Некоторые задачи на графовых базах данных

Р. И. Гуральник <guralnikr@gmail.com>
Санкт-Петербургский государственный университет,
199034, Санкт-Петербург, Университетская набережная, д. 7/9

Аннотация. Одним из наиболее популярных и актуальных подвидов нереляционных баз данных являются графовые базы данных. В данной работе рассмотрены задачи на таких базах данных, которые наиболее часто встречаются в современной литературе. Изучены задачи максимизации влияния, motif mining (MM), задача оценки схожести узлов графа, сопоставление образца в графе. Рассмотрены первичные алгоритмы каждого направления и некоторые промежуточные работы. Проанализированы алгоритмы, соответствующие текущему положению дел.

Ключевые слова: графовые базы данных; сетевые мотивы; сопоставление с образцом; максимизация влияния; simrank.

DOI: 10.15514/ISPRAS-2016-28(4)-12

Для цитирования: Гуральник Р.И. Некоторые задачи на графовых базах данных. Труды ИСП РАН, том 28, вып. 4, 2016, стр. 193-216. DOI: 10.15514/ISPRAS-2016-28(4)-12

1. Введение

Большие данные (англ. big data) – совокупность подходов, инструментов и методов обработки структурированных и неструктурированных данных огромных объемов и значительного многообразия для получения воспринимаемых человеком результатов [1]. Другими словами, большие данные – это проблема хранения и обработки гигантских объемов данных. С другой стороны, обработка больших объемов информации – это только часть «айсберга». Как правило, когда говорят о «больших данных», то используют наиболее популярное определение трех «V», что означает Volume – объем данных, Velocity – необходимость обрабатывать информацию с большой скоростью и Variety – многообразие и часто недостаточную структурированность данных. [2]

Графовые базы данных стали одним из наиболее актуальных представлений больших данных. Их популярность обусловлена их удобством применения в задачах, в которых данные имеют большое количество связей, например в пищевых цепочках, белок-белковых взаимодействиях и социальных сетях.

Кроме того, ребра графа являются хранимыми данными, а значит обход графа не требует дополнительных вычислений. Такая система оказалась естественной и востребованной в современном мире сети Интернет и социальных сетей [3].

Самым крупным разделом задач на графовых базах данных является глубинный анализ данных (data mining). Сюда входят задачи по обучению ассоциативным правилам, классификации и категоризации данных, кластерный анализ, регрессионный анализ и др. Среди менее крупных разделов задач можно отметить пространственный и статистический анализ данных, визуализацию аналитических данных [1, 4]. Описать все задачи в данном небольшом обзоре не представляется возможным.

Данная статья представляет собой обзор наиболее популярных задач на графовых базах данных из раздела data mining: оценка схожести объектов, распространение влияния узлов внутри графа, поиск часто встречающихся подграфов и сопоставление с образцом. Их популярность отражается большим набором публикаций по этим задачам на крупных конференциях последних годов. В работе будут представлены традиционные постановки задач и базовые алгоритмы их решения, предложенные 10-15 лет назад, а также рассмотрены некоторые работы, соответствующие текущему положению дел по рассматриваемому направлению.

Структурно данный обзор состоит из 4 разделов. Первый раздел посвящен задаче оценки схожести объектов и алгоритму SimRank. Второй раздел описывает задачу максимизации влияния узлов графа. В третьем разделе представлен анализ задачи сопоставления графа с образцом подграфа. В четвертом раздел содержит алгоритмы поиска сетевых мотивов (network motifs).

2. Simrank

В данном разделе мы рассмотрим такую задачу как измерение "похожести" объектов. Во множестве основных задач на графах, например, в прогнозировании связей (*link prediction*), кластеризации, обнаружении спама, поиске часто встречающегося подграфа, рекомендательных системах и др. требуется тщательный анализ схожести сущностей, а значит возникает необходимость платформы для эффективного вычисления похожести объектов.

