ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
МЕХАНИКО-МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ
КАФЕДРА ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ

В.Н. Берцун

МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ НА ГРАФАХ

Часть II

Издательство Томского университета
2013

УДК 519.17
ББК 22.174
Б 527

Рецензенты:

Доктор техн. наук, профессор ТГУ
А. Ю. Матросова
Кандидат техн. наук, доцент ТГУ
В. А. Беляев

Берцун В.Н.
Б 527 Математическое моделирование на графах. Часть 2: Томск: Изд-во Том. ун-та, 2013. − 88 с.

ISBN 978−5−7511−2211−9

Описывается математическое моделирование прикладных задач и оптимизация вычислительных алгоритмов для высокопроизводительных компьютеров (кластеров).

В книге содержатся три раздела теории графов: матрицы, связанные с графами, характеристические числа графов и параллельные алгоритмы на графах.

Для специалистов, занимающихся математическим моделированием прикладных задач, и студентов математических и физико-математических факультетов.

УДК 519.17
ББК 22.174

ISBN 978−5−7511−2211−9

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Граф – это наглядный образ, который дает максимум пространственных и структурных представлений, является одним из гибких математических объектов, способных легко приспосабливаться под любую конкретную модель [1−5].

Графами представляются схемы авиалиний и схемы метро, нейронные сети, а на географических картах – реки и железные дороги. В виде графов можно изображать химические молекулы и отношения между людьми, электронные схемы и информационную структуру алгоритмов. Свойства и алгоритмы теории графов используются в поисковых системах, обработке изображений, а также при решении задач логистики, гемодинамики, управления рисками и динамики мехатронных систем.

Большинство численных методов решения краевых задач основаны на моделировании непрерывной области изменения независимых переменных связным графом (регулярной или нерегулярной сеткой). Значительный интерес представляет решение прикладных задач по расчету, например, стержневых систем, характеристик течения в системах трубопроводов, теплового состояния электрических сетей, область определения которых является связным графом.

При создании экономичных параллельных вычислительных алгоритмов на графах большой размерности часто требуется обеспечить сбалансированную загрузку процессоров кластера на основе рационального разбиения многомерного сеточного графа на домены с учетом минимизации обменов.

3. ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ НА ГРАФАХ

Решение значительного числа прикладных задач может быть сведено к анализу соответствующих им графов.

3.1. Алгоритм Дейкстры

В большинстве алгоритмов поиска кратчайших путей используется тот факт, что любой кратчайший путь сам состоит из кратчайших путей. Наиболее эффективный алгоритм решения задачи о поиске кратчайшего пути между двумя любыми фиксированными вершинами в связном графе (например, между вершинами s и t) с неотрицательными весами предложил в 1959 г. Дейкстра [7, 9]. В основе этого алгоритма лежит принцип «жадности» (на каждом шаге выбирается локально лучший вариант). Это позволяет последовательно вычислять расстояния сначала до ближайшей к s вершине, а затем до следующей ближайшей и т. д. Метод основан на приписывании вершинам временных пометок, причем пометка вершины дает верхнюю границу длины пути от s к этой вершине. Величины этих пометок постепенно уменьшаются с помощью некоторой итерационной процедуры. На каждом шаге итерации только одна из временных пометок становится постоянной и соответствует точной длине кратчайшего пути от s к соответствующей вершине. Последовательность вершин, через которые проходит кратчайший путь, определяется с помощью последовательного вычитания из длины пути длин ребер, лежащих на кратчайшем пути (метод последовательного возвращения, внешний способ [36, 37]).

Рассмотрим этапы этого алгоритма.

Пусть $l(s)$ обозначает пометку вершины x_i (текущее расстояние от s до x_i).

Присвоение начальных значений.

Шаг 1. Положить $l(s)=0$ и считать эту пометку постоянной. Положить $l(x_i)=\infty$ для всех $x_i \neq s$ и считать эти пометки временными. Положить $p=s$.

Обновление пометок

Шаг 2. Для всех $x_i \in \Gamma(p)$, пометки которых временные, изменить их в соответствии со следующим выражением:

$$l(x_i) = \min[l(x_i),\ l(p) + c(p, x_i)]\,, \qquad (3.1)$$

где $c(p, x_i)$ - длина пути из p в x_i, $\Gamma(p)$ – множество смежных с p вершин графа.

Превращение временной пометки в постоянную

Шаг 3.Среди всех вершин $x_i \in \Gamma(p)$ с временными пометками найти такую вершину x_i^*, для которой

$$l(x_i^*) = \min[l(x_i)].$$

Шаг 4. Считать пометку вершины x_i^* постоянной и положить $p = x_i^*$.

Шаг 5. Если $p=t$, то $l(p)$ является длиной кратчайшего пути, если $p \neq t$, перейти к шагу 2.

В качестве примера рассмотрим граф, изображенный на рис. 3.1.

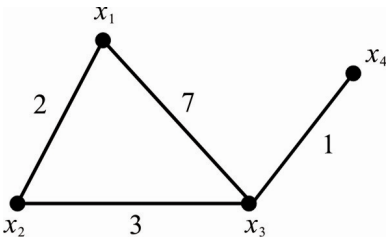
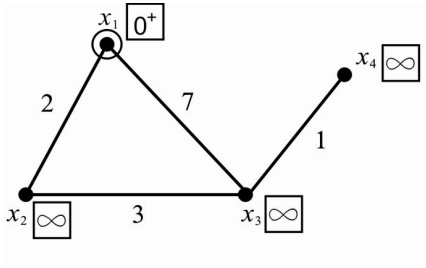


Рис. 3.1. Взвешенный граф для n=4

Пусть требуется найти кратчайший путь от вершины $x_1 = s$ до вершины $x_4 = t$. Используем алгоритм Дейкстры.

Шаг 1. Положим $l(s)$=0 и будем считать эту пометку постоянной. Положим $l(x_2) = l(x_3) = l(x_4) = \infty$.



Первая итерация

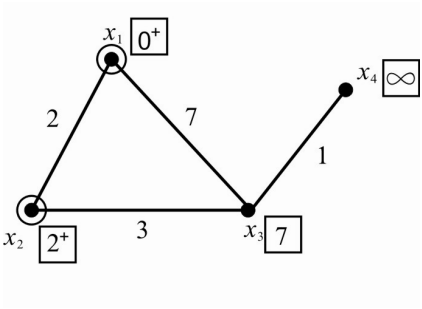
Шаг 2. $\Gamma(x_1) = \{x_2, x_3\}$,

$$l(x_2) = \min(\infty, 0+2) = 2,$$

$$l(x_3) = \min(\infty, 0+7) = 7.$$

Шаг 3. $\min(2,7)=2$, поэтому x_2 получает постоянную пометку 2^+

Шаг 4. p= x_2.



