ИПМ им.М.В.Келдыша РАН • Электронная библиотека

Препринты ИПМ • Препринт № 37 за 2017 г.

ISSN 2071-2898 (Print)
ISSN 2071-2901 (Online)

Капорин И.Е., Милюкова О.Ю.

Неполное обратное треугольное разложение в параллельных алгоритмах предобусловленного метода сопряженных градиентов

Рекомендуемая форма библиографической ссылки: Капорин И.Е., Милюкова О.Ю. Неполное обратное треугольное разложение в параллельных алгоритмах предобусловленного метода сопряженных градиентов // Препринты ИПМ им. М.В.Келдыша. 2017. № 37. 28 с. doi:10.20948/prepr-2017-37
URL: http://library.keldysh.ru/preprint.asp?id=2017-37

Ордена Ленина
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
имени М. В. Келдыша
Российской академии наук

И.Е. Капорин, О. Ю. Милюкова

Неполное обратное треугольное разложение в параллельных алгоритмах предобусловленного метода сопряженных градиентов

Москва — 2017

Капорин И.Е., Милюкова О.Ю.

Неполное обратное треугольное разложение в параллельных алгоритмах предобусловленного метода сопряженных градиентов

Для предобусловливания симметричной положительно определенной разреженной матрицы рассматривается ее приближенная обратная, представленная в виде произведения двух взаимно сопряженных разреженных треугольных матриц. Предложен алгоритм параллельной реализации построения и обращения этого предобусловливателя. Предложен новый метод предобусловливания блочного Якоби неполного обратного треугольного разложения. Проводится сравнение времени решения модельной задачи и тестовых задач из коллекции университета Флориды рассматриваемыми методами и методами сопряженных градиентов с предобусловливанием Якоби и блочного неполного обратного треугольного разложения второго порядка.

Ключевые слова: итерационное решение систем линейных алгебраических уравнений, разреженные матрицы, неполная обратная треугольная факторизация, параллельное предобусловливание, метод сопряженных градиентов

Igor Evgenevich Kaporin, Olga Yurievna Milyukova

Incomplete inverse triangular factorization in parallel algorithms of preconditioned conjugate gradient methods

A preconditioner for large sparse symmetric positive definite coefficient matrix is considered based on its approximate inverse in the form of product of a lower triangular sparse matrix by its transpose. A parallel algorithm for the construction and application of the preconditioner is proposed. A new approximate block Jacobi preconditioning method is proposed based on the use of the incomplete inverse triangular factorization of diagonal blocks. Timing results are presented for a model problem and test problems with matrices from the collection of the university of Florida for the proposed preconditioning in comparison with the 2^{nd} order Block Incomplene Inverse Cholesky and the standard point Jacobi preconditionings.

Keywords: iterative solution of linear systems, sparse matrices, incomplete inverse triangular factorization, parallel preconditioning, conjugate gradient method

Работа выполнена при финансовой поддержке РФФИ (коды проектов 17-01-00973-а, 17-07-00510)

1. Введение

Рассмотрим задачу приближенного решения системы линейных алгебраических уравнений (СЛАУ) большого размера

$$Ax = b \tag{1.1}$$

с симметричной положительно определенной разреженной матрицей A общего вида

$$A = A^T > 0.$$

Проблема построения соответствующих эффективных численных методов сохраняет свою актуальность, так как во многих важных прикладных областях продолжают возникать новые постановки таких задач. При этом сохраняется тенденция к росту размера матриц n, увеличению их заполненности ненулевыми элементами, усложнению их структуры разреженности, а также к ухудшению их обусловленности. В качестве примера такой постановки приведем конечноэлементные модели пространственных задач вычислительной механики, явившиеся источником большей части использованных в настоящей работе тестовых матриц [1] .

Известно, что существующие пакеты программ, основанные на применении «точного» треугольного разложения разреженной матрицы $A = U^T U$ (называемые «прямые решатели»), могут оказываться неэффективными из-за сильного заполнения треугольного множителя U ненулевыми элементами, которое иногда на несколько порядков превышает таковое для исходной матрицы. Для задач большого размера это может привести к неприемлемо высоким затратам объема памяти и времени счета (см., например, [2]).

В настоящая работе для решения СЛАУ большого размера используется предобусловленный метод сопряженных градиентов (CG), итерации которого осуществляются до выполнения условия

$$\|b - Ax_k\| \le \varepsilon \|b - Ax_0\|, \quad 0 < \varepsilon << 1. \tag{1.2}$$

Решение задач с матрицами очень большого размера требует применения параллельных компьютеров с большим числом процессоров. Ниже будут рассмотрены итерационные методы решения задачи (1.1), существенно использующие матрицы $H \approx A^{-1}$, называемые явными предобусловливателями. В этих методах основная доля вычислительной работы приходится на повторные умножения разреженной матрицы на вектор, а операции решения систем с треугольными матрицами отсутствуют. Поэтому реализующие их параллельные алгоритмы оказываются хорошо приспособлены к массивно-параллельной реализации на гибридных суперкомпьютерах.

В настоящей работе мы рассматриваем предобусловливание, предложенное в [3,4] (см. также [5]), основанное на предобусловливании матрицы A посредством приближенного обратного симметрично-треугольного разложения:

$$\hat{G} A \hat{G}^T = I_n + E, \tag{1.3}$$

где $\hat{G}$ - разреженная нижняя треугольная матрица с положительными диагональными элементами, а E представляет собой матрицу погрешности. Соответствующий предобусловливатель имеет вид

$$H \equiv \hat{G}^T \hat{G} \approx A^{-1}, \qquad (1.4)$$

причем структура разреженности $\hat{G}$ задается как множество ненулевых позиций, определяемых специальным образом, описанным ниже и состоящим в настоящей работе из двух этапов. Другие способы выбора структуры расположения ненулевых элементов $\hat{G}$ обсуждались, в частности, в [6-8].

Предполагается, что матрица является предварительно переупорядоченной матрицей коэффициентов, то есть $A_P = PAP^T$, где P - матрица перестановки, а A представляет собой матрицу коэффициентов исходной задачи. В формулах (1.1), (1.3), (1.4) предполагается, что матрица A уже переупорядочена, а вместо A_P стоит A. В настоящей работе применяются переупорядочения, уменьшающие среднюю ширину ленты матрицы, а именно, предложенные в работе [9] и описанные ниже, являющиеся обобщением упорядочения [10]. Подход, предложенный в этих работах, позволяет одновременно произвести разбиение области расчета на подобласти.

При построении предобусловливателя будем также предполагать, что матрица отмасштабирована, т. е. ее диагональные элементы равны единице. Это достигается с использованием формулы: $A_{SP} = D_{A_P}^{-1/2} A_P D_{A_P}^{-1/2}$, где D_{A_P} - диагональная часть матрицы A_P. Далее при описании предобусловливателей вместо A_{SP} будем использовать обозначение A, предполагая, что переупорядочение и масштабирование уже выполнены. Значения ненулевых элементов матрицы G на каждом этапе определяются из условия оптимизации K-числа обусловленности матрицы GAG^T (см. ниже формулу (2.1)). Такое предобусловливание сокращенно будем называть IIC (inverse incomplete Cholesky). При этом, очевидно, что $\hat{G} = GD_{A_P}^{-1/2}$, и структуры разреженности матриц $\hat{G}, G$ совпадают.

В настоящей работе описан метод построения предобусловливателя $G^T G$ для переупорядоченной и отмасштабированной матрицы A, предложен алгоритм его параллельной реализации, параллельный алгоритм реализации обращения предобусловливателя: $z = Hr = \hat{G}^T \hat{G} r$. Предложен новый метод предобусловливания блочного Якоби неполного обратного треугольного разложения (BJIIC). При построении этого предобусловливателя и при его обращении не требуется обмен информацией между процессорами. Приводятся результаты расчетов модельной задачи и ряда тестовых задач из коллекции университета Флориды [1]. Проводится сравнение времени решения тестовых задач рассматриваемым методом (IIC-CG), предложенным новым методом (BJIIC-CG), методом сопряженных градиентов с предобусловливанием Якоби (J-CG) и блочного неполного обратного треугольного разложения второго порядка (BIIC2-CG)[11].

2. Предобусловленный метод сопряженных градиентов

Пусть требуется решить СЛАУ (1.1), и для матрицы A ($A := A_P$) построена приближенная обратная вида $H \equiv \hat{G}^T \hat{G}$. Алгоритм предобусловленного метода CG (см., например, [12]) имеет следующий вид:

$r_0 = b - Ax_0$, $p_0 = w_0 = Hr_0$, $\gamma_0 = r_0^T p_0$,

для k=0,… пока $(r_k^T r_k) \le \varepsilon^2 (r_0^T r_0)$ выполнять

$q_k = Ap_k$,

$\alpha_k = \gamma_k / (p_k^T q_k)$,

$x_{k+1} = x_k + \alpha_k p_k$,

$r_{k+1} = r_k - \alpha_k q_k$,

$z_{k+1} = Hr_{k+1}$,

$\gamma_{k+1} = r_{k+1}^T z_{k+1}$,

$\beta_k = \gamma_{k+1} / \gamma_k$,

$p_{k+1} = z_{k+1} + \beta_k p_{k+1}$,

где $0 < \varepsilon << 1$. С учетом соотношения (1.4), очевидно, что этот алгоритм использует лишь операции умножения разреженных матриц на вектор, операции вычисления скалярных произведений и элементарные векторные операции. Поэтому принципиальная возможность его эффективной параллельной реализации не вызывает сомнений, даже при использовании большого числа процессоров (включая гибридные суперкомпьютеры).

В [4, 13] доказана оценка сходимости метода CG вида

$$\|r_k\|_H \le \left(K(HA)^{1/k} - 1\right)^{k/2} \|r_0\|_H ,$$

где

$$K(B) = (n^{-1} trace B)^n / \det B \qquad (2.1)$$

представляет собой K-число обусловленности симметричной матрицы $B = \hat{G} A \hat{G}^T$. При этом для числа итераций метода CG справедлива следующая (упрощенная) верхняя граница числа итераций:

$$k_{\varepsilon_1} \le \log_2 K(HA) + \log_2 (\varepsilon_1^{-1}), \qquad (2.2)$$

где $0 < \varepsilon_1 << 1$ задает требуемое уменьшение - H нормы невязки. Заметим, что в силу формул $A_{SP} = D_{A_P}^{-1/2} A_P D_{A_P}^{-1/2}$ и $\hat{G} = G D_{A_P}^{-1/2}$ верно: $K(HA_P) = K(H_s A_{SP})$, где $H_s = G^T G$.

