AlEM: новый параллельный алгоритм линейного программирования для кластерных вычислительных систем

Жулев Александр Эдуардович, Соколинский Леонид Борисович

Южно-Уральский государственный университет
(национальный исследовательский университет)

Челябинск

Задача линейного программирования

$$\overline{\boldsymbol{x}} = argmax\{\boldsymbol{f}(\boldsymbol{x})|A\boldsymbol{x} \leq \boldsymbol{b}\}$$

$\boldsymbol{x} \in \mathbb{R}^n$

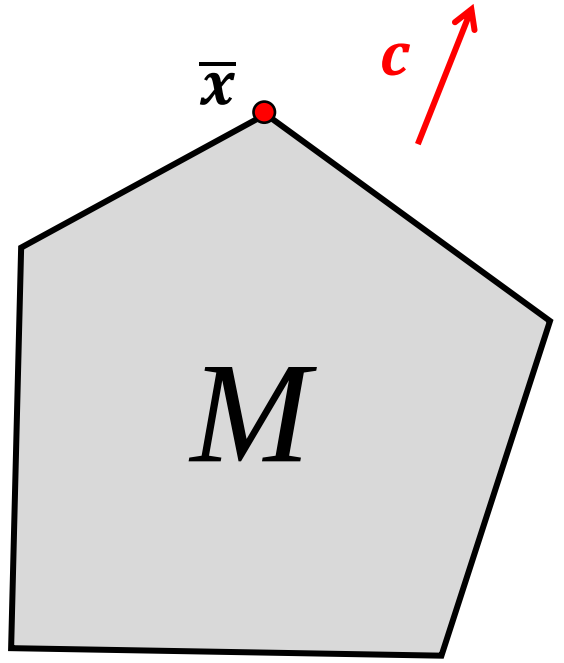
A – матрица $m \times n$

$\boldsymbol{b}$ – вектор размерности m

$\boldsymbol{c}$ – вектор размерности n

$\boldsymbol{f}(\boldsymbol{x}) = \langle \boldsymbol{c}, \boldsymbol{x} \rangle$ – целевая функция

$\langle \boldsymbol{c}, \boldsymbol{x} \rangle$ – скалярное произведение

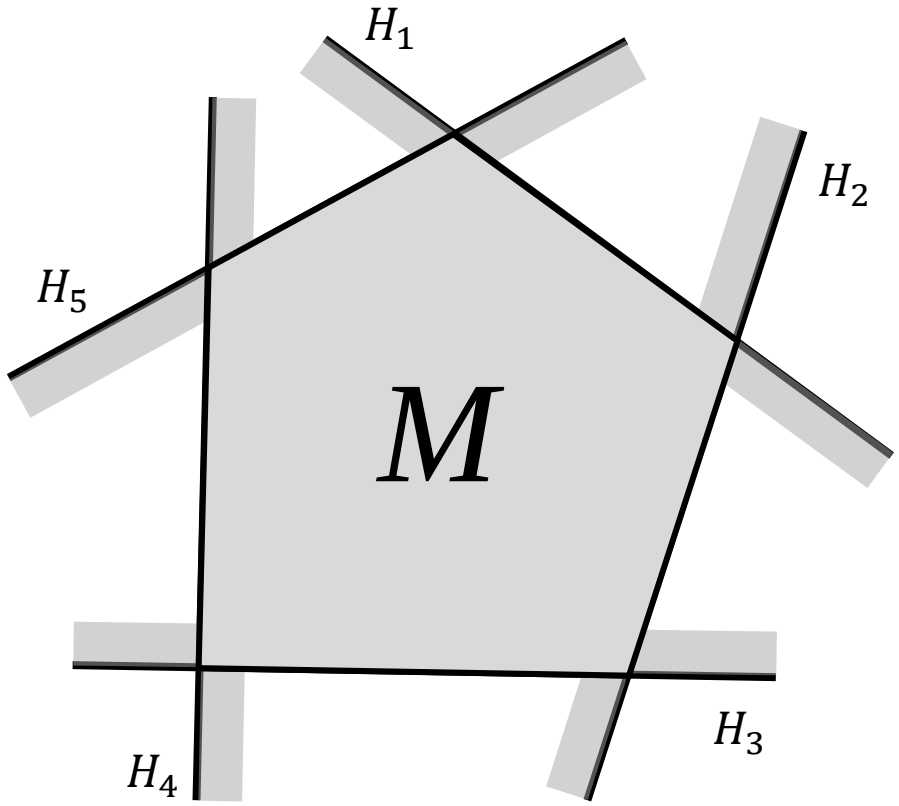


Многогранник допустимых решений
$M = \{\boldsymbol{x}|A\boldsymbol{x} \leq \boldsymbol{b}\}$

