Southwest State University
Matrosov Institute for System Dynamics and Control Theory of RAS
BOINC.ru community

USING GRID SYSTEMS FOR ENUMERATING COMBINATORIAL OBJECTS ON EXAMPLE OF DIAGONAL LATIN SQUARES

Eduard (evatutin) Vatutin, Oleg (Nauchnik) Zaikin, Alexey (alexone) Zhuravlev, Maxim (hoarfrost) Manzuk, Stepan (veinamond) E. Kochemazov, Sergey (SerVal) Yu. Valyaev, Vitaly S. Titov

2016

Latin squares: what is it?

$$A = \|a_{ij}\|$$

$$i,\ j = \overline{1,\ N}$$

$$N = |S|$$

$$S = \{0, 1, 2, ..., N-1\}$$

$$\forall i,\ j,\ k = \overline{1,\ N},\ j \neq k : \left(a_{ij} \neq a_{ik}\right) \wedge \left(a_{ji} \neq a_{ki}\right)$$

$$\forall i,\ j = \overline{1,\ N}, i \neq j : \left(a_{ii} \neq a_{jj}\right) \wedge \left(a_{N-i+1,\ N-i+1} \neq a_{N-j+1,\ N-j+1}\right)$$

$$\begin{pmatrix}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
1 & 2 & 9 & 4 & 3 & 6 & 7 & 5 & 0 & 8 \\
2 & 9 & 3 & 1 & 7 & 0 & 5 & 8 & 4 & 6 \\
3 & 4 & 1 & 2 & 8 & 7 & 9 & 6 & 5 & 0 \\
4 & 3 & 5 & 9 & 2 & 1 & 8 & 0 & 6 & 7 \\
5 & 6 & 4 & 8 & 1 & 2 & 0 & 9 & 7 & 3 \\
6 & 5 & 8 & 7 & 0 & 3 & 2 & 1 & 9 & 4 \\
7 & 8 & 6 & 0 & 9 & 4 & 1 & 2 & 3 & 5 \\
8 & 7 & 0 & 5 & 6 & 9 & 3 & 4 & 1 & 2 \\
9 & 0 & 7 & 6 & 5 & 8 & 4 & 3 & 2 & 1
\end{pmatrix}$$

Normalized Latin square with order 10

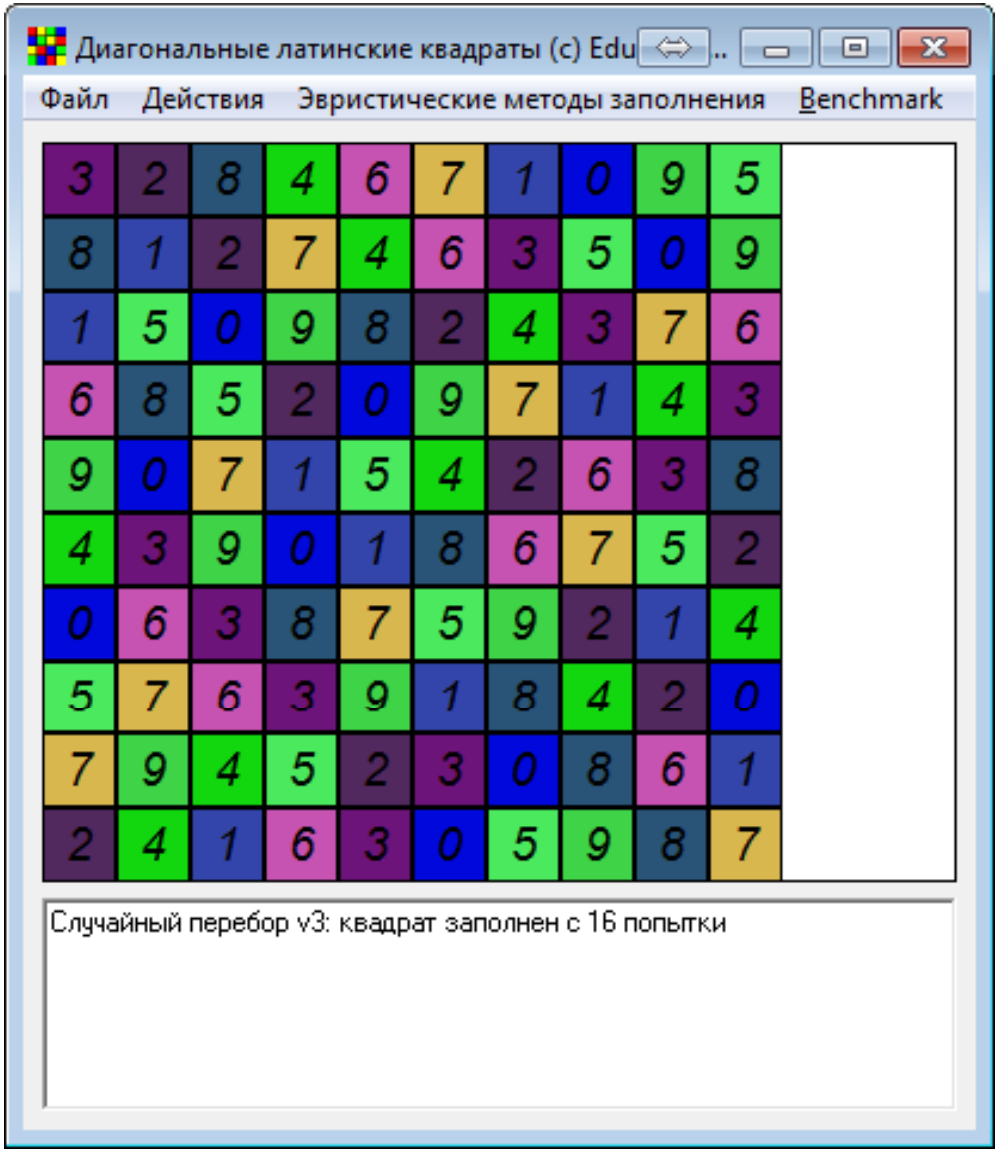
$$N! \times (N-1)!$$

$$\begin{pmatrix}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
7 & 2 & 4 & 9 & 0 & 6 & 5 & 1 & 3 & 8 \\
8 & 3 & 6 & 7 & 5 & 9 & 0 & 2 & 4 & 1 \\
2 & 6 & 8 & 5 & 1 & 7 & 4 & 0 & 9 & 3 \\
5 & 8 & 9 & 1 & 7 & 0 & 3 & 4 & 6 & 2 \\
9 & 4 & 1 & 2 & 8 & 3 & 7 & 6 & 0 & 5 \\
4 & 7 & 5 & 6 & 9 & 1 & 8 & 3 & 2 & 0 \\
3 & 0 & 7 & 8 & 2 & 4 & 1 & 9 & 5 & 6 \\
6 & 5 & 0 & 4 & 3 & 2 & 9 & 8 & 1 & 7 \\
1 & 9 & 3 & 0 & 6 & 8 & 2 & 5 & 7 & 4
\end{pmatrix}$$

First string ordered diagonal Latin square with order 10

$$(N-1)!$$

Lets try to get diagonal Latin square!