Для того, чтобы обеспечить эффективную реализацию указанного выше итерационного метода, факторизованный предобусловливатель $H \equiv \hat{G}^T \hat{G}$ должен удовлетворять следующим требованиям: обладать хорошим качеством в смысле нужной аппроксимации A^{-1}; использовать достаточно разреженную матрицу $\hat{G}$.

3. Выбор значений ненулевых элементов матрицы *G*, обеспечивающий построение *K*-оптимального предобусловливания

Итак, предположим, что матрица *A* переупорядочена и отмасштабирована. Опишем алгоритм [14] построения нижнетреугольной невырожденной матрицы *G*, оптимальной в смысле минимизации величины $K(H_s A) = K(GAG^T)$ при фиксированной структуре разреженности треугольного множителя *G*. Допустим, что заранее заданы позиции структурно ненулевых элементов *i*-й строки нижнетреугольной матрицы *G:*

$$(G)_{i,j_i(1)},...,(G)_{i,j_i(m_i)},\ 1 \le i \le n, \qquad \text{где } 1 \le j_i(1) < ... < j_i(m_i) = i$$

представляет собой *i*-й список соответствующих столбцовых индексов. Тогда, как доказано в [14], минимум *K*-числа обусловленности $K(HA)$ (индекс *s* у H_s опущен), определенного выше соотношением (2.1), достигается при

$$(G)_{i,j_i(p)} = (S_i^{-1})_{p,m_i} / \sqrt{(S_i^{-1})_{m_i,m_i}}\ .$$

Здесь S_i- представляет собой главную подматрицу размера m_i матрицы *A*, построенную на указанном *i*-м подмножестве столбцовых индексов, то есть

$$(S_i)_{pq} = (A)_{j_i(p),j_i(q)},\quad 1 \le p \le m_i,\quad 1 \le q \le m_i\ .$$

Пусть z_i - вектор структурно ненулевых элементов *i*-той строки матрицы *G* длиной m_i ($z_i(p) = (G)_{i,j_i(p)}$). Нетрудно убедиться в том, что справедлива еще более простая формула для вычисления векторов z_i, выражающая их через симметричную треугольную факторизацию [14]

$$S_i = L_i L_i^T$$

в виде

$$z_i = L_i^{-T} v_i,$$

где вектор v_i длины m_i имеет вид: $v_i = (0,...,0,1)^T$. Таким образом, для каждого *i* достаточно вычислить треугольное разложение $S_i = L_i L_i^T$ матрицы порядка m_i, и затем решить одну треугольную систему $L_i^T z_i = v_i$ того же порядка.

4. Выбор позиций ненулевых элементов матрицы *G*

В настоящей работе используется метод выбора позиций ненулевых элементов матрицы *G*, описанный в [15] (см. параграф 3.2). Эта процедура осуществляется в 2 этапа. На первом этапе множество позиций ненулевых элементов искомой матрицы выбирается в виде множества позиций ненулевых элементов матрицы A^q, где *q* - показатель степени. Затем предварительно строится матрица $\tilde{G}$, как описано в разделе 3.

На втором этапе осуществляется прореживание множества позиций ненулевых элементов построенной матрицы $\tilde{G}$: отбрасываются позиции, в которых $0 < |g_{ij}| \le \tau_0 g_{ii}$, где $j < i$, g_{ij}- элементы матрицы $\tilde{G}$, $0 < \tau_0 << 1$. Затем

строится матрица G, как описано в разделе 3, в которой используется полученное прореженное множество ненулевых позиций.

Показатель степени q подбирается так, чтобы время решения СЛАУ (1.1) предобусловленным методом CG в сумме с временем построения предобусловливателя было минимально. Заметим, что для достаточно плотных матриц $q=1$ обычно приводит к хорошим результатам.

5. Алгоритм параллельной реализации

Пусть матрица A переупорядочена и разбита на блоки, причем на блочной диагонали расположены p квадратных блоков размера $n_s \times n_s$, $1 \le s \le p$. Обозначим $k_s = n_1 + ... + n_s$.

В каждом процессоре с номером s =1,…,p будем строить матрицу G_s, содержащую соответствующие n_s строк матрицы G. Сначала в каждом процессоре с номером s =1,…,p создадим матрицы $\overline{S}_s$ размера $m_s \times m_s$, где $m_s \ge n_s$, которые будут содержать все элементы матрицы A, необходимые для построения всех строк матрицы $\tilde{G}$ с номерами $k_{s-1}+1 \le i \le k_s$. Для этого нужно осуществить соответствующие пересылки. Заметим, что матрица $\overline{S}_s$ может содержать элементы матрицы A с номерами строк $i \le k_{s-1}$, содержит строки с номерами $k_{s-1}+1 \le i \le k_s$. Затем производится масштабирование матриц $\overline{S}_s$, что осуществляется всеми процессорами одновременно и независимо. При этом находятся элементы диагональной матрицы $D_{A_p,s}$, которые являются соответствующими элементами матрицы D_{A_p}.

В каждом процессоре при построении матрицы G_s используется алгоритм, аналогичный алгоритму при построении матрицы G (см. разделы 3,4), только вместо матрицы A используется отмасштабированная матрица $\overline{S}_s$. При этом для всех строк внутри каждой подобласти все вычисления происходят независимо. Затем во всех процессорах независимо и одновременно вычисляется матрица $\hat{G}_s = G_s D_{A_p,s}^{-1/2}$.

Параллельная реализации вычислений $z = Hr = \hat{G}^T \hat{G} r$ происходит следующим образом. Сначала производится пересылка из процессоров с меньшими номерами в процессоры с большими номерами значений r_m (здесь $m \le k_{s-1}$ индекс компоненты вектора r), необходимых для вычисления $\hat{z}_s = \hat{G}_s r$ в каждом процессоре с номером s. Затем в процессоре с номером s ($1 \le s \le p$) вычисляются $\hat{z}_s = \hat{G}_s r$ и соответствующие слагаемые $z_l = \hat{G}_s^T \hat{z}_s$ ($l \le s$, l- номер процессора, содержащего соответствующую компоненту вектора z). Далее, после выполнения необходимых пересылок (из процессоров с большими номерами в процессоры с меньшими номерами), в каждом процессоре с номером s производится суммирование вида $z = z_s + \sum_{l>s} z_l$, где z_l- нужная часть

вектора z, которая вычисляется в процессоре с номером l при выполнении в нем своей доли вычислений для получения $z = \hat{G}^T(\hat{G}r)$.

Матрица A хранится в памяти в виде верхнего треугольника. Подробное описание параллельной реализации умножения матрицы на вектор в этом случае приведено в [10]. Остальные этапы параллельной реализации предобусловленного метода CG хорошо известны и не представляют труда.

6. Предобусловливание при помощи блочного метода Якоби в сочетании с неполным обратным треугольным разложением

Пусть матрица A переупорядочена и разбита на блоки, причем на блочной диагонали расположены p квадратных блоков размера $n_s \times n_s$, $1 \le s \le p$. Обозначим, как и ранее, $k_s = n_1 + ... + n_s$. Определим прямоугольные матрицы

$$W_s = \left\lfloor e_{k_{s-1}+1} \middle| ... \middle| e_{k_s} \right\rfloor,$$

столбцы которых являются единичными n-векторами, где $k_{s-1}+1,...,k_s$ представляют собой индексы s-ого блока. Построим матрицы размерами $n_s \times n_s$ $W_s^T A W_s = A_s$. Проведем масштабирование матриц A_s, получим матрицы A_{0s} Построим неполные обратные треугольные разложения для этих матриц: $\overline{G}_s^T \overline{G}_s \approx A_{0s}^{-1}$. В качестве предобусловливателя будем использовать

$$H = \sum_{s=1}^{p} W_s D_{A_s}^{-1/2} \overline{G}_s^T \overline{G}_s D_{A_s}^{-1/2} W_s^T . \qquad (6.1)$$

Предобусловливание (6.1) будем называть блочное Якоби неполное обратное треугольное разложение (BJIIC). Вычисление элементов матриц $\overline{G}_s$ (s=1,…,p) осуществляется аналогично описанному в разделах 3,4, вместо отмасштабированной матрицы A используются матрицы A_{0s}. При вычислении матрицы $\overline{G}_s$ в каждом процессоре с номером s=1,…,p не требуется информации, хранящейся в других процессорах. Кроме того, каждая строка матрицы $\overline{G}_s$ вычисляется независимо от других строк этой матрицы. В процессе вычисления матрицы предобусловливателя все процессоры могут работать одновременно и независимо, пересылок не требуется. При выполнении операции

$$z = Hr,$$

где матрица H определена в (6.1), тоже все процессоры могут работать одновременно и независимо, пересылок не требуется.

Заметим, что блочное Якоби предобусловливание имеет вид:

$$H = \sum_{s=1}^{p} W_s A_s^{-1} W_s^T .$$

7. Оценки уменьшения K-числа обусловленности в методах BJIIC и IIC

Мы предполагаем, что матрица A предварительно переупорядочена и отмасштабирована (см. выше). Прежде всего, заметим, что предобусловливания BJIIC и IIC отвечают одному и тому же методу, но с разным выбором структуры заполненности матрицы G, а именно, из структуры ненулей матрицы G метода IIC удаляются все блочно-внедиагональные позиции, и получается структура матрицы $\overline{G}$ метода BJIIC.

