Схему легко модифицировать, отказавшись от требований остроты и замкнутости опорного конуса. Подобная модификация позволит распространить ее на алгоритмы с ограниченным ресурсом параллелизма. Единственное, чего нехватает для реализации данной идеи, – это достаточного числа наполнений предложенной схемы конкретными алгоритмами. Но можно надеяться, что создание таких наполнений является лишь вопросом времени.

ЛЕКЦИЯ 9

Типовые информационные структуры

Содержание: *перемножение матриц, решение треугольных систем, неожиданный эффект, система с блочно-двухдиагональной матрицей, макро- и микрореализации, явная схема для уравнения теплопроводности, макро- и микропараллелизм, локальный алгоритм, очень "простой" пример, гипотеза о типовых структурах*.

Рассмотрим некоторые примеры графов конкретных алгоритмов. Мы не будем строить графы в пространствах итераций и не будем описывать их покрывающие функции. Со всем этим можно познакомиться в [1]. Мы будем размещать графы в подходящих пространствах малой размерности. Выбор размещения будет определяться только тем, чтобы можно было легко показать информационную структуру самих алгоритмов и продемонстрировать возможные наборы линейных разверток. В соответствии с определением графа алгоритма не нужно указывать дуги, связанные с рассылкой одних и тех же входных данных по вершинам графа. И очень часто они показываться не будут, главным образом, из-за сложности их изображения. Но иногда мы будем отступать от этого правила.

Перемножение матриц. Пусть решается классическая задача вычисления произведения $A = BC$ двух квадратных матриц B, C порядка n. Будем считать элементы матриц A, B, C числами и обозначим их a_{ij}, b_{ik}, c_{kj}. Согласно определению операции умножения матриц имеем

$$a_{ij} = \sum_{k=1}^{n} b_{ik} c_{kj}, \quad i, j = 1, 2, \ldots, n.$$

Эти формулы довольно часто используются для непосредственного вычисления элементов матрицы A. Сами по себе они не определяют алгоритм однозначно, так как не определен порядок суммирования произведений $b_{ik}c_{kj}$ под знаком суммы. Однако заметим, что явно виден параллелизм вычислений. Выражается он отсутствием указания о соблюдении какого-либо порядка перебора индексов i, j.

Если операции сложения и умножения чисел выполняются точно, то все порядки суммирования эквивалентны и приводят к одному и тому же результату. Пусть из каких-то соображений выбран следующий алгоритм реализации формул:

$$a_{ij}^{(0)} = 0,$$
$$a_{ij}^{(k)} = a_{ij}^{(k-1)} + b_{ik}c_{kj}, \quad i, j, k = 1, 2, \ldots, n,$$
$$a_{ij} = a_{ij}^{(n)}.$$

Снова явно указан параллелизм перебора индексов i, j. Однако по индексу k параллелизма нет, так как этот индекс должен перебираться последовательно от 1 до n.

Построим теперь граф этого алгоритма. Будем считать, что вершины графа соответствуют операциям вида $a + bc$. Чтобы не вносить в исследование графа алгоритма неоправданные дополнительные трудности, вершины графа нельзя располагать произвольно. Приемлемый способ их расположения подсказывает сама форма записи. Рассмотрим прямоугольную решетку в трехмерном пространстве с координатами i, j, k. Во все целочисленные узлы решетки для $1 \le i,\ j,\ k \le n$ поместим вершины графа. Анализируя запись, нетрудно убедиться в том, что в вершину с координатами i, j, k для $k > 1$ будет передаваться результат выполнения операции, соответствующей вершине с координатами i, j, $k - 1$.

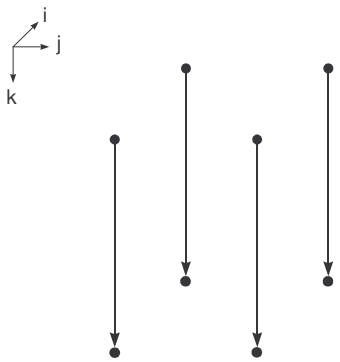

Рис. 9.1. Граф перемножения матриц.

