Computing technology for huge-scale optimization problems

A.Yu. Gornov, A.S. Anikin,
T.S. Zarodnyuk, P.S. Sorokovikov

Matrosov Institute for System Dynamics and Control Theory of SB RAS, Irkutsk, Russia
gornov@icc.ru

Russian National Conference MMPR-2019

November 26–29, 2019, Moscow, Russia

Algorithms for the solution of huge quasiseparable optimization problems

- Morse potential optimization;

- Keating potential optimization;

- Huge-Scale separable convex optimization problem;

- PageRank problem.

Morse potential optimization

A **cluster** is a structure consisting of a finite number of atoms or molecules. Occupies an intermediate position between the individual particles and the bulk material.

The interaction between the elements of the clusters described various potential functions defined multidimensional potential energy surface.

Finding the minima of the potential allows to obtain stable atomic-molecular configurations.

Such simulation in some cases replaces the field experiments.

The Cambridge Energy Landscape Database (The Cambridge Cluster Database): http://www-wales.ch.cam.ac.uk/CCD.html

Кластер – структура, состоящая из конечного числа атомов или молекул. Занимают промежуточное положение между отдельными частицами и объемным веществом.

Взаимодействие между элементами таких кластеров описывается различными функциями потенциалов, задающих (многомерные) поверхности потенциальной энергии (ППЭ).

Нахождение минимумов (стационарных точек) таких потенциалов (поверхностей) позволяет получать устойчивые атомно-молекулярные конфигурации.

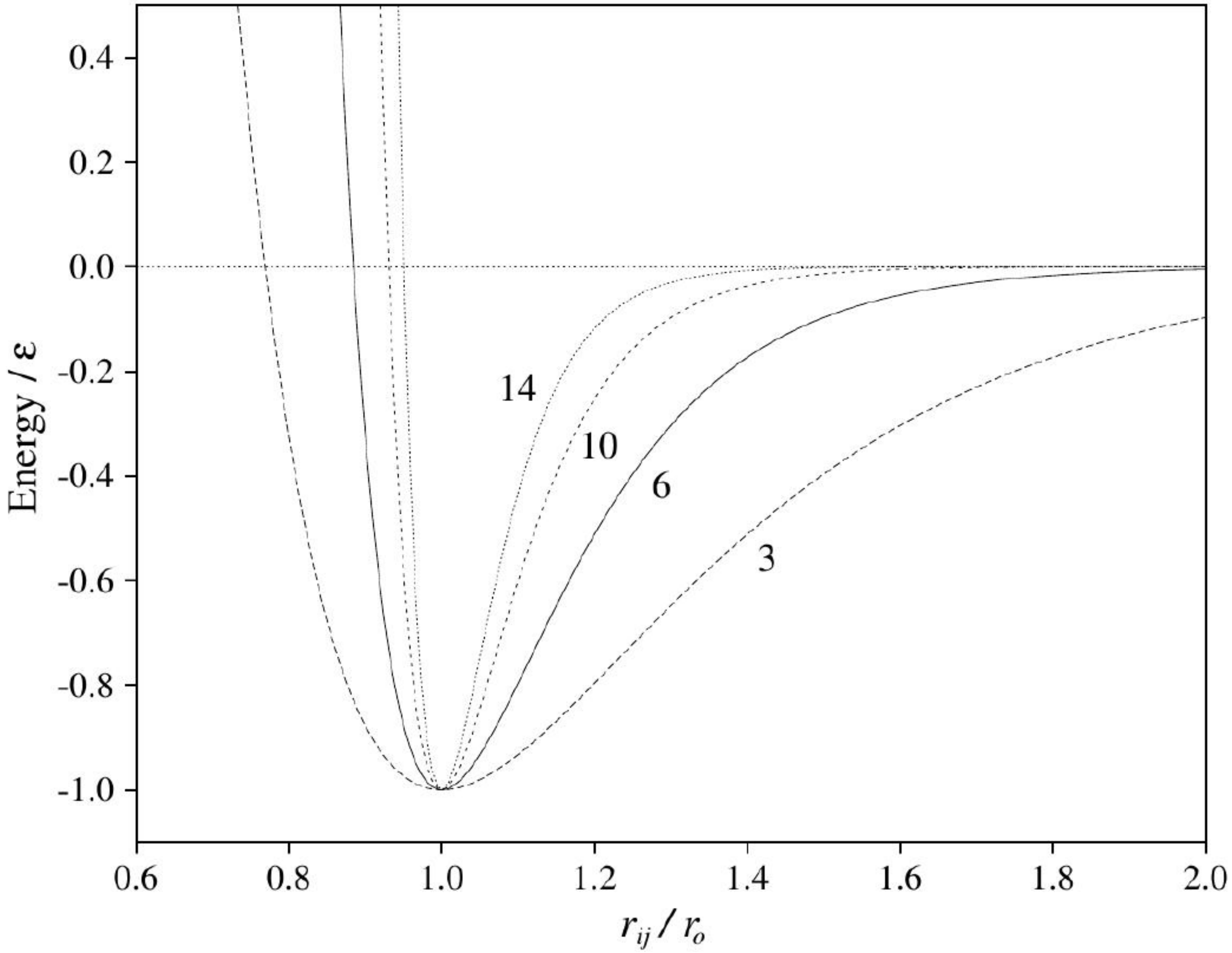
Подобное моделирование в ряде ситуаций заменяет натурные (физические) эксперименты.

Morse potential function

$$V_M = \sum_{i=1}^{n} \sum_{j>i}^{n} \left((e^{\rho_0 (1 - r_{ij})} - 1)^2 - 1 \right)$$

Morse potential function

with different ρ values



Morse potential optimization

- A global optimization problem.

- An astronomical number of local extrema. For example, for a cluster of 147 atoms experts provide estimates of the order of 10^{60}.

- The current state: “large clusters”, consisting of more than 200 atoms (600 variables).

- Задача глобальной оптимизации.

- Астрономические число локальных экстремумов. Например, для кластера из 147 атомов эксперты дают оценки порядка 10^{60}.

- Современное состояние задачи – «большие кластеры», состоящие более чем из 200 атомов (600 переменных).

Morse potential optimization

The Cambridge Cluster Database
D.J. Wales, J.P.K. Doye, A. Dullweber, M.P. Hodges, F.Y. Naumkin, F. Calvo, J. Hernandes-Rojas and T.F. Middleton.
www-wales.ch.cam.ac.uk/CCD.html

Hefei National Laboratory for Physical Sciences
at the Microscale and School of Life Sciences,
University of Science and Technology of China.
staff.ustc.edu.cn/~clj

Jorge Marques
Department of Chemistry Research
in Computational Chemistry and
Molecular Modeling
University of Coimbra, Portugal.

Applied optimization methods

Local optimization techniques

The main (universal) methods

- Conjugate Gradient;

- L-BFGS;

Additional methods

- Cauchy;

- Powell;

Special methods

- Polyak;

Global optimization techniques

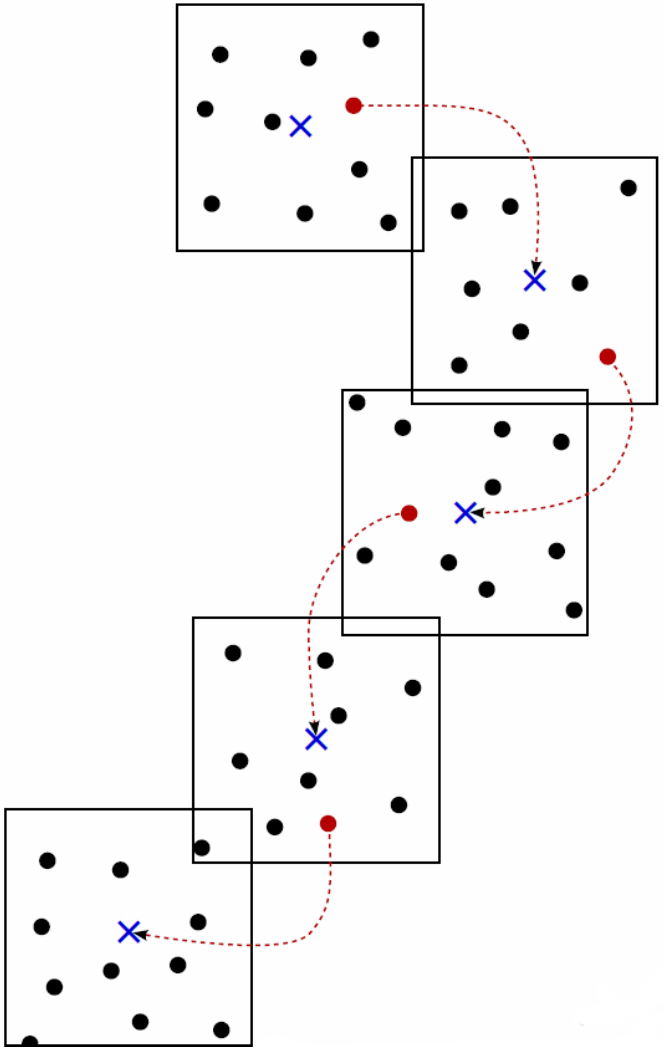
- Multi-start;

- MSBH-Monotonic Sequential Basin-Hopping;

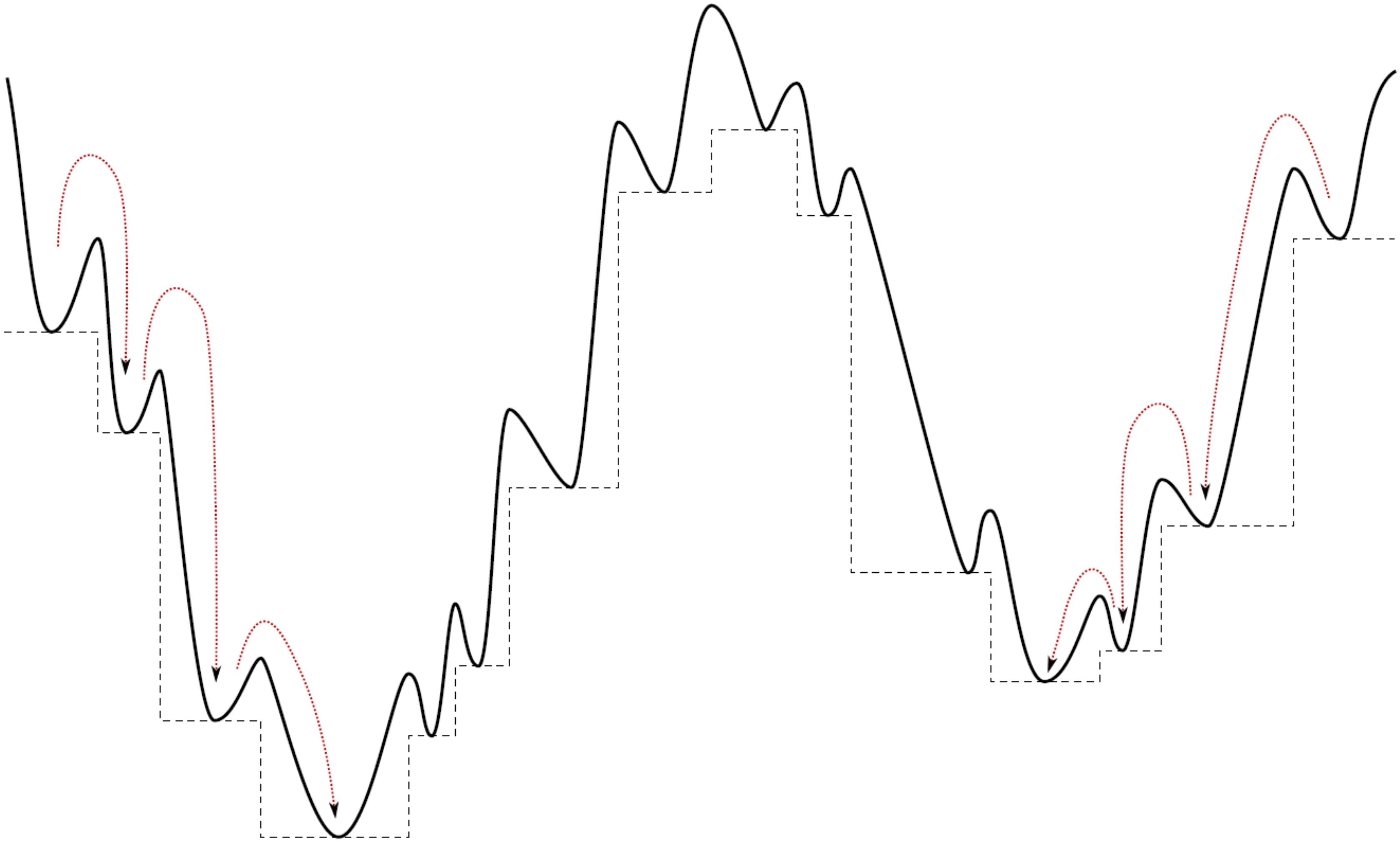
-“Big-Bang”;

-“Forest”

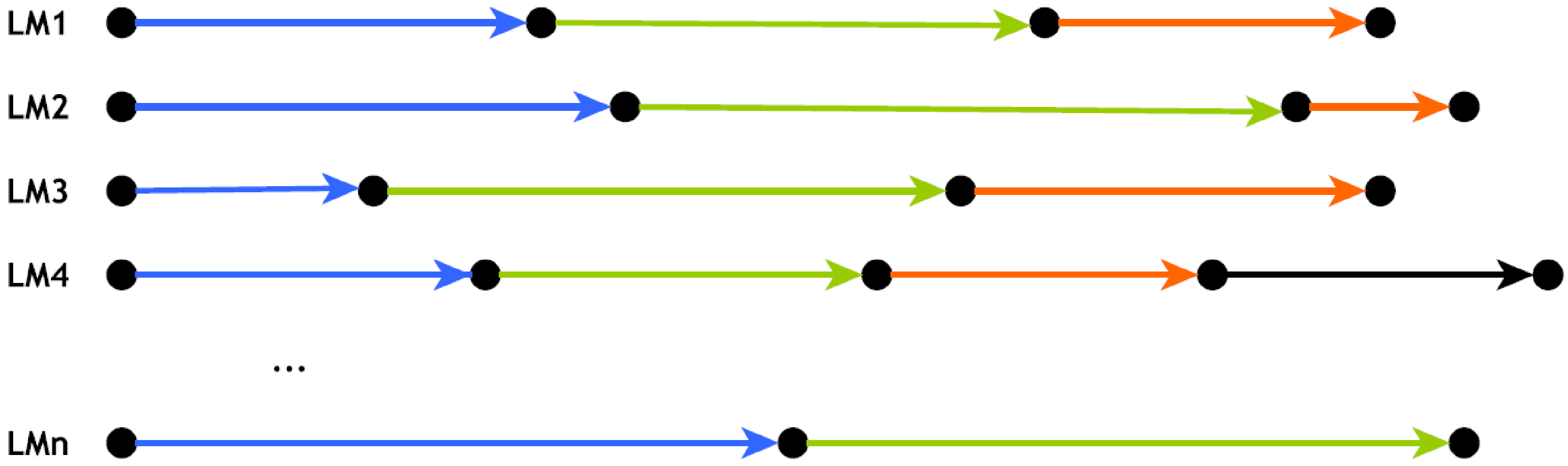
MSBH method



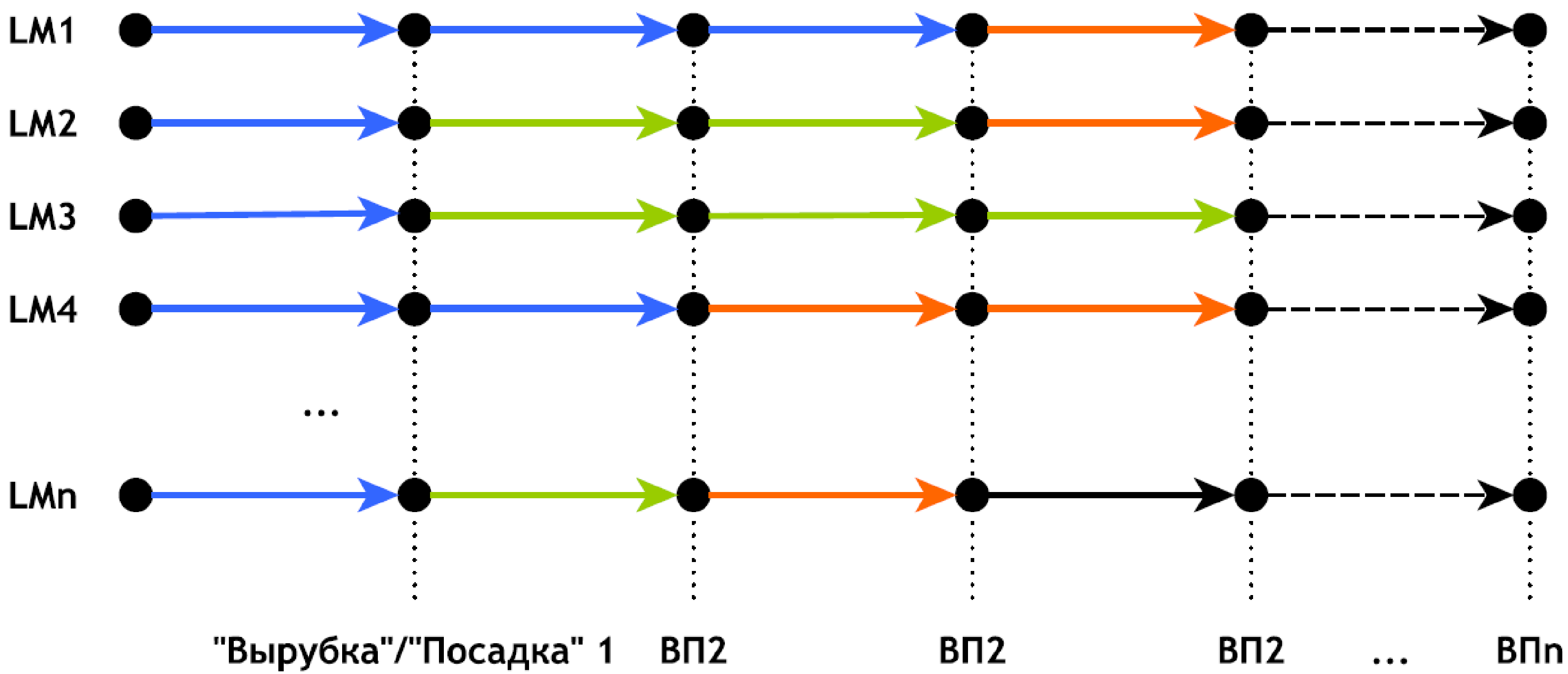
MSBH method



Forest method



Forest method



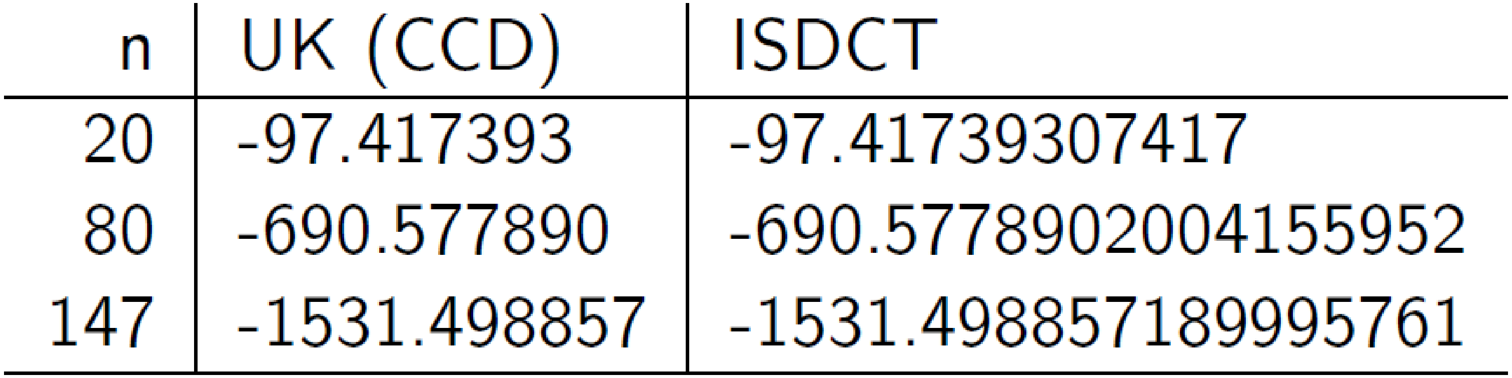
Критерии “вырубки” (рестарта) локального спуска:

- Достижение стационарной точки – проверка нормы градиента и т.д.
- Время работы – рестарт “слишком старых” ветвей
- Близость к другому экземпляру (локальному спуску)
- “Успешность” работы – рестарт экземпляров, имеющих слишком высокое значение оптимизируемой функции
- Изначально разрабатывается для параллельной реализации
- Локальные спуски разбиваются на участки, с фиксированным временем работы (“кванты”)
- Простая синхронизация
- Может быть реализован на аппаратных платформах типа GPGPU (Nvidia CUDA, OpenCL, ...)

