УДК 681.3.06: 681.323(519.6)

ПАРАЛЛЕЛЬНАЯ ДВУМЕРНАЯ УПАКОВКА НА ОСНОВЕ ВИДОИЗМЕНЕНИЯ ФОРМУЛ ВИЕТА И СОРТИРОВКИ

Ромм Я.Е., Назарьянц Е.Г.

Таганрогский институт имени А.П. Чехова (филиал),
ФГБОУ ВО «Ростовский государственный экономический университет (РИНХ)»,
Таганрог, e-mail: romm@list.ru, Elena2781@yandex.ru

Полиномиальная временная сложность двумерной упаковки n прямоугольников в положительном квадранте с минимальной площадью окаймляющего прямоугольника достигается посредством детерминированного параллельного алгоритма на основе видоизменения формул Виета и устойчивой адресной сортировки. При любом $m < n$ алгоритм содержит $\log_m n$ последовательных шагов. Глобально оптимальная упаковка получается в случае $m = n$. Если $m \neq n$, локально оптимальная упаковка достигается в каждом отдельно взятом сочетании прямоугольников на выходе шага параллельного алгоритма. Временная сложность оценивается на модели неветвящихся параллельных программ без учёта обмена, для глобально оптимальной упаковки оценка имеет вид $T(R) = O(n)$, где количество процессоров $R \leq (2^n - 2)n^2$. Даны разновидности оценок алгоритмов с локальной оптимизацией, одна из которых представлена соотношением $T(R) \leq 2q\frac{m}{m-1}\cdot(n+\log_m n+1)\tau$, $q = \text{const}$, где $R \leq 2^{m-1}\cdot\left(C_n^m m\right)^2$. Работа включает конструктивное описание алгоритмов параллельной сортировки, параллельной генерации сочетаний прямоугольников из n по m и параллельного алгоритма двумерной упаковки на их основе.

Ключевые слова: детерминированный параллельный алгоритм двумерной упаковки, полиномиальная временная сложность, видоизменение формул Виета, параллельная комбинаторная оптимизация

PARALLEL TWO-DIMENSIONAL PACKAGING BASED MODIFICATION OF VIETA'S FORMULAS AND SORTING

Romm Ya.E., Nazaryants E.G.

Taganrog Institute named A.P. Chekhov (branch),
Rostov State University of Economics (RINH), Taganrog,
e-mail: romm@list.ru, Elena2781@yandex.ru

Polynomial time complexity of two-dimensional packing n of rectangles in the positive quadrant with minimum area of the bordering rectangle is achieved by means of deterministic parallel algorithm based on modification of Vieta's formulas and sustainable sorting of address. In any $m < n$ algorithm contains $\log_m n$ of successive steps. Globally optimal packing is obtained in the case of $m = n$. If $m \neq n$, locally an optimal packing is achieved in each individual combination of rectangles in the outlet of parallel algorithm's step. Time complexity is evaluated on a model of non-branching parallel programs excluding exchange, for the globally optimal packaging estimate has the form $T(R) = O(n)$, where the number of processors $R \leq (2^n - 2)n^2$. There are given the variety of algorithms evaluations with local optimization, one of them is represented by the relation of $T(R) \leq 2q\frac{m}{m-1}\cdot(n+\log_m n+1)\tau$, $q = \text{const}$, where $R \leq 2^{m-1}\cdot\left(C_n^m m\right)^2$. The work includes constructive description of parallel sorting algorithms, parallel generation of combinations of rectangles from n till m and parallel algorithm of the two-dimensional packaging based on them.

Keywords: deterministic parallel algorithm of the two-dimensional packaging, polynomial time complexity, modification of Vieta's formulas, parallel combinatorial optimization

Рассматривается задача упаковки прямоугольников в квадрант *(2DArea Packing Problem, 2DAP)*) [9]: дано n прямоугольных предметов с размерами $w_i \times h_i$, $i = 1, 2, ..., n$, требуется найти упаковку в положительном квадранте с минимальной площадью окаймляющего прямоугольника. При этом повороты запрещены, стороны прямоугольников параллельны осям координат. Формальная запись задачи: n – количество прямоугольников; w_i, h_i – длина и ширина i-го прямоугольника, $i = 1, 2, ...$; W, H – длина и ширина окаймляющего прямоугольника; $W \times H \to \min$. Цель работы – синтез и анализ детерминированного параллельного алгоритма двумерной упаковки. Алгоритм строится на основе устойчивой параллельной сортировки с единичной временной сложностью [5] и видоизменения формул Виета восстановления коэффициентов полинома по его комплексным корням [2, 3], используемого для генерации всех сочетаний прямоугольников из заданного конечного множества.

