ИПМ им.М.В.Келдыша РАН • Электронная библиотека

Препринты ИПМ • Препринт № 2 за 2020 г.

ISSN 2071-2898 (Print)
ISSN 2071-2901 (Online)

Головченко Е.Н.

Обзор алгоритмов декомпозиции графов

Рекомендуемая форма библиографической ссылки: Головченко Е.Н. Обзор алгоритмов декомпозиции графов // Препринты ИПМ им. М.В.Келдыша. 2020. № 2. 38 с. http://doi.org/10.20948/prepr-2020-2
URL: http://library.keldysh.ru/preprint.asp?id=2020-2

О р д е н а Л е н и н а
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
имени М.В.Келдыша
Р о с с и й с к о й а к а д е м и и н а у к

Е.Н. Головченко

Обзор алгоритмов декомпозиции графов

Москва — 2020

Головченко Е.Н.

Обзор алгоритмов декомпозиции графов

Рассматриваются алгоритмы декомпозиции, реализованные в последовательных пакетах METIS, Scotch, Jostle, Chaco, Party и параллельных пакетах GridSpiderPar, ParMETIS, PT-Scotch, Zoltan и Jostle. Описываются различные методы декомпозиции сеток и разбиения графов, приводятся их достоинства и недостатки.

Ключевые слова: декомпозиция сеток, разбиение графов

Evdokia Nikolaevna Golovchenko

Survey of graph partitioning algorithms

Partitioning algorithms in the serial tools METIS, Scotch, Jostle, Chaco, Party and parallel tools GridSpiderPar, ParMETIS, PT-Scotch, Zoltan and Jostle are considered. Diverse methods of mesh decomposition and graph partitioning are described, with their advantages and shortcomings.

Key words: mesh decomposition, graph partitioning

Оглавление

1. Введение

При численном моделировании на высокопроизводительных вычислительных системах проблем механики сплошных сред и др. возникает задача рациональной декомпозиции сеток. Такие приложения обычно распараллеливаются методом геометрического параллелизма, где сетка распределяется между процессорами по геометрическому признаку. Эффективность работы параллельных приложений определяется тем, насколько равномерно распределена нагрузка между процессорами и насколько минимизированы обмены между ними.

Задача сбалансированного разбиения сетки на домены сводится к более общей задаче разбиения графа на домены (домены – части сеток). В этом случае выполняется разбиение графа, аппроксимирующего вычислительные и коммуникационные нагрузки сетки. Сетка аппроксимируется неориентированным графом $G = (V, E)$, где V – множество вершин, E – множество ребер. И вершины, и ребра имеют вес. Оптимальным считается разбиение на домены, при котором выровнен суммарный вес вершин в доменах и минимизирован суммарный вес разрезанных ребер (разрезанное ребро – ребро, соединяющее вершины из разных доменов). Поставленная задача декомпозиции графа является NP-полной, поэтому для ее решения используются различные эвристические методы.

Существуют следующие пакеты декомпозиции: последовательные пакеты METIS, Scotch, Jostle, Chaco, Party и параллельные пакеты GridSpiderPar, ParMETIS, PT-Scotch, Zoltan и Jostle. Пакет Jostle уже не развивается.

В препринте рассматриваются различные алгоритмы, реализованные в данных пакетах. В основном рассмотрены алгоритмы статической декомпозиции, когда декомпозиция выполняется один раз перед началом расчета. Алгоритмы динамической декомпозиции, вызывающиеся периодически во время счета задачи, рассмотрены в главе, посвященной алгоритмам динамической балансировки, а также в главе про диффузионные и генетические алгоритмы.

Среди методов рассмотрены параллельный алгоритм геометрической декомпозиции сеточных данных и параллельный инкрементный алгоритм декомпозиции графов пакета GridSpiderPar, разработанного в ИПМ им. М.В.Келдыша РАН Головченко Е.Н. и Якобовским М.В.

В препринте также рассмотрено построение графа по сетке и различные модели декомпозиции графов.

2. Построение графа сетки

Для учета коммуникационных нагрузок сетки задача разбиения сетки на домены обобщается на задачу разбиения соответствующего графа на домены. Для каждой сетки можно построить графы различных видов.

Нодальный граф

В нодальном графе (nodal graph) вершины графа соответствуют узлам сетки, а ребра – связям между узлами [1]. В полном нодальном графе (complete nodal graph) между вершинами существует ребро, если соответствующие узлы связаны элементом (рис. 1б). В сокращенном нодальном графе (reduced nodal graph) между вершинами есть ребро, только если соответствующие узлы связаны линией на границе элемента (рис. 1в).

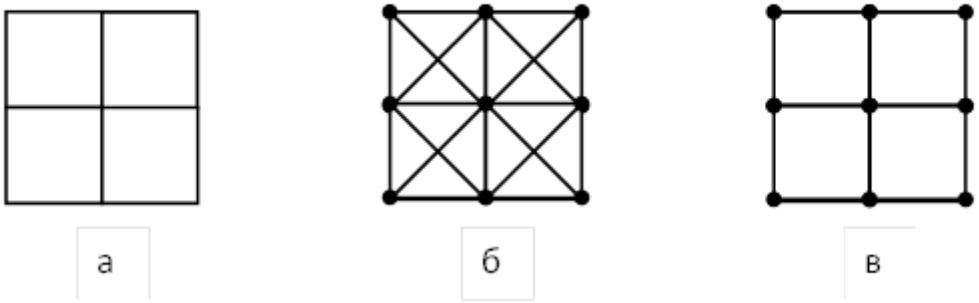

Рис. 1. Построение нодального графа, а) – исходная сетка, б) – полный нодальный граф, в) – сокращенный нодальный граф

Дуальный граф

В дуальном графе (dual graph) вершины представляют собой элементы сетки. В сокращенном дуальном графе (reduced dual graph) на Рис. 2б вершины соединены ребром, если соответствующие элементы сетки имеют общую грань (для двумерной сетки – ребро). В полном дуальном графе (complete dual graph) на рис. 2в между вершинами существует ребро, если соответствующие элементы имеют общие грань, ребро или узел (для двумерной сетки – ребро или узел).

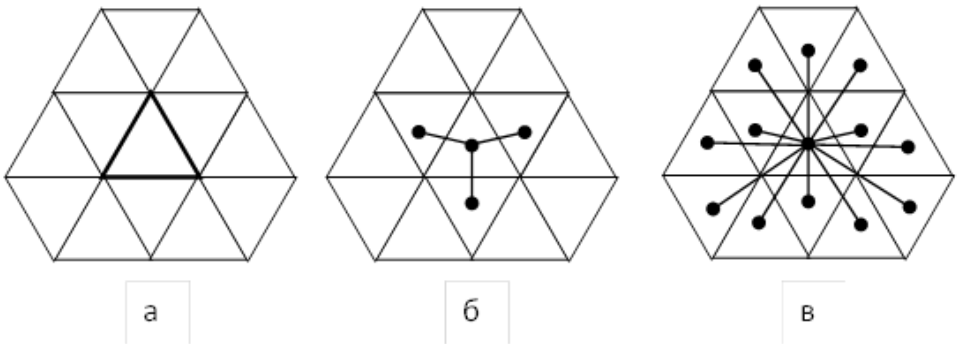

Рис. 2. Построение дуального графа, а) – исходная сетка, б) – часть сокращенного дуального графа для выделенного элемента, в) – часть полного дуального графа для выделенного элемента

3. Модели декомпозиции графов

Существует несколько моделей декомпозиции графов [2, 3], отличающихся видом графа и критериями сбалансированного разбиения: стандартная модель графа (standard graph model), модель двудольного графа (bipartite graph model), модель гиперграфа (hypergraph model), модель декомпозиции с несколькими ограничениями (multi-constraint partitioning), модель декомпозиции с несколькими целевыми функциями (multi-objective partitioning) и модель асимметричного разбиения (skewed partitioning).

В стандартной модели графа сетка аппроксимируется неориентированным графом $G = (V, E)$, где V – множество вершин, E – множество ребер. Веса вершин аппроксимируют вычислительную нагрузку в вершинах, веса ребер – коммуникационную нагрузку. Задача декомпозиции графа заключается в таком разбиении вершин на домены, в котором сбалансирован вес вершин в доменах и минимизирован суммарный вес разрезанных ребер.

У метрики разрезанных ребер выделяют несколько недостатков. Первое, поскольку вершина с одного процессора может быть связана с несколькими вершинами с другого процессора, суммарный вес разрезанных ребер не пропорционален суммарному объему коммуникаций. Второе, эффективность параллельных приложений определяется работой самого медленного процессора. Поэтому имеет смысл минимизировать объем коммуникаций, или число сообщений, каждого процессора.

Стандартная модель графа имеет следующие ограничения. Во-первых, данная модель отражает только симметричные зависимости между данными, что является проблемой для итерационных методов решения дифференциальных уравнений с несимметричными матрицами. В статье [2] описывается несколько способов устранения этого ограничения: использование ориентированного графа и варьирование весов ребер. Во-вторых, в задаче параллельного умножения матрицы на вектор стандартная модель использует одну вершину для ряда и столбца с одинаковыми индексами. В результате невозможно представить неквадратные матрицы. В-третьих, стандартная модель графа непригодна для многофазных вычислений, возникающих, например, при постановке граничных условий, моделирующих различные явления в физических вычислениях.

Одной из альтернативных моделей декомпозиции графа является модель двудольного графа (bipartite graph model) [3, 4]. В ней вершины в графе $G = (V_1, V_2, E)$ разделены на два множества V_1 и V_2 так, что ни одно ребро не соединяет вершины из одного множества, все ребра связывают вершины из разных множеств. В задаче параллельного умножения матрицы на вектор данная модель использует одно множество вершин для представления строк, другое для столбцов, что позволяет описывать несимметричные неквадратные матрицы. Модель двудольного графа также хорошо описывает задачи с двумя вычислительными операциями, например, переход между фазами в двухфазных вычислениях.

Другая альтернатива – модель гиперграфа (hypergraph model), предложенная Catalyurek, Aykanat и Pinar. Гиперграф $G = (V, H)$ состоит из множества вершин V и множества гиперребер H. Каждое гиперребро включает в себя подмножество вершин из V. Таким образом, гиперребро является обобщением ребра в графе, когда может быть связано больше двух вершин. Задача декомпозиции гиперграфа заключается в таком разбиении множества вершин на домены одинакового веса, в котором минимизировано число разрезанных гиперребер. Чтобы построить гиперграф из графа, нужно в гиперребро включить вершину и всех ее соседей. Поскольку объем вычислений зависит от того, с каким числом соседних процессоров связана вершина, минимизация числа процессоров, разрезающих гиперребро, приводит к уменьшению объема коммуникаций, в отличие от метрики разрезанных ребер. Также гиперграф является привлекательной альтернативой модели двудольного графа в несимметричных задачах с одной операцией.

Многофазные вычисления хорошо описываются моделью разбиения с несколькими ограничениями (multi-constraint partitioning), предложенной Karypis и Kumar [5, 6]. Данная модель является расширением стандартной модели графа, когда каждая вершина в графе обладает вектором весов, k-ый вес соответствует нагрузке вершины на k-ой фазе вычислений. Ребра отображают зависимости между данными на всех фазах вычислений. Задача декомпозиции состоит в том, чтобы разбить вершины на домены так, чтобы минимизировать вес разрезанных ребер и сбалансировать каждый из весов вершин, тогда будет сбалансирована каждая фаза вычислений.

Похожей моделью является модель разбиения с несколькими целевыми функциями (multi-objective partitioning), предложенная Schloegel, Karypis и Kumar. В ней одновременно минимизируются несколько целевых функций. Каждая вершина обладает вектором весов, j-ый вес соответствует j-ой целевой функции. Цель разбиения состоит в такой балансировке весов вершин, при которой минимизирована каждая из целевых функций. Данная модель хорошо описывает многофазные вычисления, в которых на каждой фазе возникает свое распределение коммуникационной нагрузки.