Criteria for “cutting down” (a restart) of a local descent

- Achieving a stationary point - checking the gradient norm etc.
- Operating time – restart of old branches
- Nearness to another local descent
- Work success – restarting local descents that have too much optimized function value
- Initially developed for parallel implementation
- Local descents are divided into sections with a fixed operating time
- Simple synchronization
- Can be implemented on hardware platforms like GPGPU (Nvidia GUDA, OpenCL, …)

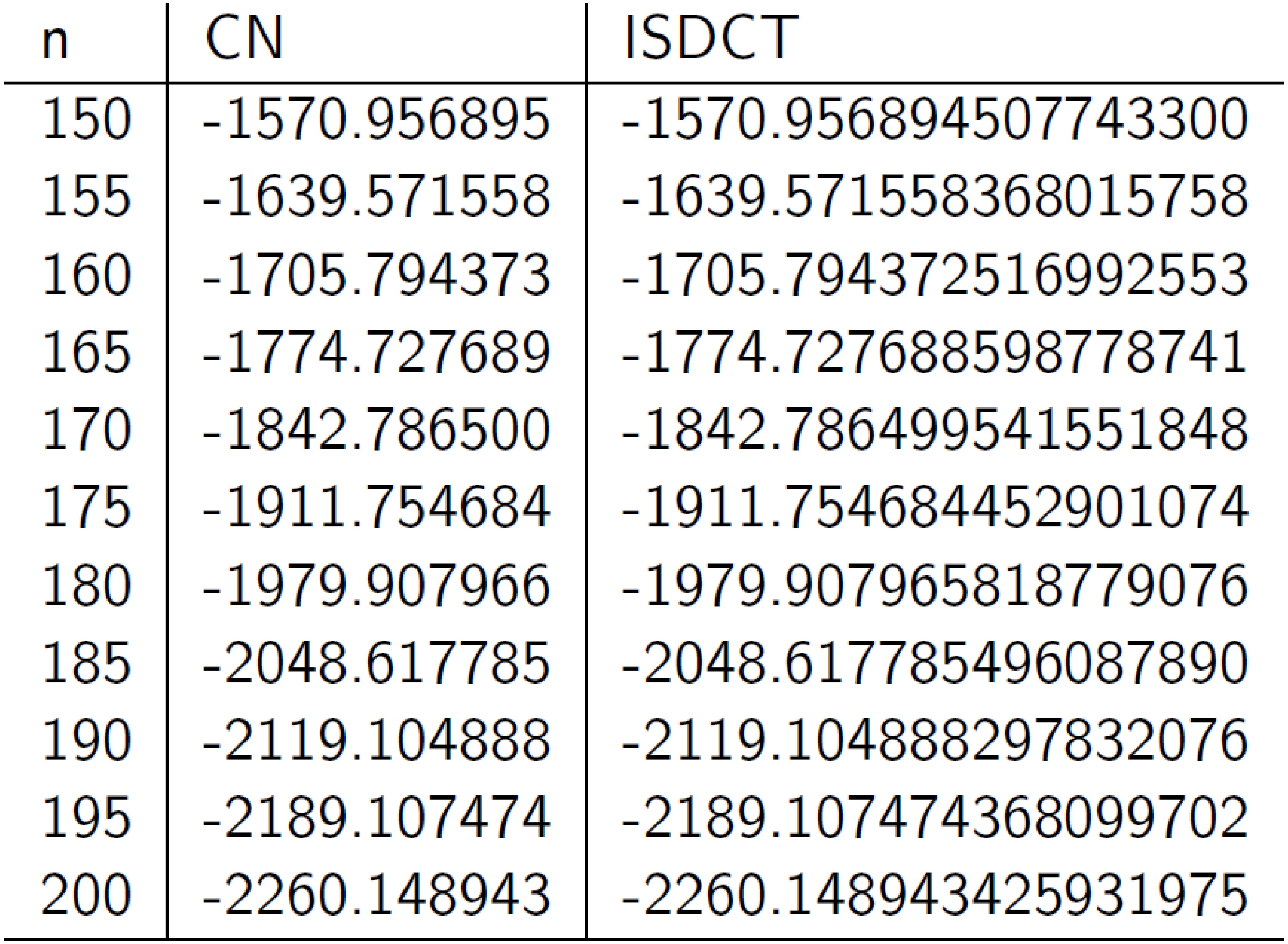
Computing Experiments MSBH/Forest



n	UK (CCD)	ISDCT
20	-97.417393	-97.41739307417
80	-690.577890	-690.5778902004155952
147	-1531.498857	-1531.49885718995761

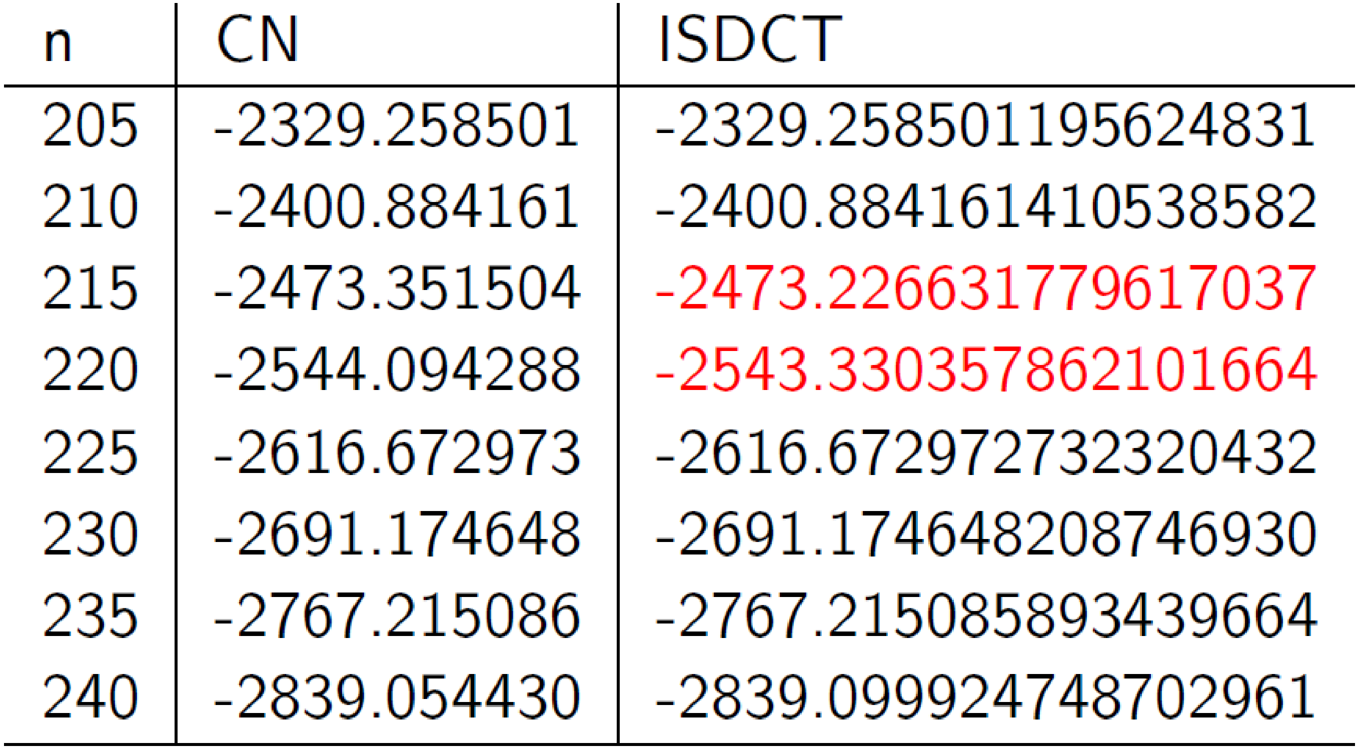
n is a number of atoms.

Computing Experiments MSBH/Forest



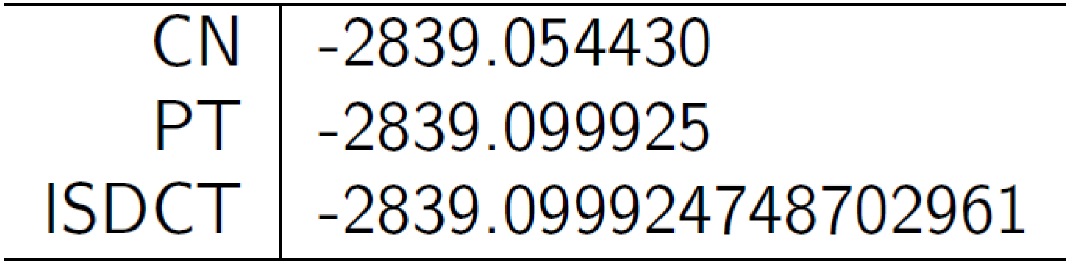
n	CN	ISDCT
150	-1570.956895	-1570.956894507743300
155	-1639.571558	-1639.571558368015758
160	-1705.794373	-1705.794372516992553
165	-1774.727689	-1774.727688598778741
170	-1842.786500	-1842.786499541551848
175	-1911.754684	-1911.754684452901074
180	-1979.907966	-1979.907965818779076
185	-2048.617785	-2048.617785496087890
190	-2119.104888	-2119.104888297832076
195	-2189.107474	-2189.107474368099702
200	-2260.148943	-2260.148943425931975

Computing Experiments MSBH/Forest



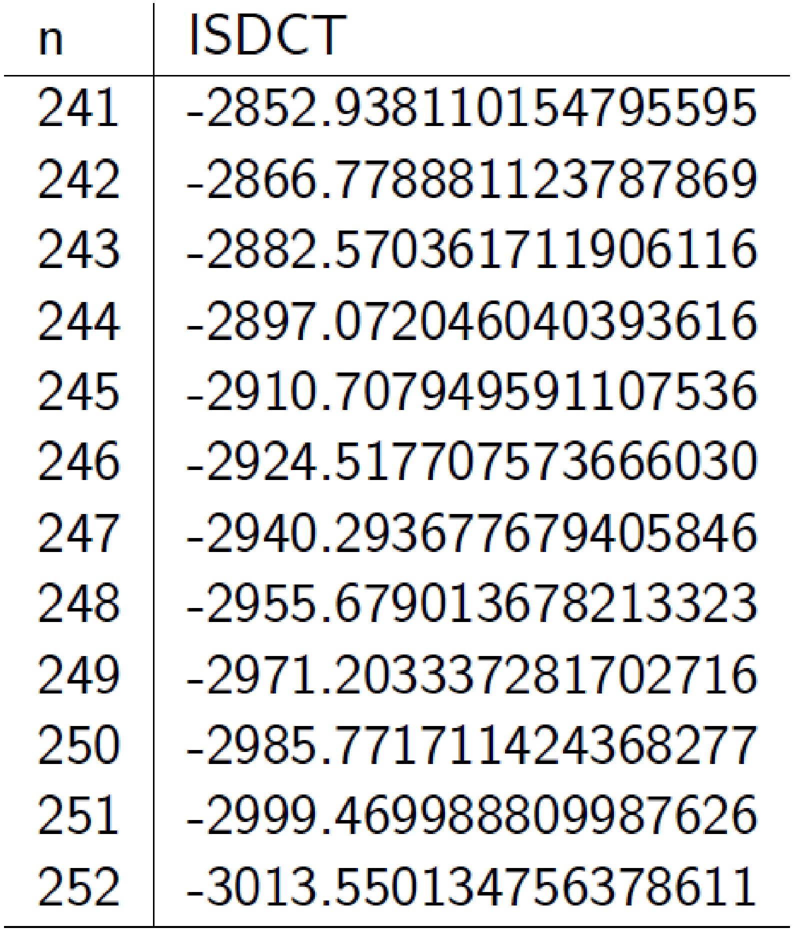
n	CN	ISDCT
205	-2329.258501	-2329.258501195624831
210	-2400.884161	-2400.884161410538582
215	-2473.351504	-2473.226631779617037
220	-2544.094288	-2543.330357862101664
225	-2616.672973	-2616.672972732320432
230	-2691.174648	-2691.174648208746930
235	-2767.215086	-2767.215085893439664
240	-2839.054430	-2839.099924748702961

Computing Experiments MSBH/Forest, n = 240



CN	-2839.054430
PT	-2839.099925
ISDCT	-2839.09924748702961

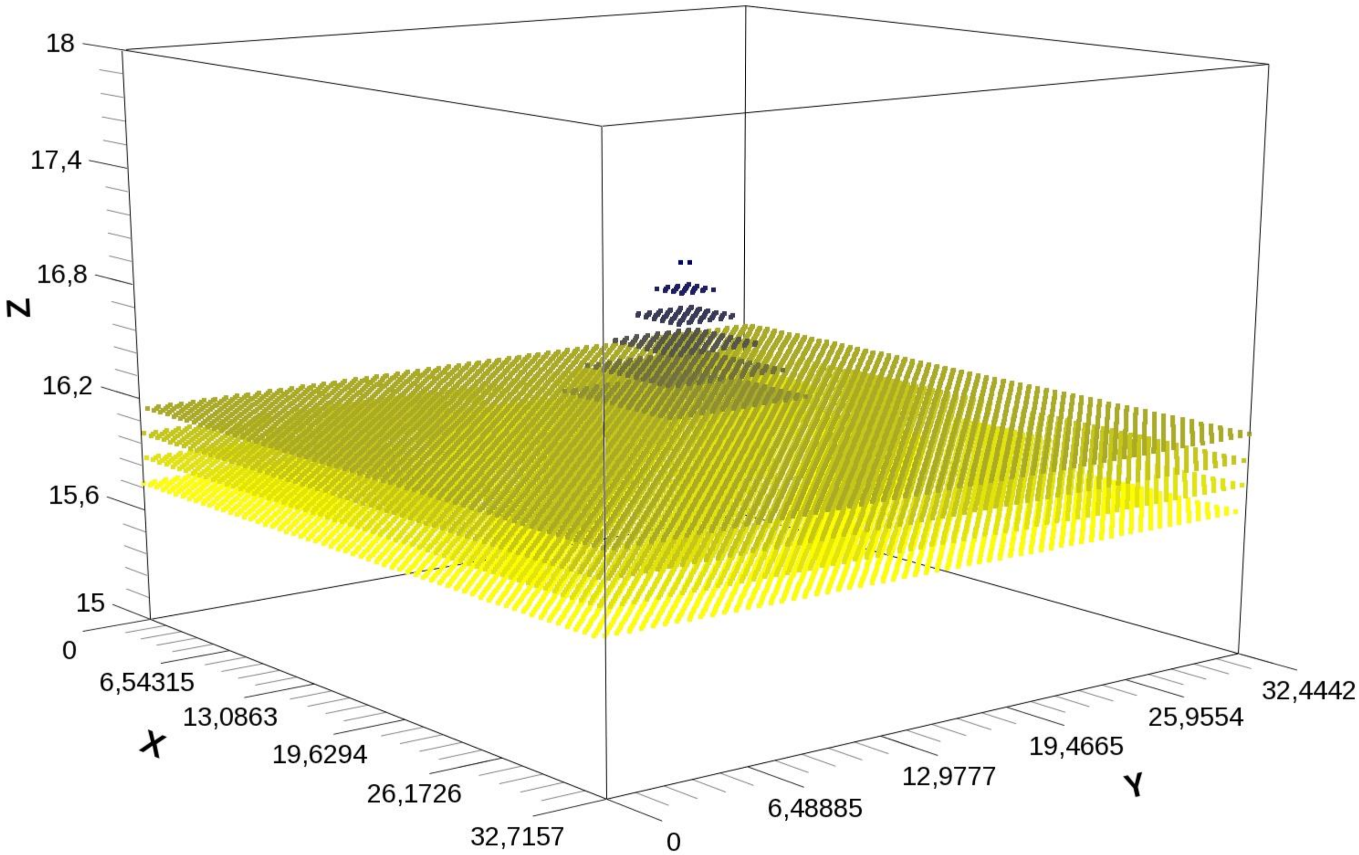
Computing Experiments
Forest



n	ISDCT
241	-2852.93811015479559 5
242	-2866.77888112378786 9
243	-2882.57036171190611 6
244	-2897.07204604039361 6
245	-2910.70794959110753 6
246	-2924.51770757366603 0
247	-2940.29367767940584 6
248	-2955.67901367821332 3
249	-2971.20333728170271 6
250	-2985.77171142436827 7
251	-2999.46998880998762 6
252	-3013.55013475637861 1

Keating potential optimization.

