Алгоритмы решения сверхбольших задач непрерывной оптимизации

Горнов А.Ю.[1]
gornov@icc.ru,

Аникин А.С.[1]
Андрианов А.Н.[2]
Гасников А.В.[3]

[1]ИДСТУ СО РАН, Иркутск

[2]ИПМ РАН, Москва

[3]ПреМоЛаб МФТИ, Москва

ММРО 2015

1. Задача нахождения PageRank-вектора
2. Транспортная задача
3. Оптимизация потенциала Китинга
4. Оптимизация потенциала Морса
5. Задача оптимального управления
6. (Квази)сепарабельные задачи выпуклой оптимизации класса Huge-Scale

$$f(x) \to \min$$

$$\underline{x} \leq x \leq \overline{x}$$

1. Выпуклые
2. Унимодальные
3. Невыпуклые

Huge-Scale

Классификация локальных задач оптимизации по числу переменных, предложенная Ю.Е. Нестеровым:

- "Small" – до 100 переменных
- "Medium" – от 10^3 до 10^4 переменных
- "Large" – от 10^5 до 10^7 переменных
- "Huge" – более 10^8 переменных

Технологии и подходы

- Новые (старые?) алгоритмы
- Учет структуры задачи
- Рандомизация
- Вычислительные деревья Л.В. Канторовича
- Кластерные системы
- Графические ускорители (GPU)
- Учет архитектурных особенностей процессоров (cache-friendly code)
- Многометодные вычислительные схемы
- ...

Критические ресурсы

- Процессорное время (вычислительная сложность)
- Оперативная память (размерность задачи, объем данных)

Требуемая память

float – 4 Б., double – 8 Б.

Размер n-элементного вектора:

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

Поисковые методы

- Метод Пауэлла-Брента (Q)
- Метод покоординатного поиска
- Безградиентный криволинейный поиск
- Метод Хука-Дживса
- Метод Лууса-Яаколы
- Партан-методы
- Метод Растригина

Градиентные методы

- Метод Ньютона (Q)
- Глобализованный метод Ньютона (Q)
- Метод эллипсоидов (Q)
- Квазиньютоновский метод DFP+BFGS (Q)
- Метод Коши
- Метод Barzilai-Borwein
- Метод L-BFGS
- Градиентный криволинейный поиск
- Декомпозиционный метод по большим градиентам
- Декомпозиционный метод по малым градиентам
- “Диагональный” метод Ньютона

Градиентные методы

- Метод интегрирования DOPRI5
- Метод интегрирования Эйлера с итерациями
- Метод интегрирования Рунге-Кутты
- Метод интегрирования Адамса “предиктор-корректор”
- Метод центров
- Метод Поляка
- Прямо-двойственный метод приведенного градиента
- Прямо-двойственный метод Поляка
- Метод Нестерова 1983 г.
- “Быстрый градиентный” метод Нестерова
- Методы сопряженных градиентов (23 варианта)

Задача нахождения PageRank-вектора

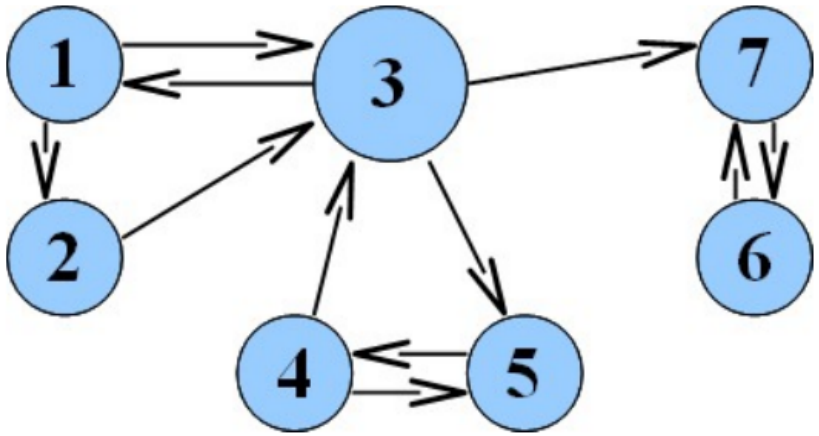
$$P^T x = x$$

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

$$\langle x, e \rangle = 1, \;\; e = (1, ..., 1)^T$$

$$x_i \geq 0, \;\; i = 1, ..n$$

P - стохастическая матрица



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

Исходную задачу можно свести к задаче выпуклой оптимизации различными способами

1. Nesterov Y. Subgradient methods for huge-scale optimization problems // Mathematical Programming, Vol. 146, No. 1-2. pp. 275-297 (2014).

2. A.V. Gasnikov, D.Yu. Dmitriev. On efficient randomized algorithms for finding the PageRank vector // Computational Mathematics and Mathematical Physics, Vol. 55, No. 3, pp. 349-365 (2015)

$$f(x) = \frac{1}{2}||Ax||_2^2 + \frac{\gamma^-}{2}\sum_{i=1}^{n}(-x_i)_+^2 + \frac{\gamma^+}{2}(\langle x, e\rangle - 1)^2 \to \min$$

где:

$A = P^T - I$, I - единичная матрица;

$(x)_+ = \begin{cases} x, & \text{если } x \geq 0 \\ 0, & \text{если } x < 0 \end{cases}$ - штраф для отрицательных элементов;

γ^- - параметр штрафа для отрицательных элементов;

γ^+ - параметр штрафа за нарушение ограничения $\langle x, e\rangle = 1$.

$$y = Ax$$

$$f(x) = \frac{1}{2}||Ax||_2^2 = \frac{1}{2}||y||_2^2$$

$$\nabla f(x) = A^T(Ax) = A^T y$$

Основная трудоемкая операция - умножение разреженной матрицы на плотный вектор. В работе используется подход быстрого пересчета (обновления) $y = Ax$, предложенный Ю.Е. Нестеровым для случая, когда требуется вычислить $y^+ = Ax^+$, где $x^+ = x + h$, а вектор h – разрежен (меняется только часть оптимизируемых переменных):

$$y^+ = y + Ah$$

Nesterov Y. Subgradient methods for huge-scale optimization problems // Mathematical Programming. 2014. Vol. 146, 1-2. P. 275—297.

Значение функции и градиента в точке x^+ может быть вычислено следующим образом:

$$f^+(h) = f(x^+) = \frac{1}{2}||y^+||_2^2 = \frac{1}{2}||y + Ah||_2^2$$

$$\nabla f^+(h) = \nabla f(x^+) = A^T y^+ = A^T y + A^T(Ah) = \nabla f(x) + A^T(Ah)$$

Отобранные методы:

1. Метод Гасникова-Нестерова (GN)
2. Метод “Diagonal Search” (DS)
3. Метод Сопряженных градиентов (Флетчер-Ривз)

Метод Гасникова-Нестерова (GN)

Основной идеей метода является спуск по такому направлению, которое не нарушает ограничение $< x, e >= 1$. Метод требует стартовую точку, удовлетворяющую этому ограничению.

На каждой итерации метод выбирает максимальный и минимальный элементы градиента и осуществляет спуск по соответствующим 2 переменным по "диагонали", что обеспечивает соблюдение ограничения $< x, e >= 1$.

Для быстрого пересчета $y = Ax$ и поиска минимального / максимального элементов градиента используется технология деревьев Ю.Е. Нестерова.

Метод Гасникова-Нестерова (GN)

1. Инициализируются деревья для поиска min/max компонентов градиента g и пересчета $y = Ax$.
2. Вычисляется начальное $y = Ax$, полный градиент $g = \nabla f(x)$, производится полный пересчет всех деревьев.
3. Выполняется поиск максимального i и минимального j элементов градиента.
4. Выполняется шаг по выбранному направлению:
   $d = (g_i - g_j)/4$,
   $x_i = x_i - d \cdot g_i$,
   $x_j = x_j + d \cdot g_j$.

Метод Гасникова-Нестерова (GN)

5. Обновление всех деревьев.
6. В случае соответствия критериям останова — переход на пункт 8.
7. Следующая итерация — переход на пункт 3.
8. Конец работы алгоритма.

Метод Гасникова-Нестерова (GN)

Таким образом, на каждой итерации метода однократно производится пересчет значений градиента функции, что требует $O(s^2)$ операций. Необходимость получения минимального/максимального компонента градиента влечет за собой пересчет деревьев $tree_{min}/tree_{max}$, что дает итоговую трудоемкость итерации равной $O(s^2 \cdot \ln n)$. Трудоемкость итераций, на которых происходит вычисление f_k – порядка $O(n)$.

s - максимальное число ненулевых элементов в строке / столбце матрицы. $s \ll n$.

Метод Гасникова-Нестерова (GN/GN$_{rand}$)

Также в работе также рассматривается рандомизированный вариант метода, в котором выбор оптимизируемых на каждой итерации переменных осуществляется случайным образом.

Этот вариант метода в дальнейшем обозначен как GN$_{rand}$.
Трудоемкость итерации такого метода – порядка $O(s^2)$.

Метод “Diagonal Search” (DS)

На основе идей, заложенных в методе GN, был предложен метод DS (Горнов А.Ю., Аникин А.С.).

Этот метод осуществляет спуск по “диагонали” 2-х переменных, выбираемых случайным образом. Т.к. рассматриваемая задача PageRank – квадратичная, то для определния шага на итерации строится параболическая интерполяция по 3-м точкам на выбранному направлению.

Быстрый пересчет значения функции производится с помощью деревьев Ю.Е. Нестерова.

Метод “Diagonal Search” (DS)

1. Инициализируется дерево для пересчета значения функции.
2. Вычисляется начальное $y = Ax$, производится полный пересчет дерева.
3. Случайным образом выбираются переменные i, j.
4. По выбранному направлению $(1, -1)$ делается 2 дополнительные пробы функции, строится параболическая интерполяция, находится её минимум α

Метод “Diagonal Search” (DS)

5. Выполняется шаг по направлению:
   $x_i = x_i + \alpha$,
   $x_j = x_j - \alpha$
6. В случае соответствия критериям останова — переход на пункт 8.
7. Следующая итерация — переход на пункт 3.
8. Конец работы алгоритма.

Метод “Diagonal Search” (DS)

Таким образом, на каждой итерации метода производится только три пересчета значения функции. Итоговая трудоемкость такой итерации – порядка $O(s \cdot \ln n)$ операций.

Метод сопряженных градиентов (Флетчер-Ривс)

В тестировании использована реализация классического варианта метода сопряженных градиентов – Флетчер-Ривс.

Для улучшения эффективности работы произведена настройка алгоритмических параметров, относящихся к процедуре одномерного поиска. Каких-либо дополнительных оптимизаций метода под рассматриваемую задачу не производилось.

Программная реализация рассматриваемых методов выполнена авторами на языке C++ и протестирована на платформе GNU/Linux с использованием компиляторов gcc-4.8.4, gcc-4.9.2, gcc-5.1.0, clang-3.4.2, clang-3.5.2, clang-3.6.1, clang-3.7.0 и icc-15.0.2.

Для тестирования применялись web-графы, загруженные с сайта Стэнфордского университета:

Stanford Large Network Dataset Collection,
snap.stanford.edu/data

Для всех задач устанавливалась требуемое $f^* = f_0 \cdot 10^{-4}$, алгоритмам давалось неограниченное время и число итераций. Стартовая точка устанавливалась равной $x_0 = 1/n \cdot e$.

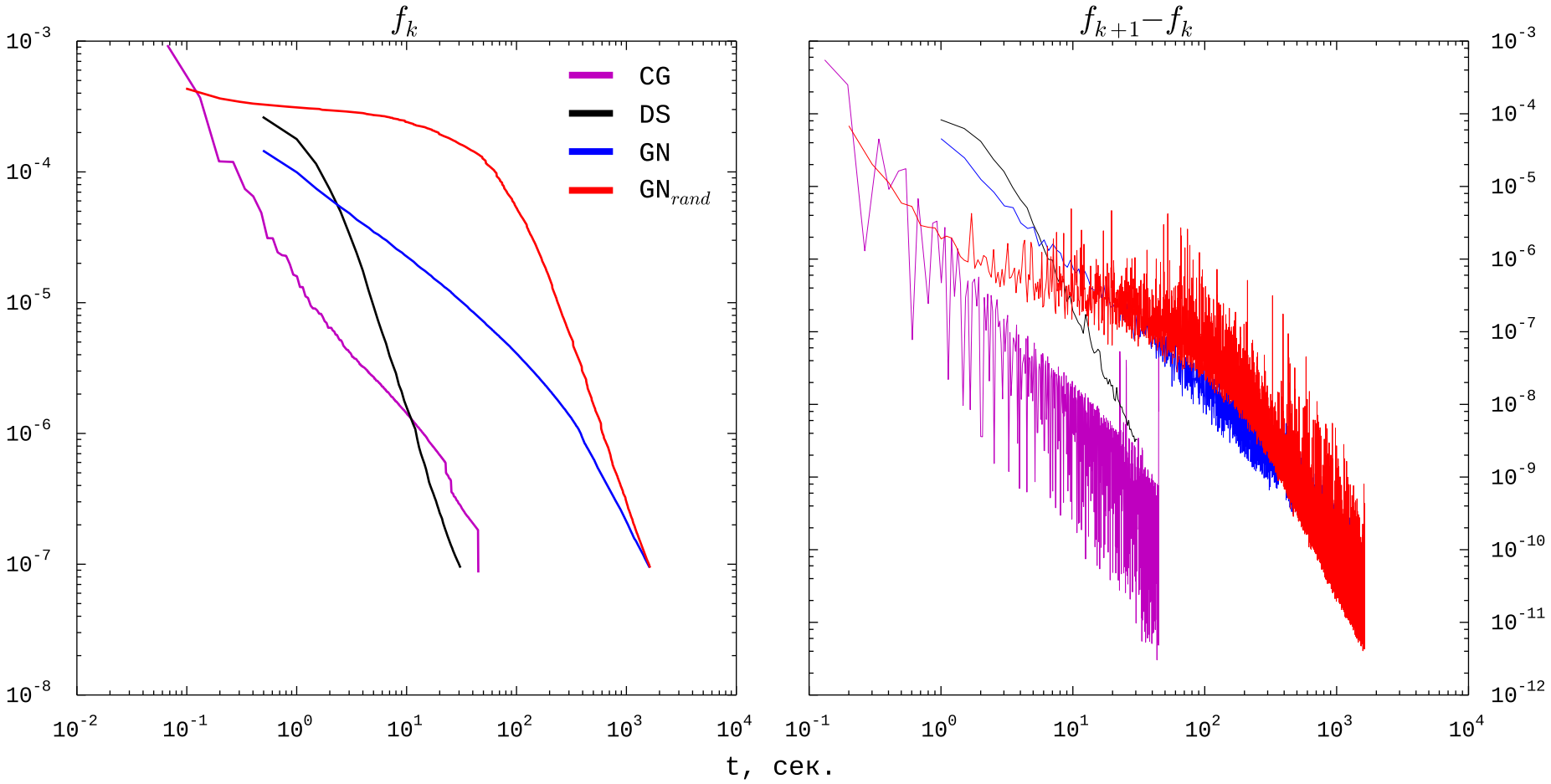
Характеристики матрицы A, построенной по базам Стэнфордского университета