- http://evatutin.narod.ru/evatutin_LsEdit.7z

Getting Latin squares: enumerating, existence, but not optimizing combinatorial problem?

$$A = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 9 & 0 & 6 & 1 & 5 & 3 & 8 & 2 & 7 & 4 \\ 8 & 3 & 0 & 5 & 7 & 1 & 2 & 6 & 4 & - \\ 2 & 4 & 5 & 9 & 1 & 0 & 7 & 8 & 6 & 3 \\ 1 & 2 & 7 & 6 & 9 & 4 & 3 & 0 & 5 & 8 \\ 7 & 8 & 4 & 2 & 0 & 9 & 5 & 1 & 3 & 6 \\ 6 & 9 & 8 & 4 & 3 & 7 & 1 & 5 & 0 & 2 \\ 4 & 6 & 9 & 8 & 2 & - & 0 & 3 & 1 & 5 \\ 3 & 5 & 1 & 7 & 6 & 8 & 9 & 4 & 2 & 0 \\ 5 & 7 & 3 & 0 & 8 & 2 & 4 & 9 & - & 1 \end{pmatrix}$$

$$S_{3,10}^{L} = \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$$

$$S_{3,10} = S \setminus \left(S_{3,10}^{L} \cup S_{3,10}^{U} \right) = \varnothing$$

$$C(A) = 3$$

$$S_{3,10}^{U} = \{4, 9\}$$

Getting Latin squares: random search approach (RS)

$$S_{ij} = \left\{ s_{ij}^1, s_{ij}^2, ..., s_{ij}^M \right\} \subseteq S$$

$$M\left(S_{ij}\right) = \left|S_{ij}\right|$$

$$f_{RS}\left(s_{ij}^l\right) = r_k, l = \overline{1, M\left(S_{ij}\right)}$$

$$C(A) \leq 1$$

N	The number of decisions
6	68
7	4
8	1
9	0
10	0

- No decisions found for N=10 during 30 s CPU-time computing experiment (Intel Atom N270 @ 1,6 GHz, Diamondville core)

Abilities estimation

$$f_{ij}^{(x)} = \underbrace{\sum_{k=j+1}^{N} |S_{ik}|}_{\text{by string}} + \underbrace{\sum_{k=i+1}^{N} |S_{kj}|}_{\text{by column}} = \sum_{k=j+1}^{N} |S_{ik}| + (N-i)|S_{i+1,\,j}| \to \max$$

$$\alpha_{ij} = \begin{cases} 0, i \neq j \\ 1, i = j \end{cases}$$

$$g_{ij}^{(x)} = f_{ij}^{(x)} + \alpha_{ij} \underbrace{\sum_{k=i+1}^{N} |S_{kk}|}_{\substack{\text{main} \\ \text{diagonal}}} + \beta_{ij} \underbrace{\sum_{k=i+1}^{N} |S_{k,\,N-k}|}_{\substack{\text{second} \\ \text{diagonal}}} \to \max$$

$$\beta_{ij} = \begin{cases} 0, i + j \neq N \\ 1, i + j = N \end{cases}$$

a) $x=1$: $f_{64}=5+2+3+3+1+4+4x4=34$

0	1	2	3	4	5	6	7	8	9
9	5	6	0	1	4	3	2	7	8
3	4	5	8	0	2	1	9	6	7
5	7	1	9	2	8	0	6	3	4
1	8	3	5	9	6	7	4	2	0
4	9	0	1?						

$S_{74}=\{2, 4, 6, 7\}$
$S_{84}=\{2, 4, 6, 7\}$
$S_{94}=\{2, 4, 6, 7\}$
$S_{10,4}=\{2, 4, 6, 7\}$

$S_{75}=\{3, 5, 6, 7, 8\}$ $S_{76}=\{3, 7\}$ $S_{77}=\{2, 5, 8\}$ $S_{78}=\{3, 5, 8\}$ $S_{79}=\{5\}$ $S_{7,10}=\{2, 3, 5, 6\}$

b) $x=6$: $f_{64}=4+3+3+4+2+4+4x4=36$

0	1	2	3	4	5	6	7	8	9
9	5	6	0	1	4	3	2	7	8
3	4	5	8	0	2	1	9	6	7
5	7	1	9	2	8	0	6	3	4
1	8	3	5	9	6	7	4	2	0
4	9	0	6?						

$S_{74}=\{1, 2, 4, 7\}$
$S_{84}=\{1, 2, 4, 7\}$
$S_{94}=\{1, 2, 4, 7\}$
$S_{10,4}=\{1, 2, 4, 7\}$

$S_{75}=\{3, 5, 7, 8\}$ $S_{76}=\{1, 3, 7\}$ $S_{77}=\{2, 5, 8\}$ $S_{78}=\{1, 3, 5, 8\}$ $S_{79}=\{1, 5\}$ $S_{7,10}=\{1, 2, 3, 5\}$

Getting Latin squares: greedy approach (G)

$$f_G\left(s_{ij}^{l}\right) = g_{ij}^{\left(s_{ij}^{l}\right)} \rightarrow \max, l = \overline{1, M\left(S_{ij}\right)}$$

