Southwest State University
Department of Computer Sciences

QUALITY ANALYSIS OF BLOCK SEPARATIONS OF GRAPH-SCHEMES OF PARALLEL CONTROL ALGORITHMS DURING LOGIC CONTROL SYSTEMS DESIGN USING GRID SYSTEMS ON VOLUNTEER BASIS

Eduard I. Vatutin

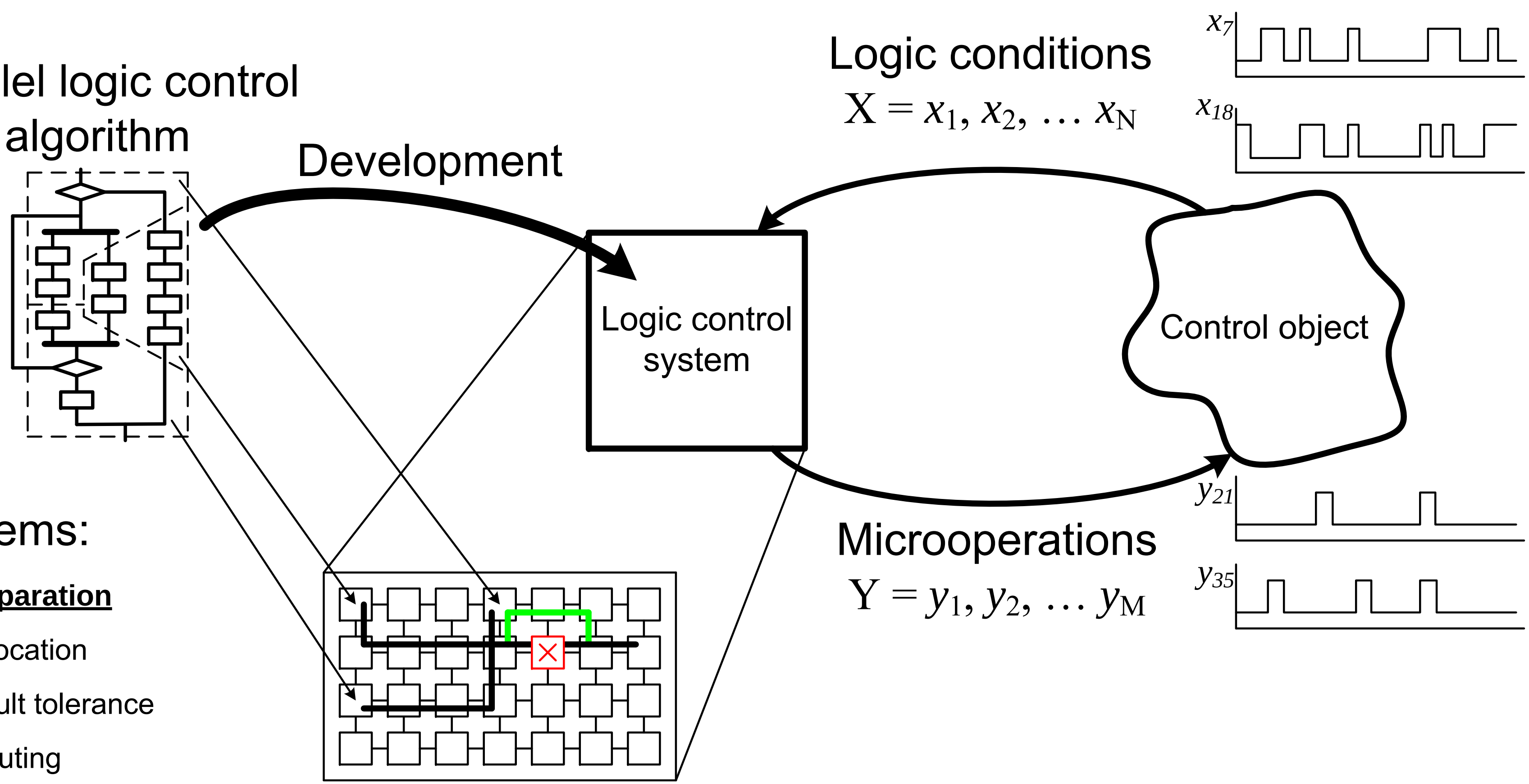
Sergey Yu. Valyaev

Vitaly S. Titov

GRID'16

Dubna, JINR, 2016

Control object and Logic Control System (LCS)



Problems:

1. Separation

2. Allocation

3. Fault tolerance

4. Routing

5. Collective interactions

etc.

Statement of a problem

$$\bigcup_{i=1}^{H} A_i = A^0, \quad A_i \neq \varnothing, \quad A_i \bigcap A_j = \varnothing, \quad i, j = \overline{1, H}, \quad i \neq j,$$

$$\neg\left(a_i \omega a_j\right) \quad \forall a_i, a_j \in A_k, \quad i \neq j, \quad k = \overline{1, H},$$

$$W\left(A_i\right) \leq W_{\max}, \quad \left|X\left(A_i\right)\right| \leq X_{\max}, \quad \left|Y\left(A_i\right)\right| \leq Y_{\max}, \quad i = \overline{1, H}.$$

Discrete combinatorial optimization problem
NP-hard problem
Multicriteria optimization problem

Quality criteria

$$H \to \min$$

$$Z_1 = \sum_{i=1}^{H} \sum_{j=1, j \neq i}^{H} \alpha\left(A_i, A_j\right) \to \min$$

$$Z_2 = \sum_{i=1}^{H} \sum_{j=1, j \neq i}^{H} \delta\left(A_i, A_j\right) \to \min$$

$$Z_3 = \sum_{i=1}^{H} \left|X\left(A_i\right)\right| - \left|X\left(A^0\right)\right| \to \min$$

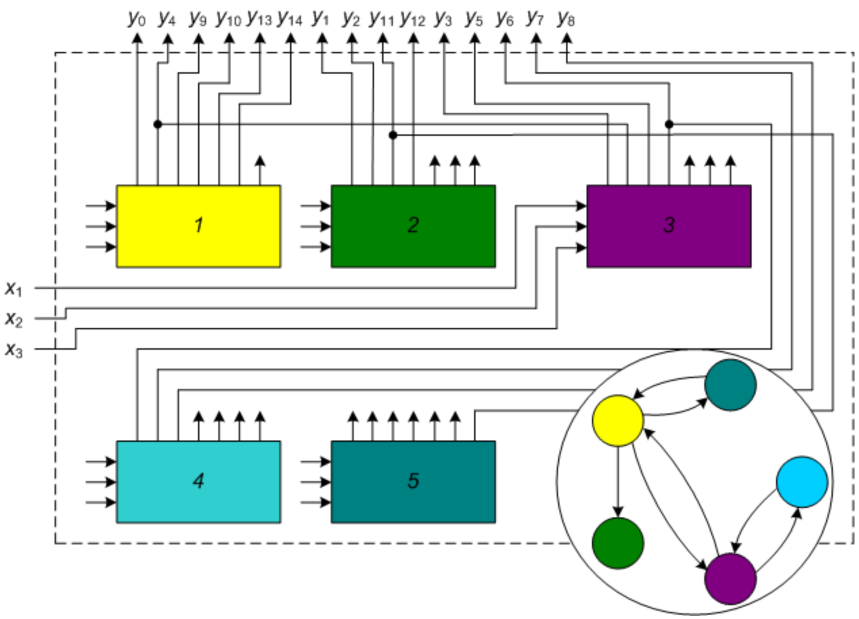
$$Z_4 = \sum_{i=1}^{H} \left|Y\left(A_i\right)\right| - \left|Y\left(A^0\right)\right| \to \min$$

