Параллельные алгоритмы на графах

В.Н. Берцун, А.В. Бородин
Томский государственный университет

Рассматриваются последовательный и параллельный алгоритмы Флойда поиска кратчайших путей и количества путей в графе. Приводятся оценки их эффективности на основе результатов вычислительного эксперимента на кластере СКИФ Cyberia.

Пусть дан ориентированный граф $G(V, E)$ на *n* вершинах и его матрица примыканий [1]

$$A^{(1)}{}_p = \begin{cases} w_{ij}, \text{ при } x_i x_j \in E, \\ 0, \text{при } i = j, \text{где } w_{ij} \text{ - вес ребра } x_i \text{из в } x_j \\ \infty \text{ при } x_i x_j \notin E. \end{cases}$$

Матрицу $A_p^{(2)}$ по матрице примыканий $A_p^{(1)}$ можно построить, находя элементы $A_p^{(2)}$ по формуле $a_{i,j}^{(2)} = \min(a_{ij}^{(1)}, a_{ik}^{(1)} + a_{kj}^{(1)}), k = \overline{1,n}$. Последовательно вычисляя $A_p^{(i)}, 2 \le i \le N$, где N – наименьшее 2^S (S – натуральное), $2^s < n–1$, получим матрицу кратчайших путей $A_p^{(N)}$.

По матрице $A_p^{(N)}$ можно найти также и некоторые другие характеристики графа: эксцентриситет его вершин, радиус и диаметр графа.

Вершины, через которые проходят кратчайшие пути, определяются с помощью записи информации о самих путях (наряду с информацией о длинах путей) в матрице $B = [b_{ij}]$. Элемент b_{ij} указывает вершину, непосредственно предшествующую вершине x_i в кратчайшем пути от x_i к x_j. Элементам матрицы B присваиваются начальные значения $b_{ij} = i$ для всех i и j.

Обновление матрицы B происходит следующим образом:

$$b_{ij} = \begin{cases} b_{ij}, \textit{если } (a_{ik} + a_{kj}) < a_{ij}, \\ \textit{неизменяется, если } (a_{ik} + a_{kj}) \ge a_{ij}. \end{cases}$$

Вершины кратчайших путей получаются непосредственно из заключительной матрицы B после вычисления $A^{(N)}{}_p$. Тогда кратчайший

путь между двумя вершинами x_i и x_j дается следующей последовательностью вершин [2,3]:

$$x_i, x_n, ..., x_2, x_1, x_j \ ,$$

где $x_1 = b_{ij}, x_2 = b_{ix_1}, x_3 = b_{ix_2}$ и т. д. до $x_i = b_{ix_n}$.

Как правило, число доступных процессоров p существенно меньше, чем число базовых задач m ($p<<m$). Возможный способ укрупнения вычислений состоит в использовании ленточной схемы разбиения матрицы примыканий A и вспомогательной матрицы B. Такой подход соответствует объединению в рамках одной базовой подзадачи вычислений, связанных с определением элементов одной или нескольких строк (горизонтальное разбиение) или столбцов (вертикальное разбиение) матриц A и B. Так как для алгоритмического языка Fortran массивы располагаются по столбцам, использовалось разбиение матриц A и B на вертикальные полосы. При таком способе разбиения данных на каждой итерации алгоритма Флойда потребуется передавать между процессорами только элементы одного из столбцов матриц A и B.

Ускорение алгоритма, полученное при выполнении на идеальной параллельной машине из p процессоров (без учета обменов) :

$$S_p = \frac{1}{S + (1 - S)/p},$$

где S – доля операций последовательной части алгоритма [6 – 8].

Ускорение можно определить и как отношение процессорного времени на выполнение последовательной программы ко времени выполнения вычислений параллельной программой на p-процессорной машине. При этом предполагается, что обе программы реализуют один и тот же алгоритм.

Формулировка закона Амдаля не учитывает временных затрат на обеспечение обмена информацией между вычислительными узлами. Поскольку эти затраты, обусловленные необходимостью выполнения коммуникационных обменов, предотвращения конфликтов памяти и синхронизации, лишь увеличивают общее время работы параллельной программы, то реальное ускорение будет всегда меньше идеального.