В 2002 г. Глен Йех и Дженифер Видом представили миру модель под названием SimRank [5] - меру оценки подобия объектов, основанную на анализе взаимоотношения этих объектов друг с другом. Основной идеей SimRank считается фраза "два объекта похожи если на них ссылаются похожие объекты". Поскольку формулировка похожести задается через саму себя, то базой этой рекурсии служит утверждение "каждый объект максимально похож на самого себя".

Формально модель описывается следующим образом. Обозначим за $I(v)$ и $O(v)$ множества входящих и выходящих соседей узла v, а за $s(a,b)$ меру схожести двух узлов. Следую рекурсивному определению $s(a,b) = 1$, если $a = b$. Иначе

$$s(a,b) = \frac{C}{|I(a)||I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} s_i(I_i(a), I_j(b)) \quad (1)$$

где C - константа со значением из $(0, 1)$. Это константа называется *коэффициентом затухания* и является необходимой для реалистичной оценки. В своих вычислениях авторы используют C = 0,8. Опять же, из-за рекурсивности меры похожести на практике используется итеративная модель вычисления, определяя для первой итерации единицей меру похожести одинаковых объектов и нулем меру похожести разный объектов. На каждой итерации похожесть как бы "распространяется" по графу следуя формуле

$$R_{k+1}(a,b) = \frac{C}{|I(a)||I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} R_k(I_i(a), I_j(b)) \quad (2)$$

Йех и Видом[5] доказывают сходимость процесса, при котором $R_k(a,b) \to s(a,b)$, при $k \to \infty$, а так же утверждают, что при k = 5 итерациям, полученные значения можно считать достаточно точными для оценки значения похожести.

В их работе так же предлагается иной способ вычисления оценки схожести с помощью *пар случайных блужданий* (*random surfer-pairs model*). В этом случае, для графа G строится соответствующий ему граф G^2, в котором узлы представляют собой пары, составленные из узлов G, а дуга из (a,b) в (c,d) присутствует в G^2 только если в G есть дуга из a в c и из b в d. Предположим, что 2 пользователя случайно блуждают по графу, начиная из узлов a и b соответственно. Тогда утверждается, что оценка $s(a,b)$ схожести узлов a и b полученная SimRank совпадает с ожидаемым числом шагов, необходимым пользователям для того, чтобы встретиться в каком-либо узле. В этом случае формула оценки выглядит следующим образом:

$$s'(a,b) = \sum_{t:(a,b)\rightsquigarrow(x,x)} P[t]c^{l(t)} \quad (3)$$

Суммирование ведется по всем t - путь в G^2 из какого либо узла в одноточечный узел (в этом случае пользователи находятся в одном и том же узле начального графа G). $l(t)$ - длина пути, а константа c играет ту же роль, что и C в (1). Вероятность выбора пути t P[t] вычисляется формулой

$$P[t] = \prod_{i=1}^{k-1} \frac{1}{|O(w_i)|} \quad (4)$$

где w_i - промежуточный узел пути t.

Описывая SimRank, полезно упомянуть родственный ему алгоритм PageRank, который является первым и наиболее популярным алгоритмом для упорядочивания результатов поисковой системы Google. PageRank подсчитывает количество и качество ссылок на страницу и грубо оценивает,

насколько эта страница важна. В основе PageRank лежит предположение, что чем важнее вебсайт, тем больше на него ссылаются другие сайты.

На рис. 1 изображен пример применения PageRank для простой сети, результаты выражены в процентном соотношении. Страница C имеет показатель PageRank выше, чем у страницы E, несмотря на то, что к странице E ведут больше ссылок. На страницу C ссылается одна важная страница и, следовательно, эта ссылка ценится выше. Если блуждания, начавшиеся со случайной страницы имеют вероятность 85% выбора исходящей ссылки со страницы, на которой они находятся, и вероятность 15% перейти на любую случайную страницу из всей сети, то в 8.1% случаев они окажутся на странице E (без фактора случайного «прыжка» все блуждания заканчивались бы на страницах A, B или C).