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

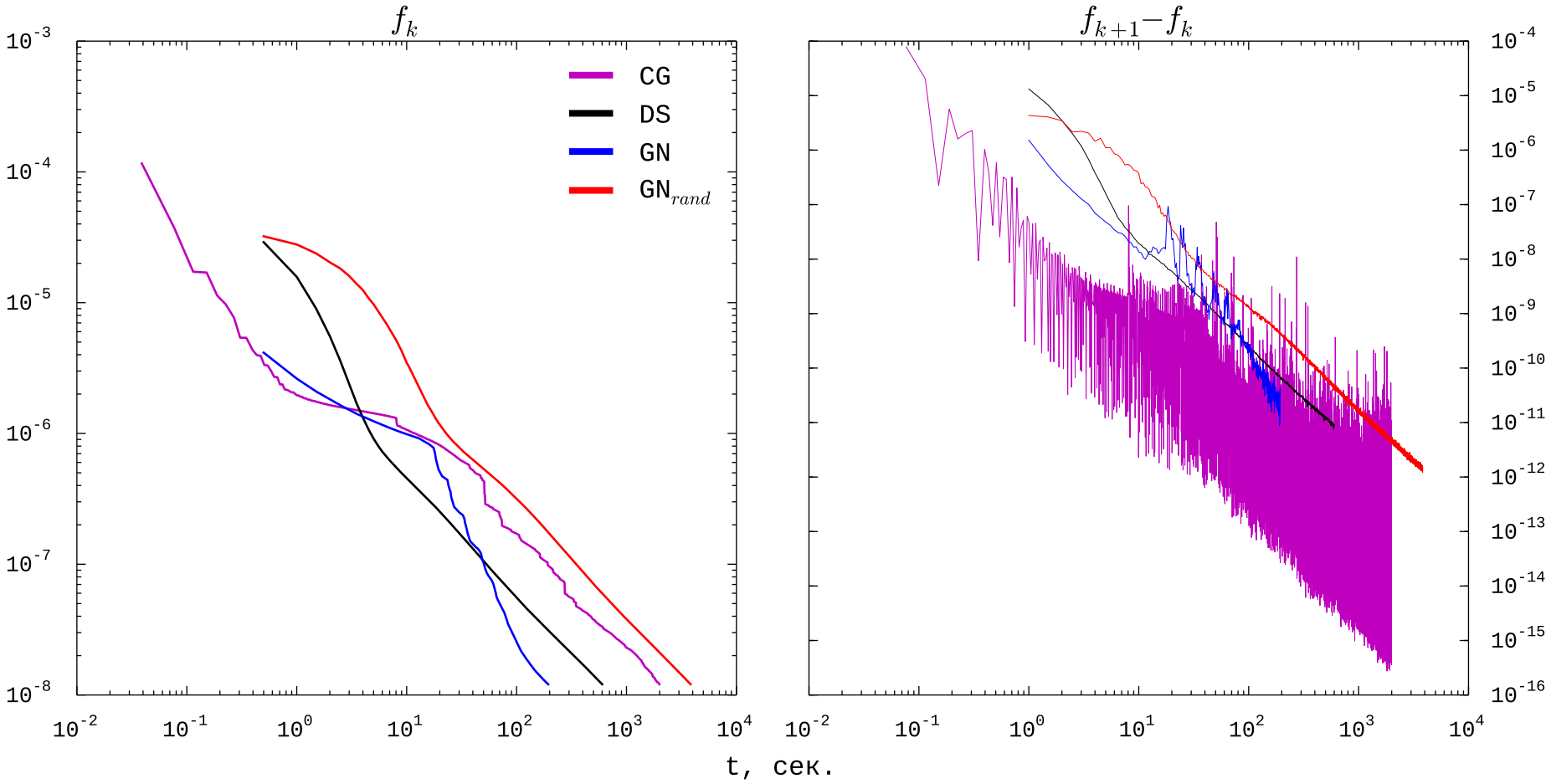
Время минимизации для матрицы A, построенной по базам Стэнфордского университета

web-graph	GN	GN_{rand}	DS	CG
Stanford, $n = 281903$	1611 сек.	1630 сек.	30 сек.	45 сек.
$f_0 = 9.57962770 \cdot 10^{-4}$	$9.5787 \cdot 10^{-8}$	$9.5795 \cdot 10^{-8}$	$9.5796 \cdot 10^{-8}$	$8.8151 \cdot 10^{-8}$
NotreDame, $n = 325729$	194 сек.	3826 сек.	600 сек.	1991 сек.
$f_0 = 1.20782983 \cdot 10^{-4}$	$1.2063 \cdot 10^{-8}$	$1.2077 \cdot 10^{-8}$	$1.2078 \cdot 10^{-8}$	$1.2070 \cdot 10^{-8}$
BerkStan, $n = 685230$	5836 сек.	18136 сек.	203.5 сек.	123.3 сек.
$f_0 = 7.96602002 \cdot 10^{-4}$	$7.9656 \cdot 10^{-8}$	$7.9659 \cdot 10^{-8}$	$7.9637 \cdot 10^{-8}$	$7.5850 \cdot 10^{-8}$
Google, $n = 875713$	2052 сек.	9578 сек.	958 сек.	2308 сек.
$f_0 = 3.29537629 \cdot 10^{-5}$	$3.2951 \cdot 10^{-9}$	$3.2952 \cdot 10^{-9}$	$3.2953 \cdot 10^{-9}$	$3.2952 \cdot 10^{-9}$
Суммарное время	9693	33170	1791.5	4467.3

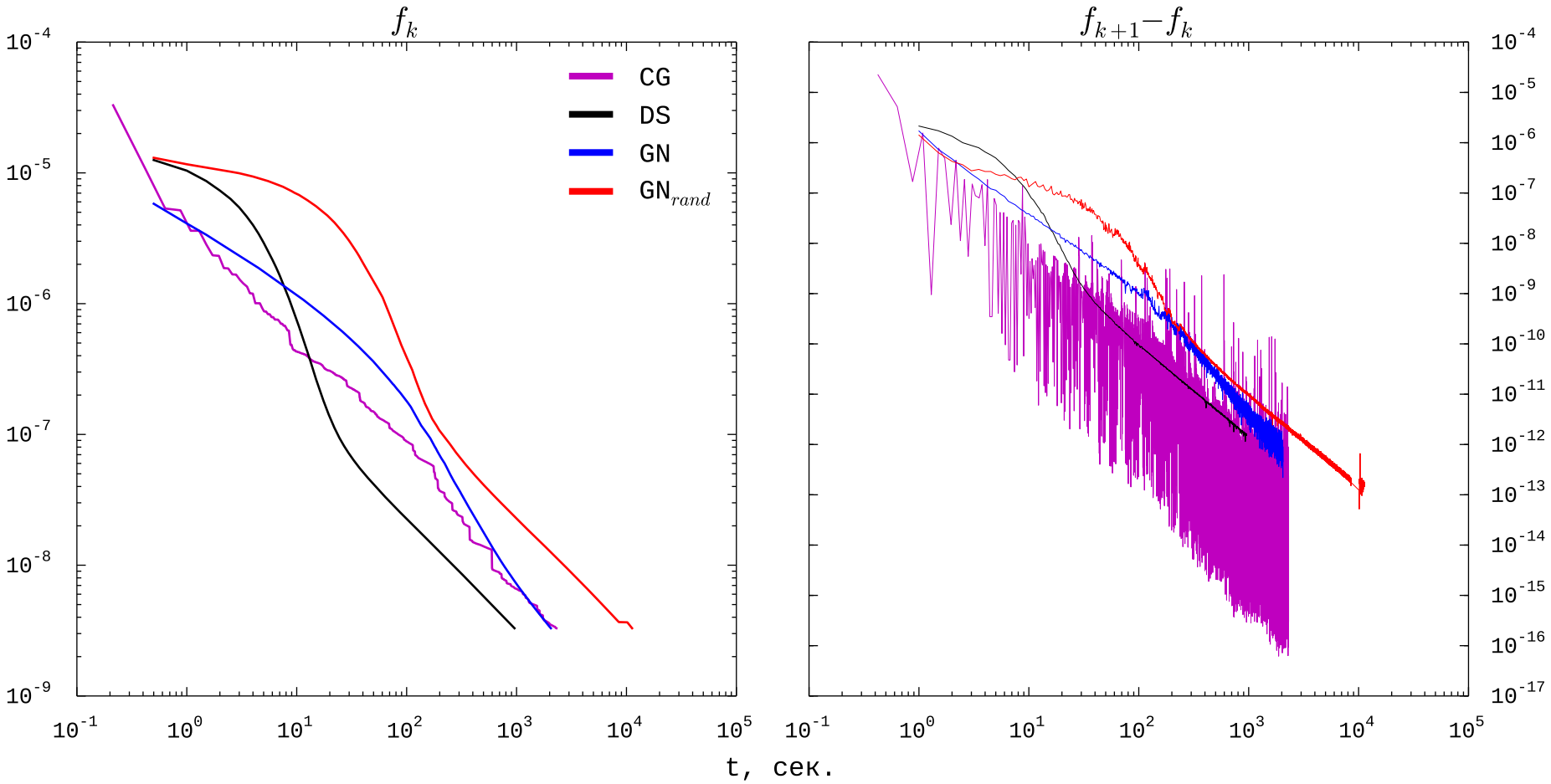
web-Stanford, сходимость методов



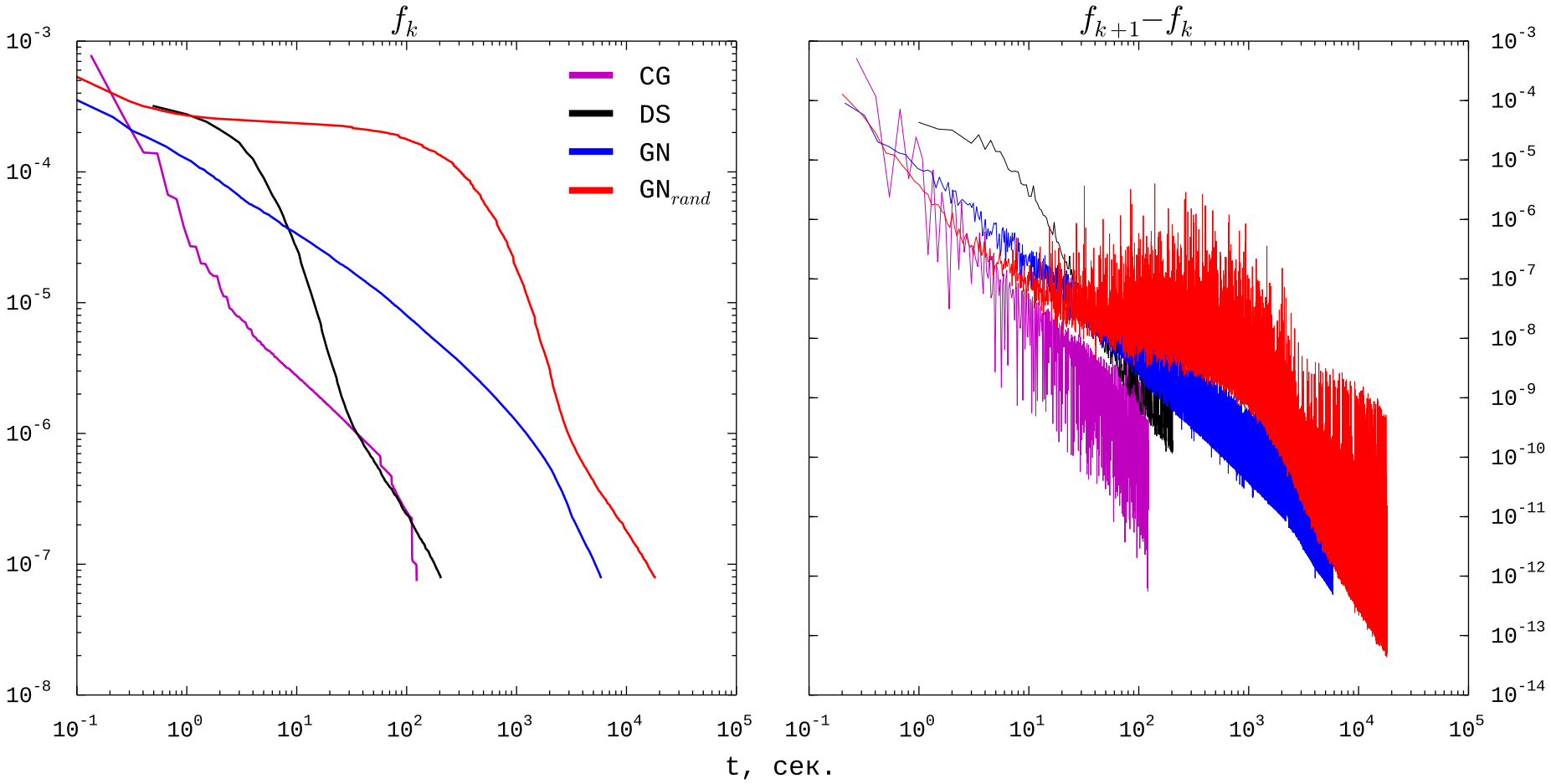
web-NotreDame, сходимость методов



web-Google, сходимость методов



web-BerkStan, сходимость методов



$$f(x) = ||x||_2^2 \to \min_{Ax=b}$$

$$A \in R^{m \times n}$$

$$x \in R^n$$

$$x_i \geq 0, \quad i = 1, ..n$$

В текущей работе она преобразована в следующую задачу безусловной минимизации (γ-модель):

$$f(x) = \frac{1}{2}||x||_2^2 + \frac{\gamma}{2}||Ax - b||_2^2 \to \min$$

Сложность решения задачи связана не только с её размерностью, но и с дополнительными требованиями к решению (вектор x), т.е. задача, по сути, является “аргументной”.