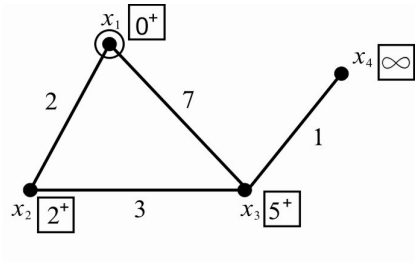
Вторая итерация

Шаг 2. $\Gamma(x_2)=\{x_1,x_3\}$, но x_1 имеет постоянную пометку. Тогда

$$l(x_3)=\min(7,2+3)=5.$$

Шаг 3. $\min(5)=5$. Вершина x_3 получает постоянную пометку 5^+.

Шаг 4. p= x_3.

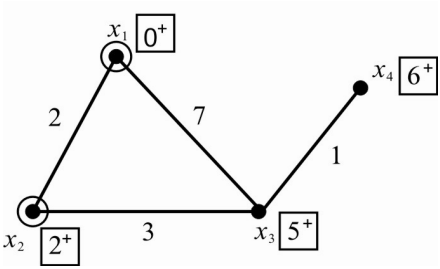


Третья итерация

Шаг 2. $\Gamma(x_3)=\{x_1,x_2,x_4\}$

$$l(x_4)=\min(\infty,5+1)=6.$$

Шаг 3. $\min(6)=6$. Так как все вершины, кроме x_4, помечены, то x_4 получает постоянную пометку 6^+.



Шаг 4. Полагаем $p = x_4 = t$. Все вершины имеют постоянные метки. Кратчайший путь от x_1 до x_4 проходит через вершины (x_2, x_3), а его длина равна 6.

Для получения верхней оценки $\Delta(n)$ вычислительной сложности алгоритма Дейкстры заметим, что на первой итерации этого алгоритма должна быть просмотрена $(n-1)$ вершина. Поскольку при этом необходимо вычислять выражение (3.1), то на первой итерации выполняется $(n-1)$ операция сложения, $(n-1)$ операция сравнения, а также производится выбор наименьшего из $(n-1)$ чисел. Таким образом, первая итерация реализуется за $3(n-1)$ операций. Вторая итерация выполняется за $3(n-2)$ операций, третья – за $3(n-3)$ операций и т. д. Поэтому

$$\Delta(n) = \sum_{i=1}^{n} 3(n-i) = 3n(n-1)/2.$$

Приведем теперь пример применения алгоритма Дейкстры для ориентированного графа из рис. 3.2.

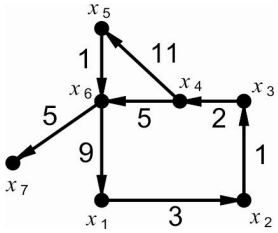


Рис. 3.2. Взвешенный орграф (n=7)

Определим расстояния от вершины x_1 до всех остальных вершин связного графа и сами эти пути, для нахождения которых будем ис-

пользовать встроенный способ [16, 37]. Матрица примыканий для рассматриваемого графа имеет вид

$$A_p = \begin{pmatrix} 0 & 3 & \infty & \infty & \infty & \infty & \infty \\ \infty & 0 & 1 & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & 2 & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & 11 & 5 & \infty \\ \infty & \infty & \infty & \infty & 0 & 1 & \infty \\ 9 & \infty & \infty & \infty & \infty & 0 & 5 \\ \infty & \infty & \infty & \infty & \infty & \infty & 0 \end{pmatrix}.$$

Введем в рассмотрение:

1. Вектор M с компонентами

$$m_i = \begin{cases} 0, \text{если вершина } x_i \text{ еще не рассмотрена}, \\ 1, \text{если вершина } x_i \text{ уже рассмотрена} \end{cases};$$

2. Вектор *B,* каждая компонента b_k которого соответствует текущему кратчайшему расстоянию от исходной вершины x_i до вершины x_k;

3. Вектор C, каждая компонента c_k которого на данной итерации есть индекс номера предпоследней вершины на кратчайшем пути от вершины x_i до вершины x_k.

Шаг 1. В цикле от 1 до 7 заполнить нулями массив M; заполнить числом i массив C; перенести i-тую строку матрицы A_p в массив B;

$M[i]:=1$; $C[i]:=0$; {i=1 – номер стартовой вершины}.

Шаг 2. Найти минимальную компоненту вектора B для неотмеченных вершин (т.е. тех k, для которых $M[k] = 0$);

пусть минимум достигается на индексе j, т.е. $b_j \le b_k$. Затем выполняются следующие операции:

$$M[j]:=1;$$
$$\text{если } b_k > b_j + w_{jk}, \text{ то } (b_k := b_j + w_{jk};\ c_k = j). \qquad (3.2)$$

Условие (3.2) означает, что путь (x_i,..,x_k) длиннее, чем путь для последовательности вершин (x_i, …,x_j ,x_k).

Если все $M[k]$ отмечены, то длина кратчайшего пути от x_i до x_k соответствует компоненте $B[k]$.

*Шаг 3.*Путь от x_i до x_k определяется в обратном порядке следующей процедурой:

1. $z:=C[k]$;

2. Выдать *z;*

3. *z:*=*C*[*z*]. Если *z* = 0, то конец,
иначе перейти к 2.

Пусть, например, требуется найти кратчайшие пути из вершины x_1 во все остальные для орграфа из рис.3.2. Содержимое массивов *M, B* и *C* после *шага 1* имеет следующий вид:

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
M	1	0	0	0	0	0	0
B	0	3	∞	∞	∞	∞	∞
C	0	1	1	1	1	1	1

Содержание массивов меняется по мере выполнения *шага 2* и представлено в табл.3.1.

Таблица 3.1

Этапы вычисления векторов B и C

		x_1	x_2	x_3	x_4	x_5	x_6	x_7
min b_k= 3	*M*	1	1	0	0	0	0	0
	B	0	3	4	∞	∞	∞	∞
	C	0	1	2	2	2	2	2
min b_k= 4	*M*	1	1	1	0	0	0	0
	B	0	3	4	6	∞	∞	∞
	C	0	1	2	3	3	3	3
min b_k= 6	*M*	1	1	1	1	0	0	0
	B	0	3	4	6	17	11	∞
	C	0	1	2	3	4	4	4
min b_k= 11	*M*	1	1	1	1	0	1	0
	B	0	3	4	6	17	11	16
	C	0	1	2	3	4	4	6

min b_k= 16		x_1	x_2	x_3	x_4	x_5	x_6	x_7
	M	1	1	1	1	0	1	1
	B	0	3	4	6	17	11	16
	C	0	1	2	3	4	4	6
min b_k= 17		x_1	x_2	x_3	x_4	x_5	x_6	x_7
	M	1	1	1	1	1	1	1
	B	0	3	4	6	17	11	16
	C	0	1	2	3	4	4	6

Таким образом, получены результирующие массивы *M, B* и *C*, в которых содержатся длины кратчайших путей из x_1 до всех остальных вершин графа, а также последовательность индексов вершин, через которые эти пути проходят. Например, кратчайший путь из x_1 в x_7 содержит следующие промежуточные вершины ($x_2, x_3, , x_4, x_6$).