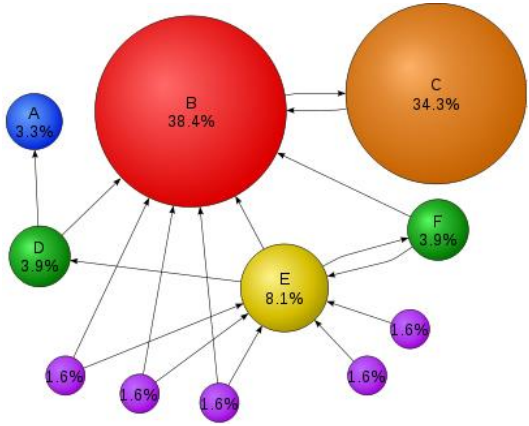


Рис. 1. Пример представления PageRank.

Fig. 1. PageRank representation.

Предложенный метод случайных блужданий для SimRank имеет вычислительную сложность в общем случае равную $O(n^4)$. Для оптимизации Йех и Видом предлагают оценивать схожесть пар, основываясь только на влиянии ближайших соседей (например на расстоянии 2-3 дуг), тогда сложность всего алгоритма составит $O(kn^2d^2)$, где d - средняя степень узла (не должна зависеть от n), а k - количество итераций. По утверждениям авторов, после всех оптимизаций, модель способна проводить вычисления для графов в пределах $n = 278{,}626$ объектов, что является практически неприемлемым в наше время ввиду того факта, что современные графовые базы данных насчитывают миллионы узлов и миллиарды связей. Так же стоит отметить такой ограничивающий фактор, как объем машинной памяти, требуемый для хранения результатов вычислений. В среднем потребуется порядка n^2 единиц памяти, что может составить терабайты для обработки баз данных с миллионом или более объектов.

За последние 15 лет для уменьшения вычислительных стоимостей был предложен целый ряд алгоритмов [6-13]. Для относительного сравнения этих работ удобно использовать табл. 1, представленную в работе Касумото *et al* [14], но для начала разберем основные типы задач для SimRank. 1) Одиночная пара: сосчитать значение похожести двух данных узлов *u* и *v*. 2) Одиночный источник: для данного узла *u* сосчитать ее значение похожести со всеми остальными узлами. 3) Все пары: сосчитать значение похожести для каждых двух узлов.

Табл. 1. Сложности алгоритмов для задач Simrank.

Table 1. Algorithmic complexities for Simrank problem types.

Алгоритм	Тип задачи	Временная сложность	Память	Метод
Kasumoto *et al* [14] (state-of-the-art)	Top-*k* поиск	$<<O(n)$	$O(m)$	Линейно-рекурсивная формулировка и Монте-Карло
Kasumoto *et al* [14] (state-of-the-art)	Top-*k* все пары	$<<O(n^2)$	$O(m)$	Линейно-рекурсивная формулировка и Монте-Карло
Li *et al* [9]	Одиноч. пара	$O(Td^2n^2)$	$O(n^2)$	Пары случайных проходов (итеративный)
Fogaras and Racz [6]	Одиноч. пара	$O(TR)$	$O(m+nR)$	Пары случайных проходов (Монте-Карло)
Jeh and Widom [5]	Все пары	$O(Td^2n^2)$	$O(n^2)$	Наивный
Lizorkin *et al* [11]	Все пары	$O(Tmin\{nm,n^3/\log n\})$	$O(n^2)$	Частичные суммы
Yu *et al* [13]	Все пары	$O(Tmin\{nm,n^w\})$	$O(n^2)$	Быстрое перемножение матриц
Li *et al* [8]	Все пары	$O(r^4n^2)$	$O(n^2)$	Разложение на сингулярные числа
Fujiwara *et al* [15]	Все пары	$O(r^4n)$	$O(r^2n^2)$	Разложение на сингулярные числа
Yu *et al* [10]	Все пары	$O(n^3+Tn^2)$	$O(n^2)$	Разложение на собственные числа