Многогранник допустимых решений M

$$H_i = \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{a_i}, \boldsymbol{x} \rangle = b_i\}$$

$$P_i = \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{a_i}, \boldsymbol{x} \rangle \leq b_i\}$$



Идея алгоритма

1. Находим начальную вершину с помощью алгоритмов BIP и VeSP
2. Находим все ребра, выходящие из текущей вершины, у которых значение целевой функции в конечной точке больше, чем в начальной
3. Если таких ребер нет, то переходим на шаг 6
4. Перемещаемся в новую вершину по ребру с максимальным значением целевой функции в конечной точке
5. Переходим на шаг 2
6. Стоп

Как найти все ребра, выходящие из вершины?

1. Определить множество I индексов всех гиперплоскостей, проходящих через вершину $\boldsymbol{v}$

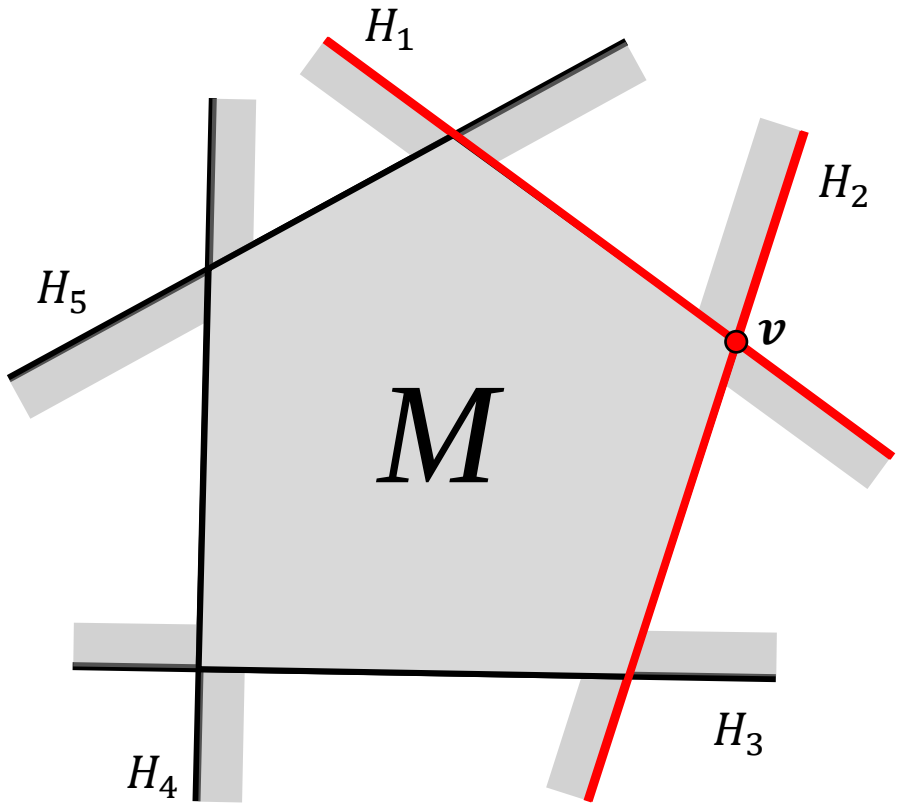
$$I = \{i \in \{1, \dots, m\} | \langle \boldsymbol{a_i}, \boldsymbol{v} \rangle = b_i\}$$

$$|I| \geq n$$

2. Каждая комбинация $\mathcal{J}$ размерности $n-1$ из элементов множества I определяет аффинное подпространство

$$L = \{\boldsymbol{x} \in \mathbb{R}^n \mid A_{\mathcal{J}}\boldsymbol{x} = \boldsymbol{b}_{\mathcal{J}} \wedge \operatorname{rank}(A_{\mathcal{J}}) = n-1\},$$