Граф алгоритма устроен достаточно просто. Он распадается на n^2 не связанных между собой подграфов. Каждый подграф содержит n вершин и представляет один путь, расположенный параллельно оси k. Как и следовало ожидать, в графе отражен тот же параллелизм, который был явно указан в записи. Для $n = 2$ граф алгоритма представлен на рис. 9.1. Граф имеет полный набор линейных разверток, в качестве направляющих векторов которых могут быть взяты, например, координатные векторы. Векторы вдоль осей i, j дают обобщенные развертки, вектор вдоль оси k – строгую.

Системы с треугольной матрицей. Пусть теперь решается система алгебраических линейных уравнений $Ax = b$ с невырожденной треугольной матрицей A порядка n методом обратной подстановки. Обозначим через a_{ij}, b_i, x_i элементы матрицы, правой части и вектор-решения. Предположим для определенности, что матрица A левая треугольная с диагональными элементами, равными 1. Тогда имеем

$$x_1 = b_1,\ x_i = b_i - \sum_{j=1}^{i-1} a_{ij} x_j \quad 2 \le i \le n.$$

Эта запись также не определяет алгоритм однозначно, так как не определен порядок суммирования. Рассмотрим, например, последовательное суммирование по возрастанию индекса j. Соответствующий алгоритм можно записать следующим образом

$$x_i^{(0)} = b_i,$$
$$x_i^{(j)} = x_i^{(j-1)} - a_{ij}x_j^{(j-1)}, \quad i = 1, 2, \ldots, n, \quad j = 1, 2, \ldots, i-1,$$
$$x_i = x_i^{(i-1)}.$$

Основная операция алгоритма имеет вид $a - bc$. Выполняется она для всех допустимых значений индексов i, j. Все остальные операции осуществляют присваивание. В декартовой системе координат с осями i, j построим прямоугольную координатную сетку с шагом 1 и поместим в узлы сетки при $2 \le i \le n$, $1 \le j \le i - 1$ вершины графа, соответствующие операциям $a - bc$. Анализируя связи между операциями, построим граф алгоритма, включив в него также вершины, символизирующие ввод данных a_{ij} и b_i. Этот граф для случая $n = 5$ представлен на рис. 9.2 слева. Верхняя угловая вершина на левом рис. 9.2 находится в точке с координатами $i = 1$, $j = 0$. Граф имеет полный набор линейных разверток, в качестве направляющих векторов которых могут быть взяты, например, координатные векторы. Векторы вдоль оси i дает обобщенную развертку, вектор вдоль оси j – строгую.

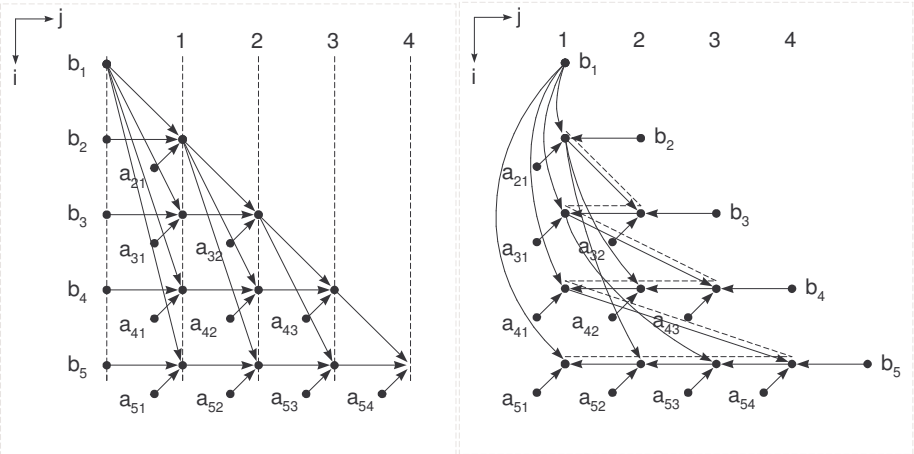

Рис. 9.2. Графы для треугольных систем.

Если при вычислении суммы мы останавливаемся по каким-то соображениям на последовательном способе суммирования, то выбор суммирования именно по возрастанию индекса j был сделан, вообще говоря, случайно. Так как в этом выборе пока не видно каких-либо преимуществ, то можно построить алгоритм обратной подстановки с суммированием по убыванию индекса j. Соответствующий алгоритм таков:

$$x_i^{(i)} = b_i,$$
$$x_i^{(j)} = x_i^{(j+1)} - a_{ij}x_j^{(j)},\ i = 1, 2, \ldots, n,\ j = i - 1, i - 2, \ldots, 1,$$
$$x_i = x_i^{(1)}.$$

Его граф для случая $n = 5$ представлен на рис. 9.2 справа. Теперь верхняя угловая вершина находится в точке с координатами $i = 1, j = 1$.

