Сравнение параллельных реализаций симплекс-метода для безошибочного решения задач линейного программирования

А.В. Панюков, В.В. Горбик

Кафедра экономико-математических методов и статистики,
Национальный исследовательский университет ЮУрГУ
Челябинск, Российская Федерация
a_panyukov@mail.ru, vgorbik@gmail.com

Международная научная конференция
"Параллельные вычислительные технологии 2011"
Москва, 28 марта – 1 апреля 2011.

Применение массивно-параллельных вычислений для решения задач линейного программирования с гарантированной погрешностью

Содержание

- Вычисления с гарантированной точностью
- Применение массивно-параллельных вычислений для реализации операций дробно-рациональной арифметики
- Техника реализации симплекс-метода на кластерных системах
- Оценка ускорения и эксперименты
- Особенности и сложности реализации

Введение

Обзор

- коммерческие профессиональные пакеты
- gnu/glpk
- exlp, qsopt-ex
- библиотека netlib и др.

Проблема точности вычислений

- бинарная мантисса (невозможность точного представления многих чисел, например 0.1)
- сложение/вычитание чисел разного порядка
- отсутствие учета погрешности

Вычисления с гарантированной точностью GNU Multiple Precision Arithmetic Library: http://gmplib.org/, 2010

Тип данных mpf_t

- mpf_t (multiple-precision floating point) входит в библиотеку gnu/mp
- gnu/mp является частью любого linux дистрибутива
- gnu/mp имеет оптимизации под множество архитектур
- длину мантиссы mpf_t можно менять динамически

Интервальные вычисления тип данных mpfi_t

mpfi_t (multiple-precision floating point interval library)

$\Uparrow$

mpfr_t (multiple-precision floating point with correct rounding)

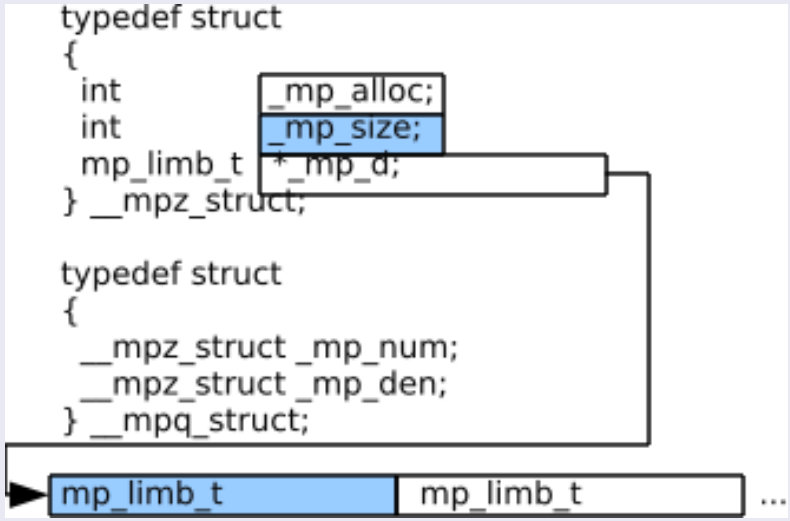
$\Uparrow$

gnu/mp mpf_t

- mpfi_t { mpfr_t, mpfr_t }
- гарантированные результаты

Тип данных mpq_t

- mpq_t входит в библиотеку gnu/mp
- дробно-рациональные вычисления без округлений



MPI: A Message Passing Interface Standard http://www.MPI-forum.org/docs/MPI-11-html/MPI-report.html

Публикации

A. V. Panyukov, V. V. Gorbik Exact and Guaranteed Accuracy Solutions of Linear Programming Problems by Distributed Computer Systems with MPI // Tambov University REPORTS: A Theoretical and Applied Scientific Journal. Series: Natural and Technical Sciences. – Volume 15, Issue 4, 2010. – P. 1392-1404. http://vestnik.tsutmb.ru/old/index.php?module=subjects&func=viewpage&pageid=165

А.В. Панюков, М.И. Германенко, В.В. Горбик. Библиотека классов "Exact Computational" / Свидетельство о государственной регистрации программы для ЭВМ № 2009612777 от 29 мая 2009г. // Программы для ЭВМ, базы данных, топологии интегральных микросхем. Официальный бюллетень Российского агентства по патентам и товарным знакам No 3. – 2009. – С. 251.

Задача линейного программирования в стандартной форме

Постановка задачи

$$\max\left\{c^T x \ : Ax = b \geq 0,\ x \geq 0;\ c, x \in \mathbf{R}^n;\ b \in \mathbf{R}^m\right\}.$$

Техника реализации

- метод симплекс таблиц;
- метод обратной матрицы (модифицированный симплекс метод).

