
}
```

```
}

if (clues > SUBSZ+2) { /* now check the slist */
uint32 basemask[3]={0};
intersection[0] = intersection[1] = intersection[2] = ~0;
for (i=0; i<81; i++) if (base[i/9][i%9]) basemask[i/32] |= 1<<(i%32);
for (i=0; i<clues-SUBSZ; i++) basemask[history[i]/32] |= 1<<(history[i]%32);
#define SLOK(Z) ( \
!(basemask[0]&~slist[Z][0]) && \
!(basemask[1]&~slist[Z][1]) && \
!(basemask[2]&~slist[Z][2]) \
)
i = -1; j = slist_len[clues-SUBSZ-1]; k = 0;
do {
uint32 tmp;
while (++i<j && SLOK(i)) {
intersection[0] &= slist[i][0];
intersection[1] &= slist[i][1];
intersection[2] &= slist[i][2];
k++;
}
while (i<--j && !SLOK(j)) ;
if (i < j) {
tmp = slist[i][0]; slist[i][0] = slist[j][0]; slist[j][0] = tmp;
tmp = slist[i][1]; slist[i][1] = slist[j][1]; slist[j][1] = tmp;
tmp = slist[i][2]; slist[i][2] = slist[j][2]; slist[j][2] = tmp;
intersection[0] &= slist[i][0];
intersection[1] &= slist[i][1];
intersection[2] &= slist[i][2];
k++;
}
} while (i < j);
slist_len[clues-SUBSZ] = k;
}

if (clues == TARGET) {

/*** TARGET clues ***/

nr_tgts++;
for (i=0; i<TARGET-SUBSZ; i++) {
j = history[i];
base[j/9][j%9] = solution[j/9][j%9];
}
if (slist_len[clues-SUBSZ] == 1) {
nr_proper++;

/* finally got a TARGET-clue subgrid with one solution ... now check minimality */

for (i=0; i<81; i++) if (base[i/9][i%9]) {
base[i/9][i%9] = 0;
filter_templates(&punctured[i],&active,base,999);
base[i/9][i%9] = solution[i/9][i%9];
nr_minchecks++;
if (count(&active,0,NULL,NULL) == 0) break;
}
if (i == 81) {
printf(">>> ");
for (i=0; i<81; i++) printf("%d",base[i/9][i%9]);
printf("\n");
#if 0
/* TO DO: make this a command line option */
{
FILE *fp;
static int prev[81] = {-1};
fp = fopen("31a_out.txt","a");
for (i=j=0; i<81; i++) if (prev[i] && prev[i]==base[i/9][i%9]) j++;
if (j<SUBSZ && prev[0]!=-1) fprintf(fp,"---\n");
for (i=0; i<81; i++) fprintf(fp,"%d",prev[i]=base[i/9][i%9]); fprintf(fp,"\n");
fclose(fp);
}
#endif
jackpots = 1;
}
}
for (i=0; i<TARGET-SUBSZ; i++) {
j = history[i];
base[j/9][j%9] = 0;
}
```

```
return jackpots;

} else {

/*** fewer than TARGET clues ***/

int good[81] = {0};
for (j=ptr; j<81; j++) {
if ( clues>=SUBSZ+2 && (intersection[j/32]&(1<<(j%32))) ) continue;
for (i=0; i<clues-SUBSZ; i++) if (!safe2[history[i]][j]) break;
if (i < clues-SUBSZ) continue;
good[j] = 1;
}
for (i=80,j=0; i>=ptr; i--) if (good[i] && ++j<TARGET-clues) good[i] = 0;
for (; ptr<81; ptr++) if (good[ptr]) {
history[clues-SUBSZ] = ptr;
jackpots += do_jobs(clues+1,ptr+1,safe2,base,solution);
}

}

if (clues == SUBSZ) {
printf("Number of %2ds encountered: %d\n",TARGET,nr_tgts);
printf(" Number that were proper: %d\n",nr_proper);
if (nr_proper) printf(" Minchecks per proper: %f\n",nr_minchecks*1.0/nr_proper);
nr_tgts = nr_proper = nr_minchecks = 0;
}

return jackpots;
}

/* ------------------------------------------------------
* count the number of TARGET-clue solves required (*** SLOW ***)
* ------------------------------------------------------
*/