Пытаясь размещать вершины, соответствующие операциям типа $a - bc$, по ярусам хотя бы какой-нибудь параллельной формы, мы обнаруживаем, что теперь в каждом ярусе может находиться только одна вершина. Объясняется это тем, что все вершины графа на правом рис. 9.2 лежат на одном пути. Этот путь показан пунктирными стрелками. Граф алгоритма имеет только одну линейную развертку, причем обобщенную. Ее направляющий вектор направлен вдоль оси i.

Полученный результат трудно было ожидать. Действительно, оба рассмотренных алгоритма предназначены для решения одной и той же задачи. Они построены на одних и тех же исходных формулах и в отношении точных вычислений эквивалентны. Оба алгоритма совершенно одинаковы с точки зрения их реализации на однопроцессорных компьютерах, так как требуют выполнения одинакового числа операций умножения и вычитания и одинакового объема памяти. На классе треугольных систем оба алгоритма даже эквивалентны с точки зрения влияния ошибок округления. Тем не менее, графы обоих алгоритмов принципиально отличаются друг от друга. Если эти алгоритмы реализовывать на параллельной вычислительной системе, имеющей n универсальных процессоров, то первый алгоритм можно реализовать за время пропорциональное n, а второй – только за время пропорциональное n^2. В первом случае средняя загруженность процессоров близка к 0,5, во втором случае она близка к 0.

Таким образом, алгоритмы, совершенно одинаковые с точки зрения реализации на "обыкновенных" компьютерах, могут быть принципиально различными с точки зрения реализации на параллельных компьютерах.

Система с блочно-двухдиагональной матрицей. Среди многих задач линейной алгебры, возникающих при решении уравнений математической физики сеточными методами, часто встречается задача решения систем линейных алгебраических уравнений с блочно-двухдиагональными матрицами. При этом внедиагональные блоки представляют диагональные матрицы, диагональные блоки — двухдиагональные матрицы. Будем считать для определенности, что матрица системы является левой треугольной. Пусть она имеет блочный порядок m и порядок блоков, равный n. Итак, рассмотрим систему линейных алгебраических уравнений $Au = f$ следующего вида:

$$\begin{bmatrix} B_1 & & & \\ D_1 & B_2 & & \mathrm{O} \\ & D_2 & B_3 & \\ & \cdots & \cdots & \cdots \\ \mathrm{O} & & & D_{m-1} B_m \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ \vdots \\ U_m \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ \vdots \\ F_m \end{bmatrix},$$

где

$$B_k = \begin{bmatrix} b_{1k} & & & \\ e_{1k} & b_{2k} & & \mathrm{O} \\ & \mathrm{e}_{2k} & b_{3k} & \\ & \cdots\cdots\cdots & & \\ \mathrm{O} & & e_{n-1,k} & b_{nk} \end{bmatrix}, \quad D_k = \begin{bmatrix} \mathrm{d}_{1k} & & \\ & d_{2k} & \mathrm{O} \\ & & d_{3k} \\ & & \cdots \\ \mathrm{O} & & d_{nk} \end{bmatrix},$$

$$U_k = \begin{bmatrix} u_{1k} \\ u_{2k} \\ \vdots \\ u_{nk} \end{bmatrix}, \quad F_k = \begin{bmatrix} f_{1k} \\ f_{2k} \\ \vdots \\ f_{nk} \end{bmatrix}.$$

Решение блочно-двухдиагональной системы определяется рекуррентно:

$$U_k = B_k^{-1}(F_k - D_{k-1}U_{k-1}),\ 1 \le k \le m,$$

если положить $U_0 = 0,\ D_0 = 0$. Макрооперация $X = B^{-1}(F - DY)$ вычисляет вектор X по векторам F, Y и матрицам B, D. Построим граф алгоритма, считая, что каждая из его вершин соответствует данной макрооперации. Очевидно, он будет таким, как показано на рис. 9.3. Большие размеры вершин и дуг на рисунке подчеркивают, что операции являются сложными и передается сложная информация.