Хотя модели разбиения с несколькими ограничениями и несколькими целевыми функциями очень привлекательны в плане общности, реализация декомпозиции в данных моделях сложна, поэтому по возможности используются более простые модели.

Еще одна альтернатива стандартной модели графа – модель асимметричного разбиения (skewed partitioning), которая также является расширением стандартной модели. В ней каждой вершине присваивается набор из k значений, каждое из которых определяет относительное предпочтение попадания данной вершины в k-ый домен. Предпочтения учитываются вместе с коммуникационной метрикой при построении разбиения. Модель асимметричного разбиения используется, например, при динамической

балансировке загрузки, когда наряду с минимизацией объема коммуникаций желательно ограничить количество перемещаемых вершин.

Несмотря на ограничения метрики разрезанных ребер, стандартный подход доказал свою успешность при параллельном решении дифференциальных уравнений и задач, основанных на сетках, по нескольким причинам [2]. Первое, вершины в сетках обычно имеют сравнительно небольшое число соседей, поэтому число разрезанных ребер отличается на небольшой множитель от реального коммуникационного объема. Второе, вычислительные сетки обычно имеют высокую степень геометрической локальности, что гарантирует существование хорошего разбиения. Если размер сетки n увеличивается, а число процессоров остается фиксированным, коммуникационный объем растет как $n^{2/3}$ в трехмерном пространстве и $n^{1/2}$ в двумерном. Аналогично, геометрическая локальность ограничивает число сообщений, которое посылает каждый из процессоров. Из всего вышесказанного следует, что при решении задач на больших сетках важна вычислительная эффективность, а критерии определения коммуникационного объема не настолько критичны.

Статическая декомпозиция сетки проводится один раз перед расчетом задачи. В отличие от нее, динамическая декомпозиция выполняется периодически в течение счета для балансировки загрузки, и на нее налагаются дополнительные ограничения, такие как небольшие объемы перемещаемых данных. В статической декомпозиции сетки хорошо работает стандартная модель графа.

4. Простые алгоритмы разбиения сеток

Наиболее простыми алгоритмами разбиения сеток являются алгоритмы, учитывающие только номера вершин. К таким алгоритмам относятся линейное распределение вершин (linear method), рассеивание (scattered method) и случайное распределение (random method) [7]. При линейном распределении непрерывный диапазон номеров вершин разбивается на интервалы, вершины из одного интервала попадают в один домен. При рассеивании номер вершины, взятый по модулю числа доменов, определяет, в какой домен попадет вершина. То есть сначала в каждый домен по очереди распределяется по одной вершине, потом по две и т.д. При случайном распределении домен вершины выбирается случайным образом среди доменов, в которых вершин меньше, чем должно быть при равномерном распределении. Описанные методы включены в пакеты Chaco [8] и Party [7, 9]. Достоинством данных алгоритмов является быстрота выполнения. Поэтому они используются для получения предварительного разбиения сетки, когда декомпозиция сетки выполняется на распределенной вычислительной системе, и перед вычислением самой декомпозиции сетку нужно неким образом распределить между процессорами. Однако разбиения, получаемые данными методами, не учитывают геометрию сетки, поэтому в качестве предварительного разбиения они подходят только для тех алгоритмов,

результат работы которых не зависит от качества начального распределения вершин по процессорам.

5. Алгоритмы геометрической декомпозиции сеток

Более сложными, но довольно быстрыми алгоритмами разбиения сетки являются алгоритмы, учитывающие геометрическую информацию о сетке. К ним относятся методы рекурсивных координатной бисекции (recursive coordinate bisection), инерциальной бисекции (recursive inertial bisection) и разбиение с использованием кривой Гильберта (Hilbert space-filling curve partitioning) [8, 10, 11, 12].

На каждом этапе рекурсивной бисекции область разбивается на две части. Соотношение размеров частей зависит от количества доменов, которые должны быть образованы в каждой из частей. Полученные подобласти разбиваются дальше аналогичным образом до тех пор, пока в подобластях не останется по одному домену. При рекурсивной координатной бисекции на этапе разбиения выбирается координатная ось, вдоль которой область имеет наибольшую протяженность. Область разбивается перпендикулярно полученной оси. Рекурсивная инерциальная бисекция отличается от координатной тем, что выбираемая ось не обязана быть параллельной одной из осей координат. Вычисляется ось инерции, которая обычно является направлением, вдоль которого область имеет наибольшую протяженность.

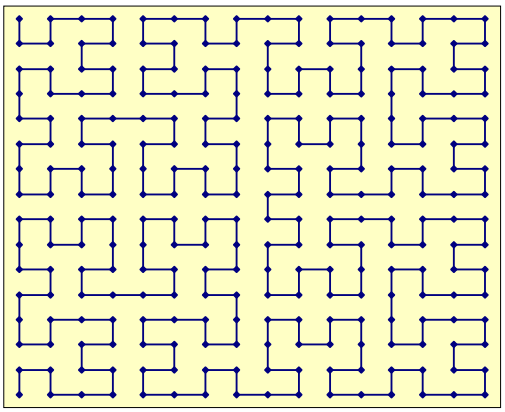

Рис. 3. Кривая Гильберта 4-го порядка на 2-мерной области

Кривая Гильберта – непрерывная кривая в смысле Жордана (кривая является непрерывным образом отрезка), целиком заполняющая некоторый гиперпараллелепипед (рис. 3). Иначе говоря, кривая Гильберта всюду плотна в области, в которой построена, то есть проходит через любую сколь угодно малую окрестность каждой точки этой области [13, 14]. Кривой Гильберта отрезок [0; 1] отображается на n-мерный гиперпараллелепипед. Точки, расположенные близко на отрезке, находятся на близком расстоянии в гиперпараллелепипеде. Алгоритм получения разбиения с использованием кривой Гильберта состоит из следующих частей [10, 15]:

- построение кривой Гильберта, отображающей отрезок [0; 1] на наименьший гиперпараллелепипед, содержащий все вершины;

- отрезок [0; 1] разбивается на малые отрезки [left, right), число которых превышает число формируемых доменов (в последний отрезок включается единица);

- слева направо выполняется суммирование весов вершин в малых отрезках, в результате чего из малых отрезков набираются большие отрезки, соответствующие доменам.

Геометрические методы входят в пакеты Chaco, Party, ParMETIS [16, 17, 18] и Zoltan [10]. В отличие от простых алгоритмов, геометрические алгоритмы позволяют получать более «компактные» домены, каждый из которых связан с меньшим числом соседей. Хотя эти алгоритмы не учитывают коммуникационные нагрузки сетки, они используются как для получения предварительных разбиений при распределении вершин по процессорам в других параллельных алгоритмах декомпозиции, так и тогда, когда важны быстрота получения разбиения и экономность использования памяти. Разбиения, полученные геометрическими методами, отличаются меньшим дисбалансом числа вершин в доменах, чем полученные алгоритмами декомпозиции графов. Поэтому геометрические алгоритмы используются еще и тогда, когда сбалансированность разбиения важнее коммуникационных нагрузок.

5.1. Параллельный алгоритм геометрической декомпозиции сеточных данных

Параллельный алгоритм геометрической декомпозиции сеточных данных пакета GridSpiderPar (Головченко Е.Н. и Якобовский М.В., ИПМ им. М.В. Келдыша РАН) основывается на методе рекурсивной координатной бисекции [10, 7]. На каждом этапе рекурсивной бисекции область разбивается на две части. Полученные подобласти разбиваются дальше аналогичным образом до тех пор, пока в подобластях не останется по одному домену.

Основные этапы алгоритма следующие [19 - 25]:

- Случайное начальное распределение вершин по процессорам. Начальное распределение может быть любым, например, в соответствии с порядковыми номерами вершин.
- Рекурсивная координатная бисекция вершин по процессорам (рис. 4):
  - Параллелепипед, заключающий в себе сетку, разбивается на две части. Выбирается координатная ось, вдоль которой параллелепипед имеет наибольшую протяженность. Параллелепипед разрезается перпендикулярно выбранной оси.
  - Группа процессоров делится на две, далее каждая из групп делит свой блок вершин аналогичным образом. Для разделения блока вершин

используется параллельная сортировка по выбранной оси координат (а также по остальным для разрезания секущей плоскости) [73].

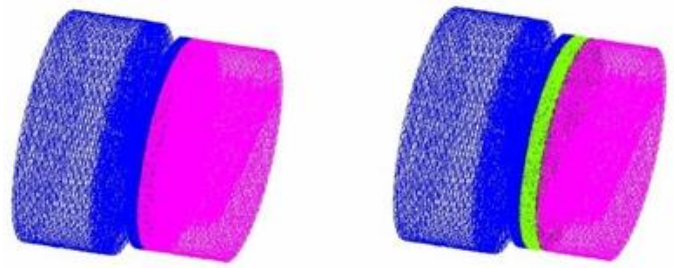

Рис. 4. Геометрическое разбиение сетки на семь доменов на трех процессорах. Первые два этапа разбиения: распределение вершин по процессорам

• Локальная рекурсивная координатная бисекция вершин по доменам. Дальнейшее разбиение на домены проводится локально на каждом процессоре (рис. 5).

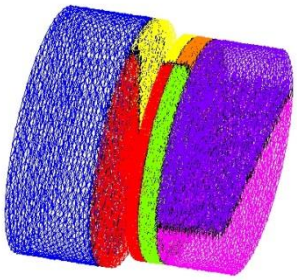

Рис. 5. Результат разбиения сетки на семь доменов на трех процессорах

Отличие рассматриваемой рекурсивной координатной бисекции от обычного метода рекурсивной координатной бисекции состоит в следующем. Секущая плоскость (медиана) при необходимости разрезается по нескольким координатам, что позволяет обрабатывать ситуации наличия на одной плоскости множества вершин с одинаковым значением координаты (Рис. 6). В результате медиана проводится точно, и в разбиении на равные домены числа вершин в доменах отличаются не больше чем на единицу.

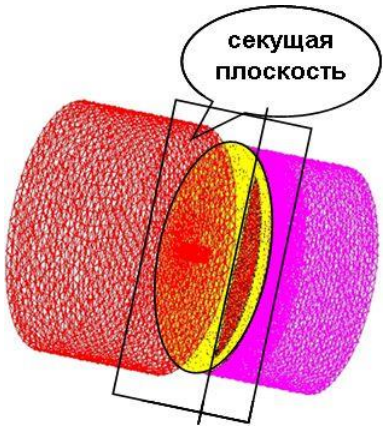

Рис. 6. Разбиение по нескольким координатам в локальной рекурсивной координатной бисекции

Для параллельной сортировки использовалась библиотека ParSort (библиотека параллельной сортировки больших объемов структурированных данных), разработанная Якобовским М.В. и Головченко Е.Н. в ИПМ им. М.В. Келдыша РАН.

6. Алгоритмы декомпозиции графов

6.1. Спектральная бисекция

В алгоритме спектральной бисекции (spectral bisection) для разбиения графа используется спектральная информация. Строится матрица Лапласа $Q = A - D$, где D – диагональная матрица, A – матрица смежности графа [26, 27]:

$$a_{i,j} = \begin{cases} ew(v_i, v_j), \text{ если } (v_i, v_j) \in E, \\ 0 \quad\quad\quad , \text{иначе}, \end{cases}$$

$$d_{i,i} = \sum_j ew(v_i, v_j) \text{ для всех } (v_i, v_j) \in E,$$

где $ew(v_i, v_j)$ – вес ребра (v_i, v_j), E – множество ребер. Вычисляется собственный вектор, соответствующий второму наибольшему собственному значению. Этот вектор называется вектором Фидлера (Fiedler vector), и его величина является мерой связей графа. Далее множество вершин графа V упорядочивается по неубыванию значений компонент вектора Фидлера y: $y_{v_i} \le y_{v_j}, V_1 = \{v_i, v_j, ..\}$. Затем определяется взвешенная медиана множества V_1, и первая часть вершин распределяется в первый домен, вторая – во второй. То есть в первый домен попадают вершины с меньшими

соответствующими значениями компонент вектора Фидлера. Вектор Фидлера вычисляется алгоритмом Lanczos. Это итеративный алгоритм, число итераций которого зависит от желаемой точности [26]. Аналогичным образом можно разбить множество вершин на четыре домена, используя второй и третий собственные вектора, и на восемь частей, используя второй, третий и четвертый собственные вектора [8, 28].