Form of a quantum dot Si-Ge



Keating potential function

$$E = \sum_{i=1}^{n} \left[\frac{3}{16} \sum_{j=1}^{4} \frac{\alpha_{ij}}{d_{ij}^2} \left\{ ||r_i - r_j||_2^2 - d_{ij}^2 \right\}^2 + \right.$$

$$\left. + \frac{3}{8} \sum_{j=1}^{4} \sum_{k=j+1}^{4} \frac{\beta_{ijk}}{d_{ij} \cdot d_{ik}} \left\{ \langle r_i - r_j, r_i - r_k \rangle + \frac{d_{ij} \cdot d_{ik}}{3} \right\}^2 \right]$$

n is the number of atoms in the crystal lattice;

- D_{ij}, d_{ik}, α_{ij}, β_{ijk} are constants set;
- $R_i = (x_i, x_{2i}, x_{3i})$ radius vector of the *i*-th node (optimized variables).

Features of the problem

- High dimensionality of 105 variables and more
- The high demands on the result accuracy

Tested optimization methods

- Cauchy method
- Conjugate Gradient Method
- Newton's Method

Difficulties with Newton's method implementation

The dimension of these problems depends of physical limits of the Hessian matrix size which flows from the available memory.

The high computational complexity due to the required long time for solving problem of such dimension

Hessian matrix

$$\begin{vmatrix} \frac{\partial f}{\partial x_1 \partial x_1} & \frac{\partial f}{\partial x_1 \partial x_2} & \frac{\partial f}{\partial x_1 \partial x_3} & \cdots & \frac{\partial f}{\partial x_1 \partial x_n} \\ \frac{\partial f}{\partial x_2 \partial x_1} & \frac{\partial f}{\partial x_2 \partial x_2} & \frac{\partial f}{\partial x_2 \partial x_3} & \cdots & \frac{\partial f}{\partial x_2 \partial x_n} \\ \frac{\partial f}{\partial x_3 \partial x_1} & \frac{\partial f}{\partial x_3 \partial x_2} & \frac{\partial f}{\partial x_3 \partial x_3} & \cdots & \frac{\partial f}{\partial x_3 \partial x_n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial x_n \partial x_1} & \frac{\partial f}{\partial x_n \partial x_2} & \frac{\partial f}{\partial x_n \partial x_3} & \cdots & \frac{\partial f}{\partial x_n \partial x_n} \end{vmatrix}$$

Storage of a dense matrix requires about *n*^{2} memory cells.

Sparse Hessian matrix

$$\begin{vmatrix} \frac{\partial f}{\partial x_1 \partial x_1} & \frac{\partial f}{\partial x_1 \partial x_2} & 0 & \cdots & \frac{\partial f}{\partial x_1 \partial x_n} \\ \frac{\partial f}{\partial x_2 \partial x_1} & \frac{\partial f}{\partial x_2 \partial x_2} & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial x_n \partial x_1} & 0 & 0 & \cdots & \frac{\partial f}{\partial x_n \partial x_n} \end{vmatrix}$$

Storage of a sparse matrix requires less then n^2 memory cells.

Methods of sparse matrix storage

- Diagonal scheme for storing circuit tape matrices,
- Profile storage scheme of symmetric matrices,
- Connected scheme of sparse storage,
- Sparse line format,

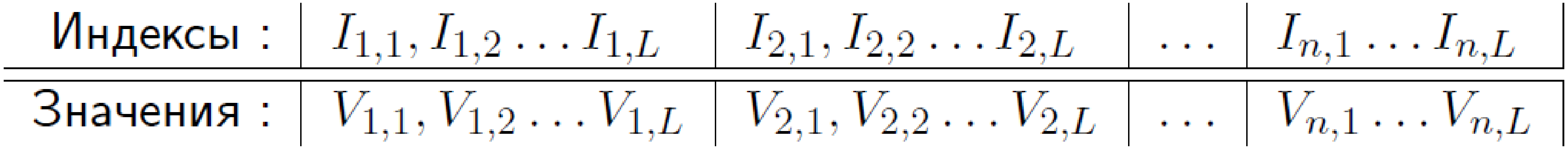
and a number of other methods, and various their modifications.

Методы хранения разреженных матриц

- Диагональная схема хранения ленточных матриц,
- Профильная схема хранения симметрических, матриц,
- Связные схемы разреженного хранения,
- Разреженный строчный формат,

а так же ряд других методов и различные их модификации.

Applied method of sparse matrix storage



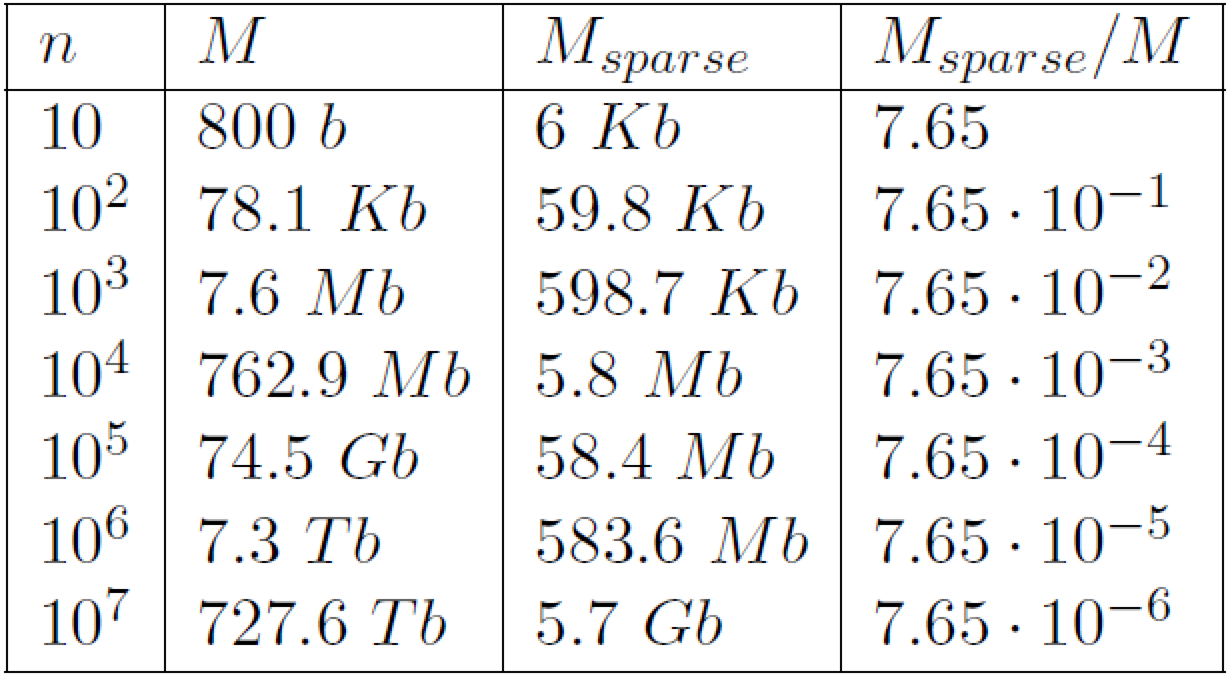
Индексы :	$I_{1,1}, I_{1,2} \dots I_{1,L}$	$I_{2,1}, I_{2,2} \dots I_{2,L}$	$\dots$	$I_{n,1} \dots I_{n,L}$
Значения :	$V_{1,1}, V_{1,2} \dots V_{1,L}$	$V_{2,1}, V_{2,2} \dots V_{2,L}$	$\dots$	$V_{n,1} \dots V_{n,L}$

L is a maximum number of nonzero elements in the Hessian line. For considered problem L = 51.

$$I_{i,1} = i$$

$$V_{j,k} = \frac{\partial f}{\partial x_j \partial x_m}, \, m = I_{j,k}$$

The ratio of dense and sparse matrices size



n	M	M_{sparse}	M_{sparse}/M
10	$800\ b$	$6\ Kb$	7.65
10^2	$78.1\ Kb$	$59.8\ Kb$	$7.65 \cdot 10^{-1}$
10^3	$7.6\ Mb$	$598.7\ Kb$	$7.65 \cdot 10^{-2}$
10^4	$762.9\ Mb$	$5.8\ Mb$	$7.65 \cdot 10^{-3}$
10^5	$74.5\ Gb$	$58.4\ Mb$	$7.65 \cdot 10^{-4}$
10^6	$7.3\ Tb$	$583.6\ Mb$	$7.65 \cdot 10^{-5}$
10^7	$727.6\ Tb$	$5.7\ Gb$	$7.65 \cdot 10^{-6}$

$M = 8\,n^2$ the size of a dense array (in bytes).

$M_{sparse} = 51\,(8 + 4)\,n$ the size of a sparse matrix (in bytes).

Features of used storage format

Selected format has a number of positive features:

- a fixed amount of memory used order $2{\cdot}L{\cdot}n$ cells

- a small number of occupied cells, about 4-5 % when the dimension of the problem is 10^5

- a quick access to the elements of the main diagonal

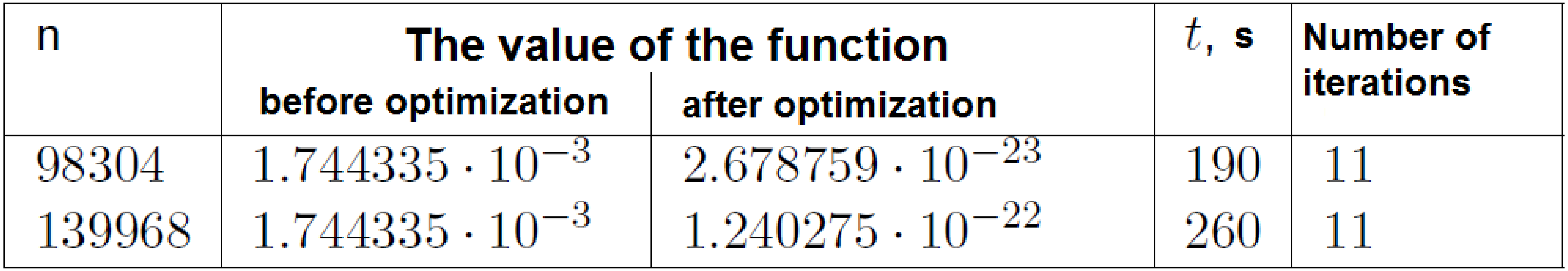
- ease of implementation procedures for sparse matrix multiplication
- on a tight vector

Выбраный формат имеет ряд положительных особенностей :

- Фиксированный размер используемой памяти порядка $2 \cdot L \cdot n$ ячеек
- Малое число незанятых ячеек –
  4 - 5% при размерности задачи 10^5
- Быстрый доступ к элементам главной диагонали
- Простота реализации процедуры умножения разреженной матрицы на плотный вектор

Computing experiments

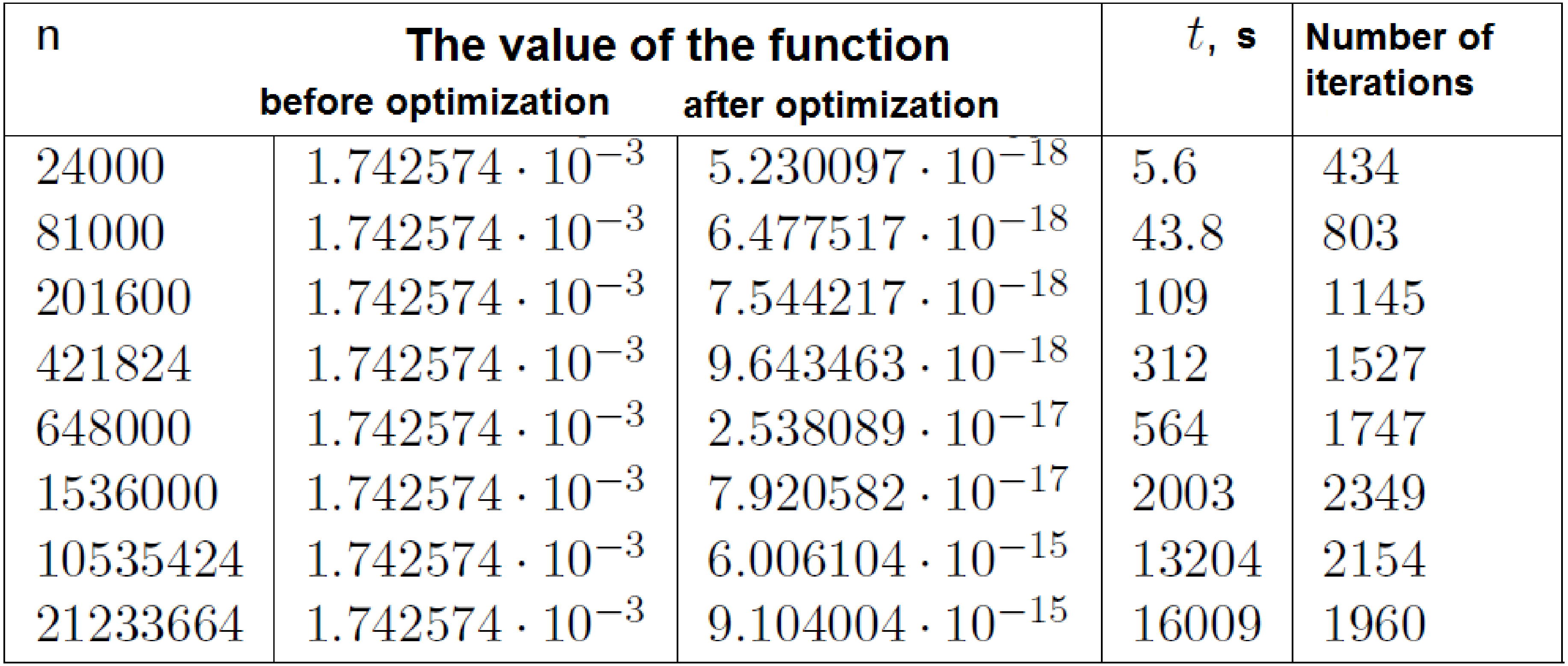
Newton's method modification



n	The value of the function		t, s	Number of iterations
	before optimization	after optimization		
98304	$1.744335 \cdot 10^{-3}$	$2.678759 \cdot 10^{-23}$	190	11
139968	$1.744335 \cdot 10^{-3}$	$1.240275 \cdot 10^{-22}$	260	11

Computing experiments

Conjugate gradient method



n	The value of the function		t, s	Number of iterations
	before optimization	after optimization		
24000	$1.742574 \cdot 10^{-3}$	$5.230097 \cdot 10^{-18}$	5.6	434
81000	$1.742574 \cdot 10^{-3}$	$6.477517 \cdot 10^{-18}$	43.8	803
201600	$1.742574 \cdot 10^{-3}$	$7.544217 \cdot 10^{-18}$	109	1145
421824	$1.742574 \cdot 10^{-3}$	$9.643463 \cdot 10^{-18}$	312	1527
648000	$1.742574 \cdot 10^{-3}$	$2.538089 \cdot 10^{-17}$	564	1747
1536000	$1.742574 \cdot 10^{-3}$	$7.920582 \cdot 10^{-17}$	2003	2349
10535424	$1.742574 \cdot 10^{-3}$	$6.006104 \cdot 10^{-15}$	13204	2154
21233664	$1.742574 \cdot 10^{-3}$	$9.104004 \cdot 10^{-15}$	16009	1960

Tests were carried out on a computer system containing 10 cores Intel Xeon X5670

Huge-Scale separable convex optimization problem

The classification of local optimization problems on the number of variables proposed by the Yu.E. Nesterov:

- “Small” – up to 100 variables
- “Medium” – from 10^3 to 10^4 variables
- “Large” – from 10^5 to 10^7 variables
- “Huge” – more than 10^8 variables

Huge-Scale separable convex optimization problem

Difficulties

- the number of variables – memory limitations
- the computational complexity – time limit
- the required amount of computation – time limit

↓

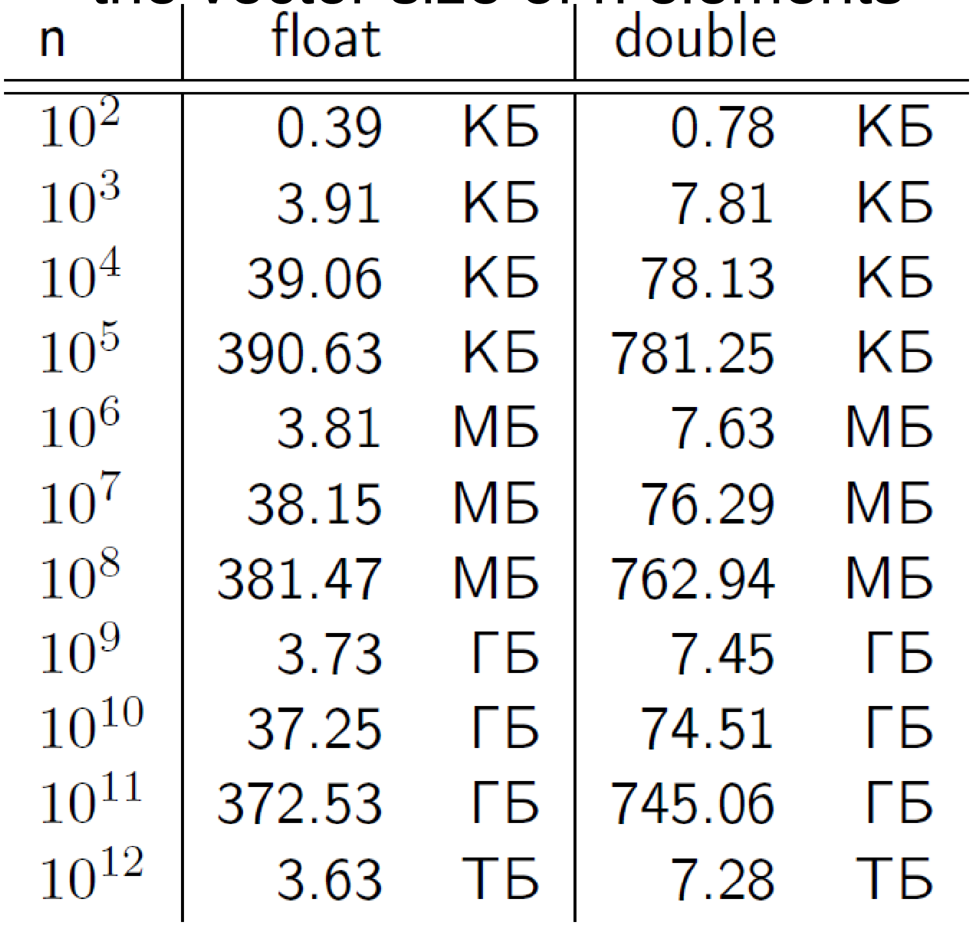
parallel computing

Required memory

- float - 4 bytes per cell
- double - 8 bytes per cell

Required memory

the vector size of n elements



n	float		double	
10^{2}	0.39	КБ	0.78	КБ
10^{3}	3.91	КБ	7.81	КБ
10^{4}	39.06	КБ	78.13	КБ
10^{5}	390.63	КБ	781.25	КБ
10^{6}	3.81	МБ	7.63	МБ
10^{7}	38.15	МБ	76.29	МБ
10^{8}	381.47	МБ	762.94	МБ
10^{9}	3.73	ГБ	7.45	ГБ
10^{10}	37.25	ГБ	74.51	ГБ
10^{11}	372.53	ГБ	745.06	ГБ
10^{12}	3.63	ТБ	7.28	ТБ

Memory (RAM) - the main hardware limitations for many modern Huge-Scale optimization problem.