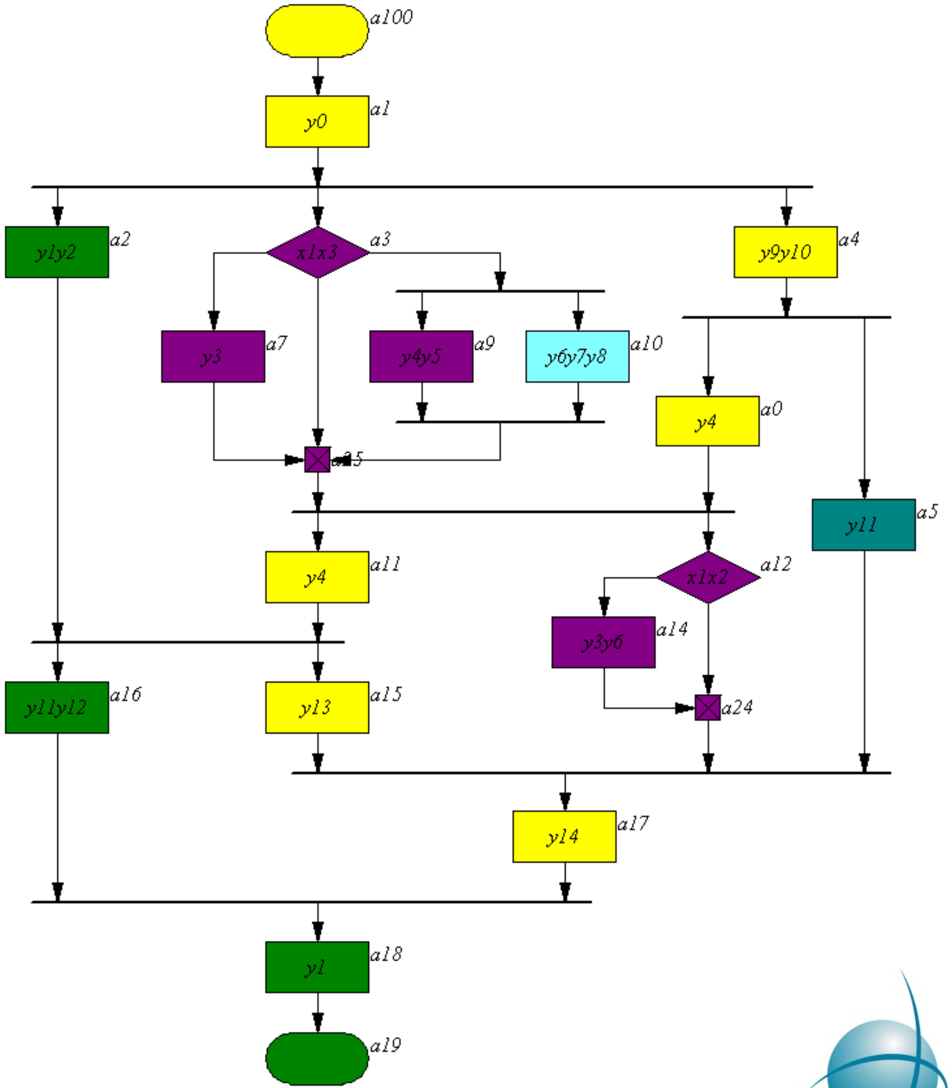
Example of separation (given with using PAE program system)

Просмотр разбиения - Параллельно-последовательный метод (PSM.dll)

№	Состав	Микрооперации	Логические условия	Вес
1	{a0,a4,a11,a15}	{y4,y9,y10,y13}	{}	4
2	{a1,a2,a16,a18,a19,a100}	{y0,y1,y2,y11,y12}	{}	4
3	{a5}	{y11}	{}	1
4	{a3,a7,a9,a12,a14,a17,a24,a25}	{y3,y4,y5,y6,y14}	{x1,x2,x3}	6
5	{a10}	{y6,y7,y8}	{}	1

Корректность разбиения:	Корректно	
Число блоков (подалгоритмов):	5 / 5	1.000
Повторяющихся логических сигналов:	0 / 3	0.000
Повторяющихся микроопераций:	3 / 15	0.060
Разность алгоритмов по сложности:	5	0.025
Сложность сети межблочных связей:	11 / 20	0.330
Интенсивность межблочного взаимодействия:	16.800 / 31.900	0.316
Время построения разбиения:	111056707 тактов, 130.976 мс	
Значение оценочной функции:		**1.731**





Methods of getting separations overview

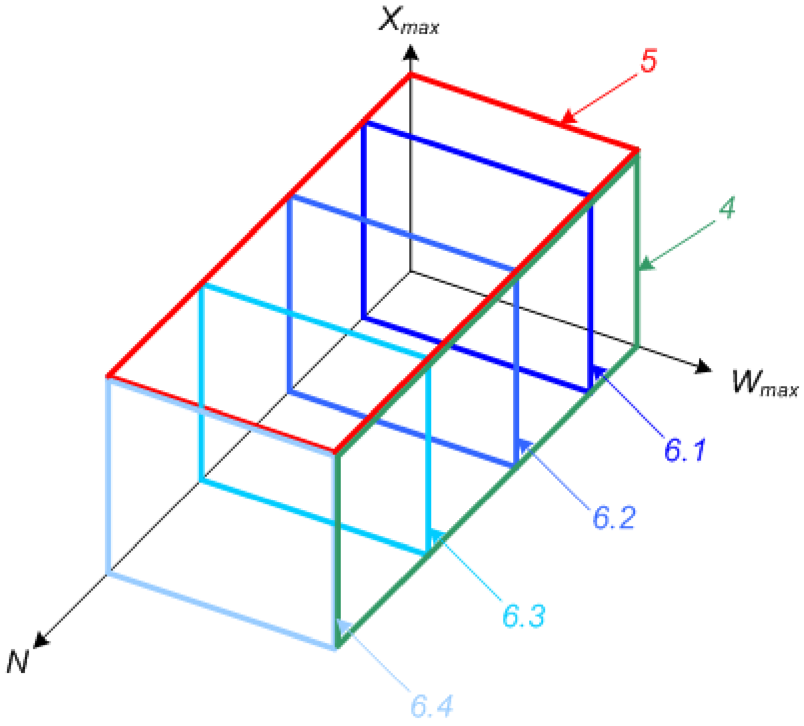
Methods of getting separations:

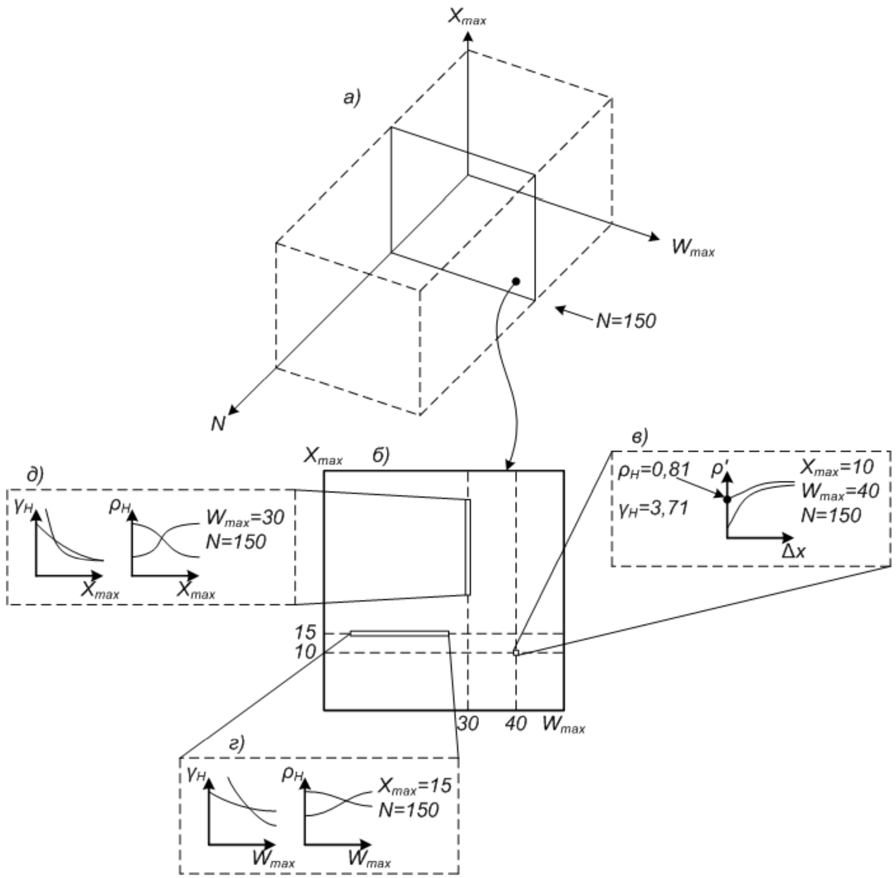
- **Brute Force** (NP-hard problem, N<10 only);
- **S.I. Baranov method** (1984, greedy approach);
- **Greedy with sequential building of blocks and using adjacent vertices** (2013, greedy with additional constraints);
- A.D. Zakrevsky method (coloring of special graph);
- **Parallel-sequential method** (1997–Now, special method with transformations of graph-schemes of parallel algorithms);
- Well known discrete combinatorial optimization methods (stochastic methods: **random search**, directed random search, ACO, BC, GA, SA, etc.).

Quality of decisions, time and memory costs, complexity and features of practical implementation are significantly differ!

Which method is the best?

