УДК 519.178: 519.688 + 519.61 / 673

БЫСТРЫЙ ВЫСОКОПРОИЗВОДИТЕЛЬНЫЙ АЛГОРИТМ ДЛЯ РАЗДЕЛЕНИЯ НЕРЕГУЛЯРНЫХ ГРАФОВ

Бувайло Д.П., к.ф.-м.н., Толок В.А., д.т.н., профессор

Запорожский государственный университет

В последнее время большое число исследователей занимается новым классом алгоритмов для разделения графа. Эти алгоритмы уменьшают размер графа, стягивая его узлы и ребра, делят меньший граф и затем рафинируют последовательность грубых графов, чтобы построить разделение для первоначального графа [4,26]. Из ранних работ было ясно, что эти многоуровневые методы сулят большие возможности. Однако не было известно, могут ли они быть реализованы так, чтобы произвести разделение высокого качества для большинства графов, возникающих в различных прикладных областях. Разрабатываемый пакет программ позволяет эффективно находить сепараторы приемлемого качества. В этой статье описывается несколько различных применяемых комбинаций эвристик для всех трех стадий: огрубление, разделение самого грубого графа, восстановление.

Ключевые слова: разделение графа, многоуровневые методы разделения графа, спектральные методы разделения графа, перестановки, Kernighan-Lin-эвристики, разреженные матричные алгоритмы.

Бувайло Д.П., Толок В.О. ШВИДКИЙ ВИСОКОПРОДУКТИВНИЙ АЛГОРИТМ ДЛЯ ПОДІЛУ НЕРЕГУЛЯРНИХ ГРАФІВ / Запорізький державний університет, Україна.

Останнім часом велике число дослідників займається новим класом алгоритмів для поділу графу. Ці алгоритми зменшують розмір графу, стягаючи його вузли і ребра, поділяють менший граф, і потім рафінують послідовність грубих графів, щоб побудувати поділ для первісного графа [4,26]. З ранніх робіт було ясно, що ці багаторівневі методи обіцяють великі можливості. Однак не було відоме, чи можуть вони бути реалізовані так, щоб зробити поділ високої якості для більшості графів, що виникають у різних прикладних областях. Розроблюваний пакет програм дозволяє ефективно знаходити сепаратори прийнятної якості. У цій статті описується кілька різних застосовуваних комбінацій евристик для всіх трьох стадій: огрубіння, поділ самого грубого графа, відновлення.

Ключові слова: поділ графа, багаторівневі методи поділу графу, спектральні методи поділу графу, перестановки, Kernighan-Lin-евристики, розріджені матричні алгоритми.

Buvailo D.P., Tolok V.A. FAST HIGH-PERFORMANCE ALGORITHM FOR IRREGULAR GRAPHS PARTITIONING / Zaporizhzhya state university, Ukraine.

Recently, a number of researchers have investigated a class of graph partitioning algorithms that reduce the size of the graph by collapsing vertices and edges, partition the smaller graph, and then uncoarsen it to construct a partition for the original graph [4, 26]. From the early work it was clear that multilevel techniques held great promise; however, it was not known if they could be made to consistently produce high quality partitions for graphs arising in a wide range of application domains. The software package allows effectively find separators of acceptable quality. In this paper some various used combinations of heuristics for all three stages are described: coarsening, most rough graph bisection, uncoarsening.

Key words: graph partitioning, multilevel partitioning methods, spectral partitioning methods, fill reducing ordering, Kernighan-Lin heuristic, sparse matrix algorithms.

РАЗДЕЛЕНИЕ ГРАФА

Разделение графа – важная проблема, которая имеет обширные применения во многих областях, включая научные вычисления, метод конечных элементов, линейное программирование, проектирование СБИС, транспортные задачи, планирование задач. Проблема состоит в том, чтобы разделить узлы графа на p приблизительно равных частей так, чтобы число ребер, соединяющих узлы из разных частей, было минимально. Проблема разделения графа на k частей определяется следующим образом: имеем граф $G=(V,E)$, $|V|=n$, делим V на k подмножеств, $V_1,V_2,...,V_k$ так, что $V_i \cup V_j = 0$ для $i \neq j$, $|V_i| \approx n/k$, и $\Sigma|V_i|=|V|$, а число ребер из E, которые инцидентны узлам из различных подмножеств, минимально. Проблема разделения графа на k частей может быть естественно расширена на взвешенные графы, которые имеют веса, ассоциированные с узлами и ребрами графа. В этом случае цель состоит в том, чтобы разделить узлы на k непересекающихся подмножеств так, что суммы весов узлов в каждом подмножестве приблизительно равны, а сумма весов ребер, которые инцидентны узлам из разных подмножеств, минимальна. Разделение V на k частей обычно представляется вектором разделения P длины n, таким, что для каждого узла $v \in V$, $P[v]$ – целое число между 1 и k, указывающее подмножество, которому принадлежит узел v. Для разделителя P число (или сумма весов) ребер, которые инцидентны узлам различных подмножеств, называется весом разделителя. Приведем примеры задач, требующих разделения графа.

Эффективное выполнение многих параллельных алгоритмов обычно требует решения проблемы разделения графа, где узлы представляют вычислительные задачи, а ребра представляют межзадачный обмен данными. Узлам назначаются веса, пропорциональные количеству вычислений, выполненного каждой задачей, а ребрам – веса, пропорциональные количеству данных, которые должны быть переданы между задачами. Разделение этого графа вычислений на k частей может использоваться, чтобы распределить задачи между k процессорами. Так как разделение назначает на каждый процессор задачи, у которых суммарные веса одинаковы, то работа будет сбалансирована между k процессорами. Кроме того, поскольку алгоритм минимизирует вес сепаратора (ориентируясь на требование сбалансированной нагрузки), общий объем межзадачных коммуникаций также будет минимизирован.