Теперь напомним, что, согласно выкладкам на с.104 диссертации [13], справедлива следующая оценка уменьшения K-числа обусловленности, которое достигается при применении IIC-предобусловливания:

$$\frac{K(G^T GA)}{K(A)} \le \exp\left(-\frac{1}{\|A\|} \sum_{i=1}^{n} a_i^T a_i \right) = \exp\left(-\frac{1}{\|A\|} \sum_{i=1}^{n} \sum_{s=1}^{m_i - 1} A_{i,j_i(s)}^2 \right),$$

где $j_i(s)$ для метода IIC были определены выше в п.3. Заметим также, что при $q \ge 1$ для выбранного нами способа построения структуры G и способа масштабирования матрицы A в методе IIC, справедлива формула

$$\sum_{i=1}^{n} \sum_{s=1}^{m_i - 1} A_{i,j_i(s)}^2 = \frac{1}{2} \|I - A\|_F^2 ,$$

где $\|*\|_F$ обозначает фробениусову норму матрицы. Если же обозначить теперь индексы ненулевых элементов $\overline{G}$ через $\bar{j}_i(s)$, то для метода BJIIC получим

$$\frac{K(\overline{G}^T \overline{G}A)}{K(A)} \le \exp\left(-\frac{1}{\|A\|} \sum_{i=1}^{n} \sum_{s=1}^{m_i - 1} A_{i,\bar{j}_i(s)}^2 \right),$$

Теперь, напоминая обозначение $A_0 = BlockDiag(A)$, можно воспользоваться очевидными соотношениями

$$\sum_{i=1}^{n} \sum_{s=1}^{m_i - 1} A_{i,\bar{j}_i(s)}^2 = \frac{1}{2} \|I - A_0\|_F^2$$

и

$$\|I - A\|_F^2 = \|I - A_0\|_F^2 + \|A - A_0\|_F^2 ,$$

так что сравнение двух предобусловливаний дают следующие два неравенства:

$$\frac{K(G^T GA)}{K(A)} \le \exp\left(-\frac{\|I - A\|_F^2}{2\|A\|} \right)$$

для метода IIC, и

$$\frac{K(\overline{G}^T \overline{G}A)}{K(A)} \le \exp\left(-\frac{\|I - A_0\|_F^2}{2\|A\|} \right) = \exp\left(-\frac{\|I - A\|_F^2}{2\|A\|} \right) \exp\left(\frac{\|A - A_0\|_F^2}{2\|A\|} \right)$$

для метода BJIIC. Таким образом, получаем, что оценка числа итераций в обоих методах меньше, чем в методе CG с предобусловливанием Якоби. Кроме того,

здесь видно, что за ухудшение оценки BJIIC по сравнению с IIC отвечает величина фробениусовой нормы $\|A - A_0\|_F$ блочно-внедиагональной части матрицы *A*. Таким образом, если матрица A несильно отличается от A_0, можно ожидать, что правые части этих оценок отличаются не очень сильно.

В частности, для задачи Дирихле для уравнения Пуассона, когда разбиение области расчета происходит на квадратные или кубические подобласти получаем $K(\overline{G}^T \overline{G} A) > K(G^T G A)$; оценка числа итераций в методе BJIIC-CG хуже, чем в методе IIC-CG.

Матрица A_0 несильно отличается от A при достаточно хорошем разбиении, когда вне блочной диагонали оказывается не слишком много ненулевых элементов A. Об одном из таких разбиений речь пойдет ниже. Желательно также, чтобы отбрасываемые элементы A были небольшими по модулю.

8. Переупорядочение и разбиение графа матрицы

Для параллельной реализации матричных операций, нам требуется определить переупорядочение и разбиение графа разреженной матрицы A, имеющей порядок n и симметричную структуру разреженности, содержащую $nz(A)$ ненулевых элементов. Соответствующий неориентированный граф образуется множеством n вершин *V* и множеством $nz(A)/2$ ребер *E*, соединяющих вершины с номерами *i* и *j*, если $(A)_{ij} \neq 0$.

Для описания неориентированного графа симметризованной структуры разреженности рассматриваемых матриц используем стандартный формат CRS, (compressed row sparse) используемый для представления разреженных матриц. Эта структура задается массивом IA длины $n+1$, где n обозначает общее число узлов графа, причем *IA(0)=0* и $IA(i+1) - IA(i)$ равно количеству узлов графа, смежных с *i*-м узлом, а *JA(IA(i)+1), … , JA(IA(i+1))* представляют собой номера этих узлов графа.

Задача разбиения разреженной матрицы ставится с целью эффективной параллельной реализации ее умножения на произвольный плотный вектор, $z = Av$. Эта операция (необходимая при реализации итерационных методов решения систем линейных алгебраических уравнений) обычно организуется путем симметричного переупорядочения $n \times n$-матрицы A (с симметричной структурой разреженности) и разбиения ее на блоки A_s размеров $n_s \times n$ (для указания границ блоков используется массив $IBL(k+1) = n_1 + ... + n_k$, где $IBL(1) = 0$), и тем же образом произведено разбиение векторов *z* и *v*. Таким образом, блочная диагональ матрицы *A* образуется p квадратными блоками A_{ss} порядка n_s, $1 \le s \le p$. При таком распределении данных по параллельным процессам, перед тем, как производить в каждом процессоре вычисление по формуле

$$z_s = A_s v, \text{ где } 1 \le s \le p,$$

требуется предварительно сделать нужные пересылки некоторых компонент подвекторов v_s между параллельными процессами. (Номера этих компонент, очевидно, являются номерами ненулевых столбцов матриц A_s, не входящих в диагональные блоки A_{ss})

Таким образом, для эффективной параллельной реализации умножения матрицы на вектор выбор перенумерации и разбиения графа разреженной матрицы достаточно подчинить требованию сдерживания коммуникационных затрат на операцию обновления данных, связанных с «приграничными» узлами каждого подграфа (блока).

Одним из главных возможных источников снижения эффективности параллельной обработки может служить нарастание дисбаланса локальных размеров вычислительных подзадач с увеличением числа процессоров. Для этого используется стратегия разбиения графа на p подграфов примерно равного размера $\approx n/p$ за счет удаления (разрезания) тех дуг, которые отвечают связям с «приграничными» узлами. При этом основным критерием качества разбиения обычно служит общее количество разрезанных ребер (EdgeCut).

Таким образом, искомые переупорядочение и разбиение должны обеспечить достаточную концентрацию большей части ненулевых элементов матрицы в ее блочно-диагональной части, состоящей из p квадратных блоков близкого размера.

Из приведенного выше обсуждения следует целесообразность выбора следующих критериев качества переупорядочения и разбиения матрицы смежности графа:

(i) Уменьшить число ненулевых элементов вне блочной диагонали (равное удвоенному числу разрезанных ребер CutSize);

(ii) Уменьшить число внешних вершин для каждого блока, общее число которых по всем блокам обозначается далее как Tot_ovl (т.е. таких вершин, которые связаны ребром с какой-либо вершиной из блока, но не принадлежат блоку);

(iii) Уменьшить число блоков, с которыми через «приграничные» множества связан каждый блок;

(iv) Обеспечить сбалансированность объемов вычислительной работы, связанной с каждым из блоков.

Опишем сначала простейшую процедуру [9,10], аналогичную предложенной в [16], которая позволяет найти p стартовых узлов, пригодных для инициализации процесса разбиения графа.

Для первоначального разбиения разреженной $n \times n$-матрицы смежности A на блоки используется следующий примитивный, но весьма надежный Алгоритм 1. Как уже упоминалось выше, целью используемой перенумерации является концентрация большей части ненулевых элементов матрицы в ее блочно-диагональной части, состоящей из p квадратных блоков почти одинакового размера.

Рассмотрим неориентированный граф с n вершинами, где i-я вершина соединена ребром с j-й, если $(A)_{ij} \neq 0$. Построим целочисленные массивы *ioc(1:n)* и *ior(1:n)*, задающие искомые перестановки строк и столбцов матрицы смежности A:

$$\hat{A} = PAP^T, \qquad (Px)_i = x_{ior(i)}, \qquad ioc(ior(i)) = i.$$

В Алгоритме 1 используются следующие три множества вершин:

(а) непомеченные вершины;

(б) помеченные вершины (вышеуказанные два множества не пересекаются, а их объединение составляет множество всех вершин);

(в) включенные (в текущий блок) вершины (являющиеся подмножеством помеченных вершин).

При этом, как только завершается построение текущего блока, все помеченные, но не включенные вершины, снова полагаются непомеченными.

Опишем этапы выполнения Алгоритма 1 подробнее.

Предварительно вычислим размеры всех блоков, полагая их равными

$$n_k = \begin{cases} \lfloor n/p \rfloor + 1, & при \quad 1 \le k \le n - p\lfloor n/p \rfloor, \\ \lfloor n/p \rfloor, & при \quad n - p\lfloor n/p \rfloor < k \le p. \end{cases}$$

Соответственно, через величины n_k определяем массив границ блоков *IBL(1:p+1).*

Сначала считаем все вершины непомеченными, полагая

$$ioc(j) := 0, \quad j = 1,...,n.$$

Инициализируем счетчик блоков $k := 1$ и счетчик включенных вершин $kc := 0$; будем также использовать счетчик помеченных вершин $kl \ge kc$.

L1: Находим любую непомеченную вершину; пусть i - ее номер. (Если же таковой не нашлось, переходим к L3.) Полагаем

$$kc := kc + 1, \qquad ior(kc) := i, \qquad ioc(i) := kc.$$

Инициализируем список помеченных, но невключенных вершин:

$$kl := kc.$$

L2: Полагаем $i := ior(kc)$ и просматриваем все вершины, связанные ребрами с i-й вершиной, помечая все непомеченные:

for all $(j : (A)_{ij} \neq 0)$ **do**

if $(ioc(j) = 0)$ **then**

$kl := kl + 1,\ ior(kl) := j,\ ioc(j) := kl$

endif

enddo

Если список помеченных, но невключенных вершин пустой, то переходим к поиску непомеченной вершины:

if $(kl = kc)$ **go to** L1;

в противном случае включаем вершину с наименьшим номером из списка помеченных, но пока не включенных:

$$kc := kc+1, \quad ioc(ior(kc)) = kc.$$

Если текущий блок не закончен (то есть $kc < n_1 + ... + n_k$), то переходим к L2; в противном случае помечаем все вершины из списка помеченных, но пока не включенных, как непомеченные:

$$ioc(ior(j)) = 0, \qquad ior(j) := 0, \quad j = kc+1,...,kl,$$

полагаем $k := k+1$ и переходим к L1.

L3: Обращаем построенный порядок нумерации вершин и порядок следования подобластей:

$$ioc(i) = n+1-ioc(i), \quad ior(ioc(i)) = i, \quad i = 1,...,n.$$

В некоторых случаях, см., напр., [10], это может существенно улучшить эффективность использования такого упорядочения.

Затраты времени на построение такого переупорядочения пропорциональны $nz(A)$ (числу ненулевых элементов матрицы A), т.е. весьма невелики.

Заметим, что некоторые блоки (обычно с небольшими номерами) в получаемом упорядочении могут быть несвязными (как, напр., первый блок на Рис. 1 в работе [10]). Это обстоятельство часто приводит к заметному увеличению дисбалансов размеров «приграничных» подмножеств блоков (определенных в п.2), а также к неоправданному увеличению числа разрезанных ребер EdgeCut. Более того, наиболее неприятным явлением может оказаться наличие неоправданно большого количества непосредственных «соседей» (т.е., других блоков, имеющих общую вершину через разрезанное ребро) у некоторых блоков (в частности, несвязных), см. выше п.(iii). Поэтому предлагается следующая модификация, которую назовем Алгоритм 2. Аналогичные подходы были рассмотрены, напр., в [17,18].