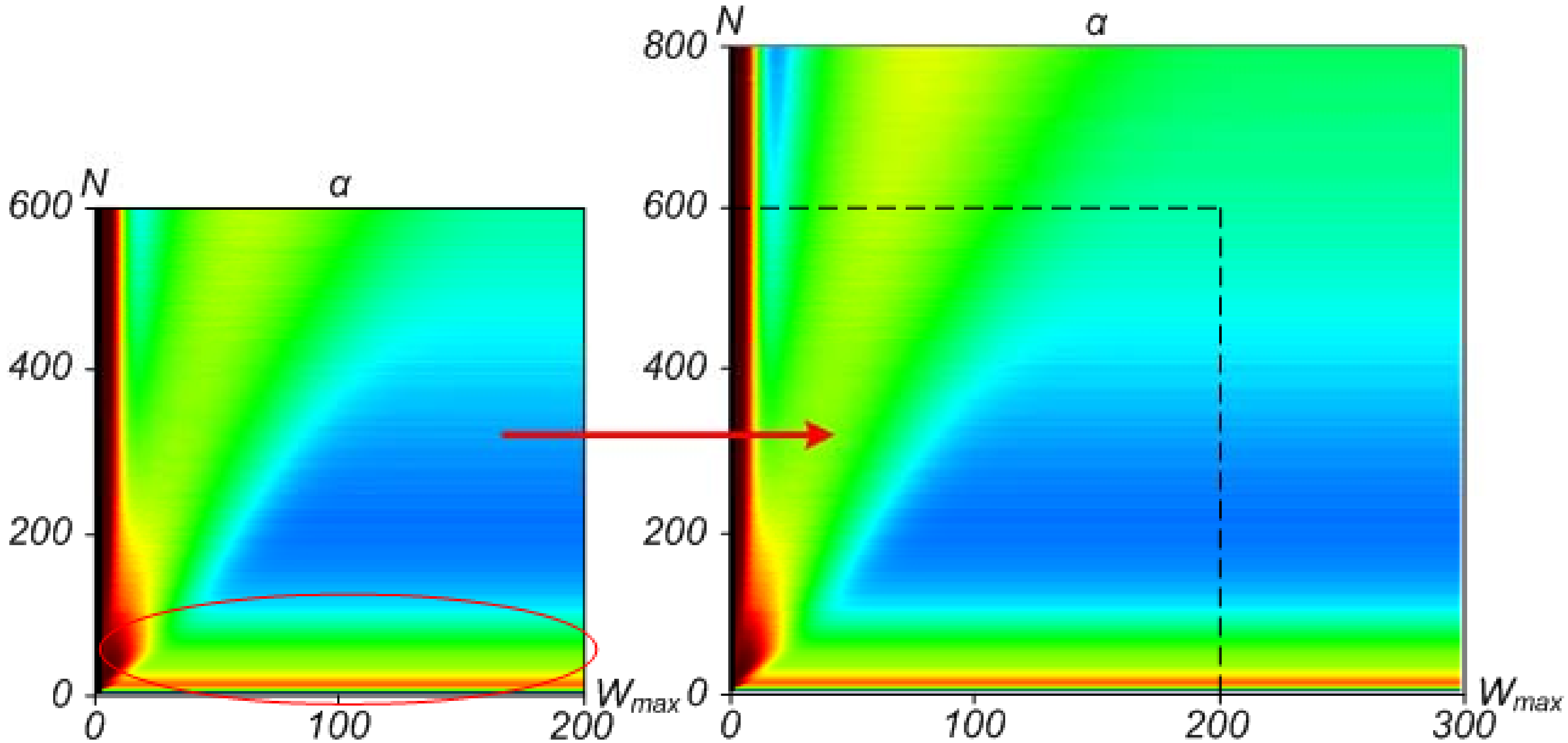
Quality analysis: parameters space and experiments



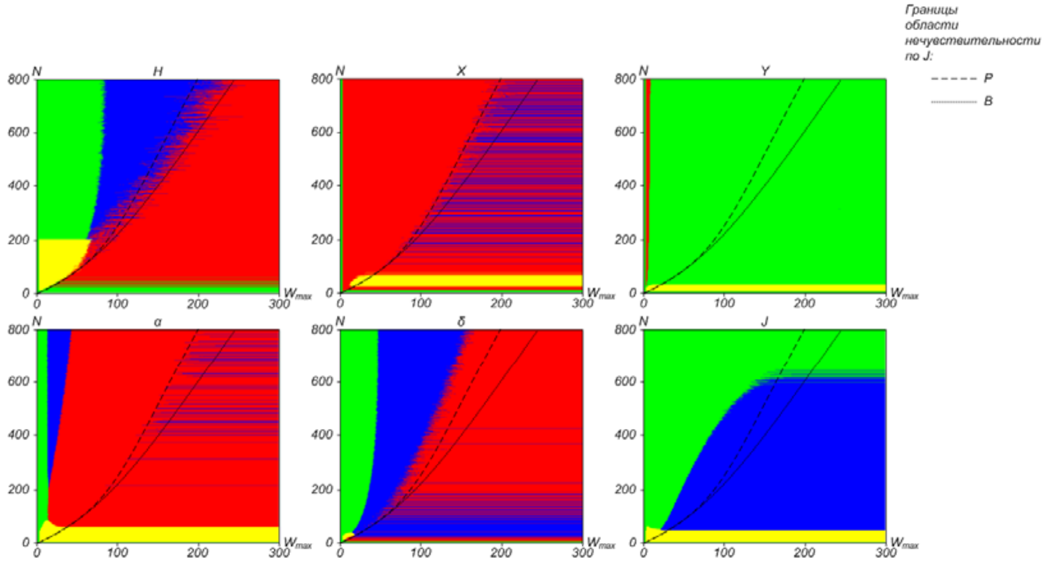


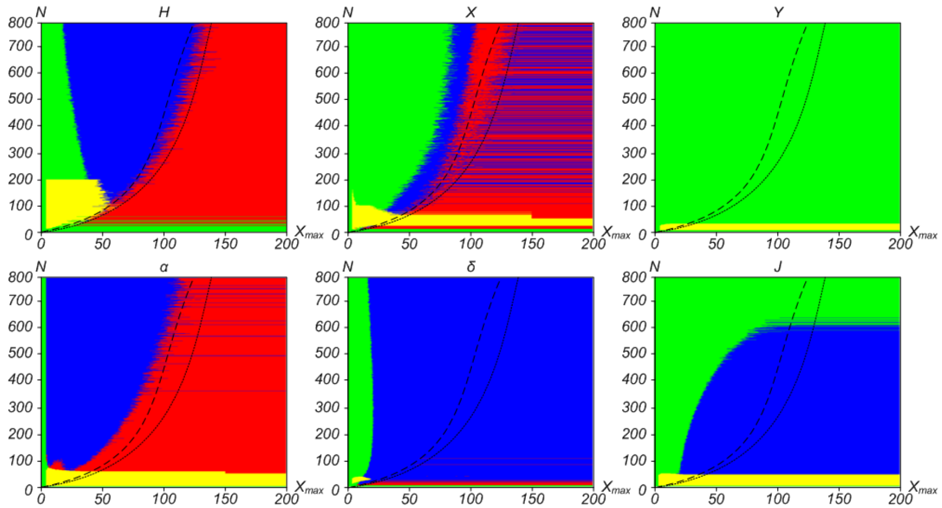
- fixed parameters experiments (2007), 10 minutes of computing time;
- variable value of 1 parameter (2008), 5 hours of computing time;
- variable values of 2 parameters (2010–Now), 430 years of computing time (with using BOINC).

Experimental results: expanding analyzed area



Example: prefer maps





red – S.I. Baranov method,
green – parallel-sequential method,
blue – greedy adjacent strategy,
yellow – random search.

Observed strong zone dependence of quality of decisions for different methods from area of parameters space!

What method is a best one? Answer depends from area of parameters space!

Southwest State University
Department of Computer Sciences

USING GREEDY METHODS ON VOLUNTEER BASIS FOR COMPARISON OF DECISIONS QUALITY OF HEURISTIC METHODS IN THE PROBLEM OF GETTING THE SHORTEST PATH IN THE GRAPH WITH GRAPH DENSITY CONSTRAINT

Eduard I. Vatutin

Sergey Yu. Valyaev

Vitaly S. Titov

GRID'16
Dubna, JINR, 2016

Classification of methods for solving discrete combinatorial optimization problems

Universal methods:

- Brute Force (branches and bounds strategy);
- Limited Depth First Search;
- Greedy approach;
- (Weighted) Random Search;
- Ant Colony Optimization;
- Simulated Annealing;
- Genetic Algorithms.

Special methods:

- Parallel-Sequential Method;
- Dijkstra's algorithm;
- Hungarian algorithm (Kuhn-Munkers algorithm);
- ...

Classification of methods for solving discrete combinatorial optimization problems

Precise methods (ensures an optimal solution):

- Brute Force (branches and bounds strategy);
- Dijkstra's algorithm;
- Hungarian algorithm (Kuhn-Munkers algorithm);
- Greedy approach (Minimal spanning tree).

Approximate methods (ensures an sub- or quasi-optimal solutions):

- Limited Depth First Search;
- Greedy approaches (most tasks);
- (Weighted) Random Search;
- Ant Colony Optimization;
- Simulated Annealing;
- Genetic Algorithms;
- Different special methods.

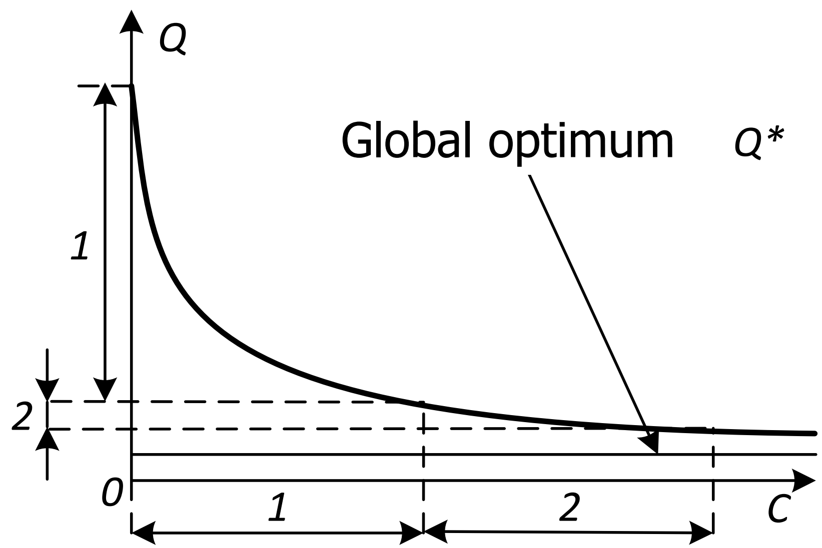
Classification of methods for solving discrete combinatorial optimization problems

Consecutive methods:

- Greedy approach;
- Some special methods (Dijkstra’s algorithm, parallel-sequential method, …).