Такими требованиями к решению является ряд метрик качества.

$$P = \frac{(i : |\frac{x_i - x_i^*}{x_i^*}| < tol)}{n} \cdot 100\%$$

x - полученное решение

x^* - точное решение

Данная метрика вычисляется для различных уровней точности (tol): 20%, 40%, 60%, 80%.

$$DA = (1 - \frac{\|x - x^*\|_2}{\|x\|_2}) \cdot 100\%$$

x - полученное решение

x^* - точное решение

$$LLA = (1 - \frac{\|Ax - b\|_2}{\|b\|_2}) \cdot 100\%$$

$$LLA_{max} = \frac{(i : |\frac{y_i}{b_i}| < tol)}{n} \cdot 100\%$$

$$y = Ax - b$$
$$tol = 1\% = 0.01$$

Отобранные методы:

1. Метод сопряженных градиентов (Флетчер-Ривс) - **CG**
2. Метод Barzilai-Borwein+Polyak - **BB**
3. Метод **CDt** (Coordinate Descent + Nesterov's Tree + Parabolic Interpolation)
4. Метод **CD2t** (Coordinate Descent + Nesterov's Tree + идеи Хука-Дживса)
5. Метод **E2** (Градиентый метод Эйлера)
6. Метод **E3** (Трехточечная модификация градиентого метода Эйлера)

Метод Barzilai-Borwein+Polyak - BB

1. Задаётся значение параметра $k_\alpha^- < 1$.
2. $k = 0$, $x_k = x_0$, $f_k = f(x_k)$, $g_k = \nabla f(x_k)$.
3. Задается начальный шаг $\alpha = f(x_k)/||g_k||_2^2$ (“шаг по Поляку”).
4. $x_{k+1} = x_k - \alpha \cdot g_k$, вычисляется $f_{k+1} = f(x_{k+1})$.
5. Если $f_{k+1} < f_k$, то переход на пункт 8.
6. Если на итерации ещё не производился “шаг по Поляку”, то $\alpha^P = f(x_k)/||g_k||_2^2$. Если $\alpha^P < \alpha$, то $\alpha = \alpha^P$ и переход на пункт 4.
7. Уменьшается шаг $\alpha = \alpha \cdot k_\alpha^-$. Если $\alpha < \alpha_{min}$ — переход на пункт 16,
   иначе — переход на пункт 4.
8. Вычисляется $g_{k+1} = \nabla f(x_{k+1})$.
9. $s_{k+1} = x_{k+1} - x_k$
10. $y_{k+1} = g_{k+1} - g_k$

Метод Barzilai-Borwein-Polyak - BB(P)

11. Увеличивается счетчик итераций $k = k + 1$.
12. Обновляется счетчик времени t.
13. $\alpha = \langle s_k, s_k \rangle / \langle s_k, y_k \rangle$
14. Проверяются условия останова: $k \geq k_{max}$, $t \geq t_{max}$, $f_k \leq f^*$, $||g_k||_2 \leq \varepsilon_g$ и т.д. Если какое-либо из условий выполняется — переход на пункт 16.
15. Переход на пункт 4.
16. Конец работы алгоритма.

Метод Barzilai-Borwein-Polyak BB(P)

На каждой итерации алгоритма, таким образом, вычисляется по 1 значению функции и градиента в случае “удачного” шага метода BB. В случае “промаха” производится попытка “шага по Поляку” и дальнейшее уменьшение шага, в случае необходимости. Вычислительные эксперименты показали высокую эффективность данного алгоритма, что связано с “легкостью” итерации по сравнению с другими методами. Применение “шага по Поляку” позволяет в ряде случаев ускорить сходимость предложенной модификации.

Метод CDt

1. Случайным образом выбирается оптимизируемая переменная.
2. Делаются 2 дополнительные пробы по выбранному направлению-координате.
3. На основе существующих 3-х точек строится параболическая аппроксимация минимизируемой функции, находится минимум этой аппроксимации.
4. Осуществляется переход в точку минимума параболы.
5. Проверяются условия останова алгоритма, в случае выполнения – переход на пункт 7.
6. Переход на пункт 1.
7. Конец работы алгоритма.

Двухметодная схема

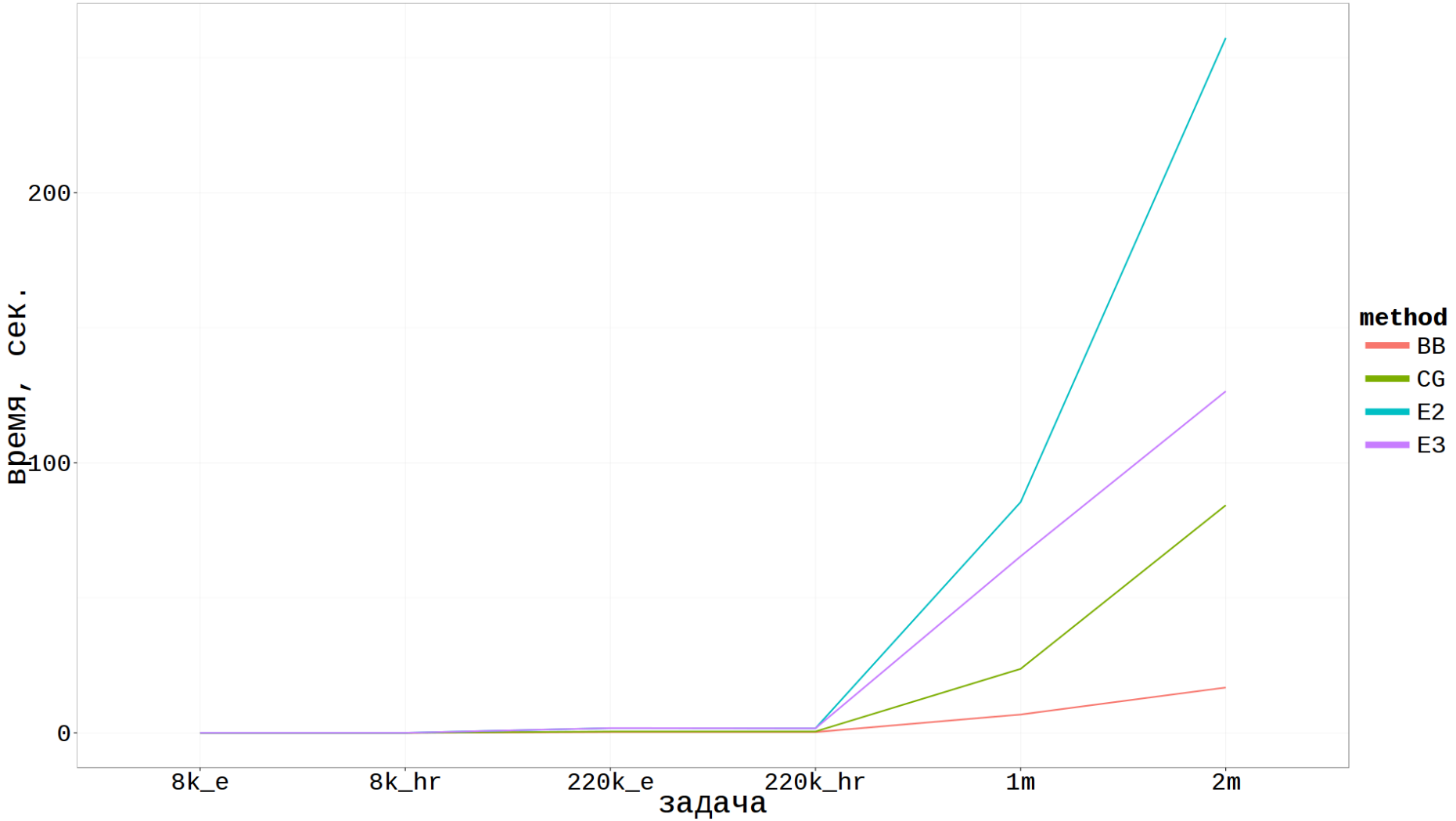
Для максимизации значения метрики LLA_{max} применена следующая вычислительная схема:

1. Устанавливается значение параметра $\gamma = 1$, $x_0 = 0$.
2. Выполняется минимизация полученной модели градиентным методом. Условие останова алгоритма — $\text{LLA}_{max} \geq 10\%$. Результат минимизации – точка x^*_G.
3. Устанавливается значение параметра $\gamma = 1/||x^*_G||^2_2$.
4. Выполняется минимизация полученной модели покоординатным методом. В качестве стартовой точки используется x^*_G. Условие останова алгоритма – $\text{LLA}_{max} = 100\%$. Результат минимизации – точка x^*_{CD}.

Задача	Размерность
8k_e	8000
8k_hr	8000
220k_e	220000
220k_hr	220000
1m	971210
2m	2000000

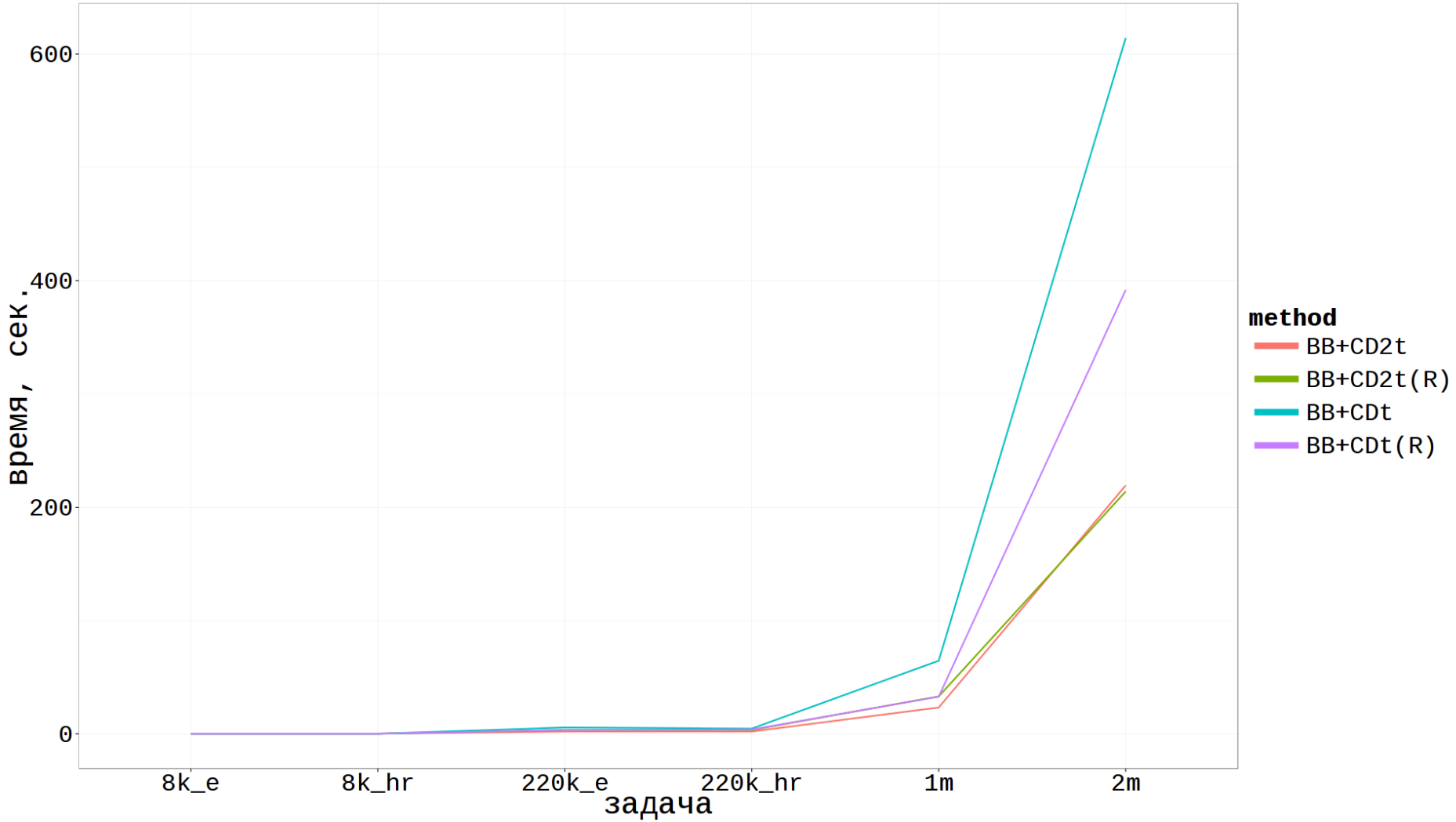
Первый этап: LLA$max = 0\% \to 10\%$

Оптимизация моделей градиентными методами.

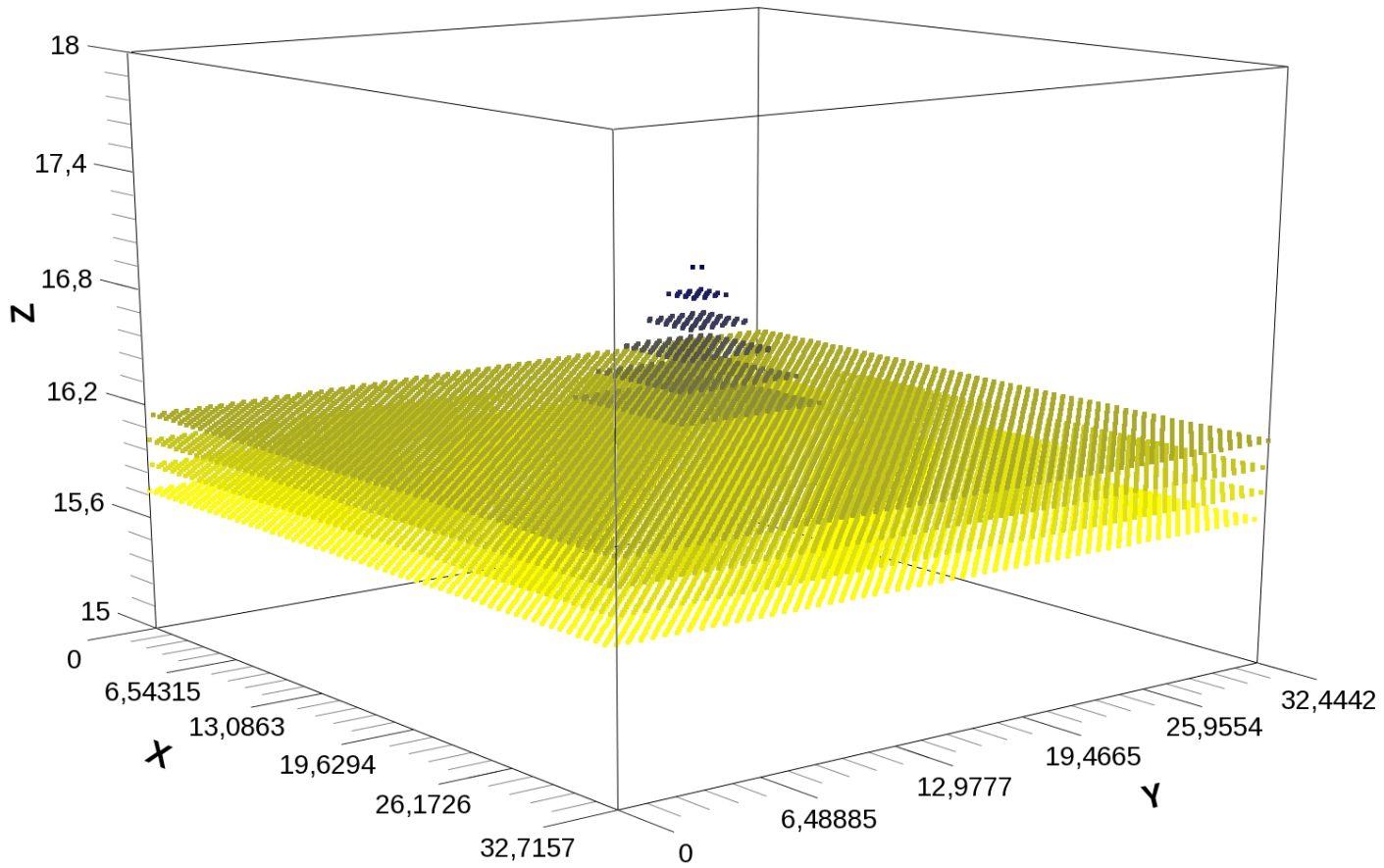


Второй этап: LLA$max = 10\% \to 100\%$

Оптимизация моделей покоординатными методами.