Видоизменение формул Виета
для параллельной генерации всех сочетаний прямоугольников

Коэффициенты полинома

$$P_n(x) = d_n x^n + d_{n-1}x^{n-1} + d_{n-2}x^{n-2} + \ldots + d_1 x + d_0$$

выражаются через его корни над полем комплексных чисел в виде [2, 3]:

$$\begin{pmatrix} d_n \\ d_{n-1} \\ \vdots \\ d_0 \end{pmatrix} = \underbrace{\left(\begin{array}{cc} \left.\begin{array}{cc} 10 & \\ x_{n-1} & 1 \\ 0 & x_{n-1} \\ 0 & 0 \\ \ldots & \ldots \\ 00 & \\ 00 & \end{array}\right\} n+1 & \begin{array}{cc} 00 & \\ 00 & \\ \vdots & \vdots \\ 00 & \\ 10 & \\ x_{n-1} & 1 \\ \ldots\ 0 & x_{n-1} \end{array} \end{array}\right)}_{n} \times$$

$$\times \underbrace{\left(\begin{array}{cc} \left.\begin{array}{cc} 10 & \\ x_{n-2} & 1 \\ 0 & x_{n-2} \\ 00 & \\ \ldots & \ldots \\ 00 & \\ 00 & \end{array}\right\} n & \begin{array}{cc} 00 & \\ 00 & \\ \vdots & \vdots \\ 00 & \\ 10 & \\ x_{n-2} & 1 \\ \ldots\ 0 & x_{n-2} \end{array} \end{array}\right)}_{n-1} \times \cdots \times \begin{pmatrix} 100 & & \\ x_2 & 10 & \\ 0 & x_2 & 1 \\ 00 & & x_2 \end{pmatrix} \times \begin{pmatrix} 10 & \\ x_1 & 1 \\ 0 & x_1 \end{pmatrix} \times \begin{pmatrix} 1 \\ x_0 \end{pmatrix}, \quad (1)$$

где x_i – i-й корень полинома, $i = 0, 1, ..., n-1$, предполагается, что $d_n = 1$. С другой стороны, по формулам Виета [1], –

$$\left.\begin{array}{l} d_n = 1, \\ d_{n-1} = -(x_0 + x_1 + x_2 + ... + x_{n-1}), \\ d_{n-2} = (x_0 \cdot x_1) + (x_0 \cdot x_2) + ... + (x_0 \cdot x_{n-1}) + ... + (x_{n-2} \cdot x_{n-1}), \\ d_{n-3} = -(x_0 \cdot x_1 \cdot x_2 + x_0 \cdot x_1 \cdot x_3 + ... + x_{n-3} \cdot x_{n-2} \cdot x_{n-1}), \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ d_{n-l} = (-1)^{n-l} \cdot (x_0 \cdot x_1 \cdot x_2 \cdot x_3 ... \cdot x_{l-1} + ... + x_{n-l-1} \cdot ... \cdot x_{n-1}), \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ d_0 = (-1)^n \cdot (x_0 \cdot x_1 \cdot x_2 \cdot ... \cdot x_{n-2} \cdot x_{n-1}). \end{array}\right\} \quad (2)$$

В (2), если не учитывать знак перед скобками, представлены все сочетания корней. Чтобы их выделить, требуется заменить знак умножения на знак «⊕», который понимается как перечисление элементов сочетания (без выполнения каких бы то ни было операций), знак суммы – на любой знак, обозначающий задание порядка сочетаний, в таком качестве ниже выбран знак «∨». Прямоугольники можно интерпретировать как взаимно однозначно сопоставленные комплексным корням полинома $x_j + Iy_j$ в виде $x_j \oplus y_j$, где x_j – длина основания, y_j – высота j-го прямоугольника, $j = 0, 1, ..., n-1$. В данной трактовке (2) означает представление всех сочетаний из n прямоугольников, положение которых в декартовых координатах задано и не меняется. Поскольку (1) алгоритмически эквивалентно (2), то описанное преобразование (1) генерирует все сочетания прямоугольников из n по m в виде