Iterative methods:

- Brute Force (branches and bounds strategy);
- Limited Depth First Search;
- (Weighted) Random Search;
- Ant Colony Optimization;
- Simulated Annealing;
- Genetic Algorithms.



Assessment of the quality of decisions, convergence rate and computing time costs

Quality of decisions:

- during $C \to \infty$ $Q \to Q^*$;
- how to determine C_{min} that $\Delta Q = |Q - Q^*|$ is acceptable for selected problem (for example, $\Delta Q/Q^* 100\% < 5\%$)?

Ability of paralleling:

- Parallel execution is difficult or ineffective (for example, parallel-sequential method – theoretically no more than 10% speedup by Amdahl's law);
- Weak or strong coupled implementations (AC with M=1 or M=100) – limiting classes of hardware;
- Trivial parallelized (random search, weighted random search).

Assessment of the quality of decisions based on samples of random source data

Sample of random source data

$$\Lambda = \{G_1, G_2, ..., G_K\}$$

Average value of quality criteria

$$\bar{Q} = \frac{\sum_{i=1}^{K} Q(G_i)\phi(G_i)}{K}, \phi(G_i) \in \{0, 1\}$$

$$\bar{Q} < Q^* ?$$

Average deviation from optimum

$$\Delta\bar{Q} = \frac{\sum_{i=1}^{K} \left(Q(G_i) - Q^*(G_i)\right)\phi(G_i)}{K}$$

Assessment of the quality of decisions based on samples of random source data

Probability of finding decision

$$\overline{p} = \frac{\sum_{i=1}^{K} \phi(G_i)}{K}$$

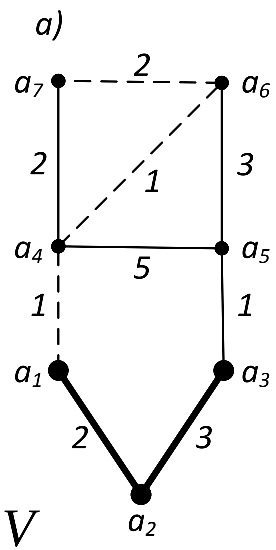
Probability of finding optimal decision

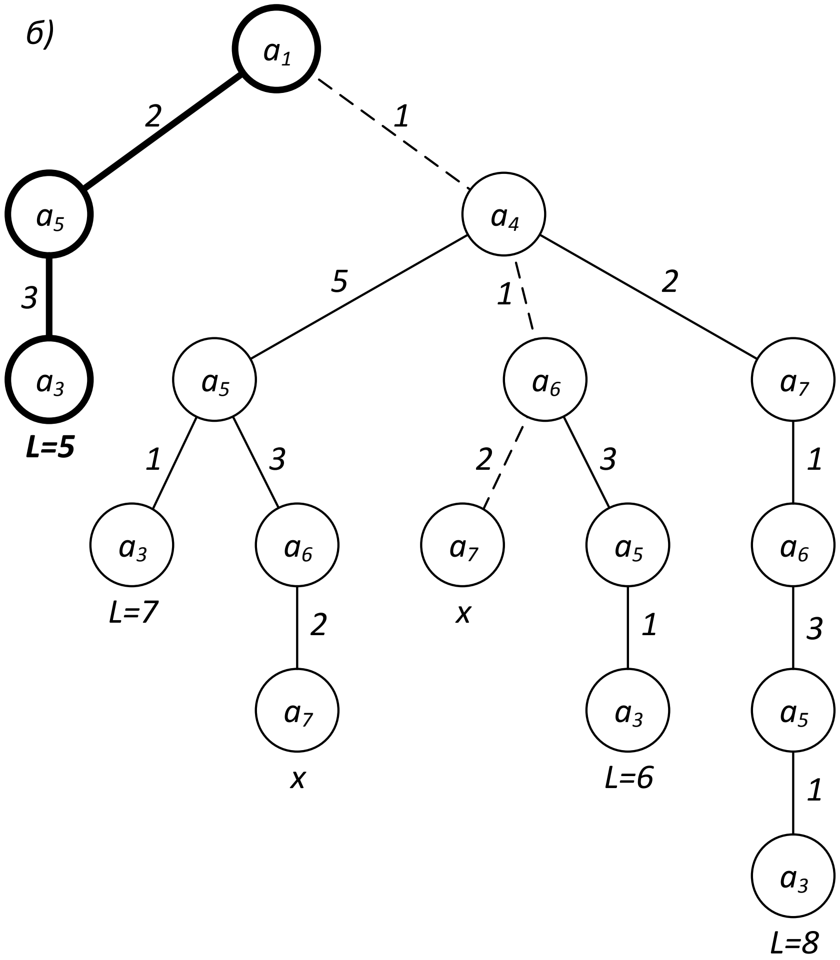
$$\overline{p}_{opt} = \frac{\sum_{i=1}^{K} \theta(G_i)}{K}, \theta(G_i) = \begin{cases} 0, & Q(G_i) > Q^*(G_i) \\ 1, & Q(G_i) = Q^*(G_i) \end{cases}$$

Average number of iterations (convergence rate)