содержащее искомое ребро



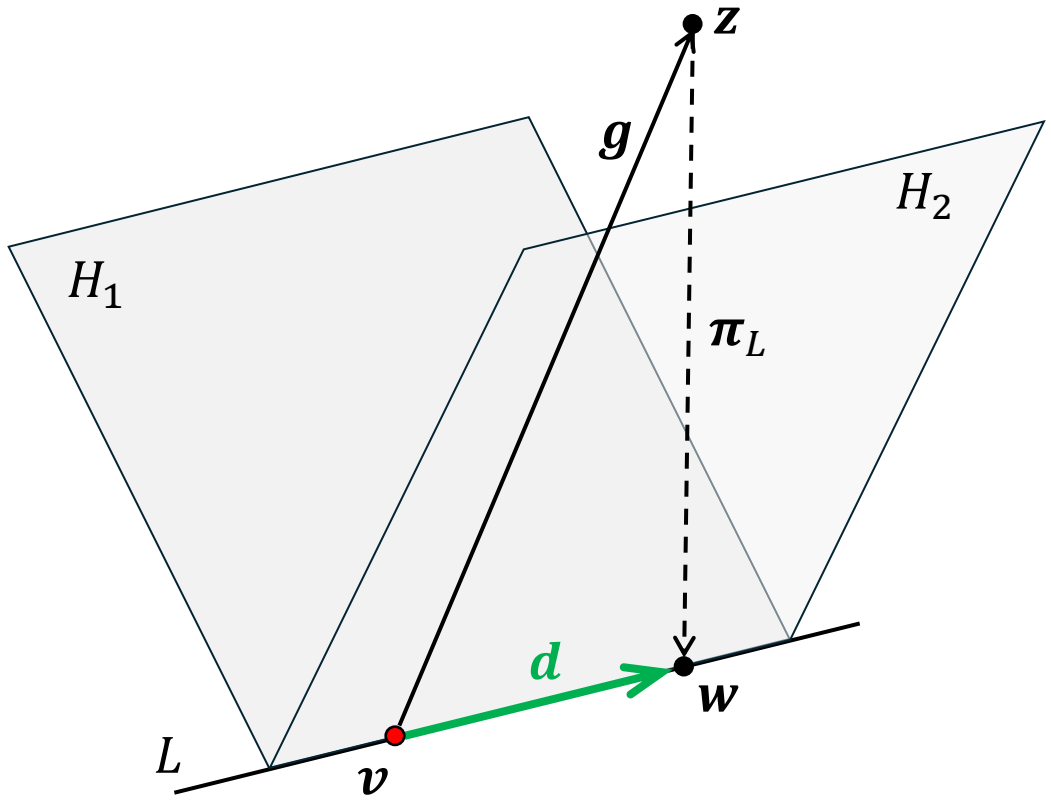
Направление движения по ребру

$\boldsymbol{\pi}_L$ – ортогональная проекция на аффинное подпространство L

$$\boldsymbol{g} \uparrow\uparrow \boldsymbol{c}$$

$$\boldsymbol{z} = \boldsymbol{v} + \boldsymbol{g}$$
$$\boldsymbol{w} = \boldsymbol{\pi}_L(\boldsymbol{z})$$
$$\boldsymbol{d} = \boldsymbol{w} - \boldsymbol{v}$$

$$\boldsymbol{d} = \boldsymbol{D}_{\mathcal{J}}(\boldsymbol{v})$$



Ортогональная проекция $\boldsymbol{\pi}_S(\boldsymbol{x})$

Ортогональная проекция $\boldsymbol{\pi}_S(\boldsymbol{x})$ точки $\boldsymbol{x} \in \mathbb{R}^n$ на аффинное подпространство S:

$$S = \{\boldsymbol{x} \in \mathbb{R}^n | A\boldsymbol{x} = \boldsymbol{b} \wedge \det(AA^T) \neq 0\}$$

$$A^+ = A^T(AA^T)^{-1}$$

$$\boldsymbol{\pi}_S(\boldsymbol{x}) = \boldsymbol{x} - A^+(A\boldsymbol{x} - \boldsymbol{b})$$

Murty, Katta G. Computational and Algorithmic Linear Algebra and n-Dimensional Geometry. WORLD SCIENTIFIC, 2014, pp 361-364. https://doi.org/10.1142/8261

