Организация хранения данных для разбиения графов методом бисекции

Костин В. И., Краснокутская М. В.

Донецкий Национальный Технический Университет, ФВТИ, ПМИ

krasnokutskaya@maus.donetsk.ua

Abstract

Kostin V.I., Krasnokutskaya M.V. Organization of data storing for graph partition with bisection method*. We describe a dataflow graph representation of a problem parallelization. Balancing of the computational load across processors is abstracted to a graph partitioning problem. We propose review algorithms to solve this problem and describe some peculiarities of their use to graphs with high number of nodes.*

Введение

Существует ряд задач, которые не могут быть решены на обычных ЭВМ за приемлемое время. Несмотря на огромную вычислительную мощность современных компьютеров, решение может занимать недели и месяцы. Такие задачи возникают при оптимизации сложных систем, при прогнозировании погоды, моделировании разнообразных технических и природных процессов и т.д. Параллельные вычисления позволяют значительно повысить эффективность и скорость обработки информации при решении подобных задач. Для того чтобы максимально эффективно использовать преимущества многопроцессорных ЭВМ, нужно переработать алгоритмы решения задач с учетом возможности параллельной обработки данных несколькими процессорами одновременно, необходимо так сбалансировать вычислительную загрузку на процессорах, чтобы минимизировать межпроцессорное взаимодействие.

Решением задачи балансировки может быть использование графов потоков данных (ГПД). Программа представляется набором вычислительных узлов-подзадач, которые имеют фиксированное количество информационных входов и выходов. Каждая подзадача выполняется на отдельном процессоре. Узлы-подзадачи – вершины графа потоков данных, а информационные потоки между ними – ребра графа. Оптимальное распределение обработки данных между процессорами минимизирует время выполнения всех вычислений. Задача распределения обработки данных на процессоры сводится к задаче разбиения графа. Необходимо разбить граф потоков данных так, чтобы количество связей между подграфами было минимальным. Практический опыт показал, что

качество распределения задач между процессорами сильно влияет на производительность, что обусловило значительный интерес к алгоритмам разбиения графов [1].

Так как задача разбиения графа является NP-сложной задачей, то все известные алгоритмы разбиения являются эвристическими и дают приближенный к оптимальному результат. Однако, не смотря на это было разработано довольно много алгоритмов разбиения графов, дающих высококачественное разбиение за малое время [2, 3].

Среди наиболее известных алгоритмов разбиения графов можно выделить алгоритмы:

- алгоритм Kernighan-Lin / Fiduccia-Mattheyses (KL/FM);
- уровневое ячеечное разбиение;
- многоуровневые схемы;
- алгоритм спектральной бисекции.

При программной реализации любого из этих алгоритмов возникает задача выбора типа данных для представления информации о графе. Существуют различные способы внутреннего представления информации о графе в оперативной памяти ЭВМ, в том числе в виде списков (массивов) вершин и ребер, списков (массивов) смежности, матриц смежности, а также в виде комбинаций этих структур хранения. Выбор внутреннего представления оказывает решающее влияние на эффективность выполнения различных операций над графами [4] и определение лучших схем представления информации о графах является чрезвычайно актуальной задачей.

Целью данной работы является исследование эффективности методов организации данных в задачах разбиения графов больших размерностей на примере алгоритма спектральной бисекции с целью выявление наиболее оптимальных для задач разбиения.

1. Алгоритм спектральной бисекции

Исходным объектом для задач разбиения служит неориентированный граф. Пространством решений служит множество всевозможных разбиений графа на непересекающиеся подграфы. На разбиения графа могут накладываться ограничения D.

Задачи декомпозиции имеют следующий набор основных критериев:

- число внешних соединений между подграфами;
- число подграфов.

Первый критерий определяет количество внешних связей, а второй критерий - число подграфов разбиения. Постановки задач могут варьироваться в зависимости от свойств графа, которые задаются моделируемым объектом.