На первом этапе применяется Алгоритм 1. Затем в каждой из p построенных подобластей выбирается один стартовый узел, например как

$$i_s = ior(IBL(s)+2+IBL(s+1))/2), \quad s=1,...,p.$$

С этих узлов стартуют p «копий» Алгоритма 1, запускаемых на равных правах, но в порядке специальной очередности. Из всех p «копий» Алгоритма 1, только одной из них разрешается сделать один шаг добавления окрестности последнего включенного в s-ю подобласть узла (шаг L2). После этого производится сравнение всех текущих значений количества включенных узлов для каждой из p строящихся подобластей, и находится та из них, для которой это значение минимально. Тогда уже для нее делается следующий шаг L2, и т.д. Таким образом, на каждом элементарном шаге может наращиваться на один узел только та подобласть, которая содержит в себе узлов не больше любой другой. Такой подход способствует лучшей балансировке подобластей. Заметим, что на завершающих этапах алгоритма может оказаться, что подобласть с наименьшим числом узлов не может быть пополнена (так как окружена узлами, уже

принадлежащими другим подобластям). В таком случае она остается неизменной, и обрабатываются другие подобласти, где еще возможно достраивание.

Описанная процедура вполне работоспособна, и если исходный граф изначально является связным, то она всегда приводит к построению связных блоков. При этом обеспечивается неплохая балансировка подобластей по количеству узлов, если только значение *p* относительно *n* и/или среднее число дуг, приходящееся на каждый узел, не слишком велики.

Понятно, что к полученному разбиению можно снова применить тот же цикл построения (при этом будут выбраны, вообще говоря, другие стартовые вершины). Так можно делать несколько раз (например, 10), и выбирать тот результат, для которого, например, значение EdgeCut будет наименьшим.

9. Предобусловливание при помощи блочного неполного обратного треугольного разложения и предобусловливание Якоби

Рассмотрим метод предобусловливания BIIC2, предложенный в [11]. Пусть матрица *A* переупорядочена и разбита на блоки, причем на блочной диагонали расположены *p* квадратных блоков размера $n_s \times n_s$, $1 \le s \le p$. Обозначим, как и ранее, $k_s = n_1 + ... + n_s$, и пусть $m_s \ge n_s$ - размеры расширенных блоков. Определим прямоугольные матрицы

$$V_s = \left[e_{j_s(1)} \middle| ... \middle| e_{j_s(m_s - n_s)} \middle| e_{k_{s-1}+1} \middle| ... \middle| e_{k_s} \right],$$

столбцы которых являются единичными *n*-векторами, где $k_{s-1}+1,...,k_s$ представляют собой индексы *s*-ого блока, а $j_s(1),..., j_{s(m_s - n_s)}$ являются индексами перекрытия, причем $j_s(.) \le k_{s-1}$. Используя приближенное треугольное разложение второго порядка «по значению», описанное в [19], для аппроксимации $m_s \times m_s$ подматриц

$$V_s^T A V_s = U_s^T U_s + U_s^T R_s + R_s^T U_s - S_s,$$

где $\|R_s\| = O(\tau), \|S_s\| = O(\tau^2)$ и $0 < \tau << 1$ - порог отсечения, определим предобусловливатель формулой

$$H = \sum_{s=1}^{p} V_s U_s^{-1} \begin{bmatrix} 0 & 0 \\ 0 & I_{n_s} \end{bmatrix} U_s^{-T} V_s^T.$$

Как известно, в предобусловливании Якоби $H = D_A^{-1}$, где D_A - диагональная часть матрицы *A*. Доказано [14], что предобусловливание Якоби в алгоритме метода CG минимизирует *K* число обусловленности на классе диагональных предобусловливателей.

10. Результаты расчетов

Все программы, реализующие применение методов IIC-CG, BJIIC-CG, J-CG, BIIC2-CG для решения системы уравнений $Ax = b$, где $A = A^T > 0$, были написаны на языке FORTRAN с использованием MPI, расчеты производились на многопроцессорной вычислительной системе МВС-10П, установленной на МСЦ РАН. Матрица *A* хранилась в распределенном CRS-формате, см. выше п.8.

Тестирование и сравнение методов производилось с помощью расчетов модельной задачи – разностной задачи Дирихле для уравнения Пуассона в единичном квадрате на ортогональной сетке, причем *n*=1048576. Использовалась стандартная 5-точечная аппроксимация лапласиана (имя матрицы 5_1048576). Для тестирования рассматриваемых параллельных методов использовались также некоторые матрицы из коллекции университета Флориды [1]. Перечислим имена используемых тестовых матриц и укажем источник их происхождения:

s3dkt3m2 - модель цилиндрической оболочки на треугольной сетке;
thread - контактная задача (соединитель с резьбовым сочленением);
x104 - моделирование конструкций (балочное сочленение);
m_t1 - моделирование конструкций (трубчатое сочленение);
hood - моделирование конструкций;
pwtk - модель воздуховода под давлением;
msdoor- моделирование конструкций (дверь среднего размера).

В таблице 1 перечислены некоторые свойства этих матриц, причем значения $Cond(A_S)$, где $A_S = (D_A)^{-1/2} A (D_A)^{-1/2}$ - матрица системы уравнений после масштабирования, взяты из работы [14], Id – количество строк без диагонального преобладания, Ip – количество положительных внедиагональных элементов, NZA – число ненулевых элементов матрицы *A*, $nz_{\min}$, $nz_{\max}$ - минимальное и максимальное число ненулевых элементов в строках матрицы *A*.

Таблица 1

Свойства некоторых матриц из коллекции университета Флориды

Матрица	N	NZA	Id	Ip	$nz_{\min}$	$nz_{\max}$	$Cond(A_S)$
m_t1	97578	9753570	1	4648398	48	237	0.47+10
Hood	220542	9895422	9910	4879422	1	77	0.55+6
Pwtk	217918	11524432	325	5407348	2	180	0.26+9
msdoor	415863	19173163	11125	9350756	1	77	0.19+9
s3dkt3m2	90449	3686223	0	1765314	7	42	0.31+11
x104	108384	8713602	2255	4059880	8	270	0.1+11
thread	29736	4444880	0	2171496	28	306	0.89+10

Решалось уравнение $Ax = b$, где правая часть $b_i \equiv 1$, начальное приближение $x_0 \equiv 0$, счет продолжался до выполнения условия (1.2), где $\varepsilon = 10^{-8}$. Для разбиения области расчета при решении всех задач использовался способ, описанный в разделе 8, с 4 итерациями. Для решения задачи с матрицей **thread** использовался также способ [10]. При построении предобусловливателей IIC во всех задачах, кроме модельной, использовались значения параметров $\tau_0 = 0.01$, $q = 1$, в модельной задаче $\tau_0 = 0.01$, $q = 2$. Такой выбор q был продиктован минимизацией времени счета задачи. При построении предобусловливателя BJIIC во всех задачах из коллекции университета Флориды использовались значения параметров $\tau_0 = 0.01$, $q = 1$. При решении модельной задачи $\tau_0 = 0.01$, $q = 1$ при p<100, и q=2 в противном случае (из соображений минимальности времени счета задачи).

При построении предобусловливателей BIIC2 в модельной задаче использовались значения параметров $q_1 = 4$, $\tau = 0.01$, в задаче **s3dkt3m2** $q_1 = 1$ $\tau = 0.01$, в задачах **hood, pwtk, msdoor** использовались значения параметров $q_1 = 2$, $\tau = 0.002$, в задаче **thread** $q_1 = 1$, $\tau = 0.002$, а в задачах **m_t1, x104** $q_1 = 1$, $\tau = 0.0002$, где q_1 глубина налегания. Выбор этих параметров был осуществлен ранее и продиктован минимизацией времени решения задачи методом BIIC2–CG при использовании разбиения [10].

В таблицах 2,5 приведены времена счета задач (в секундах), состоящие из времен вычисления предобусловливателя и счета итерационного процесса, методами IIC-CG, BJIIC-CG, J-CG, BIIC2-CG для различного числа процессоров.

Заметим, что незаполненные в таблице 2 и во всех следующих таблицах клетки соответствуют случаям, когда увеличение числа используемых процессоров может привести к ухудшению масштабируемости методов.

В таблицах 3, 6 приведены ускорения счета задач на p процессорах, по сравнению с 8 процессорами ($s_p = t_p / t_8$, где t_p - время счета на p процессорах).

В таблицах 4, 7 приведены значения чисел итераций методов IIC-CG, JBIIC-CG, J-CG, BIIC2-CG для различного числа процессоров. Как видно из таблицы 4, при использовании метода BJIIC-CG для решения задач из коллекции университета Флориды наблюдается некоторый рост числа итераций с ростом числа процессоров. Однако, благодаря значительному уменьшению числа пересылок, время счета методом BJIIC-CG обычно оказывается немного меньше времени счета методом IIC-CG (см. табл. 2).

Таблица 2

Времена счета задач методом CG: число сверху – с предобусловливателем IIC, ниже с предобусловливателями BJIIC, Якоби и BIIC2

	p=8	*p*=16	*p*=32	*p* =60	*p* =100	*p* =160	*p* =200
5_1048576	6.35	4.93	2.05	0.89	0.51	0.51	0.38
BJIIC	6.47	5.58	2.14	0.92	0.6	0.44	0.38
J	4.73	4.69	0.92	0.54	0.33	0.3	0.27
BIIC2	2.59	1.96	1.0	0.47	0.24	0.18	0.17
m_t1(IIC)	15.68	9.80	5.47	2.82	2.15	2.18	1.88
BJIIC	15.75	8.85	4.84	2.29	1.57	1.61	1.41
J	662.83	460.15	188.02	112.43	87.37	90.84	93.95
BIIC2	27.71	16.06	6.38	2.51	1.21	0.94	0.93
hood (IIC)	4.09	2.42	1.31	0.70	0.56	0.43	0.39
BJIIC	3.58	2.21	1.14	0.62	0.37	0.28	0.24
J	9.55	8.28	2.42	1.49	1.07	1.12	1.11
BIIC2	4.31	2.52	1.15	0.63	0.36	0.22	0.19
pwtk (IIC)	16.95	9.64	5.00	2.13	1.60	1.26	1.2
BJIIC	16.39	9.63	4.84	1.96	1.46	1.07	1.0
J	151.65	109.98	28.49	15.81	12.45	11.28	11.35
BIIC2	14.11	9.31	4.2	2.55	1.22	0.8	0.68
msdoor(IIC)	32.43	19.85	9.66	5.17	2.95	1.97	1.90
BJIIC	31.94	21.57	13.13	7.74	3.32	2.78	2.21
J	325.16	258.75	102.15	33.36	20.27	19.14	18.08
BIIC2	33.4	14.66	9.48	4.37	2.27	1.27	0.96
x104 (IIC)	23.11	14.55	6.84	4.34	2.98	3.12	
BJIIC	22.31	14.42	6.73	3.91	2.63	2.72	
J	1108.7	738.59	272.15	208.75	142.76	75.36	
BIIC2	75.78	52.07	25.73	8.83	7.11	6.37	
s3dtkt3m2(IIC)	7.39	4.19	2.09	1.26	1.10	1.18	
BJIIC	7.14	4.02	2.09	1.27	1.04	0.97	
J	25.88	13.64	8.44	6.36	5.39	5.61	
BIIC2	3.71	2.14	1.09	0.59	0.45	0.40	

Заметим, что, как показывают расчеты, для матриц 5_1048576**, m_t1, hood, pwtk, msdoor, x104, s3dtkt3m2** использование упорядочения, описанного выше, позволяет уменьшить рост итераций с ростом числа процессоров в методе BJIIC-CG по сравнению с его ростом при использовании упорядочения [10].