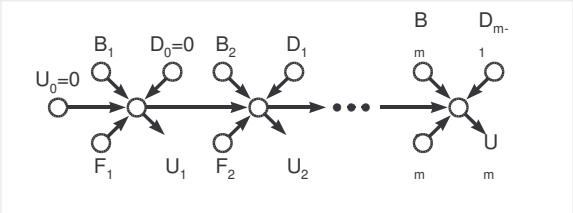

Рис. 9.3. Макрограф для блочно-диагональной системы.

Из строения графа сразу видно, что если алгоритм рассматривать как последовательность матрично-векторных операций, то он является строго последовательным и не распараллеливается. Практически не распараллеливается и каждая макрооперация в отдельности. Она также представляет решение двухдиагональной системы. Из этих фактов можно было бы сделать вывод о невозможности хорошего распараллеливания алгоритма решения рассматриваемой системы с блочно-двухдиагональной матрицей. Однако такой вывод был бы преждевременным.

Исследуем поэлементную запись алгоритма решения блочно-двухдиагональной системы. С учетом введенных ранее обозначений элементов матриц и векторов имеем:

$$u_{ik} = (f_{ik} - e_{i-1,k}u_{i-1,k} - d_{i,k-1}u_{i,k-1})b_{ik}$$

$$k = 1, 2, \ldots, m, \quad i = 1, 2, \ldots, n.$$

При этом предполагается, что $u_{i0} = u_{0k} = e_{0k} = d_{i0} = 0$ для всех i, k. В этой записи основной и, по существу, единственной является скалярная операция

$$u = b^{-1}(f - ex - dy),$$

которая вычисляет величину u по величинам f, e, x, y, b, d. Для построения графа алгоритма рассмотрим прямоугольную решетку, узлы которой имеют целочисленные координаты i, k. Во все узлы решетки для $1 \le i \le n$, $1 \le k \le m$ поместим вершины графа и будем считать, что они соответствуют операции u. Не будем указывать вершины, поставляющие входные данные и нулевые значения некоторых аргументов. Анализируя поэлементную запись, нетрудно убедиться в том, что в вершину с координатами i, k будут передаваться результаты выполнения операций, соответствующих вершинам с координатами $i-1$, k и i, $k-1$. Вся остальная информация, необходимая для реализации операции с координатами i, k, является входной и нужна для реализации только этой операции.

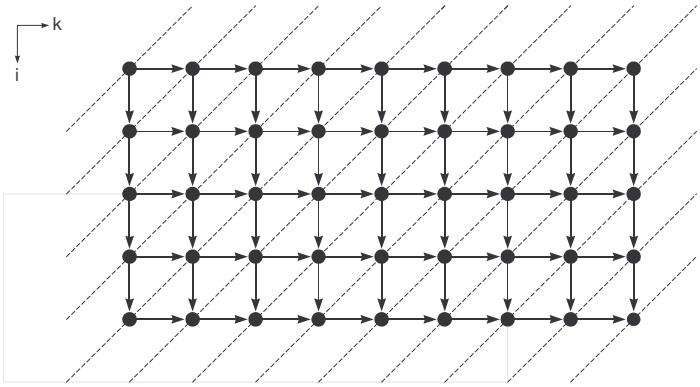

Рис. 9.4. Граф для блочно-двухдиагональной системы.

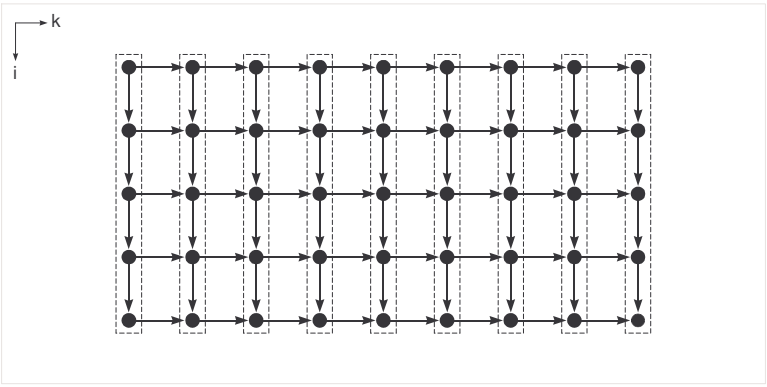

Рис. 9.5. Ярусы обобщенной параллельной формы.