$$\overline{C} = \frac{\sum_{i=1}^{K} C(G_i)}{K}$$

Statement of a problem





$$G = \langle A, V \rangle$$

$$A = \{a_1, a_2, ..., a_N\}$$

$$V = \{v_1, v_2, ..., v_M\}, A \times A \subseteq V$$

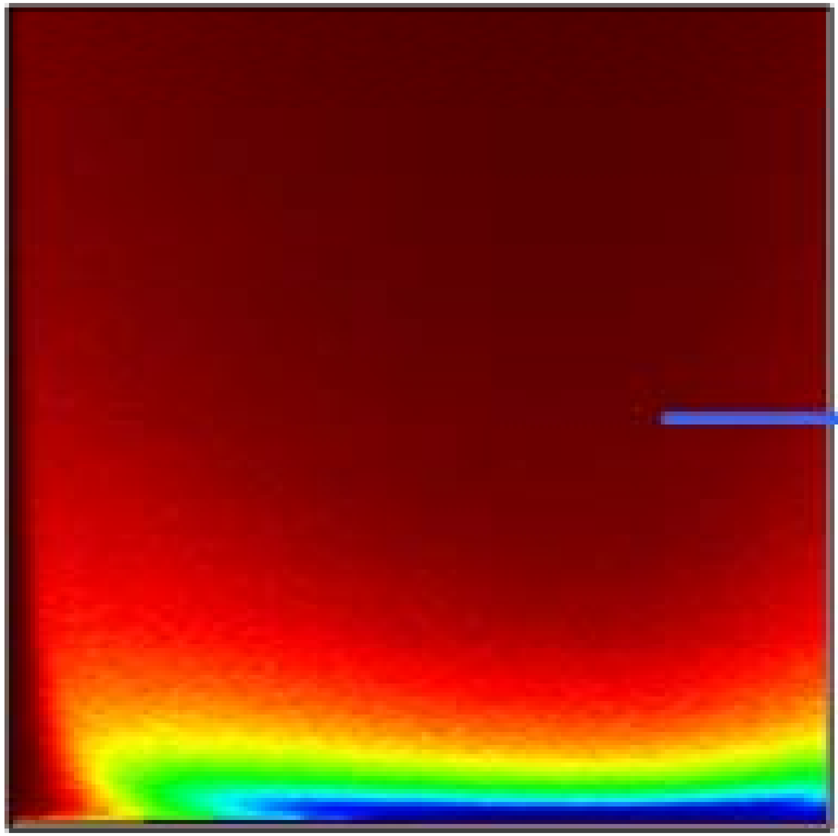
$$d = \frac{M}{N(N-1)}$$

$$l(v_i) = l_{j,k}, v_i = (a_i, a_k), \exists l(v_i) = \infty$$

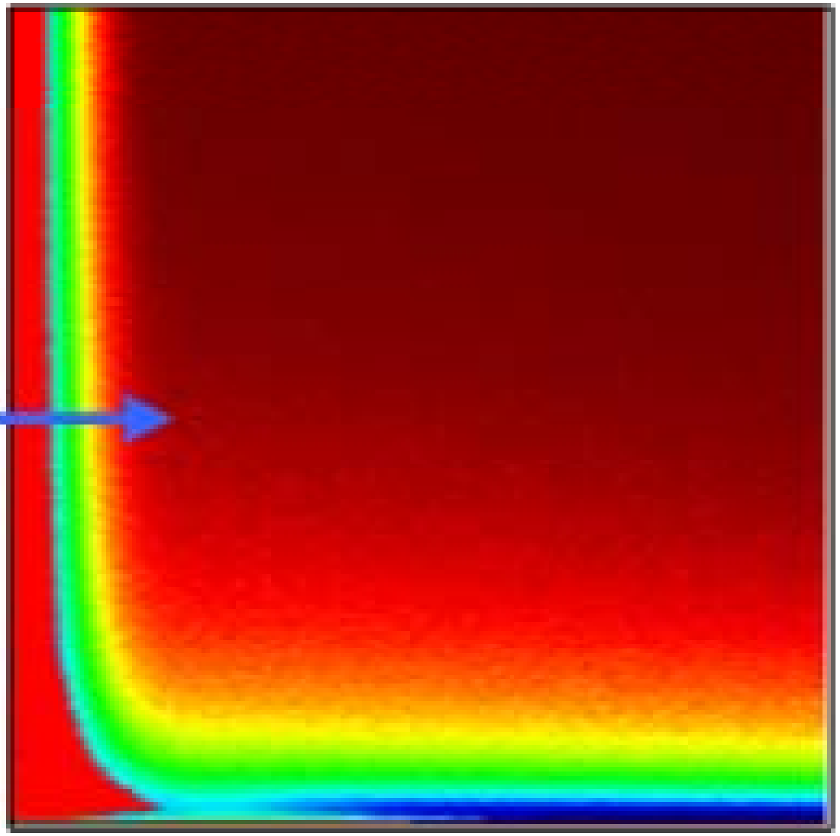
$$P = [a_{i_1}, a_{i_2}, ..., a_{i_m}]$$

$$L = \sum_{j=1}^{m-1} l_{i_j, i_{j+1}} \to \min$$

Simulated annealing: work on the bugs



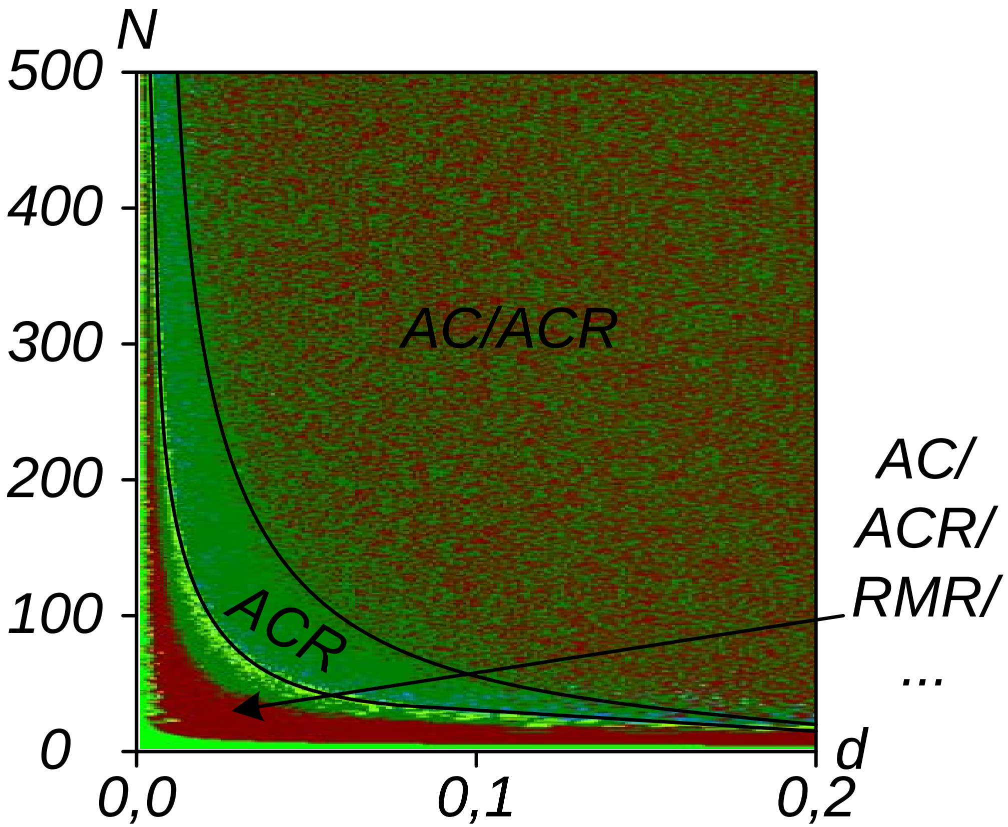
Version 1 (wrong)

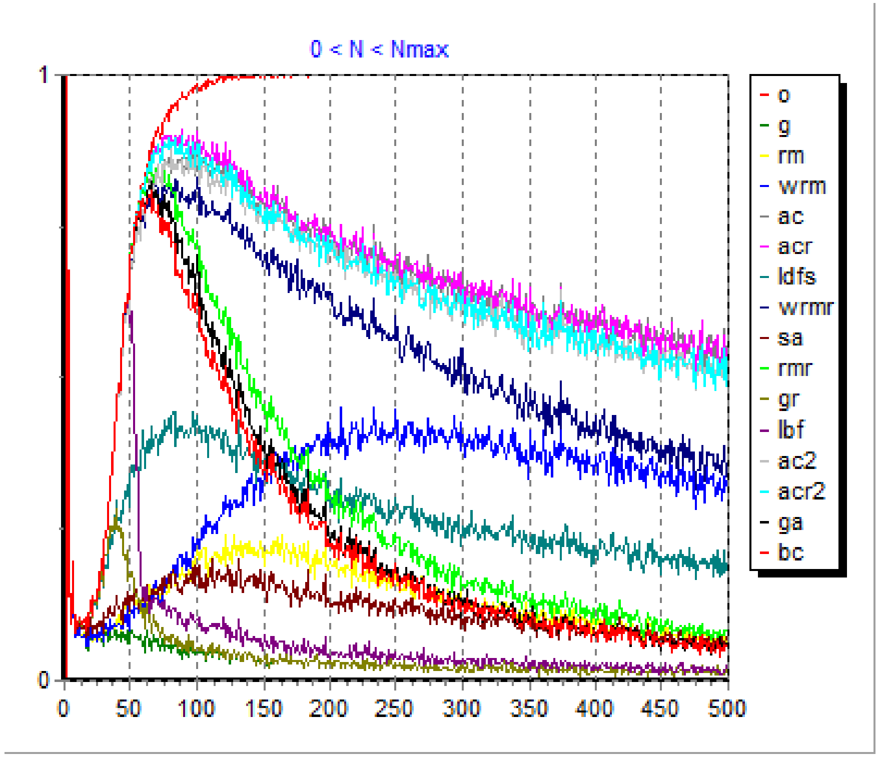


Version 2 (good)

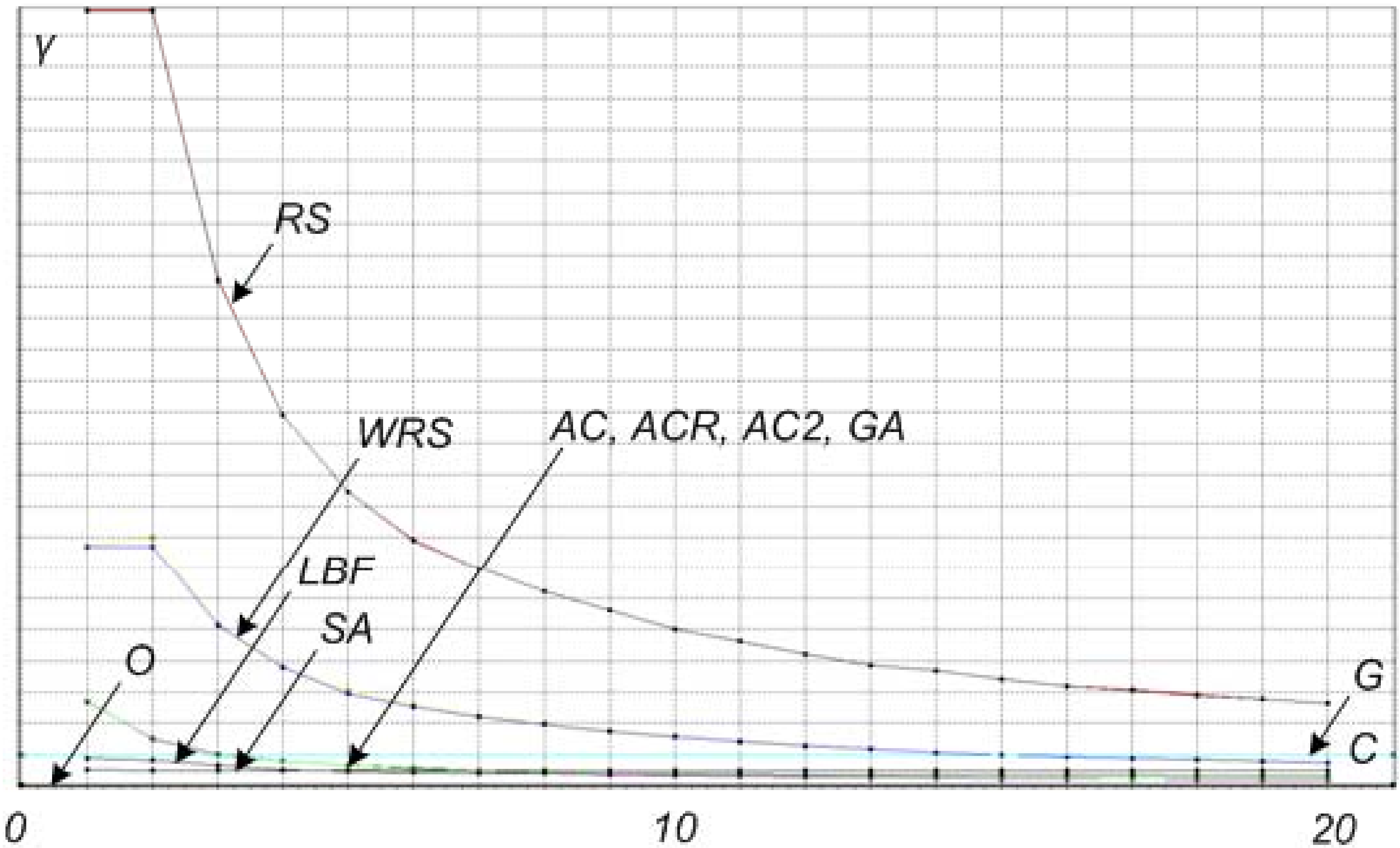
- Short (1 week) experiment within Gerasim@Home
- Version 1 was adopted for wrong conditions during meta-optimization!
- No significant changes!

