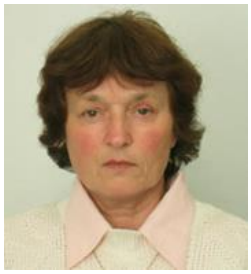
УДК 519.17

ПОИСК ЧАСТО ВСТРЕЧАЮЩИХСЯ ПОДГРАФОВ



Д.И. Черемисинов
Ведущий научный сотрудник ОИПИ НАН Беларуси, кандидат технических наук, доцент

Л.Д. Черемисинова
Главный научный сотрудник ОИПИ НАН Беларуси, доктор технических наук, доцент

Объединений институт проблем информатики НАНБ, Республика Беларусь
E-mail: {cher, cld}@newman.bas-net.by

Аннотация. Сегодня графы широко используются для моделировании данных в различных предметных областях, требующих выяснения и определения правил и схем отношений объектов, а также возможных отклонений. Graph Mining - одно из направлений интеллектуального анализа данных, в котором объемные комплексные данные представлены в виде графов, а анализ ведется для того, чтобы получить новые знания. Поиск часто встречающихся подграфов – это область исследований приложений теории графов, для которой отсутствует литература на русском языке. В докладе рассматривается задача поиска часто встречающихся подграфов в большом графе, и обсуждается применение этой задачи в ряде направлений обработки данных. В докладе приводится обзор подходов к поиску решений этой задачи.

Ключевые слова: графы, кластеризация графа, поиск часто встречающихся подграфов, изоморфизм подграфов, цветовое кодирование графа

Введение. Data mining – это процесс автоматического извлечения полезных знаний, скрытых в больших наборах данных. Эта новая дисциплина становится все более важной, поскольку успехи в сборе данных привели к бурному росту объема имеющихся данных. В последние годы большое внимание уделяется разработке алгоритмов и программных систем для преобразования данных большого объема в полезную и содержательную информацию. Для хранения и обработки данных большого объема с большим числом связей между элементами данных затруднено использование реляционных моделей. Моделирование таких данных в виде графов создает выразительную и универсальную структуру. В такой модели использование связи между данными в форме отношений позволяет увеличить скорость обработки. Поиск часто встречающихся подграфов служит перспективным средством решения задачи data mining, когда данные представлены в виде графов.

В последние годы наблюдается повышенный интерес к разработке алгоритмов интеллектуального анализа данных, работающих на графах. Такие графы возникают естественно в ряде различных областей таких как: обнаружение сетевое атак, семантический web, поведенческое моделирование, перепроектирование СБИС, анализ социальных сетей и классификация химических соединений. Когда данные сложны и имеют много взаимосвязей, становится важной задача поиска знаний о структуре данных. Алгоритмы поиска часто встречающихся подграфов играют важную роль в дальнейшем расширении использования методов интеллектуального анализа для данных большого объема, моделируемых графами. Задача поиска часто встречающихся подграфов состоит в нахождении в заданном графе всех тех подграфов, которые встречаются в нем с частотой большей, чем заданное значение. Идентификация часто встречающихся графов/подграфов в базе данных или в одном большом графе является методом,

который может использоваться для задач классификации, кластеризации графов и построения индексов при поиске подграфов.

Алгоритм наивного поиска часто встречающихся подграфов состоит из двух операций. Первая операция разыскивает все подграфы-кандидаты для данного графа G, а вторая подсчитывает частоту встречаемости каждого подграфа-кандидата. Поиск подграфов кандидатов начинается с графов, состоящих из одной вершины. Новые вершины и ребра добавляются итеративно к уже созданным подграфам для создания новых кандидатов. Для каждого кандидата рассчитывается частота встречаемости. Подграф-кандидат считается часто встречаемым, если его частота встречаемости больше или равна некоторого граничного значения. Этот алгоритм имеет огромную вычислительную сложность. Поиск подграфов-кандидатов требует перебора множества всех подмножеств (булеана) множества вершин графа G (число операций $O(2^N)$, где N число вершин в графе G). Подсчет частоты встречаемости одного подграфа-кандидата S имеет еще большую сложность ($O(2^{N\wedge 2}.N^n)$, где N и n – число вершин в графах G и S).