Для случая $n = 5$, $m = 9$ граф алгоритма представлен на рис. 9.4. Из него следует, что вопреки возможным ожиданиям граф алгоритма прекрасно распараллеливается. Граф имеет полный набор обобщенных линейных разверток, в качестве направляющих векторов которых могут быть взяты координатные векторы. Линейная развертка с направляющим вектором (1, 1) является строгой. На рис. 9.5 в этом же графе пунктирными линиями обведены группы вершин. Каждая из таких групп соответствует одной макровершине графа, представленного на рис. 9.3. Из этих рисунков видно, каким образом хорошо распараллеливаемый алгоритм может превратиться в нераспараллеливаемый при неудачном укрупнении операций.

Явная схема для уравнения теплопроводности. Рассмотрим далее решение краевой задачи для одномерного уравнения теплопроводности. Пусть требуется найти решение $u\,(y, z)$, где

$$\frac{\partial u}{\partial y} = \frac{\partial^2 u}{\partial z^2},\ \ 0 \le z, 0 \le y \le T$$

$$u(0, z) = \varphi(z),\ \ u(y, 0) = u_0(y),\ \ u(y, 1) = u_1(y).$$

Построим равномерную сетку с шагом h по z и шагом τ по y. Предположим, что по тем или иным причинам выбора выбрана явная схема

$$\frac{u_j^{(i)} - u_j^{(i-1)}}{\tau} = \frac{u_{j-1}^{(i-1)} - 2u_j^{(i-1)} + u_{j+1}^{(i-1)}}{h^2}$$

Пусть алгоритм реализуется в соответствии с формулой

$$u_j^{(i)} = u_j^{(i-1)} + \frac{\tau}{h^2}(u_{j-1}^{(i-1)} - 2u_j^{(i-1)} + u_{j+1}^{(i-1)}).$$

Для построения графа алгоритма введем прямоугольную систему координат с осями i, j. Переменные y, z связаны с переменными i, j соотношениями

$$y = \tau i,\ \ z = hj.$$

Поместим в каждый узел целочисленной решетки вершину графа и будем считать ее соответствующей скалярной операции

$$\omega = a\,(1 - \frac{2\,\tau}{h^2}) + \frac{\tau}{h^2}(b + c),$$

выполняемой для разных значений аргументов a, b, c. Для случая $h = 1/8$, $\tau = T/6$ граф алгоритма представлен на обоих рисунках 9.6. На границе области расположены вершины, символизирующие ввод начальных данных и граничных значений. Граф имеет полный набор обобщенных линейных разверток. В качестве их направляющих векторов в системе координат i ,j могут быть взяты, например, векторы (1, 1) и (1, – 1). Имеются и строгие линейные развертки. Одна из них задается направляющим вектором (1, 0).

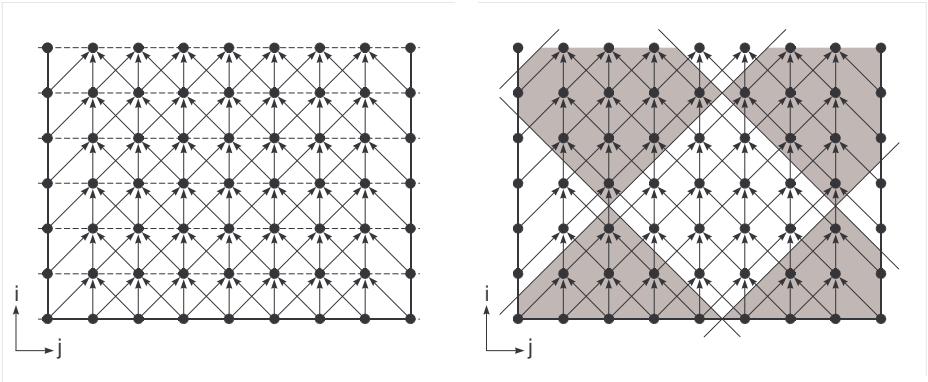

Рис. 9.6. Микро- и макропараллелизм в графе.