Total comparison of all methods



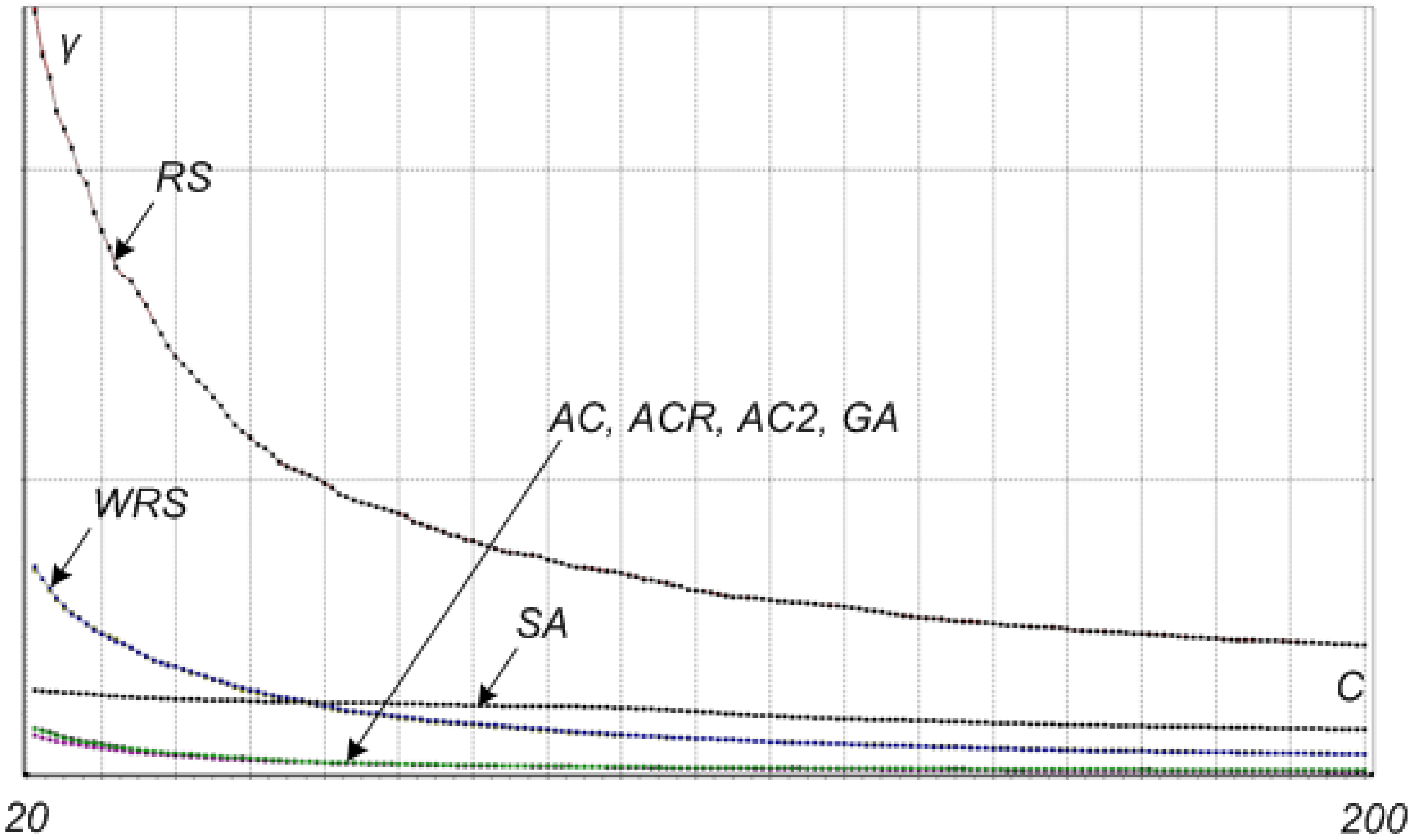


Convergence rate analysis



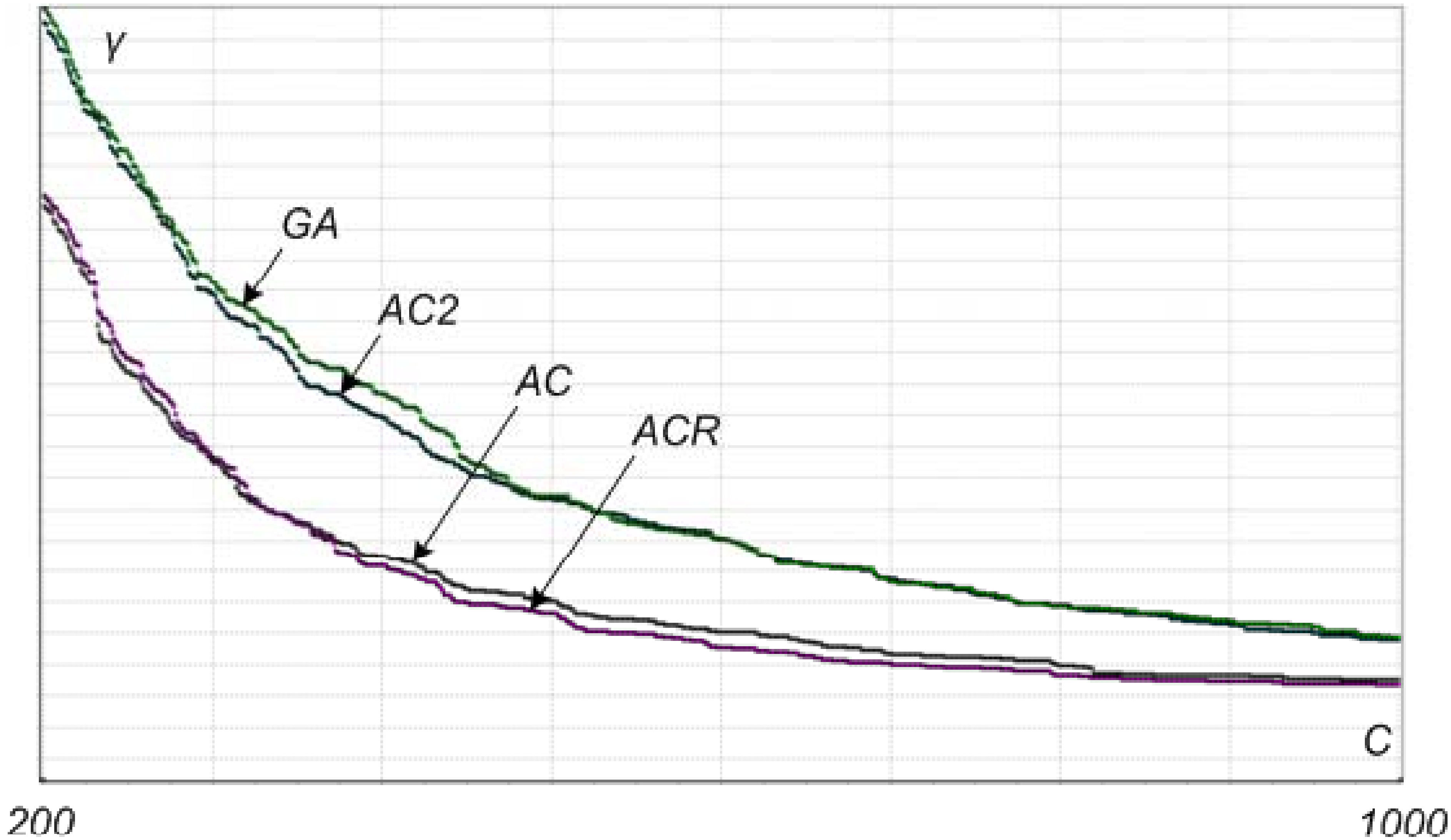
- Convergence rate significantly differ for different methods!