Вид квантовой точки Si-Ge



Потенциальная функция Китинга

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

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

n - число атомов в кристаллической решетке;
d_{ij},d_{ik}, α_{ij}, β_{ijk} - заданные константы;
$r_i = (x_i, x_{2i}, x_{3i})$ – радиус-вектор i-го узла (оптимизируемые переменные).

Особенности рассматриваемой задачи

- Высокие размерности – 10^5 переменных и выше.
- Высокие требования к точности результата .

Протестированные методы оптимизации

- Метод Коши
- Метод сопряженных градиентов
- Метод Ньютона

Сложности реализации метода Ньютона

- Размерность рассматриваемых задач - физические ограничения на размер матрицы Гессе, налагаемые объемом доступной оперативной памяти.
- Высокая вычислительная сложность - длительное время решения задач требуемой размерности.

Матрица Гессе

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

Для хранения плотной матрицы требуется порядка n^2 ячеек памяти.

Разреженная матрица Гессе

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

Для хранения разреженной матрицы требуется меньше, чем n^2 ячеек памяти.

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.

Используемый метод хранения разреженной матрицы

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 – максимальное число ненулевых элементов в строке матрицы Гессе. Для рассматриваемой задачи $L = 51$.

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

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

Соотношение размеров плотной и разреженных матриц

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	$727.6\ Tb$	$5.7\ Gb$	$7.65 \cdot 10^{-6}$

$M = n^2 \cdot 8$ – размер плотной матрицы (в байтах).

$M_{sparse} = n \cdot 51 \cdot (8 + 4)$ – размер разреженной матрицы (в байтах).

Особенности используемого формата хранения

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

Решение задачи линейной алгебры

Для решения задачи линейной алгебры в реализации метода Ньютона с использованием разреженной матрицы используется модификация метода сопряженных градиентов.

В случае, если разреженная матрица не обладает свойством положительной определенности, запускается эвристический алгоритм, манипулирующий элементами главной диагонали.

Вычислительные эксперименты

Модификация метода Ньютона

n	Значение функции		t,	Число
	до оптимизации	после оптимизации	сек	итераций
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

Вычислительные эксперименты

Метод сопряженных градиентов

n	Значение функции		t,	Число
	до оптим-ии	после оптим-ии	сек	итераций
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

Тесты проводились на вычислительной системе, содержащей 10 ядер Intel Xeon X5670.

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

- Химия
- Физика
- Материаловедение
- (Нано)электроника
- (Нано)биология
- Фармацевтика
- Военные технологии
- ...

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.

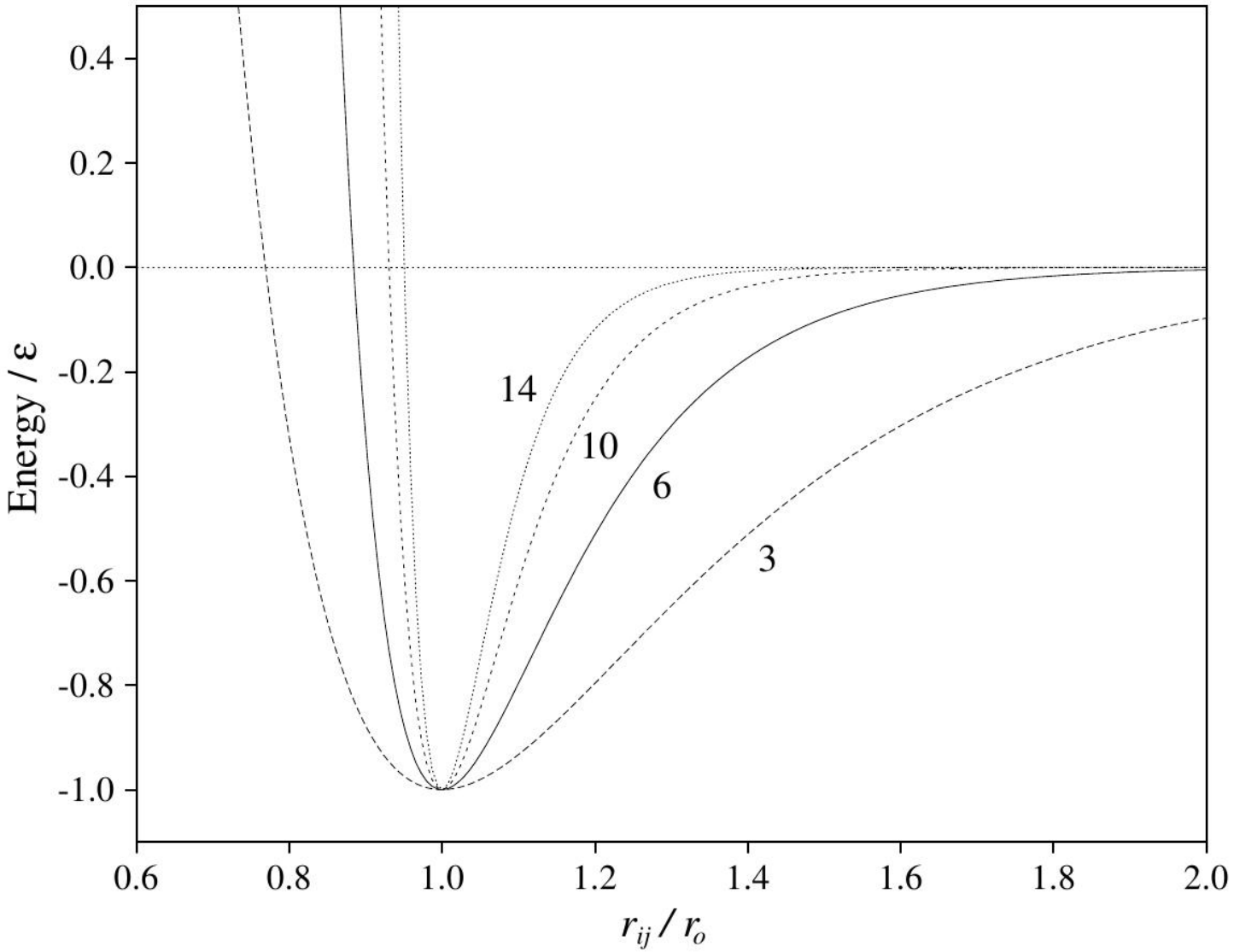
The Cambridge Energy Landscape Database
(The Cambridge Cluster Database) :

http://www-wales.ch.cam.ac.uk/CCD.html

Потенциальная функция Морса

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

Потенциальная функция Морса с различным ρ



- 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)

The Cambridge Cluster Database

D. J. Wales, J. P. K. Doye, A. Dullweber, M. P. Hodges, F. Y. Naumkin
F. Calvo, J. Hernández-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

apps.uc.pt/mypage/faculty/qtmarque/en/clusters

Примененные методы оптимизации (локальные)

Основные (универсальные) :

- Conjugate gradient (варианты, Neculai Andrei)
- L-BFGS

Дополнительные (доводчики) :

- Cauchy
- Powell

Специальные :

- Polyak

Примененные методы оптимизации (глобальные)

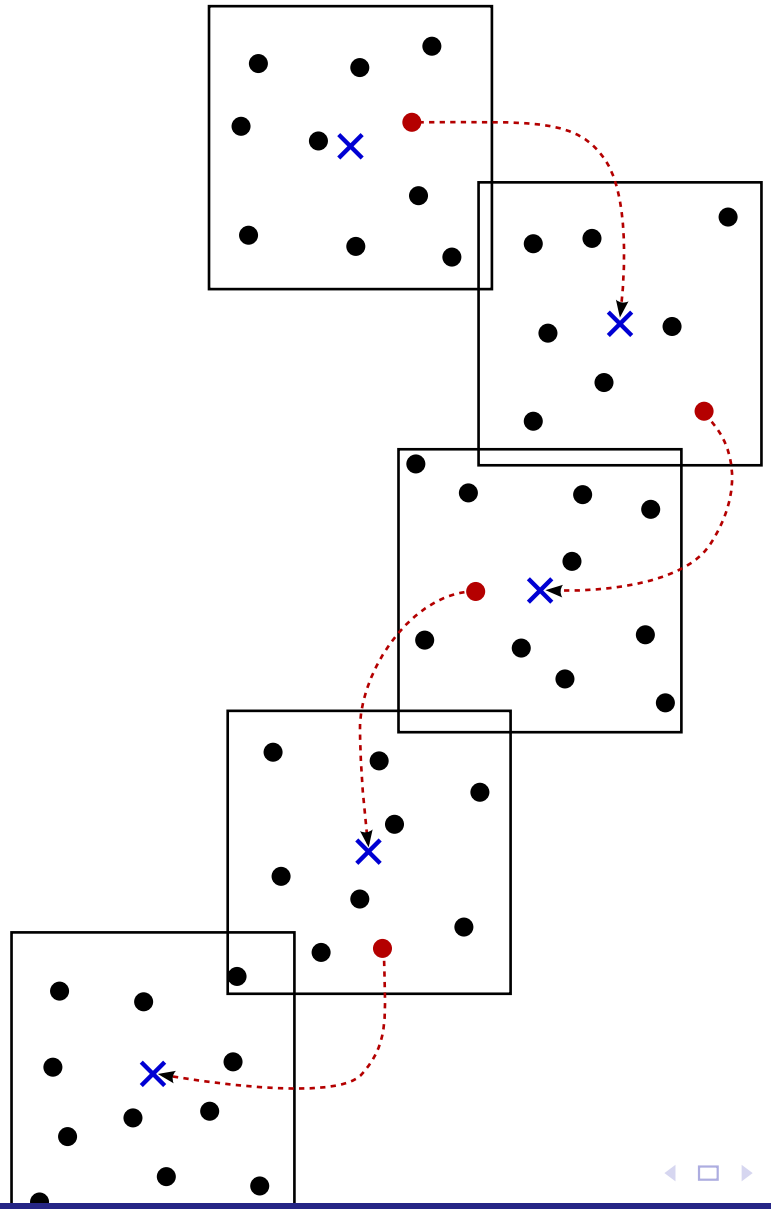
- Multi-start
- MSBH - Monotonic Sequential Basin-Hopping
- “Big-Bang”
- “Forest”

Реализация

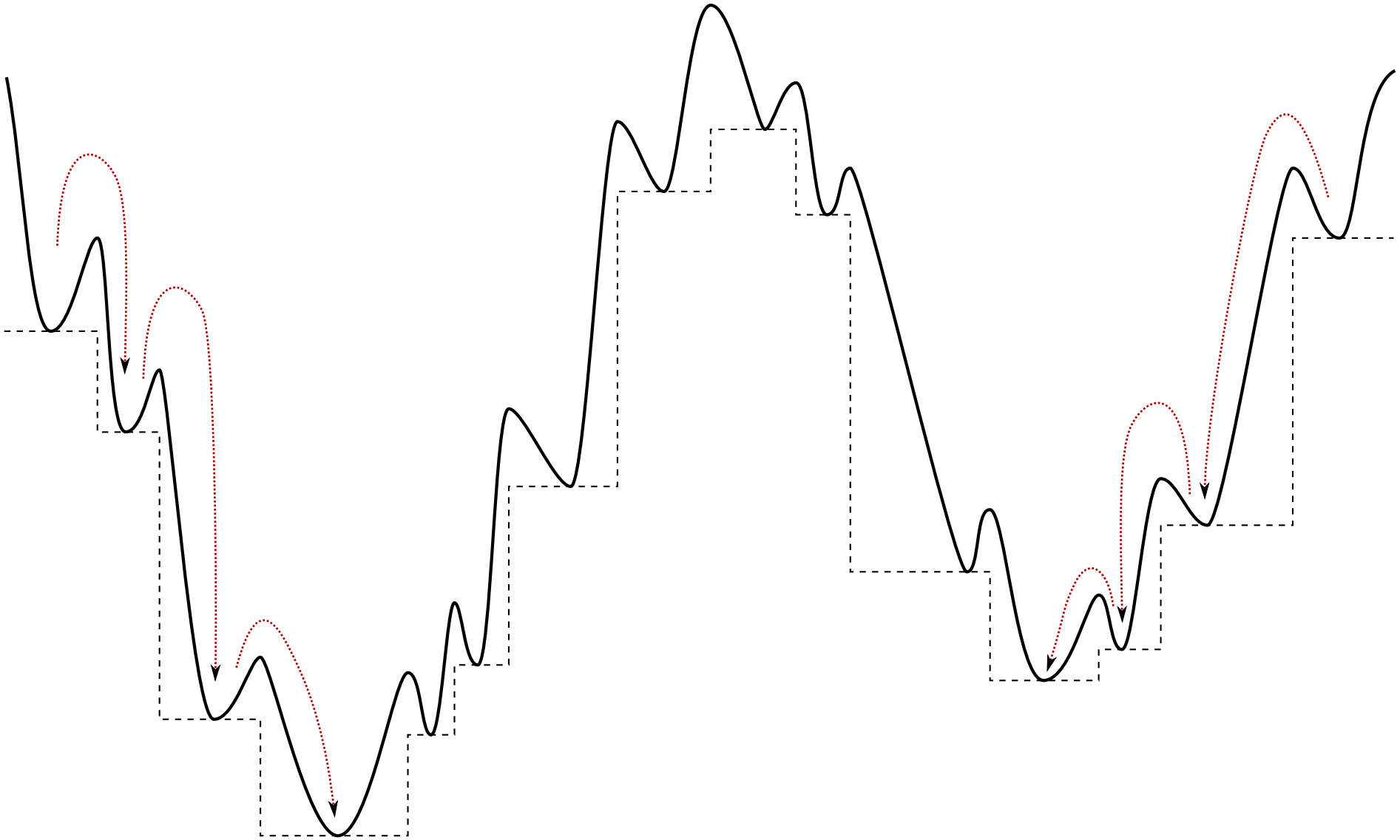
Программный комплекс OPTCON-M:

- Язык программирования C/C++ (компиляторы GCC/MinGW, Clang, ICC)
- ОС Linux, Mac OS X, Windows
- Технологии параллельного программирования OpenMP, MPI, CUDA
- Интерактивный / пакетный режим работы