#if 0
static int count_jobs(int clues, int ptr, int (*safe2)[81]) {
static int history[TARGET-SUBSZ];
int count, i;

return -999;

if (clues == TARGET) return 1;

for (count=0; ptr<81; ptr++) {
for (i=0; i<clues-SUBSZ; i++) if (!safe2[history[i]][ptr]) break;
if (i < clues-SUBSZ) continue;
history[i] = ptr;
count += count_jobs(clues+1,ptr+1,safe2);
}

return count;
}
#endif

/* ------------------------------------------------------
* PROGRAM START
* ------------------------------------------------------
*/

int main(int argc, char **argv) {
static int solution[9][9], safe1[81], safe2[81][81];
mt_state state;
char line[100];
int trials = 0;
int i, j, k;

mts_goodseed(&state);

get_unconstrained_templates(0);
fprintf(stderr,"Go!\n");
srand(time(NULL));
while (1 == scanf("%99s",line)) {
int base[9][9];

jct[0]++;
for (i=0; i<81; i++) base[i/9][i%9] = solution[i/9][i%9] = line[i] - '0';
```

```
/* generate SUBSZ clue subgrid that has multiple solutions
*/
printf("\n%d-clue subgrid #%d:\n",SUBSZ,++trials);
for (i=SUBSZ; i<81; i++) {
int pos;
do {
pos = rds_iuniform(&state,0,81);
} while (base[pos/9][pos%9] == 0);
base[pos/9][pos%9] = 0;
}
#if DEBUG
{
char *s="123578694874691523965234781216749835439856172587123946341967258692385417758412369";
/* t="000078000004601500005000080200700800409006000000003046001000250602000410058012309"; */
/*4,5,11,12,14,15*/
char *b="000000000000000000000500008020070080040900600000000004600100025060200041005801030 9";
for(i=0;i<81;i++){
base[i/9][i%9]=b[i]-'0';
solution[i/9][i%9]=s[i]-'0';
}
}
#endif
for (i=0; i<81; i++) printf("%d",base[i/9][i%9]); printf("\n");
for (i=0; i<81; i++) printf("%d",solution[i/9][i%9]); printf("\n");
printf("^123456789^123456789^123456789^123456789^123456789^123456789^123456789^123456789^\n");
filter_templates(&all_tmpl,&base_tmpl,base,999);
if (count(&base_tmpl,1,NULL,NULL) == 1) {
printf("Oops - unique solution\n");
continue;
}

/* build template lists for SUBSZ-1 clue "punctured" subgrids of that,
* checking for minimality of the SUBSZ clue grid as you do so
*/
for (i=0; i<81; i++) if (base[i/9][i%9]) {
filter_templates(&all_tmpl,&punctured[i],base,i);
if (count(&punctured[i],0,NULL,NULL) == 0) {
printf("Oops - redundant clues\n");
goto MAINLOOP;
}
}

/* TO DO, maybe: reduce punctured lists to templates where bit 95 == 1 */
/* TO DO, maybe: reduce punctured lists to templates where bit 95 == 1 */
/* TO DO, maybe: reduce punctured lists to templates where bit 95 == 1 */

/* all blanks are safe singles initially
*/
for (i=0; i<81; i++) safe1[i] = (base[i/9][i%9] == 0);

/* implied clues are unsafe
*/
for (i=0; i<81; i++) if (base[i/9][i%9] == 0) {
base[i/9][i%9] = solution[i/9][i%9];
filter_templates(&all_tmpl,&punctured[i],base,i);
base[i/9][i%9] = 0;
copy_templates(&punctured[i],&active);
if (count(&active,0,NULL,NULL) == 0) { printf("Implied clue: %2d\n",i); safe1[i] = 0; }
}

/* new clues that imply existing ones are unsafe
*/
for (i=j=0; i<81; i++) if (safe1[i]) {
safe1[i] = fast_ok_to_add1(base,i,solution[i/9][i%9]);
j += safe1[i];
}
printf(" Number of safe clues: %d\n",j);
printf(" => at most %d safe pairs\n",(j*(j-1))/2);

/* find which pairs of clues are safe to add
*/
for (i=k=0; i<81; i++) for (j=i+1; j<81; j++) {
if (safe1[i] && safe1[j]) {
safe2[i][j] = fast_ok_to_add2(base,i,j,solution[i/9][i%9],solution[j/9][j%9]);
} else {
safe2[i][j] = 0;
}
k += safe2[i][j];
}
```

```
printf(" Number of safe pairs: %d\n",k);

#if 0
/* warn the viewer how many TARGET-clue supersets we'll be solving
*/
printf("Number of %ds to solve: %d\n",TARGET,count_jobs(SUBSZ,0,safe2));
#endif