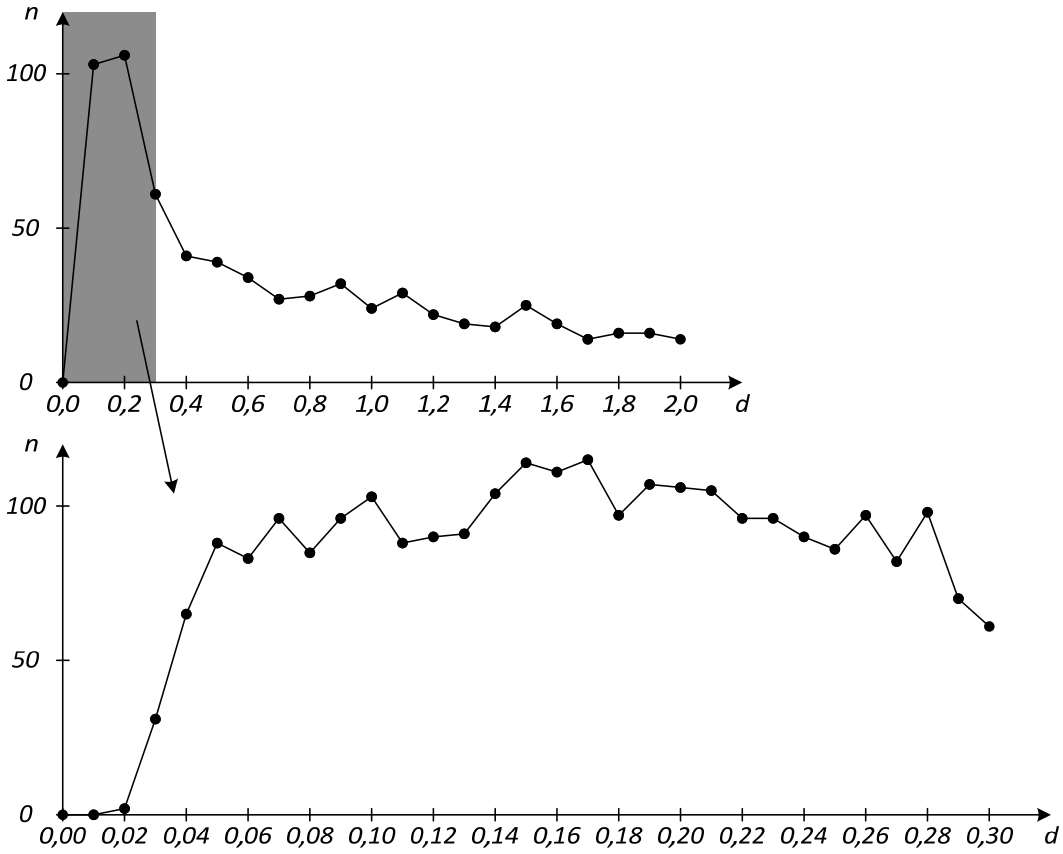
N	The number of violations
3	2
4	3
5	4
6	5
7	5
8	8
9	12
10	12

$$\begin{pmatrix}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
9 & 7 & 0 & 6 & 5 & 8 & 4 & 3 & 2 & 1 \\
8 & 0 & 6 & 7 & 9 & 4 & 5 & 1 & 3 & 2 \\
7 & 8 & 5 & 4 & 0 & 9 & 3 & 6 & 1 & – \\
6 & 5 & 8 & 0 & 3 & 7 & 9 & 4 & – & – \\
5 & 6 & 7 & 8 & – & 2 & 0 & 9 & 4 & 3 \\
4 & 3 & 9 & 5 & 6 & 0 & 8 & 2 & 7 & – \\
3 & 4 & – & 9 & 7 & 6 & 2 & 5 & 0 & 8 \\
2 & – & 4 & 1 & 8 & 3 & 7 & 0 & 9 & 6 \\
– & 2 & 3 & – & 1 & – & – & 8 & 5 & –
\end{pmatrix}$$

- No decisions found for N=3, 4, …, 10

Getting Latin squares: weighted random search approach (WRS)

$$f_{WRS}\left(s_{ij}^{l}\right)=g_{ij}^{\left(s_{ij}^{l}\right)}\cdot\left(1+2d\left(r_k-0{,}5\right)\right)\rightarrow\max,\ l=\overline{1,\ M\left(S_{ij}\right)}$$



$$C\left(A\right)\leq 1$$

N	RS	WRS
6	68	839
7	4	10
8	1	4
9	0	3
10	0	0

- No decisions found for N=10 during 30 s CPU-time computing experiment (Intel Atom N270 @ 1,6 GHz, Diamondville core)

Getting Latin squares: ant colony optimization (AC)

$$p_{ij}^{(x)} = \left[g_{ij}^{(x)}\right]^{\alpha}\left[\tau_{ij}^{(x)}\right]^{\beta} r_k \to \max$$

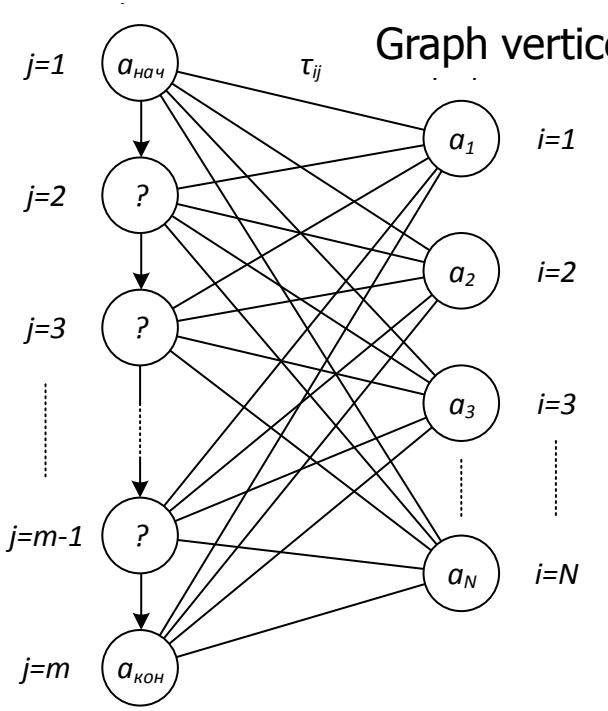
$$f_{AC}\left(s_{ij}^{l}\right) = p_{ij}^{\left(s_{ij}^{l}\right)} \to \max,\ l = \overline{1, M\left(S_{ij}\right)}$$

$$\left[\tau_{ij}^{(x)}\right]^{(t)} = \gamma\left[\tau_{ij}^{(x)}\right]^{(t-1)}$$

$$\Delta\tau = \frac{Q}{C\left(A_t\right)+1}$$

$$\alpha^{*} = 100,$$
$$\beta^{*} = 1{,}1,$$
$$\tau_0^{*} = 1{,}0,$$
$$\gamma^{*} = 0{,}99999,$$
$$Q = 1{,}0$$

Path elements



$$C\left(A\right) \le 1$$

N	RS	WRS	AC
6	68	839	9 027
7	4	10	10 050
8	1	4	64
9	0	3	20
10	0	0	6

- We get first decisions for N=10 during 30 s CPU-time computing experiment (Intel Atom N270 @ 1,6 GHz, Diamondville core)

Getting Latin squares: limited brute force (LBF)

$$L_1 = (9 \quad 7 \quad 0 \quad 6 \quad 5 \quad ? \quad ? \quad ? \quad ? \quad ?) - 132\ ms\ per\ decision$$

$$L_1 = (2 \quad 3 \quad 1 \quad 0 \quad 5 \quad ? \quad ? \quad ? \quad ? \quad ?) - 300\ ms\ per\ decision$$

AC preferences for first string:

$$(9 \quad 2 \quad 0 \quad 7 \quad 8 \quad 6 \quad 4 \quad 1 \quad 3 \quad 5),$$
$$(9 \quad 5 \quad 0 \quad 6 \quad 8 \quad 7 \quad 3 \quad 2 \quad 1 \quad 4),$$
$$(9 \quad 2 \quad 0 \quad 6 \quad 8 \quad 7 \quad 3 \quad 1 \quad 4 \quad 5),$$
$$(9 \quad 6 \quad 0 \quad 8 \quad 7 \quad 3 \quad 1 \quad 4 \quad 2 \quad 5),$$
$$(9 \quad 3 \quad 0 \quad 8 \quad 7 \quad 6 \quad 2 \quad 1 \quad 4 \quad 5).$$