$$\begin{pmatrix} d_n \\ d_{n-1} \\ \vdots \\ d_0 \end{pmatrix} \Leftrightarrow \left(\underbrace{\begin{array}{cc} \left.\begin{array}{cc} 1 & 0 \\ x_{n-1} \oplus y_{n-1} & 1 \\ 0 & x_{n-1} \oplus y_{n-1} \\ 0 & 0 \\ \dots & \dots \\ 0 & 0 \\ 0 & 0 \end{array}\right\} n+1 & \begin{array}{cc} 0 & 0 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ 1 & 0 \\ x_{n-1} \oplus y_{n-1} & 1 \\ \dots \; 0 & x_{n-1} \oplus y_{n-1} \end{array} \end{array}}_{n} \right) \otimes$$

$$\otimes \left(\underbrace{\begin{array}{cc} \left.\begin{array}{cc} 1 & 0 \\ x_{n-2} \oplus y_{n-2} & 1 \\ 0 & x_{n-2} \oplus y_{n-2} \\ 0 & 0 \\ \dots & \dots \\ 0 & 0 \\ 0 & 0 \end{array}\right\} n & \begin{array}{cc} 0 & 0 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ 1 & 0 \\ x_{n-2} \oplus y_{n-2} & 1 \\ \dots \; 0 & x_{n-2} \oplus y_{n-2} \end{array} \end{array}}_{n-1} \right) \otimes \dots \otimes \begin{pmatrix} 1 & 0 \\ x_1 \oplus y_1 & 1 \\ 0 & x_1 \oplus y_1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ x_0 \oplus y_0 \end{pmatrix}. \quad (3)$$

В силу совпадения левых частей (1), (2) результатом выполнения (3) окажется теоретико-множественный аналог соотношения (2), который можно представить в виде

$$\left.\begin{array}{l} d_n \mapsto 1; \\ d_{n-1} \mapsto \big((x_0 \oplus y_0) \vee (x_1 \oplus y_1) \vee (x_2 \oplus y_2) \vee \dots \vee (x_{n-1} \oplus y_{n-1})\big); \\ d_{n-2} \mapsto \big((x_0 \oplus y_0) \otimes (x_1 \oplus y_1)\big) \vee \big((x_0 \oplus y_0) \otimes (x_2 \oplus y_2)\big) \vee \dots \vee \big((x_0 \oplus y_0) \otimes (x_{n-1} \oplus y_{n-1})\big) \vee \dots \\ \qquad\qquad \dots \vee \big((x_{n-2} \oplus y_{n-2}) \otimes (x_{n-1} \oplus y_{n-1})\big); \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ d_{n-m} \mapsto \big((x_0 \oplus y_0) \otimes (x_1 \oplus y_1) \otimes (x_2 \oplus y_2) \dots \otimes (x_{m-1} \oplus y_{m-1}) \vee \dots \vee (x_{n-m-1} \oplus y_{n-m-1}) \otimes \dots \otimes (x_{n-1} \oplus y_{n-1})\big); \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ d_0 \mapsto \big((x_0 \oplus y_0) \otimes (x_1 \oplus y_1) \otimes (x_2 \oplus y_2) \otimes \dots \otimes (x_{n-2} \oplus y_{n-2}) \otimes (x_{n-1} \oplus y_{n-1})\big). \end{array}\right\} \quad (4)$$

Соотношение (3) параллельно по строкам за n шагов генерирует все сочетания вида (4), которые не зависят от порядка прямоугольников, – из n по m при всех m. Все сочетания из n по m при конкретном m = const соответствуют второй снизу строке (3). Одномерный аналог алгоритма дан в [6], детальное описание двумерного алгоритма – в [7].

Применение параллельной сортировки для упорядочения сочетаний и выбора оптимальной упаковки

Используется устойчивая [4] сортировка подсчетом одномерного массива (ниже – сортировка) на основе матрицы сравнений (МС) [3, 5]. Пусть по отношению «≤» требуется упорядочить элементы массива $\{ c_l \}_{l=1}^{n}$, $n < \infty$. МС принимает вид [5]:

	c_1	c_2	...	c_n
c_1	α_{11}	α_{12}	...	α_{1n}
c_2	α_{21}	α_{22}	...	α_{2n}
...	...	...	...	...
c_n	α_{n1}	α_{n2}	...	α_{nn}

;

$$\alpha_{ij} = \begin{cases} 1, & c_i < c_j; \\ 0, & c_i = c_j; \\ -1, & c_j < c_i; \end{cases} \quad i = 1, 2, ..., n; j = 1, 2, ..., n. \qquad (5)$$

Значения α_{ij} из (5) удобно заменить символами «+», «0», «–», например, $C = (6{,}1; 2{,}2; 7{,}6; 2{,}2; 7{,}3)$,