Симплекс-таблица

$$S^{(k)} = \left|\begin{array}{c|c} Z^{(k)} = c^T_{B(k)} B(k)^{-1} b & z^{(k)} = -c^T + c^T_{B(k)} B(k)^{-1} A \\ \hline X_{B(k)} = B(k)^{-1} b & B(k)^{-1} A \end{array}\right|,$$

- $B(k)$ – базисная матрица, содержащая все относящиеся к базисным переменным k-й итерации столбцы матрицы A (базисные столбцы);
- $c_{B(k)}$ – вектор коэффициентов целевой функции, относящихся к базисным переменным k-й итерации;
- $X_{B(k)} = B(k)^{-1} b$ – вектор значений базисных переменных k-й итерации;
- $Z^{(k)}$ – значение целевой функции на этом решении;
- $z^{(k)} = -c^T + c^T_{B(k)} B(k)^{-1} A$ – вектор невязок двойственной задачи.

Битовая сложность абсолютно точной реализации симплекс-метода

Утверждение

Если задача имеет m ограничений-равенств, n переменных, а один численный элемент исходных данных имеет пространственную сложность не более l, то

- элементы симплекс-таблицы и обратной матрицы имеют пространственную сложность не более $4lm^2 + lm + m\log_2 m + 1$;
- столбец симплекс-таблицы и обратной матрицы, а также вектор двойственных переменных имеют пространственную сложность не более $4lm^3 + lm^2 + m^2\log_2 m + m$;
- для представления симплекс-таблицы достаточно $4lm^3n + o(lm^2n)$ бит;
- для представления обратной матрицы достаточно $4lm^4 + o(lm^3)$ бит.

Параллельная реализация метода симплекс-таблиц

Декомпозиция симплекс-таблицы по процессорам

Процесс $K = 1, 2, \ldots, N$				
$S_{00} = Z$	$z_{\lceil \frac{(K-1)n}{N} \rceil + 1}$	$z_{\lceil \frac{(K-1)n}{N} \rceil + 2}$	$\cdots$	$z_{\lceil \frac{Kn}{N} \rceil}$
$S_{10} = X_{B1}$	$S_{1\lceil \frac{(K-1)n}{N} \rceil + 1}$	$S_{1\lceil \frac{(K-1)n}{N} \rceil + 2}$	$\cdots$	$S_{1\lceil \frac{Kn}{N} \rceil}$
$S_{20} = X_{B2}$	$S_{2\lceil \frac{(K-1)n}{N} \rceil + 1}$	$S_{2\lceil \frac{(K-1)n}{N} \rceil + 2}$	$\cdots$	$S_{2\lceil \frac{Kn}{N} \rceil}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$S_{m0} = X_{Bm}$	$S_{m\lceil \frac{(K-1)n}{N} \rceil + 1}$	$S_{m\lceil \frac{(K-1)n}{N} \rceil + 2}$	$\cdots$	$S_{m\lceil \frac{Kn}{N} \rceil}$

Алгоритм TabularSimplex

Данные: симплекс-таблица $S(K)$ для каждого процесса $K = 1, 2, 3, \ldots, N$.

Шаг 1.

Каждому процессу $K = 1, 2, 3, \ldots, N$

- найти столбец i_K : $z_{i_K} < 0$;
- если столбец i_K не найден, то положить $C(K) = \Delta(K) = i_K = 0$ и перейти на шаг 2;
- найти строку

$$l_K = \min \left\{ \mathrm{Arg} \min_{l:\ S_{l\, i_K} > 0} \left(\frac{X_{Bl}}{S_{l\, i_K}} \right) \right\};$$

- если строка l_K не найдена, то завершить выполнение всех процессов и вернуть "Не ограничена";
- вычислить $\Delta_{i_K} = -X_{B l_K} z_{i_K} / S_{l_K i_K}$, положить $\Delta(K) = \Delta_{i_K}$, $C(K) = i_K$ и перейти на шаг 2.

Шаг 2.

Для $L = 1, 2, 4, 8, \ldots, N$ каждому процессу $K = 1, 2, 3, \ldots, N$, номер которого удовлетворяет условию $((K-1)\%(2L)) < L$, осуществить обмен данными с процессом $K + L$:

- Если $C(K) = 0$, то положить $C(K) = C(K+L)$, $\Delta(K) = \Delta(K+L)$ и продолжить вычисления для следующего L.
- Если $C(K+L) = 0$, то положить $C(K+L) = C(K)$, $\Delta(K+L) = \Delta(K)$ и продолжить вычисления для следующего L.
- Если $\Delta(K) \geq \Delta(K+L)$, то положить $C(K+L) = C(K)$, $\Delta(K+L) = \Delta(K)$, иначе положить $C(K) = C(K+L)$, $\Delta(K) = \Delta(K+L)$.
- Продолжить вычисления для следующего L.

Шаг 3.

Если $K^* = 0$, то каждому процессу $K = 1, 2, 3, \ldots, N$ для

$$k = \lceil \frac{(K-1)n}{N} \rceil + 1,\ \lceil \frac{(K-1)n}{N} \rceil + 2, \ldots, \lceil \frac{Kn}{N} \rceil$$

положить

$$x_k = \begin{cases} X_{Bl}, & \text{если } (S_{lk} = 1) \bigwedge ((\forall i = 0, 1, \ldots, l-1, l+1, \ldots, m)\, S_{ik} = 0), \\ 0, & \text{в противном случае.} \end{cases}$$

Вернуть x – оптимальное решение задачи, Z – оптимальное значение целевой функции и завершить выполнение алгоритма.

Шаг 4.