Касумото *et al* [14] используют матричную формулировку SimRank: пусть P - матрица перехода для графа G, т. е.,

$$P_{i,j} = \begin{cases} \dfrac{1}{|I(j)|}, & (i,j) \in E, \\ 0, & (i,j) \notin E \end{cases} \qquad (5)$$

Пусть S - матрица, содержащие значения похожести узлов, то есть $S_{i,j} = s(i,j)$. Тогда (1) в матричном виде перепишется как

$$S = \max\{c(P^{\mathsf{T}}SP), I\} \qquad (6)$$

Для сокращения вычислительных ресурсов Касумото *et al* предлагают алгоритм вычисления top-*k* узлов, основывающийся, на четырех пунктах.

1. Линейно-рекурсивная формулировка. Касумото *et al* [14] вводят диагональную матрицу *D:*

$$S = c(P^{\mathsf{T}} SP) + D \qquad (7)$$

и доказывают, что она существует и корректно определена. Эта матрица позволяет преобразовать нелинейные рекурсивные вычисления в правой части в сходящийся ряд и использовать его частичную сумму из T слагаемых для точной приближенной оценки значения схожести одиночной пары.

2. Метод Монте-Карло. Поскольку для задачи одиночного источника (значения схожести одного узла со всеми остальными) временная сложность линейной рекурсии составит $O(Tmn)$, что не является приемлемым, авторы используют метод Монте-Карло для выборки из R независимых пар случайных проходов. Здесь предлагается алгоритм, чья сложность сводится к $O(TR)$.

3. Зависимость от расстояния. Ключевое наблюдение SimRank: значение схожести между u и v затухает очень быстро с увеличением расстояния между этими узлами. Касумото *et al* [14] провели опыты с некоторыми реальными базами данных и выяснили, что top-1000 самых схожих узлов для случайных 100 узлов лежат не дальше чем на расстоянии в 7 дуг от исходного узла.

4. Верхняя оценка значения схожести. Этот и предыдущий пункты помогают сильно сократить временную сложность предлагаемого алгоритма с помощью верхней оценки значения схожести узлов, которая зависит только от расстояния между узлами. Здесь авторы находят 2 оценки в зависимости от того, высокая или низкая степень у рассматриваемого узла.

Собирая все пункты воедино, Касумото *et al* [14] предложили эффективный двухфазный алгоритм для вычисления top-k самых схожих узлов с заданным узлом. Первая фаза - фаза обработки - находит "кандидатов", которые могут иметь большое значение схожести, а вторая фаза уже проверяет этих "кандидатов", находит их точные значения схожести с заданным узлом и выдает ответ на top-k запрос.

Рассматривая задачу SimRank, важно уделить внимание случаю неопределенных графов. Неопределенный граф G можно рассматривать как распределение вероятностей для всех реальных исходов графа - детерминированных (определенных) графов G. Главным отличием неопределенных графов при вычислении коэффициентов похожести является несправедливость формулы для k-ой степени матрицы перехода. То есть $P^{(k)} \neq (P^{(1)})^k$.

Такое наблюдение сделал и доказал в своей статье Жу *et al* [16]. Он предложил новые обобщающие формулы для вероятностей случайных проходов по неопределенному графу и значений похожести узлов. Результатом его работы стали 3 предложенных алгоритма: 1) первый алгоритм с высокой точностью вычисляет необходимые n матриц перехода $P^{(1)}, P^{(2)}, \ldots, P^{(n)}$; 2) второй алгоритм использует формирование выборки для подсчета $P^{(1)}, P^{(2)}, \ldots, P^{(n)}$ с высокой эффективностью; 3)третий алгоритм объединяет приемы двух предыдущих и сравним по эффективности со вторым алгоритмом, но обладает на порядок величины меньшей погрешностью оценки.