Рекурсивное деление пополам применяется для поиска переупорядочения, сокращающего заполнение при разложении разреженной матрицы [12,17,22]. Эти алгоритмы обычно упоминаются как алгоритмы ***вложенных сечений***. Вложенные сечения рекурсивно разделяют граф на почти равные половины, удаляя узлы сепаратора, пока не получено желательное число разделений. Один путь получения узлового сепаратора состоит в том, чтобы сначала получить деление пополам графа и затем вычислять узловой сепаратор из реберного сепаратора. Узлы графа нумеруются так, что на каждом уровне рекурсии, узлы сепаратора нумеруются после узлов в половинках графа. Эффективность и сложность алгоритма вложенных сечений зависит от алгоритма вычисления сепаратора. Вообще, маленькие сепараторы приводят к меньшему заполнению.

Проблема разделения графа NP-полна [12]. Однако разработано много приближенных алгоритмов для поиска хорошего разделения за приемлемое время. Условно эти алгоритмы можно разделить на три класса.

Спектральные методы разделения производят хорошие разделения для широкого класса проблем, и используются весьма интенсивно [17,16,14]. Однако эти методы очень дороги, так как требуют вычисления собственного вектора, соответствующего второму (меньшему) собственному значению (Fiedler-вектор) матрицы Лапласа. Время выполнения спектральных методов может быть уменьшено, если вычисление Fiedler-вектора реализовать, используя многоуровневый алгоритм [2]. Этот многоуровневый спектральный алгоритм деления пополам (MSB) обычно может ускорять спектральные методы разделения на порядок без потерь в качестве разделителя. Однако даже MSB может требовать большое количество времени. В частности в параллельном прямом решателе линейных систем, время для вычисления переупорядочения неизвестных с использованием MSB может быть на несколько порядков выше, чем время, использованное параллельным алгоритмом разложения, и таким образом время переупорядочения может доминировать в полном времени решения проблемы [18].

Второй класс методов использует для нахождения хорошего разделителя геометрическую информацию о графе. ***Геометрические алгоритмы*** разделения [13,18,19] имеют тенденцию быть быстрыми, но часто выдают разделения, качество которых хуже, чем у других методов. Среди наиболее известных из этих схем – алгоритм, описанный в [20,21]. Этот алгоритм производит разделения, которые вероятно находятся в пределах теоретических границ, существующих для некоторых специальных классов графов (включая графы, возникающие в конечноэлементных применения). Однако, из-за случайной природы этих алгоритмов, часто требуются многократные повторения (от 5 до 50), чтобы получить решения, сопоставимые по качеству со спектральными методами. Многократные повторения увеличивают полное время работы [16], но оно остается все же существенно меньше, чем у спектральных методов. Геометрические алгоритмы разделения графа применимы только, если известны координаты узлов графа. Во многих предметных областях (например, линейном программировании, проектировании СБИС), нет никакой геометрической информации, связанной с графом. Недавно был предложен алгоритм генерации координат узлов графа [6] на основе спектральных методов. Но эти методы намного дороже и превышают пределы, допустимые для полного времени работы алгоритма разделения графа.

Третий класс алгоритмов разделения графа уменьшает размер графа до приемлемой величины, огрубляя граф путем стягивания его узлов и ребер, делит меньший граф, и затем уточняет разделитель, чтобы построить разделение для первоначального графа. Это ***многоуровневые алгоритмы*** разделения графа [4,7,19,18,10,13]. Следует отметить, что многоуровневые алгоритмы не решают сами по себе проблему разделения графа. Под этим названием скрывается общий подход, основанный на переходе от проблемы разделения большого графа к проблеме разделения меньшего графа. Для меньшего графа затем гораздо дешевле использовать любой метод разделения. При этом уменьшается время вычисления разделения за счет несколько худшего качества разделения [15].

Недавно было предложено несколько многоуровневых алгоритмов [4,7,20,10], которые улучшают процесс уточнения разделителя на стадии обратного увеличения стянутого графа. Эти схемы имеют тенденцию давать хорошие разделения по разумной стоимости. Bui и Jones [4] используют случайное максимальное паросочетание к последовательно огрубляемым графам вплоть до предельного размера в несколько сотен узлов. Далее они делят самый маленький граф и затем рафинируют граф уровень за уровнем, применяя Kernighan-Lin эвристику, чтобы улучшить разделение. Hendrickson и Leland [20] расширяют этот подход, используя веса ребер и узлов, чтобы контролировать стягивание узлов и ребер. Эта последняя работа

показала, что многоуровневые схемы могут обеспечивать лучшее разделение, чем спектральные методы, при более низкой стоимости для самых разнообразных конечно-элементных проблем.

Разрабатываемый автором пакет прикладных программ продолжает направление Hendrickson и Leland. Автор экспериментирует с различными параметрами многоуровневых алгоритмов, исследуя их эффект на качество разделения и переупорядочения. Комбинируя различные эвристики для всех трех стадий – огрубление, разделение самого грубого графа, откат назад с улучшением разделителя – можно за очень эффективное время получить сепаратор приемлемого качества. В частности, представляется новая эвристика для огрубления графа, для которой размер разделителя в грубом графе является маленькой долей от размера заключительного разделителя, полученного после многоуровневого улучшения. А также предлагается новая разновидность Kernighan-Lin-алгоритма для рафинирования разделения в процессе отката огрубления, которая быстрее, чем обычная Kernighan-Lin-обработка, используемая в [20].