Ведущему процессу K^* передать остальным процессам ведущий столбец

$$S_{i_{K^*}} = \left(z_{i_{K^*}},\ S_{1i_{K^*}},\ S_{2i_{K^*}},\ \ldots,\ S_{mi_{K^*}}\right)^T$$

и номер ведущей строки l_{K^*}.

Шаг 5.

Каждому процессу $K = 1, 2, 3, \ldots, N$ пересчитать симплекс-таблицу $S(K)$ по формуле

$$\left(\ \forall j = 0, 1, 2, \ldots, n\ \right)\left(S_{lj}^{(k+1)} = \begin{cases} S_{lj}^{(k)} - \dfrac{S_{l^*j}^{(k)}}{S_{l^*i}^{(k)}} S_{li}^{(k)}, & \text{если } l \neq l^*, \\ \dfrac{S_{l^*j}^{(k)}}{S_{l^*i}^{(k)}}, & \text{если } l = l^* \end{cases}\right).$$

Перейти к следующей итерации.

Конец алгоритма

Алгебраический вычислительный ресурс реализаций метода симплекс-таблиц

Оператор	Один процессор	N процессоров	
	Количество алгебраических операций	Количество пересылаемых операндов	Нагрузка на один процесс
Проверка условия оптимальности	$[1, n]$	–	$[1, n/N]$
Определение ведущей строки	$2m + n + 2$	–	$2m + n/N + 2$
Выбор ведущего процесса	–	N	$\lceil \log_2 N \rceil$
Пересылка ведущего столбца	–	$m + 2$	$\lceil \log_2 N \rceil$
Пересчет симплекс-таблицы	$2m(n + 1)$	–	$2m(1 + n/N)$
Итого:	$C_T^A(1) = [1, n] + 2mn + 4m + n + 4$	$m + N + 2$	$C_T^A(N) = [1, n/N] + (n/N)(2m + 1) + 4m + 2 + 2\log_2 N$

Ускорение от распараллеливания

$$U_T^A = \frac{C_T^A(1)}{C_T^A(N)} = \frac{[1, n] + 2mn + 4m + n + 4}{[1, n/N] + 2mn/N + 4m/N + 2\log_2 N + 2} \xrightarrow[n,m\to\infty]{} N.$$

Декомпозиция матриц A и c по процессорам

$$c(K)^T = \left(\begin{array}{cccc} c_{\lceil \frac{(K-1)n}{N} \rceil + 1} & c_{\lceil \frac{(K-1)n}{N} \rceil + 2} & \cdots & c_{\lceil \frac{Kn}{N} \rceil} \end{array} \right) \tag{1}$$

$$A(K) = \left(\begin{array}{cccc} a_1\left(\lceil \frac{(K-1)n}{N} \rceil + 1\right) & a_1\left(\lceil \frac{(K-1)n}{N} \rceil + 2\right) & \cdots & a_1\left(\lceil \frac{Kn}{N} \rceil\right) \\ a_2\left(\lceil \frac{(K-1)n}{N} \rceil + 1\right) & a_2\left(\lceil \frac{(K-1)n}{N} \rceil + 2\right) & \cdots & a_2\left(\lceil \frac{Kn}{N} \rceil\right) \\ \vdots & \vdots & \ddots & \vdots \\ a_m\left(\lceil \frac{(K-1)n}{N} \rceil + 1\right) & a_m\left(\lceil \frac{(K-1)n}{N} \rceil + 2\right) & \cdots & a_m\left(\lceil \frac{Kn}{N} \rceil\right) \end{array} \right) \tag{2}$$

Вектор двойственных переменных y размещен в каждом процессе $K = 1, 2, \ldots, N$.

Декомпозиция матрицы B^{-1} по процессорам

$$B^{-1}(K)= \begin{pmatrix} b_{\left(\lceil \frac{(K-1)m}{N} \rceil+1\right)1} & b_{\left(\lceil \frac{(K-1)m}{N} \rceil+1\right)2} & \cdots & b_{\left(\lceil \frac{(K-1)m}{N} \rceil+1\right)m} \\ b_{\left(\lceil \frac{(K-1)m}{N} \rceil+2\right)1} & b_{\left(\lceil \frac{(K-1)m}{N} \rceil+2\right)2} & \cdots & b_{\left(\lceil \frac{(K-1)m}{N} \rceil+2\right)m} \\ \vdots & \vdots & \ddots & \vdots \\ b_{(\lceil \frac{Km}{N} \rceil)1} & b_{(\lceil \frac{Km}{N} \rceil+2)2} & \cdots & b_{(\lceil \frac{Km}{N} \rceil+2)m} \end{pmatrix} \quad (3)$$

Размещение данных по процессорам

$$X_B(K) = \begin{pmatrix} X_{B\lceil\frac{(K-1)m}{N}\rceil+1} \\ X_{B\lceil\frac{(K-1)m}{N}\rceil+2} \\ \vdots \\ X_{B\lceil\frac{Km}{N}\rceil} \end{pmatrix};\ c_B(K) = \begin{pmatrix} c_{B\lceil\frac{(K-1)m}{N}\rceil+1} \\ c_{B\lceil\frac{(K-1)m}{N}\rceil+2} \\ \vdots \\ c_{B\lceil\frac{Km}{N}\rceil} \end{pmatrix}; \qquad (4)$$

$$g(K) = \begin{pmatrix} g_{\lceil\frac{(K-1)m}{N}\rceil+1} \\ g_{\lceil\frac{(K-1)m}{N}\rceil+2} \\ \vdots \\ g_{\lceil\frac{Km}{N}\rceil} \end{pmatrix} \qquad (5)$$