Операции одного яруса этой развертки не зависят друг от друга и их можно реализовывать на разных устройствах одновременно. Но эффективность такой организации параллельных вычислений может оказаться очень низкой. На рис. 9.6 слева пунктирными линиями показаны ярусы максимальной параллельной формы алгоритма. Предположим, что операции реализуются по этим ярусам. Каждая операция одного яруса требует трех аргументов. Они являются результатами выполнения операций на предыдущем ярусе. Если данные, полученные на одном ярусе, могут быстро извлекаться из памяти, то никаких серьезных проблем с реализацией алгоритма по ярусам параллельной формы не возникает. Однако для больших многомерных задач ярусы оказываются столь масштабными, что информация о них может не поместиться в одной быстрой памяти. Тогда для ее размещения приходится использовать либо медленную, либо распределенную память. Для этих видов памяти время выборки одного числа может существенно превышать время выполнения базовой операции. Это означает, что при переходе к очередному ярусу время, затраченное на выполнение операций, может оказаться значительно меньше времени взаимодействия с памятью. Чем меньше отношение времени выполнения операций к времени доступа к памяти, тем меньше эффективность реализации алгоритма по ярусам параллельной формы. В этом случае бóльшая часть времени работы параллельного компьютера будет тратиться на осуществление обменов с памятью, а не на собственно счет.

Ранее уже отмечалось, что знание двух и более обобщенных разверток позволяет перейти от микрооописания алгоритма к его макрооописанию. Такой переход показан на рис. 9.6 справа. Поверхности уровней обобщенных разверток разбивают область задания графа на многогранники. Исходный

алгоритм теперь можно реализовывать по полученным многогранникам, в том числе, параллельно. При этом одному процессору всегда поручается выполнение всех операций, относящихся к одному многограннику. При параллельной реализации сначала выполняются операции, соответствующие нижним заштрихованным многогранникам на правом рис. 9.6. Затем параллельно выполняются операции, соответствующие соседним незаштрихованным многогранникам и т. д.

В новом процессе операции одного многогранника становятся макрооперацией. Время выполнения макрооперации определяется числом вершин в многограннике, что примерно пропорционально его площади. Информационную связь между собой многогранники осуществляют через вершины, лежащие около границ. Следовательно, время на извлечение из памяти информации, необходимой для реализации макрооперации, определяется длиной границы многогранника. Размеры многогранников можно выбрать произвольно. Они зависят только от того, какие ярусы обобщенных параллельных форм формируют их границы. Всегда можно выбрать такое разбиение области задания графа, при котором для большинства многогранников отношение длин границ к их площадям будет сколь угодно малым. При реализации таких макроопераций влияние времени доступа к медленной памяти будет снижено очень сильно.

Метод Жордана. Рассмотрим решение системы линейных алгебраических уравнений методом Жордана без выбора ведущего элемента. Этот метод можно записать многими различными способами. Пусть, например, он записан так:

1. $l_1 = 1/a_{1k}, \qquad k = 1,\ldots,n,$
2. $l_p = -a_{pk}, \qquad k = 1,\ldots,n,\ p = 2,\ldots,n,$
3. $u_j = a_{1j}l_1, \qquad k = 1,\ldots,n,\ j = k+1,\ldots,n+1,$
4. $a_{i-1,j} = a_{ij}+l_iu_j, \quad k = 1,\ldots,n,\ j = k+1,\ldots,n+1,\ i = 2,\ldots,n,$
5. $a_{nj} = u_j, \qquad k = 1,\ldots,n,\ j = k+1,\ldots,n+1.$

Будем считать, что $n \geq 3$. Расположим опорные многогранники операторов 1 – 3, 5 вокруг опорного многогранника оператора 4. Область, занимаемая вершинами графа алгоритма, изображена на рис. 9.7.

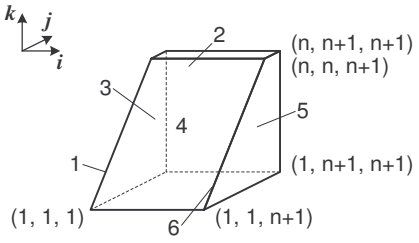

Рис. 9.7. Расположение вершин метода Жордана.

Она представляет усеченную пирамиду, из которой исключено ребро 6. Вершины опорного многогранника 1 расположены на ребре 1, многогранника 2 – на грани 2, многогранника 3 – на грани 3, многогранника 5 – на грани 5, вершины опорного многогранника 4 расположены в остальной части пирамиды. Основной объем вычислений приходится на оператор 4, который является телом тройного цикла. Неоднородности вычислений, связанные с операторами1 – 3, 5, локализованы на границе. Структура связей в графе показана на рис. 9.8.