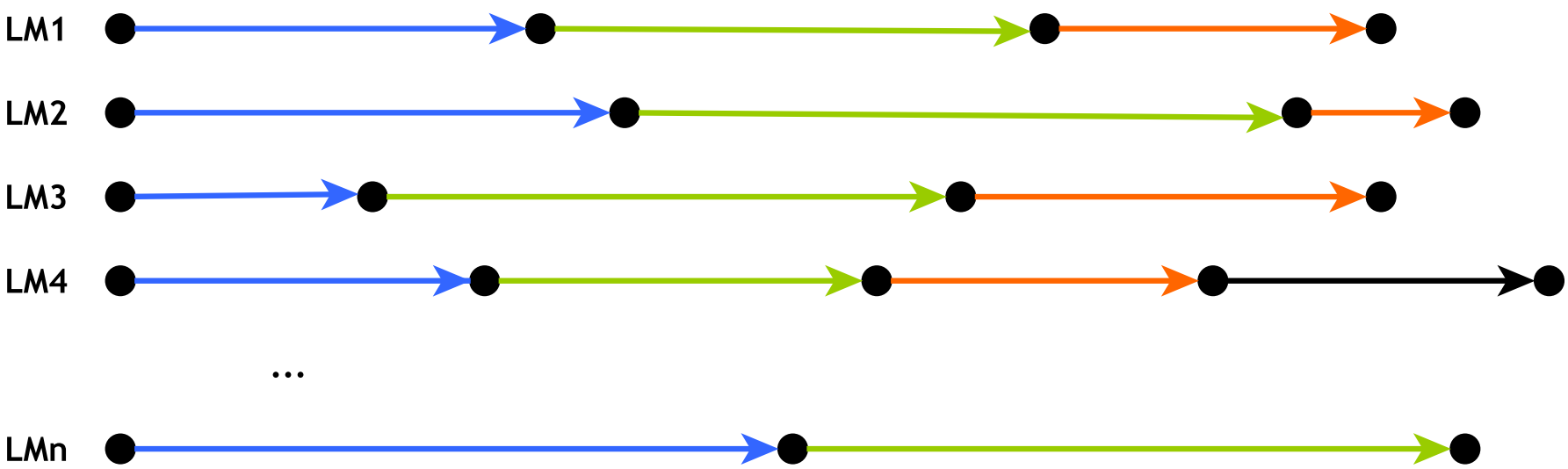
Метод MSBH



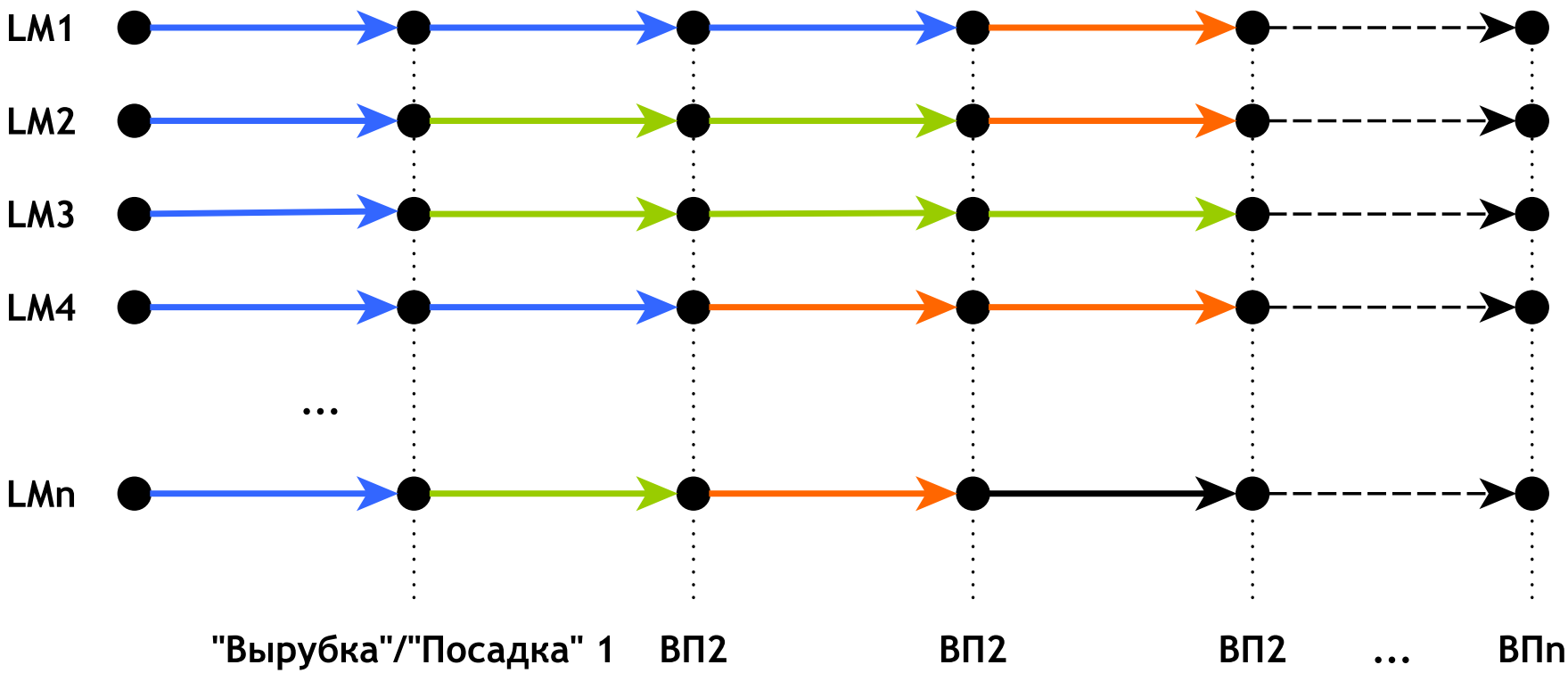
Метод MSBH



Метод “Forest”



Метод “Forest”



Метод “Forest”

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
- ...

Метод “Forest”

- 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, ...)

Вычислительные эксперименты. MSBH/“Forest”

n - число атомов

n	UK (CCD)	ISDCT
20	-97.417393	-97.41739307417
80	-690.577890	-690.577890200415 5952
147	-1531.498857	-1531.498857189995761

Вычислительные эксперименты. MSBH/“Forest”

n - число атомов

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

Вычислительные эксперименты. MSBH/“Forest”

n - число атомов

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

Вычислительные эксперименты. MSBH/“Forest”, n = 240

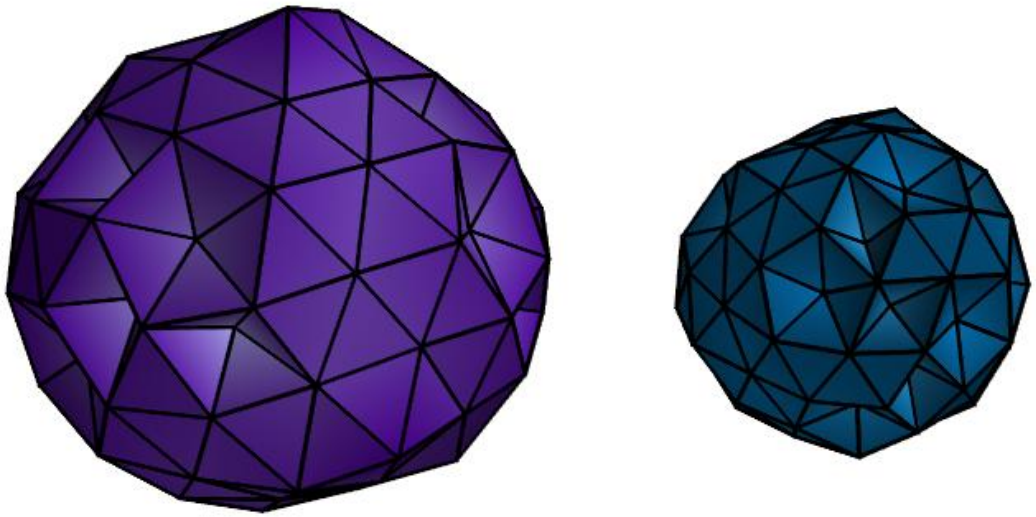
CN	-2839.054430
PT	-2839.099925
ISDCT	-2839.099924748702961

Вычислительные эксперименты. “Forest”

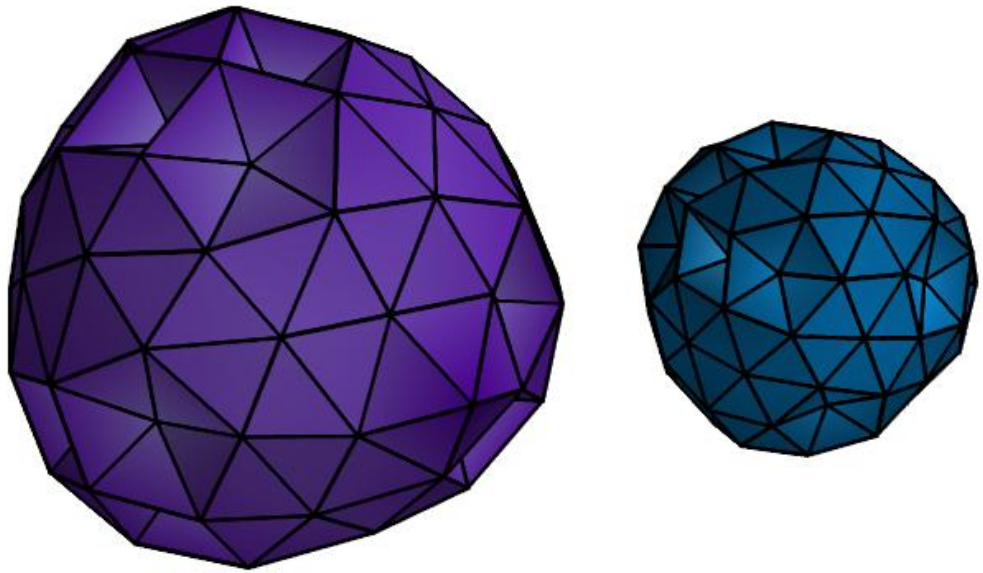
n - число атомов

n	ISDCT
241	-2852.938110154795595
242	-2866.778881123787869
243	-2882.570361711906116
244	-2897.072046040393616
245	-2910.707949591107536
246	-2924.517707573666030
247	-2940.293677679405846
248	-2955.679013678213323
249	-2971.203337281702716
250	-2985.771711424368277
251	-2999.469988809987626
252	-3013.550134756378611