	6,1	2,2	7,6	2,2	7,3
6,1	0	–	+	–	+
2,2	+	0	+	0	+
7,6	–	–	0	–	–
2,2	+	0	+	0	+
7,3	–	–	+	–	0

Чтобы определить номер в отсортированном массиве j-го элемента входного массива, достаточно подсчитать число нулей и плюсов в j-м столбце над диагональю, включая диагональный элемент, и сложить это число с числом плюсов j-го столбца ниже диагонали. В обозначении отсортированного массива $c1$ получится: $c1[3] = c[1]$; $c1[1] = c[2]$; $c1[5] = c[3]$; $c1[2] = c[4]$; $c1[4] = c[5]$. Все сравнения в (5) взаимно независимы, инвертируют знак относительно диагонали, сортировка максимально параллельна с временной сложностью $T\left(N^2/2\right) = O(1)$ [3, 5]. Она сохраняет порядок равных элементов, является адресной с обратной адресацией на основе явного задания [3, 5] взаимно однозначного соответствия входных и выходных индексов. Сортировка будет применяться для максимально параллельного упорядочения элементов сочетаний и выбора оптимальной упаковки за минимальное время.

Описание алгоритма параллельной двумерной упаковки

Вначале **алгоритм** излагается на уровне содержательного описания.

Этап I. Задание веса элементов

I.1. Все прямоугольники фиксируются в последовательности, определяемой условием задачи, произвольно задается $m = \text{const}$, $1 < m < n$.

I.2. Каждому прямоугольнику взаимно однозначно сопоставляется свой целочисленный вес. Конкретно, входные прямоугольники в порядке расположения нумеруются от единицы, i-му прямоугольнику присваивается вес i, $i = 1, 2, ..., n$. Выбираются первые

$$\tilde{n} = m\left[\frac{n}{m}\right].$$

На данном этапе

$$n - m\left[\frac{n}{m}\right] = n - \tilde{n}$$

оставшихся номеров не будут подаваться на вход алгоритма генерации сочетаний, они группируются в отдельное сочетание (в дальнейшем – остаточное) и обрабатываются отдельно.

Этап II. Генерация и упаковка сочетаний

II.1. На основе (3) генерируются все сочетания из $\tilde{n}$ по m. Остаточное сочетание рассматривается отдельно. Структура (3) сохраняется с заменой n на $\tilde{n}$, обуславливая обозначения: $d_{\tilde{n}-1}$ – все сочетания из $\tilde{n}$ по 1, $d_{\tilde{n}-2}$ – все сочетания из $\tilde{n}$ по 2,..., $d_{\tilde{0}}$ – сочетание из $\tilde{n}$ по $\tilde{n}$, $x_j \oplus y_j$ – размеры j-го прямоугольника, $j = 0, 1, \ldots, \tilde{n} - 1$.

Матрицы из (3) имеют следующие особенности. Последний множитель справа – столбец, у которого верхний элемент равен единице, нижний – первый выбранный прямоугольник $x_0 \oplus y_0$. Матрицы в (3) прямоугольные: число строк на единицу больше, чем число столбцов. Количество матриц соответствует количеству прямоугольников, на каждый прямоугольник приходится одна матрица.

Сочетания генерируются в соответствии последовательному умножению алгебраических матриц справа налево. Сначала преобразуются последняя и предшествующая ей матрица, в результате получится единственный столбец новой правосторонней матрицы с числом строк на единицу большим числа строк предшествующей правосторонней матрицы:

$$\begin{pmatrix} d_{\tilde{n}} \\ d_{\tilde{n}-1} \\ \vdots \\ d_{\tilde{0}} \end{pmatrix} \Leftrightarrow \left(\underbrace{\begin{array}{cc|ccc} 1 & 0 & 0 & 0 \\ x_{\tilde{n}-1} \oplus y_{\tilde{n}-1} & 1 & 0 & 0 \\ 0 & x_{\tilde{n}-1} \oplus y_{\tilde{n}-1} & \vdots & \vdots \\ 0 & 0 & 0 & 0 \\ \dots & \dots & 1 & 0 \\ 0 & 0 & x_{\tilde{n}-1} \oplus y_{\tilde{n}-1} & 1 \\ 0 & 0 & \dots\ 0 & x_{\tilde{n}-1} \oplus y_{\tilde{n}-1} \end{array}}_{\tilde{n}} \right\} \tilde{n}+1 \otimes$$