Test optimization problem 1

$$f(x) = \sum_{i=1}^{n} (x_i^2 + x_i^6)$$

This test function is convex, separable, the minimum value is known ($f_{\min} = 0$).

The calculation of values of the function $f(x)$ and its gradient $\nabla f(x)$ performed in parallel on different CPU cores, for large-scale problems it is impossible for one compute node due to physical limitations on the amount of RAM.

Computing experiments

Computational experiments were carried out with

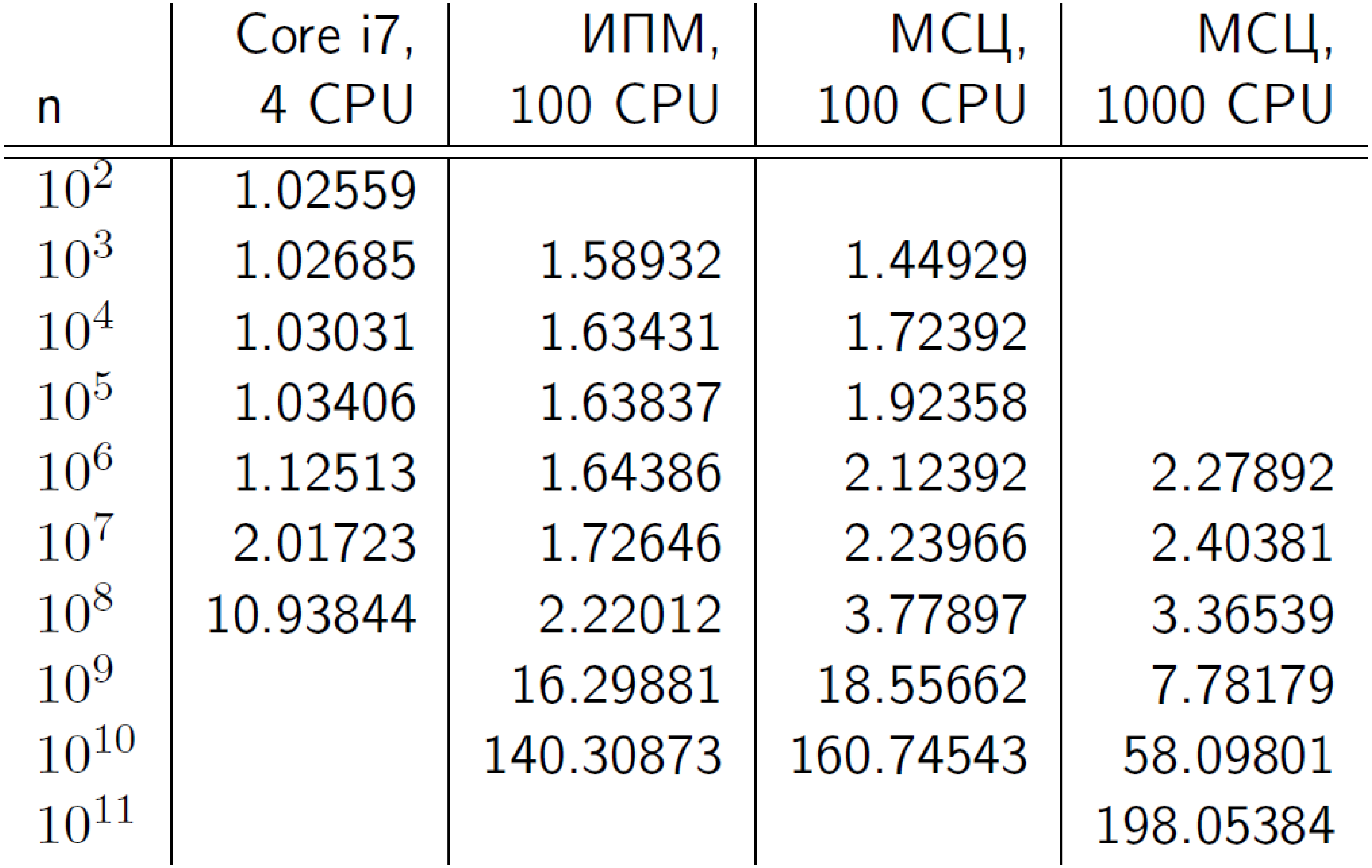
•Computing cluster MBC-100K of Interdepartmental Supercomputer Center.
RAM – 1 Gb, 1 CPU.

•Computing cluster MBC-100K of Keldysh Institute of Applied Mathematics RAS.

•Computing cluster “Academician V.M. Matrosov”, unit “Tesla”.
RAM – 250 Gb, 32 CPU.

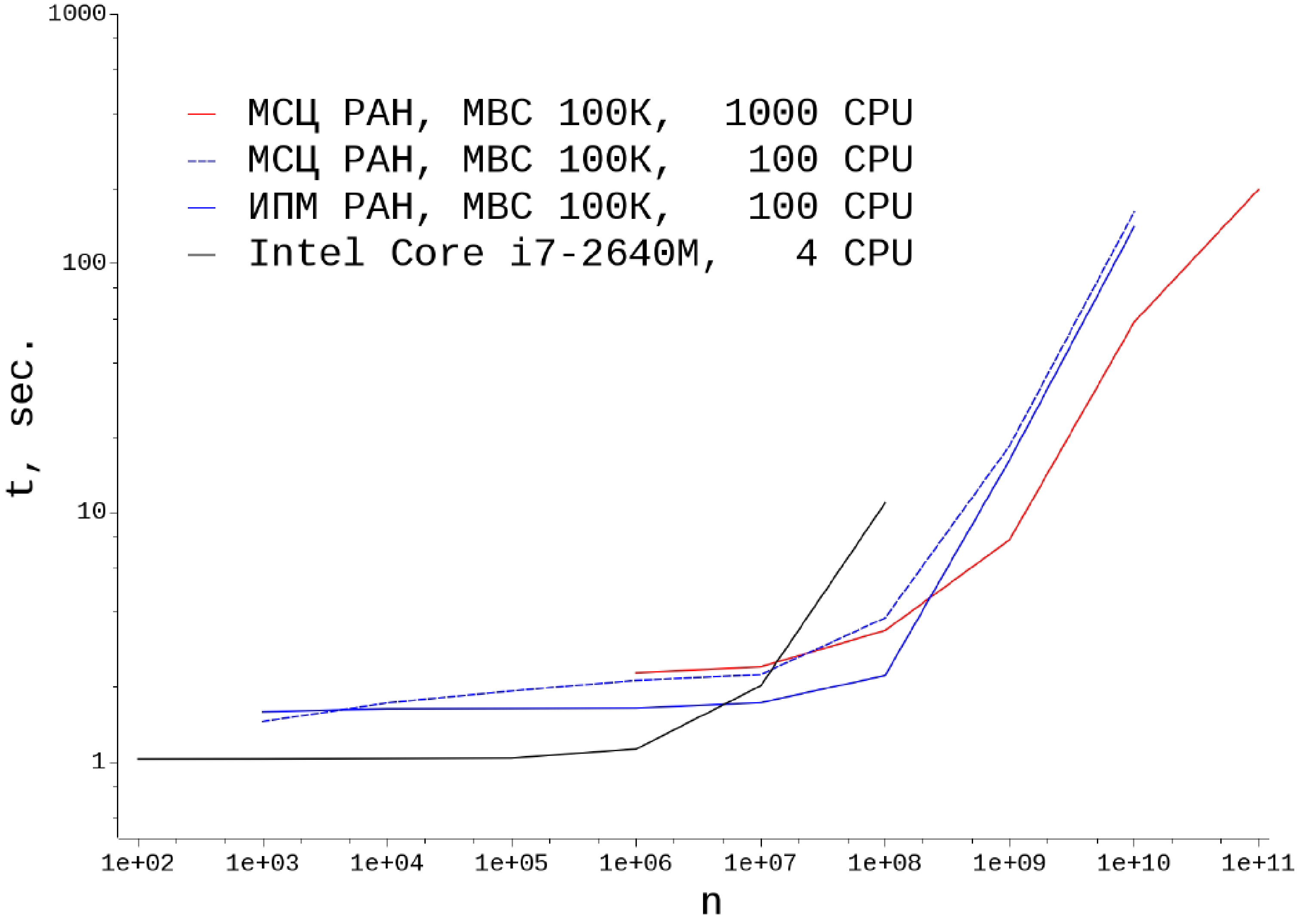
Computing experiments

the running time of algorithm

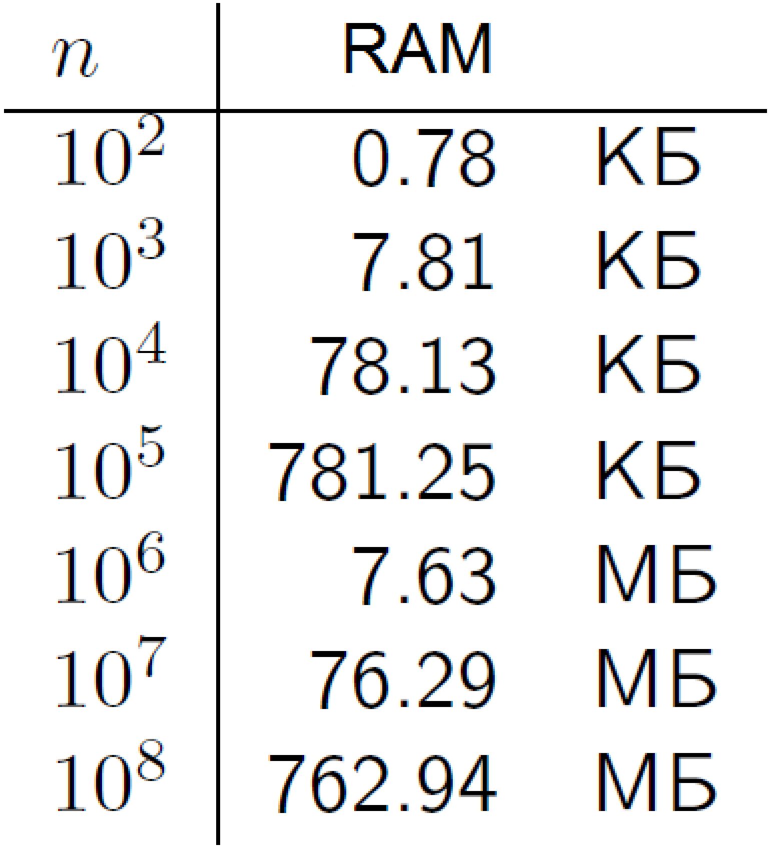


n	Core i7, 4 CPU	ИПМ, 100 CPU	МСЦ, 100 CPU	МСЦ, 1000 CPU
10^2	1.02559			
10^3	1.02685	1.58932	1.44929	
10^4	1.03031	1.63431	1.72392	
10^5	1.03406	1.63837	1.92358	
10^6	1.12513	1.64386	2.12392	2.27892
10^7	2.01723	1.72646	2.23966	2.40381
10^8	10.93844	2.22012	3.77897	3.36539
10^9		16.29881	18.55662	7.78179
10^{10}		140.30873	160.74543	58.09801
10^{11}				198.05384

Computing experiments

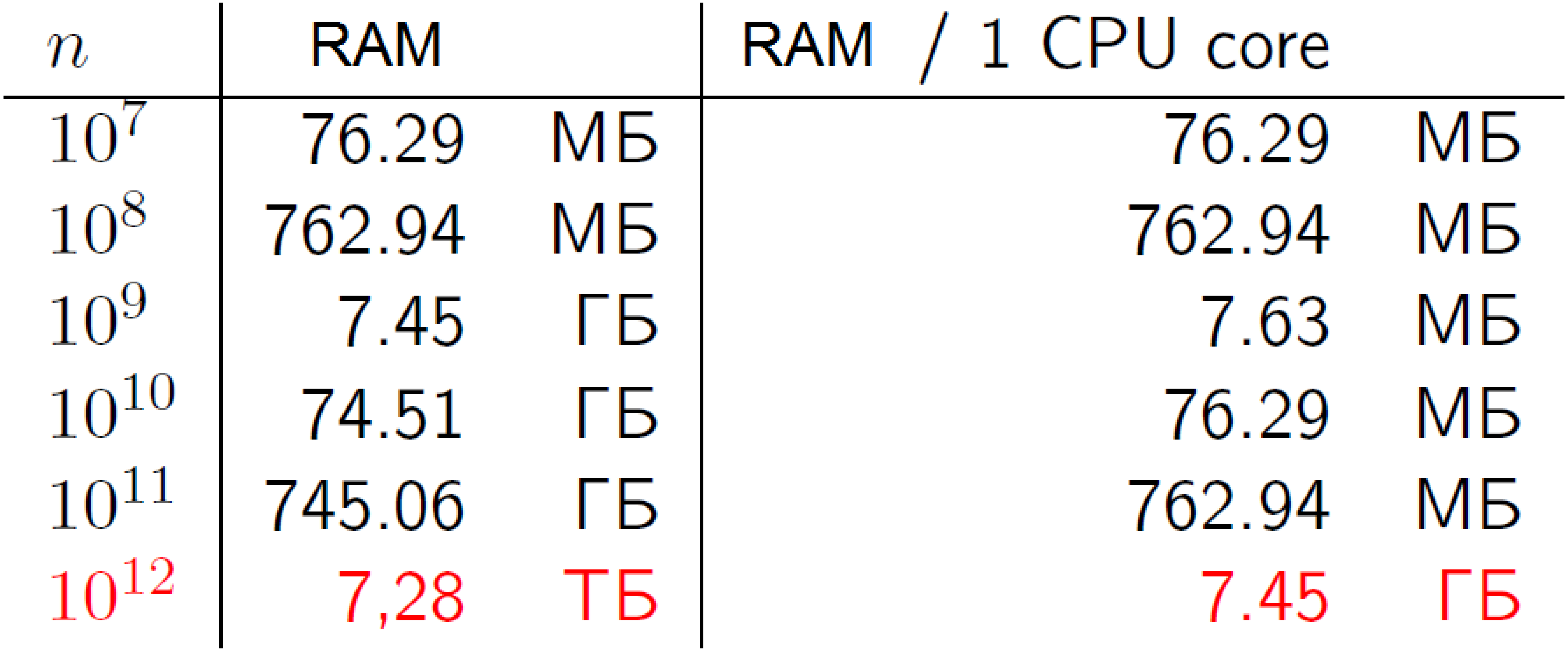


Required memory (1 core)



n	RAM	
10^2	0.78	КБ
10^3	7.81	КБ
10^4	78.13	КБ
10^5	781.25	КБ
10^6	7.63	МБ
10^7	76.29	МБ
10^8	762.94	МБ

Required memory (100 cores)



n	RAM		RAM / 1 CPU core	
10^7	76.29	МБ	76.29	МБ
10^8	762.94	МБ	762.94	МБ
10^9	7.45	ГБ	7.63	МБ
10^{10}	74.51	ГБ	76.29	МБ
10^{11}	745.06	ГБ	762.94	МБ
10^{12}	7,28	ТБ	7.45	ГБ

Computing experiments

- good scalability of the proposed implementation;

- the main limiting factor - the amount of available RAM;

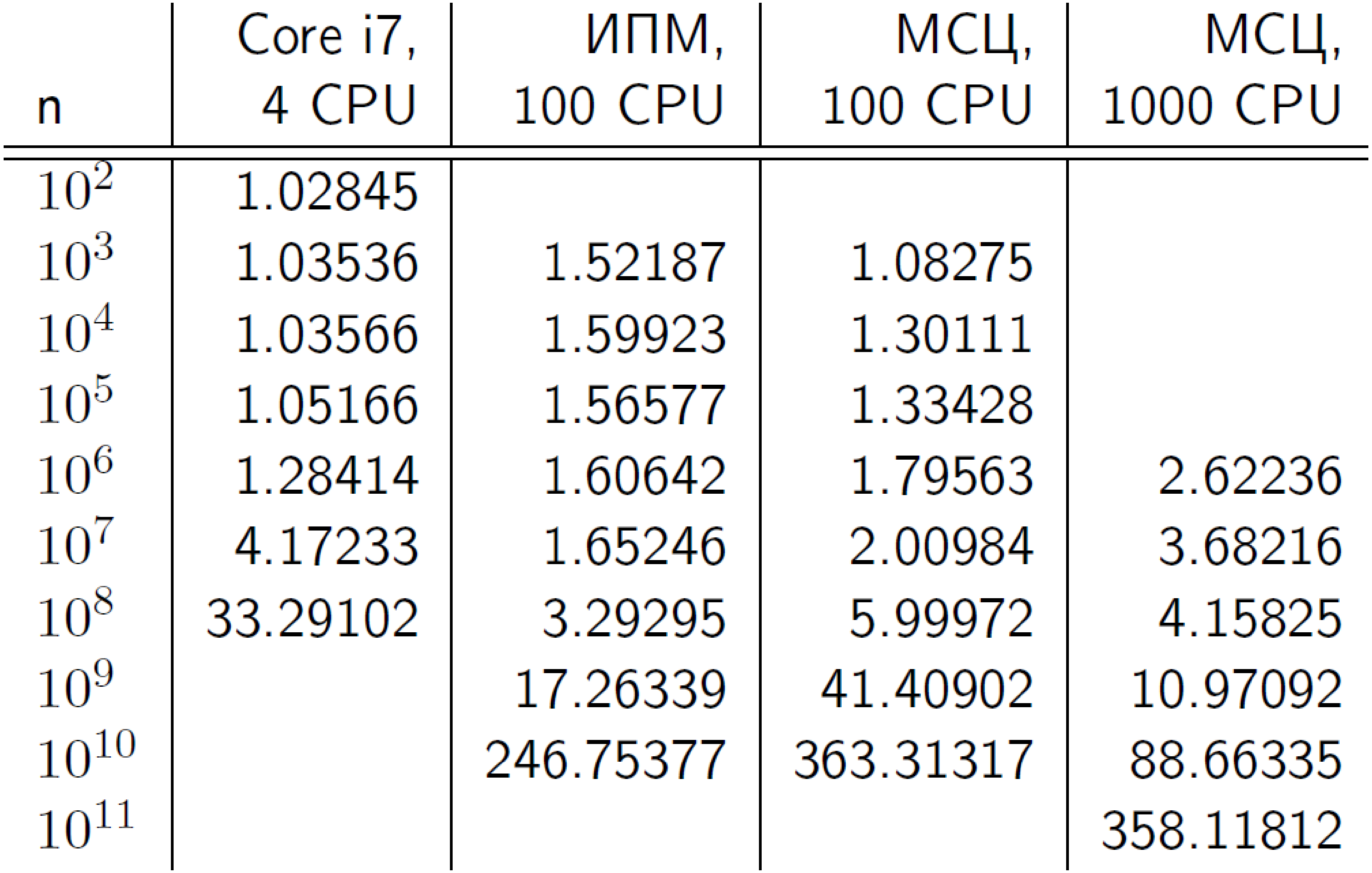
- High-performance for separable problems.