Рекурсивная спектральная бисекция включена в пакеты METIS, Chaco, Party (через интерфейс к библиотеке Chaco), а также в TOP/DOMDEC [29], HARP [30] и S-HARP [31, 32] (параллельная версия HARP). Алгоритм формирует разбиения неструктурированных сеток, лучшие по качеству, чем получаемые геометрическими методами, но его вычислительная стоимость зависит нелинейно от размера сетки, поэтому алгоритм рекурсивной спектральной бисекции в основном используется в рамках иерархического подхода.

6.2. Алгоритм локального уточнения

Изначально алгоритм локального уточнения был разработан Kernighan и Lin (KL) как алгоритм бисекции графа. Затем Fiduccia и Mattheyses (FM) предложили линейную реализацию данного алгоритма. Поэтому обычно алгоритм локального уточнения называется KL/FM алгоритмом. Hendrickson и Leland усовершенствовали данный алгоритм, распространив его на произвольное число доменов, и реализовали в пакете Chaco [33]. Они также ввели случайность в алгоритм, что сделало его более устойчивым, и отказались от пересчета всех структур на каждой итерации внешнего цикла при перемещении малой доли вершин, что ускорило работу алгоритма. Именно алгоритм локального уточнения Hendrickson и Leland описан ниже.

В основе алгоритма лежит понятие выигрыша от перемещения вершины v_i из домена l в домен k:

$$g^k(v_i) = \sum_{(v_i, v_j) \in E} \begin{cases} ew(v_i, v_j), \text{ если } P(v_j) = k, \\ -ew(v_i, v_j), \text{ если } P(v_j) = l, \\ 0, \text{ в остальных случаях,} \end{cases}$$

где $g^k(v_i)$ – выигрыш от перемещения вершины v_i из домена l в домен k, $ew(v_i, v_j)$ – вес ребра (v_i, v_j), E – множество ребер, $P(v_j)$ – домен, которому принадлежит вершина v_j. Таким образом, выигрыш от перемещения

вершины – это уменьшение общего веса разрезанных ребер, полученное от перемещения этой вершины из одного домена в другой.

Сам алгоритм состоит из двух циклов, внешнего и внутреннего:

Внешний цикл

приравнивание лучшего разбиения к текущему разбиению

вычисление начальных выигрышей

Внутренний цикл

выбор перемещения с наибольшим выигрышем

перемещение

обновление выигрышей у соседей перемещенной вершины

запоминание лучшего разбиения

Конец внутреннего цикла

приравнивание текущего разбиения к лучшему разбиению, полученному во внутреннем цикле

Конец внешнего цикла.

На первой итерации внешнего цикла вычисляются выигрыши от перемещений всех вершин в другие домены. На каждой итерации внутреннего цикла выбирается перемещение с наибольшим выигрышем. При этом учитывается критерий балансировки: вершины перемещаются только из доменов, размер которых больше среднего, в домены, размер которых меньше или равен среднему. Каждую вершину можно перемещать только один раз в течение внутреннего цикла, что предотвращает циклическое перемещение одной вершины между несколькими доменами. Наибольший выигрыш может оказаться отрицательным. Важным качеством KL алгоритма является способность выбираться из локальных минимумов, то есть несколько перемещений с отрицательными выигрышами могут, в конечном счете, привести к разбиению с меньшим общим весом разрезанных ребер. После каждого перемещения обновляются выигрыши соседей перемещенной вершины. В течение всего внутреннего цикла запоминается лучшее разбиение, полученное во внутреннем цикле. Критерии завершения внутреннего цикла могут быть разными [33, 34, 26]. Например, когда больше не существует перемещений из больших доменов в меньшие. Или когда получена слишком большая разница между качеством текущего разбиения и наилучшего разбиения, полученного к данному моменту во внутреннем цикле. Этот критерий используется в иерархическом подходе, когда считается, что локальное уточнение не должно производить слишком большие изменения полученного разбиения. Перед началом следующей итерации внешнего цикла восстанавливается наилучшее разбиение, полученное во внутреннем цикле. Для ускорения процесса пересчитываются не все выигрыши, а только выигрыши вершин, перемещенных во внутреннем цикле, и их соседей. Внешний цикл завершается, когда несколько его итераций не привели к получению лучшего разбиения.

Поскольку обычно существует несколько перемещений с одинаковыми выигрышами, начальные выигрыши на первой итерации внешнего цикла подсчитываются в случайном порядке. В дальнейшем при пересчете выигрыша от перемещения вершины этот выигрыш попадает на первое место среди таких же выигрышей. Поскольку обновляются выигрыши у всех соседей перемещенной вершины, их выигрыши рассматриваются вместе, что соответствует духу всего алгоритма.

Время выполнения алгоритма локального уточнения $O((s-1) \cdot m)$ [33], где s – число доменов, m – число ребер в графе. И хотя алгоритм рассчитан на произвольное число доменов, он плохо масштабируется по числу доменов и числу вершин [8].

Качество разбиений, получаемых алгоритмом локального уточнения, сильно зависит от выбираемого начального разбиения, поэтому его ценность не в использовании алгоритма самого по себе, а в улучшении разбиений, получаемых другими алгоритмами, например, в иерархических методах.

Алгоритм локального уточнения используется в таких пакетах, как METIS (ParMETIS), Chaco, Party и Jostle. В них реализовано несколько вариаций данного алгоритма. В частности, в алгоритме локального уточнения пакета Jostle рассматриваются только вершины на границах между доменами [34] и добавлена балансировка загрузки, использующая диффузионный алгоритм, разработанный Hu и Blake [35].

Один из алгоритмов локального уточнения из пакетов METIS и ParMETIS, – жадное уточнение (greedy refinement) [36, 37]. Этот алгоритм, как и остальные, используется в рамках иерархического подхода, который подробно описан ниже. В жадном уточнении предполагается, что перемещение одной вершины огрубленного графа аналогично перемещению группы вершин исходного графа, поэтому можно не ожидать улучшения разбиения после нескольких перемещений, а перемещать только вершины с положительным выигрышем. В результате, во внутреннем цикле все вершины навещаются в случайном порядке, и если вершина лежит на границе между доменами, она перемещается:

- в домен с наибольшим выигрышем, если выигрыш положительный и не нарушается сбалансированность разбиения;
- в домен с нулевым выигрышем, если баланс улучшается;
- если таких доменов нет, вершина не перемещается.

Алгоритм имеет меньшую зависимость от числа доменов, но теряет способность выбираться из локальных минимумов.

В последовательном пакете Party реализован алгоритм полезных множеств (helpful-sets), который очень похож на алгоритм локального уточнения, только работает не с выигрышами отдельных вершин, а с целыми наборами [7].

6.3. Иерархические алгоритмы

Иерархические алгоритмы состоят из следующих основных частей: поэтапное огрубление графа, декомпозиция самого маленького из полученных графов и отображение разбиения на предыдущие графы с периодическим локальным уточнением границ доменов.

Огрубление графа

На стадии огрубления граф аппроксимируется цепочкой графов с меньшим числом вершин. В основе огрубления лежит понятие стягивания ребра [33]. Когда ребро стягивается, две вершины, лежащие на его концах, превращаются в одну с весом, равным сумме весов стягиваемых вершин. На Рис. 7 первоначальный граф состоит из трех вершин, веса всех вершин и ребер равны единице. Стягивается ребро между вершинами с номерами 1 и 2, на рисунке оно покрашено в синий цвет. В результате получается граф, состоящий из двух вершин с весами 1 и 2. Веса ребер остаются неизменными, за исключением случая, когда обе стягиваемые вершины связаны с одним соседом. В этом случае новое ребро замещает собой два, и его вес становится равным сумме весов соответствующих ребер. На рис. 7 обе вершины 1 и 2 связаны с вершиной с номером 0. Вес итогового ребра между вершиной 0 и новой вершиной становится равным двойке.

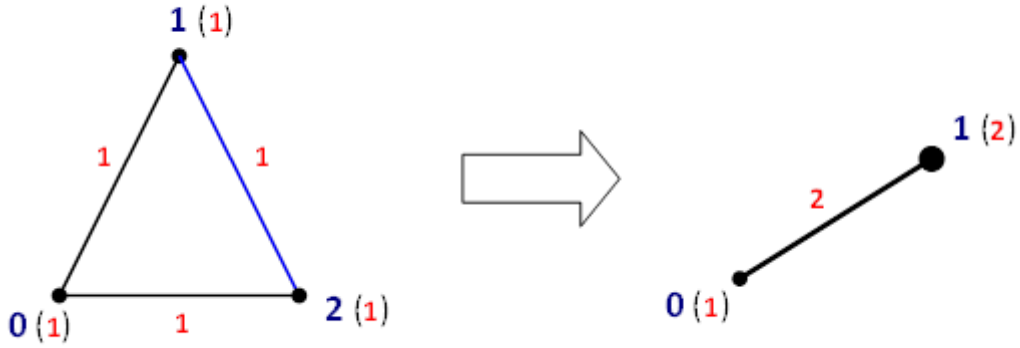

Рис. 7. Стягивание ребра

Поскольку обычно граф состоит из множества ребер, вначале находится так называемое максимальное покрытие (maximal matching), то есть набор ребер, никакие два из которых не связаны с одной вершиной, и в набор нельзя добавить ни одного ребра без нарушения этого свойства. Наиболее простой способ нахождения такого покрытия (он используется в пакете Chaco [33]) – проход по всем вершинам в случайном порядке. Если вершина не была отмечена ранее, случайным образом выбирается ее неотмеченный сосед, и ребро помещается в максимальное покрытие стягиваемыми ребрами. Более

эффективным способом выбора соседа является выбор ребра с наибольшим весом [26, 38], такой способ используется в пакетах METIS (ParMETIS) и Jostle. В таком случае разрезанными ребрами окажутся менее тяжелые ребра. В пакете Jostle при выборе среди одинаково тяжелых ребер предпочтение отдается соседу с наименьшей степенью (имеет наименьшее число соседей), во избежание получения сильно связанных вершин. В пакете WGPP [39, 40] используется комбинация огрублений по самому тяжелому ребру и самому тяжелому треугольнику (имеет наибольшую сумму весов трех ребер).

Далее выполняется непосредственное стягивание ребер и определение весов вершин и ребер полученного графа. На рис. 8 и 9 представлены исходный граф и результаты двух его последовательных огрублений. Размер вершин на рисунках увеличивается с увеличением их веса: чем больше вес ребер, тем более красного оттенка их цвет. У исходного графа веса всех вершин и ребер равны единице.