Алгоритм InvBasMatrSimplex

Данные:

В каждом процессе $K = 1, 2, 3, \ldots, N$ размещены
y – вектор двойственных переменных, построенный по текущему базисному решению прямой задачи;
$M(K)$ – вектор номеров базисных переменных процесса;
$X_B(K)$ – вектор значений базисных переменных процесса;
$c_B(K)$ – вектор значений коэффициентов целевой функции при базисных переменных процесса;
$A(K)$ – блок матрицы ограничений процесса;
$c(K)^T$ – блок коэффициентов целевой функции процесса;
$B^{-1}(K)$ – блок обратной матрицы процесса.

Шаг 1.

Каждому процессу $K = 1, 2, 3, \ldots, N$

- найти столбец
$$i_K : \ z_{i_K} = A(K)^T_{i_K} y - c(K)_{i_K} < 0;$$
- если столбец i_K не найден, то положить $C(K) = \Delta(K) = i_K = 0$; иначе положить $\Delta(K) = z_{i_K}$, $C(K) = i_K$.

Шаг 2.

Для $L = 1, 2, 4, 8, \ldots, N$ каждому процессу $K = 1, 2, 3, \ldots, N$, номер которого удовлетворяет условию $((K-1)\%(2L)) < L$, осуществить обмен данными с процессом $K + L$:

- Если $C(K) = 0$, то положить $C(K) = C(K+L)$, $\Delta(K) = \Delta(K+L)$ и продолжить вычисления для следующего L.
- Если $C(K+L) = 0$, то положить $C(K+L) = C(K)$, $\Delta(K+L) = \Delta(K)$ и продолжить вычисления для следующего L.
- Если $\Delta(K) \leq \Delta(K+L)$, то положить $C(K+L) = C(K)$, $\Delta(K+L) = \Delta(K)$, иначе положить $C(K) = C(K+L)$, $\Delta(K) = \Delta(K+L)$.
- Продолжить вычисления для следующего L.

Шаг 3

. Если $K^c = 0$, то сформировать решение задачи:

- положить $x[1:n] = 0$;
- каждому процессу $K = 1, 2, 3, \ldots, N$ для

$$r = \lceil \frac{(K-1)m}{N} \rceil + 1,\ \lceil \frac{(K-1)m}{N} \rceil + 2, \ldots, \lceil \frac{Km}{N} \rceil$$

положить

$$x_{M(K)_r} = X_{B(K)_r};$$

- вернуть x – оптимальное решение задачи, y – оптимальное решение двойственной задачи и завершить выполнение алгоритма.

Шаг 4.

Процессу K^c разослать всем процессам $K = 1, 2, \ldots, N$ столбец $A_{i_{K^c}}$ и значение коэффициента $c_{i_{K^c}}$.

Шаг 5.

Каждому процессу $K = 1, 2, \ldots, N$ вычислить $g(K) = B(K)^{-1} A_{i_{K^c}}$ и

$$r(K) = \arg \min_{l:\ g(K)_l > 0} \left[h(K, l) = \frac{X_B(K)_l}{g(K)_l} \right].$$

Если $r(K)$ не найдено, то положить $C(K) = 0$, в противном случае положить $C(K) = r(K),\ \Delta(K) = h(K, r(K))$.

Шаг 6.

Для $L = 1, 2, 4, 8, \ldots, N$ каждому процессу $K = 1, 2, 3, \ldots, N$, номер которого удовлетворяет условию $((K-1)\%(2L)) < L$, осуществить обмен данными с процессом $K + L$:

- Если $C(K) = 0$, то положить $C(K) = C(K+L)$, $\Delta(K) = \Delta(K+L)$ и продолжить вычисления для следующего L.
- Если $C(K+L) = 0$, то положить $C(K+L) = C(K)$, $\Delta(K+L) = \Delta(K)$ и продолжить вычисления для следующего L.
- Если $\Delta(K) \leq \Delta(K+L)$, то положить $C(K+L) = C(K)$, $\Delta(K+L) = \Delta(K)$, иначе положить $C(K) = C(K+L)$, $\Delta(K) = \Delta(K+L)$.
- Продолжить вычисления для следующего L.

Шаг 7.

Ведущему процессу K^r разослать всем процессам $K = 1, 2, 3, \ldots, N$ ведущую строку $r^* = r(K^r)$ обратной матрицы

$$\left(B(K)^{-1}\right)^{(r^*)} = \left(\; b_{r^*1}, \quad b_{r^*2}, \quad \cdots, \quad b_{r^*m} \; \right)$$

и значение $g(K^r)_{r^*}$. Положить $M(K)_{r^*} = i_{K^c}$, $c_B(K)_{r^*} = c_{i_{K^c}}$.