$$\otimes \left(\underbrace{\begin{array}{cc|ccc} 1 & 0 & 0 & 0 \\ x_{\tilde{n}-2} \oplus y_{\tilde{n}-2} & 1 & 0 & 0 \\ 0 & x_{\tilde{n}-2} \oplus y_{\tilde{n}-2} & \vdots & \vdots \\ 0 & 0 & 0 & 0 \\ \dots & \dots & 1 & 0 \\ 0 & 0 & x_{\tilde{n}-2} \oplus y_{\tilde{n}-2} & 1 \\ 0 & 0 & \dots\ 0 & x_{\tilde{n}-2} \oplus y_{\tilde{n}-2} \end{array}}_{\tilde{n}-1} \right\} \tilde{n} \otimes \dots \otimes \begin{pmatrix} 1 & 0 & 0 \\ x_2 \oplus y_2 & 1 & 0 \\ 0 & x_2 \oplus y_2 & 1 \\ 0 & 0 & x_2 \oplus y_2 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ x_0 \oplus y_0 \vee x_1 \oplus y_1 \\ (x_0 \oplus y_0) \otimes (x_1 \oplus y_1) \end{pmatrix}.$$

Полученная матрица из одного столбца и предшествующая ей слева матрица снова преобразуется соответственно алгебраическому умножению матриц. Получится матрица из одного столбца с числом строк на единицу большим матрицы-результата предыдущего шага. Преобразования выполняются до тех пор, пока не останется один результирующий столбец матрицы с числом строк, равным числу компонентов вектора в левой части (3).

II.2. В строке, сопоставленной $d_{\tilde{n}-j}$ (кратко – в строке $d_{\tilde{n}-j}$), располагаются все сочетания элементов-прямоугольников из $\tilde{n}$ по j, сочетания нумеруются в порядке расположения слева направо, начиная от единицы, число таких сочетаний есть $C_{\tilde{n}}^{j}$, $d_{\tilde{n}-j} \Leftrightarrow C_{\tilde{n}}^{j}$, $j = \overline{1,\tilde{n}}$. В строке $d_{\tilde{n}-j}$ множество всех элементов всех сочетаний образует аналог двумерного массива – по своему индексу-весу в последовательности и по индексу номера сочетания в последовательности 1,2, …,$C_{\tilde{n}}^{j}$. Строка $d_{\tilde{n}-j}$ фиксируется при $j = m$, остальные строки на время изложения алгоритма не рассматриваются.

II.3. **Выбор различных сочетаний из $\tilde{n}$ по m, не содержащих общих элементов**

II.3.1. У элементов строки $d_{\tilde{n}-m}$ как аналога двумерного массива первый «индекс» – вес элемента (от единицы до $\tilde{n}$) (фактически это не индекс, поскольку он может повторяться у элементов разных сочетаний, в сочетании это именно назначенный априори вес выбранного элемента). Второй индекс (он действительно индекс) – номер сочетания. Выполняется максимально параллельная сортировка всех $\tilde{n}$ элементов только по их весу: ключ, по которому выполняется сортировка, совпадает с весом элемента сочетания. Эта сортировка устойчива – сохраняет порядок равных элементов, поэтому все равные веса в отсортированном массиве расположатся друг за другом по неубыванию. Вторые индексы элементов переставятся вместе с элементами, так что элементы с равными весами, следуя подряд друг за другом, образуют вторыми индексами последовательность номеров (вообще говоря, неупорядоченную) различных сочетаний, в которые эти элементы входят.

II.3.2. Из последовательности, полученной в п. II.3.1, выбираются элементы, у которых первый «индекс» равен единице (вес $i = 1$). Каждому такому элементу сопоставляются элементы сочетания, в которое этот элемент входил, – в сочетание с ним объединяются элементы, имеющие одинаковый с ним второй индекс. Таким образом, данная операция восстанавливает сочетания, полученные априори.

Каждое выбранное сочетание оптимально упаковывается (синхронно и независимо друг от друга) каким-либо одним и тем же способом. После упаковки всех сочетаний с элементом единичного веса фиксируются окаймляющие прямоугольники каждого сочетания с минимальной вследствие упаковки площадью. После упаковки сохраняется второй (истинный) индекс сочетания. Все значения минимальных площадей сохраняются вместе со своими сочетаниями и параллельно сортируются по неубыванию с соответствующим смещением индекса сочетания. Получится неубывающая последовательность площадей окаймляющих сочетания прямоугольников, которые содержат элемент единичного веса.