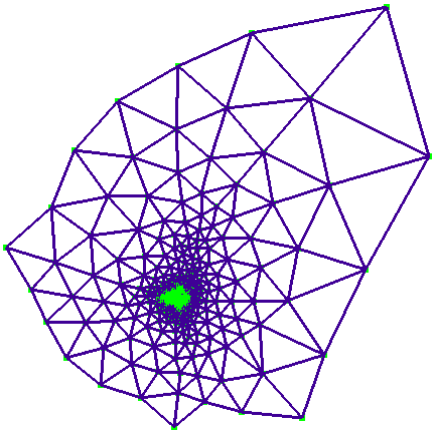

Рис. 8. Исходный граф

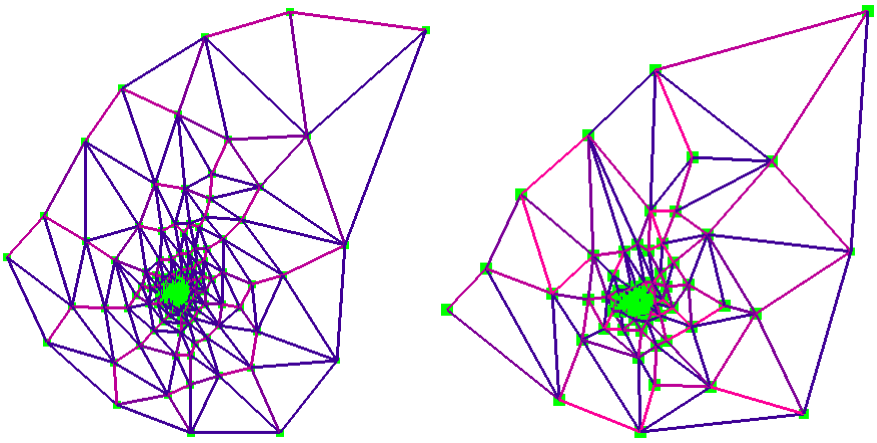

Рис. 9. Результаты первого и второго огрублений

На каждой стадии огрубления число вершин уменьшается в лучшем случае вдвое. Огрубление графа имеет несколько важных свойств. Первое, каждая вершина при огрублении переходит в одну вершину, поэтому легко

спроецировать разбиение огрубленного графа в разбиение исходного графа: вершины исходного графа распределяются в те домены, в которые попали вершины, порожденные ими. Второе, поскольку веса вершин суммируются, сохраняются ограничения на размеры доменов, которые зависят только от суммарного веса вершин, распределенных в каждый из доменов. Третье, так как суммируются веса ребер, сохраняется функция стоимости, зависящая от весов разрезанных ребер. Таким образом, для большинства метрик качества разбиений хорошее разбиение огрубленного графа, скорее всего, окажется хорошим и для исходного графа.

Итак, на стадии огрубления проводятся несколько огрублений графа до тех пор, пока размер графа не станет меньше порогового, или когда уменьшение числа вершин окажется меньше заданного.

Разбиение огрубленного графа

Алгоритмы получения самого разбиения могут быть разными, однако они должны поддерживать веса вершин и ребер для возможности работы с огрубленным графом. Часто на данной стадии используется алгоритм спектральной бисекции (Chaco, METIS) [33, 26], поскольку он дает качественные разбиения, а иерархический подход позволяет разбивать не большой исходный граф, а маленький огрубленный. Что критично в силу нелинейной зависимости времени работы этого алгоритма от размера сетки. Еще одним способом получения разбиения является исключение данной стадии путем огрубления графа до числа вершин, равного числу доменов. Однако на заключительных стадиях огрубление дает все меньший выигрыш в уменьшении числа вершин, и такой вариант может оказаться невозможным. Если же огрубление удастся, веса полученных доменов окажутся сильно несбалансированными. Однако именно такой способ используется в пакете Jostle [41, 38], и проблема дисбаланса решается на этапе локального уточнения.

Восстановление разбиения

Каждая из вершин огрубленного графа является объединением одной или двух вершин исходного графа. Поэтому домен вершины огрубленного графа становится доменом ее прообразов в исходном графе. И так по всей цепочке графов. Поскольку вес вершины огрубленного графа является суммой весов составляющих ее вершин исходного графа, в процессе восстановления графа вес вершин в доменах сохраняется. Аналогичным образом сохраняется суммарный вес разрезанных ребер.

Локальное уточнение разбиения

Восстановленный граф имеет больше степеней свободы, чем его огрубленный вариант, поэтому лучшее разбиение огрубленного графа необязательно окажется оптимальным для его предшественника. Для нахождения оптимального разбиения через каждые несколько этапов

восстановления графа выполняется локальное уточнение доменов либо алгоритмом KL/FM, описанным ранее, либо диффузионным, либо генетическим алгоритмами, которые будут рассмотрены далее. Локальное уточнение KL/FM используется в пакетах Chaco, METIS, ParMETIS, Jostle, Scotch [42], диффузионный алгоритм – в пакете PT-Scotch (параллельная версия пакета Scotch) [43, 44, 45].

Параллельные реализации иерархических алгоритмов

В большинстве существующих параллельных пакетов декомпозиции сеток используются именно иерархические алгоритмы, поэтому хочется остановиться подробнее на вопросах параллельной реализации данных алгоритмов.

Одной из первых реализаций иерархического подхода был алгоритм PMRSB, алгоритм параллельной рекурсивной спектральной бисекции [46, 47]. Данный вариант алгоритма отличался от той структуры иерархических методов, которая описана в данном параграфе. В нем вычислялся вектор Фидлера огрубленного графа, после чего проводилась интерполяция для формирования приближений к векторам Фидлера предыдущих графов с уточнением обратной итерацией. В результате формировался вектор Фидлера исходного графа, и дальше действия проводились в соответствии с обычным алгоритмом спектральной бисекции. Алгоритм PMRSB был реализован на Cray платформах.

В остальных пакетах параллельной декомпозиции сеток реализован традиционный вариант иерархического алгоритма и используется библиотека MPI.

Одним из наиболее используемых параллельных пакетов декомпозиции сеток является пакет ParMETIS. В нем после огрубления все части графа собираются на всех процессорах [48, 49, 50]. При огрублении графа меньше некоторого порога дальнейшее огрубление выполняется на половине процессоров, чтобы избавиться от задержек при отправке сообщений на маленьких графах, когда эти задержки начинают доминировать [37, 46]. Разбиение вычисляется алгоритмом рекурсивной бисекции, основанным на вложенном рассечении и жадном локальном уточнении. Недостатком разбиений, получаемых пакетом ParMETIS, является получение сильно несбалансированных разбиений при разбиении на большое число доменов.

В пакете Jostle граф огрубляется параллельно до того момента, как его можно будет поместить на один процессор [38]. После сбора на одном процессоре граф огрубляется до числа вершин, равного числу доменов, стадия разбиения исключается. После огрубления веса полученных вершин могут сильно отличаться, но проблема дисбаланса решается на этапе локального уточнения. Считается, что больший дисбаланс вершин после огрубления позволяет найти разбиение лучшего качества в процессе локального уточнения [34]. В пакете Jostle, в отличие от пакета ParMETIS, на каждом процессоре

образовывается только один домен, поскольку пакет рассчитан больше на динамическую декомпозицию [51 - 53].

В пакете PT-Scotch декомпозиция графа является частным случаем отображения графа процессов программы на граф архитектуры [43]. Огрубление графа позволяет собрать его на одном процессоре и там разбить диффузионным алгоритмом. Поскольку обычный вариант локального уточнения KL/FM плохо масштабируем по числу процессоров, что ухудшает его свойства выхода из локальных минимумов, на этапах локального уточнения используется граничный вариант диффузионного алгоритма. Вообще граничный метод в пакете PT-Scotch используется в качестве надстройки над другими методами. В нем рассматриваются только вершины на расстоянии от границы, не превышающем заданное, что позволяет использовать дорогостоящие методы путем рассмотрения только части вершин. Обычно берут расстояние в три вершины от границы. Считается, что граничный вариант при использовании с локальным уточнением, стартуя от хорошего разбиения огрубленного графа, не позволяет алгоритму зациклиться на локальных минимумах восстановленного графа [43]. В статье [45] предполагается, что результат граничного локального уточнения не будет сильно отличаться от результата обычного локального уточнения, потому как локальный минимум можно найти на следующих уровнях восстановления графа. То есть на нескольких уровнях можно скомпенсировать то, что не удалось сделать на одном. В процессе восстановления графа наступает момент, когда граф снова распределен по процессорам, однако подграфы границы собираются на каждом из процессоров и локальное уточнение происходит на процессорах независимо. Выбирается наилучший вариант, и он проецируется дальше.

В пакете Zoltan при разбиении графов либо используются интерфейсы к алгоритмам разбиений графов пакетов ParMETIS, Jostle и PT-Scotch, либо разбивается соответствующий гиперграф алгоритмом пакета Zoltan [10, 54]. При формировании гиперграфа вершины остаются такими же, как и в графе. Для каждой вершины формируется гиперребро, которое состоит из всех соседей вершины и ее самой. Таким образом, гиперребро является обобщением ребра в графе, когда может быть связано больше двух вершин. Задача декомпозиции гиперграфа заключается в таком разбиении множества вершин на домены одинакового веса, в котором минимизировано число разрезанных гиперребер.

6.4. Диффузионные и генетические алгоритмы

Диффузионные алгоритмы

У большого класса параллельных задач на нерегулярных сетках вычислительная структура постепенно меняется в процессе счета. Например, при использовании адаптивных сеток [55] некоторые области огрубляются, другие сгущаются для получения более точных результатов. В итоге меняются

веса вершин и ребер в графе, что вызывает дисбаланс существующего разбиения, и возникает необходимость подкорректировать разбиение и перераспределить нагрузку между процессорами. В динамических алгоритмах разбиения важны быстрота получения разбиения и минимизация объема перераспределяемых данных между процессорами в соответствии с полученным разбиением. Именно в динамических разбиениях в основном используются диффузионные алгоритмы. Это объясняется тем, что по своей сути диффузионные алгоритмы являются инкрементными, что обеспечивает небольшую миграцию вершин между процессорами при перемещении их в соответствии с полученным разбиением. Поскольку после первоначального разбиения сетка уже распределена по процессорам в соответствии с полученным разбиением и идет счет задачи, при динамических балансировках загрузки она перебивается на том же числе процессоров, равном числу доменов. Для ускорения сходимости диффузионные алгоритмы обычно применяют в иерархических методах. Далее описывается параллельный иерархический диффузионный алгоритм динамического разбиения адаптивных сеток пакета ParMETIS, который похож на аналогичный алгоритм в пакете Jostle.

В соответствии с обсуждаемым алгоритмом, сначала выполняется огрубление графа, в результате которого граф аппроксимируется цепочкой графов с меньшим числом вершин [56, 57]. Затем происходит диффузия вершин на самом маленьком из графов, после чего разбиение уточняется на цепочке графов. В алгоритме используется два вида диффузии: неориентированная и ориентированная. При неориентированной диффузии вершины на границах доменов обрабатываются в случайном порядке. Вершина переполненного домена перемещается в домен с меньшим весом, а если таких несколько, то в тот домен, перемещение в который обеспечит меньший итоговый вес разрезанных ребер. Вершина из домена среднего веса перемещается в домен, который не переполнится при уменьшении веса разрезанных ребер. Если таких доменов несколько, то выбирается перемещение, обеспечивающее наиболее сильное уменьшение веса разрезанных ребер. Вершины из незаполненных доменов не перемещаются. Процесс продолжается итерационно до наступления баланса или прекращения улучшения баланса.

При ориентированной диффузии в соответствии с алгоритмом Hu и Blake [58] вычисляется матрица, определяющая, какой вес вершин должен быть перемещен между соседними доменами. Вершины на границах доменов обрабатываются в случайном порядке. Если вершина соседствует с доменами, в которые, в соответствии с матрицей, нужно перенести нагрузку, она переносится в тот из них, который даст больший выигрыш в весе разрезанных ребер. Матрица обновляется. Когда проверены все вершины на границах доменов, процесс повторяется до тех пор, пока не будет достигнут баланс или не прекратится его улучшение. И при ориентированной, и при неориентированной диффузии может оказаться невозможным сбалансировать

самый грубый граф, потому как там может быть недостаточно подходящих вершин. Тогда граф восстанавливается на один уровень, и процесс диффузии выполняется на этом уровне.

Локальное уточнение напоминает неориентированную диффузию, и его целью является уменьшение суммарного веса разрезанных ребер и миграции вершин между процессорами.