Шаг 8.

Каждому процессу $K = 1, 2, \ldots, N$ вычислить новые значения базисных переменных процесса $X_B(K)$, новый блок обратной матрицы процесса $B^{-1}(K)$, и двойственное субрешение блока

$$y(K) = \left(B(K)^{-1}\right)^T c_B(K).$$

Шаг 9.

Для $L = 1, 2, 4, 8, \ldots, N$ каждому процессу $K = 1, 2, 3, \ldots, N$, номер которого удовлетворяет условию $((K-1)\%(2L)) < L$, осуществить обмен данными с процессом $K + L$:

- $\tilde{y} = y(K) + y(K + L)$;
- $y(K) = y(K + L) = \tilde{y}$;
- Продолжить вычисления для следующего L.

Алгебраический вычислительный ресурс реализаций метода метода обратной матрицы

Оператор	Один процессор	N процессоров	
	Количество алгебраических операций	Количество пересылаемых операндов	Нагрузка на один процесс
Проверка условия оптимальности	$2m[1,n]$	–	$2m[1,n/N]$
Выбор c-ведущего процесса	–	N	$\log_2 N$
Передача ведущего столбца	–	$m+1$	$\log_2 N$
Нахождение g	$m(2m-1)$	–	$m(2m-1)/N$
Нахождение ведущей строки	$2m$	N	$2m/N+\log_2 N$
Модификация X_B	$1+2m$	1	$2m/N$
Модификация B^{-1}	$3m^2$	m	$3(m^2/N)$
Вычисление $y(K)$	–	–	$(m/N)(2(m/N)-1)$
Вычисление y	$m(2m-1)$	N	$\log_2 N$
Итого:	$C_I^A(1)=$ $2m[1,n]+$ $7m^2+2m-1$	$2m+3N+2$	$C_I^A(N)=$ $2m[1,n/N]+$ $7m^2/N+2m/N+$ $4\log_2 N$

Сопоставление методов

Ускорение от распараллеливания

$$U_T^A = \frac{C_T^A(1)}{C_T^A(N)} = \frac{[1,n] + 2mn + 4m + n + 4}{[1,n/N] + 2mn/N + 4m/N + 2\log_2 N + 2} \xrightarrow[n,m\to\infty]{} N.$$

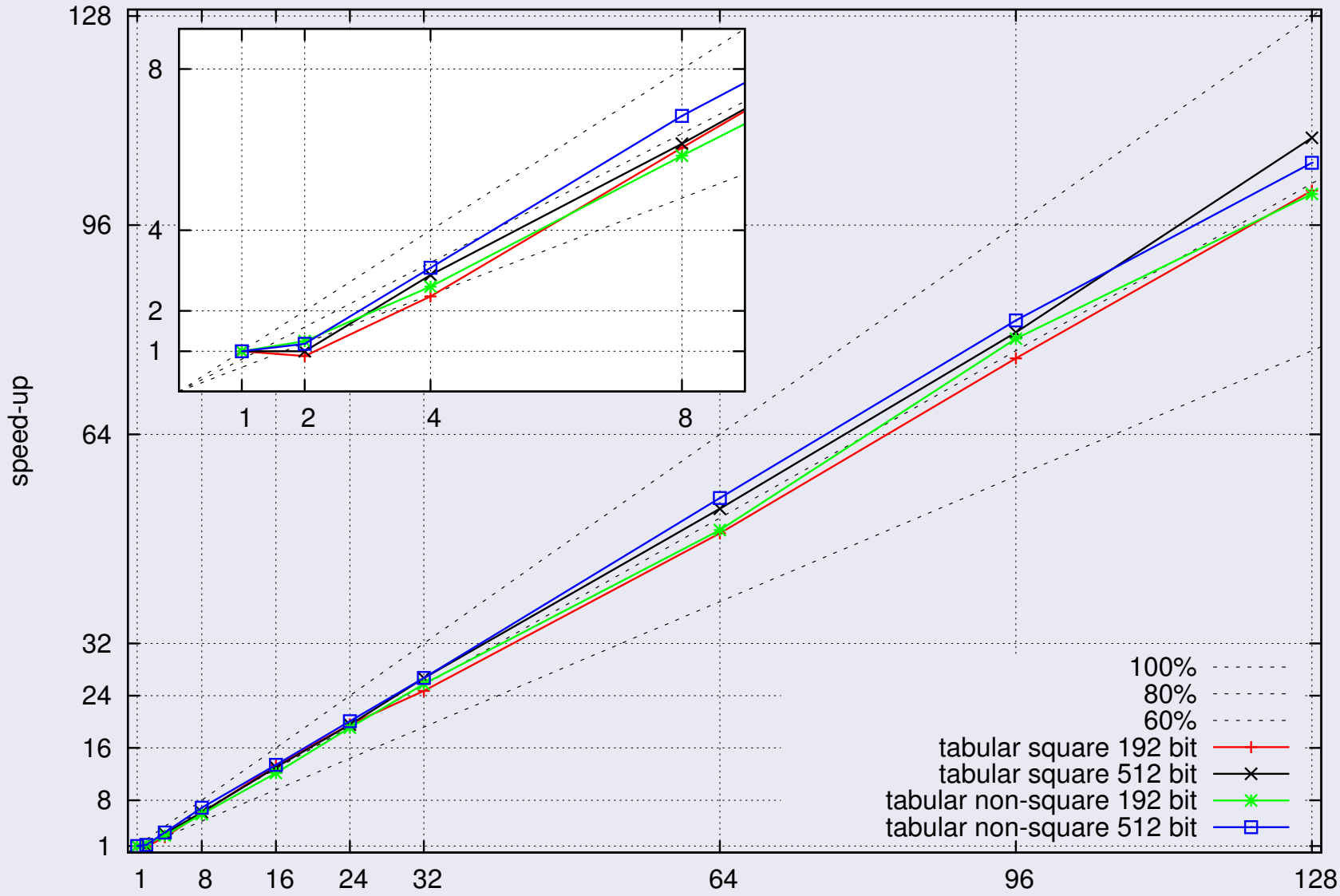
$$U_I^A = \frac{C_I^A(1)}{C_I^A(N)} = \frac{2m[1,n] + 7m^2 + 2m - 1}{2m[1,n/N] + 7m^2/N + 2m/N + 4\log_2 N} \xrightarrow[n,m\to\infty]{} N.$$