Первый элемент отсортированной таким образом последовательности – наименьшая площадь, находящееся в нем упакованное сочетание идентифицируется по его второму индексу. Если у различных сочетаний, содержащих элемент единичного веса, площади окаймляющих прямоугольников равны, то согласно сортировке выбирается первое из этих сочетаний. Остальные сочетания с элементом единичного веса, соответствующие отсортированным площадям, удаляются из дальнейшего рассмотрения.

Для выбранного сочетания фиксируется его приоритетное местоположение. Учитывая единичный вес определившего сочетание элемента и второй его индекс, определивший все элементы сочетания, обозначим данное сочетание парой индексов $(1, i_1)$.

Способ исключения различных сочетаний с общими элементами заключается в следующем. Каждый из элементов сочетания $(1, i_1)$ попарно сравнивается по своему весу с элементами сочетаний соответственных отсортированной последовательности весов. Все сочетания с совпадающими по весу элементами, кроме $(1, i_1)$, удаляются из дальнейшего рассмотрения (в различных упакованных сочетаниях не должно быть одних и тех же прямоугольников). При этом адресация к каждому сочетанию и его идентификация при удалении выполняется по второму индексу элемента рассматриваемого веса.

Тем самым завершены все операции рассматриваемого способа с весом $i = 1$.

Далее, рассматривается линейка неудаленных весов. Все соответственные линейке отсортированных элементов сочетания обозначаются (k, i_k), $k \geq 2$. Рассмотрение выполняется для каждого веса $k = \text{const}$, ближайшего к рассмотренному и не удаленного на предыдущем шаге. Если неудаленных элементов с весом k несколько, то к линейке отсортированных элементов этого веса заново применяются все операции, описанные для $(1, i_1)$: предварительно выбираются все сочетания, содержащие вес k, и каждое такое сочетание отдельно упаковывается, выбирается сочетание с наименьшей окаймляющей площадью, оно фиксируется с приоритетом местоположения, а сравнение весов и удаление сочетаний будет теперь выполняться по отношению к сочетанию с элементом веса k. Аналогично, обрабатываются оставшиеся сочетания, элементы которых упорядочены по весу k, $k \geq 3$. В результате выполнения последовательности данных шагов останется последовательность упакованных сочетаний прямоугольников с индексами (k, i_k), $1 \leq k \leq \tilde{n}$, среди которых некоторые значения k могут отсутствовать. Остаточное сочетание из $n - \tilde{n}$ элементов подвергается упаковке синхронно с сочетаниями $(1, i_1)$.

На выходе данного шага оказываются в наличии n прямоугольников, так как на входе были все возможные сочетания из $\tilde{n}$ по m и остаточное сочетание из $n - \tilde{n}$, а удалению подвергались лишь те из них, которые содержали общие элементы, так что остался набор сочетаний, включающий все попарно различные прямоугольники. Эти прямоугольники упакованы в сочетания, включая остаточное, общее количество которых составит $\lceil n/m \rceil$.

II.3.3. На выходе предыдущего шага получилась последовательность упакованных сочетаний (k, i_k), $1 \leq k \leq n$, включая остаточное, и окаймляющие эти сочетания прямоугольники минимизированной площади. Именно эти окаймляющие прямоугольники вместе с входящими в них упакованными элементами на входе следующего шага интерпретируются теперь как новые элементы для описанной обработки: они заново получают веса от единицы и второй индекс по сочетанию, сочетания генерируются по тому же алгоритму на основе формулы (3). Иными словами, по отношению к новым элементам (окаймляющим прямоугольникам) применяются этап I и шаги на нем I.1, I.2, затем этап II и шаги на нем II.1, II.2, II.3.1, II.3.2, II.3.3. Количество таких новых элементов равно $[n/m]$.

Упаковка окаймляющих прямоугольников строится по числу не исключенных на предыдущем шаге входящих в них сочетаний. Таким образом, новые сочетания образуются из $[n/m]$ по m элементов. После установления единственности сочетаний и попарно различных в них новых элементов на выходе последнего шага рассматриваемого этапа останется количество упакованных сочетаний равное $\lceil \lceil n/m \rceil / m \rceil \sim n/m/m$.

Формализация алгоритма

К первому шагу алгоритма относится вся совокупность описанных выше операций над входными элементами – прямоугольниками. Выходные данные первого шага (оптимизированные окаймляющие прямоугольники) полагаются входными элементами-прямоугольниками для второго шага, и ко второму шагу относится повторение всех операций первого шага над поступившими входными элементами. Выходные элементы второго шага («иерархически» оптимизированные окаймляющие прямоугольники) полагаются входными элементами для третьего шага, на котором воспроизводится аналогичная их обработка, и т.д. Процесс воспроизводится до тех пор, пока результат от деления в соотношении $\lceil ...\lceil n/m\rceil/m/.../m\rceil$ остается большим 1, – пока не получится единственный заключительный элемент («иерархически» оптимизированный по площади окаймляющий прямоугольник). Количество шагов алгоритма не превзойдет $\sim\log_m n$.