Отметим, что решение задачи об определении кратчайшего расстояния от вершины x_1 до всех остальных вершин связного графа фактически привело к построению *покрывающего ориентированного дерева* с корнем в вершине x_1 .

Если граф путей содержит дуги и ребра, то, заменив каждое ребро на две противоположно направленные дуги равного веса, исходную задачу об определении кратчайшего пути можно свести к аналогичной задаче для ориентированного графа.

Важный практический интерес представляет задача определения в графе не только кратчайшего пути, но и второго, третьего и т. д. по длине пути. Например, если пассажир опоздал на минимальный по времени автобусный маршрут между городами, то возникает задача выбора второго по оптимальности маршрута. Для решения такой задачи можно воспользоваться *алгоритмом двойного поиска*.

Когда некоторые из дуг графа имеют отрицательные веса, кратчайший путь из одной вершины можно определить по *алгоритму Флойда – Уоршелла* или *Беллмана- Форда.*

Отметим, что всего в графе из n вершин порядка $O(n^2)$ путей, поэтому применение алгоритма Дейкстры для всех пар вершин $(i=1,\ldots,n-1)$ имеет сложность $O(n^3)$.

В задачах сетевого планирования и управления (СПУ) для топологически отсортированного ациклического орграфа возникает необ-

ходимость определения максимального (критического) пути между источником *s* и стоком *t* в сети. Такую задачу можно решить, например, с помощью алгоритма Дейкстры, если на *шаге* 3 этого алгоритма выбирается вершина x_i^*, для которой

$$l(x_i^*) = \max[l(x_i)] .$$

3.2. Алгоритм Флойда и его модификация

Алгоритм Флойда (1962 г.) позволяет найти кратчайшие расстояния между всеми парами вершин (ор)графа. Он основан на систематическом анализе *треугольного оператора* из рис. 3.3 с вершинами x_i, x_k, x_j и расстояниями a_{ij}, a_{ik}, a_{kj}.

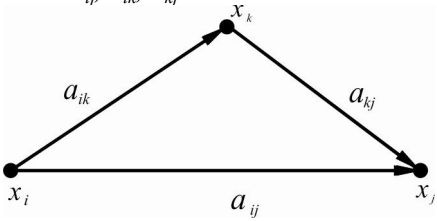


Рис. 3.3. Ациклический орграф для n=3

Если $a_{ij} > a_{ik}+a_{kj}$, то путь $x_i \to x_k \to x_j$ короче пути $x_i \to x_j$.

Рассмотрим граф, изображенный на рис. 3.4.

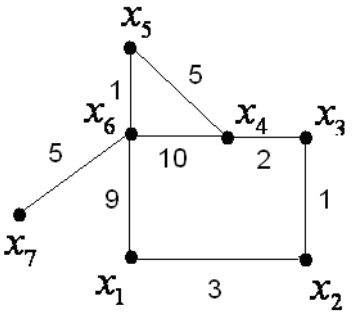


Рис. 3.4 Взвешенный граф

Матрица примыканий для этого графа имеет вид

$$A_p^{(1)} = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{array} \begin{array}{c} \begin{array}{ccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \end{array} \\ \begin{pmatrix} 0 & 3 & \infty & \infty & \infty & 9 & \infty \\ 3 & 0 & 1 & \infty & \infty & \infty & \infty \\ \infty & 1 & 0 & 2 & \infty & \infty & \infty \\ \infty & \infty & 2 & 0 & 5 & 10 & \infty \\ \infty & \infty & \infty & 5 & 0 & 1 & \infty \\ 9 & \infty & \infty & 10 & 1 & 0 & 5 \\ \infty & \infty & \infty & \infty & \infty & 5 & 0 \end{pmatrix} \end{array}.$$

Такая матрица соответствует кратчайшим путям, содержащим только одно ребро.

Отметим, что кратчайший путь из двух ребер между вершинами графа *u* и *w* проходит еще в точности через одну вершину. Например, на рис. 3.4 всякий путь из двух ребер между вершинами x_1 и x_5 может проходить через одну из вершин x_2, x_3, x_4, x_6, x_7. Если проанализировать сумму весов ребер (x_1, x_k) и (x_k ,x_5), где k=2,3,4,6,7, то минимальное значение суммы весов соответствует длине кратчайшего пути из двух ребер (x_1, x_6) и (x_6, x_5). Поэтому элементы матрицы $A_p^{(2)}$, позволяющие определить длину кратчайшего пути из двух или менее ребер для графа из n вершин, можно определить по формуле

$$\begin{aligned} &a_{ij}^{(2)} = \min(a_{ij}^{(1)}, a_{ik}^{(1)} + a_{kj}^{(1)}),\ k = 1,...,n\ , \\ &i \neq j\ , a_{ii}^{(2)} = 0. \end{aligned} \qquad (3.1)$$

Таким образом, для графа на рис.3.4 получим

$$A_p^{(2)} = \begin{pmatrix} 0 & 3 & 4 & 19 & 10 & 9 & 14 \\ 3 & 0 & 1 & \infty & \infty & \infty & \infty \\ 4 & 1 & 0 & 2 & 7 & 12 & \infty \\ 19 & \infty & 2 & 0 & 5 & 6 & 15 \\ 10 & \infty & 7 & 5 & 0 & 1 & 6 \\ 9 & \infty & 12 & 6 & 1 & 0 & 5 \\ 14 & \infty & \infty & 15 & 6 & 5 & 0 \end{pmatrix}.$$

Пусть теперь $A_p^{(k)}$ - матрица кратчайших путей, содержащих не более k ребер графа. Обобщая формулу (3.1), для вычисления элементов матрицы кратчайших путей получим итерационную формулу

$$A_p^{(k+1)}[i,j] = \min\left(A_p^{(k)}[i,j],\ A_p^{(k)}[i,k] + A_p^{(k)}[k,j]\right), \qquad (3.2)$$
$$k = \overline{1,n-1},\ i \neq j,\ A_p^{(k+1)}[i,i] = 0, i.j = \overline{1,n}.$$

По формуле (3.2) осуществляется выбор меньшего из двух путей: содержащего вершину с индексом k и не содержащего ее.

В рассмотренном алгоритме необходимо вычислить $n-1$ матрицу $A_p^{(2)}, \ldots, A_p^{(n)}$, каждая из которых состоит из n^2 элементов, требующих для своего вычисления двух операций (сложение и сравнение). Поэтому алгоритм Флойда реализуется за $O(n^3)$ операций, а используя матрицу $A_p^{(1)}$, итерационную формулу (3.2) можно реализовать тремя вложенными циклами :

```
for k = 1 to n-1
  for i = 1 to n
    for j = 1 to n
      A[i,j] = min(A[i,j], A[i,k] + A[k,j]).
```

При увеличении количества вершин графа G_n в два раза время расчетов по алгоритму Флойда возрастает в восемь раз.