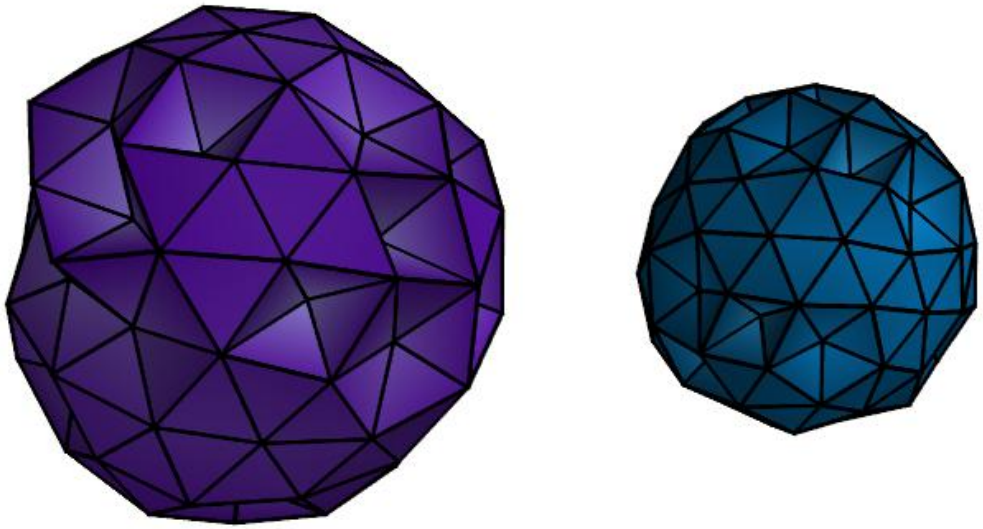
Кластер из 150 атомов; CN, ISDCT



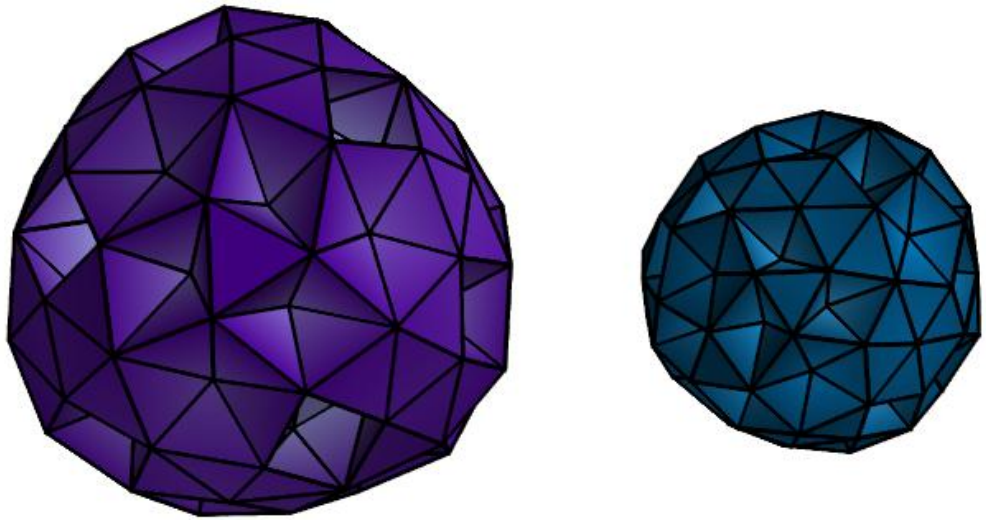
Кластер из 155 атомов; CN, ISDCT



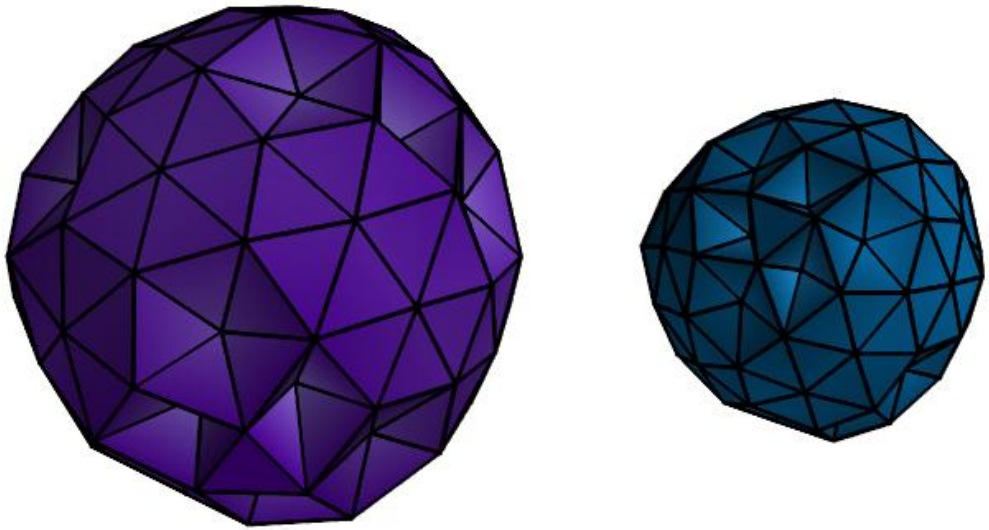
Кластер из 160 атомов; CN, ISDCT



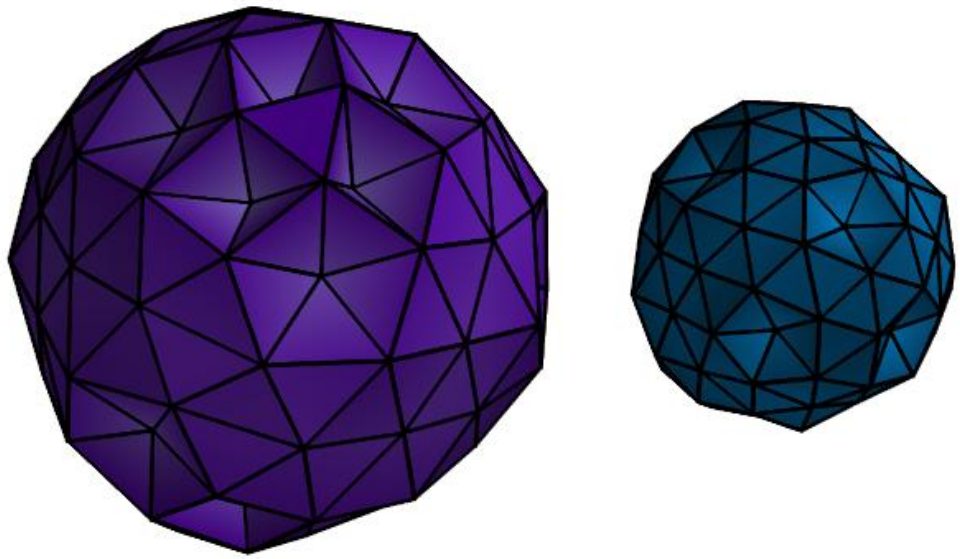
Кластер из 165 атомов; CN, ISDCT



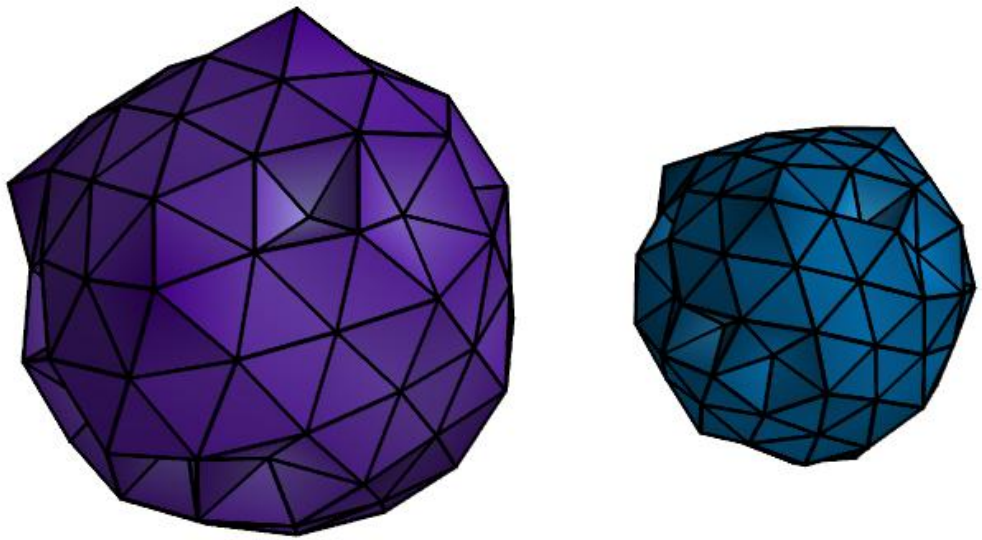
Кластер из 170 атомов; CN, ISDCT



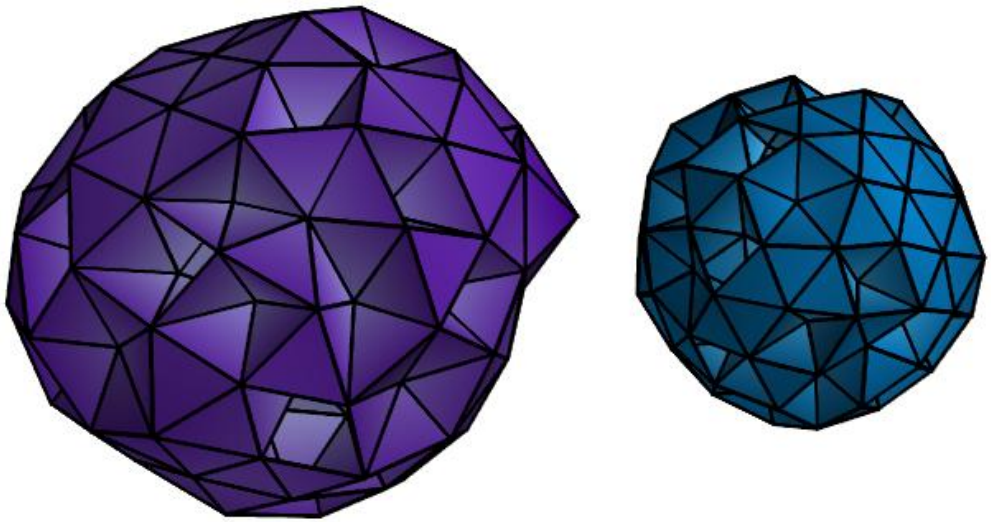
Кластер из 175 атомов; CN, ISDCT



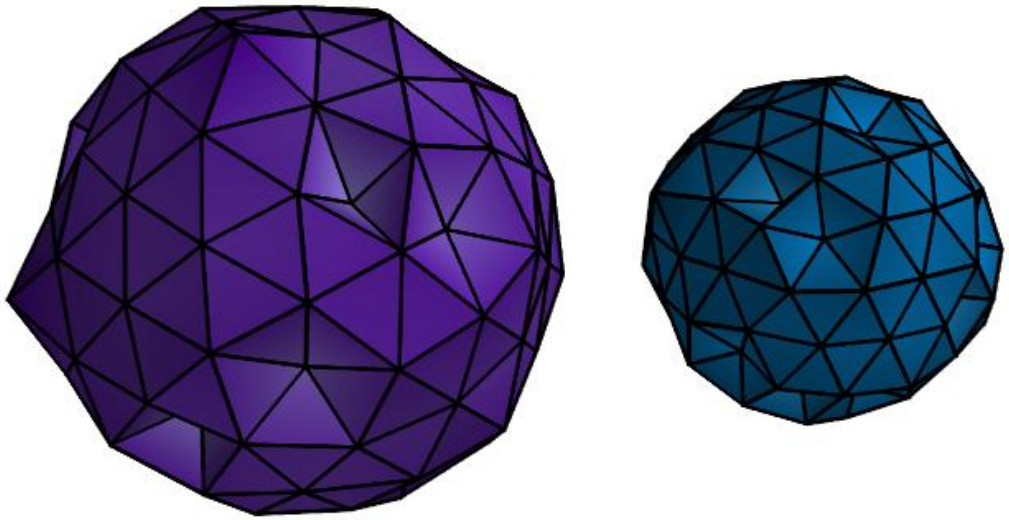
Кластер из 180 атомов; CN, ISDCT



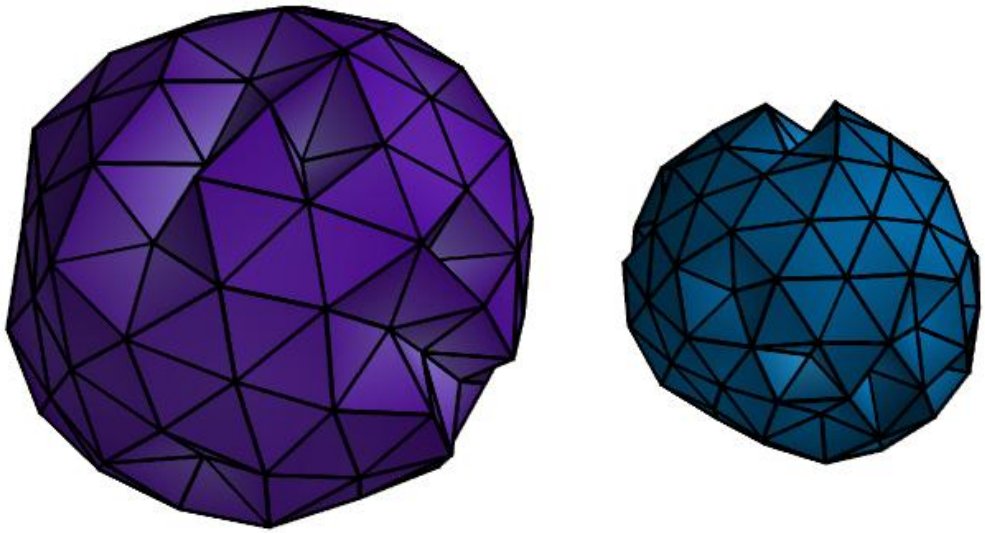
Кластер из 185 атомов; CN, ISDCT



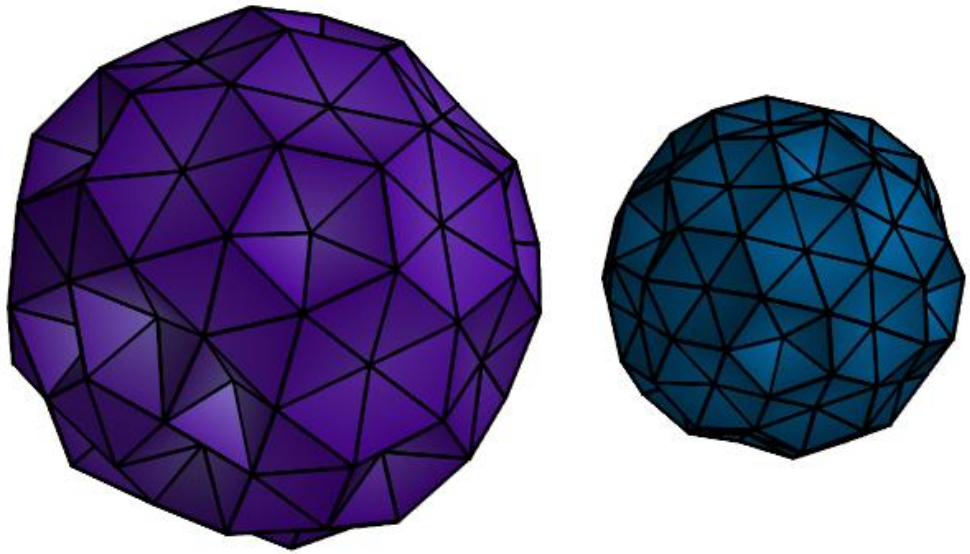
Кластер из 195 атомов; CN, ISDCT



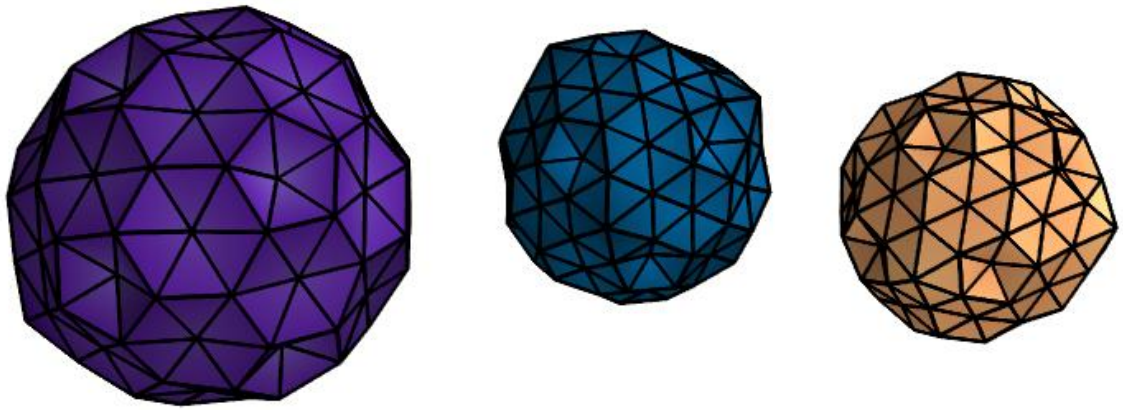
Кластер из 200 атомов; CN, ISDCT



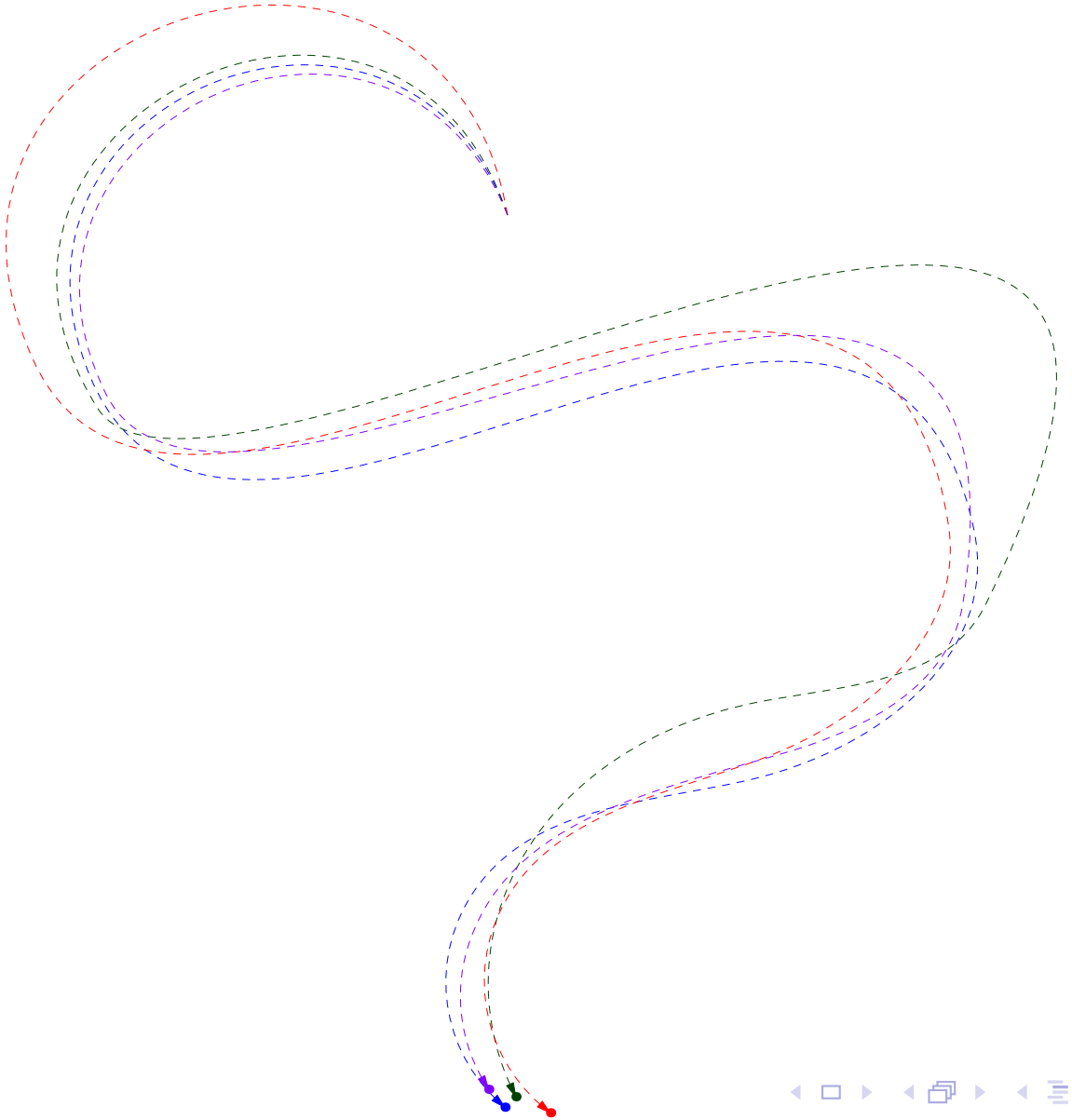
Кластер из 230 атомов; CN, ISDCT



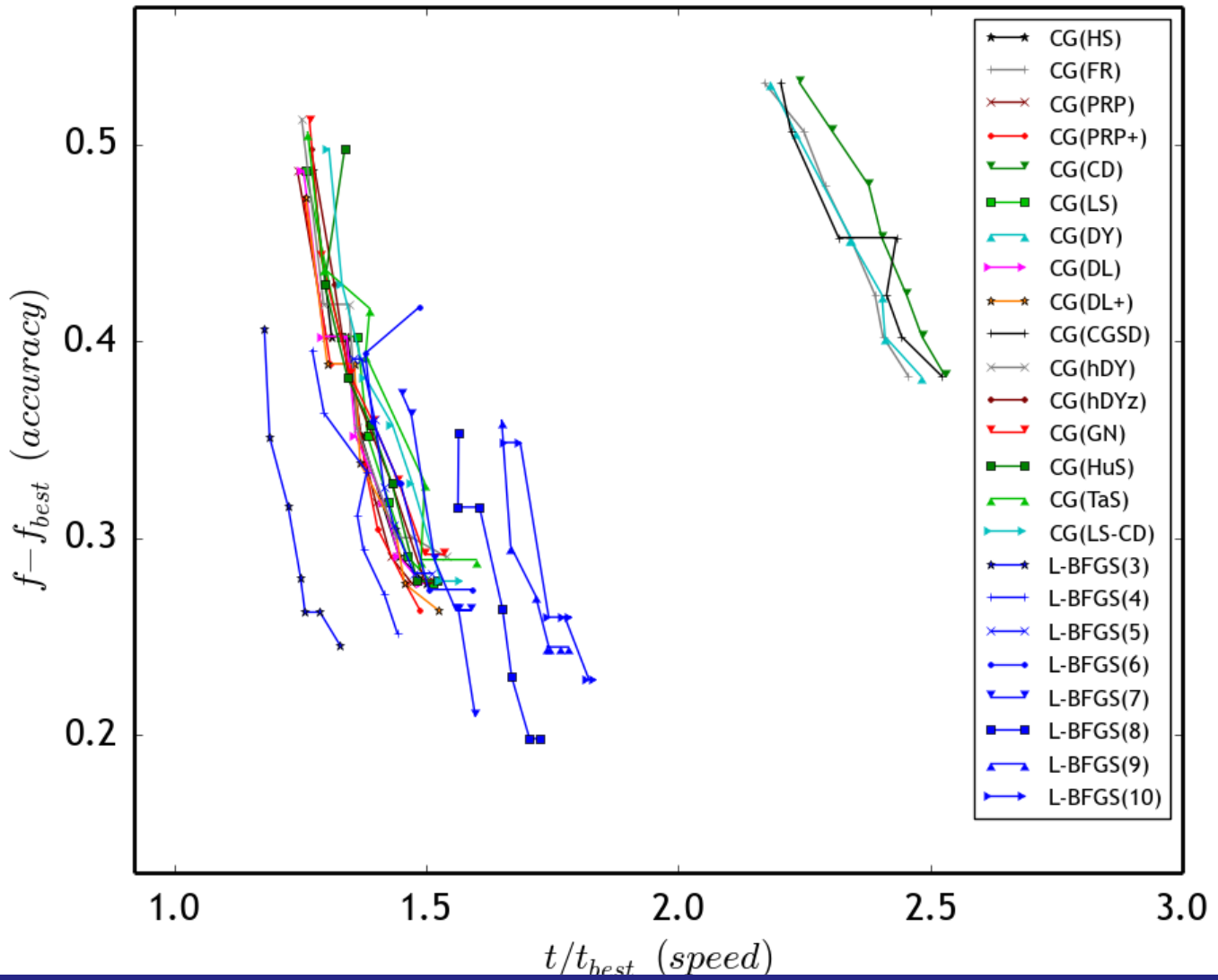
Кластер из 240 атомов; CN, ISDCT, PT



Тестирование локальных методов



Тестирование локальных методов



$$\dot{x} = f(x(t), u(t), t)$$

$$x_0 = x(t_0)$$

$$t \in [t_0, t_1]$$

$$\underline{u} \leq u(t) \leq \overline{u}$$

$$I(u) = \phi(x(t_1)) \to \min$$

Аппроксимативная задача

$$x(t+h) = f_h(x(t), u(t), t, h)$$

$$N = (t_1 - t_0)/h$$

$$\dot{x_1} = x_2$$

$$\dot{x_2} = u - sin(x_1)$$

$$t \in [0, 5]$$

$$x_1(0) = 5$$

$$x_2(0) = 0$$

$$|u(t)| \leq 1$$

$$I(u) = x_1^2(5) + x_2^2(5) \to \min$$

Для решения задач Коши использовался метод интегрирования из семейства Рунге-Кутты 2 порядка (т.н. метод “Эйлера с итерациями”). Расчет прерывался при выполнении условия $I(u^{k-1}) - I(u^k) < 10^{-6}$. Все задачи решались, начиная с одинаковых начальных приближений $u^0(t) \equiv -0.5$. В этой задаче, очевидно, известна гарантированная, но неоптимальная нижняя оценка значения функционала, равная 0.

Эксперимент выполнялся с использованием стандартного программного комплекса OPTCON-A.

Метод Поляка

Поляк Б.Т.
Минимизация негладких функционалов // Журнал вычислительной математики и вычислительной физики. 1969. Т. 9. № 3. С. 509–521.

Метод Поляка

$$x_{i+1} = x_i - \frac{f(x_i) - f^*}{\|\nabla f(x_i)\|^2} \cdot \nabla f(x_i)$$

f^* — известное минимальное значение функционала

Метод Поляка

Для одномерного случая метод совпадает с методом Ньютона для решения уравнения $f(x) = f^*$.

Модификация метода Поляка

$$x_{i+1} = x_i - K \cdot \frac{f(x_i) - f^*}{\|\nabla f(x_i)\|^2} \cdot \nabla f(x_i)$$

Модификация метода Поляка

Предлагается задать $K = 2$. В случае невозможности спуска (отсутствии улучшения по направлению) на очередной итерации — задать $K = 1$ (исходный метод) и повторить попытку спуска.

Сходимость предлагаемой модификации обеспечивается включением в неё исходного метода с доказанной сходимостью.

Модификация метода Поляка

Поскольку в данной модификации есть вероятность “отката” к предыдущему состоянию, текущая программная реализация требует удвоенного объема памяти, по сравнению с исходным методом Поляка.