Преимущества метода обратной матрицы

- требуется найти решение двойственной задачи;
- число n существенно превосходит m;
- матрица A разрежена;
- возможна оптимизация проверки допустимости двойственного решения.

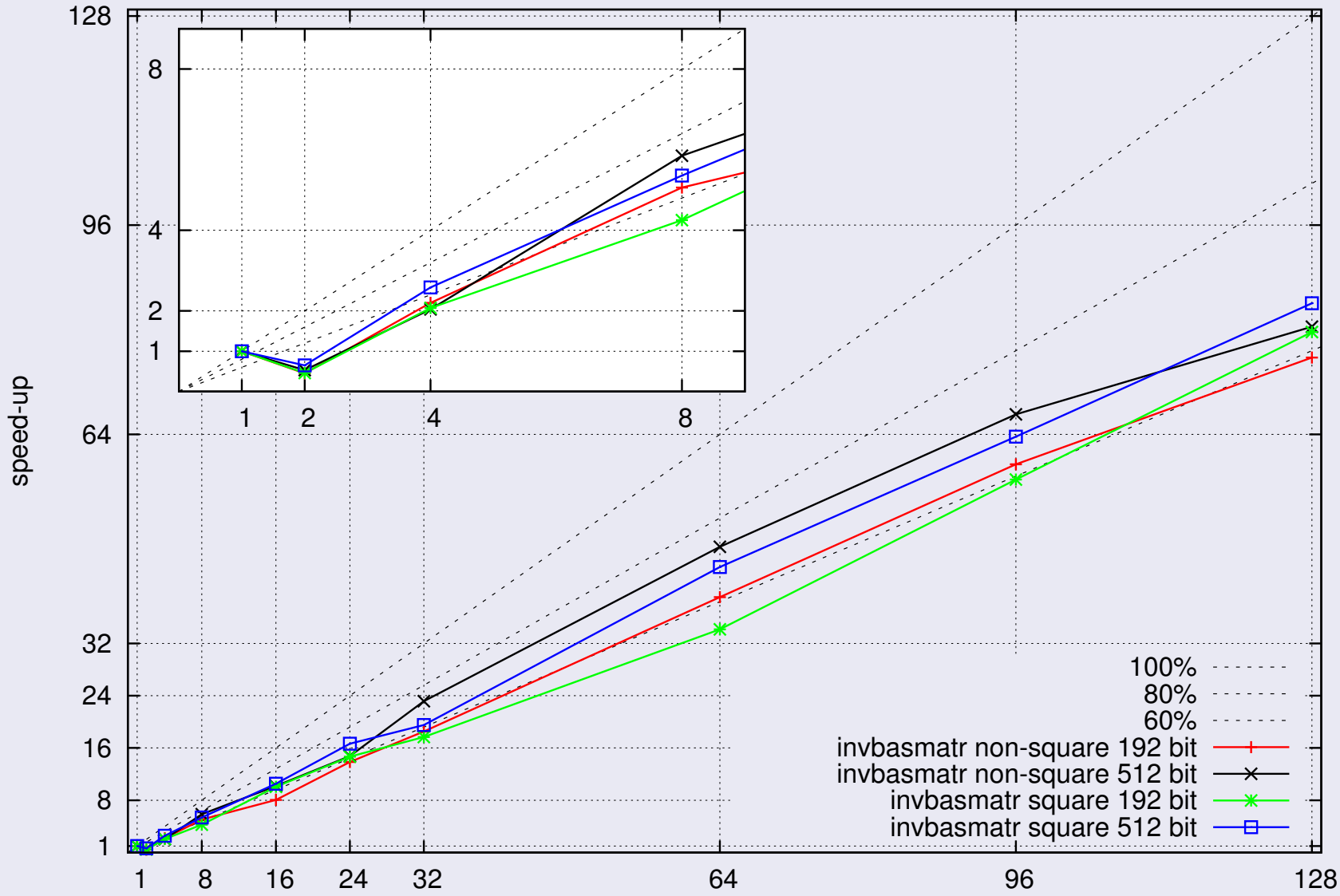
Ускорение реализации метода симплекс таблиц