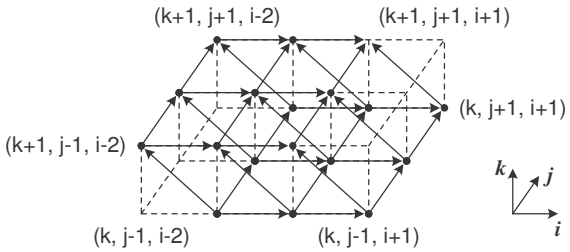

Рис. 9.8. Граф метода Жордана.

Граф является регулярным. В каждую его вершину входят и выходят дуги, определяемые векторами (1, 0, –1), (0, 1, 0), (0, 0, 1) в системе координат k, j, i. Рассылка элементов u_j осуществляется вдоль прямых, параллельных оси i. Она начинается на грани 3 и заканчивается на грани 5, проходя через соответствующие точки пирамиды. Рассылка элементов l_p начинается на грани 2 и ребре 1 и проходит через соответствующие точки пирамиды на прямых,

параллельных оси *j*. При этом полагается, что *p* совпадает с *i*. Входные данные алгоритма, т.е. элементы матрицы и правой части, поступают через нижнюю грань. Граф имеет полный набор обобщенных линейных разверток. В качестве их направляющих векторов в системе координат *k*, *j*, *i* могут быть взяты, например, векторы (1, 0, 0), (1, 0, 1) и (0, 1, 0). Имеются и строгие линейные развертки. Одна из них задается направляющим вектором (2, 1, 1).

Заметим, что в отличие от других графов описанная укладка графа для метода Жордана появилась далеко не сразу. Пришлось перепробовать много вариантов, прежде чем удалось найти столь красивое расположение графа. Возможно, что у графа каждого отработанного и хорошо изученного алгоритма можно найти какие-то изящные укладки. Вот только обнаружить их очень не просто. Тем не менее, кажется, что их поиск достоин большего внимания. Ведь наличие каталога красиво уложенных графов типовых алгоритмов имело бы большое значение как для теории, так и для практики.

Локальный алгоритм. Яркий пример нерегулярного локального алгоритма дает использование явной схемы на нерегулярной или адаптивной сетке. На рис. 9.9 приведен некоторый гипотетический пример. Граф локальный, имеет полный набор разверток. С помощью разверток осуществляется разбиение алгоритма на параллельно выполняемые фрагменты. На рис. 9.9 графы фрагментов заштрихованы.

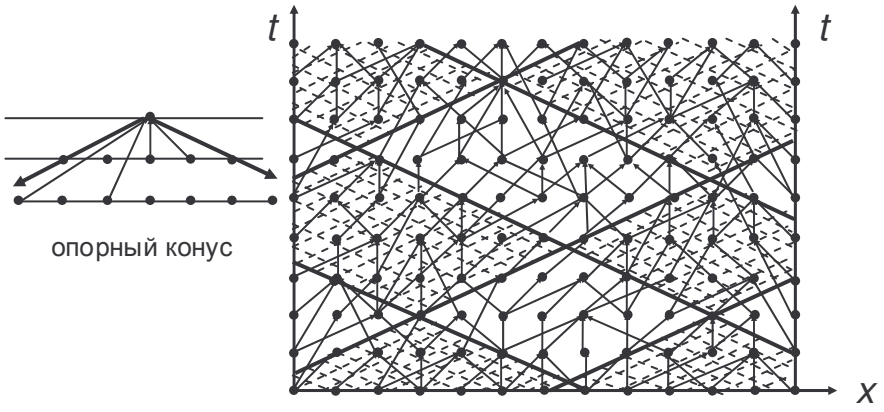

Рис. 9.9. Граф локального алгоритма.

Очень "простой" пример. Этот пример не был взят ни из теории, ни из практики. Он специально придуман для того, чтобы показать, насколько сложными могут быть информационные связи даже у вроде бы самых простых алгоритмов. Формально данный пример реализует некоторую сортировку из *n* чисел. Впоследствии выяснилось, что он является очень хорошим тестом для проверки того, насколько эффективно та или иная технология определяет параллелизм в записях алгоритмов. Заметим, что на этом тесте "сломались",