Известно, что если $A = \left[a_{ij}^1 \right]_l^n$ - квадратная матрица, то элементы матрицы A^2 находятся по формуле

$$a_{ij}^{(2)} = \sum_{k=1}^{n} a_{ik}^{(1)} a_{kj}^{(1)}\ ,\ \text{i , j = 1,…,n.} \qquad (3.3)$$

Заменим в этом алгоритме умножения матриц операцию сложения операцией взятия минимума, а операцию умножения сложением. Тогда из (3.3) получим формулы (3.1), вычисляющие кратчайшие пути, содержащие не более двух ребер. Поэтому матрицу $A_p^{(3)}$ можно построить по матрицам $A_p^{(2)}$ и $A_p^{(1)}$, так как кратчайший путь из трех или менее ребер должен состоять из кратчайшего пути из двух или менее ребер, за которыми следует кратчайший путь из одного или менее ребер. Матрицу $A_p^{(4)}$ можно получить либо по матрицам $A_p^{(3)}$и $A_p^{(1)}$, либо только по матрице $A_p^{(2)}$. Тогда очевидно, что последовательность вычисления степеней матриц $A_p^{(k)}$, k=2, ... ,n можно заменить вычислением матриц четных степеней:$A_p^{(2)}$, $A_p^{(4)}$, $A^{(8)}$, ,…,$A_p^{(N)}$, где N – наи-

меньшее 2^S, которое превышает $n-1$, учитывая, что у графа с n вершинами кратчайший путь может содержать не более $n-1$ ребра. Поэтому все кратчайшие расстояния в графе являются элементами матрицы $A_p^{(N)}$. Использование вычисления только четных степеней матриц $A_p^{(2)}$, $A_p^{(4)}$, $A^{(8)}$, ,…,$A_p^{(N)}$ позволяет в алгоритме Флойда почти в два раза уменьшить количество операций, которые требуются при применении соответствующего алгоритма Дейкстры.

Для неориентированного графа матрицы $A_p^{(i)}$ являются симметричными, поэтому достаточно вычислять либо верхнюю либо нижнюю треугольную часть этих матриц.

Приведенный выше алгоритм находит только длины кратчайших путей, но не позволяет определить последовательность вершин, через которые они проходят.

Однако эти вершины можно найти с помощью записи информации о самих путях (наряду с информацией о длинах путей). В этой модификации алгоритма в дополнение к матрицам $A^{(k)}{}_p$ хранится и обновляется вспомогательная матрица $B^{(k)}[b_{ij}]$. Элемент b_{ij} указывает индекс вершины, непосредственно предшествующей вершине x_j в кратчайшем пути от x_i к x_j. Если вершины графа обозначены целыми числами, то элемент b_{ij} является просто номером предшествующей вершины в кратчайшем пути от вершины i к вершине с номером j.

Элементам матрицы $B^{(1)}$ присваиваются начальные значения $b_{ij} = i$ для всех i и j. Обновление матрицы $B^{(k)}$ происходит одновременно с вычислением матриц $A^{(k)}{}_p$ следующим образом :

$$b_{ij} = \begin{cases} b_{kj}, \text{если } (a_{ik} + a_{kj}) < a_{ij}, \\ \text{не изменяется, если } (a_{ik} + a_{kj}) \geq a_{ij}. \end{cases}$$

Вершины кратчайших путей определяются непосредственно из заключительной матрицы $B^{(N)}$ после вычисления $A^{(N)}{}_p$. Тогда кратчайший путь между двумя вершинами x_i и x_j соответствует следующей последовательности индексов вершин:

$$i, \nu_n, ..., \nu_2, \nu_1, j,$$

где $v_1 = b_{ij}, v_2 = b_{iv_1}, v_3 = b_{iv_2}$,…, $i = b_{iv_n}$.

В качестве примера рассмотрим граф, изображенный на рис. 3.4. В результате расчетов по алгоритму Флойда получим матрицу кратчайших путей (s=3) и вспомогательную матрицу:

$$A_p^{(8)} = \begin{pmatrix} 0 & 3 & 4 & 6 & 10 & 9 & 14 \\ 3 & 0 & 1 & 3 & 8 & 9 & 14 \\ 4 & 1 & 0 & 2 & 7 & 8 & 13 \\ 6 & 3 & 2 & 0 & 5 & 6 & 11 \\ 10 & 8 & 7 & 5 & 0 & 1 & 6 \\ 9 & 9 & 8 & 6 & 1 & 0 & 5 \\ 14 & 14 & 13 & 11 & 6 & 5 & 0 \end{pmatrix} \qquad B^{(8)} = \begin{pmatrix} 1 & 1 & 2 & 3 & 6 & 1 & 6 \\ 1 & 2 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 3 & 3 & 4 & 5 & 6 \\ 1 & 2 & 3 & 4 & 4 & 5 & 6 \\ 1 & 2 & 3 & 4 & 5 & 5 & 6 \\ 1 & 3 & 3 & 4 & 5 & 6 & 6 \\ 1 & 1 & 4 & 4 & 5 & 6 & 7 \end{pmatrix}.$$

Используя эти матрицы, определим, например, последовательность вершин кратчайшего пути длины 6 из вершины x_1 в вершину x_4. Из матрицы $B^{(8)}$ восстановим индексы вершин этого кратчайшего пути по алгоритму, описанному выше:

$$\nu_1 = b_{14} = 3\,,\ \nu_2 = b_{1\nu_1} = b_{13} = 2\,, \nu_3 = b_{1\nu_2} = b_{12} = 1\,.$$

Таким образом, этот путь проходит через вершины x_2 и x_3.

Медиана - это вершина графа, у которой сумма кратчайших расстояний от неё до остальных вершин графа минимально возможная. Поиск медианы графа связан с задачами оптимального размещения пунктов обслуживания. Для рассматриваемого графа, как это следует из анализа суммы строк матрицы A_p^8, медианой является вершина x_4.

Рассмотрим теперь орграф из рис.3.5.

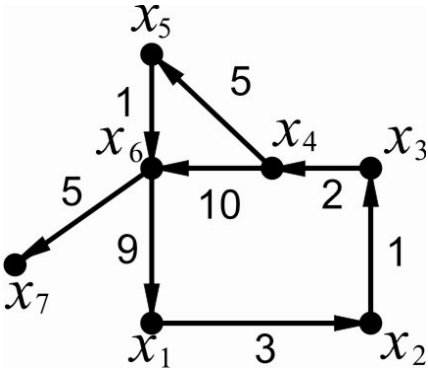


Рис. 3.5. Взвешенный орграф для n=7

Для этого графа искомые матрицы имеют вид:

$$A_p^{(8)} = \begin{pmatrix} 0 & 3 & 4 & 6 & 11 & 12 & 17 \\ 18 & 0 & 1 & 3 & 8 & 9 & 14 \\ 17 & 20 & 0 & 2 & 7 & 8 & 13 \\ 15 & 18 & 19 & 0 & 5 & 6 & 11 \\ 10 & 13 & 14 & 16 & 0 & 1 & 6 \\ 9 & 12 & 13 & 15 & 20 & 0 & 5 \\ \infty & \infty & \infty & \infty & \infty & \infty & 0 \end{pmatrix} B^{(8)} = \begin{pmatrix} 1 & 1 & 2 & 3 & 4 & 5 & 6 \\ 6 & 2 & 2 & 3 & 4 & 5 & 6 \\ 6 & 1 & 3 & 3 & 4 & 5 & 6 \\ 6 & 1 & 2 & 4 & 4 & 5 & 6 \\ 6 & 1 & 2 & 3 & 5 & 5 & 6 \\ 6 & 1 & 2 & 3 & 4 & 6 & 6 \\ 7 & 7 & 7 & 7 & 7 & 7 & 7 \end{pmatrix}.,$$