Вычисления на основе интервального типа данных mpfi_t



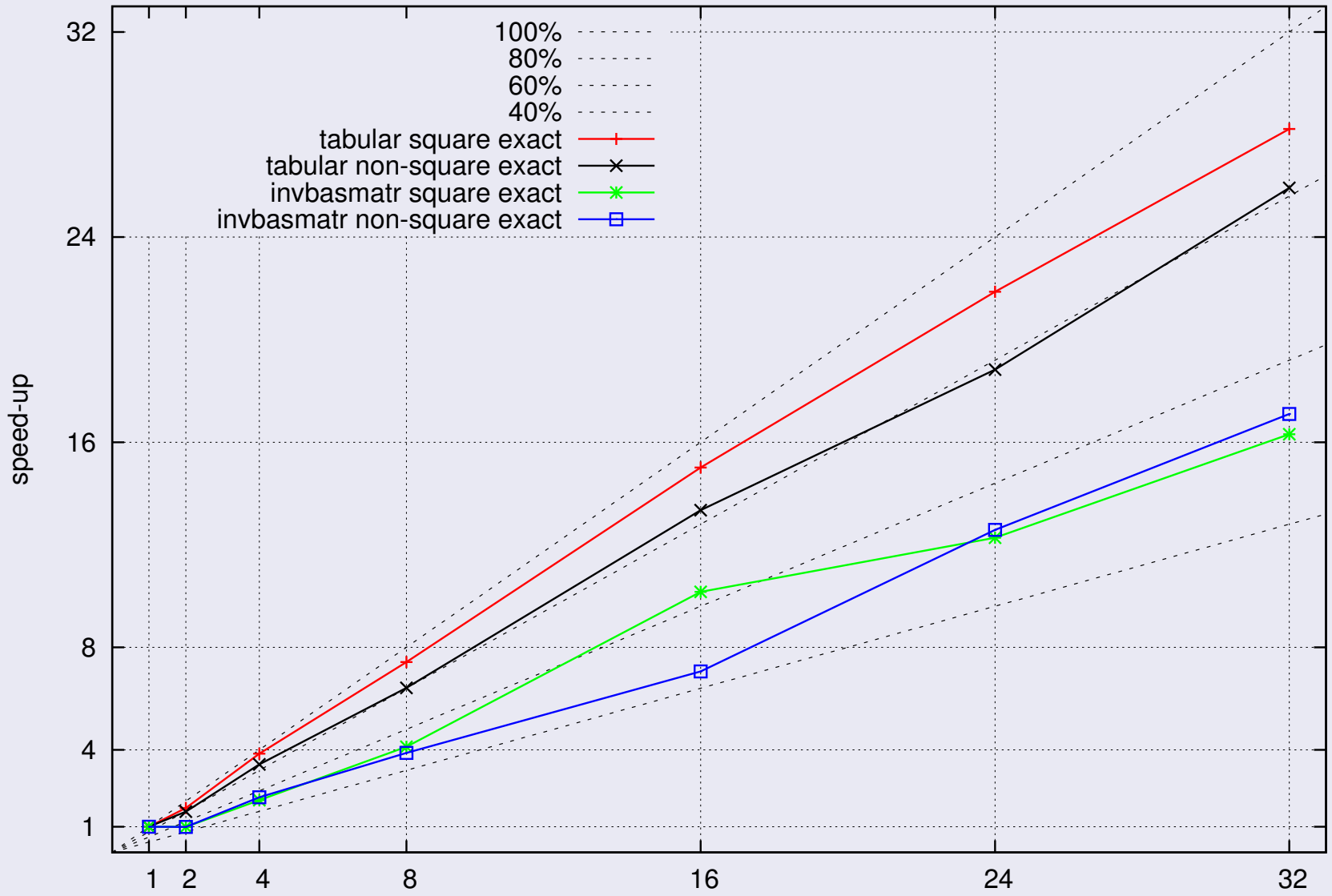
Ускорение реализации метода обратной матрицы