Конечная точка ребра

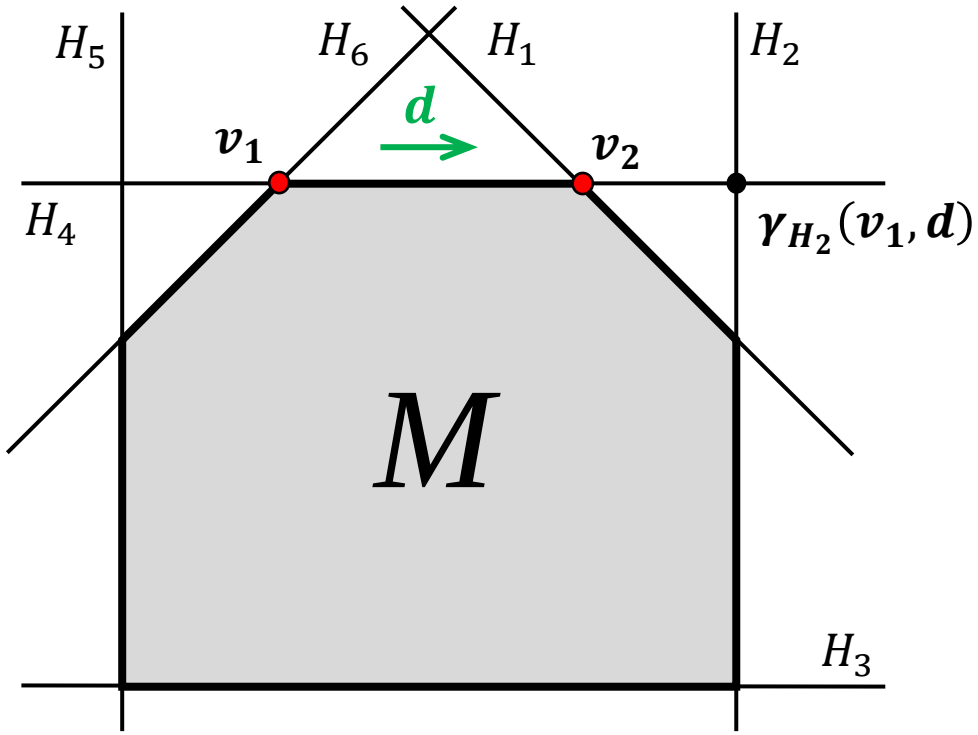
Косоугольная проекция точки $\boldsymbol{x}$ на H_i:

$$\boldsymbol{\gamma_{H_i}(x, d) = x - \frac{\langle a_i, x\rangle - b_i}{\langle a_i, d\rangle} d}$$

Конечная точка ребра лежит на ближайшей гиперплоскости по направлению вектора движения

$$j = \underset{i\in\{1,\ldots,m\}}{argmin}\left\{\left\|\boldsymbol{v - \gamma_{H_i}(v, d)}\right\| \middle| \boldsymbol{v \notin H_i}\right\}$$

$$\boldsymbol{\mu(v, d) = \gamma_{H_j}(v, d)}$$



$$\boldsymbol{v_2 = \mu(v_1, d) = \gamma_{H_1}(v_1, d)}$$

Переход к следующей вершине

1. **function Move($\boldsymbol{v}$)**
2. $I \coloneqq \boldsymbol{BelongTo}_H(\boldsymbol{v})$
3. $\boldsymbol{v_{max}} \coloneqq \boldsymbol{v}$
4. **Twiddle_Init(**$I, n-1$**)**
5. **repeat**
6. $\mathcal{J}$ **≔ Twiddle_GetNextCombination()**
7. **if** $\boldsymbol{Rank}(A_{\mathcal{J}}) == n-1$ **then**
8. $\boldsymbol{d} \coloneqq \boldsymbol{D}_{\mathcal{J}}(\boldsymbol{v})$
9. $\boldsymbol{v_{next}} \coloneqq \boldsymbol{\mu}(\boldsymbol{v}, \boldsymbol{d})$
10. **if** $\langle \boldsymbol{c}, \boldsymbol{v_{next}} \rangle > \langle \boldsymbol{c}, \boldsymbol{v_{max}} \rangle$ **then**
11. $\boldsymbol{v_{max}} \coloneqq \boldsymbol{v_{next}}$
12. **end if**
13. **end if**
14. **until Twiddle_IsDone()**
15. **return** $\boldsymbol{v_{max}}$
16. **end function**