Разбиение графа на подграфы. Задача разбиения графа на подграфы(graph partition) [1] состоит в представление исходного графа $G=\langle A,V\rangle$ в виде множества подмножеств вершин Sep $A=\{A_1,A_2,...,A_n\}$, $A_i \subseteq V$ по определенным правилам. Обычно по условию задачи требуется, чтобы $\bigcup_{i=1}^{n} A_i = A$, то есть все вершины исходного графа должны быть распределены по подмножествам, причём $A_i \neq \emptyset$. В зависимости от дополнительных условий, накладываемых на блоки разбиения A_i, возникает несколько задач этого типа. Например, в k-блочном разбиении делят множество вершин на *k* или меньше отдельных блоков и так, чтобы число ребер, соединяющих блоки, было минимальным. В задаче равномерного разбиения графа стремятся получить блоки примерно одинакового размера и минимизировать связи между блоками.

Эффективная реализация параллельных программ обычно требует решения проблемы разбиения графа, в котором вершины представляют вычислительные задачи, а ребра представляют обмен данными. K-блочное разбиении графа вычислений может использоваться для назначения задач в компьютере с k процессорами. Поскольку каждый блок разбиения содержит одинаковое количество вычислительных задач, то решение задачи обеспечивает балансировку загрузки этих k процессоров, и поскольку при решении задачи минимизирует мощность множества разрезающих ребер, то одновременно минимизируются межпроцессорный обмен.

Еще одно важное применение рекурсивного деления пополам заключается в том, чтобы найти порядок заполнения при факторизации разреженных матриц. Этот тип алгоритмов обычно называют алгоритмами упорядочения вершин путем вложенных диссекций. Вложенная диссекция рекурсивно разбивает граф на почти равные половины, пока не будет получено требуемое количество разделов. Достигаемое при этом упорядочения вершин обеспечивает ускорение умножения разреженной матрицы на вектор.

В последнее время проблема разбиения графов нашла новые применения в области кластеризации для обнаружения сообществ в социальных и биологических сетях. При поиске сообществ в социальных сетях пытается извлечь структуру из графа социальной сети, разделив его вершины на непересекающиеся подграфы (сообщества), такие, что соединения расположены внутри подграфов, а соединений между подграфами мало.

Задача разбиения на подграфы может быть расширена на гиперграфы, где ребро может соединять более двух вершин [2]. Гиперребро не разрезается, если все вершины находятся в одном блоке и разрезается ровно один раз в противном случае, независимо от того, сколько вершин находится с каждой стороны. Эта постановка задачи возникает в задаче поиска подсхем в схеме из транзисторов, возникающей в процессе проектирования СБИС.

Задача K-блочного разбиения графа может решаться рекурсивным делением пополам [1]. То есть, сначала получаем 2-блочное разбиение V, а затем подразделяем каждую часть путем 2-блочных разбиений. После log k операций граф G разбивается на k частей. Таким образом, задача k-блочного разбиения графа сводится к последовательности 2-блочных разбие-

ний. Хотя эта схема не обязательно приводит к оптимальному разбиению, она широко используется из-за ее простоты.

Поскольку для задач на графах трудно найти хорошие приближенные решения, на практике для решения задачи k -блочного разбиения графа используются эвристические многоуровневые алгоритмы. Эти алгоритмы состоят из трех фаз. На фазе огрубления граф сжимается, чтобы получить иерархию меньших графов. В фазе разбиения применяется алгоритм 2-блочного разбиения к наименьшему графу, построенному на первой фазе. На фазе восстановления это разбиение проецируется на граф меньшего уровня иерархии, вызывая разбиение графа этого уровня, на каждом уровне используется локальный метод поиска для улучшения разбиения, индуцированного более низким уровнем.

Формально алгоритм многоуровневой бисекции графа работает следующим образом: дан граф $G_0 = (V_0, E_0)$. Алгоритм многоуровневой бисекции графа состоит из следующих трех фаз.

Фаза огрубления: граф G0 преобразуется в последовательность меньших графов G_1, G_2,. , , , G_m таких, что $| V_0 | > | V_1 | > | V_2 | > \cdot \cdot \cdot > | V_m |$.

Фаза разбиения: вычисляется 2-блочное разбиение P_m графа $G_m = (V_m, E_m)$, которое разбивает V_m на две части, каждая из которых содержит половину вершин G_0.