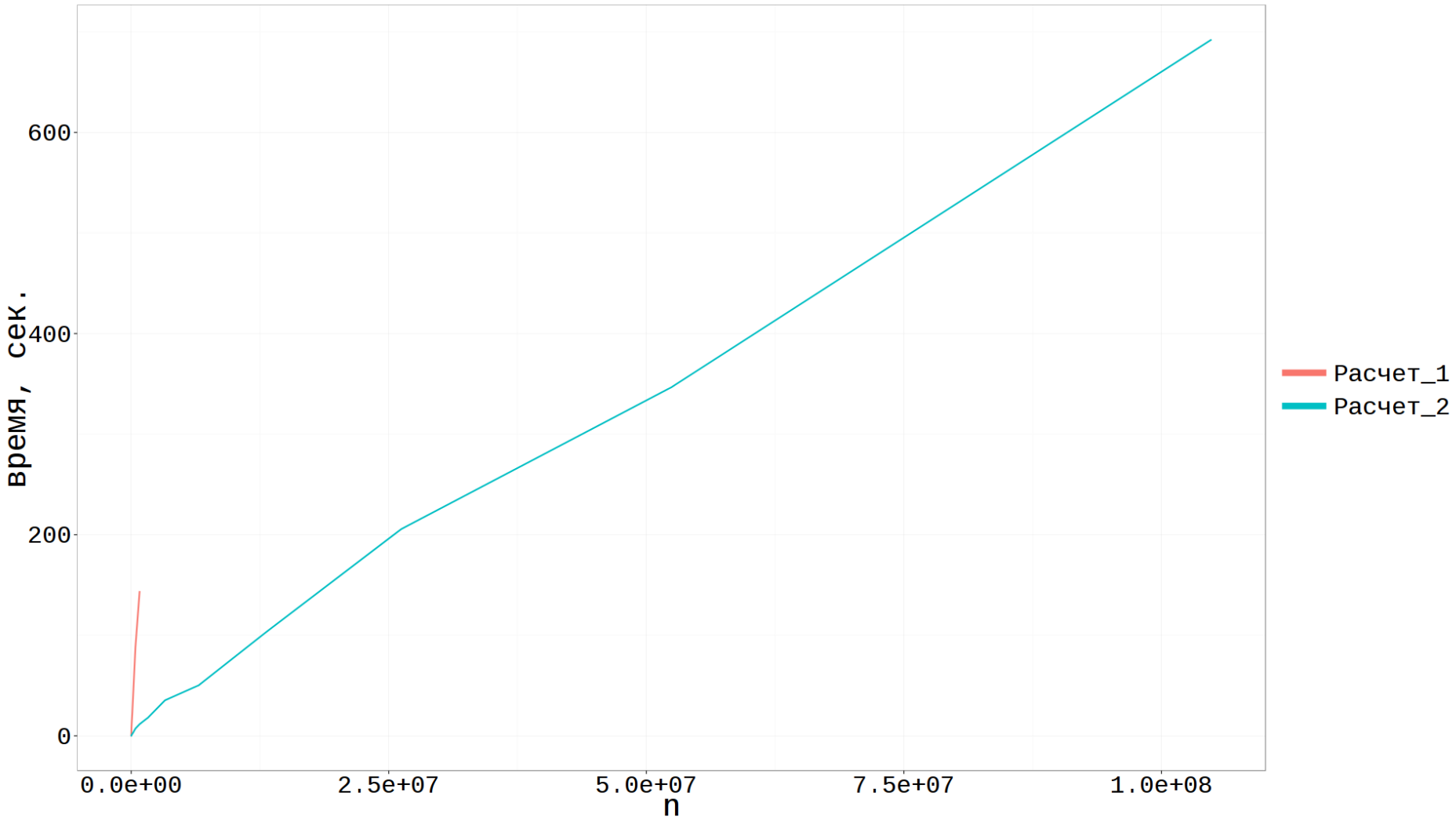
Время решения (сек.) задачи оптимального управления программным комплексом OPTCON-A. N - число узлов сетки дискретизации.

N	$I(u^*)$	t, сек.	з.Коши	$\|\|g\|\|_2$	итераций
101	1.191521713372e+01	0	135	5.7e-3	15
201	1.190999877810e+01	0	317	2.4e-4	23
401	1.190841888965e+01	0	276	8.0e-5	24
801	1.190817286834e+01	0	182	2.2e-4	21
1601	1.190804296061e+01	0	241	1.2e-5	21
3201	1.190804923806e+01	0	167	6.8e-5	17
6401	1.190801439789e+01	1	109	1.8e-5	16
12801	1.190805357054e+01	3	206	3.5e-5	19
25601	1.190806771202e+01	8	237	4.7e-5	21
51201	1.190812331446e+01	13	203	4.4e-5	19
102401	1.190811071137e+01	22	167	3.8e-5	17
204801	1.190815091482e+01	44	166	4.0e-5	17
409601	1.190814768992e+01	88	166	2.9e-5	17
819201	1.190847626971e+01	144	131	2.9e-5	15

Второй эксперимент производился на специальной программной реализацией алгоритма, проверенной с использованием нескольких популярных компиляторов на нескольких различных компьютерах. В эксперименте участвовали следующие пары “процессор-компилятор”:

1. Intel Core i7-2640M; Linux; icc-15.0.1, gcc-4.8.2, gcc-4.9.1, clang-3.4.2, clang-3.5.0
2. Intel Core i7-2640M; Mac OS X; gcc-4.8.3, clang-3.4.2
3. Intel Core i5 4260U; Mac OS X; gcc-4.8.3, clang-600.0.56 (based on LLVM 3.5svn)
4. AMD Opteron 6220; Linux; icc-14.0.0

N	AMD 6220 Linux	i7-2640M Linux			i7-2640M Mac OS X		i5 4260U Mac OS X	
	icc 14.0.0	icc 15.0.1	gcc 4.8.2	clang 3.4.2	gcc 4.8.3	clang 3.4.2	gcc 4.8.3	clang 3.5SVN
101	0.006	0.001	0.006	0.009	0.001	0.001	0.002	0.002
201	0,014	0,003	0,012	0,005	0,003	0,003	0,004	0,004
401	0,018	0,007	0,014	0,014	0,006	0,006	0,009	0,008
801	0,033	0,010	0,023	0,028	0,011	0,010	0,015	0,015
1601	0,019	0,010	0,019	0,014	0,006	0,007	0,008	0,008
3201	0,085	0,031	0,065	0,070	0,044	0,034	0,045	0,046
6401	0,078	0,041	0,099	0,100	0,046	0,051	0,066	0,069
12801	0,289	0,159	0,271	0,311	0,156	0,154	0,220	0,255
25601	0,674	0,298	0,620	0,687	0,366	0,380	0,500	0,495
51201	1,030	0,521	1,060	1,234	0,627	0,613	0,870	0,942
102401	1,890	0,887	1,799	2,023	1,142	1,160	1,505	2,100
204801	3,546	1,765	4,415	4,015	2,352	2,319	2,866	3,044
409601	7,251	3,576	7,133	8,034	4,588	4,511	5,509	5,938
819201	11,662	5,714	11,608	12,883	7,447	7,315	8,752	8,682
1638401	18,197	8,929	18,203	20,443	11,588	11,381	13,547	13,313
3276801	35,353	17,975	35,947	40,622	22,972	25,862	27,110	26,504
6553601	50,279	26,991	52,610	59,321	33,680	33,090	39,626	39,432
13107201	103,237							
26214401	205,603							
52428801	346,577							
104857601	692,295							


600
400
200
0
время, сек.
0.0e+00
2.5e+07
5.0e+07
7.5e+07
1.0e+08
n
Расчет_1
Расчет_2

Проведенные эксперименты показали принципиальную возможность решения аппроксимативных задач оптимального управления с размерностями, превышающими 10^8 даже без применения параллельных вычислительных технологий.