Test optimization problem 2

$$f(x) = \sum_{i=1}^{n} x_i^2 + \sum_{i=2}^{n} (x_i - x_{i-1})^2$$

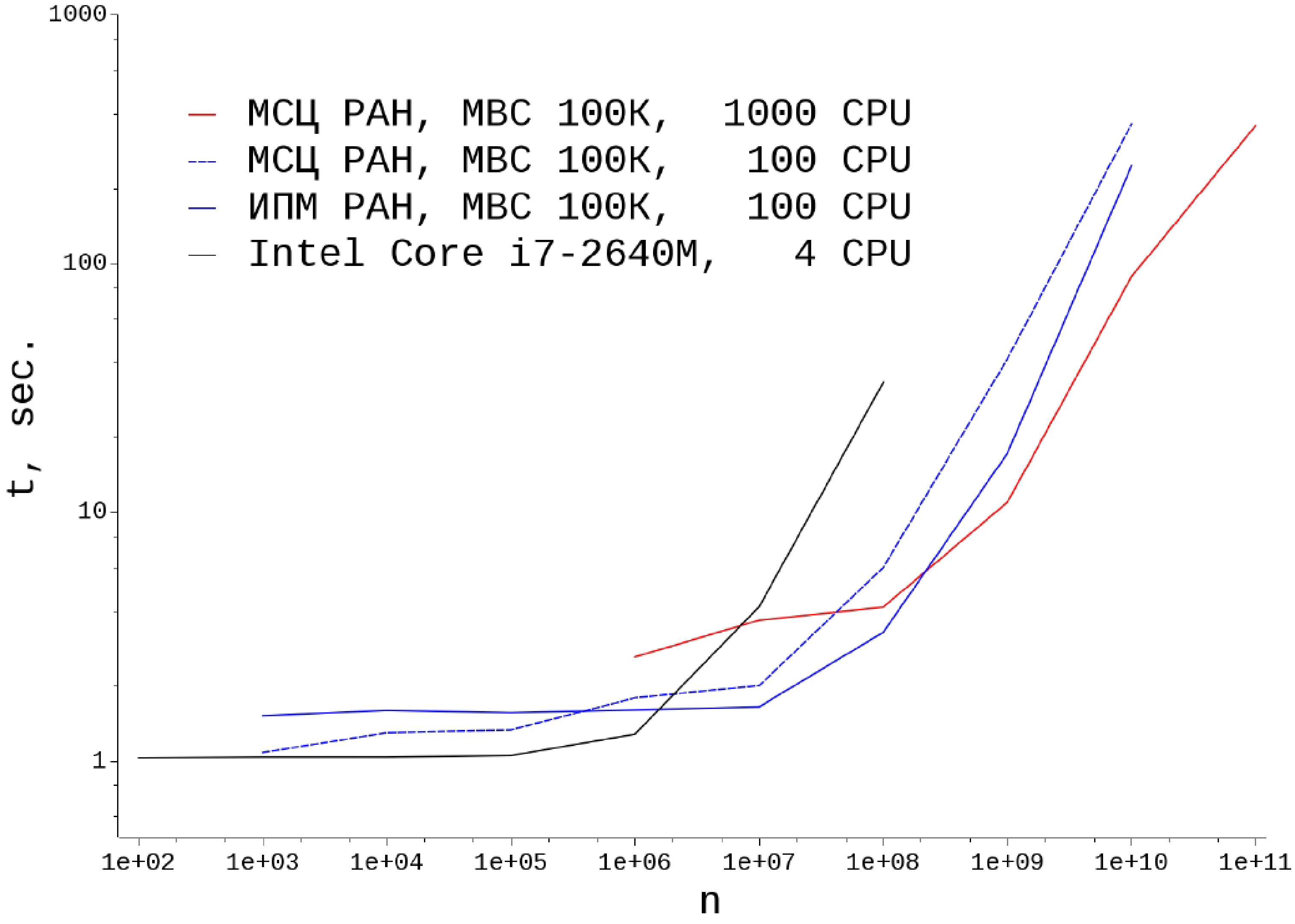
Computing experiments

the running time of algorithm



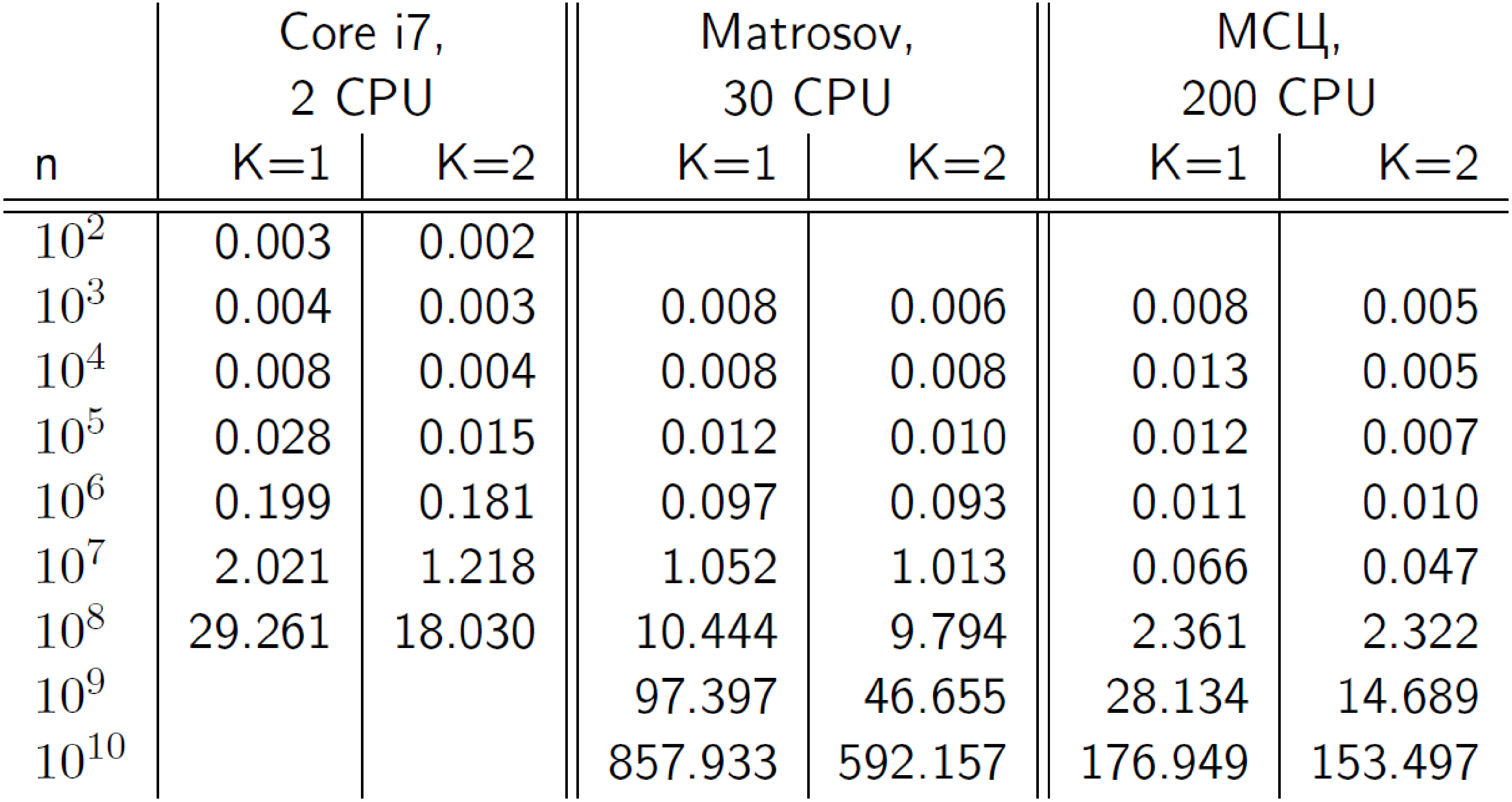
n	Core i7, 4 CPU	ИПМ, 100 CPU	МСЦ, 100 CPU	МСЦ, 1000 CPU
10^2	1.02845			
10^3	1.03536	1.52187	1.08275	
10^4	1.03566	1.59923	1.30111	
10^5	1.05166	1.56577	1.33428	
10^6	1.28414	1.60642	1.79563	2.62236
10^7	4.17233	1.65246	2.00984	3.68216
10^8	33.29102	3.29295	5.99972	4.15825
10^9		17.26339	41.40902	10.97092
10^{10}		246.75377	363.31317	88.66335
10^{11}				358.11812

Computing experiments



Computing experiments, test 2

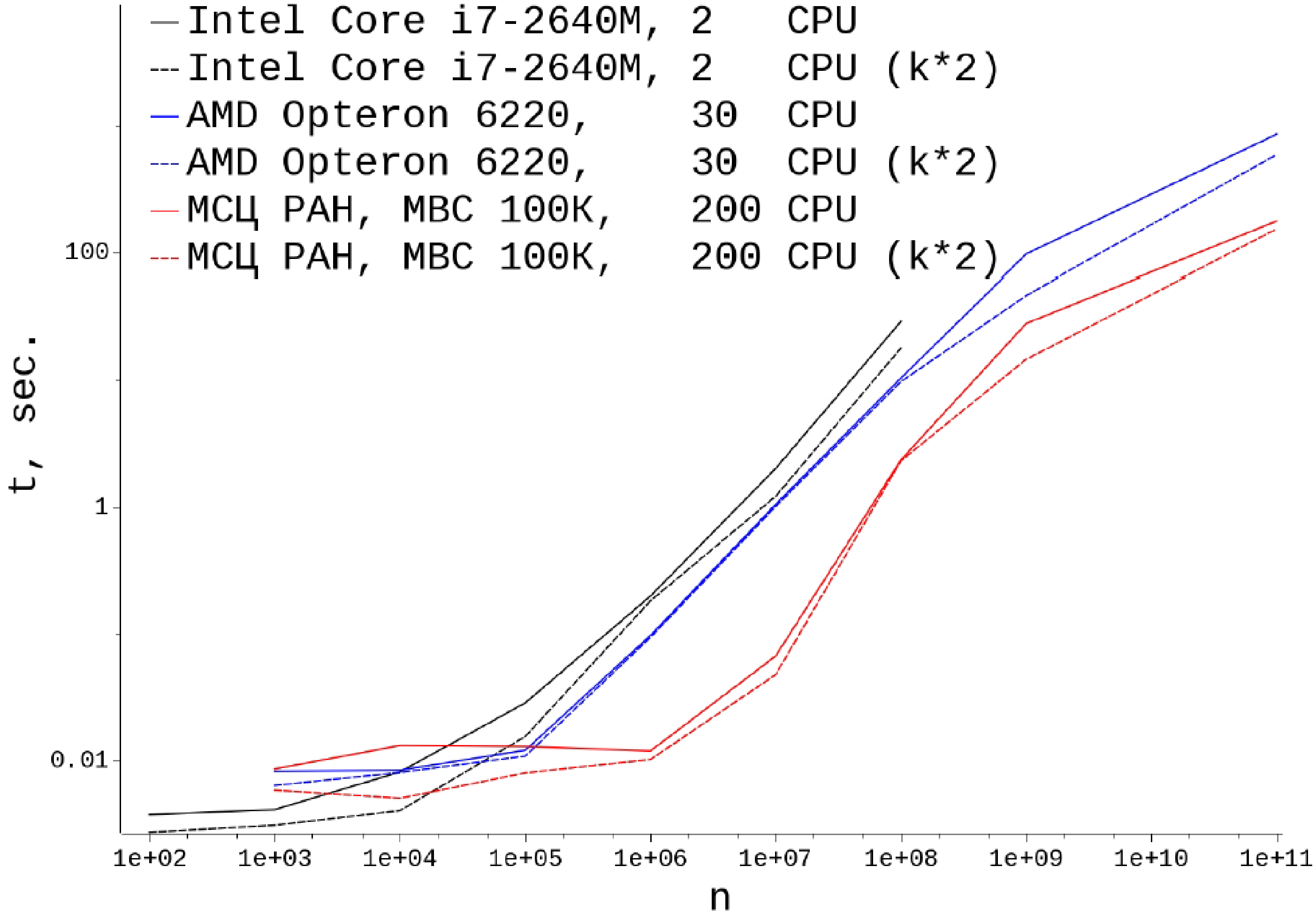
Modification of Polyak method, time (s)



	Core i7, 2 CPU		Matrosov, 30 CPU		МСЦ, 200 CPU	
n	K=1	K=2	K=1	K=2	K=1	K=2
10^2	0.003	0.002				
10^3	0.004	0.003	0.008	0.006	0.008	0.005
10^4	0.008	0.004	0.008	0.008	0.013	0.005
10^5	0.028	0.015	0.012	0.010	0.012	0.007
10^6	0.199	0.181	0.097	0.093	0.011	0.010
10^7	2.021	1.218	1.052	1.013	0.066	0.047
10^8	29.261	18.030	10.444	9.794	2.361	2.322
10^9			97.397	46.655	28.134	14.689
10^{10}			857.933	592.157	176.949	153.497

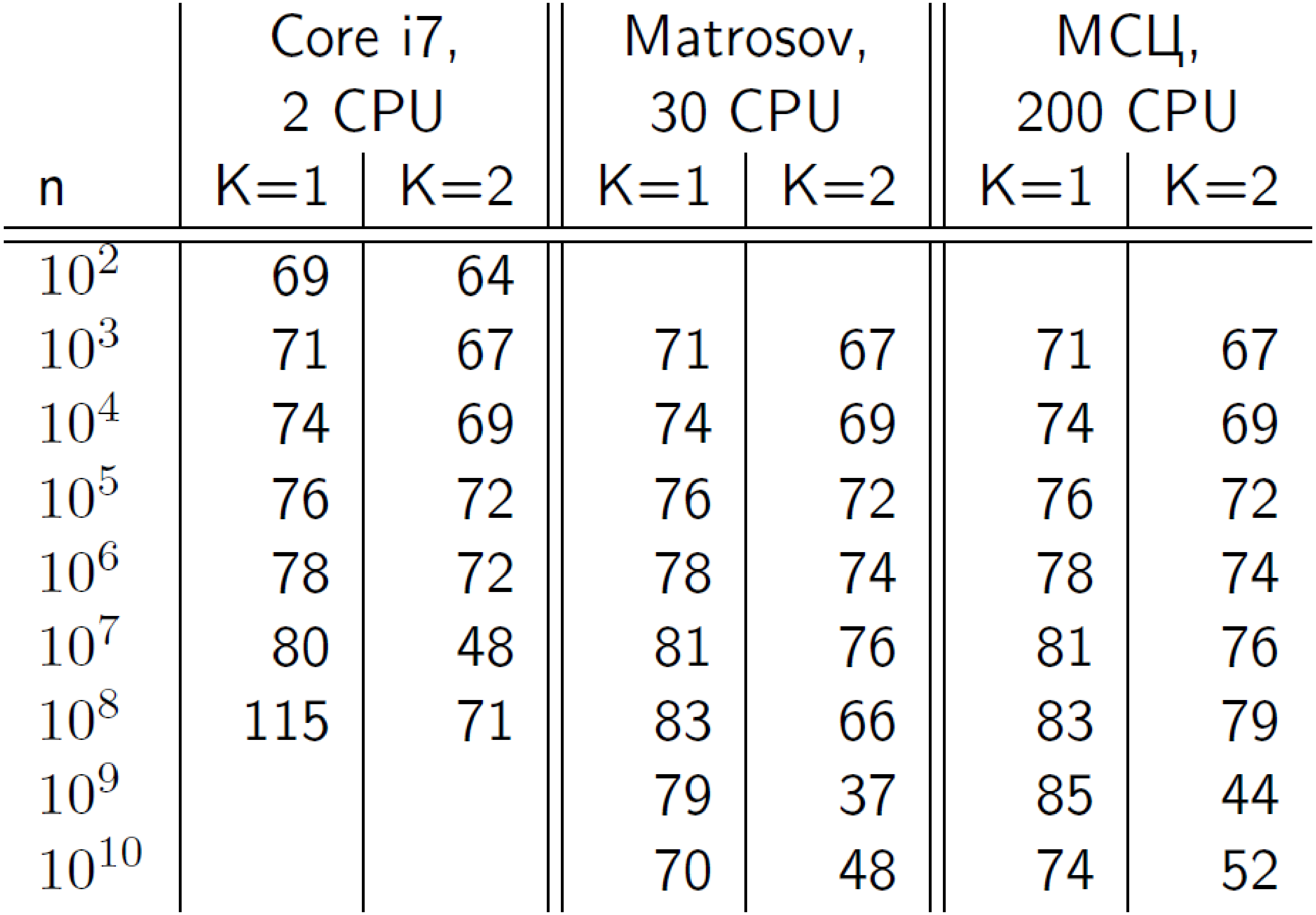
Computing experiments, test 2

Modification of Polyak method, time (s)