Method	$C(A) \le 1$	$C(A) = 0$
RS	7 934	124
WRS	13 341	352
AC	179 387	9 422
LBF	–	346 572

- We get many decisions for N=10 during **16 hours** CPU-time computing experiment (**Intel Core i7 4770 @ 3,4 GHz, Haswell core**)
- What pace of generation? 3–7 DLS/s!
- Vatutin E.I., Zhuravlev A.D., Zaikin O.S., Titov V.S. Features of using weighting heuristics in the problem of finging diagonal latin squares (in Russian) // Proceeding of Southwest State University. Series: Control, Computer Science, Informatics. Medical Devices. 2015. № 3 (16). P. 18–30. http://evatutin.narod.ru/evatutin_co_01_ls_g_rs_wrs_ac.pdf

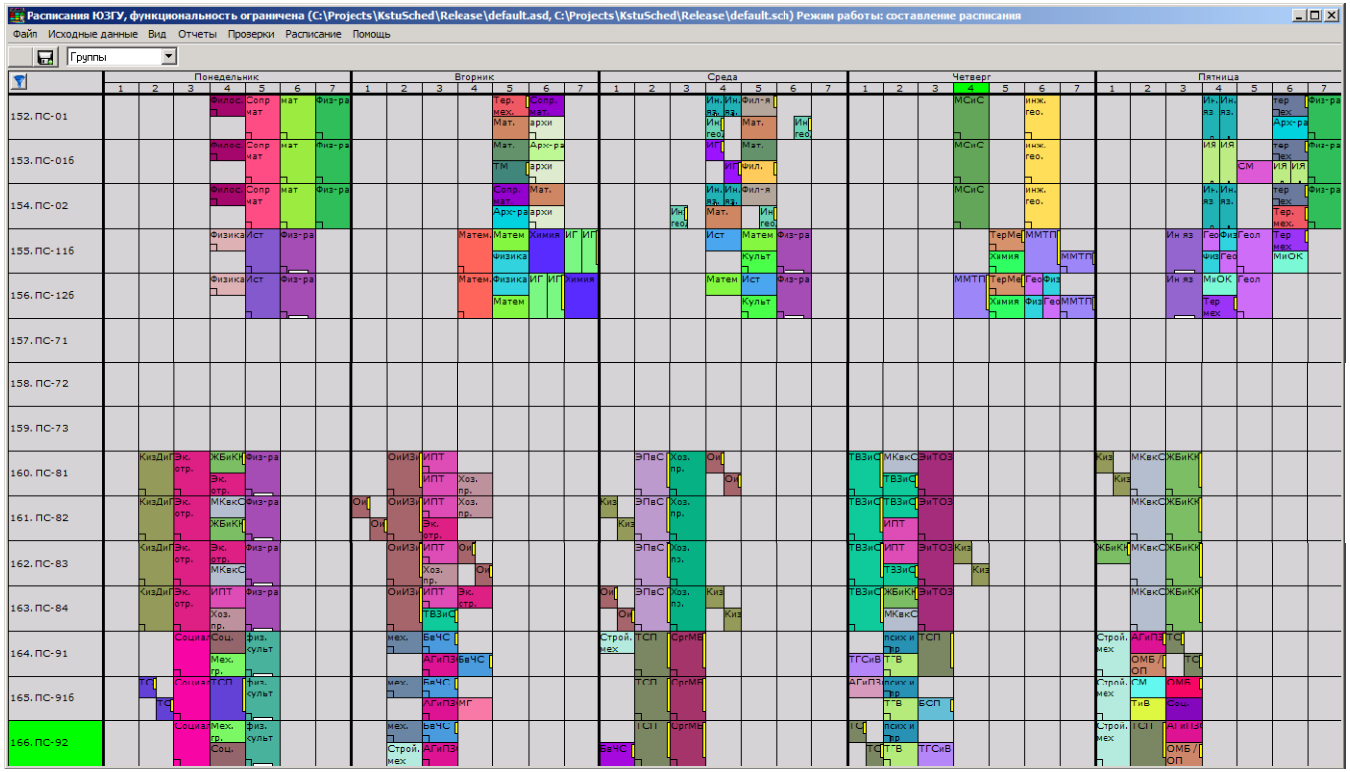
Lets try to improve pace! Diagonals first fill

1	*2*	*3*	*4*	*5*
13	*6*	*14*	*10*	*15*
16	*17*	*7*	*18*	*19*
20	*11*	*21*	*8*	*22*
12	*23*	*24*	*25*	*9*

Method	Sequential fill	Diagonals first fill	Gain
RS	≈ 0 DLS/s (0,4 DLS/s with *C*=1)	0,5 ДЛК/с (18 DLS/s with *C*=1)[1]	45x
WRS	≈ 0 DLS/s (0,7 DLS/s with *C*=1)	0,8 ДЛК/с (16 DLS/s with *C*=1)[1]	23x
AC	≈ 0 DLS/s (0,05 DLS/s with *C*=1)	0,14 DLS/s	–
LBF	≈ 0 DLS/s	28 DLS/s	–

- Pace is **28 DLS/s**!
- Only for DLS, not for LS!

Scheduling problem



- Vatutin E.I., Romanchenko A.S., Titov V.S. Investigation of influence of the pairs order consideration to schedule quality within greedy approach (in Russian) // Proceeding of Southwest State University. 2013. № 1 (46). P. 58–64. http://evatutin.narod.ru/evatutin_sched_03_greedy.pdf
- Vatutin E.I., Bobyntsev D.O., Romanchenko A.S. Investigation of influence of the partial pairs ordering and pair vicinity improvement to schedule quality within greedy approach (in Russian) // Proceeding of Southwest State University. Series: Control, Computer Science, Informatics. Medical Devices. 2014. № 1. P. 8–16. http://evatutin.narod.ru/evatutin_sched_04_gbgn.pdf