Пусть задан неориентированный взвешенный граф G(V,E) порядка n, где $V=\{v_1,\ldots,v_n\}$ – множество вершин; $E \subseteq V \times V$ – множество ребер.

Требуется определить разбиение множества вершин V графа G(V,E) на k – подмножеств $(V_1,\ldots,V_k)$ таким образом, чтобы для частей графа $G1(V_1,E_1),\ldots, Gk(V_k,E_k)$ выполнялись следующие требования:

$$V_i \cap V_j = \varnothing, \text{ для } \forall\ i \neq j, \text{ где } i,j = \overline{1,k};$$

$$\bigcup_{i=1}^{k} V_i = V; \qquad (1)$$

$$|V_1| = n_1,\ldots, |V_k| = n_k,\ n_1+\ldots+n_k = n,$$

Сечением разбиения $C(V_1,\ldots,V_k)$ будем называть совокупность ребер, соединяющих вершины, которые принадлежат разным подграфам.

В качестве критерия оптимальности Q разбиения графа методом бисекции $(V_1,\ldots,V_k)$ будем рассматривать вес сечения, являющейся суммой всех ребер этого сечения:

$$Q(V_1,V_2,\ldots,V_k) = \frac{1}{2}\sum_{L=1}^{k-1}\sum_{i\in E_L}\sum_{j\notin E_L} 1 \to \min \qquad (2)$$

В этом случае оптимальным k-разбиением является решение $(V_1^*,\ldots,V_k^*)$ экстремальной задачи (2) разбиения $(V_1^*,\ldots,V_k^*)$ с минимальным весом сечения $C(V_1^*,\ldots,V_k^*)$.

Частным случаем задачи k-разбиения является бисекции графа , то есть при k=2. В этом случае решение задачи разбиения графа (V_1,V_2) будем называть бисекцией.

Система требований (2) определяет область поиска D задачи разбиения графа. Так как данная задача относится к задачам переборного типа, то общее число допустимых решений |D| можно вычислить из выражения:

$$\frac{n!}{n_1!\cdot n_2!\cdot\ldots\cdot n_k!\cdot t!} \qquad (3)$$

где t – общее число подграфов, имеющих одинаковые размерности [5].

Алгоритм спектральной бисекции, основанный на использовании собственных векторов и чисел матрицы, заслуживает большого внимания, так как дает хорошее соотношение между универсальностью, качеством и эффективностью [1].

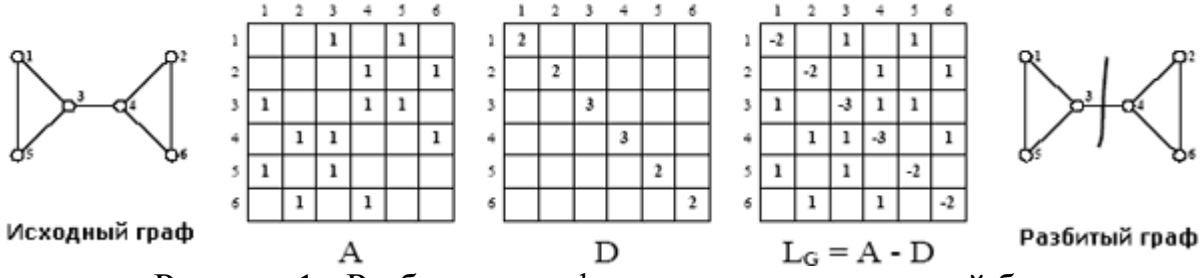

Рисунок 1 - Разбиение графа методом спектральной бисекции

На рисунке изображено разбиение графа методом спектральной бисекции. По этому алгоритму в соответствии каждой вершине графа ставится переменная x, равная +1 или –1, таким образом, что сумма всех x-ов равна 0. Первое условие подразумевает разбиение на два различных набора, а второе требует, чтобы наборы были равного размера, учитывая четность исходного количества. Будем называть вектор x вектором-индикатором, который будет показывать принадлежность каждой вершины к набору.