Computing experiments, test 2

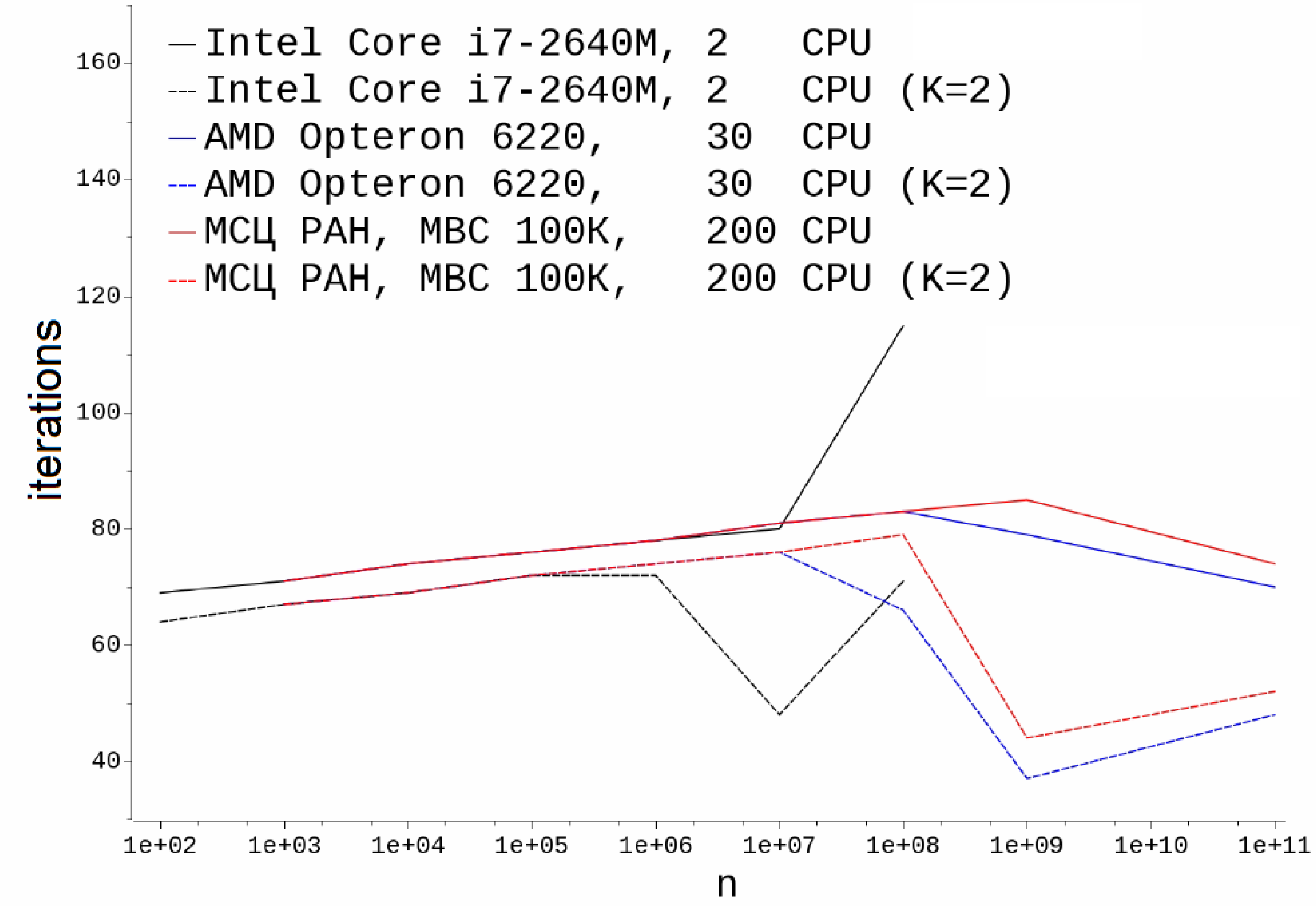
Modification of Polyak method, iterations number



	Core i7, 2 CPU		Matrosov, 30 CPU		МСЦ, 200 CPU	
n	K=1	K=2	K=1	K=2	K=1	K=2
10^2	69	64				
10^3	71	67	71	67	71	67
10^4	74	69	74	69	74	69
10^5	76	72	76	72	76	72
10^6	78	72	78	74	78	74
10^7	80	48	81	76	81	76
10^8	115	71	83	66	83	79
10^9			79	37	85	44
10^{10}			70	48	74	52

Computing experiments, test 2

Modification of Polyak method, iterations number



The problem of finding PageRank-vector

$$P^T x = x$$

$$P \in R^{n \times n}, \quad x \in R^n$$

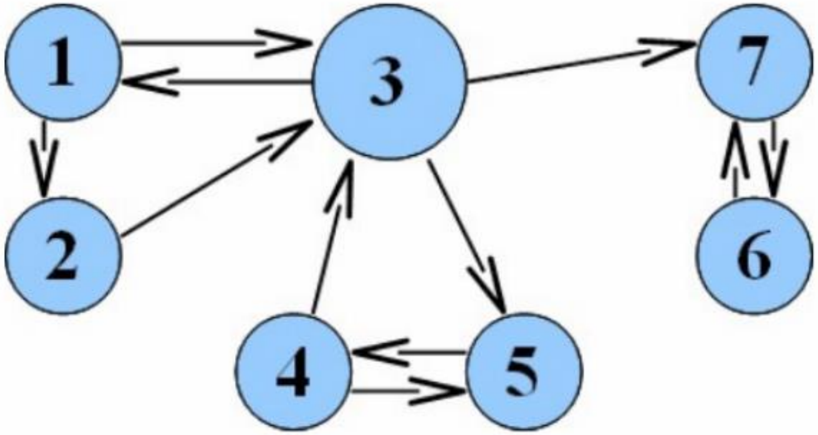
$$\langle x, e \rangle = 1, \quad e = (1, \ldots, 1)^T$$

$$x_i \geq 0, \quad i = \overline{1, n}$$

P is a stochastic matrix that defines the original graph.

It is implemented the Fletcher-Reeves conjugate gradient method for PageRank problem.

PageRank problem



$$P^T = \begin{pmatrix} 0 & 0 & 1/3 & 0 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/2 & 1 & 0 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1/3 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1/3 & 0 & 0 & 1 & 0 \end{pmatrix}$$

PageRank problem

$$f(x) = \frac{1}{2}||Ax||_2^2 \to \min_{x \in S_n(1)} \qquad (1)$$

$$f(x) = ||Ax||_\infty \to \min_{x \in S_n(1)} \qquad (2)$$

$$f(x) = \frac{1}{2}||Ax||_2^2 + \frac{\gamma}{2}(\langle x, e\rangle - 1)^2 \to \min \qquad (3)$$

where $A = P^T - I$, I is unit matrix, $S_n(1)$ is unit simplex in;

$e = (1,...,1)$;

γ is penalty parameter for missing constrair $\langle x, e\rangle = 1$.

Traditional gradient methods

We make complete (“normal”) calculation of the optimized function and its gradient at each iteration.

Computational complexity of order $O(s\,n)$.

Tested implementation (CPU + GPU):

- Conjugate gradient method (**CG**, different versions);
- Conjugate gradient method of Yuri Nesterov;
- Barzilai-Borwein method (**BB**);
- B.T. Polyak method;
- Cauchy method.

Computational experiments

Were performed on system with the following characteristics:

- Intel Core i5-2500K, 16 GB RAM, GeForce GTX 580 (512 CUDA Cores)
- gcc-5.2.1
- CUDA toolkit 7.5

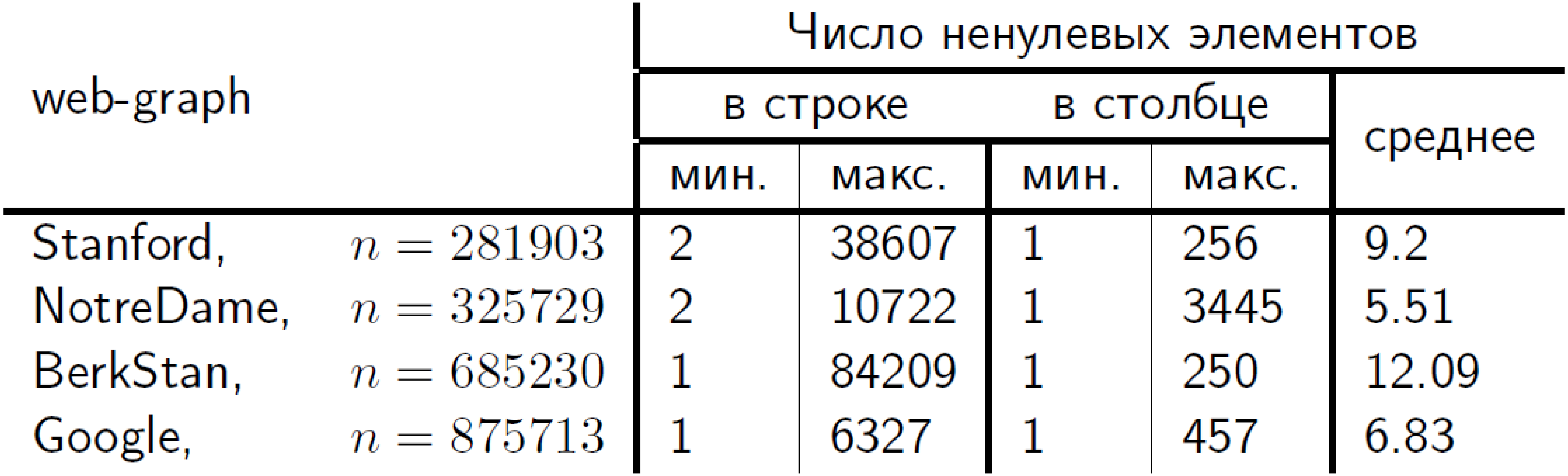
The assembly is made in Release mode.
Compilation flags: $-O2\ -\mathrm{std} = \mathrm{c}++\ 11\ -\mathrm{mcmodel} = \mathrm{small}$.

Test web-graphs were downloaded from Stanford University website:

- Stanford Large Network Dataset Collection (snap.stanford.edu/data)

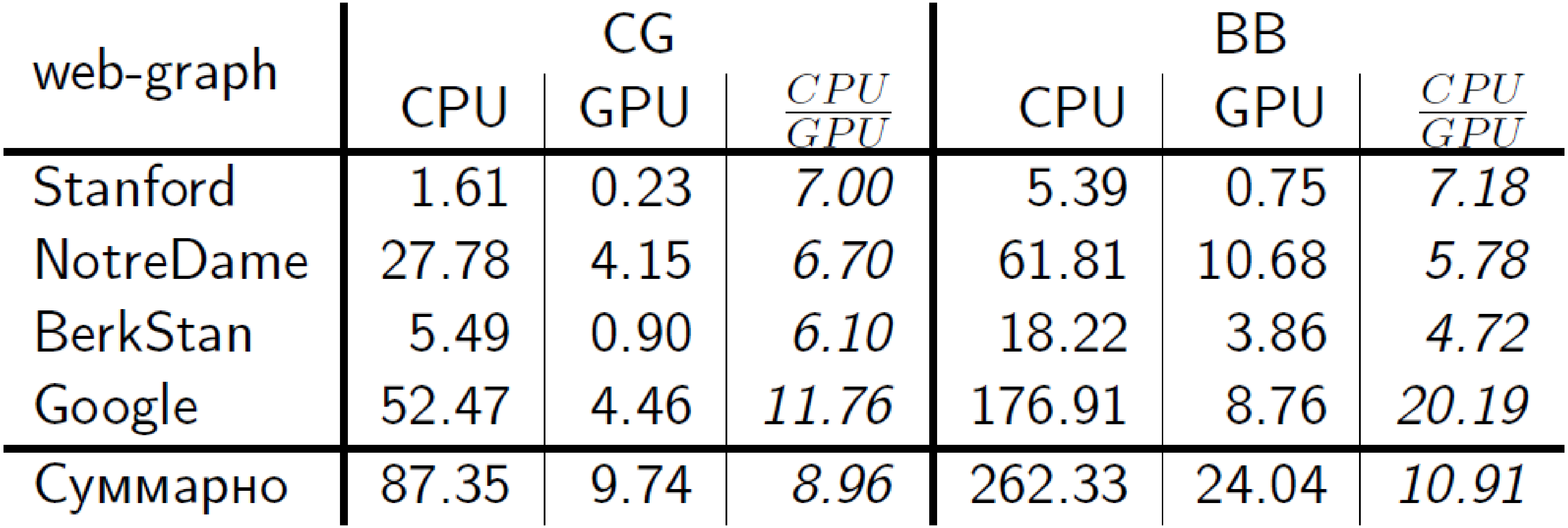
- For all the tasks we set $f^* = f_0\ 10^{-4}$, the algorithms were allowed to use unlimited time and iterations. The starting point was set $x_0 = 1/n \cdot e$.

Characteristics of the *A* matrix



web-graph		Число ненулевых элементов				
		в строке		в столбце		среднее
		мин.	макс.	мин.	макс.	
Stanford,	$n = 281903$	2	38607	1	256	9.2
NotreDame,	$n = 325729$	2	10722	1	3445	5.51
BerkStan,	$n = 685230$	1	84209	1	250	12.09
Google,	$n = 875713$	1	6327	1	457	6.83

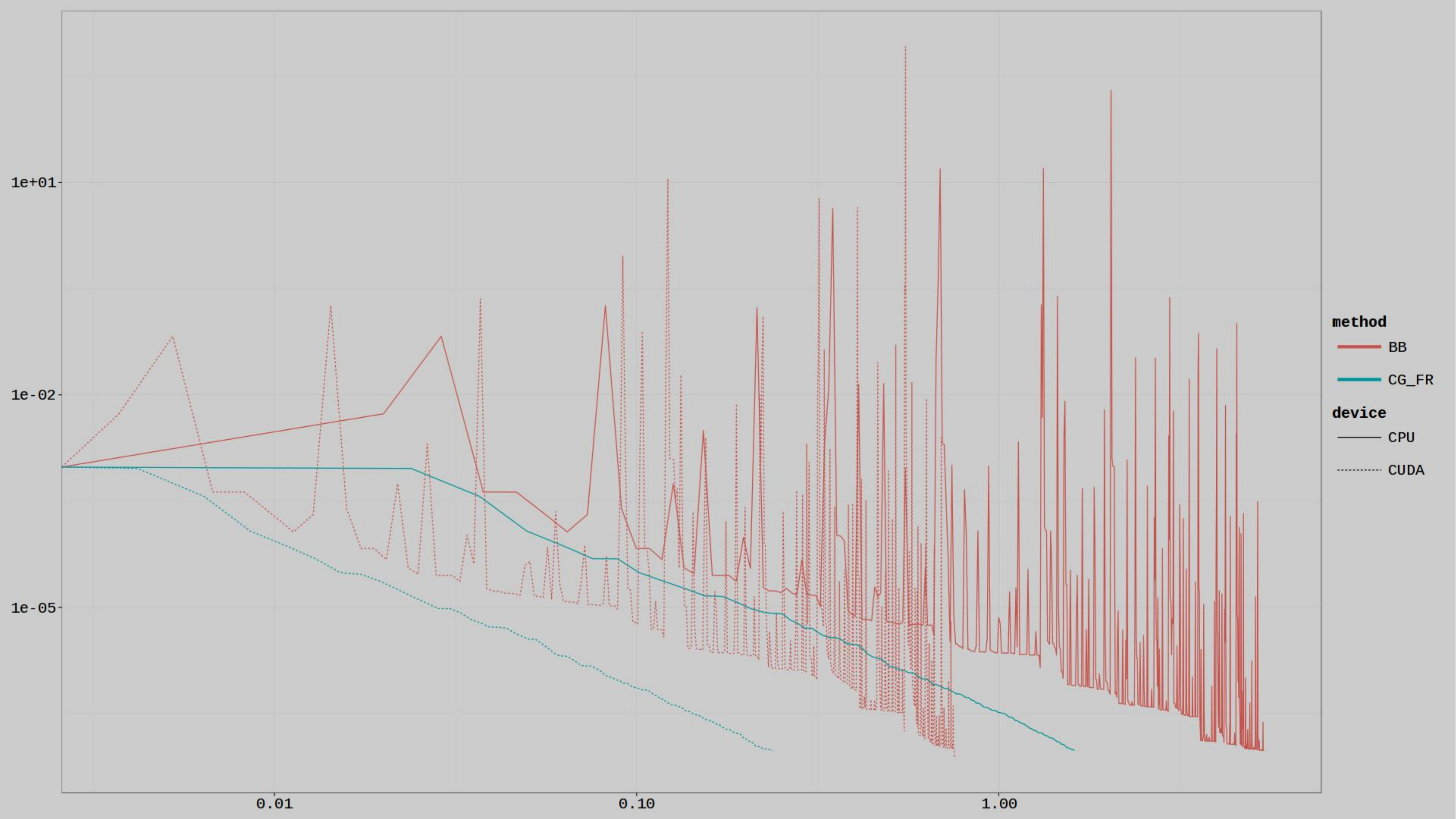
Minimization time



web-graph	CG			BB		
	CPU	GPU	$\frac{CPU}{GPU}$	CPU	GPU	$\frac{CPU}{GPU}$
Stanford	1.61	0.23	*7.00*	5.39	0.75	*7.18*
NotreDame	27.78	4.15	*6.70*	61.81	10.68	*5.78*
BerkStan	5.49	0.90	*6.10*	18.22	3.86	*4.72*
Google	52.47	4.46	*11.76*	176.91	8.76	*20.19*
Суммарно	87.35	9.74	*8.96*	262.33	24.04	*10.91*