Фаза восстановления: разбиение P_m графа G_m проецируется обратно в G_0, проходя промежуточные разбиения P_{m-1}, P_{m-2},. , , , P_1, P_0.

Лучшие программы многоуровневой бисекции позволяют получить на персональном компьютере k -блочные разбиения графов с сотнями тысяч вершин за приемлемое время, когда k не превышает сотни. Следует отметить, что решения задачи k-блочного разбиения графа недостаточно для поиска часто встречающихся подграфов. Эта задача может служить разумным упрощением операции поиска подграфов-кандидатов.

Обзор алгоритмов поиска часто встречающихся подграфов. Целью этих алгоритмов является поиск полного множества часто встречающихся подграфов с использованием некоторой меры частоты встречаемости. Множество часто встречающихся подграфов составляют подграфы, у которых мера частоты встречаемости (support) больше заданной границы. Для решения задачи поиска полного множества часто встречающихся подграфов используются эвристические алгоритмы, эффективная работа которых зависит от общих характеристик исходного графа, например, свойства, которые позволяет отличить графы реального мира от случайных графов. Обычно предполагается, что в исследуемом графе большинство вершин имеют небольшую степень, и мало вершин имеют большое количество соединений. Это свойство выражается формально степенным законом (power law distribution) распределения числа вершин в зависимости от степени.

Задача поиска часто встречающихся подграфов стала популярной областью исследований в последнее десятилетие, и к настоящему времени библиография этой задачи насчитывает сотни публикаций. Литература на русском языке по этой проблеме практически отсутствует. За это время было предложено много алгоритмов поиска часто встречающихся подграфов. Алгоритмы, предложенные последними, превосходят ранние разработки с точки зрения требований к памяти и на нескольких порядков по быстродействию. Но ни один из них не полностью решает проблему огромной сложности вычислении. Способы снижения сложности за счет использования эвристик, не гарантирующих нахождение полного множества часто встречающихся подграфов, все еще решают задачу за экспоненциальное время.

Алгоритмы поиска часто встречающихся подграфов строятся по аналогии с подходами, используемыми в хорошо известной в Data minin задаче поиска часто встречающегося множества атрибутов (frequent item set mining). В последней задаче разыскиваются множества элементов, встречающиеся в достаточно большом количестве транзакций данной базы данных. Для этой задачи предложены алгоритмы, трудоемкость которых меньше полиномиальной,

хотя множество всех решений – это булеан. Трудоемкость поиска часто встречающихся графов имеет более высокий порядок из-за необходимости решать задачу изоморфизма подграфа. Алгоритм перебора подграфов-кандидатов должен избегать генерации кандидатов, которые не часто встречаются или изоморфных друг другу. Тест изоморфизма подграфа должен проводиться как можно реже.

Если вершины и ребра графа имеют уникальные метки, то граф можно смоделировать множеством меток ребер, а затем использовать существующие алгоритмы поиска часто встречающегося множества атрибутов, чтобы найти все часто встречающиеся подграфы. Так как отображение вершин и ребер на метки не уникально, то это сведение невозможно, однако переход к маркированным (раскрашенным) графам позволяет сделать тест изоморфизма подграфа более эффективным. Кроме того можно использовать подход к генерации frequent item set, Apriori принцип, который для задачи поиска часто встречающихся подграфов формулируется следующим образом [3]. Все графы, содержащие не часто встречающийся подграф, тоже не относятся к часто встречающимся.

Маркированный (раскрашенный) граф $G = (V, E, L, l)$, представляется множеством вершин V, множеством дуг $E \subseteq V \times V$ и функцией $l : V \cup E \rightarrow L$ во множество меток L.

Граф $S = (V_S, E_S, L_S, l)$ – это подграф графа $G = (V, E, L, l)$ если $V_S \subseteq V$, $E_S \subseteq E$ и $L_S(u) = L(u) \forall u \in V_S \cup E_S$.

Изоморфизм подграфов между графами g_1 и G является изоморфизмом между g_1 и подграфом S графа G. Граф g_1 называется шаблоном, а S вхождением шаблона $g1$ в G.