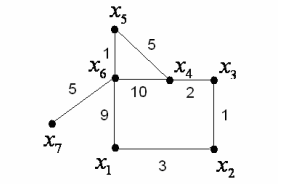


Рис. 1. Неориентированный граф

Для неориентированного графа из рис. 1 в результате расчетов по алгоритму Флойда получены матрица кратчайших путей и вспомогательная матрица:

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

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

$x_1^p = b_{14} = 3,\ x_2^p = b_{1x_1^p} = b_{13} = 2,\ x_3^p = b_{1x_2^p} = b_{12} = 1$.

Таким образом, кратчайший путь проходит через вершины x_2^p, x_1^p, т.е. через вершины x_2 и x_3.

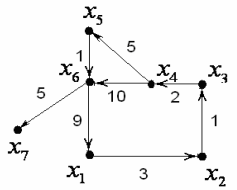


Рис. 2. Ориентированный граф

В случае ориентированного графа из рис. 2 искомые матрицы имеют вид

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

Например, кратчайший путь из вершины x_1 в вершину x_7 по этим матрицам определится так:

$$x_1^p = b_{17} = 6,\ x_2^p = b_{1x_1^p} = b_{16} = 5,\ x_3^p = 4,\ x_4^p = 3,\ x_5^p = 2,\ x_6^p = 1.$$

Таким образом, кратчайший путь проходит через вершины $x_5^p, x_4^p, x_3^p, x_2^p, x_1^p$, т.е. через вершины x_2, x_3, x_4, x_5, x_6.

В таблице приведены результаты оценки эффективности распараллеливания вычислений на вычислительном кластере ТГУ СКИФ Cyberia.

Количество вершин графа	Последовательный алгоритм, с	Параллельный алгоритм					
		10 процессоров		50 процессоров		100 процессоров	
		Время,с	Ускорение	Время,с	Ускорение	Время, с	Ускорение
500	3,75	0,44	8,52	0,26	14,42	0,30	12,5
1000	54,64	6,34	8,61	2,78	19,65	1,89	28,90
3000	2588,6	268,6	9,63	59,06	43,83	32,2	80,25

Таким образом, с увеличением количества вершин графа растет ускорение параллельного алгоритма, так как количество обменов между процессорными элементами растет медленнее, чем количество арифметических операций в алгоритме Флойда, сложность которого $O(n^3)$.

Известно [2, 3], что если $A = (\alpha_{ij})_{n\times n}$ – матрица смежности графа G без петель и $A^l = (\gamma_{ij})_{n\times n}$, где $l \in N$, тогда γ_{ij} равно числу (v_i, v_j) маршрутов длины l . Это верно и для графов с петлями, если считать, что каждая петля имеет два обхода (этим число маршрутов, проходящих через данную петлю, увеличивается в два раза).

Поскольку матрица смежности A графа G является вещественной симметрической матрицей и так как она ортогонально подобна некоторой вещественной диагональной матрице D то

$$A = T^{-1}DT \ , \ A^l = T^{-1}D^lT ,$$

где матрица D имеет отличные от нуля элементы только на главной диагонали, и они совпадают со спектром графа G.

Это соотношение позволяет вычислять степени матрицы A^l по известному спектру графа, используя параллельные алгоритмы умножения матриц.

Литература

1. **Макконелл Дж.** Анализ алгоритмов. Вводный курс. М.:Техносфера, 2002. 304 с.
2. **Харари Ф.** Теория графов. М.: Мир, 2003. 300 с.
3. **Кристофидес Н.** Теория графов: алгоритмический подход. М.,1978. 432 с.
4. **Берцун В.Н.** Математическое моделирование на графах. Ч. 1. Томск: Изд-во НТЛ, 2006. 88 с.
5. **Оре О.** Графы и их применение. М.:Мир, 2002. 71 с.
6. **Воеводин В.В.** Математические модели и методы в параллельных процессах. М.: Наука, 1986. 296 с.
7. **Старченко А.В., Есаулов А.О.** Параллельные вычисления на многопроцессорных вычислительных системах. Томск: Изд-во ТГУ, 2002. 55 с.
8. **Дацюк В.Н., Букатов А.А., Жегуло А.И.** Многопроцессорные системы и параллельное программирование. Ч. 2. Среда параллельного программирования MPI. Ростов: Изд-во РГУ, 2000. 65 с.