Определим функцию, вычисляющую вес сечения, то есть число граней, пересекающихся между наборами. Для минимизации этой функции преобразуем ее к матричной форме, используя матрицу Лапласа L.

$$\text{Минимизировать} \quad \frac{1}{4} x^T L\, x$$
$$\text{При условии:} \quad x^T I = 0, \quad x_i = \pm 1 \qquad (4)$$

Так как разбиение графа является NP-трудной задачей, то ослабим ограничения дискретности на x и сформулируем новую непрерывную задачу, которая будет приближением к дискретной, и ее решения должны быть отображены обратно к ±1 в соответствии с некоторой схемой. Идеально, когда решение близко к ±1 .

$$\text{Минимизировать} \quad \frac{1}{4} x^T L\, x$$
$$\text{При условии:} \quad x^T I = 0, \quad x^T x = n \qquad (5)$$

Если U_1, U_2 …- нормализованные собственные векторы L с соответствующими собственными значениями $\lambda_1 <= \lambda_2 <= \lambda_3$…., то матрица L имеет следующие свойства:

- L – симметрична;
- U_i попарно ортогональны;
- $U_1 = n^{-0.5} 1$, $\lambda_1 = 0$;
- Если граф замкнутый, то только λ_1 принимает нулевое значение.

Выразим X в терминах собственных векторов L: $x = \Sigma \alpha_i U_i$, где α_i – вещественные константы, такие, что $\Sigma(\alpha_i)^2 = n$. Второе свойство гарантирует, что это всегда возможно сделать. Заменой x мы получаем функцию минимизации, зависящую от собственного значения матрицы Лапласа λ_2.
$f(x) = 0.25(\alpha_2^2 \lambda_2 + \alpha_3^2 \lambda_3 + \ldots + \alpha_n^2 \lambda_n)$, начиная с $\lambda_1 = 0$. Учитывая упорядоченность собственных величин $f(x) >= n\, \lambda_2/4$, видно, что

$(\alpha_2^2 + \alpha_3^2 + \ldots + \alpha_n^2)\, \lambda_2 <= (\alpha_2^2\, \lambda_2 + \alpha_3^2\, \lambda_3 + \ldots + \alpha_n^2\, \lambda_n)$.

Мы можем минимизировать $f(x) = n\, \lambda_2/4$, выбрав $x = \sqrt{n} U_2$. Полученный вектор x является решением непрерывной задачи. Поэтому необходимо привести вектор x к дискретному виду. Для этого определим

медиану значений $x_{i.}$ После чего отобразим вершины выше значения медианы в одно множество, а ниже – в другое. Если несколько вершин имеют значение медианы, то они распределяются так, чтобы не нарушать равновесия. Полученное решение будет самой близкой дискретной точкой к непрерывному оптимуму.

2. Методы организации данных

Как отмечалось выше, при реализации алгоритма бисекции встает задача выбора типа данных для представления информации о графе.

Задание графов с помощью матриц удобно для алгоритмов, использующие матричные вычисления (например, алгоритм спектральной бисекции). Однако, при обработке графа большой размерности (N=1000, 10000), матрицы занимают большой объем внутренней памяти. При этом необходимо учитывать, что матрицы графов сильно разрежены.

В качестве предварительного исследования рассматривалось умножение матрицы Лапласа на вектор. Рассмотрим следующие представления матрицы при программной реализации алгоритма:

– двумерный массив;

– массив динамических массивов (первый элемент динамического массива – число ненулевых элементов в строке матрицы);

– три массива: массив ненулевых элементов матрицы, массивы индексов строк и столбцов, соответствующих этим элементам;

– матрица в формате RR(C)O, что обозначает "Row - wise Representation Complete and Ordered" (строчное представление, полное и упорядоченное) [6].