т.е. не смогли обнаружить никакого параллелизма, все доступные нам параллелизующие компиляторы и автономные системы как отечественные, так и зарубежные. Пример записывается следующим образом:

$$u_{i+j} = u_{2n+1-i-j}, \quad i = 1,\ldots, n, \quad j = 1,\ldots, n.$$

Мы не будем исследовать структуру данного алгоритма, поскольку она детально изучена в [1]. Заметим лишь, что нет никакого параллелизма ни по i, ни по j. Однако если

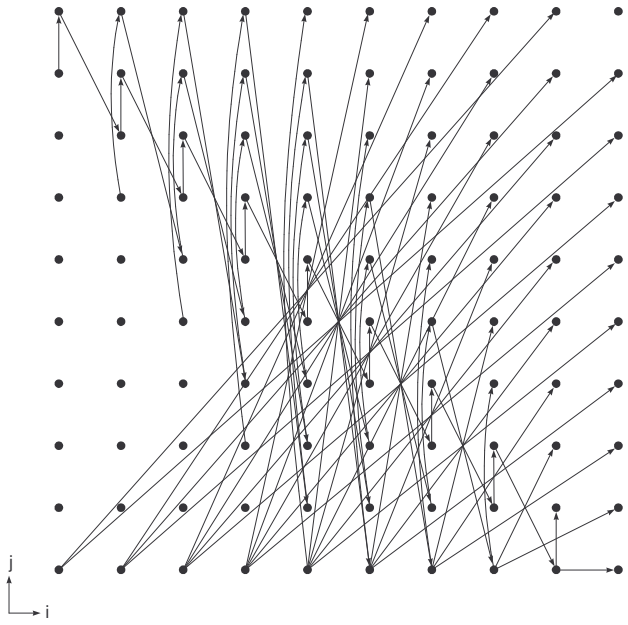

Рис. 9.10. Граф "простого" примера.

алгоритм записать несколько иначе, именно

$$u_{i+j} = u_{2n+1-i-j}, \quad i = 1,\ldots, n, \quad j = 1,\ldots, n-i, \quad j = n-i+1,\ldots, n,$$

то при любом фиксированном i операции по j в каждой из двух групп уже можно выполнять параллельно. Сами же группы надо по-прежнему выполнять последовательно. Граф рассматриваемого алгоритма действительно устроен очень сложно. Это видно хотя бы по рис. 9.10, на котором граф изображен для $n = 10$.

В связи с рассмотренными примерами обратим внимание на следующее обстоятельство. Несмотря на внешнее разнообразие, графы многих алгоритмов имеют немало общего. Точнее, с помощью не очень сложных преобразований их можно свести не только к регулярным графам, но и к регулярным графам с координатными векторами связей за счет некоторого усложнения вершин-операций. Мы уже видели, что даже произвольный локальный граф поддается такому преобразованию. Техника проведения подобных преобразований частично была описана ранее. Более детально применительно к разным ситуациям с ней можно познакомиться в [1].

Построение графов для большого числа конкретных алгоритмов выявило удивительную закономерность: большое разнообразие алгоритмов не приводит к такому же разнообразию информационных структур. Многие графы формально различных алгоритмов оказались изоморфными в главном, отличаясь друг от друга содержанием вершин и дуг. Например, на всем множестве алгоритмов линейной алгебры различных по существу типов графов оказалось всего лишь порядка десятка. Все это привело к предположению, что, возможно, верна

Гипотеза. Типовых информационных структур алгоритмов в конкретных прикладных областях немного.

Пока практика подтверждает эту гипотезу. Если гипотеза о типовых структурах окажется верной, то откроется много новых связей и направлений исследований. Изложение численных методов может быть поставлено на общий информационный фундамент, распараллеливание типовых информационных структур может быть заранее изучено и реализовано с помощью специальных программных средств, по типовым структурам могут быть построены спецпроцессоры, осуществляющие быстрое решение нужных алгоритмов. Отсюда уже недалеко и до построения заказных вычислительных систем, ориентированных на эффективное решение классов задач из конкретных прикладных областей. В частности, математические модели спецпроцессоров для таких заказных систем вполне можно строить, используя описанную ранее гомоморфную свертку графов тех же самых типовых структур.

ЛЕКЦИЯ 10

Параллельные вычисления и математическое образование

Содержание: *что заставляет менять образование, параллельные вычисления на стыке дисциплин, последовательные вычисления маскируют проблемы развития, необходимость учить решать задачи эффективно, причина многих трудностей – незнание структуры алгоритмов, возможные пути изменения ситуации.*