Convergence rate analysis

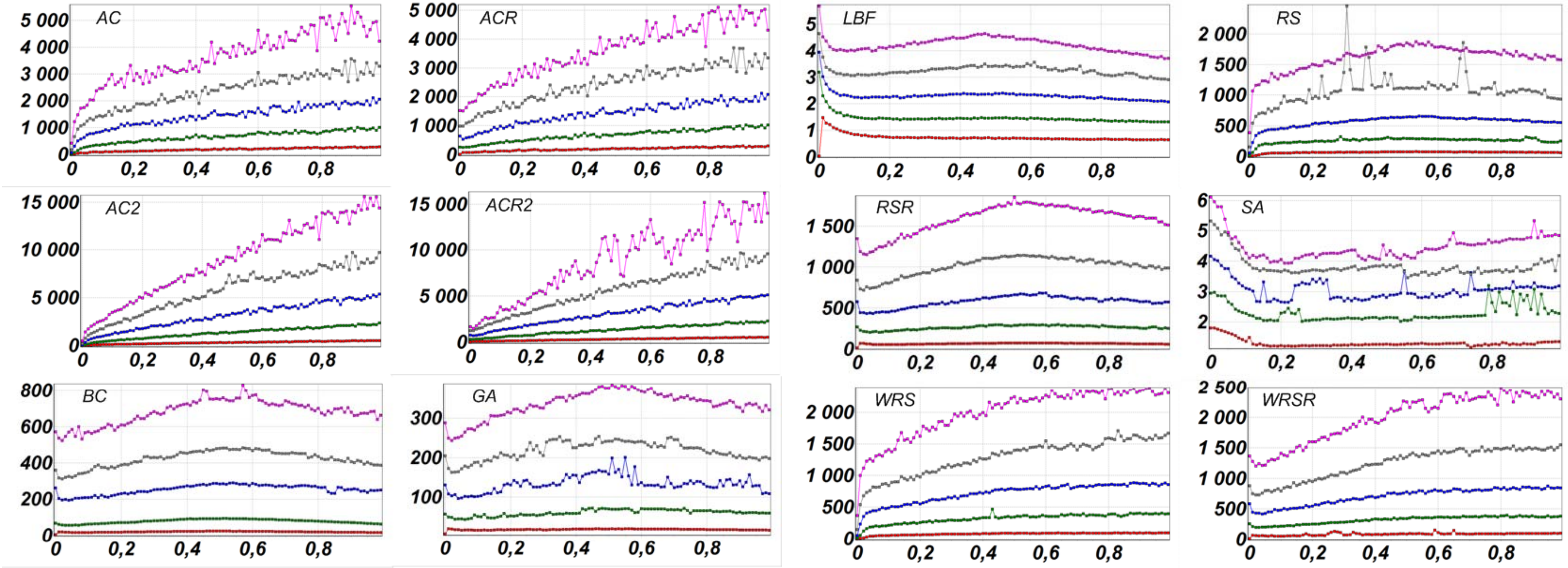


- Methods can be combined?

Convergence rate analysis



Careful analysis: computing time costs



“Long” set

“Short” set

- Time depends from graph density!
- Long (ACR2) vs. Short (LBF) comparison: 3000 times differ!
- Time costs must be considered in the convergence rate experiments!

Southwest State University
Matrosov Institute for System Dynamics and Control Theory
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 E. Kochemazov, Vitaly S. Titov

GRID'16
Dubna, JINR, 2016

Latin squares: what is it?

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

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

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

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

$$N = \left|S\right|$$

$$S = \left\{0, 1, 2, \ldots, N-1\right\}$$

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

Normalized Latin square with order 10

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

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

First string ordered diagonal Latin square with order 10

$$\left(N-1\right)!$$

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

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

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

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

$$C(A) = 3$$

$$S^U_{3,10} = \{4, 9\}$$

Getting Latin squares: random search approach (RS)

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

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

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

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

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)

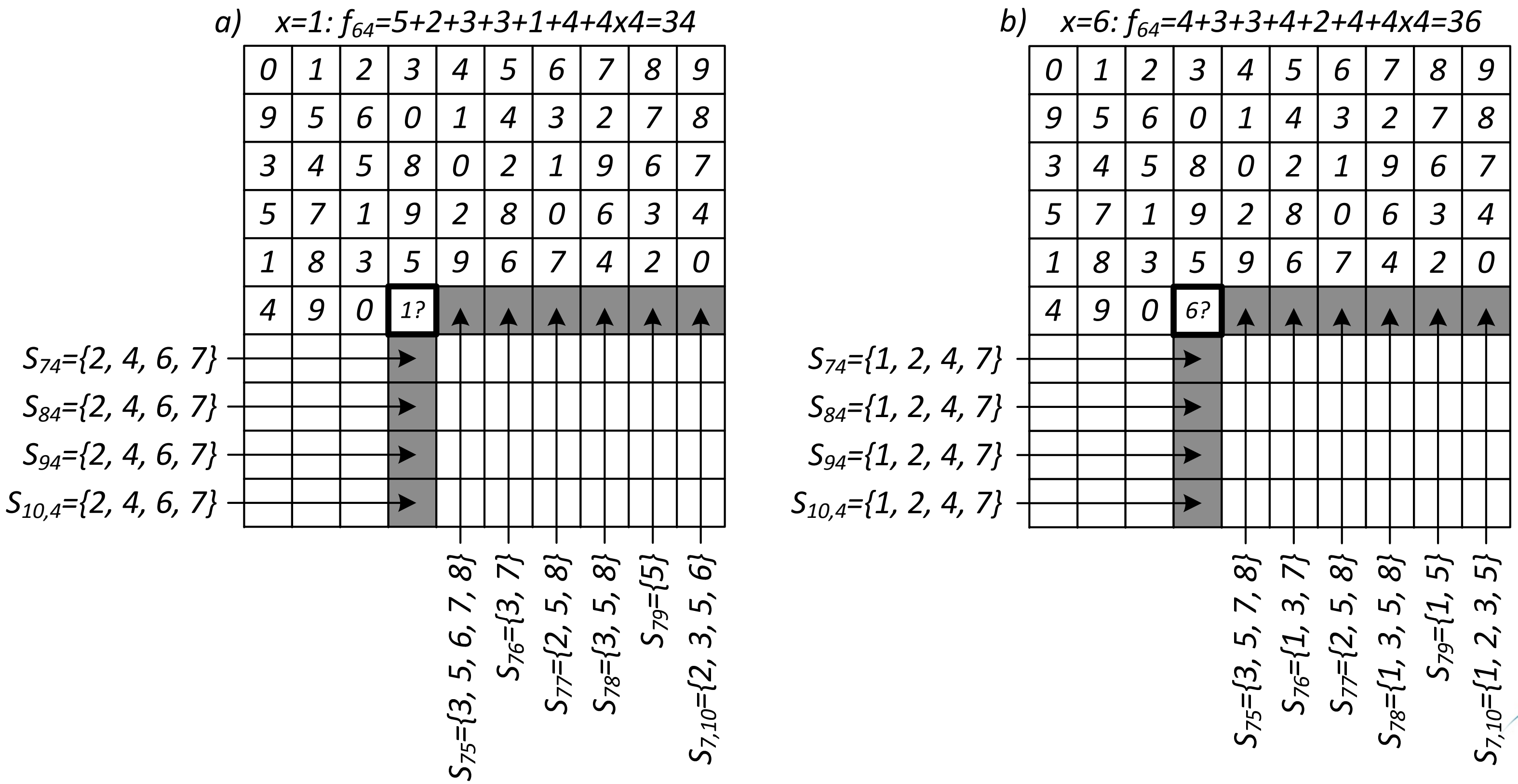
Abilities estimation

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

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

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

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



Getting Latin squares: greedy approach (G)