В параллельной версии ориентированная диффузия выполняется только на самом маленьком из графов последовательно. Граф копируется на все процессоры. Поскольку в диффузию включена случайность, каждый процессор получает свой результат. Выбирается лучший из них. На остальных графах, если не удалось сбалансировать более грубые графы, параллельно выполняется неориентированная диффузия.

В пакете PT-Scotch диффузионный алгоритм внедрен в иерархический подход несколько иным способом, упомянутым ранее [43, 42, 59, 44]. Подразумевается статическое разбиение, то есть разбиение, выполняемое один раз перед началом расчета. Диффузионные методы позволяют построить разбиения с небольшими и гладкими границами. Однако диффузионные алгоритмы являются дорогостоящими по времени, и их использование в иерархических методах позволяет избежать этого недостатка.

Диффузионный алгоритм пакета PT-Scotch выглядит следующим образом. Вершины в графе представляются бочками, а ребра трубами. В систему заливаются из инициирующих бочек две жидкости, scotch и anti-scotch (описывается бисекция). Граница между доменами определяется границей между вершинами, содержащими различные жидкости. Из каждой бочки в единицу времени вытекает в качестве потерь количество жидкости, равное ее весу, что не позволяет каждой из жидкостей заполнить больше половины бочек. Пропускная способность труб равна весам соответствующих ребер. Из каждой бочки оставшаяся от потерь жидкость передается соседям. Когда встречаются одинаковые количества обеих жидкостей, они исчезают. Полного схождения алгоритма не ждут, а ждут только стабильного определения того, в какой бочке, какая жидкость доминирует. Алгоритм сходится, так как $|V|$ (суммарный вес вершин) обеих жидкостей заливается в целом в единицу времени в систему, и все бочки могут терять такое же суммарное количество жидкости за то же время.

В граничном варианте описываемого диффузионного алгоритма рассматриваются только вершины, находящиеся на расстоянии в три вершины от границы между доменами. Две дополнительные вершины-якоря заменяют остальные части графа. Вес каждой вершины-якоря равен суммарному весу удаленных вершин с соответствующей стороны от границы между доменами. Вершины-якоря соединяются с граничным графом. Эти вершины и являются инициирующими бочками. Можно представить, что граница между доменами является периметром кругов вершин, завуалированных в каждой инициирующей бочке. Тогда жидкости заливаются как бы из центров кругов.

Таким образом, просто решен вопрос определения источников жидкостей, который является дорогостоящей операцией при работе со всеми вершинами в графе.

Диффузионные алгоритмы хороши в рамках иерархического подхода при выполнении динамической декомпозиции, поскольку вызывают малую миграцию вершин между процессорами. В статической декомпозиции при использовании на этапах локального уточнения в иерархических алгоритмах они обеспечивают формирование доменов с гладкими границами.

Генетические алгоритмы

Генетические алгоритмы позволяют решать многокритериальные оптимизационные задачи, используя эволюционный подход. Это итерационные методы, моделирующие эволюцию популяций индивидуумов, которые представляют собой решения задачи, выбирая наиболее подходящих индивидуумов для рождения следующих поколений. Генетические алгоритмы сходятся медленно, поэтому в задаче декомпозиции графа они обычно используются в иерархических методах. Разработчики пакета PT-Scotch реализовали на общей памяти иерархический метод, в котором граничный вариант генетического алгоритма вызывается на этапе локального уточнения [45]. В обсуждаемом генетическом алгоритме в задаче декомпозиции графа каждая вершина может находиться в одном из трех состояний: лежать на границе между доменами или в одном из двух доменов. Таким образом, каждый индивидуум в популяции представляется линейным массивом, моделирующим хромосому, который сопоставляет каждой вершине число от нуля до двух. Оператор скрещивания применяется к случайно выбранной позиции в массивах двух индивидуумов. Он меняет местами значения в массивах до данной позиции для порождения двух потомков. Оператор мутации меняет состояния случайно выбранных вершин у некоторых индивидуумов. Поскольку операторы не гарантируют, что после скрещивания или мутации индивидуумы будут отображать возможные решения, затем индивидуумы проверяются на корректность, в результате чего некоторые вершины помещаются на границу между доменами, а другие, наоборот, оттуда удаляются. Индивидуумы оцениваются функцией пригодности, которая комбинирует такие безразмерные величины, как доля вершин на границе между доменами, дисбаланс числа вершин в доменах и доля ребер, соединяющих вершины на границе между доменами. Первое поколение составляется из индивидуумов, мутированных от спроектированного разбиения, и случайных индивидуумов, обеспечивающих генетическое многообразие. Для выбора и мутации индивидуумов используются различные алгоритмы. Затем индивидуумы подбираются в пары по убыванию пригодности, и рождают потомство, сохраняя популяцию постоянной. В статье [45] утверждается, что, хотя данный алгоритм медленнее традиционных иерархических алгоритмов с локальным уточнением KL/FM, он лучше масштабируется по числу процессоров и даёт более качественные

результаты. К тому же их качество может легко варьироваться (в терминах размера популяций и числа поколений) для учета ограничений на время выполнения и качество разбиений. Однако алгоритмы были реализованы на общей памяти, и максимальное число потоков, которое использовалось, – 64, что совсем немного.

Разработчики пакета Jostle реализовали последовательную комбинацию метода эволюционного поиска и иерархического подхода [62]. В данной комбинации конструируется популяция вариантов исходного графа, отличающихся только весами ребер. Иерархический метод используется в качестве оператора, как «черный ящик», для определения пригодности вариантов путем вычисления разбиения каждого из вариантов. Каждое из разбиений может оказаться также хорошим разбиением исходного графа. Популяция эволюционирует либо мутацией индивидуумов, либо их скрещиванием для рождения отличающегося, и предположительно, более пригодного ребенка. Полученные результаты оказались лучше результатов разбиений, полученных пакетами Jostle, METIS и Chaco. Однако небольшие сетки (10^4 вершин) считались часами, и даже днями, поэтому разработчики не стали реализовывать параллельную версию алгоритма.

Таким образом, использование генетических алгоритмов в рамках иерархического подхода дает лучшие разбиения, однако является чересчур затратным по времени.

6.5. Оптимизация характеристических отношений доменов

Недостатком иерархических алгоритмов является образование доменов, границы которых состоят из неоптимальных наборов сегментов [63, 44]. В частности, домены могут оказаться несвязными. Такое ухудшение качества доменов для некоторых задач является критичным. Например, на доменах с длинными границами или сложной конфигурацией алгоритмы решения систем линейных уравнений сходятся за большее число итераций. Диффузионные алгоритмы, встроенные в иерархический метод, позволяют получать домены с небольшими и гладкими границами. Еще одной попыткой решения данной проблемы стал критерий оптимизации характеристических отношений доменов (aspect ratio). Выполнение этого критерия стремится сделать домены шарообразной формы, в результате чего они получаются более компактными. Первые шаги в оптимизации характеристических отношений доменов были в выставлении весов вершин в зависимости от расстояния до центра домена и включении этих весов в функции стоимости итерационных алгоритмов, таких как KL и других.

Варианты вычисления характеристического отношения домена могут быть разными:

$L_{\max} / L_{\min}$ – отношение самого длинного ребра на границе домена к самому короткому [63],

R_o^2 / R_i^2 – отношение площади наименьшего описанного круга к площади наибольшего вписанного,

R_o^2 / S, где S – площадь домена,

$C^2 / 16S$, C – периметр домена, это отношение площади квадрата с периметром C к площади домена,

$C / 4\sqrt{S}$ – отношение периметра домена к периметру квадрата с такой же площадью [64, 65],

$C / 2\sqrt{\pi S}$ – отношение периметра домена к периметру круга с такой же площадью.

$L_{\max} / L_{\min}$ плохо отражает форму у нерегулярных сеток [63]. Круги дают лучшее отношение периметра к площади, но соседние домены становятся вогнутыми. Также основанная на кругах метрика не учитывает зигзаги и вписанные углы.

Разработчики последовательного пакета Party добавили оптимизацию характеристических отношений доменов в пузырьковый метод с балансировкой диффузионным алгоритмом, который будет рассмотрен ниже [63].

Разработчики пакета Jostle вставили оптимизацию характеристических отношений доменов в последовательный иерархический алгоритм [64, 65]. В нем используется отношение $C / 2\sqrt{\pi S}$. Вершины в графе представляют собой элементы сетки, ребра есть между теми вершинами, элементы которых имеют общую грань (дуальный граф). Веса ребер равны размеру поверхности, к которой они относятся. С таким взвешиванием ребер обычный поиск разбиения с минимальным весом разрезанных ребер оптимизирует характеристические отношения доменов. При огрублении стягивание самого тяжелого ребра аналогично стягиванию через самую большую поверхность. В алгоритме локального уточнения выигрыши зависят от характеристических отношений доменов. Описываемый иерархический алгоритм, оптимизирующий характеристические отношения доменов, реализован только в последовательном варианте, и работа оптимизации в параллельной реализации пока не проверена. К тому же вычисление характеристических отношений требует знания геометрии графа, которое не всегда доступно.

6.6. Алгоритмы наращивания доменов

Еще одной попыткой устранения несвязности получаемых доменов стали алгоритмы наращивания доменов.

Алгоритм Фархата

Жадный алгоритм Фархата (Farhat) основан на поиске в ширину. Вершина наименьшей ненулевой степени (имеет наименьшее число соседей) становится инициализирующей вершиной нулевого домена. Остальные вершины для этого домена находятся поиском в ширину. Когда число вершин в домене станет равным $\left\lceil |V| / p \right\rceil$, где p – число доменов, вершина, соседняя с нулевым доменом, становится инициализирующей вершиной первого домена. Остальные домены заполняются подобным образом. Алгоритм выполняет разбиения достаточно быстро и ориентирован на формирование преимущественно связных доменов, но границы доменов могут быть длинными, поскольку при поиске в ширину выбор между соседями вершины может оказаться неоптимальным. Также результаты алгоритма чувствительны к нумерации вершин в графе, то есть при смене нумерации могут быть получены отличающиеся разбиения. Описанный алгоритм реализован в последовательном пакете Party [7]. Похожий алгоритм роста доменов (graph growing partition algorithm) вызывается в иерархическом методе на этапе бисекции в последовательном пакете METIS [26].

Жадный алгоритм с выигрышами

В пакете Party реализован жадный алгоритм, аналогичный алгоритму Фархата, но учитывающий выигрыши от перемещений вершин [7]. Выигрыши высчитываются так же, как и в алгоритме локального уточнения KL. При выборе следующей вершины из всех невыбранных соседей текущего домена выбирается тот сосед, который минимизирует общий вес разрезанных ребер [66].

Похожий алгоритм (greedy graph growing partition algorithm) реализован в пакете METIS [26]. Алгоритм менее чувствителен к выбору инициализирующей вершины, чем ранее рассмотренный алгоритм graph growing partition algorithm пакета METIS, не учитывающий выигрыши от перемещения вершин.

Алгоритм пузырькового роста

Рассмотренные жадные алгоритмы плохо масштабируются на большое число доменов. В алгоритме пузырькового роста (bubble growing algorithm), реализованном в пакете Party [7], рост всех доменов происходит одновременно. Случайные вершины становятся инициирующими для доменов. Домены наращиваются поиском в ширину. Затем для каждого домена вычисляется его центр. Это вершина с наименьшей суммой длин кратчайших путей до всех вершин домена. Центры становятся новыми инициирующими вершинами, и домены заново наращиваются. Процесс прекращается, когда инициирующие вершины перестают меняться или когда возникает конфигурация, полученная ранее. Метод не гарантирует получения сбалансированного разбиения, поэтому после выполнения данного алгоритма для балансировки загрузки приходится

вызывать другие эвристические методы, что делает его дорогостоящим по времени [44]. Как упоминалось ранее, разработчики пакета Party адаптировали алгоритм пузырькового роста к оптимизации характеристических отношений доменов.