Phillip J. Chase. 1970. Algorithm 382: combinations of M out of N objects [G6]. Commun. ACM 13, 6 (June 1970), 368. https://doi.org/10.1145/362384.362502

Последовательный алгоритм

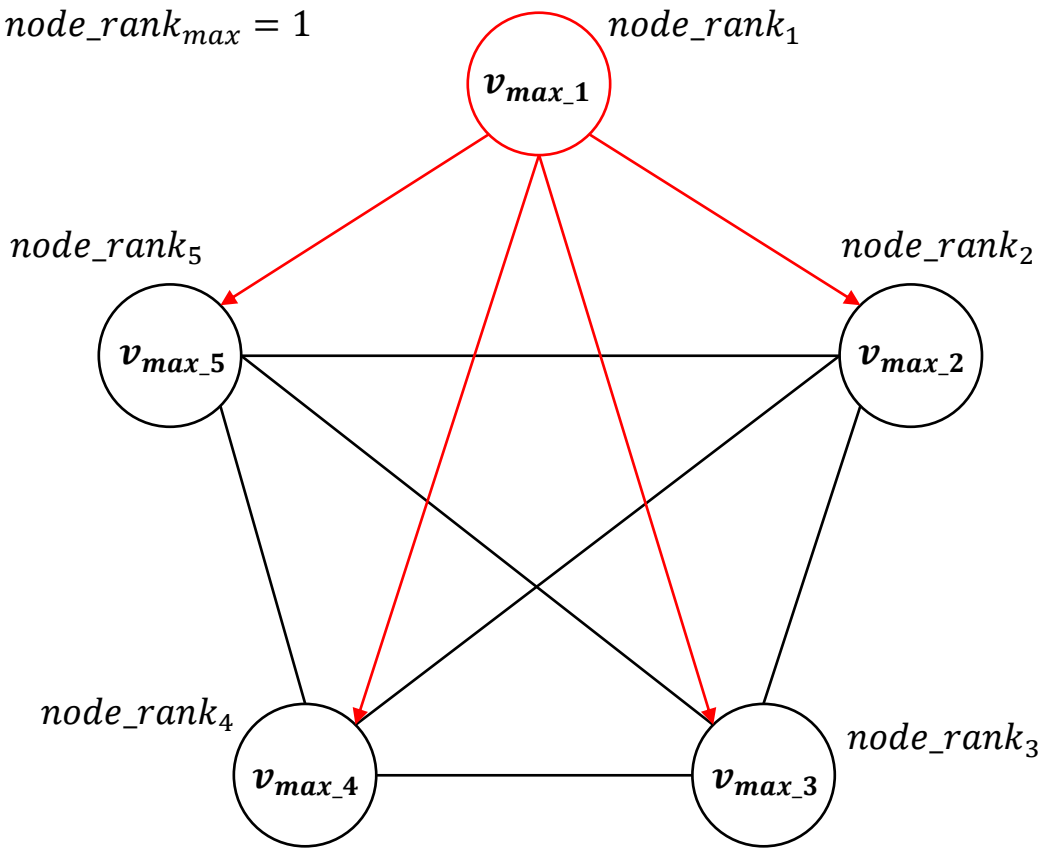
1. **input** $m, n, A, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{v_0}$
2. $\boldsymbol{g} := \frac{\boldsymbol{c}}{\|\boldsymbol{c}\|}\delta$
3. $i := 0$
4. **repeat**
5. $\quad \boldsymbol{v}_{(i+1)} := \mathbf{Move}(\boldsymbol{v}_i)$
6. $\quad i := i + 1$
7. **until** $\langle \boldsymbol{c}, \boldsymbol{v}_i \rangle - \langle \boldsymbol{c}, \boldsymbol{v}_{(i-1)} \rangle < \boldsymbol{\varepsilon}$
8. **output** $\boldsymbol{v}_i$
9. **stop**