Например, кратчайший путь из вершины x_1 в вершину x_7 длины 17 соответствует индексам промежуточных вершин:

$$\nu_1 = b_{17} = 6,\ \nu_2 = 5,\ \nu_3 = 4, \nu_4 = 3,\ \nu_5 = 2, \nu_6 = 1.$$

Таким образом, кратчайший путь содержит промежуточные вершины x_2, x_3, x_4, x_5, x_6. Отметим, что элементы $B^{(8)}[7,j]=7$, так как вершина x_7 является стоком (пути из x_7 не существует).

Заметим, что у графа с n вершинами $O(n^2)$ путей, каждый из которых содержит $O(n)$ вершин. Вычисление вспомогательной матрицы $B^{(N)}$ позволяет на порядок сократить объем хранимой о путях информации [13].

3.4. Параллельный алгоритм Флойда

Как следует из общей схемы алгоритма Флойда, основная вычислительная нагрузка при решении задачи поиска кратчайших путей состоит в выполнении операции выбора минимальных значений и номеров предшествующих вершин. Очевидно, что алгоритм умножения матриц (3.4) легко модифицируется в алгоритм подсчета кратчайших расстояний, соответствующих матрице $A_p^{(N)}$. Поэтому для $N>>1$ подсчет кратчайших расстояний в графе можно реализовать на основе известных параллельных алгоритмов вычисления произведения двух матриц [38−40]. В качестве фундаментальной подзадачи на k-той итерации можно выбрать вычисление элементов a_{ij} матрицы путей из (3.2), так как для любой пары индексов (i,j) на k-той итера-

ции элементы $a_{ik}^{(k+1)}$ и $a_{kj}^{(k+1)}$ не изменяются. Поэтому если для q процессоров выделить n^2/q укрупненных подзадач, то за n итераций потребуется выполнить n^3/q операций. Тогда для ускорения и эффективности алгоритма Флойда получим

$$S_q = \frac{n^3}{\left(n^3/q\right)} = q, \quad E_q = \frac{E_q}{q} = 1.$$

Известен параллельный алгоритм, основанный на использовании блочной схемы разбиения матрицы примыканий $A^{(k)}{}_p$ и вспомогательной матрицы $B^{(k)}$. Такой подход предполагает горизонтальное или вертикальное разбиение матриц на ленты (блоки) одинаковой размерности. Рассмотрим вариант разбиения матриц на вертикальные полосы, что соответствует алгоритмическим языкам, у которых элементы матриц хранятся по столбцам. При таком способе разбиения данных на каждой итерации алгоритма Флойда потребуется передавать между процессорными элементами (ПЭ) только элементы одного из блоков матриц $A_p^{(k)}$ и $B^{(k)}$. Параллельный алгоритм для графа из n вершин состоит из следующих этапов:

1. Определение целого N, где N – наименьшее 2^S, которое превышает $n-1$. Величина s соответствует количеству произведений матриц из последовательности

$$A_p^{(1)} \bullet A_p^{(1)}, A_p^{(2)} \bullet A_p^{(2)}, \cdots, A_p^{(N/2)} \bullet A_p^{(N/2)} .$$

2. Задание размерности n, элементов матрицы примыканий $A_p{}^{(1)}$, вспомогательной матрицы $B^{(1)}$.Задание номеров и числа ПЭ: $ПЭ_0,\ldots$, $ПЭ_{q-1}$; $n=L \cdot q$, L − количество полос.

3. $ПЭ_0$ пересылает всем остальным матрицу $A_p{}^{(1)}$ и вспомогательную матрицу $B^{(1)}$.

4. Вычисление элементов соответствующей вертикальной полосы матрицы $A_p{}^{(2i)}$ и обновление вертикальной полосы вспомогательной матрицы $B^{(2i)}$.

5. Каждый ПЭ отправляет свою рассчитанную полосу матриц $A_p{}^{(2i)}$ и $B^{(2i)}$ всем остальным ПЭ и заполняет недостающие полосы данными, поступающими от других процессоров.

6. Выполняются пункты 4−5, пока количество итераций не превысит рассчитанное число *s*.

По закону Амдаля [22] для ускорения алгоритма, полученного при выполнении на идеальной параллельной машине из q процессоров (без учета обменов), имеет место формула

$$S_q = \frac{1}{r + (1-r)/q},$$

где r – доля операций последовательной части алгоритма.

Реальное ускорение на практике можно определить как отношение процессорного времени на выполнение последовательной программы к времени выполнения вычислений параллельной программой на q-процессорной машине. При этом предполагается, что обе программы реализуют один и тот же алгоритм.

В табл. 3.2 приведено время вычисления по параллельному алгоритму Флойда для различных значений числа вершин и используемых процессоров для кластера *СКИФ Cyberia* [41].

Таблица 3.2

Зависимость времени расчета от числа вершин графа и числа процессоров

n	Время работы алгоритма					
	q=10	S_q	q=50	S_q	q=100	S_q
500	0,44	8,5	0,26	14,4	0,32	12,5
1000	6,34	8,6	2,78	19,7	1,89	28,9
3000	268,61	9,6	59,06	43,8	32,25	80,2

Как следует из табл. 3.2, при больших n с ростом числа процессоров время расчетов существенно сокращается.

3.5. Параллельный алгоритм нахождения коэффициентов характеристического многочлена графа

Рассмотрим один из параллельных алгоритмов нахождения коэффициентов характеристического многочлена графа

$$P_G(\lambda) = \det(A - \lambda E) = (-1)^n[\lambda^n + c_1\lambda^{n-1} + \ldots + c_n], \tag{3.4}$$

когда $n>>1$.