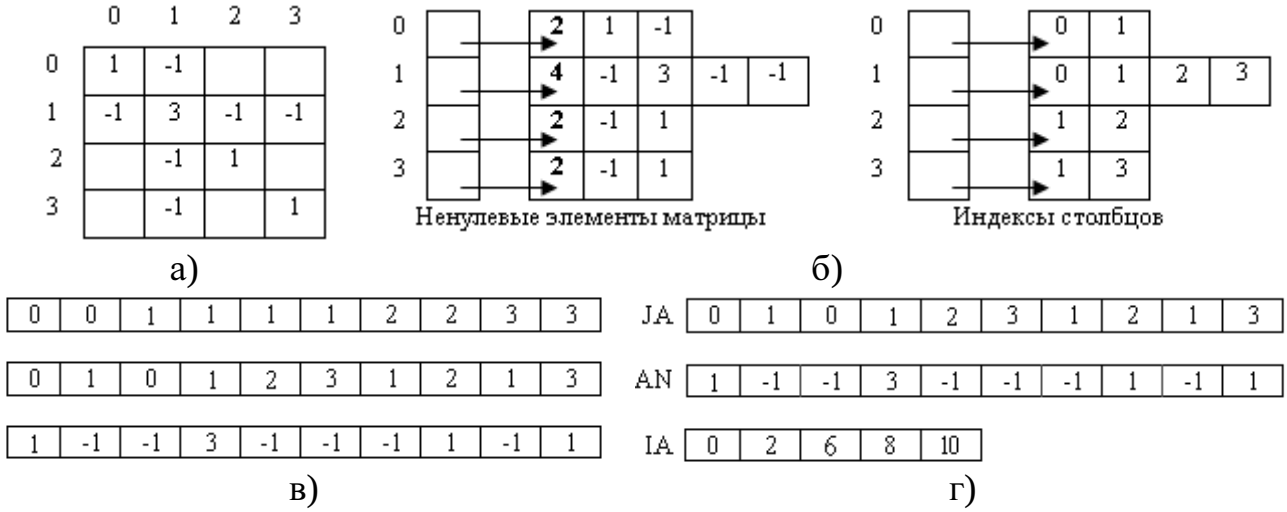

Рисунок 2 – Способы представления матрицы: а) двумерный массив; б) массив динамических массивов; в) три массива; г) матрица в формате RR(C)O

Представляя матрицу Лапласа двумерным массивом мы занимаем N^2 единиц памяти. Массив динамических массивов занимает 2N+M единиц памяти, где M – число ребер графа. Три динамических массива – 3M+N единицы памяти. Матрица в формате RR(C) – 2M +N+1 единицы.

На рис. 3 приведены результаты предварительных исследований, диаграммы зависимости времени умножения матриц от степени разреженности матрицы для матрицы N*N, где N равно 1000. Для вычисления каждой точки графика задавалась размерность матрицы N, число ненулевых элементов Z. По этим данным случайным образом создавалось 10 матриц. Каждая матрица умножалась на вектор 100 раз. Для построения диаграмм бралось среднее время умножения всех этих матриц на вектор. Вычисления проводились на компьютере с процессором Intel Celeron с тактовой частотой 2,93 GHz и оперативной памятью 1GB.