Распределение комбинаций между процессорными узлами

```
1.  function DistributedMove(v)
2.      I := BelongTo_H(v)
3.      v_max := v
4.      Twiddle_Init(I, n − 1)
5.      i := node_rank
6.      repeat
7.          J := Twiddle_GetCombination(i)
8.          i := i + nodes_count
9.          if Rank(A_J) == n − 1 then
10.             d := D_J(v)
11.             v_next := μ(v, d)
12.             if ⟨c, v_next⟩ > ⟨c, v_max⟩ then
13.                 v_max := v_next
14.             end if
15.         end if
16.     until Twiddle_IsDone()
17.     return v_max
18. end function
```

Параллельный алгоритм

1. **input** $m, n, A, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{v_0}$
2. $\boldsymbol{g} := \frac{\boldsymbol{c}}{\|\boldsymbol{c}\|}\delta$
3. $i := 0$
4. **repeat**
5. $\quad \boldsymbol{v}_{(i+1)} :=$ **DistributedMove**$(\boldsymbol{v}_i)$
6. $\quad node\_rank_{max} :=$ **MaxLocReduce**$(\langle \boldsymbol{c}, \boldsymbol{v}_{(i+1)} \rangle)$
7. $\quad$ **if** $node\_rank = node\_rank_{max}$ **then**
8. $\quad\quad$ **Broadcast**$(\boldsymbol{v}_{(i+1)})$
9. $\quad$ **else**
10. $\quad\quad$ **Receive**$(\boldsymbol{v}_{(i+1)})$
11. $\quad$ **end if**
12. $\quad i := i + 1$
13. **until** $\langle \boldsymbol{c}, \boldsymbol{v}_i \rangle - \langle \boldsymbol{c}, \boldsymbol{v}_{(i-1)} \rangle < \boldsymbol{\varepsilon}$
14. **if** $node\_rank = node\_rank_{max}$ **then**
15. $\quad$ **output** $\boldsymbol{v}_i$
16. **end if**
17. **stop**



Среда проведения экспериментов

mpicxx (g++)

Режим оптимизации O3

240 вычислительных узлов было выделено для экспериментов

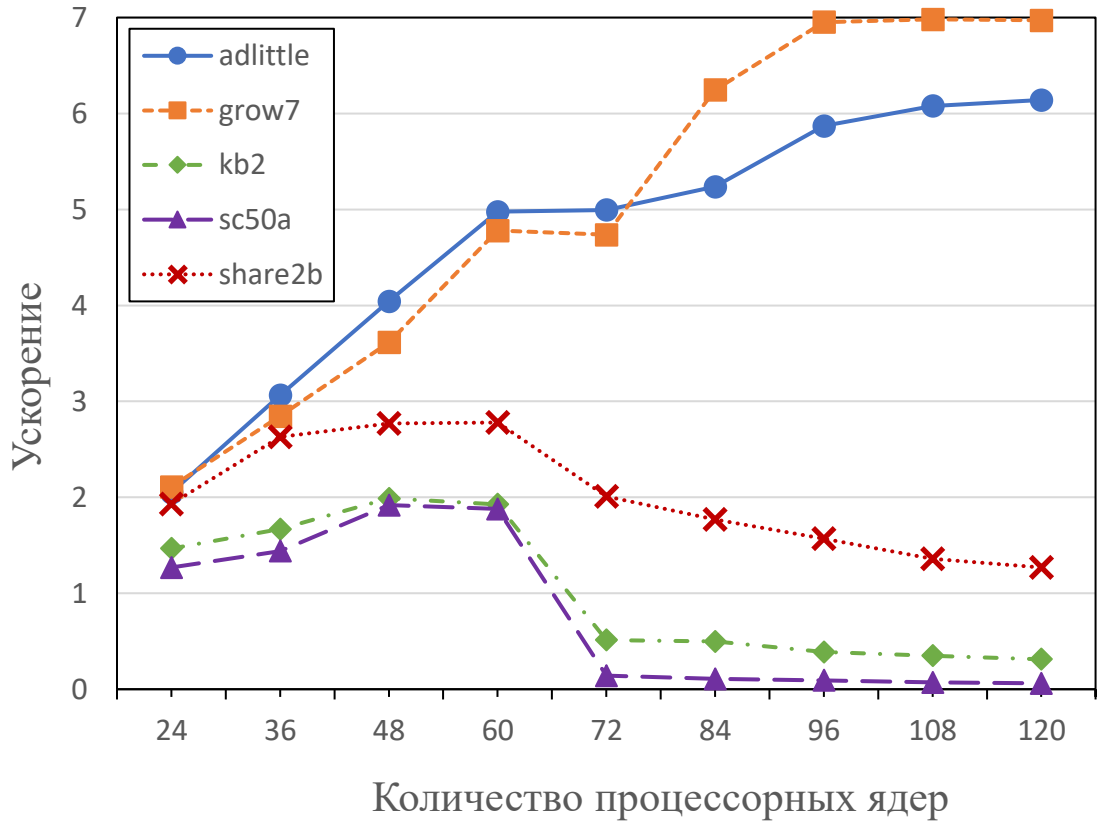
Характеристики вычислительного кластера «Торнадо ЮУрГУ»

