УДК 519.688

ПАРАЛЛЕЛЬНЫЙ ДПФ-АЛГОРИТМ ВЫЧИСЛЕНИЯ ПРИСОЕДИНЕННОЙ МАТРИЦЫ В КОЛЬЦЕ ПОЛИНОМОВ МНОГИХ ПЕРЕМЕННЫХ С ЦЕЛЫМИ КОЭФФИЦИЕНТАМИ

© **А. О. Лапаев**

Ключевые слова: параллельная компьютерная алгебра, дискретное преобразование Фурье, быстрое преобразование Фурье, полиномиальные алгоритмы, вычисление присоединенной матрицы, параллельные матричные алгоритмы.

Предлагается параллельный алгоритм вычисления присоединенной матрицы над кольцом полиномов многих переменных с целыми коэффициентами, использующий дискретное преобразование Фурье над простым полем.

1 Введение

Одной из задач параллельной компьютерной алгебры является разработка эффективных матричных алгоритмов, в частности для матриц, элементами которых являются полиномы многих переменных.

Классическим методом ускорения вычислений является использование Китайской теоремы об остатках (КТО) [1] для числовых и полиномиальных модулей: задача решается в гомоморфных образах над конечным полем и затем результат интерполируется в кольцо полиномов с использованием схем Ньютона или Лагранжа [1]. Такой подход позволяет избежать сильного роста коэффициентов и степеней полиномов при вычислениях.

В данной статье предлагается вместо КТО для полиномиальных модулей использовать дискретное преобразование Фурье [2]. Преимуществом данного подхода является восстановление результата по полиномиальным модулям за меньшее количество операций. Например, в случае использования m простых полиномиальных модулей сложность восстановления по КТО будет оцениваться как $\Theta(m^2)$, а в случае использования ДПФ – $\Theta(m \log_2 m)$.

В параграфе 2 приводится алгоритм вычисления присоединенной матрицы над кольцом $\mathbb{Z}[x_1, \ldots, x_n]$, использующий дискретное преобразование Фурье (ДПФ) в конечном поле, и вычисляются выражения для сложности приведенного алгоритма. Кроме того, проводится сравнение теоретических оценок для алгоритмов, использующих ДПФ [1, 3, 4] и использующих КТО.

В параграфе 3 рассматривается распараллеливание алгоритма вычисления присоединенной матрицы.

В параграфе 4 приводятся результаты экспериментов с полученным параллельным алгоритмом вычисления присоединенной матрицы, который в конечном поле использует ДПФ.

2 Оценки сложности

Алгоритм для вычисления присоединенной матрицы с использованием ДПФ и его оценки сложности для случая полиномов одной переменной были получены в работе [5]. Теперь рассмотрим случай полиномов многих переменных.

Пусть $A = (a_{ij}(x)) \in \mathbb{Z}^{n\times n}[x_1, \ldots, x_d]$,

$$a_{ij}(x_1, x_2, \ldots, x_d) = \sum_{k_1=0}^{s_{ij}^1-1} \sum_{k_2=0}^{s_{ij}^2-1} \cdots \sum_{k_d=0}^{s_{ij}^d-1} a_{ij}^{k_1,k_2,\ldots,k_d} x_1^{k_1} x_2^{k_2} x_d^{k_d}.$$

Пусть $\max\limits_{i,j,k_1,k_2,\ldots,k_d} |a_{ij}^{k_1,k_2,\ldots,k_d}| \leqslant \alpha$ и $\deg_u a_{ij}(x_1, x_2, \ldots, x_d) < S_u, u = 1, \ldots, d$, где $\deg_u$ – степень полинома по переменной x_u. Обозначим $s = S_1 S_2 \ldots S_u$.

Максимальное количество мономов в элементах присоединенной матрицы A^* не будет превосходить $m = n^d s$. Количество точек для вычисления ДПФ по переменной x_k будет $N_k = 2^{\lceil \log_2(nS_k)\rceil}$ [4, 6, 7]. Обозначим $N = N_1 N_2 \ldots N_d$.

Найдем количество простых 32-битных числовых модулей. Будем использовать оценку абсолютной величины коэффициента определителя матрицы A.

Определитель матрицы A можно вычислить по формуле:

$$\det A = \sum_{(j_1,\ldots,j_n)} (-1)^t a_{1j_1} a_{2j_2} \ldots a_{nj_n}, \tag{1}$$

где $(j_1, \ldots, j_n)$ – перестановка чисел от 1 до n, t – четность этой перестановки.