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

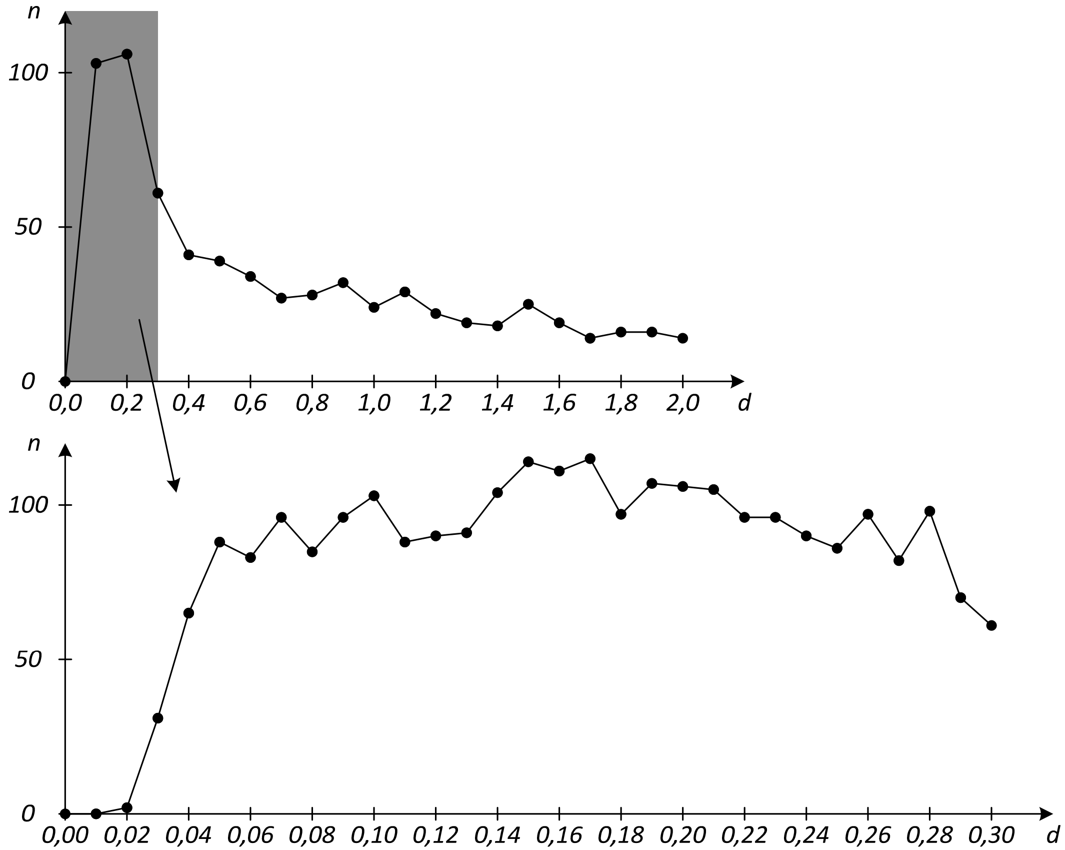
N	The number of violations
3	2
4	3
5	4
6	5
7	5
8	8
9	12
10	12

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

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

Getting Latin squares: weighted random search approach (WRS)

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



$$C(A)\leq 1$$

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)

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

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

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

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

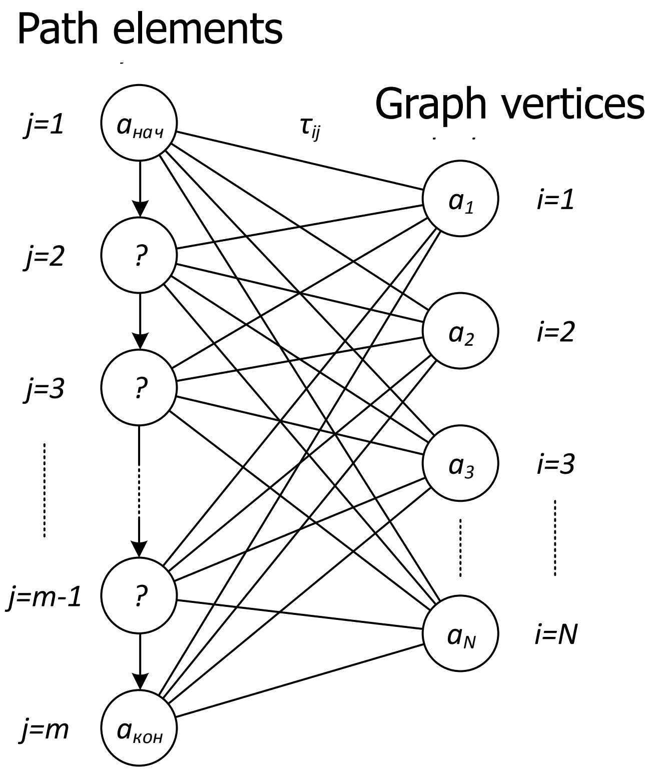
$\alpha^* = 100,$

$\beta^* = 1{,}1,$

$\tau_0^* = 1{,}0,$

$\gamma^* = 0{,}99999,$

$Q = 1{,}0$



$$C(A) \le 1$$

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)

$$L_1 = \begin{pmatrix}9 & 7 & 0 & 6 & 5 & ? & ? & ? & ? & ?\end{pmatrix} - 132\ ms\ per\ decision$$

$$L_1 = \begin{pmatrix}2 & 3 & 1 & 0 & 5 & ? & ? & ? & ? & ?\end{pmatrix} - 300\ ms\ per\ decision$$

AC preferences for first string:

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

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

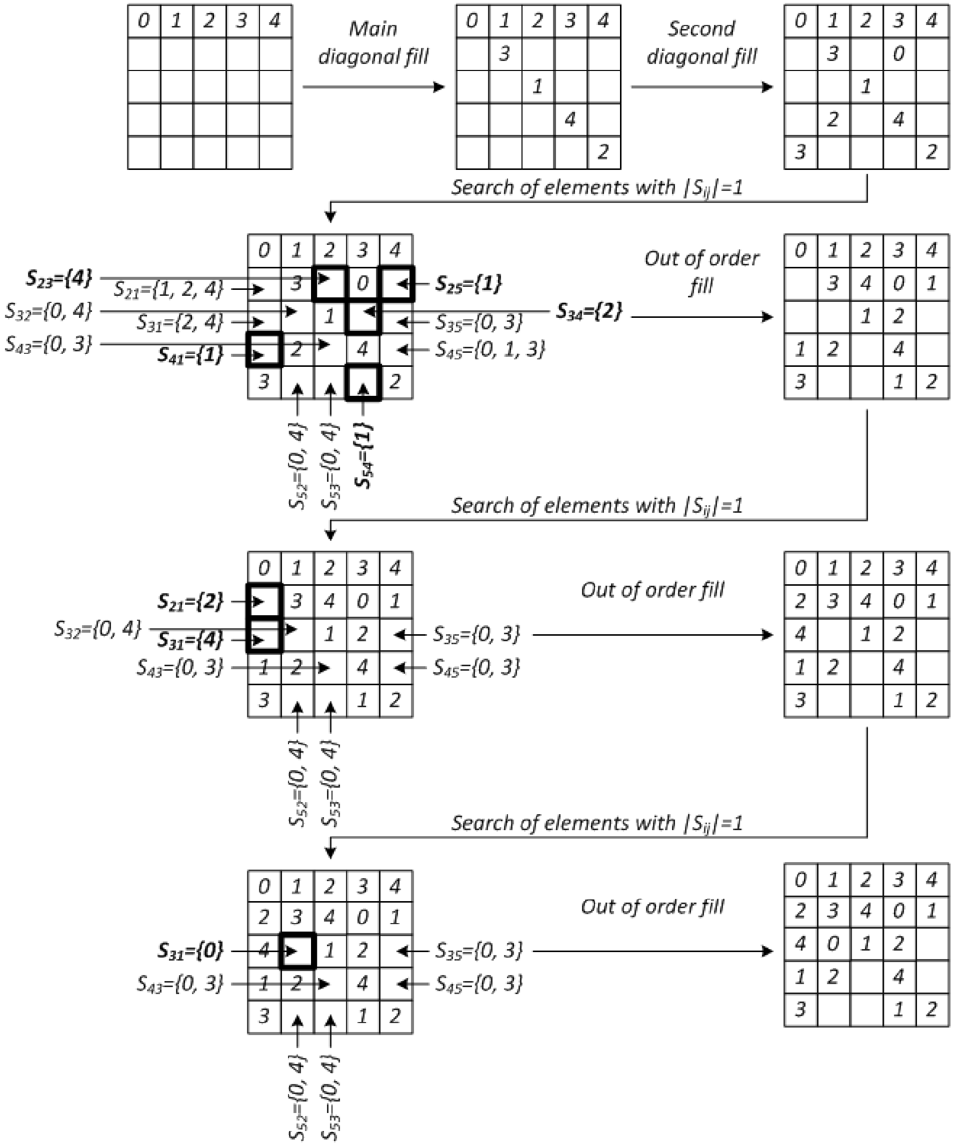
1	*2*	*3*	*4*	*5*
13	*6*	*14*	*10*	*15*
16	*17*	*7*	*18*	*19*
20	*11*	*21*	*8*	*22*
12	*23*	*24*	*25*	*9*

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

Diagonals first fill with out of order fill

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



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

Diagonals first fill with out of order fill

$$\left|S_{ij}\right| = 1$$

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

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

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

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)

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

a)

0	1	2	3	4
3	2		1	
		3		
	0		4	
2				1

b)

0	1	2	3	4
3	2		1	0
	4	3		2
1	0		4	
2				1

c)

0	1	2	3	4
3	2		1	0
	4	3		2
1	0	—	4	
2				1

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

```
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

Conclusion and prospects of future work

1. Current pace is **1 800 000 DLS/s**. This is end? GPU?
2. We can enumerate DLS and MOLS (LS is enumerated: OEIS A000479, A000315).
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. This problems are weakly coupled and can be solved with volunteer computing support!

- 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) // Proceedinds of Southwest State University. Accepted for publication
- Vatutin E.I., Zaikin O.S., Zhuravlev A.D., Manzuk M.O., Kochemazov S.E., Titov V.S. On the effect of the order of cells filling to the rate of generation of diagonal Latin squares // Diagnostics 2016. Accepted for publication

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