Вычисления на основе интервального типа данных mpfi_t



Ускорение реализаций симплекс-метода

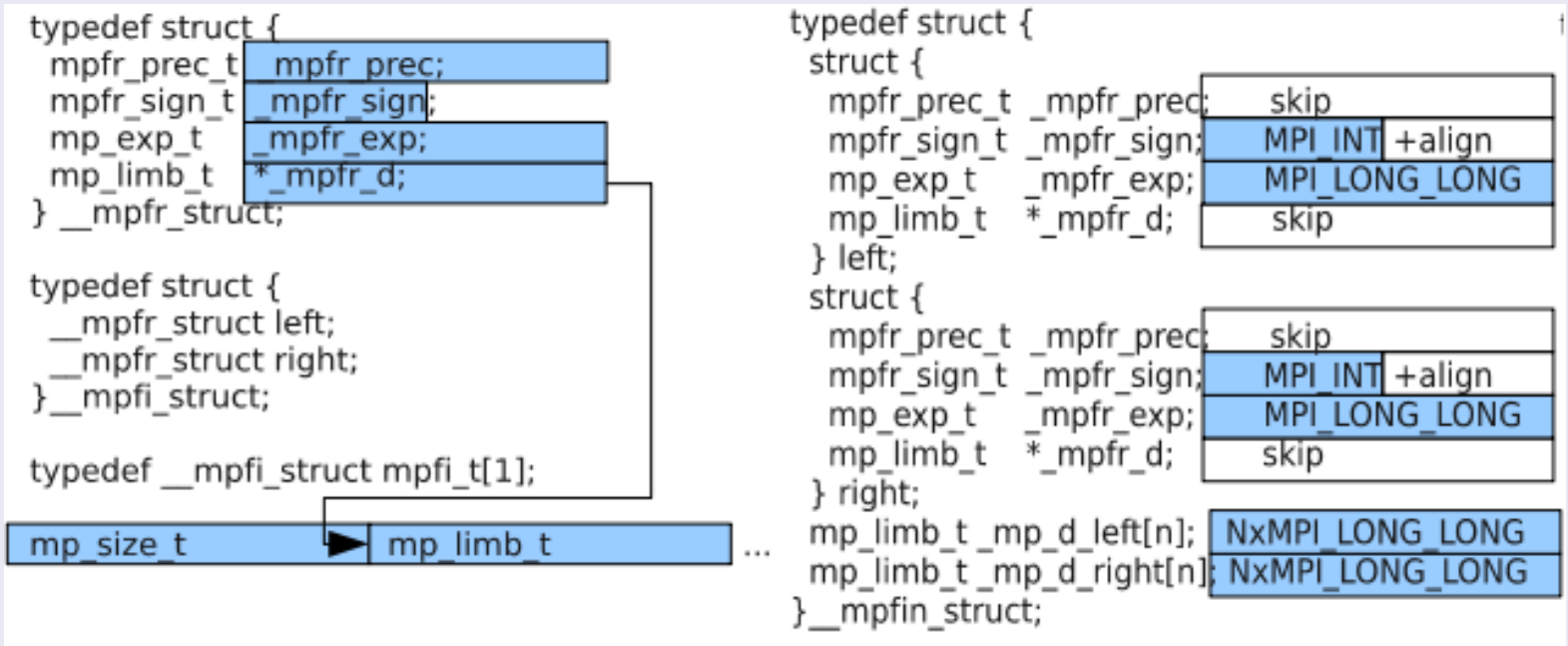
Вычисления на основе дробно-рационального типа mpq_t



Особенности вычислений на основе интервального типа данных mpfi_t

Адаптация к MPI

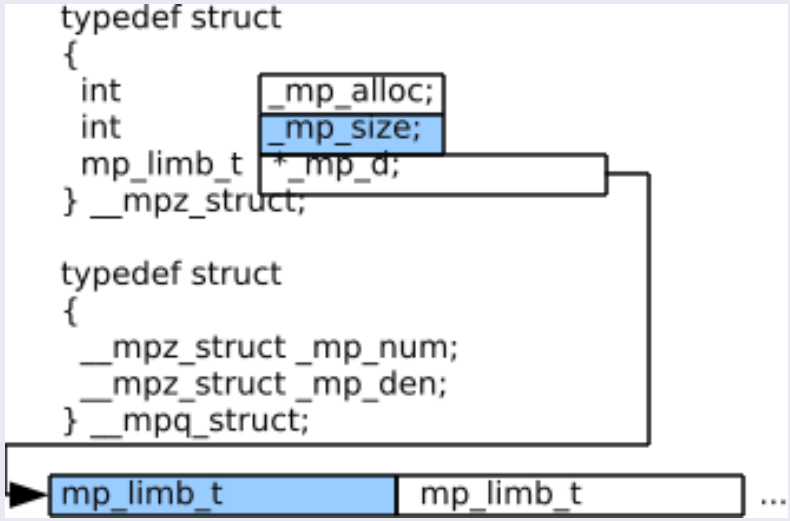
- переопределение типа и фиксирование точности
- бинарная совместимость и эффективность
- возможно определение MPI-типа
- отсутствие необходимости в пост обработке



Особенности вычислений на основе дробно-рационального типа данных mpq_t

Адаптация к MPI

- объекты переменной длины, непоследовательны в памяти
- нет возможности определить MPI-тип
- передача объектов на основе неполной сериализации



Особенности вычислений на основе дробно-рационального типа данных mpq_t

Пересылка объектов

- точка-точка (MPI_Send/MPI_Recv/...)

Источник	Приемник
сериализация объектов	
MPI_send	
	MPI_probe (получение размера буфера)
	MPI_recv
	десериализация объектов

- широковещательная передача от одного - многим (MPI_Bcast)

Источник	Приемник
сериализация объектов	
MPI_bcast	MPI_bcast (получение размера буфера)
MPI_bcast	MPI_bcast (получение данных)
	десериализация объектов

- вычисление max/min (MPI_Allreduce)
- сложная коллективная коммуникация (MPI_Gather/MPI_Scatter/MPI_Allgatherv/MPI_Alltoallv/...)

Спасибо за внимание!