В работе [67] алгоритм пузырькового роста получил дальнейшее развитие. Вместо вычисления центров доменов решается система линейных уравнений. Алгоритм, хотя и параллельный, но очень медленный и также формирует несбалансированные разбиения, для балансировки которых приходится вызывать жадные алгоритмы.

6.7. Инкрементный алгоритм декомпозиции графов

Инкрементный алгоритм декомпозиции графов разработан Якобовским М.В. из ИПМ им. М.В.Келдыша РАН [68]. Алгоритм является последовательным. Достоинством инкрементного алгоритма является формирование преимущественно связных доменов. Инкрементный алгоритм не основывается на иерархическом подходе. Наиболее близкими к нему являются алгоритм пузырькового роста и диффузионные алгоритмы. Однако алгоритм пузырькового роста, в отличие от инкрементного алгоритма, не гарантирует получение сбалансированного разбиения. А одним из отличий инкрементного алгоритма от диффузионных алгоритмов является то, что в инкрементном алгоритме происходит освобождение части вершин из плохих доменов и последующий рост доменов.

Вначале выбираются инициализирующие вершины для доменов. Далее разбиение проводится посредством итерационного процесса, на каждом шаге которого выполняются следующие действия:

- Инкрементный рост доменов и диффузное перераспределение вершин между доменами (Рис. 10 и 11).

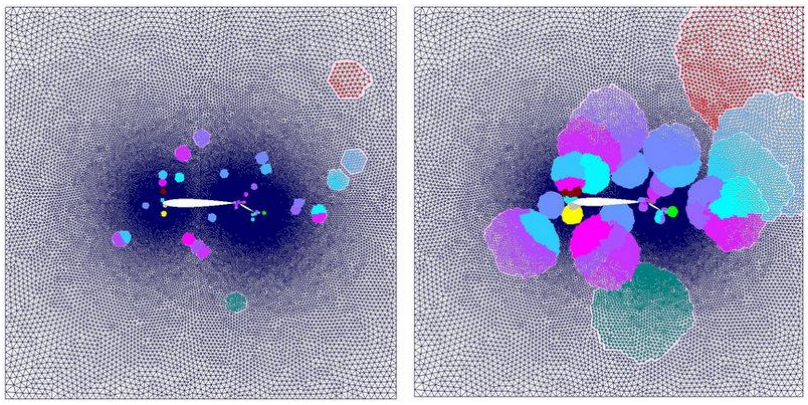

Рис. 10. Инкрементный рост доменов

- Локальное уточнение доменов алгоритмом KL/FM (рис. 11). Границы доменов стали более гладкими. Однако в разбиении присутствуют несвязные домены (желтый домен).

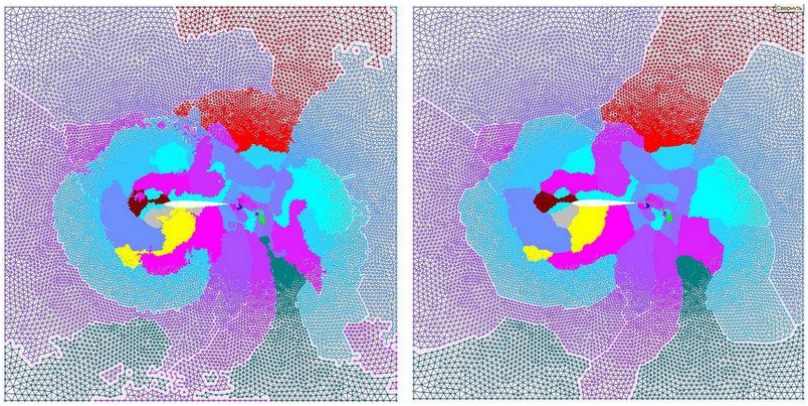

Рис. 11. Диффузное перераспределение вершин между доменами (слева) и локальное уточнение (справа)

- Проверка качества доменов. Если качество доменов соответствует заданному, разбиение считается найденным и происходит выход из цикла, иначе – переход к следующему этапу.
- Освобождение части вершин плохих доменов и переход к первому этапу. Плохими доменами считаются домены, качество которых не соответствует заданному, и соседи таких доменов. В плохих доменах часть вершин освобождается, то есть вновь считается нераспределенной (Рис. 12). Освобождается часть внешних оболочек, а затем все компоненты связности, кроме той, которая содержит наибольшее число вершин.

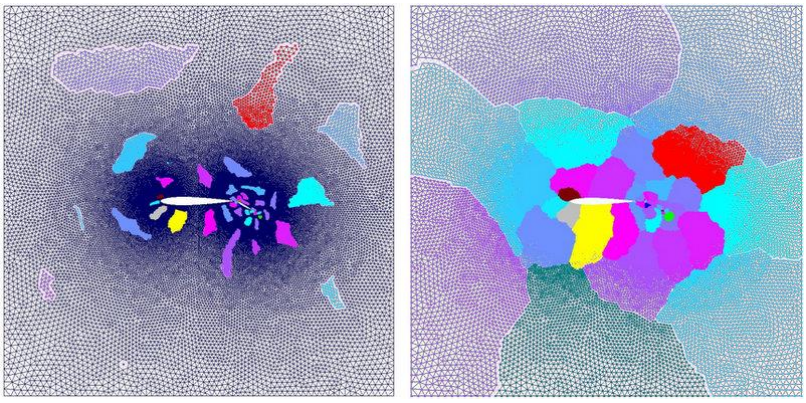

Рис. 12. Освобождение части вершин плохих доменов (слева) и результирующее разбиение (справа)

В результирующем разбиении на рис. 12 видно, что теперь желтый домен является связным.

Качество доменов проверяется следующим образом. Все вершины, расположенные на геометрической границе графа (вершины, аппроксимирующие границу моделируемой области) и на границах между доменами, считаются принадлежащими первой оболочке своего домена. В каждом домене вершинами второй оболочки считаются соседи вершин из

первой оболочки данного домена, которые также принадлежат этому домену и не попали в первую оболочку. Остальные оболочки вычисляются аналогично. На рис. 13 представлены оболочки домена при условии, что вся сетка принадлежит одному домену. Проверяется связность оболочек каждого домена, и вычисляется номер несвязной оболочки с наименьшим номером (номер первой несвязной оболочки) в каждом домене. Хорошими считаются те домены, номер первой несвязной оболочки в которых не меньше заданного порогового значения.

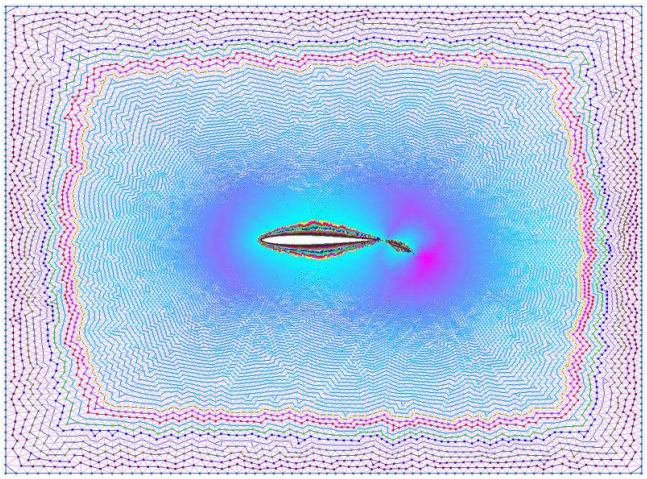

Рис. 13. Оболочки домена

6.8. Параллельный инкрементный алгоритм декомпозиции графов

Параллельный инкрементный алгоритм декомпозиции графов пакета GridSpiderPar (Головченко Е.Н. и Якобовский М.В., ИПМ им. М.В. Келдыша РАН) основан на последовательном инкрементном алгоритме декомпозиции графов [19, 24, 25]. Его достоинством также является формирование преимущественно связных доменов.

Параллельный инкрементный алгоритм предполагает выполнение следующих этапов:

- Геометрическое распределение вершин по процессорам с помощью параллельного алгоритма геометрической декомпозиции сеточных данных пакета GridSpiderPar, описанного ранее.
- Перераспределение малых блоков вершин (Рис. *14*).

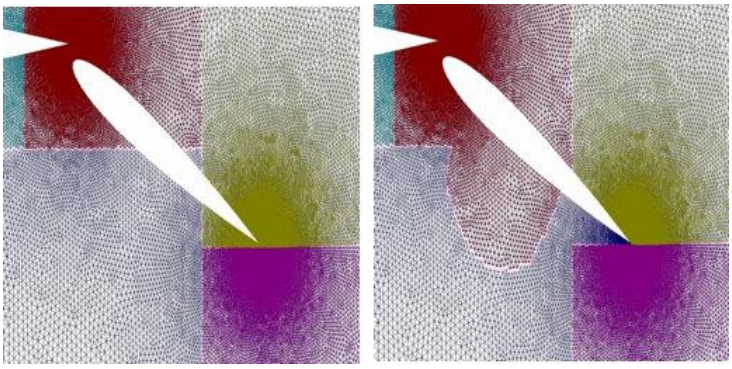

Рис. 14. Фрагменты геометрического распределения вершин по процессорам (слева) и перераспределения малых блоков вершин (справа)

- Локальное разбиение вершин на каждом из процессоров на домены последовательным инкрементным алгоритмом декомпозиции графов. На рис. 15 слева четко выражены границы между процессорами, домены не пересекают эти границы.
- Перераспределение плохих групп доменов. Сбор каждой группы плохих доменов на одном процессоре.
- Повторное локальное разбиение плохих групп доменов инкрементным алгоритмом декомпозиции графов. На рис. 15 справа видно, что после перераспределения плохих групп доменов и их повторного разбиения домены вышли за границы между процессорами.

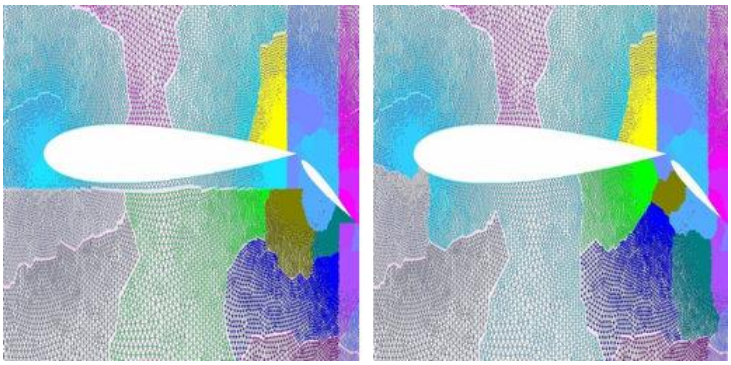

Рис. 15. Локальное разбиение (слева), сбор плохих групп доменов и их повторное разбиение (справа)

Параллельный инкрементный алгоритм декомпозиции графов реализован на MPI.

7. Алгоритмы многопоточного разбиения

В пакете MT-METIS реализован алгоритм многопоточного разбиения графов (Multi-Threaded Graph Partitioning) [69]. Алгоритм является потоковой реализацией иерархического алгоритма пакета ParMETIS.

На этапе огрубления вершины графа разделяются между потоками, и каждый поток отвечает за огрубление своих вершин. Реализованы несколько вариантов отмечания вершин, ребра между которыми будут стянуты:

- При fine-grain matching (мелкозернистое отмечание) используется фиксированное число семафоров (или других запирающих средств), большее числа потоков. Так как вершин слишком много, поставить семафор на каждую вершину невозможно. Хэш-функция отображает множество вершин на один и тот же семафор. Семафоры ставятся только на запись. Доступ к семафорам происходит в восходящем порядке, во избежание deadlock-ов.
- При multi-pass matching (отмечание в несколько проходов) предпочтение отдается отмечанию с вершинами, обрабатываемыми этим же потоком. Запросы на отмечание чужих вершин помещаются в буфера запросов для каждого потока. На следующем проходе каждый поток обрабатывает свои буфера запросов, отклоняя или подтверждая их. После нескольких проходов неотмеченные вершины отмечаются сами с собой.
- При unprotected matching (незащищенное отмечание) записи выполняются в незащищенном режиме. После того как все записи выполнены, каждый поток проходит по списку своих локальных вершин, и если у пары вершин отметки оказываются некорректными (в пару к первой вершине отмечена вторая, а в пару ко второй – третья), вершины будут стянуты сами с собой. Поскольку вершин очень много, таких несимметричных отмечаний оказалось 0.001% на более точных графах и 0.13% на огрубленных.