Сумма диагональных элементов квадратной матрицы A называется *следом матрицы*

$$Sp(A) = \sum_{i=1}^{n} a_{ii}.$$

Характеристический многочлен $P_G(\lambda)$ можно построить методом Леверье, используя понятие следа матрицы [28]. Этот метод основан на формулах Ньютона для сумм степеней корней характеристического уравнения. Пусть λ_i, $i = \overline{1,n}$, – спектр графа, тогда λ_i^k, $i = \overline{1,n}$ являются собственными значениями матрицы A^k. Обозначим

$$S_1 = \lambda_1 + \lambda_2 + \ldots + \lambda_n = SpA,$$
$$S_2 = \lambda_1{}^2 + \lambda_2{}^2 + \ldots + \lambda_n{}^2 = SpA^2,$$
$$\ldots\ldots\ldots\ldots\ldots\ldots$$
$$S_n = \lambda_1{}^n + \lambda_2{}^n + \ldots + \lambda_n{}^n = SpA^n,$$

тогда при $k \le n$ справедливы соотношения Ньютона:

$$S_k + c_1 S_{k-1} + \ldots + c_{k-1} S_1 = -kc_k. \tag{3.5}$$

Из (3.5) получим:

$c_1 = -S_1$. если $k = 1$,

$c_2 = -\frac{1}{2}(S_2 + c_1 S_1)$, если $\kappa = 2$,

……

$c_n = -\frac{1}{n}(S_n + c_1 S_{n-1} + c_2 S_{n-2} + \ldots + c_{n-1} S_1)$, *если* $\kappa = n$.

Следовательно, коэффициенты $c_i, i = \overline{1,n}$ характеристического многочлена (3.4) можно определить, если известны следы степеней матрицы A (суммы: $S_1, S_2, \ldots, S_n$.). Вычисление степеней и следов матрицы смежности A можно осуществить в параллельном режиме.

Возведение матрицы A размерности n в степень n выполняется за $2n^4$ арифметических операций, нахождение следов матриц требует еще n^2-1 операций, вычисление коэффициентов характеристического многочлена по методу Леверье осуществляется за n^2 операций. Таким образом, сложность этого алгоритма $O(n^4)$.

По рассмотренному параллельному алгоритму получим характеристический многочлен замкнутой фуллереновой структуры C_{20}. Она соответствует графу додекаэдра, плоское представление которого (диаграмма Шлегеля) представлено на рис. 3.7

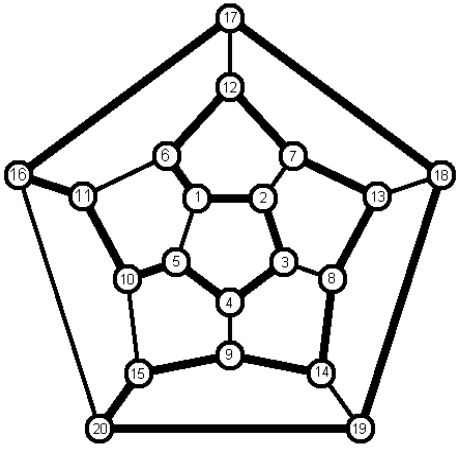


Рис. 3.7 Плоский граф додекаэдра

Матрица смежности этого графа

$$A_p(C_{20}) = \begin{pmatrix}
0&1&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&1&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&1\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&1\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&1&0
\end{pmatrix}.$$

Характеристический многочлен графа, вычисленный по параллельному методу Леверье, имеет вид

$$P_{C_{20}}(\lambda) = \lambda^{20} - 29\lambda^{18} - \lambda^{17} + 349\lambda^{16} + \lambda^{15} - 2267\lambda^{14} + 208\lambda^{13} + 8623\lambda^{12} - 2126\lambda^{11} -$$
$$-19299\lambda^{10} + 8775\lambda^{9} + 23439\lambda^{8} - 16699\lambda^{7} - 10965\lambda^{6} + 12490\lambda^{5} - 1900\lambda^{4} - 600\lambda^{3}$$

Коэффициент многочлена $c_1=0$, так как след матрицы смежности всегда равен нулю.

Замечание 1. По теореме Гамильтона − Кэли невырожденная матрица A удовлетворяет своему характеристическому многочлену, поэтому из (3.4)

$$A^{-1} = -\frac{1}{c_n}\left(A^{n-1} + c_1 A^{n-2} + \ldots + c_{n-1}E\right), \qquad (3.6)$$

где E – единичная матрица.

Вычисляя правую часть (3.6) по уже найденным степеням матрицы A по описанному выше параллельному алгоритму, можно определить элементы обратной матрицы.

Замечание 2. Одной из важных характеристик матрицы является число обусловленности $cond(A)$, которое характеризует влияние погрешности в коэффициентах матрицы A на погрешность решения системы линейных уравнений $A\bar{x} = \bar{b}$. Для симметричных матриц его можно определить по формуле

$$cond(A) = \frac{\max_i |\lambda_i|}{\min_i |\lambda_i|} \geq 1.$$

Для этих целей можно воспользоваться теоремой Гершгорина и параллельным алгоритмом сплайновой интерполяции при нахождении собственных чисел [42, 43].

Замечание 3. Нахождение спектра матрицы существенно зависит от точности определения коэффициентов характеристического многочлена из (3.4).

3.6. О разделении графа на домены

Для решения задач на сеточных графах большой размерности с помощью многопроцессорных вычислительных систем часто используют декомпозицию расчетной области, в результате которой сеточный граф большой размерности разделяется на домены (подграфы, части сетки). Это связано с необходимостью сбалансирования нагрузки для выделенных q процессоров.

Объем передачи данных между процессорами зависит от числа ребер (веса сечения), соединяющих вершины, принадлежащие разным доменам, распределенным по процессорам. Для разделения графов используются следующие геометрические методы [45, 46]: покоординатное разбиение, рекурсивный инерционный метод деления пополам, деление сети с использованием кривых Пеано. Эти методы основываются на координатной информации об узлах графа.

Разделение графа методом спектральной бисекции предполагает, что каждой I – той вершине графа ставится в соответствие компонента x_i вектора $\bar{x}$, равная +1 или –1, таким образом, что

$$\sum_{i=1}^{n} x_i = 0, \quad \sum_{i=1}^{n} x_i^n = n.$$

При этих ограничениях минимизируется квадратичная форма

$$(L(G)\overline{x},\overline{x}) = \sum_{(i,k)\in E} (x_i - x_k)^2 .$$

Приближенное решение этой задачи сводится к определению собственного вектора, соответствующего максимальному ненулевому собственному значению спектральной матрицы Лапласа L, которое Фидлер назвал индексом связности графа [47–49]. Этот метод минимизирует суммарный вес ребер, соединяющих вершины из двух разных доменов V_1 , V_2. При этом для четных n

$$V_1 \cup V_2 = \varnothing,\ n_1 + n_2 = n, \left|V_1\right| = \left|V_2\right|.$$

Метод спектральной бисекции позволяет, используя алгоритм рекурсивной бисекции (*Recursive Spectral Bisection, RSB*), разделить граф на произвольное число частей.

Рассмотрим основные этапы этого метода, например, для графа из рис. 3.8

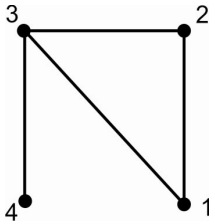


Рис. 3.8 Связный граф для n=4

1. Сформируем матрицу Лапласа для этого графа

$$L = \begin{pmatrix} -2 & 1 & 1 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & 1 & -3 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix}.$$

Эта матрица отрицательно полуопределенная, имеет наибольшее собственное значение $\lambda_1 = 0$, симметричная с действительными собственными числами.