Формула (1) содержит ровно $n!$ слагаемых. Используя формулу (1) и формулу Стирлинга для верхней оценки значения $n!$, получим верхнюю оценку для максимального числового коэффициента $\det A$. Тогда количество r простых 32-битных модулей равно

$$r = \lceil \log_h 2(\sqrt{2\pi n} \left(\frac{n}{e}\right)^n e^{\frac{1}{12n}} s^{n-1} \alpha^n)\rceil, \tag{2}$$

где $h = 2^{32}$.

Алгоритм вычисления присоединенной матрицы

1. Выбираются простые числовые модули $p_1, \ldots, p_r$.
2. Устанавливается номер простого модуля $t = 1$.
3. Для каждого элемента матрицы A вычисляется дискретное преобразование Фурье $F_{p_t}(A) = (F_{p_t}(a_{ij})) = ((\hat{a}_{ij}^{1,t}, \hat{a}_{ij}^{2,t}, \ldots, \hat{a}_{ij}^{N,t}))$ по простому модулю p_t [4]. Каждый элемент такой матрицы – одномерный вектор длины $N = 2^{\log_2 m}$, где $\hat{a}_{ij}^{u,t}, u = 1, \ldots, N$ являются элементами простого поля $\mathbb{Z}_{p_t}$.
4. Для матриц $F_t(A^{(u)}) = (a_{ij}^{u,t}), u = 1, \ldots, N$ вычисляются присоединенные матрицы $F_t(A^{*(u)}) = (a_{ij}^{*u,t})$.
5. Составляется матрица $F_t(A^*) = ((a_{ij}^{*1,t}, a_{ij}^{*2,t}, \ldots, a_{ij}^{*N,t}))$. Для ее элементов вычисляется обратное преобразование Фурье $A_t^* = F^{-1}(F_t(A^*))$.
6. Увеличиваем t на 1. Если $t \leqslant r$, то переходим к шагу 3.
7. По образам $A_1^*, A_2^*, \ldots, A_r^*$ в кольцах $\mathbb{Z}_{p_1}[x], \mathbb{Z}_{p_2}[x], \ldots, \mathbb{Z}_{p_r}[x]$ вычисляется матрица A^*, используя КТО [3] для числовых модулей $p_1, p_2, \ldots, p_r$.

Получим выражения сложности алгоритма.

Количество операций над словами при вычислении дискретного преобразования Фурье по алгоритму быстрого преобразования Фурье для n^2 полиномов на N точках при использовании r числовых простых модулей равно

$$n^2 r(7s\lceil \log_h \alpha \rceil + 9N \log_2 N).$$

Каждая кольцевая операция над образами полиномов состоит из rN арифметических операций над словами и rN вычислений остатка от деления. Будем считать, что вычисление остатка от деления занимает столько же времени, как 7 сложений слов. Тогда количество арифметических операций над образами полиномов исходной матрицы A при использовании алгоритма прямого хода [8] равно

$$8n^3 rN.$$

Для восстановления полинома по его образам необходимо выполнить r обратных преобразований Фурье и восстановить $n^2 \cdot n^d s$ чисел. Тогда число операций для получения присоединенной матрицы по ДПФ-образу ее элементов будет

$$n^2(9rN \log_2 N + 2r^2 s n^d).$$

Таким образом, сложность алгоритма будет равна

$$n^2 r(7s\lceil \log_h \alpha \rceil + 9N \log_2 N) + 8n^3 rN + n^2(9rN \log_2 N + 2r^2 s n^d). \quad (3)$$

Также приведем выражение сложности классического метода, использующего КТО как для числовых, так и для полиномиальных модулей:

$$n^2 r(7s\lceil \log_h \alpha \rceil + n^d s^2) + 8n^3 r n^d s + n^2(2(n^d s)^2 r + 2r^2 s n^d). \quad (4)$$

Если $n^d s$ равно степени числа 2 и если пренебречь величиной $7\lceil \log_h \alpha \rceil n^{-d}$ по сравнению с n, то оценки (3) и (4) примут более простой вид (5) и (6):

$$n^{2+d} rs(18 \log_2(n^d s) + 8n + 2r), \quad (5)$$

$$n^{2+d} rs(s(2n^d + 1) + 8n + 2r). \quad (6)$$

Вычислим выражения (3) и (4) при некоторых значениях параметров α, s, n, d и представим результат в виде таблицы. Для полиномов одной переменной с целыми коэффициентами, занимающими 8 бит, получим следующую таблицу, у которой в левом столбце указан порядок матрицы (n), а в верхней строке – размеры исходных полиномов (s).

Таблица 1

Отношение числа операций алгоритмов, использующих КТО и ДПФ, при $d = 1$ и $\alpha = 256$.