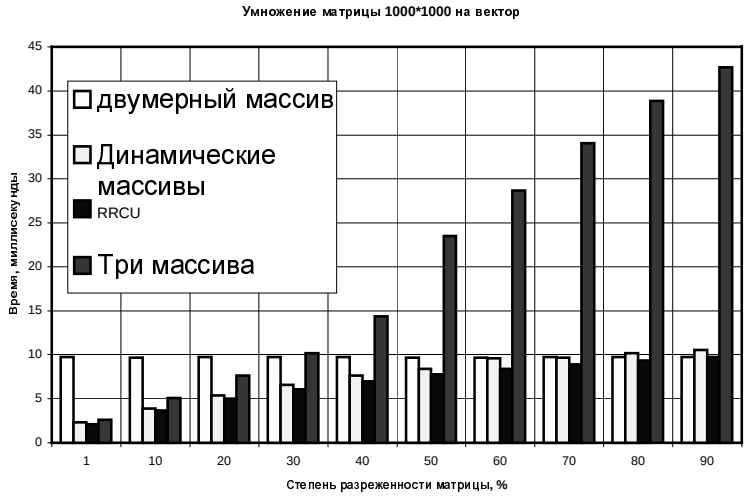

Рисунок – 3. Время умножения матрицы 1000×1000 на вектор

Из рисунка 3 видно, что степень разреженности матрицы мало влияет на скорость умножения, когда матрица задана двумерным массивом. При организации матрицы с помощью массива динамических массивов, трех массивов и в формате RR(C)O видно, что чем больше число ненулевых элементов в матрице, тем больше время умножения. Причем до определенного значения Z* это время меньше соответствующего времени для двумерного массива (для матрицы, заданной динамическими массивами и в формате RR(C)O), а для матрицы, заданной тремя массивами время ее умножения на вектор больше времени умножения двумерного массива при любой степени разреженности матрицы.

Представление матрицы двумерным массивом проще в реализации. Объем занимаемой памяти и время умножения матрицы на вектор при такой реализации – постоянны. Такой способ оптимально подходит для неразреженных матриц с большим числом ненулевых элементов. Разреженные матрицы лучше представлять массивом динамических

массивов или в формате RR(C)O, так как при такой реализации объем занимаемой памяти и время умножения матрицы на вектор зависит от размерности и степени разреженности графа. В дальнейшем исследовались организация данных в виде двумерного массива и матрицы в формате RR(C)O для задач разбиения графов методом бисекции.

Разработана программа, определяющая разбиения графа, используя метод спектральной бисекции. Случайным образом были сгенерированы графы с числом вершин N=1000 и различными степенями разреженности Z матрицы Лапласа (Z=5%, 10%, …, 95%, 100%). Для каждого графа 10 раз находилось разбиение и определялось среднее время, затраченное на разбиение. На рис. 4 показана диаграмма зависимости времени разбиения графа от Z (то есть от количества ребер). Данные приведены для двух случаев: когда при разбиении для представления матрицы Лапласа использовался двумерный массив (матрица) и формат RR(C).

Вычисления проводились на компьютере с процессором Intel Pentium 4 с тактовой частотой 3.00 GHz и оперативной памятью 512Mb.

Рисунок 4 – Разбиение графа, 1000 вершин

Из графика видно, что разбиение графа с использованием формата RRCU занимает больше времени, чем с использованием двумерного массива. Однако эта разница не существенна, что обусловлено тем, что при использовании формата RR(C) усложняется доступ к элементу матрицы. Добавление ненулевого элемента к матрице в формате RR(C) или, наоборот, превращение элемента матрицы в 0 ведет за собой манипуляции с динамической памятью, что так же приводит к временным затратам.

Что касается экономии памяти, то использование формата RR(C) для разреженных матриц выгоднее чем использование двумерного массива. К тому же метод Хаусхолдера [7,8], использовавшийся при решении, подразумевает, что каждом шаге вычислений используется подматрица меньшей размерности, следовательно остальные элементы матрицы можно принять за 0 и не учитывать. Следовательно, на каждом шаге

алгоритма объем занимаемой памяти будет уменьшаться. На рис. 5 показано изменение занимаемой памяти для матрицы Лапласа в формате RR(C) для N=1000.