Далее вершины нового огрубленного графа распределяются между потоками, и каждый поток отвечает за формирование списков соседей своих вершин.

На этапе разбиения огрубленного графа реализовано несколько вариантов:

- При parallel bisectioning (параллельная бисекция) каждый поток разбивает огрубленный граф на две части. Выбирается наилучшее разбиение. Затем половина потоков разбивает одну часть, вторая половина – другую.
- При parallel k-sectioning (параллельное разбиение на k частей) каждый поток независимо выполняет полное разбиение графа. Затем выбирается наилучшее разбиение.

На этапе разогрубления выполняется жадное уточнение, при котором граничные вершины разделяются между потоками, и каждый поток помещает свои вершины в свою очередь приоритетов. Далее есть несколько вариантов:

- При fine-grain refinement (мелкозернистое уточнение) вершина достается из очереди, выбирается ее наилучшее перемещение. Блокируются с хэш-функцией вершина, ее соседи, ее домен и домен, в который она перейдет. Снова проверяются условия уменьшения edge-

cut (веса разрезанных ребер) и сохранения баланса весов доменов. Выполняется перемещение, вся информация обновляется, и снимаются блокировки. Во избежание узких мест при обновлении весов доменов число доменов должно быть больше числа потоков.

- При coarse-grain refinement (крупнозернистое уточнение) вершины перемещаются только в одном направлении. У каждого потока есть буфер обновлений. Поток обновляет информацию своих вершин, а для чужих помещает новую информацию в буфера обновлений обрабатывающих их потоков. После того как каждый поток удалит из очереди приоритетов определенное количество вершин, все потоки передают предполагаемые веса доменов после их перемещений. Часть перемещений отменяется для сохранения баланса. Затем из буферов обновлений обновляется локальная информация. Все повторяется до тех пор, пока очереди приоритетов не окажутся пустыми.

Потоки можно создавать на отдельный блок (например, огрубение), а потом уничтожать, или на работу всей программы.

Данные могут быть распределены между потоками перед каждым блоком или сразу на время выполнения всей программы. Во втором случае равномерность распределения данных между потоками может нарушаться. Например, когда при огрублении графа у потока остаются стянутые вершины его части исходного графа.

Алгоритм реализован на OpenMP. Заявляется, что MT-METIS оказался в среднем в два раза быстрее ParMETIS и PT-Scotch на графах с 10^7 вершин, а незащищенное отмечание сильно снизило время огрубления, самой затратной стадии иерархического алгоритма. Также MT-METIS использует меньше памяти, чем алгоритмы, реализованные на MPI.

8. Алгоритмы динамической балансировки

В параллельных расчетах, использующих динамические адаптивные сетки, когда в процессе расчета сетка уточняется или разуточняется в зонах интереса или в местах возникновения больших ошибок, в процессе расчета требуется периодическое переразбиение расчетной сетки, что определяет актуальность разработки алгоритмов динамической балансировки загрузки на основе параллельных масштабируемых методов рациональной декомпозиции сеток.

Переразбиение (динамическая балансировка загрузки) должно не только балансировать объем вычислений на процессорах и минимизировать коммуникационную нагрузку на процессоры во время счета, но и минимизировать объем перераспределяемых данных между процессорами для удовлетворения новому разбиению. Существует несколько подходов к решению этой задачи [70]. В первом выполняется поиск абсолютно нового разбиения расчетной сетки и его отображение на существующее разбиение, что позволяет найти наилучшее разбиение, но объем перераспределяемых данных между процессорами может оказаться очень большим. Во втором подходе

новое разбиение ищется, используя уже существующее, что позволят минимизировать объем перераспределяемых данных, но качество разбиения может оказаться хуже, чем при первом подходе. Во втором подходе обычно используются диффузионные алгоритмы. Использование геометрических методов в первом подходе зачастую позволяет получить новые разбиения, требующие меньшее перераспределение данных. Причиной является детерминированность геометрических методов.

Метод адаптивного переразбиения сетки пакета ParMETIS является параллельной реализацией объединенного алгоритма переразбиения [70]. Как и в других иерархических алгоритмах, в нем выполняются поэтапное огрубление графа, декомпозиция самого маленького из полученных графов и отображение разбиения на предыдущие графы с периодическим локальным уточнением границ доменов. При огрублении стягиваются только ребра с вершинами из одного домена. Декомпозиция огрубленного графа вычисляется двумя способами, оптимизированными вариантами алгоритмов scratch-remap и глобальной диффузии (global diffusion) [71]. Для каждого из разбиений определяется его качество на основе вычисленного ITR Фактора, параметра, определяющего отношение между временем межпроцессорного взаимодействия при расчете задачи и временем перераспределения данных в соответствии с новым разбиением. Разбиение наилучшего качества отображается на предыдущие графы и локально уточняется. Разработчики пакета ParMETIS объясняют использование двух алгоритмов тем, что для каких-то задач один алгоритм дает лучшие результаты, для каких-то – другой.

В параллельном алгоритме динамического разбиения графов адаптивных неструктурированных сеток пакета Jostle также сначала выполняется огрубление графа, а потом уже его разбиение. Для разбиения алгоритмом Hu и Blake вычисляются веса вершин, которые должны быть перемещены между доменами [51]. Затем определяются конкретные вершины, которые будут перемещены. Вершины определяются в зависимости от выигрышей от их перемещения между доменами, от того, насколько изменится вес разрезанных ребер в результате их перемещения.

В пакете Zoltan разработан алгоритм динамической балансировки, использующий модель гиперграфа переразбиения (Repartitioning Hypergraph Model for Dynamic Load Balancing) [72]. Гиперграфы более точно моделируют актуальную коммуникационную нагрузку и имеют большую область применения (могут моделировать несимметричные и неквадратные системы). Эпохой называется период между перебалансировками. Эпоха может состоять из нескольких итераций.

Гиперграф переразбиения формируется следующим образом. В гиперграфе эпохи каждое гиперребро умножается на количество итераций в данной эпохе. Для каждого домена добавляется вершина с нулевым весом. Для каждой вершины гиперграфа добавляется гиперребро в вершину домена, если она оказалась в этом домене на предыдущей итерации. Миграционные гиперребра

отвечают за миграционные затраты, если эти вершины будут перенесены в другие домены, и их вес – размер данных, ассоциированных с вершинами. Новые вершины связываются с вершинами доменов, в которых они были созданы.

9. Заключение

Рассмотрены различные модели декомпозиции графов. Рассмотрены алгоритмы декомпозиции, реализованные в последовательных пакетах METIS, Scotch, Jostle, Chaco, Party и параллельных пакетах GridSpiderPar, ParMETIS, PT-Scotch, Zoltan и Jostle. Описаны различные методы декомпозиции сеток и разбиения графов, приведены их достоинства и недостатки. В частности, методы геометрической декомпозиции являются менее затратными по времени выполнения и количеству используемой памяти, также они формируют разбиения с меньшим дисбалансом весов доменов. Однако методы геометрической декомпозиции не учитывают коммуникационную нагрузку сетки, поэтому обычно они используются на этапе предварительного разбиения. Для получения качественных разбиений вызываются методы разбиения графов.

Численные сравнения разбиений, полученных методами различных пакетов, как между собой, так и на задачах газовой динамики, представлены в статьях [24, 25].

10. Библиографический список

1. B. H. V. Topping. Mesh Partitioning for Parallel and Distributed Computations // Lecture in the course High Performance Computations for Engineering, University of Pecs, Hungary. 2009.
2. Bruce Hendrickson, Tamara G. Kolda. Graph partitioning models for parallel computing. Parallel Computing, 26, 2000, 1519-1534.
3. Bruce Hendrickson. Graph Partitioning and Parallel Solvers: Has the Emperor No Clothes? Sandia National Labs, Albuquerque.
4. Bruce Hendrickson and Tamara G. Kolda. Partitioning Sparse Rectangular Matrices for Parallel Computations of Ax and A^T v. Applied Parallel Computing in Large Scale Scientific and Industrial Problems, PARA’98, number 1541 in Lecture Notes in Computer Science, Springer, Berlin, 1998, pp. 239 - 247.
5. George Karypis and Vipin Kumar. Multilevel Algorithms for Multi-Constraint Graph Partitioning. Technical Report # 98-019, Department of Computer Science, University of Minnesota, Minneapolis, MN, 1998.
6. Kirk Schloegel, George Karypis and Vipin Kumar. Parallel static and dynamic multi-constraint graph partitioning. Concurrency and Computation: Practice and Experience, 2002, 14, 219-240.

7. Robert Preis and Ralf Diekmann. PARTY – A Software Library for Graph Partitioning. Advances in Computational Mechanics with Parallel and Distributed Processing, CIVIL-COMP PRESS, 1997, 63 – 71.
8. Bruce Hendrickson and Robert Leland. The Chaco User's Guide, Version 2.0. SAND95-2344, Unlimited Release, 1995.
9. Robert Preis. The PARTY Graphpartitioning – Library. User Manual – Version 1.99 // Universitat Paderborn, Germany, October 13, 1998, 1 – 8.
10. Erik Boman, Karen Devine, Umit Catalyurek, Doruk Bozdag, Bruce Hendrickson, William F. Mitchell, James Teresco // Zoltan: Parallel Partitioning, Load Balancing and Data-Management Services. Developer's Guide, Version 3.3. Sandia National Laboratories, Copyright©2000-2010, http://www.cs.sandia.gov/Zoltan/dev_html/dev.html .
11. Horst D. Simon. Partitioning of Unstructured Problems for Parallel Processing. Numerical Aerodynamic Simulation (NAS) Systems Division, NASA Ames Research Center, 1994.
12. Roy D. Williams. Performance of Dynamic Load Balancing Algorithms for Unstructured Mesh Calculations. Concurrent Supercomputing Facility, California Institute of Technology, 1990.
13. Ho-Kwok Dai and Hung-Chi Su. Approximation and Analytical Studies of Inter-clustering Performances of Space-Filling Curves. Discrete Mathematics and Theoretical Computer Science AC, 2003, 53-68.
14. William Gilbert. A Cube-filling Hilbert Curve // Mathematical Intelligencer. 1984. 6(3). 78.
15. Karen Devine, Erik Boman, Robert Heaphy, Bruce Hendrickson and Courtenay Vaughan. Zoltan Data Management Services for Parallel Dynamic Applications. Computing in Science & Engineering, 2002, 90 – 97.
16. George Karypis. METIS – A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices. Version 5.0 // University of Minnesota, Minneapolis, August 4, 2011.
17. George Karypis, Kirk Schloegel and Vipin Kumar. ParMETIS – Parallel Graph Partitioning and Sparse Matrix Ordering Library, Version 3.1 // University of Minnesota, Minneapolis, 2003.
18. George Karypis, Kirk Schloegel. ParMETIS – Parallel Graph Partitioning and Sparse Matrix Ordering Library, Version 4.0 // University of Minnesota, Minneapolis, August 4, 2011.
19. Головченко Е.Н. Комплекс программ параллельной декомпозиции сеток // Вычислительные методы и программирование. 2010. Т. 11. 360-365.
20. Головченко Е.Н. Параллельный пакет декомпозиции больших сеток // Математическое моделирование. 2011. Т. 23. № 10. 3-18.
21. Бухановский А.В., Марьин С.В., Князьков К.В., Сиднев А.А., Жабин С.Н., Баглий А.П., Штейнберг Р.Б., Шамакина А.В., Воеводин В.В., Головченко Е.Н., Фалалеев Р.Т., Духанов А.В., Тарасов А.А., Шамардин Л.В., Моисеенко А.И. Результаты реализации проекта «Мобильность молодых