3. Influence maximization.

Модели процессов, при которых информация распространяется по социальным сетям, изучаются и применяются уже довольно давно. В 2001 году Домингос и Ричардсон [17] поставили одну из фундаментальных алгоритмических задач для социальных сетей: если мы сможем убедить принять какой-либо продукт или новшество определенную группу людей с целью запустить череду принятия продукта или новшества другими людьми, то каким образом нужно выбрать эту определенную группу? Первыми, кто посмотрели на максимизацию влияния как на задачу оптимизации и предложили ее решение стали Кемпе *et al* [18]. Их работа смотивировала целый ряд исследований алгоритмов по сокращению вычислительных затрат [19-22], а сама задача максимизации влияния развивалась во многих направлениях: тематическая MB [23], пространственно-ориентированная MB [24,25], MB при наличии конкурентов [26,27]. В данной статье мы рассмотрим такие алгоритмы как *жадный* подход, *Reverse Influence Maximization*(*RIM*), *Two-phase Influence Maximization*(*TIM*) и *TIM+*, предложенные Кемпе *et al* [18], Боргс *et al* [28] и Танг *et al* [29] соответственно.

Кемпе *et al* [18] рассмотрел 2 модели диффузии (распространения влияния) в социальной сети: *independent cascade* (*IC*) и *linear threshold* (*LT*) модели. Здесь мы рассмотрим IC модель. Задача максимизации влияния ставится следующим образом:

Пусть G социальная сеть (граф отношений и взаимодействий) с набором узлов V и набором направленных дуг E. $|V| = n$, $|E| = m$. Предположим, что каждой направленной дуге e соответствует *вероятность распространения* (*propagation probability*) $p(e) \in [0,1]$. При данной G, модель *independent cascade* рассматривает пошаговое (относительно времени) распространение влияния следующим образом:

- На шаге 1, мы *активируем* выбранное множество узлов S из G, определяя все остальные узлы в G *неактивными.*
- Если узел u был активирован на шаге i, тогда для каждой дуги e, исходящей из u к неактивному узлу v, u имеет вероятность $p(e)$ активировать v на шаге $i + 1$. После шага $i + 1$, u больше не может активировать ни один узел.
- Если узел стал активным, то он остается таковым на всех последующих шагах.

Пусть $I(S)$ - количество активных узлов после завершения процесса (процесс останавливается, если ни один узел не может быть активирован). S называют начальным множеством (*seed set*), а $I(S)$ - распространением (*spread*) S.

При данном графе G, константном k, задача максимизации влияния при модели IC требует найти начальный набор S, $|S| = k$, с максимальным ожидаемым распространением $E[I(S)]$. Другими словами, мы ищем множество, которое с большей вероятностью сможет активировать наибольшее число узлов.

Подход, предложенный Кемпе *et al* (называемый жадным) фактически начинает с пустого начального множества $S = \varnothing$, а затем на каждой итерации добавляет в множество элемент u, при котором прирост распространения $E[I(S)]$ будет наибольшим.

Другими словами:

$$\arg\max_{v \in V} \left(E\left[I(S \cup \{v\})\right] - E\left[I(S)\right] \right) \quad (8)$$

Хотя *жадный подход* является довольно примитивным, вычисление ожидания распространения $E[I(S)]$ является *#P*-сложной задачей. В связи с этим, для адекватной оценки $E[I(S)]$ Кемпе *et al* используют метод Монте Карло: для каждой дуги e мы "подбрасываем монету" и с вероятностью $1 - p(e)$ убираем дугу e. Полученный граф назовем g, а $R(S)$ - множество узлов g, для которых существует путь из какого-либо узла S. Кемпе *et al* доказали, что ожидаемый размер $R(S)$ равен $E[I(S)]$. Таким образом, мы генерируем несколько экземпляров g, вычисляем $R(S)$ для каждого случая, а затем берем их среднее значение для оценки $E[I(S)]$.

При достаточно большом количестве экземпляров r для измерения $E[I(S)]$ *жадный* подход с достаточно большой вероятностью предоставляет $(1-1/e-\varepsilon)$ -приближенное решение при IC модели [18] где ε - константа, зависящая от G, и от r. Кемпе *et al* предлагает рассматривать r = 10000, и последние представленные работы используют этот же выбор.