$n \setminus s$	2	4	8	16	32	64
2	0,58	0,57	0,67	0,91	1,39	2,28
4	0,63	0,69	0,87	1,24	1,95	3,30
8	0,75	0,87	1,15	1,69	2,75	4,75
16	0,91	1,09	1,47	2,23	3,71	6,51
32	1,05	1,29	1,79	2,76	4,67	8,36
64	1,17	1,46	2,04	3,20	5,48	9,93
128	1,25	1,58	2,23	3,52	6,06	11,08

Табл. 1 делится побочной диагональю на две области: в верхнем треугольнике более эффективен алгоритм, использующий КТО для полиномиальных модулей, в нижнем – алгоритм, использующий ДПФ.

Приведем аналогичную таблицу для матриц, элементами которых являются полиномы двух переменных.

Из табл. 2 видно, что алгоритм, использующий ДПФ, является эффективным при вычислении присоединенной матрицы для матриц, элементами которых являются полиномы двух переменных при порядке матрицы $n > 4$.

Таблица 2

Отношение числа операций алгоритмов, использующих КТО и ДПФ, при $d = 2$ и $\alpha = 256$

n/s	2	4	8	16	32	64
2	0,54	0,62	0,84	1,27	2,08	3,58
4	0,82	1,16	1,82	3,06	5,37	9,69
8	1,63	2,65	4,54	8,13	14,93	27,79
16	3,65	6,41	11,73	21,81	41,25	78,20
32	8,39	15,56	29,45	56,33	108,40	209,35
64	18,99	36,35	70,29	136,68	266,59	520,87
128	41,69	81,24	159,12	312,52	614,74	1210,26

3 Параллельный алгоритм

Пусть имеется параллельная ЭВМ с t процессорами с номерами $q = 0, \ldots, t-1$. Пусть на процессоре с номером 0 имеется матрица $A \in \mathbb{Z}[x_1, \ldots, x_d]$. Приведем шаги параллельного алгоритма.

1. Процессор с номером 0 выполняет рассылку матрицы A на остальные процессоры.

2. Процессоры с номерами, отличными от 0, выполняют прием матрицы A.

3. Каждый процессор оценивает количество простых модулей r и максимальную степень каждой переменной x_i в элементах присоединенной матрицы A^*.

4. Для упрощения описания предположим, что r делится на t нацело. В случае, если это не так, первые $r \mod t$ процессоров вычисляют присоединенную матрицу по $\lfloor \frac{r}{t} \rfloor + 1$ модулям, а остальные – по $\lfloor \frac{r}{t} \rfloor$ модулям.

5. На каждом процессоре выполняется разбиение массива простых чисел на t частей согласно условию, описанному на предыдущем шаге.

6. На каждом процессоре выполняется вычисление ДПФ для элементов матрицы последовательно по каждому простому числовому модулю данного процессора. В случае, если $r < t$, оставшиеся процессоры используются для распараллеливания вычисления ДПФ.

7. Для полученной матрицы образов ДПФ вычисляется присоединенная A^*_{dft} по каждому простому числовому модулю на данном процессоре.

8. Для элементов матрицы A^*_{dft} вычисляется обратное преобразование Фурье по каждому простому числовому модулю на данном процессоре.

9. Полученные матрицы A^*_{dft} делятся на t частей, и каждый процессор посылает часть с номером q на процессор с номером q.

10. Каждый процессор восстанавливает коэффициенты своей части полиномов по Китайской теореме об остатках.

11. Каждый процессор посылает свою часть восстановленных на шаге 10 данных процессору с номером 0.

12. На процессоре 0 выполняется объединение полученных от других процессоров частей присоединенной к A матрицы.

Приведем псевдокод данного алгоритма.

```
//Данный код выполняется на всех процессорах кластера.
matrixSend(A); //Рассылка матрицы A на все процессоры.
/*Вычисление количества простых модулей и максимальной степени по каждой переменной.*/
getBounds();
/*Инициализация структуры, в которой будут хранится промежуточные данные*/
pcs = new PolCoeffStore[primesLocalAmount[rank]];
/*В элементе массива primesLocalAmount с номером j хранится количество простых чисел для
процессора с номером j. Переменная rank хранит номер данного процессора.*/
for (int i = 0; i < primesLocalAmount[rank]; i++) {
//В массиве primes находятся простые модули.
int p = primes[i];
//Вычисляем ДПФ-образ матрицы.
matrFft = fftComputation(matr, p);
//Вычисляем присоединенную матрицу.
adjMatrFft = adjointComputation(matrFft);
//Вычисляем обратное ДПФ для присоединенной матрицы.
pcs[i] = fftInverseComputation(fftImRes, p);
}
/*Выполняется восстановление результата по КТО и его сборка на процессоре 0.*/
buildResult();
//Процессор с номером 0 возвращает результат.
if (rank == 0) return result; else return null;
```

4 Эксперименты

Рассмотренный алгоритм был программно реализован. С полученными программами были проведены эксперименты на кластере МВС100К МСЦ РАН. Результаты экспериментов представлены в табл. 3. В таблице приведено время вычислений и эффективность, которая вычислялась по формуле:

$$q_{m,k} = \frac{\frac{t_m}{t_k} - 1}{\frac{k}{m} - 1}.$$

Здесь t_m, t_k – время вычислений на m и k процессорах соответственно.