Таблица 3

Ускорения счета задач на p процессорах по сравнению с 8 процессорами

Идеальное ускорение	*p*=8	*p* =16 2	*p* =32 4	*p* =60 7.5	*p* =100 12.5	*p* =160 20	*p* =200 25
5_1048576(IIC)	1.0	1.29	3.09	7.13	12.45	12.45	16.71
BJIIC	1.0	1.16	3.02	7.03	10.78	14.7	17.02
J	1.0	1.0	5.14	8.76	14.33	15.76	17.51
BIIC2	1.0	1.32	2.59	5.51	10.8	14.38	15.23
m_t1(IIC)	1.0	1.61	2.88	5.59	7.33	7.23	8.39
BJIIC	1.0	1.77	3.25	6.87	10.03	9.78	11.17
J	1.0	1.44	3.52	5.9	7.58	7.29	7.05
BIIC2	1.0	1.72	4.34	11.04	22.9	29.5	29.8
hood(IIC)	1.0	1.69	3.12	5.84	7.35	9.51	10.48
BJIIC	1.0	1.62	3.14	5.77	9.67	12.78	14.91
J	1.0	1.15	3.94	6.41	8.92	8.52	8.6
BIIC2	1.0	1.71	3.74	6.84	11.97	19.59	22.68
pwtk(IIC)	1.0	1.75	3.39	7.95	10.59	13.45	14.12
BJIIC	1.0	1.70	3.38	8.36	11.22	15.31	16.39
J	1.0	1.37	5.32	9.59	12.18	13.44	13.36
BIIC2	1.0	1.51	3.36	5.53	11.56	17.63	20.75
msdoor(IIC)	1.0	1.63	3.36	6.27	10.99	16.46	17.06
BJIIC	1.0	1.48	2.43	4.12	9.62	11.48	14.45
J	1.0	1.25	3.18	9.74	16.04	16.98	17.98
BIIC2	1.0	2.27	3.52	7.64	14.71	26.29	34.79
x104(IIC)	1.0	1.63	3.37	5.32	7.75	7.4	
BJIIC	1.0	1.54	3.31	5.7	8.48	8.2	
J	1.0	1.50	4.07	5.31	7.76	14.71	
BIIC2	1.0	1.45	2.94	8.58	10.66	11.89	
s3dtkt3m2(IIC)	1.0	1.76	3.53	5.86	6.71	6.26	
BJIIC	1.0	1.77	3.41	5.62	6.86	7.36	
J	1.0	1.89	3.06	4.07	4.8	4.61	
BIIC2	1.0	1.73	3.40	6.28	8.24	9.27	

Таблица 4

Числа итераций при расчетах задач методом CG
с различными предобусловливателями

	p =8	p =16	p =32	p =60	p =100	p =160	p =200
5_1048576(IIC)	1211	1176	1200	1169	1162	1164	1199
BIIC	1824	1777	1953	1949	1517	1548	1560
J	1898	1898	1898	1898	1898	1898	1898
BIIC2	300	310	330	338	350	364	369
m_t1(IIC)	3053	3066	3044	3012	2938	2903	2887
BJIIC	3334	3292	3438	3599	3769	4271	4336
J	536545	536667	534807	536474	537466	537249	535102
BIIC2	1500	1311	1143	1053	1134	1041	1291
hood (IIC)	426	410	415	410	415	410	428
BJIIC	426	448	460	469	454	502	491
J	6078	6056	6052	6061	6059	6063	6066
BIIC2	71	60	55	48	46	54	57
pwtk (IIC)	3693	3575	3597	3541	3545	3514	3500
BJIIC	3775	3773	3964	3962	4158	4239	4343
J	74111	74065	74119	74692	74676	74029	74088
BIIC2	1550	1317	1261	1542	1345	1374	1436
msdoor (IIC)	3935	3888	3853	3845	3800	3793	3760
BJIIC	4137	4427	5638	6537	5217	6366	5566
J	83510	83496	83531	83493	83466	83504	83488
BIIC2	1106	823	923	798	671	597	596
x104 (IIC)	8354	8405	8084	8146	7910	7569	
BJIIC	8411	8864	9237	9250	9517	8411	
J	884621	888104	882639	884943	882803	887428	
BIIC2	15581	14034	13350	10647	10501	9168	
s3dtkt3m2(IIC)	6594	6674	6463	6463	5971	6086	
BJIIC	6859	7107	7327	7527	7495	7814	
J	61661	61608	60266	61025	62299	60280	
BIIC2	1241	1195	1673	1527	1669	1800	

Ниже приведены результаты расчетов задачи с матрицей **thread**, в которых при относительно небольшом числе процессоров при использовании упорядочения, описанного выше, наблюдается неожиданно большое число итераций метода BJIIC-CG. Возможно, это связано с высокой плотностью матрицы и ее небольшим размером, а также большими по модулю значениями элементов матрицы $A - A_0$. В таблицах 5-7 в верхней части приведены результаты расчета с описанным выше упорядочением, в нижней части – с упорядочением [10].

Таблица 5

Время счета задачи с матрицей **thread** методом CG с различными предобусловливателями при различных способах разбиения

	p=8	*p*=16	*p*=32	*p*=60	*p*=100	*p*=160
thread(IIC)	9.28	6.79	4.26	2.92	3.00	3.26
BJIIC	93.71	87.51	77.00	21.92	9.84	29.71
J	188.52	146.87	110.44	122.91	135.25	169.28
BIIC2	23.29	21.93	12.33	3.85	4.28	5.34
thread (IIC)	9.27	5.09	3.32	2.93	2.8	3.05
BJIIC	53.8	25.83	24.21	21.03	9.68	28.76
J	187.8	106.7	86.6	125.4	137.0	166.5
BIIC2	10.31	8.56	6.03	3.8	3.7	4.8

Таблица 6

Ускорения счета задачи с матрицей **thread** на p процессорах по сравнению с 8 процессорами при различных способах разбиения

	p=8	*p*=16	*p*=32	*p*=60	*p*=100	*p*=160
thread(IIC)	1.0	1.37	2.18	3.18	3.09	3.09
BJIIC	1.0	1.07	1.22	4.27	9.52	9.52
J	1.0	1.28	1.7	1.53	1.39	1.39
BIIC2	1.0	1.06	1.89	6.05	5.42	5.42
thread (IIC)	1.0	1.82	2.79	3.16	3.31	3.03
BJIIC	1.0	2.08	2.22	2.56	5.56	1.87
J	1.0	1.76	2.16	1.5	1.37	1.13
BIIC2	1.0	1.2	1.71	2.71	2.78	2.15

Таблица 7

Числа итераций при расчетах задачи с матрицей **thread** методом CG с различными предобусловливателями при различных способах разбиения

	p=8	*p*=16	*p*=32	*p*=60	*p*=100	*p*=160
thread(IIC)	4734	5137	5101	3798	3970	3918
BJIIC	124671	162015	252373	74697	31788	81813
J	474022	476645	478433	478372	477343	476817
BIIC2	6285	6112	6102	3834	3887	3906
thread (IIC)	4113	3891	3766	3798	3970	3918
BJIIC	75103	64463	95232	74697	31788	81813
J	477502	47453	480264	478372	477343	476817
BIIC2	4299	4184	4181	3834	3887	3906

На рис. 1-9 представлены графики зависимости времен счета этих задач от числа процессоров в логарифмическом масштабе для различных методов. Заметим, что дальнейшее увеличение числа используемых процессоров может привести к ухудшению масштабируемости методов.