Не смотря на то, что *жадный* подход является обобщенным и эффективным, он влечет за собой огромные вычислительные расходы, т. к. его временная сложность равняется $O(kmnr)$ (k итераций для r графов по n узлов и m дуг).

Только недавно (2014г.) Боргс *et al* [28] сделали теоретический прорыв и предложили алгоритм с временной сложностью $O\left(kl^2(m+n)\log^2 n/\varepsilon^3\right)$. Боргс *et al* показали, что их алгоритм предоставляет $(1-1/e-\varepsilon)$ - приближенное решение с как минимум $1-n^{-l}$ вероятностью и доказали, что этот результат является близким к оптимальному, так как любой алгоритм, который предоставляет такое приближение с хотя бы постоянной вероятностью должен работать за время $\Omega(m+n)$.

Основной причиной неэффективности *жадного* подхода является необходимость оценивать ожидаемое распространение множества узлов при каждой итерации алгоритма. Действительно, большинство из этих оценок проводятся впустую, так как при каждой итерации ищется набор узлов с наибольшим ожидаемым распространением. Таким образом, например, на первой итерации, при поиске первого узла для начального набора узлов, без каких-либо имеющихся данных об ожидаемом распространении узлов нам будет необходимо оценить $E[I(\{v\})]$ для каждого v, из G. В этом случае вычислительные расходы только первой итерации составляют $O(mnr)$.

Боргс *et al* [28] постарался избежать ограничений *жадного* подхода и предложил совершенно иной метод максимизации влияния для модели IC. В работе Танга *et al* [29], к которой мы обратимся позже, этот метод называют *Reverse Influence Sampling* (*RIS*). *RIS* основывается на понятии *Reverse Reachable Set* или *RR*(*v*) -- набор всех узлов *u*, из которых есть путь в *v* в *g*, где *g* - экземпляр вероятностного графа *G*. Боргс *et al* показали, что если *RR*(*v*) пересекается с каким-то набором узлов *S* с вероятностью *p*, то если мы выберем *S* в качестве начального множества, при распространении влияния в *G* мы будем иметь вероятность *p* активировать узел *v*. Основываясь на этих результатах, *RIS* выполняется в два шага:

1. Сгенерировать определенное количество случайных *RR* множеств из *G* (*RR* множество для случайного узла *v* из случайного экземпляра *g*)
2. Рассмотреть задачу максимального покрытия [30]: выбрать *k* узлов, которые покроют максимальное количество сгенерированных *RR* множеств. (*v покрывает S тогда и только тогда, когда* $v \in S$).

На интуитивном уровне этот алгоритм объясняется так: если узел принадлежит большому количеству *RR* множеств, то его ожидаемое распространение должно быть велико.

Несмотря на то, что сложность алгоритма является окололинейной, он может требовать большие вычислительные расходы в связи с необходимым количеством генерируемых *RR* множеств. Боргс *et al* предложили генерировать эти множества до тех пор, пока количество рассмотренных узлов и дуг не превысит порог τ. Они показали, что если для τ установлено значение $\Theta\left(k(m+n)\log^2 n/\varepsilon^3\right)$, то *RIS* работает за линейное по отношению к τ время и возвращает $(1-1/e-\varepsilon)$ -приближенное решение с хотя бы постоянной вероятностью. Затем они увеличивают вероятность успеха до хотя бы $1-n^{-l}$, увеличивая τ на множитель с l и запуская *RIS* $\Omega(l \log n)$ раз. Алгоритм не обладает практической эффективностью и требует больших временных затрат уже при обработке графов с десятками тысяч узлов и дуг. Множитель (ε^{-3}) в оценке временной сложности появляется из-за необходимости выбрать τ достаточно большим, т. к. при малом τ часто встречающиеся элементы *RR* множеств повлияют на выбор конечного ответа, и результат будет далек от оптимального.