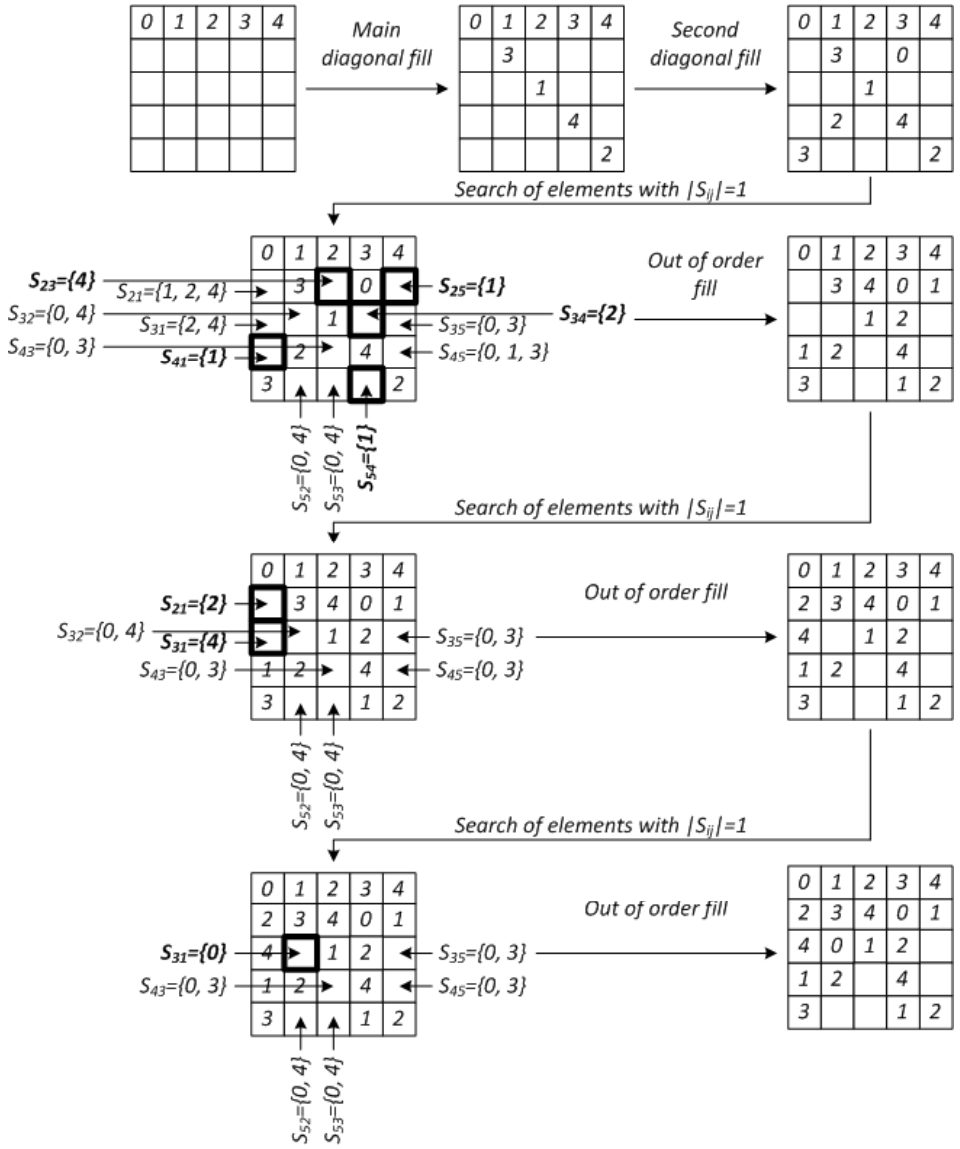
Scheduling: the same effect!

№	1	2	3	4	5	6	7	8	9	10	11	12
Random order, best 5 from 30												
1	646	16	7765	4600	1499	101	473	0	729	0	234	1,023
2	579	15	9345	4940	1503	106	427	6	725	3	265	1,071
3	653	30	8635	4880	1466	123	527	0	742	0	249	1,111
4	577	23	9105	5185	1542	90	557	8	707	1	234	1,150
5	625	25	8695	5600	1560	97	488	8	739	0	262	1,156
Groups first order, best 5 from 15												
6	457	2	5950	4290	1200	129	296	3	729	10	302	0,854
7	438	9	5859	4165	1207	130	324	6	728	0	247	0,912
8	422	2	6390	3180	1263	120	316	13	723	12	277	0,945
9	465	6	4610	5065	1216	101	335	10	738	7	282	0,945
10	431	9	6220	4880	1223	106	293	11	727	7	258	0,951
Tutors first order, best 5 from 15												
11	285	39	5425	2805	1031	97	478	4	713	0	259	1,060
12	335	38	4840	5180	1019	107	452	11	732	0	208	1,115
13	312	54	5710	4495	1003	106	490	11	727	0	264	1,219
14	301	31	5215	4215	990	100	520	26	726	0	253	1,245
15	316	46	5830	2875	1036	99	509	18	736	0	283	1,247

- How get optimal order?

Diagonals first fill with out of order fill

$$|S_{ij}| = 1$$



- Universal rule, can be used in different combinatorial problems!

Diagonals first fill with out of order fill

$$|S_{ij}| = 1$$

Method	Without out of order fill	With out of order fill	Gain
RS	1,0 DLS/s	164 DLS/s	164x
WRS	3,1 DLS/s	203 DLS/s	65x
AC	0,2 DLS/s	224 DLS/s	1120x
BF	363 DLS/s	13 000 – 15 000 DLS/s	36x – 41x

- Pace is **15 000 DLS/s**!

Fast checking of ability sets

$$S_{ij} = U \setminus \bigcup_{k=1}^{N} \{a_{ik}\} \setminus \bigcup_{k=1}^{N} \{a_{kj}\} \setminus \underbrace{\bigcup_{k=1}^{N} \{a_{kk}\}}_{\substack{\text{only for main} \\ \text{diagonal elements} \\ \text{with } i=j}} \setminus \underbrace{\bigcup_{k=1}^{N} \{a_{k,\,N-k}\}}_{\substack{\text{only for second} \\ \text{diagonal elements} \\ \text{with } i+j=N}},$$

$$s_i = \bigcup_{k=1}^{N} \{a_{ik}\} \qquad r_j = \bigcup_{k=1}^{N} \{a_{kj}\} \qquad d_1 = \bigcup_{k=1}^{N} \{a_{kk}\} \qquad d_2 = \bigcup_{k=1}^{N} \{a_{k,\,N-k}\}$$

Method	Without fast checking	With fast checking	Gain
RS	164 DLS/s	662 DLS/s	4x
WRS	203 DLS/s	740 DLS/s	3,6x
AC	224 DLS/s	781 DLS/s	3,5x
BF	13 000 – 15 000 DLS/s	38 000 DLS/s	2,5x – 2,9x

- Pace is **38 000 DLS/s**!