/* solve!
*/
j = do_jobs(SUBSZ,0,safe2,base,solution);
if (j) {
jct[0]--;
if (j > 999) j = 1000;
jct[j]++;
}

MAINLOOP:
/* report status
*/
{
double sum_ct=0, sum_x_ct=0, sum_x_x_ct=0;
printf("\nStatus\n------\n");
for (j=0; j<1000; j++) if (jct[j]) {
printf("jct[%4d ] = %7d\n",j,jct[j]);
sum_ct += jct[j];
sum_x_ct += j*jct[j];
sum_x_x_ct += j*j*jct[j];
}
printf("jct[1000+] = %7d\n",jct[1000]);
if (jct[1000]) printf("*** WARNING: jct[1000]>0 -- following stats could be wrong ***\n");
printf("Estimated mean number of proper minimal %2ds per grid = %e\n", TARGET,
sum_x_ct/sum_ct * exp(lgamma(82)-lgamma(82-SUBSZ)-lgamma(1+TARGET)+lgamma(1+TARGET-SUBSZ))
);
printf(" Estimated relative error = ");
if (sum_ct<=1 || sum_x_ct==0) printf("unknown\n");
else printf("%5.2f%%\n",100*sqrt( (sum_ct*sum_x_x_ct/sum_x_ct/sum_x_ct-1) / (sum_ct-1) ));
}
}

return 0;
}
```

Back to top profile pm

eleven

Joined: 10 Feb 2008
Posts: 539

Posted: Sat Oct 17, 2009 6:01 am Post subject: quote

> **denis_berthier wrote:**
>
> If not, this is very unexpected: the mean SER is smaller than for the 30s (but the sd is higher)
> 30s: mean = 6.76, sd = 1.7 (based on 481 unbiased 30s)
> 32s: mean = 6.18, sd = 2.3

I am sure that the trend "higher clues - harder" also will hold for the 30+ puzzles. From 283 32's (run for 200k grids not finished) now i got a mean SER of 7.04.
I also think, that there is another general trend "more clustered - easier" (Coloin, do you have any data for that ?). In this case there are 2 11 puzzles clusters with SER 5.73.

Back to top profile pm

Red Ed

Joined: 06 Jun 2005
Posts: 810

Posted: Sat Oct 17, 2009 6:08 am Post subject: quote

eleven, what source of solution grids are you using?

btw, did anyone compute the SER for Havard's two 39s?

Back to top profile pm

denis_berthier

Joined: 19 Jun 2007
Posts: 939
Location: Paris, France

Posted: Sat Oct 17, 2009 7:16 am Post subject: quote

> **eleven wrote:**
>
> > **denis_berthier wrote:**
> >
> > If not, this is very unexpected: the mean SER is smaller than for the 30s (but the sd is higher)

> 30s: mean = 6.76, sd = 1.7 (based on 481 unbiased 30s)
> 32s: mean = 6.18, sd = 2.3
>
> I am sure that the trend "higher clues - harder" also will hold for the 30+ puzzles. From 283 32's (run for 200k grids not finished) now i got a mean SER of 7.04.

This is much more consistent with what I expected.
Can you also compute the standard deviation?

Back to top profile pm www

eleven

Posted: Sat Oct 17, 2009 8:29 am Post subject: quote

Joined: 10 Feb 2008
Posts: 539

The 39's have SER 8.3 and 8.4.

I am using about 200000 grids randomly selected from gsf's collection, but it will need some more hours (currently 170000).
The current stats i get for the jct[] values, when applying what Red Ed posted today are

Estimated mean number of proper minimal 32s per grid = 4.054757e+10
Estimated relative error = 11.28%

I will post the results tomorrow.

Back to top profile pm

Red Ed

Posted: Sat Oct 17, 2009 10:15 am Post subject: quote

Joined: 06 Jun 2005
Posts: 810

I guess "randomly selected" = decompress each in turn and select with probability 200000/5472730538 ?

Interesting that we're both running against 32s at the moment (I'm still building up that collection for Denis - I was going to wait until I'd got to 10% relative error). I'm running through about 5000 solution grids an hour, which seems much slower than you.

My estimates at this point: number=3.98e10, relative error=11.7%. That's after 240K solution grids, so it's surprising that my estimated relative error is higher than yours. So: when you post your results, could you also post your jct[] table? Thanks.

Back to top profile pm

Display posts from previous: All Posts Oldest First Go

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

All times are GMT - 8 Hours
Goto page Previous 1, 2, 3 ... , 38, 39, 40 Next

Page 39 of 40

Jump to: General/puzzle Go