Количество процессорных узлов	480
Процессоры	Intel Xeon X5680 (Gulftown, 6 ядер по 3.33 GHz)
Количество процессоров на узле	2
Оперативной памяти на узле	24 ГБ DDR3
Соединительная сеть	InfiniBand QDR (40 Gbit/s)
Операционная система	Linux CentOS 6.2

Ускорение на малых задачах (малая размерность и малое количество комбинаций)

Задача	m	n	k	p	t_p
adlittle	56	97	3403	120	0.083
grow7	140	301	161	96	3.430
kb2	43	41	25	48	0.006
sc50a	49	48	28	48	0.001
share2b	96	97	66	48	0.018

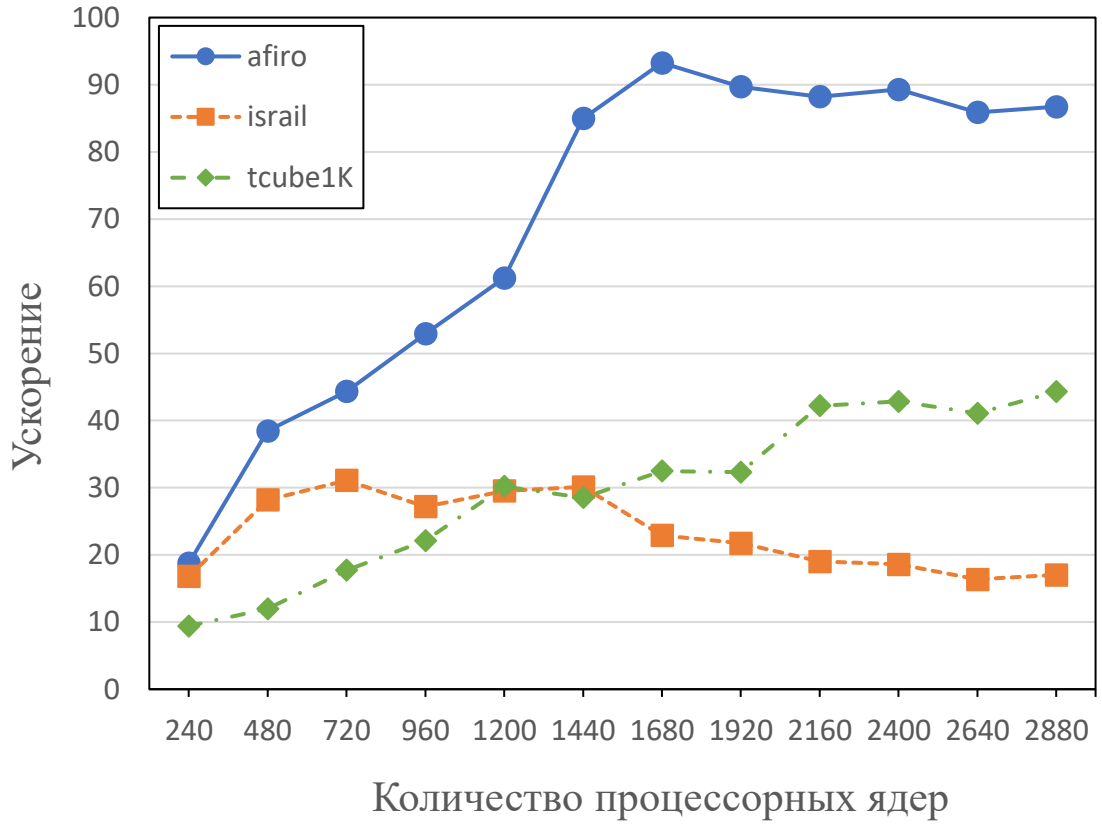
m – количество ограничений
n – количество переменных
k – количество комбинаций
p – граница масштабируемости
t_p – время на границе масштабируемости