Для подграфа G_s и графа G два вхождения G_s в G называются одинаковыми, если они используют в G один и тот же набор ребер. Два вхождения G_s в G называются непересекающимися вершинами, если они не имеют общих вершин в G. Два вхождения G_s в G называются непересекающимися ребрами, если они не имеют общих ребер в G.

В подходе на основе принципа Apriori [4] перед генерацией подграфов-кандидатов размера k+1 необходимо иметь все кандидаты размером k; где размер графа определяется числом его вершин. Чтобы создать следующий кандидат, два подграфа размера k объединяются вместе, формируя граф размера k+1. Чтобы выбрать два кандидата для соединения, нужно чтобы они имели общий подграф (тест изоморфизма подграфов). Сгенерированный граф может быть изоморфным кандидату, сгенерированному ранее (тест изоморфизма графов), в этом случае он отбрасывается. Сгенерированный граф может быть не часто встречающимся (массовый тест изоморфизма подграфов [4]), в этом случае он тоже отбрасывается. При подсчете вхождений кандидата нужно учитывать, что вхождения могут перекрываться. Если это произойдет, необходимо указать, какие вхождения должны быть приняты во внимание, чтобы Apriori принцип выполнялся.

В методе grow-and-store [5], позволяющем уменьшить число проверок изоморфизма, и выполняются следующие шаги.

1. Находятся вершины, у которых число вхождений больше заданной границы, и сохраняются все найденные вхождения в списке вхождений.

2. Каждое вхождение из списка расширяется на одну вершину, образуя подграф-кандидат, оценивается частота его вхождения и, если он часто встречающийся, то сохраняются все его вхождения. Это действие выполняется до тех пор, пока находятся новые часто встречающийся подграфы.

Оценка частоты подграфа здесь не требует теста изоморфизма. Основным узким местом этого подхода является создание списка для хранения всех вхождений каждого часто встречающегося подграфа. Этот список неприемлемо велик для вычисления и хранения.

Генерирование кандидатов из часто встречающихся подграфов меньшего размера позволяет радикально уменьшить пространство поиска часто встречающихся подграфов заданного графа. Однако проблемой остается тест на включение кандидата в множество часто встреча-

ющихся подграфов. Тривиальным решением этой проблемы является выполнение теста кандидата на изоморфизм с каждым графом множества часто встречающихся подграфов. Более экономным способом эта проверка выполняется, если каждый граф множества часто встречающихся подграфов канонизирован [5] и хранится в таблице по хеш-адресу, вычисленному по его канонической маркировке. Когда в этой таблице по хеш-адресу, вычисленному по канонической маркировке подграфа-кандидата, хранится подграф, то он изоморфен тестируемому подграфу. Набольший выигрыш получается, если кандидаты генерируются в канонической маркировке. Канонические маркировки графов, включенных в решение, лексикографически упорядочены, и построение подграфа кандидата заключается в нахождении маркировки, которая больше всех уже построенных маркировок.

Другой класс алгоритмов подсчета частоты подграфов основан на мощной методике цветового кодирования (color coding) [7]. Идея цветового кодирования заключается в случайном раскраске графа с использованием k цветов (где k обозначает размер шаблона), и подсчете количества «красочных» вхождений шаблона, в которых все вершины имеют разные цвета. Затем это количество масштабируется по вероятности того, что вхождение шаблона является красочным. Среднее из масштабированных оценок частоты вхождений шаблона после нескольких итераций с различными случайными раскрасками исходного графа дает частоту вхождений шаблона. Метод цветового кодирования позволяет подсчитать частоту вхождений некоторого заданного шаблона в большом графе приблизительно. С вычислительной точки зрения, возможно, главным недостатком алгоритмов цветового кодирования является их потребность в памяти, препятствующая их использованию с ростом размера шаблона.

Теоретически, самый быстрый алгоритм подсчета вхождений k-вершинного подграфа в n-вершинный граф данных выполняется за время $n^{\omega k/3}$, где $O(n^{\omega})$ – временная сложность матричного умножения (в настоящее время $\omega \approx 2.38$). Это лучше временной сложности тривиальный алгоритм $O(2^{n\wedge 2}.n^{k})$, но все равно чрезмерно даже для шаблонов с k большим нескольких десятков.