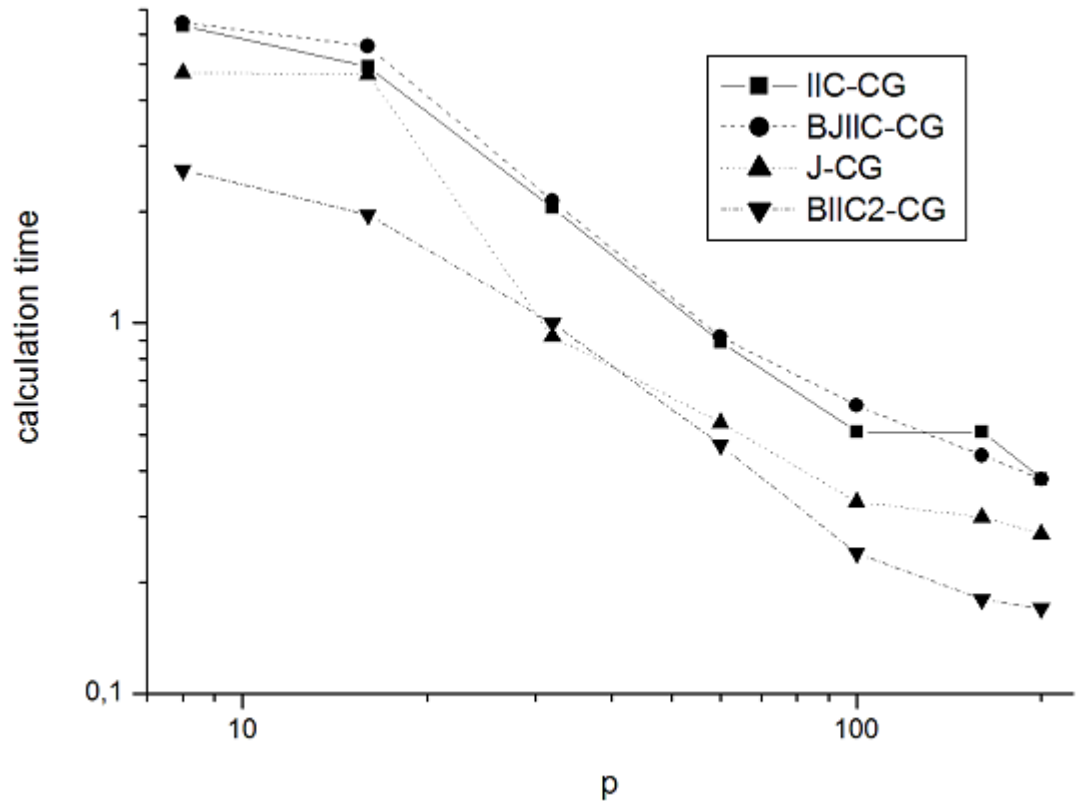

Рис.1. Время счета задачи с матрицей 5_1048576 различными методами

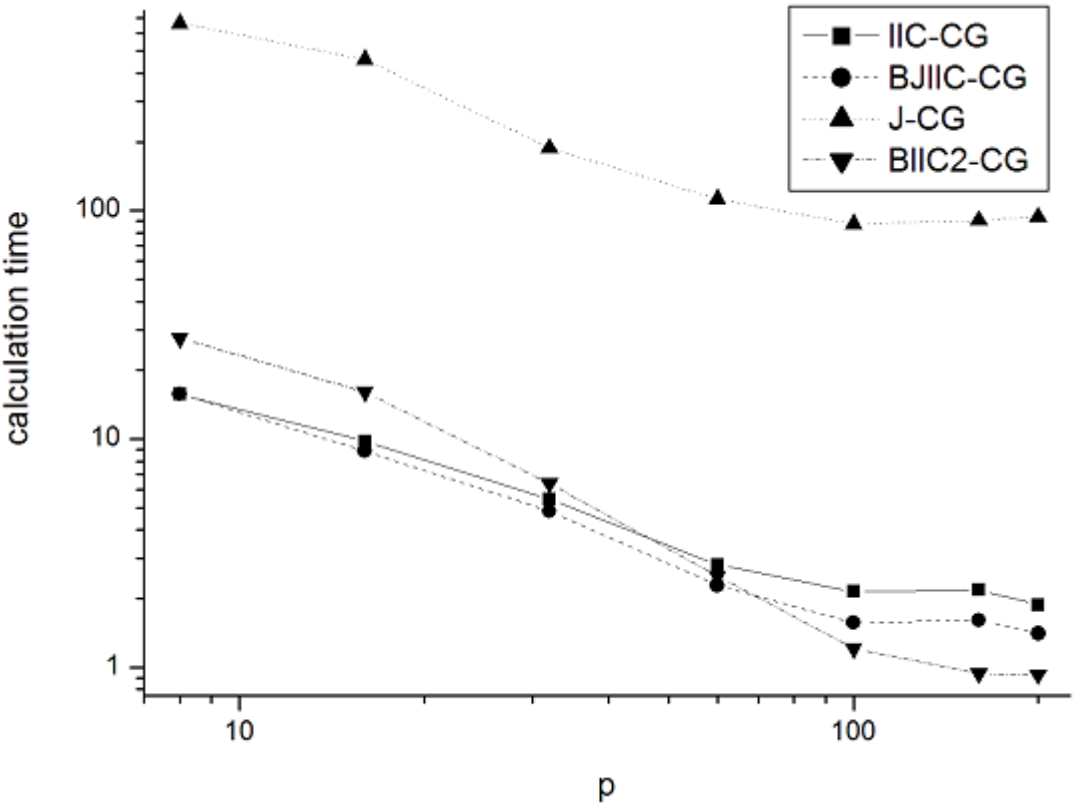

Рис.2. Время счета задачи с матрицей **m_t1** различными методами

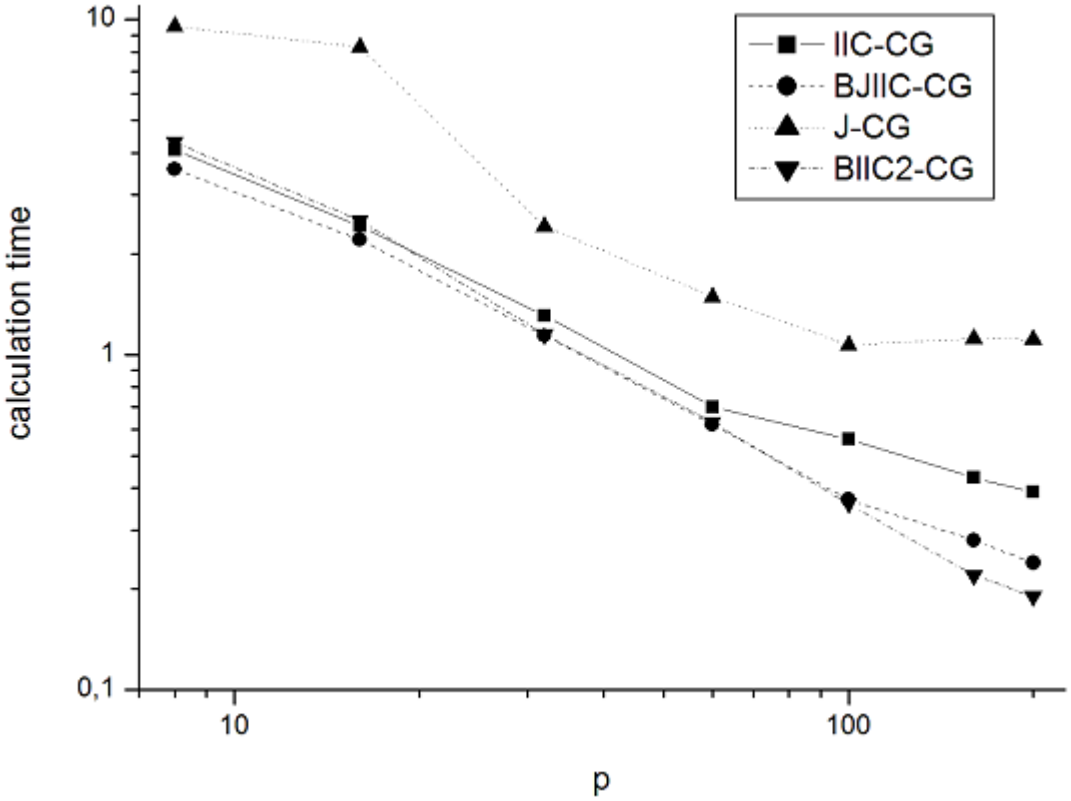

Рис.3. Время счета задачи с матрицей **hood** различными методами

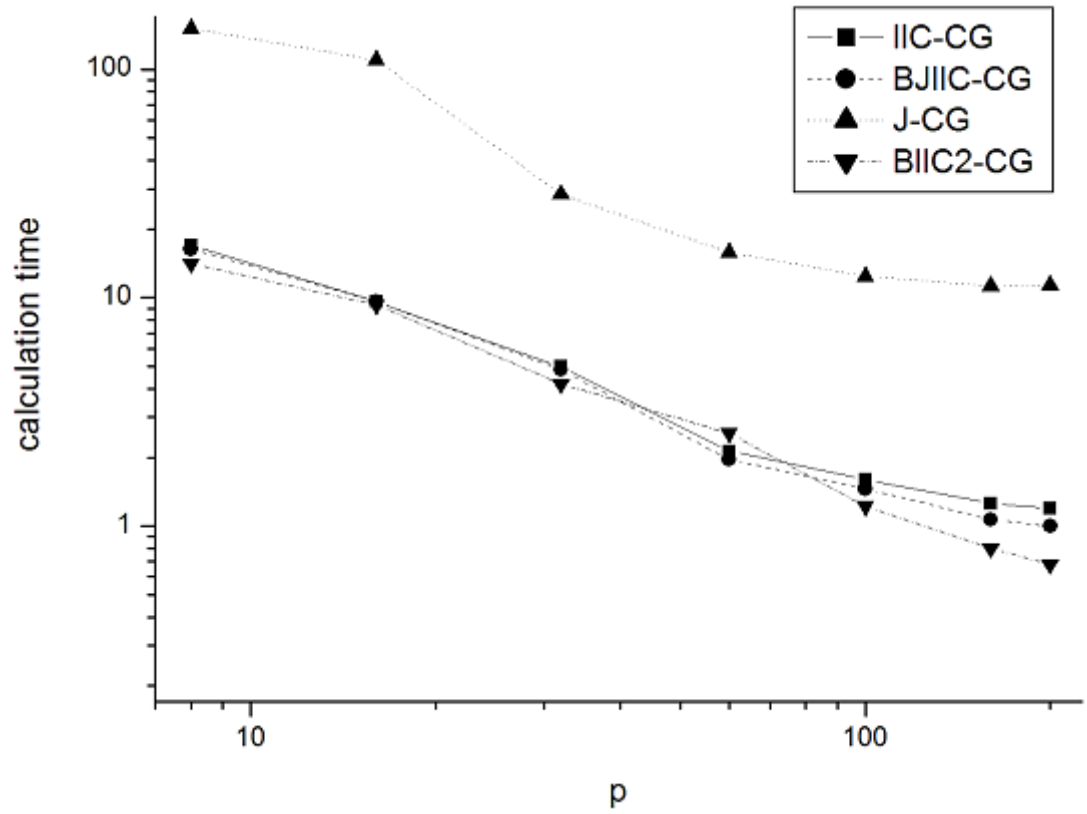

Рис.4. Время счета задачи с матрицей **pwtk** различными методами

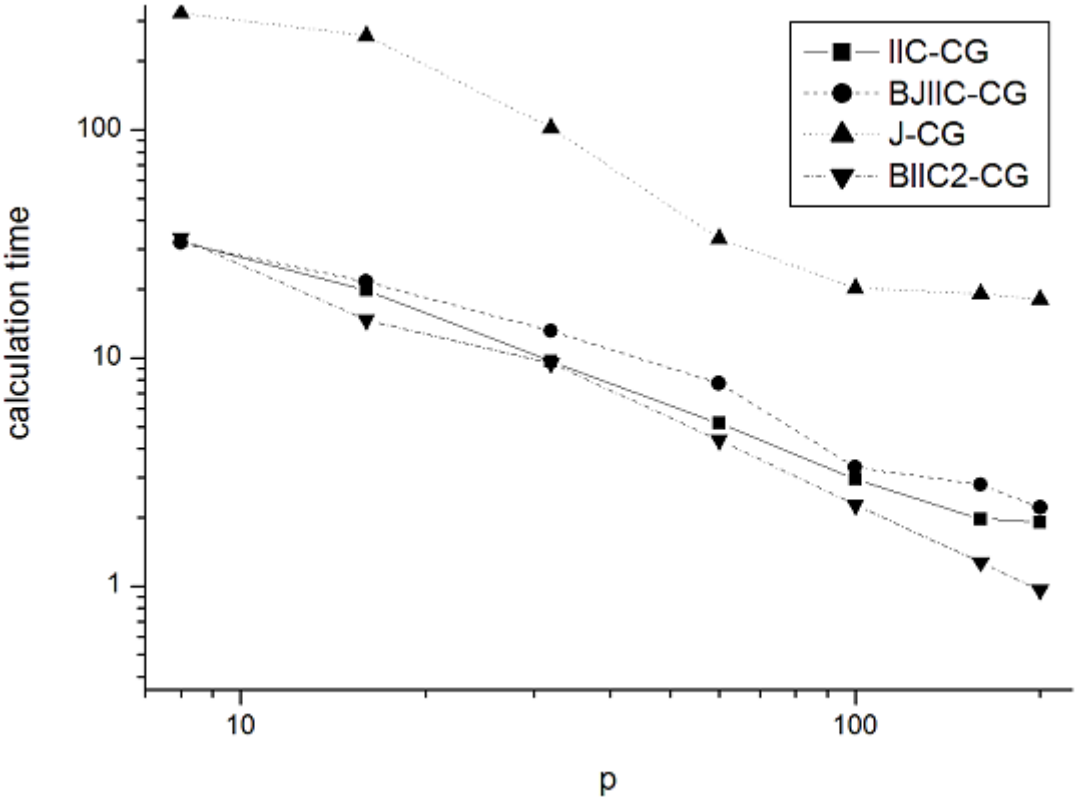

Рис.5. Время счета задачи с матрицей **msdoor** различными методами

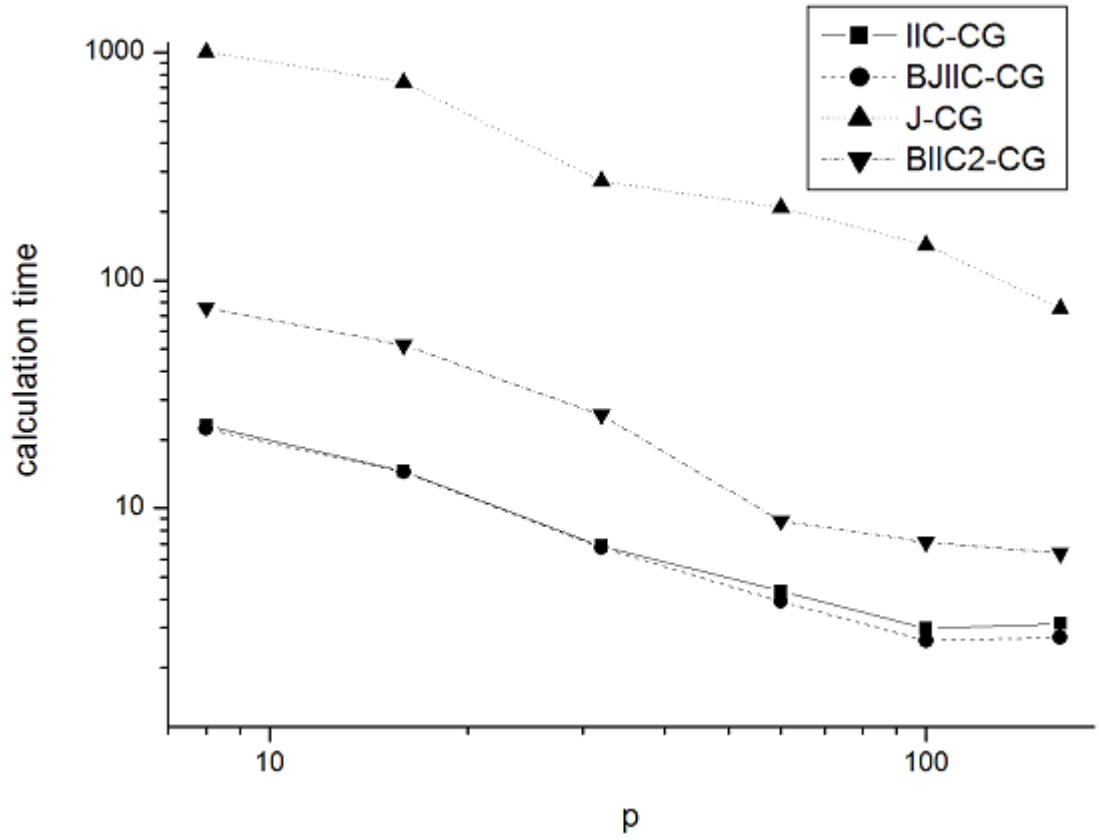

Рис.6. Время счета задачи с матрицей **x104** различными методами

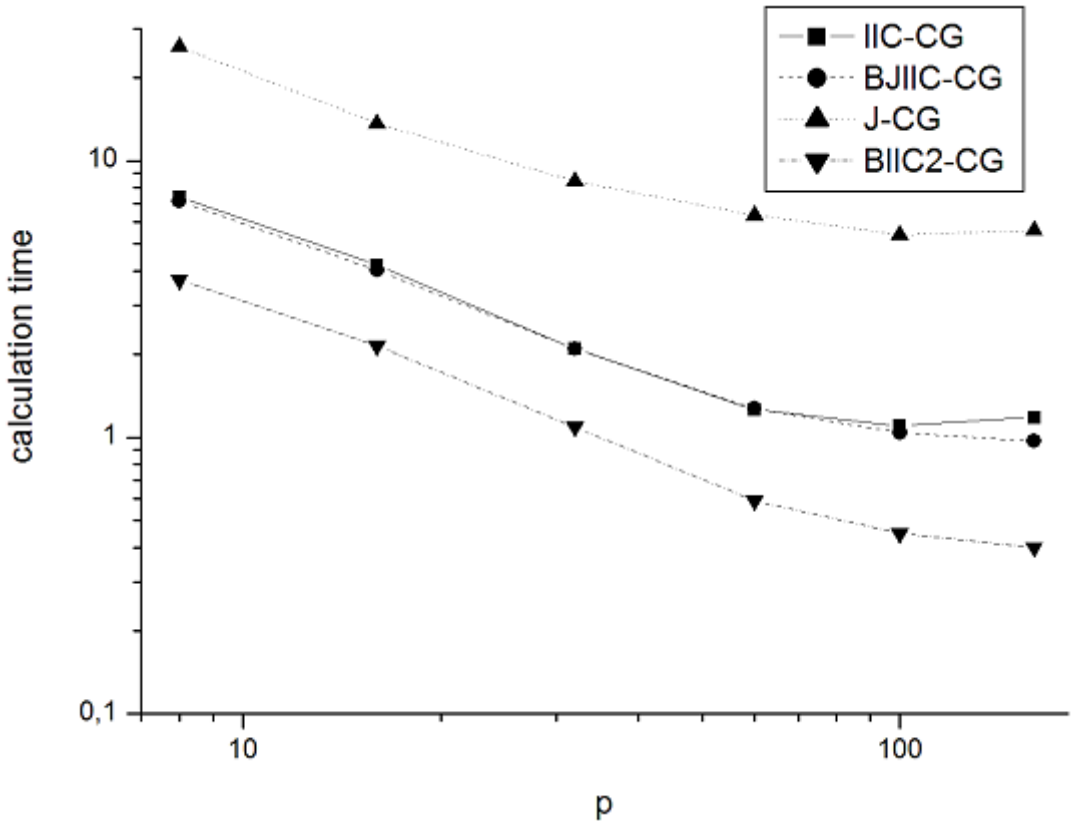

Рис.7. Время счета задачи с матрицей **s3dtkt3m2** различными методами

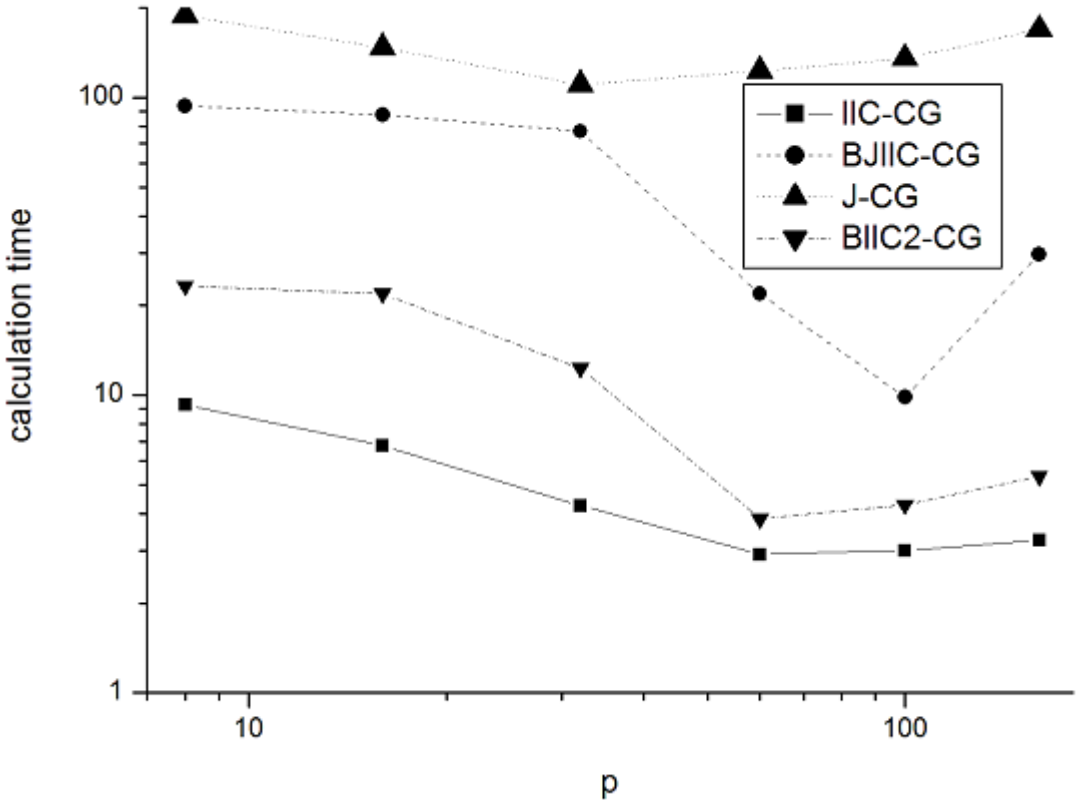

Рис.8. Время счета задачи с матрицей **thread** различными методами

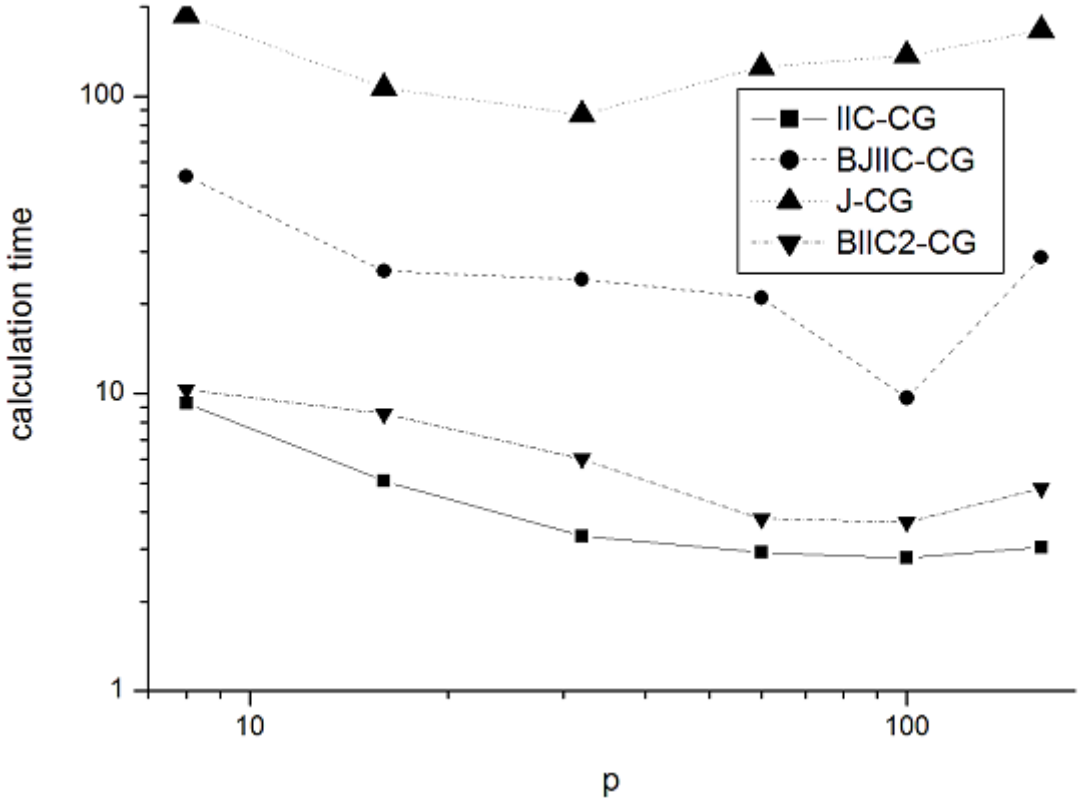

Рис.9. Время счета задачи с матрицей **thread** при использовании разбиения [10]

Из рис. 2-5 и табл. 2 видно, что времена решения соответствующих задач методами IIC-CG, BJIIC-CG, BIIC2-CG не очень сильно отличаются друг от друга и значительно меньше времени решения этих задач методом J-CG. Из рис.6-7 и табл. 2 видно, что времена решения соответствующих задач методами IIC-CG, BJIIC-CG, тоже не очень сильно отличаются друг от друга и значительно меньше времени решения этих задач методом J-CG. Однако, на рис. 6 время решения задачи с матрицей **x104** методами IIC-CG, BJIIC-CG существенно меньше времени ее решения методом BIIC2-CG. Время решения задачи с матрицей **s3dtkt3m2** методами IIC-CG, BJIIC-CG существенно больше времени ее решения методом BIIC2-CG.

При решении модельной задачи (см. рис.1) время решения ее методом J-CG было меньше, чем методами IIC-CG, BJIIC-CG, что скорее всего связано с сильной разреженностью матрицы.

На рисунках 8 и 9 приведены времена счета задачи с матрицей **thread** при использовании двух различных способов разбиения области расчета. В обоих случаях применение метода IIC-CG приводит к наименьшему времени счета, а использование метода J-CG к наибольшему времени счета этой задачи. Использование разбиения [10] позволяет решить задачу методом BJIIC-CG существенно быстрее, чем при использовании разбиения, описанного в [9] (на относительно небольшом числе процессоров). Это связано с большим числом итераций метода BJIIC-CG при использовании описанного выше упорядочения.