Ускорение на больших задачах
(большая размерность или большое количество комбинаций)

Задача	m	n	k	p	t_p
afiro	27	32	28048800	1680	2.596
israel	174	142	10153	720	0.884
tcube1K	1001	1000	1000	2880	106.4

m – количество ограничений
n – количество переменных
k – количество комбинаций
p – граница масштабируемости
t_p – время на границе масштабируемости



Преимущество метода AlEM по сравнению с Simplex-методом

- Stalling – отсутствие улучшения целевой функции при смене базиса на протяжении нескольких итераций из-за вырожденности опорного решения
- Cycling – возникновение бесконечного цикла из-за возвращение к ранее встречавшемуся базису

Задача	Количество итераций	
	AlEM	Simplex
hamck26e	2	∞
hamck26s	1	∞

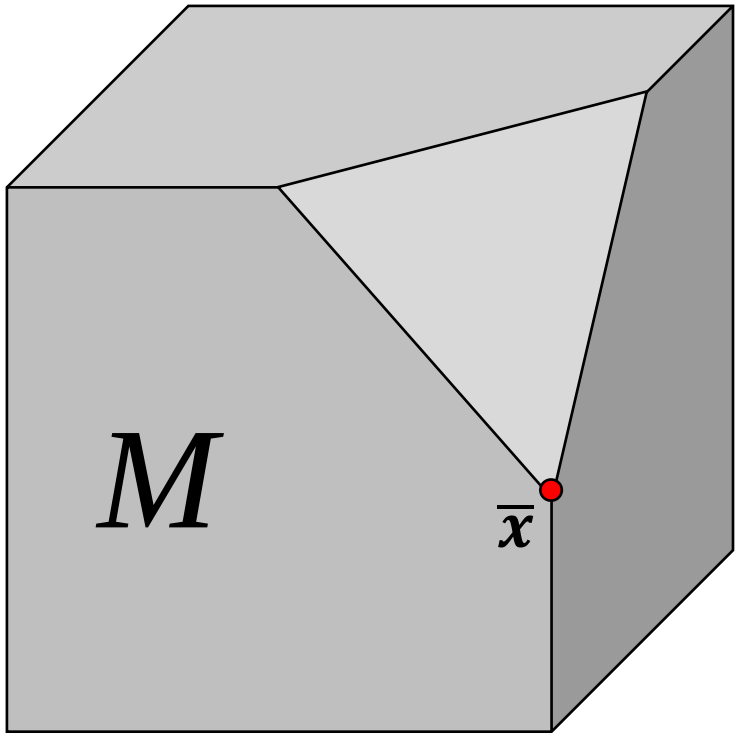
Hall, J., McKinnon, K. The simplest examples where the simplex method cycles and conditions where expand fails to prevent cycling. Math. Program., Ser. B 100, 133–150 (2004). https://doi.org/10.1007/s10107-003-0488-1

Спасибо за внимание!

Масштабируемая задача tcube

Гиперкуб с отсеченной вершиной:

$$\begin{cases} x_0 & & & & \leq & 200 \\ & x_1 & & & \leq & 200 \\ & & \ddots & & \cdots & \cdots \\ & & & x_{n-1} & \leq & 200 \\ x_0 & + x_1 & \cdots & + x_{n-1} & \leq & 200(n-1)+100 \\ & & & & & \\ x_0 & & & & \geq & 0 \\ & x_1 & & & \geq & 0 \\ & & \ddots & & \cdots & \cdots \\ & & & x_{n-1} & \geq & 0 \end{cases}$$



$$c = (1, 2, \ldots, n)$$

$$\bar{x} = (200, \ldots, 200, 100)$$

Сравнение с Simplex-методом

Задача	m	n	$t_{Simplex}$	t_{AlEM}	$\frac{t_{AlEM}}{t_{Simplex}}$
adlittle	56	97	0.018	0.083	4.611
israel	174	142	0.023	0.884	38.43
tcube1K	1001	1000	74.15	106.4	1.434