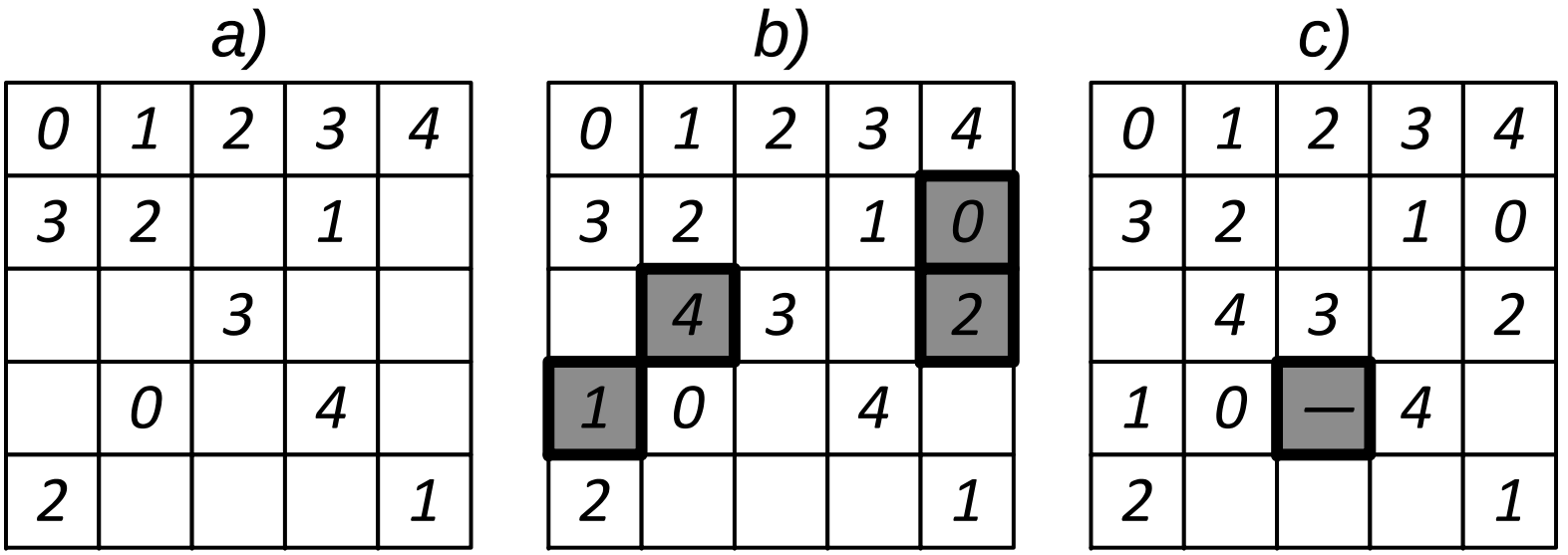
Excluding background CPU load (Hyper-Threading, caches, …)

Gerasim@Home and SAT@Home through BOINC – background CPU load must be excluded!

Method	Before	After	Gain
RS	662 DLS/s	1 040 DLS/s	1,6x
WRS	740 DLS/s	1 130 DLS/s	1,5x
AC	781 DLS/s	1 190 DLS/s	1,5x
BF	38 000 DLS/s	56 000 DLS/s	1,5x

- Pace is **56 000 DLS/s** for single-threaded program!

Clipping and early combinatorial returns



Method	Without	With	Gain
RS	1 040 DLS/s	1 040 DLS/s	–
WRS	1 130 DLS/s	1 170 DLS/s	+3,5%
AC	1 190 DLS/s	–	
BF	56 000 DLS/s	101 000 DLS/s	1,8x

- AC with returns → WRS with returns
- Pace is **101 000 DLS/s** for single-threaded program!

Diagonals first fill with none sequential fill with values

First string fill

1	2	3	4	5	6

Main diagonal fill

	7				
		8			
			9		
				10	
					11

Second diagonal fill

				12	
			13		
		14			
	15				
16					

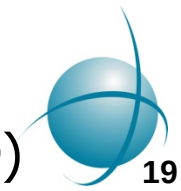
Remaining elements fill (DLS 1)

17		18	27		28
33	34			19	20
35	36			25	26
22		21	29		30
	32	23	31	24	

Remaining elements fill (DLS 2)

17		19	18		20
24	25			33	34
21	22			35	36
23		26	30		31
	27	28	29	32	

- Pace is **200 000 DLS/s** for recurrent Delphi program
- Pace is **217 000 DLS/s** for recurrent C++ program (VS2012)
- Pace is **240 000 DLS/s** for recurrent C++ program (VS2012 + PGO)



Diagonals first fill with none sequential fill with filled elements

```
procedure BruteForceRecurse_FillAll_Fast_v3_Iterative(X, Y: Integer; var LS: TLatinSquare);
var
  Stack: array [0..N*N-1] of record
    CurrK, CurrX, CurrY: Integer;
  end;
  CurrDepth, I: Integer;   { Текущая глубина рекурсии }
  NextX, NextY, K: Integer;
  PresentZeroCountElems: Boolean;
begin
  CurrDepth := 0;

  Stack[0].CurrX := X;
  Stack[0].CurrY := Y;

  for I := 0 to High(Stack) do begin
    Stack[I].CurrK := -1;

    GetNextIndexes(X, Y, NextX, NextY);

    Stack[I+1].CurrX := NextX;
    Stack[I+1].CurrY := NextY;

    X := NextX;
    Y := NextY;

    if X = N then
      break;
  end;

  while CurrDepth >= 0 do begin
    X := Stack[CurrDepth].CurrX;
    Y := Stack[CurrDepth].CurrY;

    { Условие завершения рекурсии }
    if X = N then begin
      BruteForce_PrintDecision(LS);

      { Рекуррентный возврат }
      Dec(CurrDepth);

      if (CurrDepth >= 0) and (Stack[CurrDepth].CurrX {X} <> Stack[CurrDepth].CurrY {Y}) an
        UnsetLsItem_Fast_WithoutDiagonals(LS, Stack[CurrDepth].CurrX {X}, Stack[CurrDepth].
```

```
for (LS[77] = 0; LS[77]< N; LS[77]++) {
    if (!Strs[7][LS[77]] || !Rows[7][LS[77]] || !d1[LS[77]])
        continue;

    Strs[7][LS[77]] = 0;
    Rows[7][LS[77]] = 0;
    d1[LS[77]] = 0;

for (LS[33] = 0; LS[33]< N; LS[33]++) {
    if (!Strs[3][LS[33]] || !Rows[3][LS[33]] || !d1[LS[33]])
        continue;

    Strs[3][LS[33]] = 0;
    Rows[3][LS[33]] = 0;
    d1[LS[33]] = 0;

for (LS[36] = 0; LS[36]< N; LS[36]++) {
    if (!Strs[3][LS[36]] || !Rows[6][LS[36]] || !d2[LS[36]])
        continue;

    Strs[3][LS[36]] = 0;
    Rows[6][LS[36]] = 0;
    d2[LS[36]] = 0;

for (LS[63] = 0; LS[63]< N; LS[63]++) {
    if (!Strs[6][LS[63]] || !Rows[3][LS[63]] || !d2[LS[63]])
        continue;

    Strs[6][LS[63]] = 0;
    Rows[3][LS[63]] = 0;
    d2[LS[63]] = 0;

for (LS[66] = 0; LS[66]< N; LS[66]++) {
    if (!Strs[6][LS[66]] || !Rows[6][LS[66]] || !d1[LS[66]])
        continue;
```