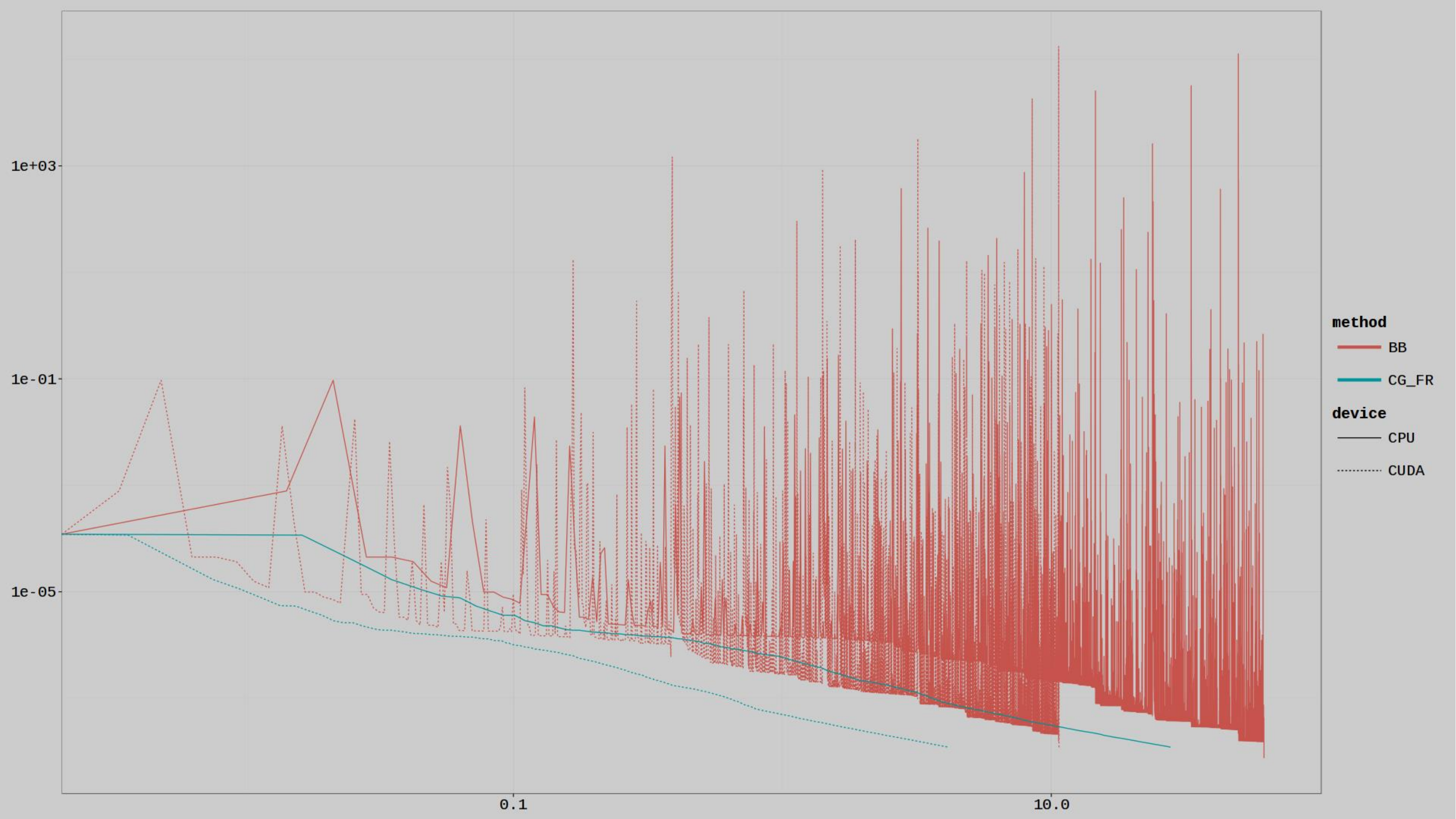
Stanford Problem

Methods convergence



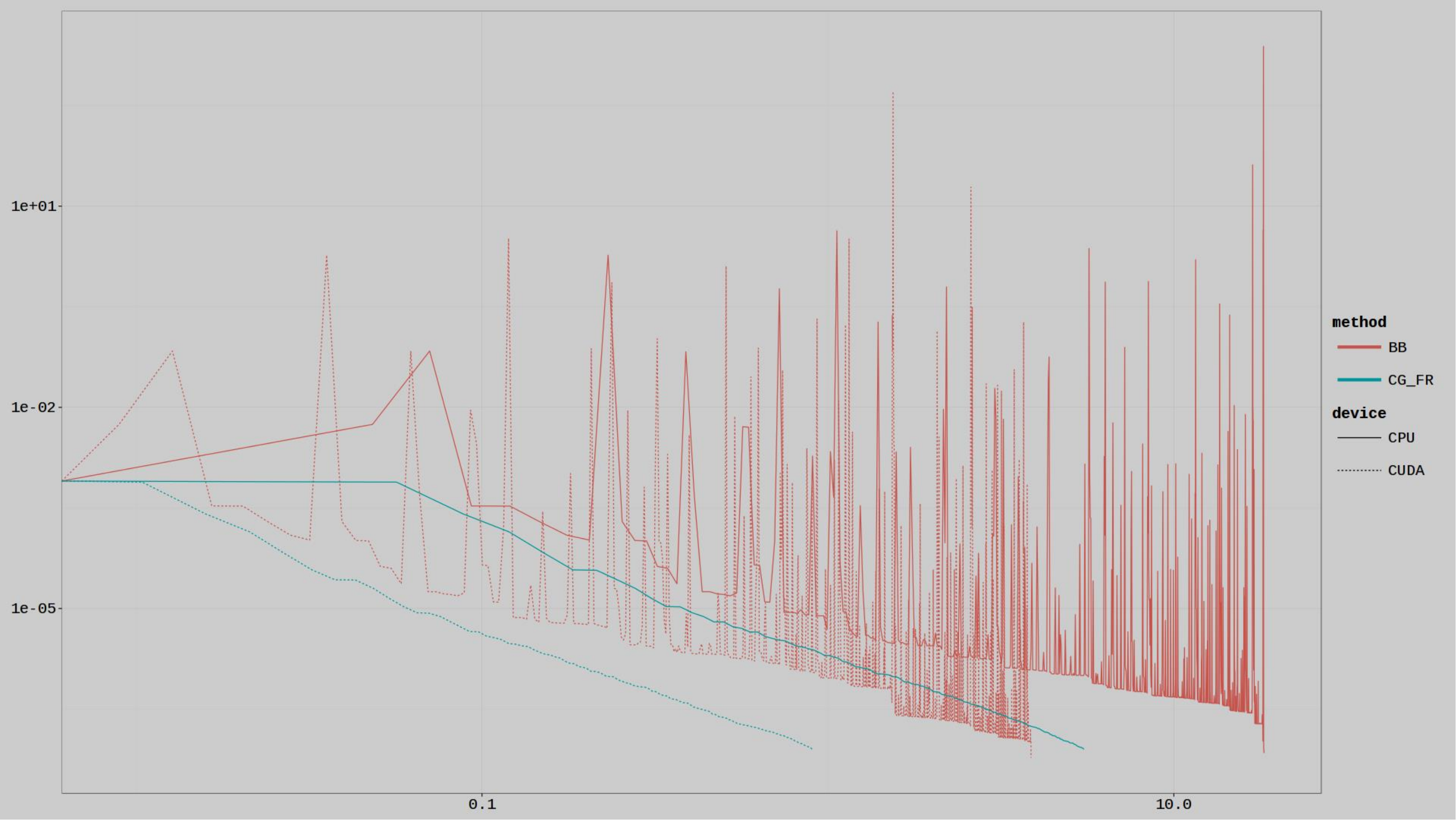
NotreDame Problem

Methods convergence



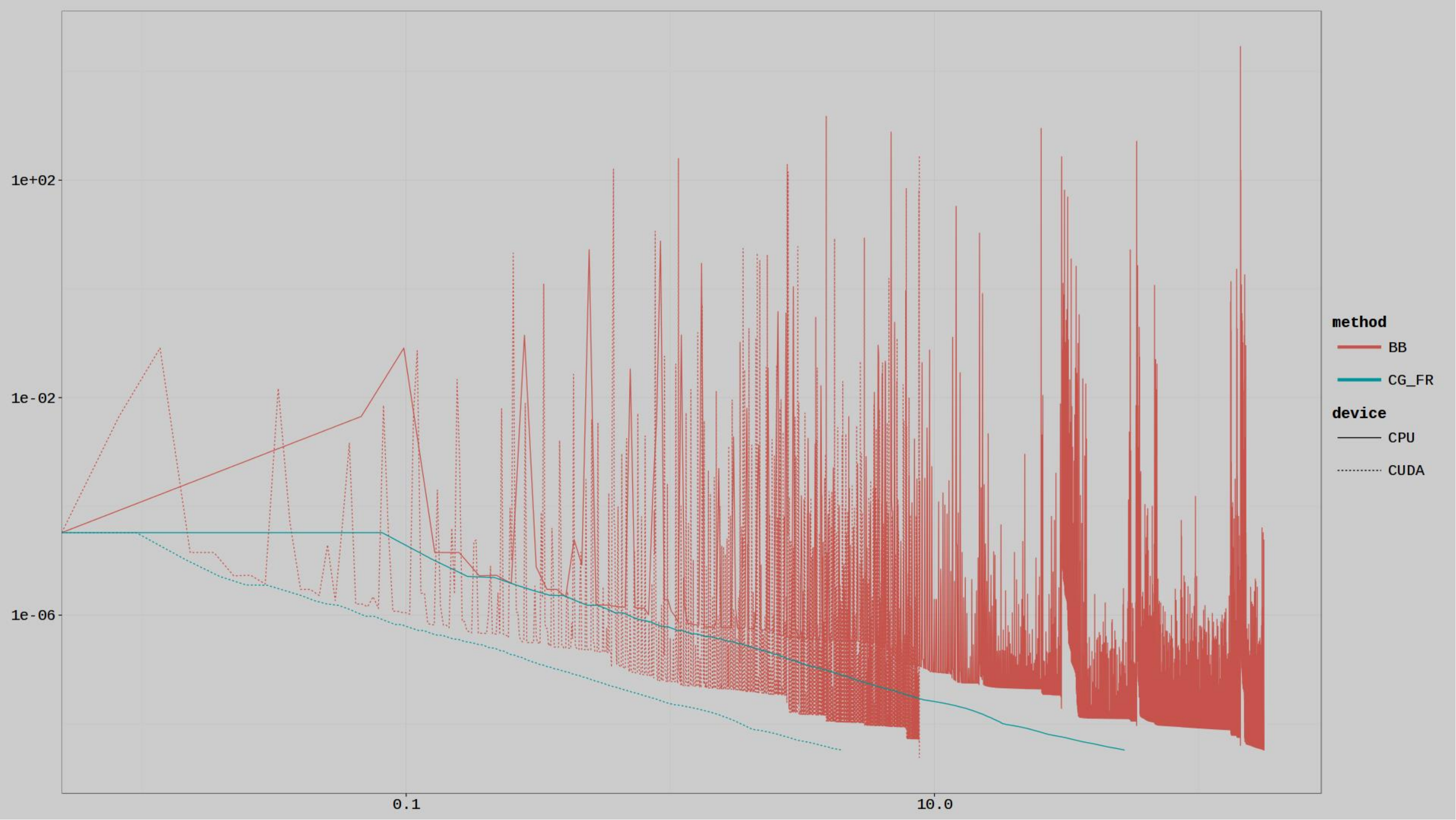
BerkStan Problem

Methods convergence



Google Problem

Methods convergence



Update methods

The basic idea, which can allow to solve effectively such problems, is to take into account the matrix sparseness when selecting optimization method, and implementing the program.

Considered methods at each iteration minimize not all the components of the vector x, but only several (1-2 variables). This approach is associated with a sparse statement (the matrix A), and with the sparseness of the solution (a vector x) for this class of problems.

This approach allows one to build effective in complexity evaluating methods, but often requires a number of non-trivial steps for efficient software implementations.

«Пересчетные» методы

Основной идеей, позволяющей эффективно решать такие задачи является правильный учет разреженности исходной постановки как на уровне выбираемого метода оптимизации, так и на уровне его последующей программной реализации.

Рассматриваемые методы относятся к покомпонентным, т.е. на каждой итерации производится минимизация не по всем компонентам вектора x, а лишь по небольшой его части (1-2 переменных). Данный подход связан как с разреженностью постановки (матрица A), так и с разреженностью самого решения (вектор x) для рассматриваемого класса задач.

Такой учет фактора разреженности позволяет построить эффективные методы с точки зрения оценки сложности, но зачастую требует выполнения ряда нетривиальных шагов для получения эффективных программных реализаций.

Example of Update Iteration

Accordint to the philosophy of componentwise methods for each iteration, we slightly change the optimized vector $x_{k+1} = x_k + e_k$. Here the vector e_k consists mainly of zeros, so these “full” calculations become too “expensive”.
We turn to the updating function and its gradient:

$$b_{k+1} = Ax_{k+1} = A(x_k + e_k) = b_k + Ae_k;\ O(s||e_k||_0)$$

$$g_{k+1} = A^T Ax_{k+1} = A^T Ax_k + A^T Ae_k = g_k + A^T Ae_k;\ O(s^2||e_k||_0)$$

Obviously, the complexity of these operations is substantially less than one when using the traditional approach.

Implemented Methods

Application of described updating ideology allows us to create effective methods for this class of problems, which have significantly better estimates regarding to the “traditional” ones.

We propose 3 of these methods:

- **NL1** – direct gradient method in the 1-norm;
- **FW** – Frank-Wolf method of conditional gradient;
- **GK** – randomized mirror descent in the Grigoriadias-Khachiyan form.

NL1

Direct gradient method

$$x_{k+1} = x_k + h \cdot y_k$$

$$h = \frac{1}{L}(g_{\max} - g_{\min}) = \frac{1}{3}(g_{\max} - g_{\min})$$

$$y = (0, ..., 0, 1^{\max}, 0, ...0, 1^{\min}, 0, ...0), ||y||_0 = 2$$

$$g_{\max} = \underset{i=1,...,n}{\operatorname{argmax}} \; \partial f(x_k)/\partial x^i$$

$$g_{\min} = \underset{i=1,...,n}{\operatorname{argmin}} \; \partial f(x_k)/\partial x^i$$

Here we got 2 function and gradient computation at one iteration.

FW

Frank-Wolf method of conditional gradient

$$x_{k+1} = (1-\gamma_k)x_k + \gamma_k y_k, \ \gamma_k = \frac{2}{k+1}, \ \ k = 1, 2, ...$$

$$\langle \nabla f(x_k), y \rangle \to \min_{y \in S_n(1)}$$

$$y_k = (0, ..., 0, 1, 0, ..., 0),$$

Where 1 is on the position:

$$i_k = \operatorname*{argmin}_{i=1,\dots,n} \ \partial f(x_k)/\partial x^i$$

Here we got 1* function and gradient computation at one iteration.

GK

Saddle statement of problem and randomized mirror descent

$$f(x) = ||Ax||_\infty \to \min_{x \in S_n(1)}$$

This problem can be rewritten in a saddle form:

$$\min_{x \in S_n(1)} \max_{||y||_1 \leq 1} \langle Ax, y \rangle .$$

As a result, the problem can be rewritten, preserving the properties of sparseness as:

$$\min_{x \in S_n(1)} \max_{\omega \in S_{2n}(1)} \langle \omega, \widetilde{A}x \rangle .$$

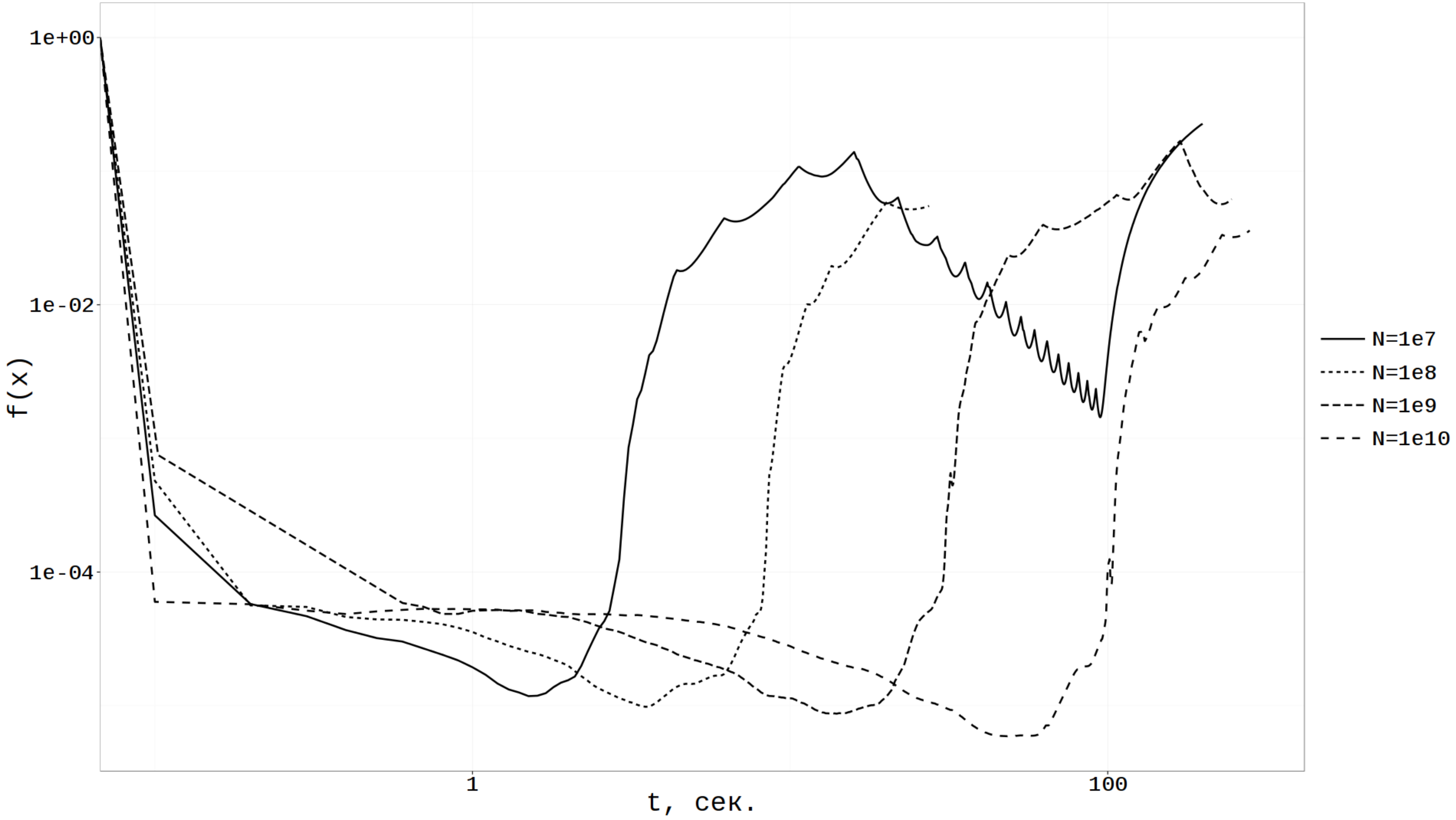
Аникин А.С., Гасников А.В., Горнов А.Ю., Камзолов Д.И., Максимов Ю.В., Нестеров Ю.Е.Эффективные численные методы решения задачи PageRank для дважды разреженных матриц //Труды МФТИ. 2015. Т. 7, № 4, С. 70-91.