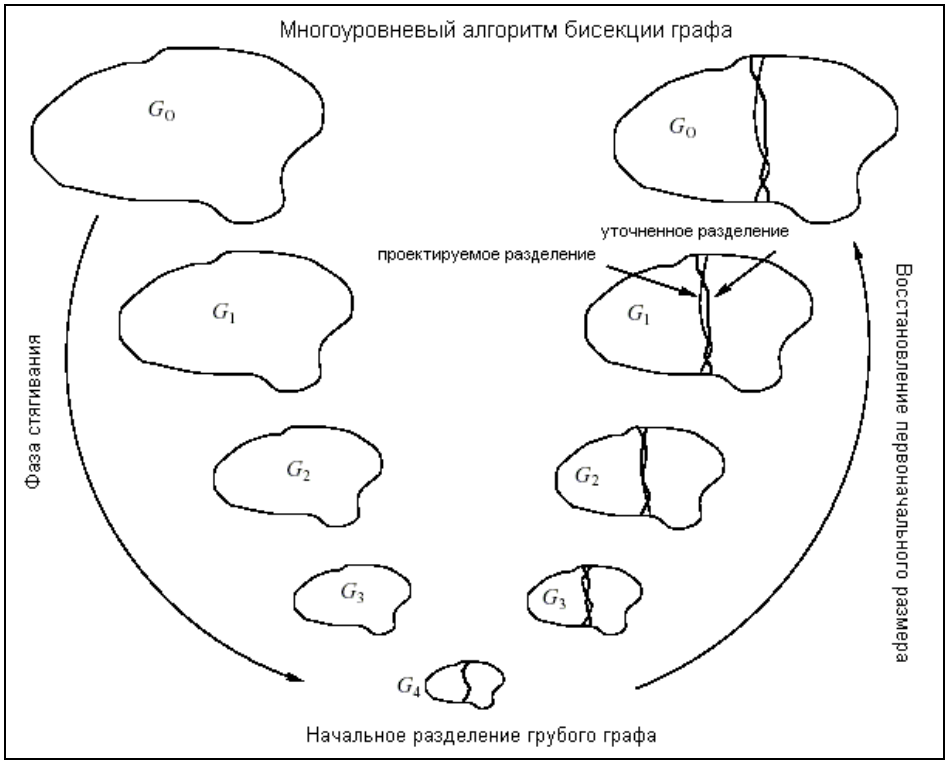

Рис. 1. Различные стадии многоуровневого деления пополам графа. В процессе стадии огрубления размер графа последовательно уменьшается; в процессе стадии начального разделения вычисляется деление пополам меньшего графа; в процессе восстановления деление пополам последовательно проектируется к большему графу и рафинируется. На стадии восстановления пунктирные линии указывают проектируемые разделения, а сплошные линии – разделения, полученные после рафинирующей обработки.

Многоуровневая бисекция графа. Граф G может быть разделен пополам, используя многоуровневый алгоритм. Основная структура многоуровневого алгоритма очень проста. Граф G сначала огрубляется до нескольких сотен узлов, вычисляется деление пополам этого намного меньшего графа, и затем это разделение проектируется (уточняется) назад к первоначальному графу. На каждом шаге восстановления грубого графа разделение улучшается (рафинируется). Так как более точный (больший) граф имеет большее число степеней свободы, такие обработки обычно уменьшают вес сепаратора. Этот процесс проиллюстрирован на рис 1.

Формально, многоуровневый алгоритм деления графа пополам работает следующим образом: рассмотрим взвешенный граф $G_0=(V_0,E_0)$, со взвешенными узлами и ребрами. Многоуровневый алгоритм деления графа пополам состоит из следующих трех стадий.

1. **Огрубление:** Граф G_0 преобразуется в последовательность меньших графов G_1, G_2, ..., G_m так, что $|V_0|>|V_1|>|V_2|>\ldots>|V_m|$.
2. **Разделение:** Бисекция P_m графа $G_m = (V_m,E_m)$ вычисляется с разделением V_m на две части, каждая из которых содержит примерно половину узлов G_0.

3. **Уточнение:** Разделение P_m графа G_m проектируется назад к G_0, проходя промежуточные разделения P_{m-1}, P_{m-2},…, P_1,P_0. Каждое промежуточное проектируемое разделение рафинируется для улучшения сепаратора.

СТАДИЯ ОГРУБЛЕНИЯ

В процессе стадии огрубления строится последовательность меньших графов, каждый с меньшим количеством узлов. Огрубление графа может быть достигнуто различными способами. Некоторые возможности показываются на рис. 2.

В большинстве схем огрубления подмножество узлов G_i объединяется, чтобы сформировать один новый узел для более грубого граф G_{i+1} следующего уровня. Пусть V_i^v – множество узлов G_i, объединенных, чтобы сформировать узел v из G_{i+1}. Мы будем называть узел v мультиузлом. Чтобы деление пополам более грубого графа, было хорошим относительно первоначального графа, вес узла v установлен равным сумме весов узлов в V_i^v. Также, чтобы в более грубом графе сохранить информацию о смежности, ребра v – объединение ребер узлов из V_i^v. В случае, когда более чем один узел V_i^v содержит ребро к тому же самому узлу u, вес ребра (v,u) равен сумме весов этих ребер. Это полезно, когда мы оцениваем качество разделения в более грубом графе. Вес сепаратора разделения в более грубом графе будет равен весу сепаратора того же самого разделения в более точном графе. Обновление весов более грубых граф проиллюстрировано на рис. 2.

Два главных подхода были предложены для получения более грубых графов. Первый подход основан на построении случайного паросочетания и стягивании согласованных узлов в мультиузлы [4,16], в то время как второй подход основан на создании мультиузлов из сильно связных групп узлов [7,19,20,10]. Более поздний подход более удовлетворителен для графов, возникающих в СБИС применениях, так как эти графы имеют сильно связные компоненты. Однако для графов, возникающих в конечно-элементных применениях, большинство узлов имеют сходство в образцах смежности (то есть степень каждого узла довольно близка к средней степени графа). В остальной части этого раздела описываются основные идеи огрубления с использованием паросочетаний.