2. Решение системы $L\bar{x} = \lambda\bar{x}$ дает собственные значения этой матрицы $\lambda_{1..4} = \{0, -1, -3, -4\}$. Собственный вектор Фидлера $\bar{x}_2$, соответствующий максимальному ненулевому собственному значению λ_2 матрицы L, состоит из весовых множителей, связанных с вершинами графа. В данном случае $\lambda_2 = -1$, $\bar{x}_2 = (-0.408, -0.408, 0, 0.816)$.

3. Осуществляется сортировка множества номеров вершин графа по не убыванию значений компонент вектора Фидлера. В рассматриваемом примере порядок номеров вершин соответствует компонентам вектора S= (1, 2, 3, 4).

4. Используя вектор S, можно распределить вершины графа в два домена V_1, V_2 так, что

$$d = n_1 / n_2,\ V_1 \cup V_2 = \varnothing,\ n_1 + n_2 = n.$$

Для d=1 в первый домен попадают вершины (1,2), а во второй – (3,4), что соответствует оптимальному разбиению графа на две равные части, представленному на рис.3.9.

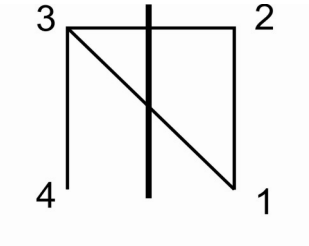


Рис. 3.9 Разделение графа на два домена

Рассмотрим теперь граф из рис. 3.10

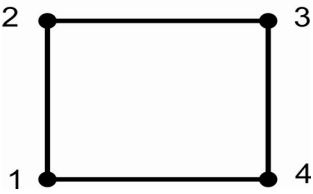
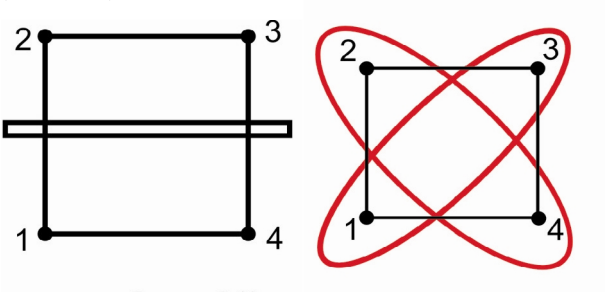


Рис. 3.10 Граф C_4

Собственные значения его матрицы Лапласа $\lambda_{1..4} = \{0, -2, -2, -4\}$. Рассмотрим два случая разделения этого графа. На рис. 3.10a показано разделение графа для λ_2= -2, S={4, 1, 3, 2}, а на рис.3.10b для λ_4= -4, S={2, 4, 1, 3}



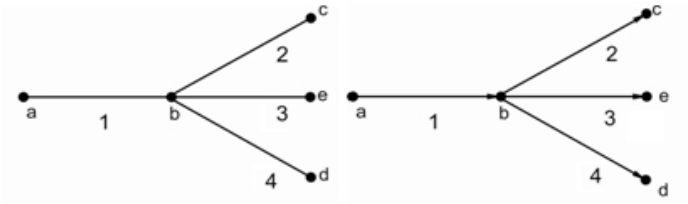
a b

Рис. 3.10 Разделение графа на домены (a : λ_2= -2, b: λ_4= -4)

Как следует из этих рисунков, во втором случае два домена связаны уже не двумя, а четырьмя ребрами. Таким образом, вектор Фидлера действительно минимизирует связи между доменами.

3.7. Математическое моделирование теплообмена в стержневых системах

При проектировании различных сетевых объектов часто возникает задача оценки влияния отдельных элементов оборудования на тепловое состояние всей системы. Рассмотрим звездный граф $K_{1,4}$ из рис. 3.11a, соответствующий конструкционному элементу из четырёх тонких стержней с изолированной боковой поверхностью и различными теплофизическими свойствами [50–52].



a b

Рис. 3.11 Конструкционный элемент $K_{1,4}$

Требуется определить тепловое состояние системы в различные моменты времени t при условиях:

1) в точке b идеальный тепловой контакт;

2) в точках a, c, d, e осуществляется теплообмен по закону Ньютона с соответствующими коэффициентами теплоотдачи $\alpha_1, \alpha_2, \alpha_3, \alpha_3$;

3) начальная температура элемента T_n, а температура внешней среды T_e.

Неориентированному графу из рис 3.11a поставим в соответствие ориентированный граф из рис. 3.11b. Тогда с математической точки зрения необходимо решить следующую краевую задачу:

$$\frac{\partial u_m}{\partial t} = \beta_m^2 \frac{\partial^2 u_m}{\partial x_m^2}, \quad m = \overline{1,4};$$

$$\lambda_1 \frac{\partial u_1}{\partial x_1}\Big|_{x_1=a} = \alpha_1 (u_1\big|_{x_1=a} - T_e), \ \lambda_2 \frac{\partial u_2}{\partial x_2}\Big|_{x_2=c} = \alpha_2 (T_e - u_2\big|_{x_2=c});$$

$$\lambda_3 \frac{\partial u_3}{\partial x_3}\Big|_{x_3=d} = \alpha_3 (T_e - u_3\big|_{x_3=d}), \ \lambda_4 \frac{\partial u_4}{\partial x_4}\Big|_{x_4=e} = \alpha_4 (T_e - u_4\big|_{x_4=e});$$

$$\lambda_1 \frac{\partial u_1}{\partial x_1}\Big|_{x_1=b} = \lambda_2 \frac{\partial u_2}{\partial x_2}\Big|_{x_2=b} + \lambda_3 \frac{\partial u_3}{\partial x_3}\Big|_{x_3=b} + \lambda_4 \frac{\partial u_4}{\partial x_4}\Big|_{x_4=b} \qquad (3.7)$$

$$u_1(b,t) = u_2(b,t) = u_3(b,t) = u_4(b,t);$$

$$u_m(x,t)\Big|_{t=0} = T_n, \ m = \overline{1,4},$$

$$x_1 \in [a,b], \ x_2 \in [b,c], \ x_3 \in [b,d], \ x_4 \in [b,e], \ t \in (0,T],$$

где m - номер стержня, t - время, $\beta_m^2 = \lambda_m / (\rho_m c_m)$, λ_m, ρ_m, c_m, α_m – заданные параметры.