Computational experiments

The behavior of these methods was studied on the PageRank problem with matrices of 3 types:

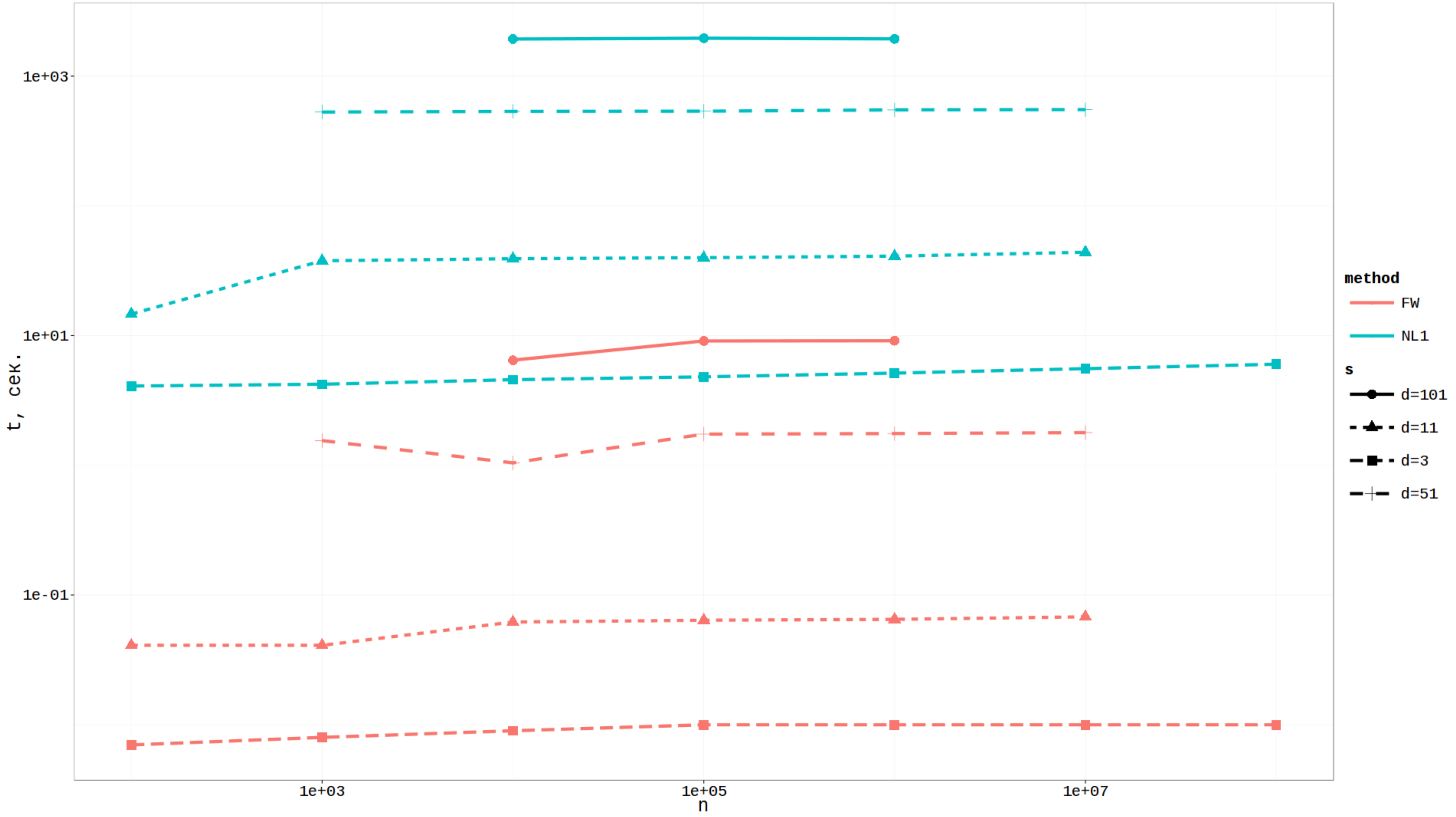
- Diagonal, with given number of the diagonals: $n_d = 1,3,5,$ ... Each matrix row / column contain: $(n_d - 1)/2 + 1 \leq s \leq n_d$ nonzero elements;
- Randomly generated structure. Each matrix row / column contains exactly s of non-zero elements;
- Stanford University problems. Matrix contain any number of non-zero elements.

GK Method with different *N*



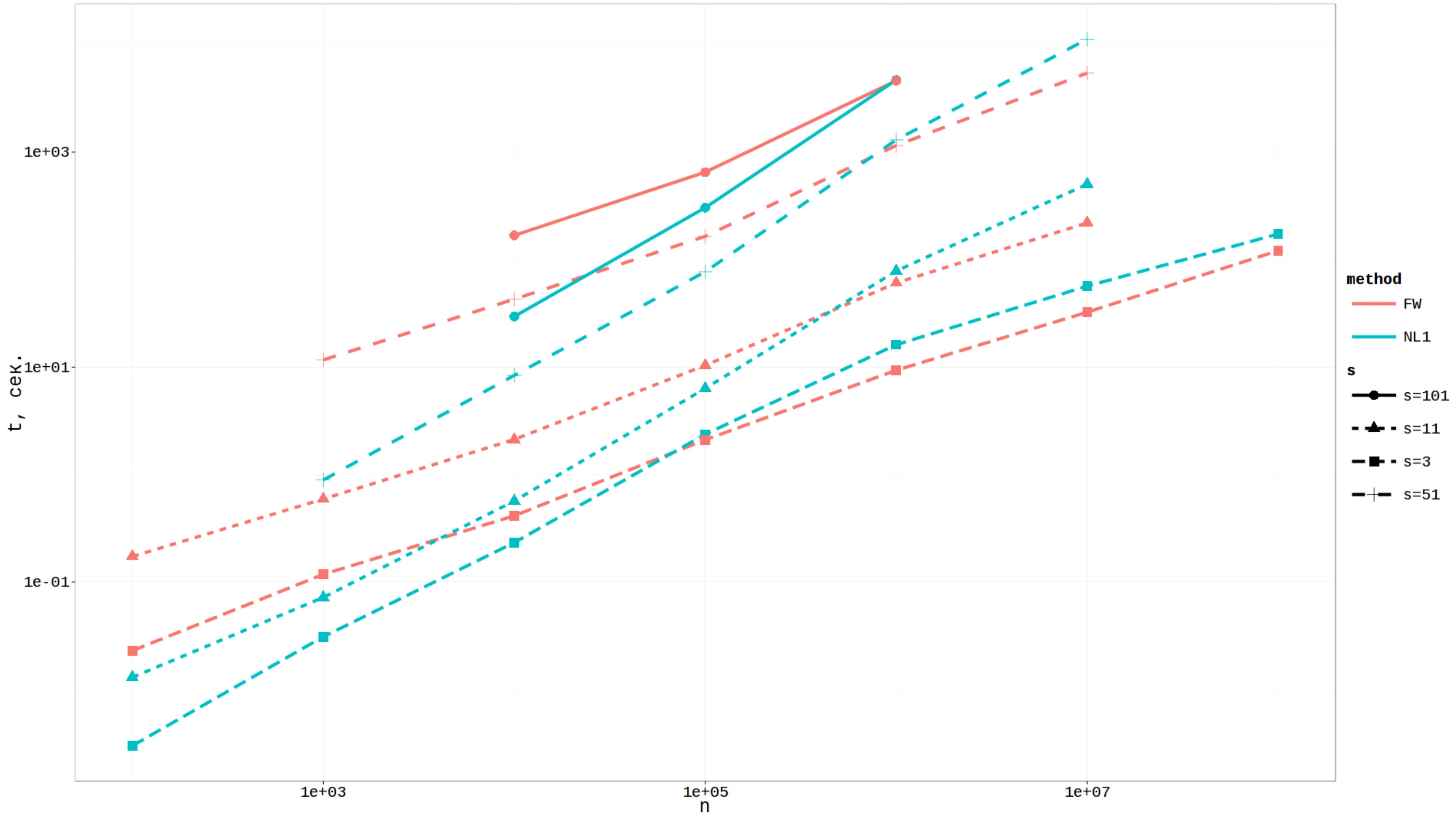
A is random, $n=10^2$, $s=3$.

FW vs NL1



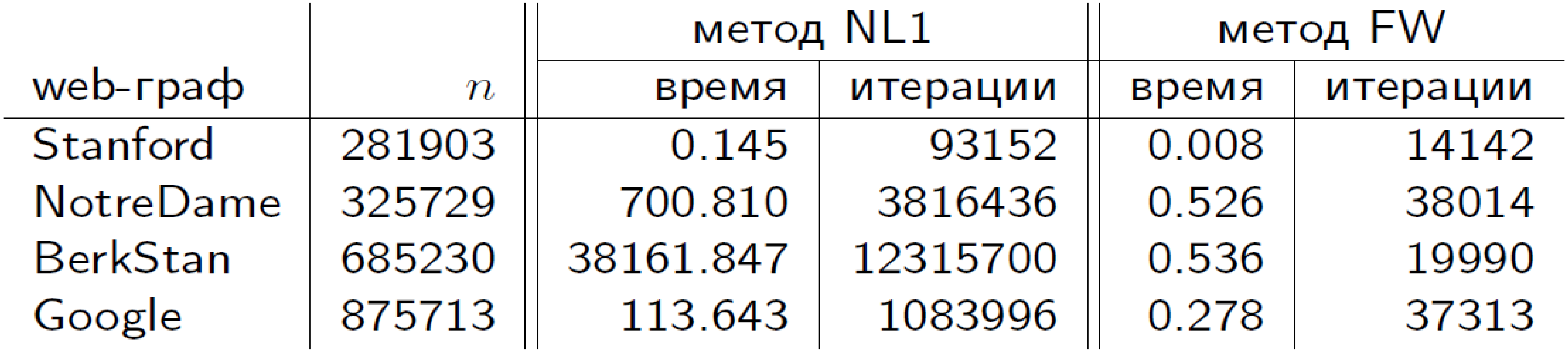
A is diagonal.

FW vs NL1



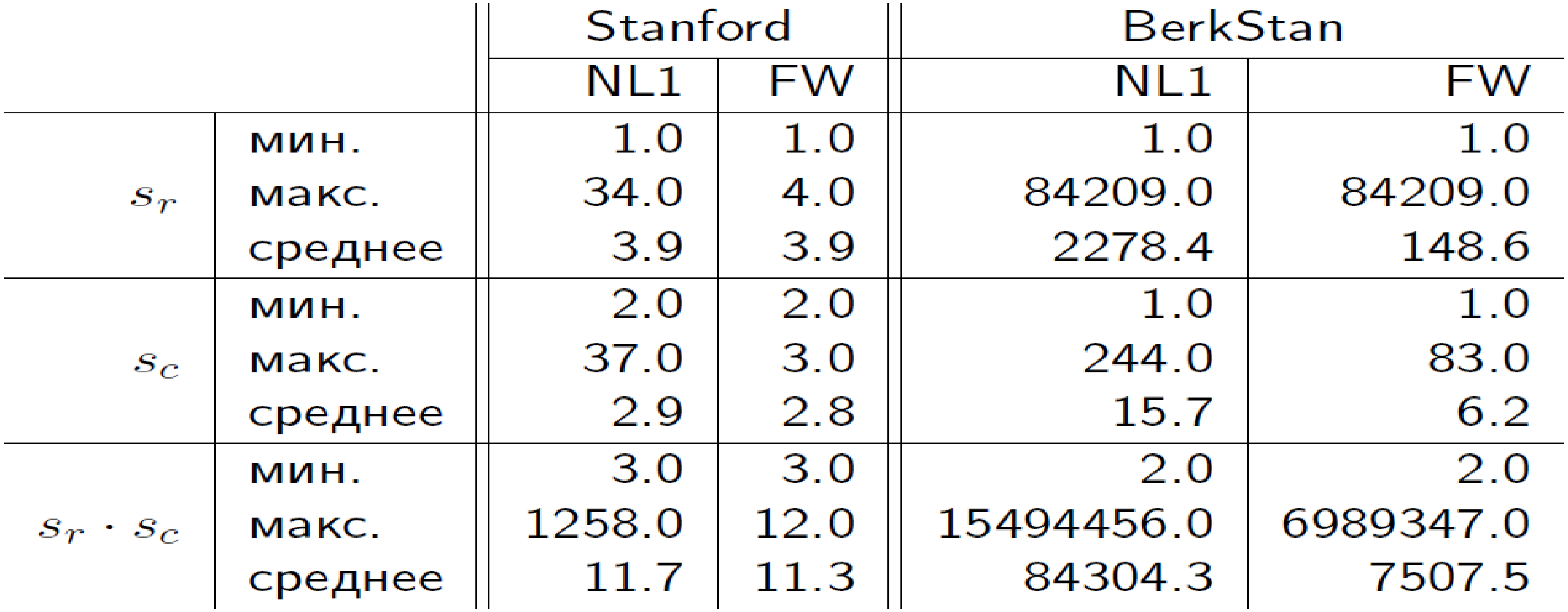
A is random.

Time (sec.) for solving PageRank problem for web-graphs



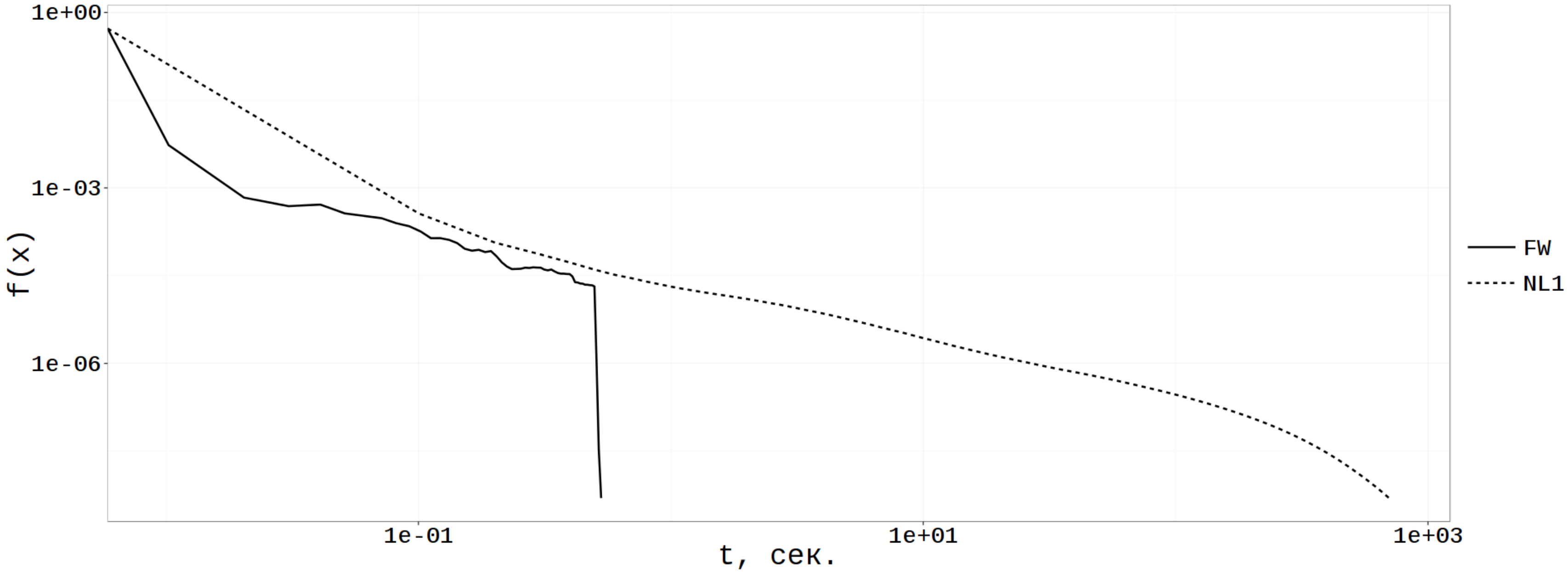
		метод NL1		метод FW	
web-граф	n	время	итерации	время	итерации
Stanford	281903	0.145	93152	0.008	14142
NotreDame	325729	700.810	3816436	0.526	38014
BerkStan	685230	38161.847	12315700	0.536	19990
Google	875713	113.643	1083996	0.278	37313

Iteration costs for web-graphs problem

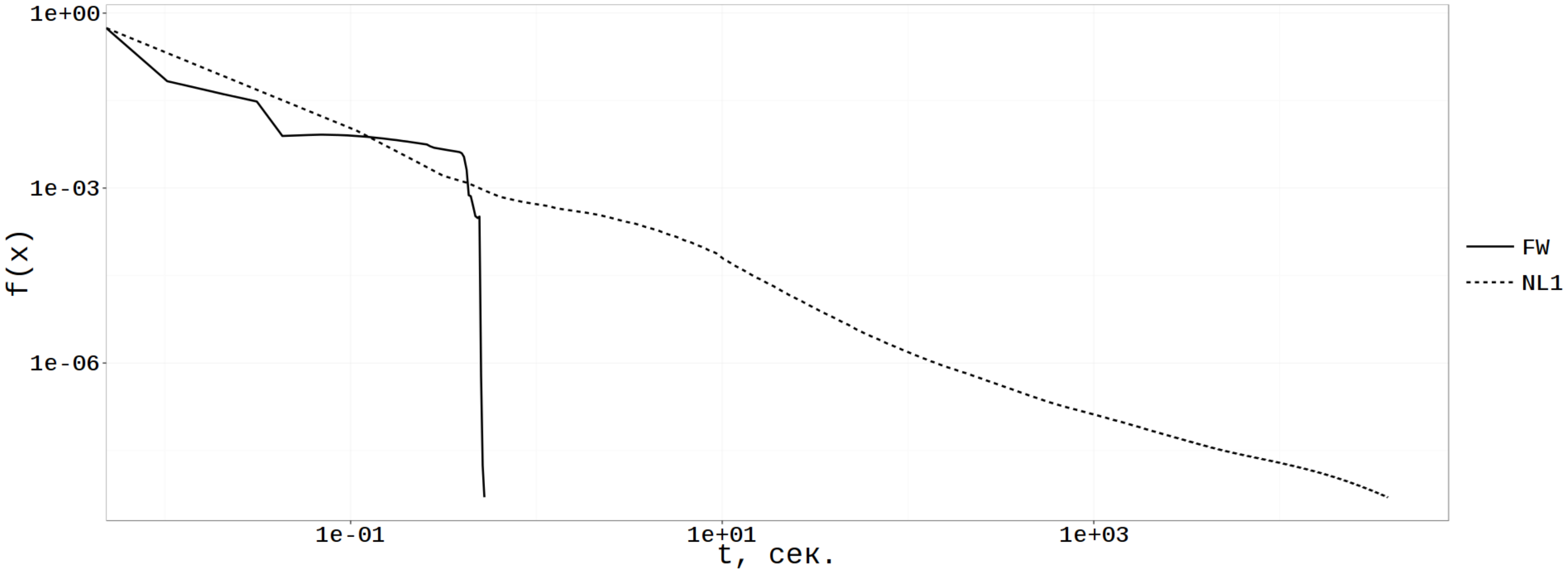


		Stanford		BerkStan	
		NL1	FW	NL1	FW
s_r	мин.	1.0	1.0	1.0	1.0
	макс.	34.0	4.0	84209.0	84209.0
	среднее	3.9	3.9	2278.4	148.6
s_c	мин.	2.0	2.0	1.0	1.0
	макс.	37.0	3.0	244.0	83.0
	среднее	2.9	2.8	15.7	6.2
$s_r \cdot s_c$	мин.	3.0	3.0	2.0	2.0
	макс.	1258.0	12.0	15494456.0	6989347.0
	среднее	11.7	11.3	84304.3	7507.5

Web-NotreDame problem solution



Web-BerkStan problem solution



Conclusions

- The optimization problem of large dimensions **can be** solved (for complex productions - the principle “best-of-known”);

- These problems **should be** studied deeply and actively by a wide range of specialists;

- The correct choice of methods is an important issue, especially for the class of Huge-Scale problems; correct setting of optimization techniques parameters significantly affect their performance and efficiency.

Thanks for your attention!

A.Yu. Gornov, A.S. Anikin,
T.S. Zarodnyuk, P.S. Sorokovikov

Matrosov Institute for System Dynamics and
Control Theory of SB RAS, Irkutsk, Russia
gornov@icc.ru

Russian National Conference MMPR-2019

November 26–29, 2019, Moscow, Russia