Танг *et al* [29] предложил новый алгоритм *Two-phase Influence Maximization*(*TIM*), который использует идеи *RIS*, но обходит его ограничения, предлагая временную сложность $O\left((k+l)(m+n)\log n/\varepsilon^2\right)$. Такая сложность отличается на множитель ($\log n$) от сложности $\Omega(m+n)$, которая, как доказали Боргс *et al* [28], является нижней границей сложности любого алгоритма (при фиксированных *k*, *l* и ε). Как следует из названия и подобия алгоритму *RIS*, *TIM* состоит из 2х шагов:

1. <u>*Оценка параметра.*</u> Этот шаг оценивает нижнюю границу максимально

возможного ожидаемого распространения по всем k-размерным множествам узлов и использует эту границу для получения параметра θ.

2. *Выбор узлов.* На этом шаге генерируются θ случайных RR множеств из G, а затем выбирается k-размерное множество узлов S*, которое покрывает наибольшее число RR множеств. R* возвращается в качестве результата работы алгоритма.

Выбор узлов в *TIM* схож с выбором узлов в *RIS*, за исключением того, что здесь генерируется уже заранее определенное число случайных *RR* множеств. Выбор множества *S** происходит с помощью *жадного* подхода решения задачи *максимального покрытия*, т. е. выбором *k* узлов, *покрывающих* наибольшее количество множеств узлов. Этот подход возвращает $(1-1/e-\varepsilon)$ - приближенное решение и имеет реализацию с линейным временем:

1: *S* = ∅;*
2: *for j=1 to k do*
3: *Найти узел v(j), который покрывает наибольшее число RR множеств.*
4: *Добавить v(j) в S*.*
5: *Убрать все RR множества, покрытые v(j).*
6: *return S**

Для оценки параметра (шаг 1), авторы приводят и доказывают теорему:

Th: Если θ удовлетворяет неравенству

$$\theta \geq (8+2\varepsilon)n \cdot \frac{l\log n + \log C_k^n + \log 2}{OPT \cdot \varepsilon^2} \qquad (9)$$

то шаг 2 алгоритма TIM возвращает $(1-1/e-\varepsilon)$*-приближенное решение с хотя бы* $1-n^{-l}$ *вероятностью.*

Поскольку значение *θ* трудно выбрать согласно неравенству (так как значение *OPT* неизвестно) Танг *et al* [] находят нижнюю оценку значения *OPT* (называя ее *KPT*) *и* используют ее в конечной формуле для:

$$\theta = \lambda / KPT \qquad (10)$$

где

$$\lambda = (8+2\varepsilon)\frac{l\log n + \log C_k^n + \log 2}{\varepsilon^2} \qquad (11)$$

Отличие *TIM+* от *TIM* заключается в том, что *TIM+* добавляет промежуточный шаг для улучшения *KPT* (нижней оценки *OPT*), что влечет за собой некоторое небольшое уменьшение вычислительных расходов алгоритма.

4. Pattern Matching.

Проблема поиска и сопоставления с образцом в графовых базах данных имеет широкое применение в таких областях как социальные сети, компьютерный дизайн, машинное зрение, электроника и биология. Из-за разнообразия в характеристиках графов этих областей, а так же различия постановок проблемы, графовое сопоставление с образцом объединяет в себе целый набор взаимосвязанных задач.

Основная цель *pattern matching* – поиск всех вхождений заданного шаблона в графовой базе данных. Формально:

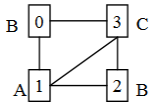
- *Граф* $G = (V, E)$, с множеством вершин V и множеством дуг E. Вершины и/или дуги могут быть помечены и/или иметь различные атрибуты.
- *Граф-образец* **(**или *образец для запроса***)** $P = (V_p, E_p)$ определяет структурные и семантические требования, которыми должен обладать подграф M графа G, чтобы соответствовать образцу P.