Итак, в настоящей работе рассмотрен метод IIC-CG, предложен параллельный алгоритм его реализации. Предложен метод предобусловливания блочного Якоби неполного обратного треугольного разложения BJIIC, не требующий обменов информацией между процессорами как при построении, так и при обращении предобусловливателя. Расчеты модельной задачи и ряда тестовых задач из коллекции университета Флориды с плотными матрицами показывают, что время счета задач методом IIC-CG значительно меньше, по сравнению с методом J-CG (за исключением простейшей модельной задачи). Время счета задач методом IIC-CG как правило (за исключением одной сложной задачи) близко к времени их счета методом BJIIC-CG. С точки зрения программной реализации метод BJIIC-CG проще, чем метод IIC- CG.

Список литературы

1. Davis T., Hu Y.F. University of Florida sparse matrix collection.//ACM Trans. on Math.~Software. 2011. V.38, N.1// http://www.cise.ufl.edu/research/sparse/matrices
2. George T., Gupta A., Sarin V.An experimental evaluation of iterative solvers for large SPD systems of linear equations: IBM Res. Report RC 24479. Jan.25. 2008.
3 Капорин И.Е. Предобусловленный метод сопряженных градиентов для решения дискретных аналогов дифференциальных задач //Дифференц. ур-ния. 1990. Т. 26. №7. - С.1225-1236.
4. Kaporin I.E.New convergence results and preconditioning strategies for the conjugate gradient method // Numer. Linear Algebra Appls. 1994. V.1. - P.179-210.
5. Kolotilina L.Yu., Yeremin A.Yu.Factorized sparse approximate inverse preconditionings. I.~Theory // SIAM J. Matrix Anal. Appl. 1993. V.14. - P.45-58.
6. Chow E.A priori sparsity patterns for parallel sparse approximate inverse preconditioners // SIAM J. Sci. Comput. 2000. V.21. N.5. - P.1804-1822.
7. Chow E.Parallel implementation and practical use of sparse approximate inverse preconditioners with a priori sparsity patterns // Internat. J. High Performance Comput. Appl. 2001. V.15. N.1. - P.56-74.
8. Kolotilina L.Yu., Nikishin A.A., Yeremin A.Yu.Factorized sparse approximate inverse preconditionings.IV.~Simple approaches to rising effciency // Numer. Linear Algebra Appl. 1999. V.6. - P.515-531.
9. Kaporin I.E. Reordering and splitting of sparse matrices into overlapping blocks for massively parallel preconditioning of iterative methods//Presented at NUMGRID-2012, A.A.Dorodnicyn Computing Center RAS, Moscow, Dec.17-18, 2012.
10. Капорин И.Е., Милюкова О.Ю. Массивно-параллельный алгоритм предобусловленного метода сопряженных градиентов для численного решения систем линейных алгебраических уравнений // Сб.трудов отдела проблем прикладной оптимизации ВЦ РАН (под ред.В.Г.Жадана). М.: Из-во ВЦ РАН. 2011. – С. 132-157.