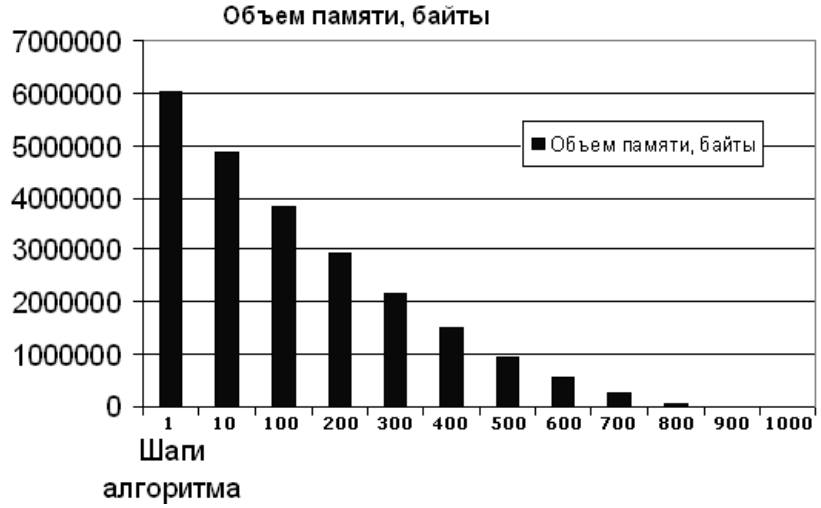

Рисунок 5 – Объем памяти, занимаемый матрицей на различных шагах алгоритма Хаусхолдера

3. Перспектива дальнейших исследований

Формат RR(C) является статической структурой хранения. Этот подход позволяет легко модифицировать матрицу и может использоваться в прямых алгоритмах разреженной линейной алгебры. Альтернативой этому походу является использование динамических структур данных, что позволяет быстро вычислять произведение разреженной матрицы и вектора, что находит свое применение в итеративных алгоритмах [6].

В основе динамического формата лежит использование хэш-таблицы, что позволяет быстрее получать доступ к элементу матрицы по его индексу. Однако добавление элемента в матрицу требует больше времени, особенно если нужно увеличивать хэш-таблицу в связи с её заполнением. Для эффективного доступа к строке/столбцу рекомендуется использовать односвязный список, в котором каждый элемент матрицы связан с двумя соседями - соседом справа и соседом снизу, так как хеш-таблица не предназначена для подобных операций. Такой список позволяет наиболее эффективно работать со строками и столбцами. То есть каждый элемент динамической структуры является одновременно и частью хэш-таблицы, и частью списка [6].

В таблице 1 представлена трудоемкость основных операций на матрице размером M×N с Z ненулевыми элементами.

В дальнейшем планируется реализация алгоритма спектральной биекции с использованием динамического подхода для хранения разреженной матрицы.

Таблица 1 – Трудоемкость основных операций

операция	статический формат	динамический формат
доступ к элементу	O(N)	O(1)
чтение строки	O(N)	O(N)
чтение столбца	O(Z)	O(M)
вставка элемента	n/a	O(1)
модификация элемента	n/a	O(1)
удаление элемента	n/a	O(1)
перестановка строк	n/a	O(1)
перестановка столбцов	n/a	O(1)
умножение матрицы на вектор	O(Z)	n/a

Литература

1. Bruce Hendrickson, Robert Leland. Multidimensional Spectral Load Balancing. Sandia National Laboratories Albuquerque, 1993
2. Bradford L. Chamberlain. Graph Partitioning Algorithms for DistributingWorkloads of Parallel Computations, 1998
3. Kirk Schloegel, George Karypis, Vipin Kumar. Graph Partitioning for High Performance Scientific Simulations.University of Minnesota, Department of Computer Science, Minneapolis, 2000
4. А. Чернобаев. Графы. AGraph: библиотека классов для работы с помеченными графами. http://www.caravan.ru/~alexch/AGraph
5. Д. И. Батищев, Н. В. Старостиным. Методические указания по проведению лабораторных работ «задачи декомпозиции графов» по курсу «Эволюционно-генетические алгоритмы решения оптимизационных задач» для студентов факультета ВМК специальности «Прикладная информатика». Нижегородский государственный университет, 2001
6. В. Быстрицкий. Представление разреженных матриц. http://alglib.sources.ru/articles/zeromatr.php.
7. Т. Шуп. Решение инженерных задач на ЭВМ: Практическое руководство. Пер. с англ. – М.: Мир, 1982. – 238 с
8. А. Д. Мышкис. Математика для втузов. Специальные курсы. М.: Наука, 1971. – 632 с