- Pace is **212 000 DLS/s** for iterative Delphi program
- Pace is **340 000 DLS/s** for iterative C++ program (VS2012 + PGO)

SAT-generation of DLSs

Number of DLS	Time	Pace
1 000	0,34 s	2 941 DLS/s
10 000	8,528 s	1 173 DLS/s
100 000	504,468 s	198 DLS/s

- Pace decreased during increasing number of restrictions

Constant fill order: maximum restrictions (minimum abilities) rule

$$\left|S_{ij}\right| \to \min$$

a)

x	x	x	x	x
1	2	1	2	1
1	1	**3**	1	1
1	2	1	2	1
2	1	1	1	2

b)

x	x	x	x	x
1	**3**	2	3	1
2	2	x	2	2
1	3	2	3	1
3	1	2	1	3

c)

x	x	x	x	x
2	x	3	**4**	2
2	3	x	2	2
1	4	2	4	1
3	2	2	1	1

d)

x	x	x	x	x
3	x	4	x	3
2	3	x	3	2
1	**5**	2	5	1
4	2	2	2	4

e)

x	x	x	x	x
3	x	4	x	3
2	4	x	3	2
2	x	3	**6**	2
5	3	2	2	4

f)

x	x	x	x	x
3	x	4	x	3
2	4	x	4	2
3	x	4	x	3
5	3	2	3	5

g)

x	x	x	x	x
4	x	4	x	3
3	4	x	4	2
4	x	4	x	3
x	4	3	4	**6**

h)

x	x	x	x	x
4	x	4	x	4
3	4	x	4	3
4	x	4	x	4
x	**5**	4	5	x

i)

x	x	x	x	x
4	x	4	x	4
3	5	x	4	3
4	x	4	x	4
x	x	5	**6**	x

j)

x	x	x	x	x
4	x	4	x	4
3	5	x	5	3
4	x	4	x	4
x	x	**6**	x	x

k)

x	x	x	x	x
4	x	**5**	x	4
3	5	x	5	3
4	x	5	x	4
x	x	x	x	x

l)

x	x	x	x	x
5	x	x	x	5
3	5	x	5	3
4	x	**6**	x	4
x	x	x	x	x

m)

x	x	x	x	x
5	x	x	x	5
3	5	x	5	3
5	x	x	x	5
x	x	x	x	x

n)

x	x	x	x	x
x	x	x	x	**6**
4	5	x	5	3
6	x	x	x	5
x	x	x	x	x

o)

x	x	x	x	x
x	x	x	x	x
4	5	x	5	4
6	x	x	x	6
x	x	x	x	x

p)

x	x	x	x	x
x	x	x	x	x
5	5	x	5	4
x	x	x	x	**7**
x	x	x	x	x

q)

x	x	x	x	x
x	x	x	x	x
5	5	x	5	5
x	x	x	x	x
x	x	x	x	x

r)

x	x	x	x	x
x	x	x	x	x
x	**6**	x	6	6
x	x	x	x	x
x	x	x	x	x

s)

x	x	x	x	x
x	x	x	x	x
x	x	x	**7**	7
x	x	x	x	x
x	x	x	x	x

t)

x	x	x	x	x
x	x	x	x	x
x	x	x	x	**8**
x	x	x	x	x
x	x	x	x	x

u)

1	2	3	4	5
18	7	16	8	19
22	23	6	24	25
20	9	17	10	21
11	13	15	14	12

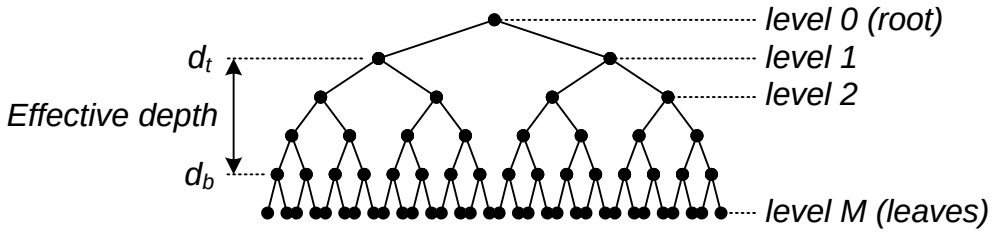
- Pace is **790 000 DLS/s** for iterative C++ program (VS2012 + PGO)

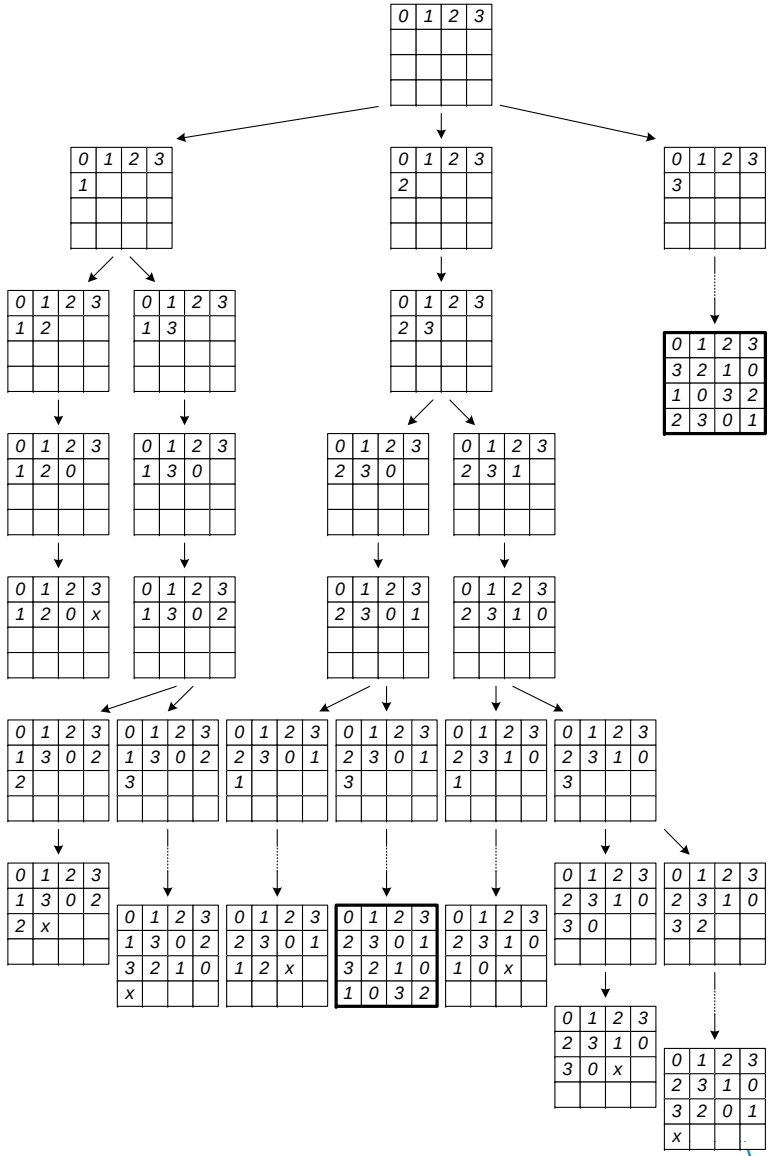
Results: number of DLS of order N<9

N	Number of DLS with fixed first row	$N!$	Total number of DLS	Computing time on 1 CPU core
1	1	1	1	< 1 second
2	0	2	0	< 1 second
3	0	6	0	< 1 second
4	2	24	48	< 1 second
5	8	120	960	< 1 second
6	128	720	92 160	< 1 second
7	171 200	5 040	862 848 000	2 seconds
8	7 447 587 840	40 320	300 286 741 708 800	30 hours