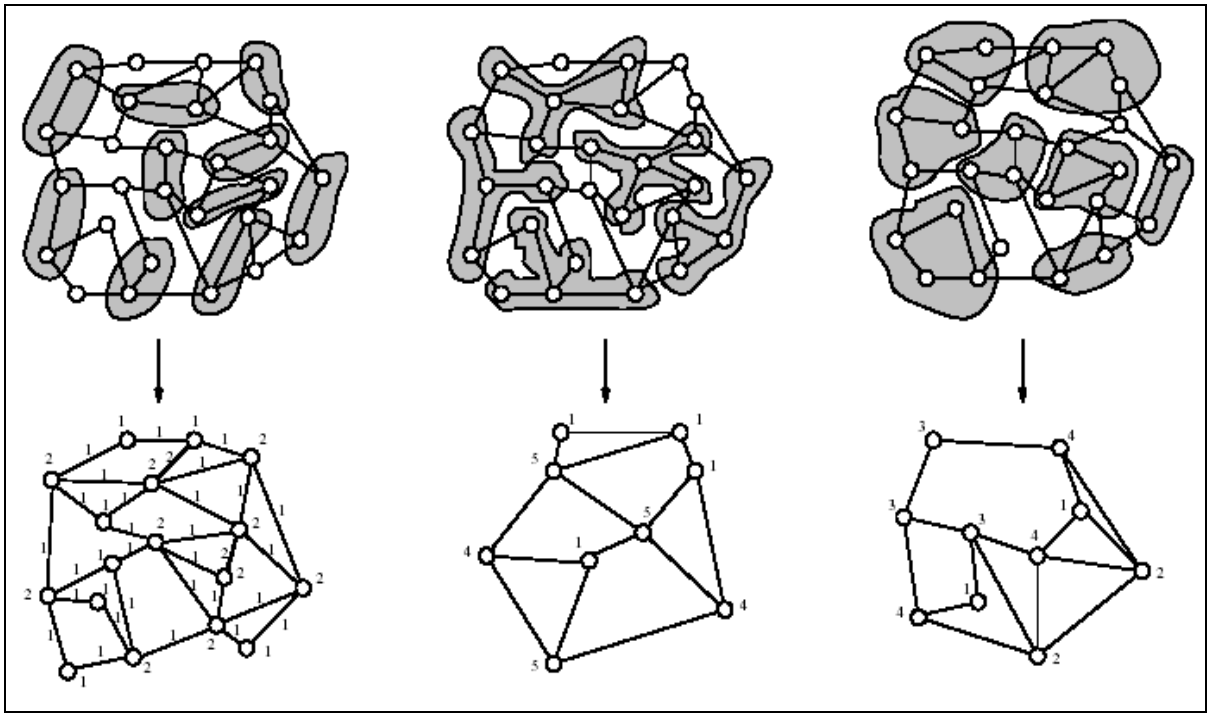

Рис. 2. Различные примеры стягивания графа.

Имея граф $G_i=(V_i,E_i)$, более грубый граф можно получить, стягивая смежные узлы. Таким образом, ребро между двумя узлами удаляется и создается мультиузел, состоящий из этих двух узлов. Эта идея стягивания ребра может быть формально определена в терминах ***паросочетаний***. Паросочетанием в графе является набором ребер, в котором любые два ребра не инцидентны общему узлу. Таким образом, следующий уровень более грубого графа G_{i+1} строится из G_i, путем нахождения паросочетаний в G_i и стягивания в мультиузел узлов, входящих в одну пару. Непарные узлы просто копируются в G_{i+1}. Так как цель стягивания узлов с использованием паросочетаний состоит в том, чтобы уменьшить размер графа G_i, паросочетание должно содержать большое количество ребер. По этой причине используются ***насыщенные паросочетания***, чтобы получить последовательно грубые графы. Паросочетание насыщено, если любое ребро, не вошедшее в паросочетание, имеет, по крайней мере, один конечный узел, который инцидентен

ребру, вошедшему в паросочетание. Обратите внимание, что в зависимости от того, как паросочетание вычислено, число ребер, принадлежащих к насыщенному паросочетанию, может быть различно. Насыщенное паросочетание, которое имеет максимальное число ребер, называется ***максимальным паросочетанием***. Однако, поскольку сложность вычисления максимального паросочетания [11] в общем выше, чем сложность вычисления насыщенного паросочетания, последнее предпочтительнее.

Граф, огрубленный с использованием паросочетаний, сохраняет много свойств первоначального графа. Например, если G_0 планарен, G_i также планарен [14]. Это свойство используется, чтобы показать, что многоуровневый алгоритм производит разделения, которые вероятно хороши для плоских графов [17].

Так как для огрубления графа используется насыщенное паросочетание, число узлов в G_{i+1} не может быть меньше, чем половина от числа узлов в G_i. Таким образом, требуется по крайней мере $O(\log(n/n_0))$ стадий огрубления, чтобы уменьшить граф G_0 до графа с n_0 узлами. Однако, в зависимости от смежности G_i, размер насыщенного паросочетания может быть намного меньше, чем $|V_i|/2$. В этом случае, отношение числа узлов V_i/V_{i+1} может быть намного меньше, чем 2. Если отношение становится меньше, чем заданный предел, то лучше остановить стадию огрубления. Однако, этот тип патологического состояния обычно возникает, после многих уровней огрубления, когда G_i уже довольно мал. Таким образом, прерывание выполнения уменьшения не ухудшает полную производительность алгоритма. Далее описываются три метода, которыми можно строить насыщенные паросочетания для уменьшения графа.

Случайное паросочетание (RM). Насыщенное паросочетание может быть эффективно получено с использованием случайного выбора, как это описано в [4,16]. Узлы посещаются в случайном порядке. Если узел u не был включен в паросочетание, то мы беспорядочно выбираем один из его смежных узлов, который также не включен в паросочетание. Если такой узел v существует, мы включаем ребро (u,v) в паросочетание и маркируем узлы u и v как посещенные. Если не имеется никакого немаркированного смежного узла v, то узел u остается свободным и переходит в следующий граф. Сложность вышеупомянутого алгоритма – O(E).