Из табл. 3 видно, что алгоритм обладает хорошей масштабируемостью, т. к. время вычислений уменьшается при увеличении количества процессоров. Например, при при увеличении количества процессоров с 2 до 16 для матриц порядка $n = 16$ время уменьшилось в 6 раз.

Таблица 3

Время вычисления присоединенной матрицы с использованием k процессоров и эффективность алгоритма для матриц порядка n с полиномами двух переменных степени 4 по каждой с 64-битными коэффициентами

k (процессоры)	2	4	8	16
n (порядок матрицы)	Время,с	Время,с / $q_{2,4}$	Время,с / $q_{2,8}$	Время,с / $q_{2,16}$
4	3	2/50%	2/17%	-
8	38	18/111%	11/82%	-
16	1457	661/120%	362/101%	230/76%

5 Заключение

В статье описан алгоритм параллельного вычисления присоединенной матрицы в кольце $\mathbb{Z}[x_1, \ldots, x_d]$. Приведены выражения сложности данного алгоритма для случая, когда элементами матрицы являются полиномы многих переменных. Из данных оценок следует, что использование ДПФ имеет преимущество при восстановлении результата. Получена схема распараллеливания данного алгоритма для суперЭВМ с распределенной памятью. Проведены эксперименты на кластере МСЦ РАН, которые показали, что при увеличении количества процессоров в 8 раз время уменьшается приблизительно в 6 раз, что говорит о хорошей масштабируемости рассмотренной схемы распараллеливания.

Следующим этапом исследований является построение и реализация параллельных ДПФ-алгоритмов для умножения матриц, вычисления характеристического полинома и определителя.

ЛИТЕРАТУРА

1. *Кнут Д.Э.* Искусство программирования. Т.2. Получисленные алгоритмы, 3-е изд. М.: Издательский дом «Вильямс», 2001.

2. *Лапаев А.О.* Вычислительные алгоритмы для полиномиальных матриц с использованием ДПФ // International conference Polynomial Computer Algebra. St.Petersburg, PDMI RAS, 2009. С. 141-146.

3. *Ноден П., Китте К.* Алгебраическая алгоритмика (с упражнениями и решениями)/ Пер. с франц. М. Мир, 1999.

4. *Лапаев А.О.* О вычислении многомерного дискретного преобразования Фурье в простом поле // Вестник Тамбовского университета. Сер. Естественные и технические науки. Том 14. Вып. 4. 2009. С. 729-731.

5. *Malaschonok G. I., Valeev Yu. D., Lapaev A. O.* On the choice of multiplication algorithm for polynomials and polynomial matrices // Zapiski POMI. 2009. V. 373. P. 157-188.

6. *Кормен, Томас Х., Лейзерсон, Чарльз И., Ривест, Рональд Л., Штойн, Клиффорд.* Алгоритмы: построение и анализ, 2-е издание/ Пер. с англ. М.: Издательский дом «Вильямс», 2005. С. 1296.

7. *Лапаев А.О.* Параллельное вычисление дискретного преобразования Фурье полинома в простом поле. Материалы 9-й международной конференции-семинара «Высокопроизводительные параллельные вычисления на кластерных системах». Владимир, 2009. С. 272-273.

8. *Malaschonok G.I.* Algorithms for computing determinants in commutative rings // Diskret. Mat. 1995. Vol. 7. No. 4. P. 68-76.

БЛАГОДАРНОСТИ: Работа выполнена при поддержке программы «Развитие потенциала высшей школы» (проект 2.1.1/10437).

Поступила в редакцию 12 ноября 2010 г.

UDK 519.688

PARALLEL DFT-ALGORITHM FOR COMPUTING OF ADJOINT MATRIX IN THE RING OF INTEGRAL POLYNOMIALS WITH MANY VARIABLES

© **A. O. Lapaev**

Key words: parallel computer algebra, discrete Fourier transform, fast Fourier transform, polynomial algorithms, computation of adjoint matrix, parallel matrix algorithm.

Parallel algorithm for computing of adjoint matrix over $\mathbb{Z}[x_1, \ldots, x_n]$ in terms of discrete Fourier transform over finite field is described.