Введём в рассмотрение сеточный граф ω:

$$\omega_m = \{x_{mi} = i \cdot h_m, i = \overline{0, N_m}\}, m = \overline{1,4};$$
$$h_1 = (b-a)/N_1,\ h_2 = (c-b)/N_2, h_3 = (e-b)/N_3,$$
$$h_4 = (d-b)/N_4, \qquad \omega = \omega_1 \cup \omega_2 \cup \omega_3 \cup \omega_4$$

и сквозную нумерацию узлов сетки, представленную на рис. 3.13,

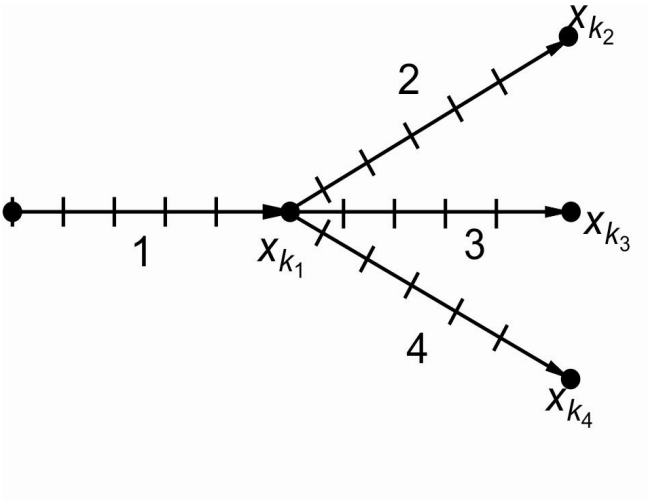


Рис. 3.13 Сеточный граф вторичной топологии

где

$$K_1 = N_1,\ K_2 = K_1 + N_2,\ K_3 = K_2 + N_3,\ K_4 = K_3 + N_4.$$

Тогда для рис. 3.11 получим разностную схему на графе вторичной топологии:

$$\lambda_1 \frac{\hat{u}_1 - \hat{u}_0}{h_1} = \alpha_1(\hat{u}_0 - T_e);$$

$$\lambda_1 \frac{\hat{v} - \hat{u}_{N-1}}{h_1} = \lambda_2 \frac{\hat{u}_{K_1+1} - \hat{v}}{h_2} + \lambda_3 \frac{\hat{u}_{K_2+1} - \hat{v}}{h_3} + \lambda_4 \frac{\hat{u}_{K_3+1} - \hat{v}}{h_4};$$

$$\frac{\hat{u}_i - u_i}{\tau} = a_1^2 \Lambda \hat{u}_i, i = \overline{1, K_1 - 1}, \quad \hat{v} = \hat{u}_{1N} = \hat{u}_{20} = \hat{u}_{30} = \hat{u}_{40};$$

$$\lambda_2 \frac{\hat{u}_{K_2} - \hat{u}_{K_2-1}}{h_2} = \alpha_2(T_e - \hat{u}_{K_2}); \qquad (3.8)$$

$$\frac{\hat{u}_i - u_i}{\tau} = a_2^2 \Lambda \hat{u}_i, i = \overline{K_1 + 1, K_2 - 1}, \; \lambda_3 \frac{\hat{u}_{K_3} - \hat{u}_{K_3-1}}{h_3} = \alpha_3(T_e - \hat{u}_{K_3});$$

$$\frac{\hat{u}_i - u_i}{\tau} = a_3^2 \Lambda \hat{u}_i, i = \overline{K_2 + 1, K_3 - 1},$$

$$\lambda_4 \frac{\hat{u}_{K_4} - \hat{u}_{K_4-1}}{h_4} = \alpha_4(T_e - \hat{u}_{K_4}),$$

$$\frac{\hat{u}_i - u_i}{\tau} = a_4^2 \Lambda_4 \hat{u}_i, i = \overline{K_3 + 1, K_4 - 1},$$

где $\Lambda \hat{u}_i = \dfrac{u_{i-1}^{n+1} - 2u_i^{n+1} - u_{i+1}^{n+1}}{h_k^2}$, $u_i^n = u(x_i, n \cdot \tau)$, $k = \overline{1,4}$;

$\hat{u} = u^{n+1}, u = u^n, n$ - номер слоя, τ - шаг по времени, $t_n = n\tau$.

Система неявных разностных уравнений (3.8) аппроксимирует задачу (3.7) с порядком $O(\tau, h)$ и имеет матрицу при неизвестных следующей структуры:

$$\begin{pmatrix}
\times & \times & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\
\times & \times & \times & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\
0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & \cdots & 0 \\
0 & 0 & \times & \times & \times & 0 & \cdots & 0 & \times & 0 & \cdots & 0 & \times & 0 & \cdots & 0 \\
0 & 0 & 0 & \times & \times & \times & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots & \cdots & 0 \\
\vdots & \vdots & \vdots & 0 & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & 0 & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & 0 & 0 & 0 & \vdots & \vdots & \cdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \times & \times & \times & 0 & 0 & \vdots & \vdots & \cdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 & \times & \times & 0 & \vdots & \vdots & \cdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \times & \times & \times & 0 & \vdots & \cdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & 0 & \cdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \times & \times & \times & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 & \times & \times & \times \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \times & \times
\end{pmatrix}$$

Эта матрица имеет обратную, но не является трёхдиагональной из-за уравнения, стоящего на строчке N_1.

Для решения системы (3.8) на каждом ребре графа воспользуемся формулами метода левой или правой прогонки [43]:

$$\begin{aligned}
&\hat{u}_i = P_{1i} \cdot \hat{u}_{i+1} + Q_{1i},\ i = \overline{0, K_1 - 1}\ , \\
&\hat{u}_i = P_{2i} \cdot \hat{u}_{i+1} + Q_{2i},\ i = K_2 - 1, \ldots, K_1\ , \\
&\hat{u}_i = P_{2i} \cdot \hat{u}_{i+1} + Q_{2i},\ i = K_3 - 1, \ldots, K_2 + 1, \\
&\hat{u}_i = P_{3i} \cdot \hat{u}_{i+1} + Q_{3i},\ i = K_4 - 1, \ldots, K_3 + 1,
\end{aligned} \qquad (3.9)$$

где P_{mi} , Q_{mI}, – прогоночные коэффициенты.

Определив необходимое количество прогоночных коэффициентов, из разностного уравнения в точке *b,* получим формулу для вычисления температуры $\hat{v}$ на слое n+1. Зная решение в этой точке, по формулам (3.9) определяется решение во всех узлах сеточного графа для $t = n\tau$, $n = 1, \ldots, M$, τ $M = T$.

Если число ребер графа $m>>1$ и $K_4>>1$, то для решения соответствующей системы неявных разностных уравнений можно использовать разделение графа на домены и параллельные варианты метода прогонки или итерационные методы решения системы (3.8) на каждом слое по времени.

Научное издание

Владимир Николаевич БЕРЦУН

МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ НА ГРАФАХ

Часть II

Редактор В.С. Сумарокова
Компьютерная верстка Т.В. Дьяковой

Подписано в печать 11.11.2013.
Формат 60x84 1/16. Бумага офсетная № 1. Печать офсетная.
Печ. л.5,25; усл. печ. л.4,9; уч.-изд. л.4,5. Тираж 500. Заказ

ООО «Издательство ТГУ», 634029, г. Томск, ул. Никитина, 4
ООО «Интегральный переплет», 634040, г. Томск, ул. Высоцкого, 28, стр. 1