Паросочетание из тяжелых ребер (HEM). Случайное соответствие – простой и эффективный метод вычислить насыщенное паросочетание – быстро огрубляет граф за минимальное число проходов. Однако наша конечная цель состоит в том, чтобы найти разделение, которое минимизирует вес сепаратора. Рассмотрим граф $G_i=(V_i,E_i)$, паросочетание M_i, который используется для уменьшения G_i, и более грубый граф $G_{i+1}=(V_{i+1},E_{i+1})$, построенный на основе M_i. Если A – набор ребер, определим W(A) как сумму весов ребер в A. Можно показать, что

$$W(E_{i+1}) = W(E_i) - W(M_i). \qquad (1)$$

Таким образом, полный реберный вес более грубого графа уменьшается на вес исключенного паросочетания. Следовательно, выбирая насыщенное паросочетание M_i, чьи ребра имеют большой вес, мы можем уменьшить реберный вес грубого графа в большей степени. Как показал анализ в [17], так как более грубый граф имеет меньший реберный вес, он также имеет и меньший вес сепаратора. Обнаружение насыщенного паросочетания, которое содержит ребра с большим весом – основа эвристики паросочетания из тяжелых ребер. Паросочетание из тяжелых ребер вычисляется с использованием случайного алгоритма, подобного описанному ранее для вычисления случайного паросочетания. Узлы снова посещаются в случайном порядке. Однако теперь, вместо беспорядочного сочетания узла u с одним из его смежных немаркированных узлов, мы сочетаем u с узлом v таким, что вес ребра (u,v) максимален для всех доступных ребер (наиболее тяжелое ребро). Этот алгоритм не гарантирует, что полученное паросочетание имеет максимальный вес (по всем возможным паросочетаниям), но эксперименты показывают, что он работает очень хорошо. Сложность вычисления паросочетания из наиболее тяжелых ребер – O(E), что асимптотически подобно сложности для вычисления случайного паросочетания.

Паросочетание из тяжелых клик (HCM). Клика взвешенного графа G=(V,E) – полностью связанный подграф G. Рассмотрим набор узлов U из V ($U \subset V$). Подграф G, порождаемый U, определяется как $G_U=(U,E_U)$, так что E_U состоит из всех ребер $(v_1,v_2) \in E$ таких, что и v_1, и v_2 принадлежат U. По количеству элементов U и E_U мы можем определить, как близко U к клике. В частности, отношение $2|E_U|/(|U|(|U|-1))$ стремится к 1, если U клика, и мало, если U далеко от того, чтобы быть кликой. Определим это отношение как ***реберную плотность***.

Алгоритм построения паросочетаний из тяжелых клик вычисляет паросочетание, стягивая узлы, которые имеют высокую плотность ребер. Таким образом, эта схема вычисляет паросочетание, чья плотность ребер максимальна. Мотивация создания этой схемы в том, что подграфы из G_0, которые являются кликами или почти клики, не будут наиболее вероятно разделены при бисекции. Поэтому, создавая мультиузлы, что содержат эти подграфы, мы облегчаем алгоритму бисекции поиск хорошего деления пополам. Эта схема является продолжением схем огрубления графа, которые основаны на обнаружении высоко связанных компонент в работах [7,19,20,10].

Как и в предыдущих схемах вычисления паросочетания, мы вычисляем паросочетание из тяжелых клик, используя случайный алгоритм. Для вычисления плотности ребер, пока мы только имели дело со случаем,

когда узлы и ребра первоначального графа $G_0=(V_0,E_0)$ имеют единичный вес. Рассмотрим грубый граф $G_i=(V_i,E_i)$. Для каждого узла $u\in V_i$, определим vw(u) – вес узла. Этот вес равен сумме весов узлов в первоначальном графе, которые стянуты в u. Определим ce(u) – сумма весов стянутых ребер u. Эти ребра были стянуты, чтобы сформировать мультиузел u. Наконец, для каждого ребра $e\in E_i$ определим ew(e) – вес ребра. Это – сумма весов ребер, которые в ходе огрубления были стянуты в e. Плотность ребер между узлами u и v определяется как:

$$\frac{2(ce(u)+ce(v)+ew(u,v))}{(vw(u)+vw(v))\times(vw(u)+vw(v)-1)}\ . \qquad (2)$$

Случайный алгоритм работает следующим образом. Узлы посещаются в случайном порядке. Немаркированный узел u сочетается с его немаркированным смежным узлом v так, что плотность ребер мультиузла, созданного объединением u и v, максимальна среди всех возможных мультиузлов, включающих u и другой немаркированный сосед узла u. Схема HCM очень похожа на HEM. Единственное различие в том, что HEM сочетает узлы, которые только связаны тяжелым ребром независимо от суммарного реберного веса узлов, в то время как HCM сочетает узлы, если они связаны тяжелым ребром и если каждый из этих двух узлов имеет большой реберный вес.

СТАДИЯ РАЗДЕЛЕНИЯ

Вторая стадия многоуровневого алгоритма вычисляет высококачественное деление пополам (сепаратор с маленьким весом) P_m для грубого графа $G_m=(V_m,E_m)$ так, что каждая часть содержит примерно половину узлового веса первоначального графа. Поскольку веса узлов и ребер уменьшенных графов в процессе огрубления вычислялись так, чтобы в целом не потерять существенную информацию о весе узлов и ребер большего графа, G_m содержит достаточную информацию, чтобы эффективно осуществить сбалансированное разделение и выполнить требование приемлемой минимальности веса сепаратора.

Разделение G_m может быть получено, используя различные алгоритмы:

- спектральное деление пополам [47,46,2,24];
- геометрическое деление пополам [17] (если координаты доступны, например координаты для узлов последовательных более грубых графов могут быть построены, беря среднее значение координат объединенного узла);
- комбинаторные методы [11,3,12,17,5,21].

Так как размер более грубого графа G_m мал ($|V_m|<100$), этот шаг не требут большого количества времени. Были апробированы четыре различных алгоритма для разделения грубого графа. Первый алгоритм использует спектральное деление пополам. Другие три алгоритма – комбинаторные по характеру, и пробуют произвести бисекцию с маленьким весом сепаратора, используя различные эвристики. Алгоритмы геометрического деления пополам не использовались, поскольку информация о координатах вершин на практике для большинства графов не доступна.