ученых» в 2010 году: развитие функциональных элементов технологии iPSE и расширение состава прикладных сервисов // Известия высших учебных заведений. Приборостроение. 2011. Т. 54. №10. 80 - 86.
22. Е.Н. Головченко. Разбиение больших сеток // Вестник Нижегородского университета им. Н.И. Лобачевского. Серия «Информационные технологии». Нижний Новгород: Издательство ННГУ. 2012. № 5(2). С. 309-315.
23. Evdokia Golovchenko, Elizaveta Dorofeeva, Irina Gasilova and Alexey BOLDAREV. Numerical Experiments with New Algorithms for Parallel Decomposition of Large Computational Meshes // Parallel Computing: Accelerating Computational Science and Engineering (CSE). Advances in Parallel Computing. IOS Press. 2014. Vol. 25. 441 - 450.
24. Evdokia N. Golovchenko, Marina A. Kornilina, Mikhail V. Yakobovskiy. Algorithms in the parallel partitioning tool GridSpiderPar for large mesh decomposition // Proceedings of the 3rd International Conference on Exascale Applications and Software (EASC 2015). University of Edinburgh. 2015. 120-125.
25. Е.Н. Головченко, М.В. Якобовский. Пакет параллельной декомпозиции больших сеток GridSpiderPar // Вычислительные методы и программирование. 2015. Т. 16. 507-517.
26. G. Karypis, V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM J. Sci. Comput. 1998. Vol 20, No 1, pp 359-392.
27. Якобовский М.В. Обработка сеточных данных на распределенных вычислительных системах. // Вопросы атомной науки и техники. Сер. Математическое моделирование физических процессов. 2004. Вып.2. с. 40-53.
28. B. Hendrickson, R. Leland. An improved spectral graph partitioning algorithm for mapping parallel computations // SIAM J. Sci. Comput. 1995. 16(2). 452-469.
29. C. Farhat, H. D. Simon. TOP/DOMDEC – a Software Tool for Mesh Partitioning and Parallel Processing // Report RNR-93-011. NASA Ames Research Center. 1993.
30. H. D. Simon, A. Sohn, R. Biswas. HARP: A dynamic spectral partitioner // Journal of Parallel and Distributed Computing. 1998. 50. pp. 88-103.
31. A. Sohn, H. Simon. S-HARP: A Parallel Dynamic Spectral Partitioner // NASA JOVE Program, NASA Ames Research Center.
32. A. Sohn, H. Simon. JOVE: A Dynamic Load Balancing Framework for Adaptive Computations on an SP-2 Distributed-Memory Multiprocessor // NJIT CIS Technical Report 94-60. 1994.
33. Hendrickson B., Leland R. A Multilevel Algorithm for Partitioning Graphs // SAND93-1301, Unlimited Release, 1993.
34. C. Walshaw, M. Cross. Mesh Partitioning: A Multilevel Balancing and Refinement Algorithm. SIAM J. Sci. Comput., Vol. 22, № 1, 2000, pp. 63 – 80.

35. Y. F. Hu, R. J. Blake, D. R. Emerson. An optimal migration algorithm for dynamic load balancing // Concurrency: Practice and Experience. 1998. 10. 467-483.
36. G. Karypis, V. Kumar. METIS - Unstructured Graph Partitioning and Sparse Matrix Ordering System // University of Minnesota, Department of Computer Science, Minneapolis. 1995.
37. G. Karypis, V. Kumar. Multilevel k-way Partitioning Scheme for Irregular Graphs // Journal of Parallel and Distributed Computing. 1998. 48. 96-129.
38. C. Walshaw, M. Cross. JOSTLE: parallel multilevel graph-partitioning software – an overview // School of Engineering, University of Swansea.
39. A. Gupta. Fast and Effective Algorithms for Graph Partitioning and Sparse Matrix Ordering // IBM Research Report. RC 20496 (90799). Computer Science / Mathematics. 1996.
40. A. Gupta. WGPP: Watson Graph Partitioning (and sparse matrix ordering) Package. Users Manual: Version 3.2 // IBM Research Report. RC 20453 (90427). Computer Science / Mathematics. 1996.
41. C. Walshaw, M. Cross. Parallel Mesh Partitioning on Distributed Memory Systems. Invited lecture. In: Proc. Parallel & Distributed Computing for Computational Mechanics, Weimar, Cermany, 1999.
42. Francois Pellegrini. Scotch and libScotch 5.1 User's Guide // Bacchus team, INRIA Bordeaux Sud-Ouest, ENSEIRB & LaBRI, UMR CNRS 5800, Universite Bordeaux I. 2010. 1-133.
43. Francois Pellegrini. PT-Scotch and libScotch 5.1 User's Guide. Bacchus team, INRIA Bordeaux Sud-Ouest, ENSEIRB & LaBRI, UMR CNRS 5800, Universite Bordeaux I, Talence, France, 2010, 1- 78.
44. Francois Pelegrini. A parallelizable multi-level banded diffusion scheme for computing balanced partitions with smooth boundaries // ENSEIRB, LaBRI and INRIA Futurs, Universite Bordeaux I.
45. Cedric Chevalier, Francois Pellegrini. Improvement of the Efficiency of Genetic Algorithms for Scalable Parallel Graph Partitioning in a Multi-Level Framework // LaBRI and INRIA Futurs, Universite Bordearux I.
46. S. Barnard. PMRSB – Parallel Multilevel Recursive Spectral Bisection // Cray Research, Inc.
47. S. T. Barnard, H. D. Simon. A Fast Multilevel Implementation of Recursive Spectral Bisection for Partitioning Unstructured Problems // Report RNR-92-033. NAS Systems Division. Applied Research Branch. 1992.
48. G. Karypis, V. Kumar. Parallel Multilevel k-Way Partitioning Scheme for Irregular Graphs // SIAM REVIEW. 1999. Vol. 41. No 2. 278-300.
49. G. Karypis, V. Kumar. A Coarse-Grain Parallel Formulation of Multilevel k-way Graph Partitioning Algorithm // Proceeding of the 8th SIAM Conference on Parallel Processing for Scientific Computing.
50. K. Schloegel, G. Karypis, V. Kumar. Parallel Multilevel Algorithms for Multi-constraint Graph Partitioning // Euro-Par 2000, LNCS 1900. 2000. 296-310.

51. C. Walshaw,M. Cross and M.G. Everett. Parallel dynamic graph-partitioning for unstructured meshes // Centre for Numerical Modelling and Process Analysis, University of Greenwich, Mathematics Research Report 97/IM/20. March 27, 1997. 1-10.
52. C. Walshaw, M. Cross, M. G. Everett. Parallel dynamic graph-partitioning for unstructured meshes // Mathematics Research Report 97/IM/20, Centre for Numerical Modelling and Process Analysis, University of Greenwich. 1997.
53. C. Walshaw, M. Cross, M. G. Everett. Parallel Dynamic Graph Partitioning for Adaptive Unstructured Meshes // Journal of Parallel and Distributed Computing. 1997. 47. 102-108.
54. K. D. Devine, E. G. Boman, R. T. Heaphy, R. H. Bisseling, U. V. Catalyurek. Parallel Hypergraph Partitioning for Scientific Computing // Tech. Report, Computer Science Research Institute, Sandia.
55. R. Biswas, R. C. Strawn. A new procedure for dynamic adaptation of three-dimensional unstructured grids // Applied Numerical Mathematics. 1994. 13. 437-452.
56. K. Schloegel, G. Karypis, V. Kumar. Multilevel diffusion schemes for repartitioning of adaptive meshes // Technical Report TR 97-013, University of Minnesota, Department of Computer Science. 1997. http://www.cs.umn.edu/karypis.
57. K. Schloegel, G. Karypis, V. Kumar. Parallel Multilevel Diffusion Algorithms for Repartitioning of Adaptive Meshes // Technical Report TR 97-014, University of Minnesota, Department of Computer Science. 1997.
58. Y. F. Hu, R. J. Blake. An optimal dynamic load balancing algorithm // Technical Report DL-P-95-011, Daresbury Laboratory, Warrington, UK. 1995.
59. Francois Pellegrini. PT-Scotch and libPTScotch 6.0 User's Guide // Bacchus team, INRIA Bordeaux Sud-Ouest, Universite Bordeaux 1 & LaBRI, UMR CNRS 5800, Talence, France, 2012, 1- 90.
60. J. Horn, N. Nafpliotis, D. E. Goldberg. A niched Pareto genetic algorithm for multi-objective optimization // IEEE World Congress on Computational Intelligence. 1994. 1. 82-87.
61. P. Moscato. On evolution, search, optimization, genetic algorithms and martial arts, towards memetic algorithms // Technical Report 826, California Institute of Technology, Pasadena, CA 91125, U.S.A. 1989.
62. A. J. Soper, C. Walshaw, M. Cross. A Combined Evolutionary Search and Multilevel Optimization Approach to Graph-Partitioning // Journal of Global Optimization. 2004. 29. 225-241.
63. R. Diekmann, R. Preis, F. Schlimbach, C. Walshaw. Aspect Ratio for Mesh Partitioning // Euro-Par'98, LNCS 1470. 1998. 347-351.
64. C. Walshaw, M. Cross, R. Diekmann, F. Schlimbach. Multilevel Mesh Partitioning for Optimizing Domain Shape // The International Journal of High Performance Computing Applications. 1999. Vol 13. No 4. pp. 334-353.

65. C. Walshaw, M. Cross, R. Diekmann, F. Schlimbach. Multilevel Mesh Partitioning for Optimizing Subdomain Aspect Ratio // Developments in Computational Mechanics with High Performance Computing, Civil-Comp Press, Edinburgh. 1999. 9-19.
66. R. Preis. Efficient partitioning of very large graphs with the new and powerful helpful-set heuristic // Diplomarbeit, Universitat-GH Paderborn, Germany. 1994.
67. H. Meyerhenke, S. Schamberger. Balancing parallel adaptive FEM computations by solving systems of linear equations // Proc. Europar. 2005. 209-219.
68. М. В. Якобовский. Инкрементный алгоритм декомпозиции графов. Вестник Нижегородского университета им. Н.И.Лобачевского. Серия «Математическое моделирование и оптимальное управление», Вып. 1(28). Нижний Новгород: Издательство ННГУ, 2005, с. 243-250.
69. Dominique LaSalle and George Karypis. Multi-Threaded Graph Partitioning // Department of Computer Science & Engineering. University of Minnesota. Minneapolis, Minnesota 55455, USA.
70. Kirk Schloegel, George Karypis, Vipin Kumar. A Unified Algorithm for Load-balancing Adaptive Scientific Simulations // Department of Computer Science and Engineering. University of Minnesota. 0-7803-9802-5/2000/$10.00.
71. Kirk Schloegel, George Karypis, Vipin Kumar. Wavefront Diffusion and LMSR: Algorithms for Dynamic Repartitioning of Adaptive Meshes // TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 12, NO. 5, MAY 2001.
72. Umit V. Catalyurek, Erik G. Boman, Karen D. Devine, Doruk Bozdaˇ , Robert Heaphy, Lee Ann Riesen. Hypergraph-based Dynamic Load Balancing for Adaptive Scientific Computations // 1-4244-0910-1/07/$20.00 c
2007 IEEE.
73. Якобовский М.В. Параллельные алгоритмы сортировки больших объемов данных. В кн.: Фундаментальные физико-математические проблемы и моделирование технико-технологических систем: Сб. науч. тр., Выпуск 7 / Под ред. Л.А. Уваровой. – М.: Издательство "Янус-К", 2004. – с. 235-249.