Оценки временной сложности

Временная сложность, обозначаемая $T(R)$, где R – число процессоров, оценивается на модели неветвящихся параллельных программ без учёта обмена [8]. Поскольку $n \le m^{\lceil \log_m n\rceil}$, число шагов формализованного алгоритма $\tilde{K} \le \lceil \log_m n\rceil$. Пусть $\tilde{t}$ – временная сложность упаковки одного сочетания прямоугольников. С учетом максимального параллелизма используемой сортировки, числа шагов алгоритма и последовательности операций на шаге элементарным подсчетом получается оценка [7]:

$$T(R) \le q\sum_{i=1}^{\lceil \log_m n\rceil}\left(\frac{n}{m^{i-1}}+\frac{m}{m-1}\right)\cdot(1+\tilde{t});$$

$$q = \text{const}; \quad R \le \frac{\left(C_n^m m\right)^2}{2}.$$

Отсюда

$$T(R) \le q\frac{m}{m-1}\cdot(n+\log_m n+1)\cdot(1+\tilde{t});$$

$$q = \text{const}; \quad R \le \frac{\left(C_n^m m\right)^2}{2}, \qquad (6)$$

где не учитывается число процессоров для упаковки каждого сочетания из m прямоугольников. Таким образом, имеет место

Лемма 1. Двумерная упаковка на основе сортировки и видоизменения формул Виета может быть выполнена с помощью детерминированного параллельного алгоритма с временной сложностью (6).

Следствие 1. Глобально оптимальная упаковка получится при $m = n$.

При $m \ne n$ глобальный оптимум не является гарантированным, но наилучшая (локально оптимальная) упаковка достигается в каждом отдельном сочетании прямоугольников на каждом шаге.

Теорема 1. Если каждое сочетание из m прямоугольников упаковывается параллельным перебором всех 2^m вариантов, то сохраняются утверждения леммы 1 и следствия 1, при этом временная сложность выполнения всего алгоритма составит

$$T(R) \le 2q\frac{m}{m-1}\cdot(n+\log_m n+1)\tau;$$

$$q = \text{const}; \quad R \le 2^{m-1}\cdot\left(C_n^m m\right)^2, \qquad (7)$$

где τ – время бинарного сравнения, в оценке R учитывается число процессоров для максимально параллельной упаковки каждого рассматриваемого сочетания.

Следствие 2. Выполнение данного алгоритма параллельно по всем $m < n$ даст выбор наилучшего по m из локально оптимальных вариантов упаковки с временной сложностью

$$T(R) \le 2q\frac{m}{m-1}\cdot(n+\log_m n+1)\tau;$$

$$q = \text{const}; \quad R \le \left(2^n-2\right)\sum_{m=2}^{n}\left(C_n^m m\right)^2.$$

Следствие 3. Глобально оптимальная упаковка получится при $m = n$ с временной сложностью,

$$T(R) \le 2q\frac{n}{n-1}\cdot(n+2)\tau;$$

$$q = \text{const}; \quad R \le (2^n-2)n^2,$$

или $\qquad T((2^n-2)n^2) = O(n).$

Следствие 4. При $m = \lceil n/m\rceil$, или, $m \sim \sqrt{n}$, упаковка заканчивается за 2 шага, и согласно (7)

$$T(R) \le 2q\frac{\sqrt{n}}{\sqrt{n}-1}\cdot(n+3)\tau;$$

$$q = \text{const}; \quad R \le 2^{\sqrt{n}-1}\left(C_n^{\sqrt{n}}\right)^2 n,$$

так что

$$T(R) = O(n); \quad R \le 2^{\sqrt{n}-1}\left(C_n^{\sqrt{n}}\right)^2 n.$$

При этом все элементы каждого сочетания на первом шаге будут упакованы оптимально, равно как и на втором, однако глобальный оптимум на выходе алгоритма не является гарантированным.