Спектральное деление пополам (SB). В спектральном алгоритме деления пополам, чтобы разделить граф используется спектральная информация [2,16]. Этот алгоритм вычисляет собственный вектор y, соответствующий второму по величине собственному значению матрицы Лапласа Q=D-A, где $A_{i,j}=ew(v_i,v_j)$, если $(v_i,v_j)\in E_m$, а иначе 0. Этот собственный вектор еще называют Fiedler-вектором. Матрица D диагональная, так что $D_{i,i}=\sum ew(v_i,v_j)$ для $(v_i,v_j)\in E_m$. После получения вектора y, множество узлов V_m делится на две части следующим образом. Устанавливаем r=y[i]. Устанавливаем P[j]=1 для всего узлов, таких что $y_j\le r$, и P[j]=2 для всех других узлов. Так как мы заинтересованы в бисекции на равные части, величина r выбирается как взвешенное среднее величин y_i. Собственный вектор y вычислялся с использованием алгоритма Ланцоша [42]. Этот алгоритм итеративен, а требуемое число итераций зависит от желаемой точности.

Kernighan-Lin алгоритм (KL). Kernighan-Lin алгоритм [21] итеративен по своей природе. Он начинается с начального разделения графа пополам. На каждой итерации он ищет поднабор узлов от каждой части графа, такой, что обмен этими поднаборами ведет к разделению с меньшим сечением. Если такие поднаборы существуют, то обмен выполняется, и это становится разделением для следующей итерации. Алгоритм продолжается, повторяя полный процесс. Если он не может найти два таких поднабора, то алгоритм заканчивается, так как для разделения достигнут местный минимум, и никакое дальнейшее усовершенствование KL алгоритмом не может быть сделано. Каждое повторение из KL алгоритма, описанного в [21], имеет сложность *O(E*log(E))*. Разработано несколько усовершенствований первоначального KL алгоритма. Один такой алгоритм – Fiduccia и Mattheyses [9], который уменьшает сложность до O(E), используя соответствующие структуры данных.

Kernighan-Lin алгоритм находит локальный минимум для разделения, когда он начинает с хорошего начального разделения и когда средняя степень узлов графа велика [4]. Если не известно хорошее начальное разделение, KL алгоритм повторялся с различными случайно отобранными начальными разделениеми, и отбиралось то разделение, которое дает самый маленький вес сепаратора. Требование многократного повторения вычислений, может быть достаточно обременительным, особенно если граф велик. Однако, так как мы разделяем только намного меньший грубый граф, многократное выполнение требует очень небольшого времени. Опыты показали, что KL алгоритм требует только 5–10 различных повторений, чтобы найти хорошее разделение. Наша реализация Kernighan-Lin алгоритма основана на алгоритме, описанном Fiduccia и Mattheyses [9], с некоторыми модификациями, которые значительно уменьшают время выполнения. Алгоритм, описанный Fiduccia и Mattheyses (FM) [9], слегка отличается от первоначального алгоритма Kernighan и Lin (KL) [11]. Различие в том, что на каждом шаге FM алгоритм перемещает единственный узел из одной части в другую, в то время как KL алгоритм выбирает пару узлов, по одному от каждой части, и обменивает их.

Предположим, что P – начальное разделение узлов G=(V,E). Выгода g_v для узла v определяется как сокращение веса сепаратора, если узел v перемещается из одной части разделения в другую. Эта выгода вычисляется как:

$$g_v = \sum_{(v,u)\in E \wedge P[v]\neq P[u]} w(v,u) - \sum_{(v,u)\in E \wedge P[v]=P[u]} w(v,u), \qquad (3)$$

где w(v,u) – вес ребра (v,u). Если выгода g_v положительна, то перемещаем v в другую часть разделения, уменьшая вес сепаратора на g_v. Иначе, если выгода g_v отрицательна, вес сепаратора увеличивается на то же количество. Если узел v перемещен из одной части разделения в другую, то выгода от перемещения узлов, смежных с v, может измениться. Таким образом, после перемещения узла, мы должны модернизировать коэффициенты выгодности его смежных узлов.

Согласно этому определению коэффициента выгоды, KL-алгоритм многократно выбирает из большей части узел v с самой большой выгодой и перемещает его в другую часть. После перемещения v, v маркируется, чтобы он не рассматривался снова на этой итерации, а выгоды узлов, смежных с v, пересчитываются, чтобы отразить изменения в бисекции. Оригинальный KL-алгоритм [9] продолжает перемещать узлы между разделением, пока все узлы не будут перемещены. Однако, в нашей реализации KL-алгоритм заканчивается, когда вес сепаратора не уменьшается после n перемещений узлов. Так как последнее n-е перемещение узла не уменьшило вес сепаратора (а, возможно, фактически увеличило его), оно отменяется. Мы нашли, что значение n=40 работает весьма хорошо на тестовых задачах. Завершение KL-итераций в этом духе значительно уменьшает время выполнения KL-алгоритма.

Эффективное выполнение вышеупомянутого алгоритма зависит от метода, который используется, чтобы вычислить выгоды узлов графа, и от типа структуры данных, используемой для хранения этих коэффициентов выгоды.

Алгоритм возрастающего графа (GGP). Другой простой путь деления графа пополам состоит в том, чтобы начать с одного узла и наращивать область вокруг этого первого узла динамически, пока не будет включена половина узлов (или половина общего веса узлов) [12,17]. Качество алгоритма растущего графа чувствительно к выбору узла, с которого начинается рост графа – различные начальные узлы приводят к сепараторам различного веса. Чтобы частично решить эту проблему, можно беспорядочно выбрать 10 узлов, и вырастить 10 различных областей. Результат с меньшим весом сепаратора отбирается как окончательное разделение. Это разделение далее подается на вход KL алгоритма как начальное приближение и рафинируется. И снова, поскольку G_m очень мал, этот шаг занимает маленький процент от полного времени работы.