11. Капорин И.Е., Коньшин И.Н. Параллельное решение симметричных положительно-определенных систем на основе перекрывающегося

разбиения на блоки // Ж. вычисл. матем. и матем. физики. 2001. Т. 41. № 4. - С. 515–528.
12. Axelsson O.Iterative solution methods. New York: Cambridge Univ. Press, 1994.
13. Капорин И.Е. Предобусловливание итерационных методов решении систем линейных алгебраических уравнений. Дисс. на соиск. ученой степени д.ф.-м.н. 2011. Москва. 216 С.
14.Капорин И.Е. Использование полиномов Чебышева и приближенного обратного треугольного разложения для предобусловливания метода сопряженных градиентов // Ж. вычисл. матем. и матем. физики. 2012. Т.52. № 2. – С.1-26.
15. Kaporin I. Fast matrix scaling, fine-coarse grid partitioning, and highly parallel preconditioning // Numerical geometry, grid generation, and high performance computing (Yu.G.Evtushenko, V.A.Garanzha, M.K.Kerimov, eds.), Procs. Int. Conf. NUMGRID2010, Moscow, 11-13 Oct. 2010, P. 110-116.
16. Farhat C. A simple and effcient automatic FEM domain decomposer // Computers and Structures. 1988. V. 28. N. 5. - P.579-602.
17. Diekmann R., Preis R., Schlimbach F., Walshaw C. Shape-optimized mesh partitioning and load balancing for parallel adaptive FEM // Parallel Computers and Structures. 1988. V.28. N.5. - P.579-602.
18. Якобовский М.В. Инкрементный алгоритм декомпозиции графов. // Вестник Нижегородского Университета им. Н.И.Лобачевского. Серия «Математическое моделирование и оптимальное управление». 2005. Вып. 1(28). Нижний Новгород: Издательство ННГУ. - С. 243-250.
19. Kaporin I.E. High quality preconditionings of a general symmetric positive definite matrix based on its $U^TU+U^TR+R^TU$ - decomposition // Numer. Lin. Alg. Appl. 1998. V. 5. – P.483-509.

Оглавление

1. Введение 3
2. Предобусловленный метод сопряженных градиентов 5
3. Выбор значений ненулевых элементов матрицы *G*, обеспечивающий построение *K*-оптимального предобусловливания 6
4. Выбор позиций ненулевых элементов матрицы *G* 6
5. Алгоритм параллельной реализации 7
6. Предобусловливание при помощи блочного метода Якоби в сочетании с неполным обратным треугольным разложением 8
7. Оценки уменьшения K-числа обусловленности в методах BJIIC и IIC 9
8. Переупорядочение и разбиение графа матрицы 10
9. Предобусловливание при помощи блочного неполного обратного треугольного разложения и предобусловливание Якоби 14
10. Результаты расчетов 15

Список литературы 26