Тестовая задача оптимизации (1)

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

Тестовая задача оптимизации (1)

- Выпуклая
- Сепарабельная
- $f_{min} = 0$ (заранее известно)

Программная реализация метода

Вычисление значение функции $f(x)$ и её градиента $\nabla f(x)$ выполняется параллельно на различных ядрах CPU, т.к. для задач большой размерности это невозможно на одном вычислительном узле из-за физических ограничений на объем ОЗУ

Вычислительные эксперименты

Вычислительные эксперименты проводились на:

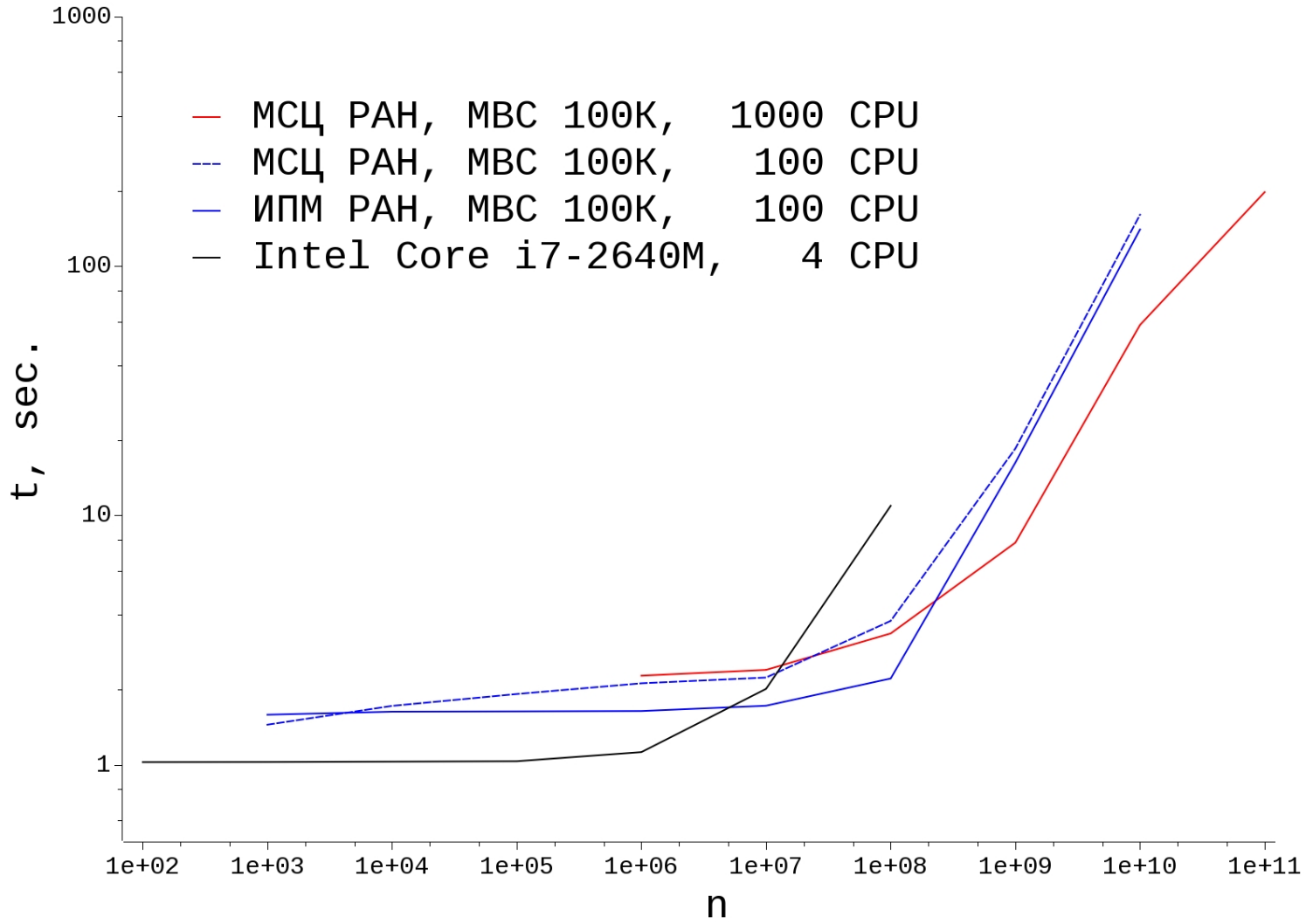
- Вычислительном кластере МВС-100К Межведомственного Суперкомпьютерного Центра.
  ОЗУ — 1 ГБ / 1 CPU.
- Вычислительном кластере МВС-100К ИМП РАН.
- Вычислительном кластере “Академик В.М. Матросов”, узел “Tesla”.
  ОЗУ — 250 ГБ, 32 CPU.

Вычислительные эксперименты (1)

Время работы алгоритма, сек.

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

Вычислительные эксперименты (1)



Требуемая память (1 core)

n	ОЗУ	
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

Требуемая память (1000 cores)

n	ОЗУ		ОЗУ / 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

Вычислительные эксперименты (1)

- Хорошая масштабируемость предложенной реализации
- Основной лимитирующий фактор - объем доступной ОЗУ
- Высокая эффективность для сепарабельных задач (малый объем данных для обмена между узлами)

Тестовая задача оптимизации (2)

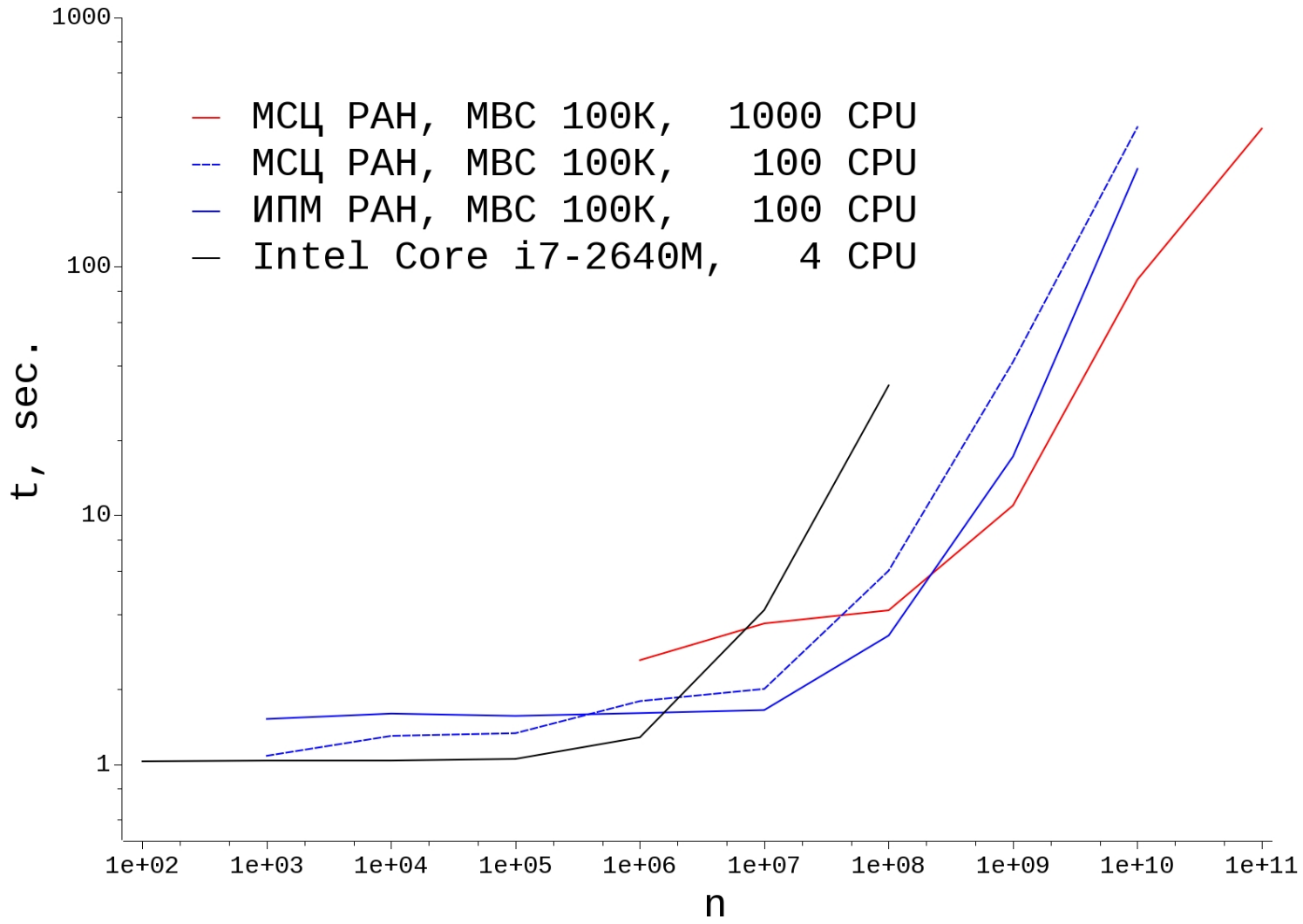
$$f(x) = \sum_{i=1}^{n} x_i^2 + \sum_{i=2}^{n} (x_i - x_{i-1})^2$$

Вычислительные эксперименты (2)

Время работы алгоритма, сек.

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

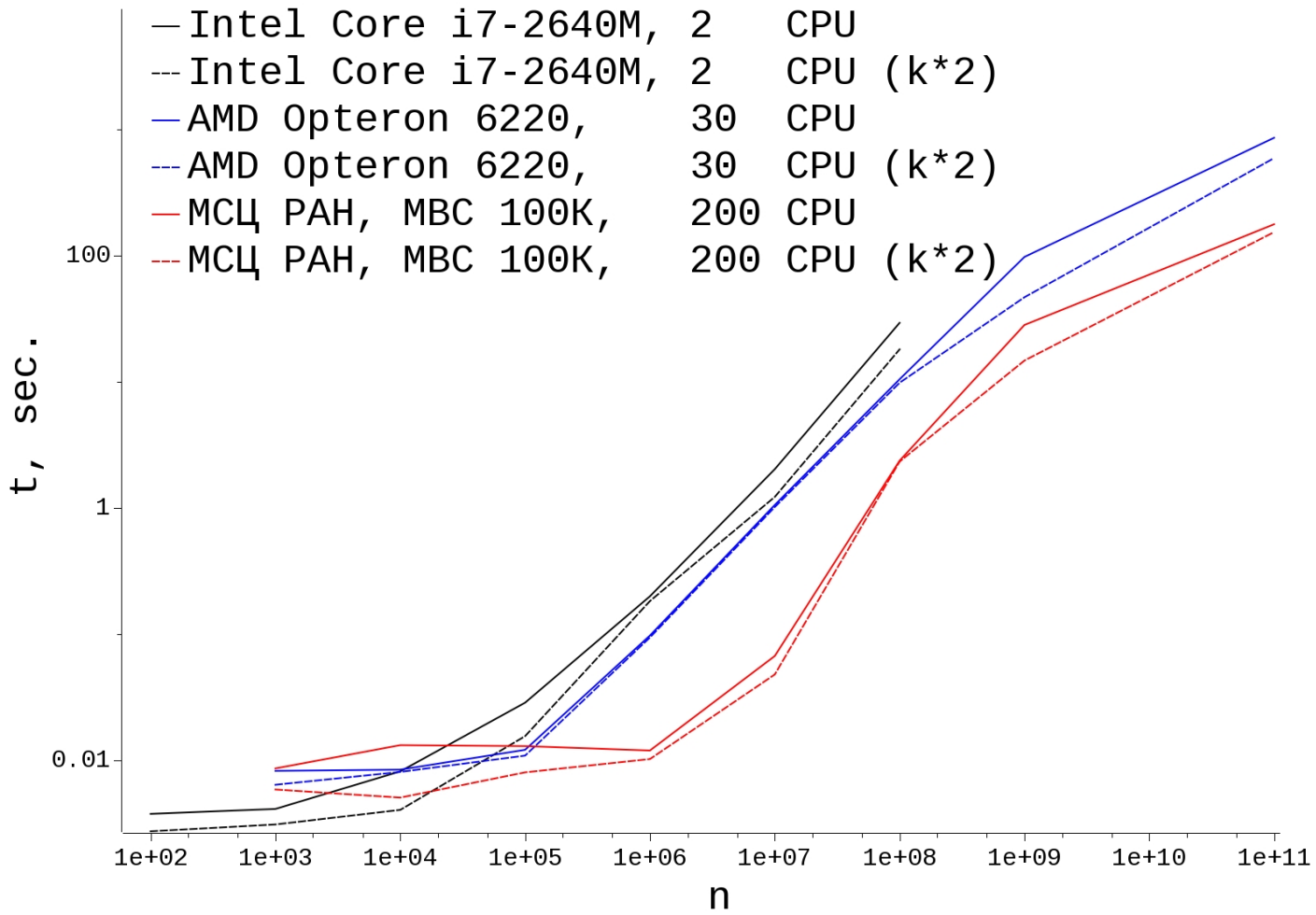
Вычислительные эксперименты (2)



Вычислительные эксперименты, задача 2
Модификация метода Поляка; время работы, сек.

n	Core i7, 2 CPU		Matrosov, 30 CPU		МСЦ, 200 CPU	
	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

Вычислительные эксперименты, задача 2
Модификация метода Поляка; время работы, сек.



Заключение

- Задачи оптимизации больших размерностей, рекордные в своем классе, могут быть решены с использованием предложенных подходов.
- Удивительно конкурентоспособные результаты показывают “простые” алгоритмы.
- Границы возможностей решения “больших” задач оптимизации существенно шире, чем ожидалось.

1. Задача нахождения PageRank-вектора
2. Транспортная задача
3. Оптимизация потенциала Китинга
4. Оптимизация потенциала Морса
5. Задача оптимального управления
6. (Квази)сепарабельные задачи выпуклой оптимизации класса Huge-Scale

Спасибо за внимание

Алгоритмы решения сверхбольших задач непрерывной оптимизации

Горнов А.Ю.[1]
gornov@icc.ru,

Аникин А.С.[1]
Андрианов А.Н.[2]
Гасников А.В.[3]

[1]ИДСТУ СО РАН, Иркутск

[2]ИПМ РАН, Москва

[3]ПреМоЛаб МФТИ, Москва

ММРО 2015