Clippings on the selected depth

$$0 \le d_t < d_b \le N^2 - N$$





- Clippings are effective only at 41<d<61 for N=9
- Pace is **1 100 000 DLS/s**

Magic squares and its magic constants

$$\sum_{i=1}^{N} a_{ij} = \sum_{i=1}^{N} a_{ji} = \sum_{i=1}^{N} a_{ii} = \sum_{i=1}^{N} a_{i,\,N-i+1} = 0+1+\ldots+(N-1) = \frac{N(N-1)}{2},\ \forall j = \overline{1,\,N}.$$

$$a_{2,10} = \frac{10(10-1)}{2} - a_{21} - a_{22} - \ldots - a_{29}$$

a)

–	–	–	–	–	–	–	–	–
19	1	15	16	20	17	18	2	21
24	25	5	22	26	23	6	27	28
52	53	54	9	50	10	55	56	57
64	65	66	67	0	68	69	70	71
58	59	60	11	51	12	61	62	63
31	32	7	29	33	30	8	34	35
38	3	41	36	39	37	42	4	40
13	46	47	43	45	44	48	49	14

b)

–	–	–	–	–	–	–	–	–
19	1	15	16	20	17	18	2	21
24	25	5	22	26	23	6	27	28
54	55	56	9	52	10	57	58	59
60	62	64	44	0	46	66	68	69
61	63	65	11	53	13	67	70	71
31	32	7	29	33	30	8	34	35
38	3	41	36	39	37	42	4	40
12	48	49	43	47	45	50	51	14

- Filling order is slightly differ
- Pace is **1 800 000 DLS/s**

Magic with bits arithmetic

Set of available elements

$$x = s_{ik} \vee c_{jk} \left[\vee d_k^{(1)} \right] \left[\vee d_k^{(2)} \right]$$

Fast selecting of available elements

$$y = \left(x \oplus \underbrace{11\ldots1}_{N} \right) \wedge \left(- \left(x \oplus \underbrace{11\ldots1}_{N} \right) \right)$$

- Square is stored as bit masks
- Pace is **6 600 000 DLS/s** (Visual Studio 2015)

Results: number of DLS of order N<10

A274171 (Number of diagonal Latin squares of order n with first row 1..n)
1, 0, 0, 2, 8, 128, 171200, 7447587840, 5056994653507584

A274806 (Number of diagonal Latin squares of order n)
1, 0, 0, 48, 960, 92160, 862848000, 300286741708800,
1835082219864832081920

$$L_{10} \simeq (7{,}6 \div 10{,}9) \cdot 10^{22}$$

~250 000 years at Gerasim@Home distributed computing project
~1 year at 1 PFLOP/s supercomputer (who can help us? :))

- Gerasim@Home (~500 PCs, ~3 months, 2–5 TFLOP/s), http://gerasim.boinc.ru
- Matrosov academician computing cluster (~500 24/7 CPU cores, ~3 months)
- https://oeis.org (Online Encyclopedia of Integer Sequences, OEIS)

Our publications (only enumeration of DLS!)

- Vatutin E.I., Zhuravlev A.D., Zaikin O.S., Titov V.S. Features of using weighting heuristics in the problem of finging diagonal Latin squares (in Russian) // Proceeding of Southwest State University. Series: Control, Computer Science, Informatics. Medical Devices. 2015. № 3 (16). P. 18–30. http://evatutin.narod.ru/evatutin_co_01_ls_g_rs_wrs_ac.pdf
- Vatutin E.I., Zaikin O.S., Zhuravlev A.D., Manzuk M.O., Kochemazov S.E., Titov V.S. Using grid systems for enumerating combinatorial objects on example of diagonal Latin squares // Distributed computing and grid-technologies in science and education (GRID'16): book of abstracts of the 7th international conference. Dubna: JINR, 2016. p. 114–115. http://evatutin.narod.ru/evatutin_co_ls_dls_1_8_eng.pdf
- Vatutin E.I., Zhuravlev A.D., Zaikin O.S., Titov V.S. Using algorithmic features in the problem of generating diagonal Latin squares (in Russian) // Proceedings of Southwest State University. 2016. № 2 (65). P. 46–59. http://evatutin.narod.ru/evatutin_co_ls_S_0_and_1.pdf
- Vatutin E.I., Zaikin O.S., Zhuravlev A.D., Manzuk M.O., Kochemazov S.E., Titov V.S. The effect of filling cells order to the rate of generation of diagonal Latin squares (in Russian) // Information-measuring and diagnosing control systems (Diagnostics – 2016). Kursk: SWSU, 2016. P. 33–39. http://evatutin.narod.ru/evatutin_co_ls_dls_1_8.pdf
- Vatutin E.I., Titov V.S., Zaikin O.S., Kochemazov S.E., Valyaev S.Yu., Zhuravlev A.D., Manzuk M.O. Using grid-systems for enumerating combinatorial objects on example of diagonal Latin squares of order 9 (in Russian) // Information technologies and mathematical modeling of systems 2016. M.: Center for Information Technology in the design of the RAS, 2016. P. 154–157. http://evatutin.narod.ru/evatutin_co_ls_dls_9.pdf

Conclusion and prospects of future work

1. Current pace is **6 600 000 DLS/s**. This is end? GPU?
2. We can enumerate DLS and MOLS of some kinds.
3. We can find all orthogonal pairs of DLS and MOLS for small N.
4. We can organize parameters space exploration with heuristic methods, local analysis with LBF and find some orthogonal pairs of DLS for big N.
5. We can get isomorphism classes and canonical forms for DLS.
6. We can use transversal-based approaches.
7. We can use special block or intercalate structure of DLS (citerra, whitefox).
8. This problems are weakly coupled and can be solved with volunteer computing support!

The End.
Thanks!

The authors would like to thank all volunteers who took part in the calculation within the distributed computing project Gerasim@Home!

WWW: http://evatutin.narod.ru
E-mail: evatutin@rambler.ru
LJ: http://evatutin.livejournal.com
Skype: evatutin
vk, ok, facebook