Следствие 5. Если выполнять упаковку по алгоритму леммы 1, при этом по тому же алгоритму (а не перебором) выполнять упаковку каждого сочетания на текущем шаге, то временная сложность (6) перейдет в оценку

$$T(R) \le q\frac{m}{m-1}\cdot\left(n+\log_m n+1\right)\times$$

$$\times\left(1+q\frac{\tilde{m}}{\tilde{m}-1}\cdot\left(m+\log_{\tilde{m}} m+1\right)\right)\tau;$$

$$q=\text{const};\ R\le\frac{\left(C_n^m m\right)^2}{2}\cdot\frac{\left(C_m^{\tilde{m}}\tilde{m}\right)^2}{2},$$

где $\tilde{m} < m$, $\tilde{m}=\text{const}$ – число элементов сочетания из m по $\tilde{m}$. В частности, при $m \sim \sqrt{n}$ и $\tilde{m} \sim \sqrt{m}$ получится [7]:

$$T\left(\frac{1}{4}\left(C_n^{\sqrt{n}}\right)^2\left(C_{\sqrt{n}}^{\sqrt[4]{n}}\right)^2 n^{\frac{3}{2}}\right)=O\left(n^{\frac{3}{2}}\right).$$

Последняя оценка означает, что время возрастает, сохраняя полиномиальный рост порядка $n^{\frac{3}{2}}$ при снижении количества процессоров до

$$\frac{1}{4}\left(C_n^{\sqrt{n}}\right)^2\left(C_{\sqrt{n}}^{\sqrt[4]{n}}\right)^2 n^{\frac{3}{2}}.$$

Такую рекурсивную вложенность алгоритма можно было бы продолжить, однако ввиду отсутствия гарантии глобального оптимума упаковки на выходе алгоритма целесообразность более глубоких рекурсивных вложений не очевидна.

Представленные оценки в явном виде не указывают временную сложность генерации сочетаний. Но число шагов (3) не больше n, на каждом шаге – не больше числа входных элементов на шаге, поэтому временная сложность параллельной генерации сочетаний на основе (3) с точностью до постоянного коэффициента не повлияет на значения порядков данных выше оценок.

Заключение

Представлен синтез и анализ параллельного детерминированного алгоритма двумерной упаковки. Алгоритм построен на основе устойчивой параллельной сортировки и видоизменения формул Виета восстановления коэффициентов полинома по его корням. Видоизменение использовано для параллельной генерации всех сочетаний прямоугольников из заданного конечного множества. Временная сложность алгоритма имеет полиномиальную оценку, близкую к линейной, выполняемая упаковка оптимальна в каждом сочетании на каждом последовательном шаге, но глобально оптимальной она оказывается лишь в случае экспоненциального роста числа процессоров.

Список литературы

1. Березин И.С., Жидков Н.П. Методы вычислений. Т.1. – М.: Наука, 1966. – 632 с.
2. Ромм Я.Е. Бесконфликтные и устойчивые методы детерминированной параллельной обработки: дис. ... д-ра техн. наук. – Таганрог: ТРТУ, 1999. – 546 с.
3. Ромм Я.Е. Локализация и устойчивое вычисление нулей многочлена на основе сортировки. Ч. II // Кибернетика и системный анализ. – 2007. – № 2. – С. 161–174.
4. Ромм Я.Е. Параллельная сортировка слиянием по матрицам сравнений. Ч. I // Кибернетика и системный анализ. – 1994. – № 5. – С. 3–23.
5. Ромм Я.Е., Заика И.В. Численная оптимизация на основе алгоритмов сортировки с приложением к дифференциальным и нелинейным уравнениям общего вида // Кибернетика и системный анализ. – 2011. – № 2. – С. 165–180.
6. Ромм Я.Е., Назарьянц Е.Г. Параллельные алгоритмы решения булевой задачи о рюкзаке на основе сортировки и видоизменения формул Виета // Фундаментальные исследования. – 2015. – № 2–12. – С. 2575–2580.
7. Ромм Я.Е., Назарьянц Е.Г. Параллельный детерминированный алгоритм двумерной упаковки на основе сортировки и видоизменения формул Виета / ТИ имени А.П. Чехова (филиал) ФГБОУ ВО РГЭУ (РИНХ). – Таганрог, 2016. – С. 39. Деп. в ВИНИТИ 14.04.2016, № 61-В2016.
8. Солодовников В.И. Верхние оценки сложности решения систем линейных уравнений // Теория сложности вычислений. I: Записки научных семинаров ЛОМИ АН СССР. – Л., 1982. – т. 118. – С. 159–187.
9. Филиппова А.С., Валиахметова Ю.И. Теория оптимального использования ресурсов Л.В. Канторовича в работах Уфимской научной школы // Омский научный вестник. – 2013 – № 1(117) – С. 39–42.