Алгоритм возрастающего графа с учетом выгод (GGGP). Алгоритм возрастающего графа, описанный выше, выращивает разделение случайным образом. Однако, как в KL алгоритме, для каждого узла v мы можем определить выгоду в весе сепаратора, полученную от вставки v в возрастающий регион. Таким образом, мы можем упорядочить граничные узлы графа в неубывающем порядке согласно их выгоде. Узел с самым большим уменьшением (или самым маленьким увеличением) в весе сепаратора вставляется первым. Когда узел вставляется в растущее разделение, то выгоды его смежных узлов, находящихся на границе, обновляются, и те, возможно, уходят с границы. Структуры данных, требуемые, для осуществления этой схемы – по существу те, что требуются KL алгоритмом. Единственное различие состоит в том, что вместо предвычисления всех выгод для всех узлов, мы производим вычисления только для узлов, которые стоят на границе. Этот выгодный алгоритм также чувствителен к выбору начального узла, но меньше чем GGP. В нашей реализации мы случайно выбираем 5-6 узлов как отправные точки алгоритма и выбираем разделение с меньшим весом сепаратора. Эксперименты показывают, что GGGP требует несколько меньшего количества времени, чем GGP, для разделения грубого графа (потому что ему достаточно меньшего количества повторов), и начальное разделение, найденное в соответствии с этой схемой, лучше, чем найденное GGP.

СТАДИЯ ВОССТАНОВЛЕНИЯ

В ходе стадии восстановления разделение P_m более грубого графа G_m проектируется назад к первоначальному графу, проходя графы $G_{m-1},G_{m-2},...,G_1$. Так как каждый мультиузел G_{i+1} содержит уникальный поднабор узлов G_i, то получить P_i от P_{i+1} можно просто назначая набору узлов V_i^v, стянутых в $v \in G_{i+1}$, метки разделения, унаследованные от $P_{i+1}[v]$ (то есть, $P_i[u]=P_{i+1}[v]$; $\forall u \in V_i^v$).

Даже притом, что P_{i+1} – местное минимальное разделение G_{i+1}, проектируемое разделение P_i может не быть в местном минимуме относительно G_i. Поскольку G_i более точный граф, он имеет большее количество степеней свободы, которые можно использовать, чтобы улучшить P_i и уменьшить вес сепаратора. Следовательно, все еще возможно улучшить проектируемое разделение G_{i-1} местной эвристикой обработки. По этой причине, после проектирования разделения, используется алгоритм рафинирования разделения. Основная цель алгоритма рафинирования сепаратора состоит в том, чтобы выбрать два поднабора узлов, по одному от каждой части так, что после обмена ними результирующее разделение имеет меньший вес сепаратора. То есть, если A и B – две части деления пополам, алгоритм рафинирования выбирает $A' \subset A$ и $B' \subset B$ такие, что $(A \backslash A') \cup B'$ и $(B \backslash B') \cup A'$ являются новыми частями деления пополам с меньшим весом сепаратора.

Имеется класс алгоритмов, основанных на Kernighan-Lin-алгоритме разделения, которые имеют тенденцию производить очень хорошие результаты. KL-алгоритм начинается с начального разделения, и на каждой итерации он находит поднаборы A' и B' с вышеупомянутыми свойствами. Далее описываются два различных алгоритма рафинирования, которые основаны на подобных идеях, но отличаются по времени обработки.

Kernighan-Lin рафинирование. Основная идея Kernighan-Lin рафинирования состоит в том, чтобы использовать спроектированное разделение G_{i+1} для G_i как начальное разделение для Kernighan-Lin-алгоритма, описанного для стадии разделения. Причина в том, что это проектируемое разделение уже является хорошим разделением; таким образом, KL будет сходиться в пределах нескольких итераций к лучшему разделению. Для наших тестов KL обычно сходится за 3-5 итераций.

Так как мы начинаем с хорошего разделения, только небольшое число обменов узлами уменьшает вес сепаратора, и любые дальнейшие обмены увеличат вес разреза (узлы с отрицательной выгодой). Итерация KL-алгоритма останавливается, как только 50 обменов выполнено, а вес сепаратора не уменьшается. Это уменьшает время выполнения, когда KL применяется как алгоритм рафинирования, поскольку только небольшое число узлов обладает потенциальной выгодой сокращения веса сепаратора. Эксперименты показывают, что для тестовых задач это обычно достигается после того, как только маленький процент от числа узлов был перемещен (меньше чем 5%), что приводит к существенной экономии в полном времени выполнения этого алгоритма обработки.

Так как мы заканчиваем каждый проход KL алгоритма, когда невозможно никакое дальнейшее уменьшение веса сепаратора, сложность KL схемы обработки, описанной в предыдущем разделе, определяется временем, необходимым, чтобы вставить узлы в соответствующие структуры данных. Таким образом, несмотря на то, что мы значительно уменьшили число перемещаемых узлов, асимптотическая оценка полной сложности не изменяется. Кроме того, опыты показывают, что самое большое уменьшение веса сепаратора получается на первом проходе. В KL(1)-алгоритме обработки, мы используем это преимущество, выполняя только один повтор KL-алгоритма. Это обычно уменьшает полное время обработки в 2-4 раза.

Граничный алгоритм Kernighan-Lin рафинирования. И в KL и KL(1)-алгоритмах, мы должны вставить выгоды для всех узлов в структуры данных. Однако, так как мы заканчиваем оба алгоритма, если не можем далее уменьшать вес сепаратора, большинство этих вычислений пропадает впустую. Кроме того, из-за природы алгоритмов рафинирования, большинство узлов перемещается любым KL или KL(1)-алгоритмами по границе разреза. Граница определена, как множество узлов, которые имеют ребра, рассеченные разделением.

В граничном Kernighan-Lin-алгоритме рафинирования мы первоначально вставляем в структуры данных выгоды только для граничных узлов. Как и в KL-алгоритме, после перемещения узла v мы модернизируем выгоды узлов, смежных v, но еще не перемещенных. Если любой из этих смежных узлов станет граничным узлом из-за обмена v, мы вставляем его в структуры данных, если он имеет положительную выгоду. Граничный алгоритм обработки весьма похож на KL-алгоритм, с добавлением того преимущества, что только необходимые узлы попадают в структуры данных и работа по вычислению выгод не потрачена впустую.