Несмотря на то, что существует обширная литература, посвященная исследованию задачи поиска часто встречающихся подграфов, эффективность алгоритмов недостаточна для обработки графов тех размеров, которые встречаются в приложениях. Прежде чем поиск часто встречающихся подграфов станет практичным подходом в приложениях для интеллектуального анализа данных, нужно разработать методы поиска, масштабируемые в соответствии с размером графов, моделирующих большие данные [5]. При интеллектуальном анализе данных речь идет не о том, можно ли получить полное множество часто встречающихся подграфов при определенных ограничениях эффективно, но и о том, как получить ограниченный набор часто встречающихся подграфов, которые наиболее полезны в приложениях.

Заключение. Проблема поиска часто встречающихся подграфов (frequent subgraph mining – FSM) заключается в том, чтобы найти все подграфы в заданного графа, которые встречаются с частотой выше некоторого минимального порога. Проблема FSM стала актуальной в последнее десятилетие и имеет две постановки: поиск часто встречающихся вхождений шаблонов в транзакционных базах данных (база данных, содержащая много небольших графов) и поиск подграфов в одном большом графе. В последние годы этой проблеме в первой постановке уделялось большое внимание. Тем не менее, разработка алгоритмов, способных находить шаблоны в единственном графе, получила гораздо меньшее внимание, несмотря на то, что эта проблема более универсальна и применима к более широкому диапазону наборов данных и областей приложений. Причиной этого является то, что задача поиска часто встречающихся подграфов в заданного графа сложнее на обоих этапах поиска шаблона и требует большего объема вычислений.

Список литературы

[1]. Karypis G., Kumar V. Multilevel Graph Partitioning Schemes // Proc. 24th Intern. Conf. Par. Proc., III – CRC Press, 1995. – pp. 113-122.

[2]. Schlag S., Henne V., Heuer T., Meyerhenke H., Sanders P., Schulz C.. k-way Hypergraph Partitioning via n-Level Recursive Bisection // 18th Workshop on Algorithm Engineering and Experiments (ALENEX), 2016. – pp. 53–67.

[3]. Inokuchi, A.; Washio, T.; Motoda, H. An apriori-based algorithm for mining frequent substructures from graph data // European Symposium Principle of Data Mining and Knowledge Discovery (PKDD'00), 2000.– pp. 13–23.

[4]. Sun Z., Wang H., Wang H., Shao B., Li J. Efficient subgraph matching on billion node graphs //Proc. VLDB Endow., vol. 5, no. 9, , 2012.– pp. 788–799.

[5]. Kuramochi, M.; Karypis, G. Frequent subgraph discovery // International Conference on Data Mining (ICDM'01), San Jose, CA, Nov. 2001. – pp. 313–320.

[6]. Slota G. M., Madduri K., Fast approximate subgraph counting and enumeration // 42nd International Conference on Parallel Processing (ICPP), 2013. – pp. 210-219.

[7]. Alon N., Yuster R., Zwick U. Color-coding // J. ACM, 42(4), 1995. – pp. 844–856.

FREQUENT SUBGRAPH MINING

D.I. CHEREMISINOV, PhD
Leading Researcher of the Institute of Information Technologies of the National Academy of Sciences of Belarus, Associate Professor

L.D. CHEREMISINOVA,
Doctor of Technical Sciences
Chief Scientific Researcher of the Institute of Contemporary Social Sciences of the National Academy of Sciences of Belarus, Associate Professor

United Institute of Informatics Problems, National Academy of Sciences of Belarus, Republic of Belarus
E-mail: {cher, cld}@newman.bas-net.by

Abstract. Graphs are common data structures used to represent/model real-world systems. Graph Mining is one of the arms of Data mining in which voluminous complex data are represented in the form of graphs and mining is done to infer knowledge from them. Frequent sub graph mining is a sub section of graph mining domain, which is extensively used for graph classification, building indices and graph clustering purposes. The frequent sub graph mining from the point of view of analyzing a large single graph is addressed and viewed in different directions based upon the domain expectations. In this paper, a survey is done on the approaches in targeting frequent sub graphs and various scalable techniques to find them.

Key words: Graph, Graph partitioning, Frequent Subgraph Mining, Subgraph Isomorphism, Color coding method of subgraph counting.