Как с KL, мы можем выбирать выполнение единственного прохода граничной модификации KL(1) обработки (BKL(1)) или многократное повторение граничного варианта Kernighan-Lin-обработки (BKL) с выбором пока алгоритм обработки не сходится. В противоположность негранычным алгоритмам обработки, стоимость выполнения многократных проходов граничных алгоритмов мала, поскольку исследуются только граничные узлы.

Для дальнейшего уменьшения времени выполнения граничной обработки при поддержании способностей обработки BKL и скорости BKL(1) можно объединить эти схемы в гибридную схему, которую обозначим как BKL(*,1). Идея в основе BKL(*,1) эвристики – использовать BKL, пока граф маленький, и переключаться на BKL(1), когда граф большой. Мотивация этой схемы в том, что отдельные обмены узлов в более грубых графах ведут к большему уменьшению веса сепаратора, чем в более точных графах. Таким образом, используя BKL в этих более грубых графах, мы достигаем лучшей обработки, а поскольку эти графы очень маленькие по сравнению с размером первоначального графа, BKL алгоритм не требует много времени. Для всех экспериментов, представленных в этой статье, если число узлов в границе грубого графа меньше, чем 2% от числа узлов в первоначальном графе, обработка выполняется BKL, иначе используется BKL(1). Этот выбор условного переключения связывает размер границы разделения, которая пропорциональна стоимости выполнения рафинирования графа, с первоначальным размером графа, чтобы определить, когда недорого исполнить BKL относительно размера графа.

ЗАКЛЮЧЕНИЕ

Эксперименты с многоуровневыми схемами показали, что они работают весьма хорошо для многих различных типов огрубления, начального разделения, рафинирования. В частности, все схемы огрубления хорошо учитывают глобальные свойства графа, а Kernighan-Lin алгоритм или его варианты, используемые для обработки на стадии восстановления, хорошо учитывают локальные свойства графа. Из-за объединения глобального и локального взгляда на граф, обеспеченного схемами огрубления и восстановления, выбор алгоритма для деления грубого графа имеет незначительный эффект на общее качество разделения. В частности, нет никакого преимущества от использования спектральной бисекции для разделения самого грубого графа. Поэтому можно обойтись более дешевыми (в смысле сложности) эвристиками без ущерба для качества сепаратора.

ЛИТЕРАТУРА

1. Stephen T. Barnard and Horst D. Simon. A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. In *Proceedings of the sixth SIAM conference on Parallel Processing for Scientific Computin*g, pages 711–718, 1993.
2. Earl R. Barnes. An algorithm for partitioning the nodes of a graph. *SIAM J. Algebraic Discrete Method*s, N3(4):541–550, December 1984.
3. T. Bui and C. Jones. A heuristic for reducing fill in sparse matrix factorization. In *6th SIAM Conf. Parallel Processing for Scientific Computin*g, pages 445–452, 1993.
4. T. N. Bui, S. Chaudhuri, F. T. Leighton, and M. Sipser. Graph bisection algorithms with good average case behavior. *Combinatoric*a, N7:171–191, 1987.
5. Tony F. Chan, John R. Gilbert, and Shang-Hua Teng. Geometric spectral partitioning. Technical report, Xerox PARC Tech. Report., 1994. Available at ftp://parcftp.xerox.com/pub/gilbert/index.html.
6. Chung-Kuan Cheng and Yen-Chuen A. Wei. An improved two-way partitioning algorithm with stable performance. *IEEE Transactions on Computer Aided Desig*n, N10:1502–1511, December 1991.
7. C. M. Fiduccia and R. M. Mattheyses. A linear time heuristic for improving network partitions. In *In Proceedings 19th IEEE Design Automation Conferenc*e, pages 175–181, 1982.
8. J. Garbers, H. J. Promel, and A. Steger. Finding clusters in VLSI circuits. In *Proceedings of IEEE International Conference on Computer Aided Desig*n, pages 520–523, 1990.
9. George. Nested dissection of a regular finite-element mesh. *SIAM Journal on Numerical Ananlysi*s, N10:345–363, 1973.
10. А. Джордж, Дж. Лиу, Численное решение больших разреженных систем уравнений, Москва, Мир, 1984
11. John R. Gilbert, Gary L. Miller, and Shang-Hua Teng. Geometric mesh partitioning: Implementation and experiments. In *Proceedings of International Parallel Processing Symposiu*m, 1995.
12. T. Goehring and Y. Saad. Heuristic algorithms for automatic graph partitioning. Technical report, Department of Computer Science, University of Minnesota, Minneapolis, 1994.
13. Lars Hagen and Andrew Kahng. Fast spectral methods for ratio cut partitioning and clustering. In *Proceedings of IEEE International Conference on Computer Aided Desig*n, pages 10–13, 1991.
14. Bruce Hendrickson and Robert Leland. An improved spectral graph partitioning algorithm for mapping parallel computations. Technical Report SAND92-1460, Sandia National Laboratories, 1992.

15. Bruce Hendrickson and Robert Leland. A multilevel algorithm for partitioning graphs. Technical Report SAND93-1301, Sandia National Laboratories, 1993.

16. B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journa*l, 1970.

17. R. J. Lipton and R. E. Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematic*s, N36:177–189, 1979.

18. Gary L. Miller, Shang-Hua Teng, W. Thurston, and Stephen A. Vavasis. Automatic mesh partitioning. In A. George, John R. Gilbert, and J. W.-H. Liu, editors, *Sparse Matrix Computations: Graph Theory Issues and Algorithm*s. (An IMA Workshop Volume). Springer-Verlag, New York, NY, 1993.

19. Gary L. Miller, Shang-Hua Teng, and Stephen A. Vavasis. A unified geometric approach to graph separators. In *Proceedings of 31st Annual Symposium on Foundations of Computer Scienc*e, pages 538–547, 1991.

20. C. C. Paige and M. A. Saunders. Solution to sparse indefinite systems of linear equations. *SIAM Journal on Numerical Ananlysi*s, N12:617–629, 1974.

21. Alex Pothen, Horst D. Simon, and Kang-Pu Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal of Matrix Analysis and Application*s, 11(3):430–452, 1990.