Требуется найти множество всех подграфов M, «соответствующих» шаблону P. Чаще всего, сопоставление с образцом описывается в терминах *изоморфизма подграфов*, что, как известно, является NP-сложной задачей. Как описывает Галахер [31] в своем обзоре работ по этой проблеме, все известные алгоритмы поиска изоморфизма подграфа имеют экспоненциальную сложность от размера графа, что не позволяет напрямую решать эту задачу на больших графах. В качестве возможного решения предлагаются либо аппроксимирующие алгоритмы поиска изоморфизма, либо точные алгоритмы, но применяемы только к конкретной части всего объема данных. Второй подход обычно достигается предварительной обработкой, которая позволяет отбросить части данных, которые наверняка не принесут результата. Эта фильтрация данных обычно называется *выборкой кандидатов.*

Первыми такой подход использовали в своем алгоритме *GraphGrep* Шаша *et al* [32]. Алгоритм содержит 3 компоненты: (1) построение метаданных для графа - индексирование с помощью наборов путей (этот шаг выполняется только единожды), (2) фильтрование базы данных на основе полученного шаблона и индексирования с целью уменьшения пространства поиска, (3) выполнение точного поиска.

1. Построение индекса. На этом этапе строится «представление графа с помощью путей». Авторы рассматривают базу данных из нескольких графов, что можно интерпретировать как несколько компонент связности одного графа. Так как каждый узел имеет метку (*label*) и уникальный идентификатор (*id*), то «представление с помощью путей» является набором путей меток или label-path (пути длиной от 1 до l_p, Shasha *et al* [32] берут значение $l_p = 4$), где каждый путь меток – набор из путей идентификаторов или id-path. Эти данные формируют хэш-таблице, где ключами являются хэш-значения путей меток, а в ячейки записываются количества путей идентификаторов по этому ключу. Такую хэш-

таблицу для графа авторы называют сигнатурой (*fingerprint*) графа. Пример построения показан на рис. 2.



(a)

A={(1)} AB={(1, 0), (1,2)} AC ={(1, 3)} ACBA={…}
ABCA={(1 ,0 ,3 ,1),(1, 2, 3, 1)} CB={(3,0),(3,2)} C={(3)}
CBAB={((3, 0, 1 ,2),(3 ,2 ,1 ,0)} B={(0),(2)} BA={(0,1),(2,1)}
BAB={(0,1,2), (2,1,0)} ABC ={(1, 3, 0), (1,3,2)} ACB={…}
ABCB={…} BC={…} BAC={…} BCB={…} CBA={…}
BABC={…}CBAC={…} CABC={…} CAB={(3,1,0), (3,1,2)}
BACB={…}BCBA={…}BCAB={…} BCA={…} CA={(3,1)}

(b)

Key	g_1
h(CA)	1
……	
h(ABCB)	2

(c)

Рис. 2. (*a*) *входной граф* g_1, (*b*) *его представление с помощью путей и* (*c*) *сигнатура.*

Fig. 2. (*a*) *input graph* g_1, (*b*) *its path representation and* (*c*) *fingerprint*.

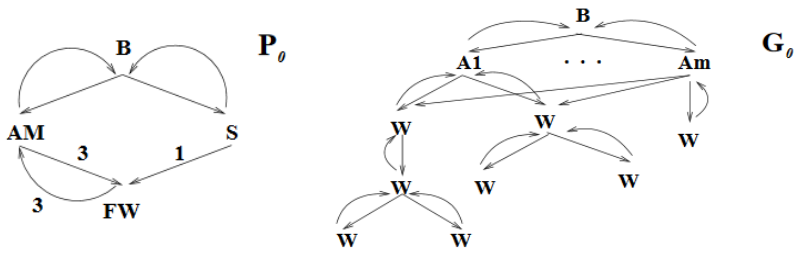
Пусть n и k количество узлов и максимальная степень узла в графе соответственно. Тогда в худшем случае сложность индексирования и представления с помощью путей составляет $O(nk^{l_p})$, а занимаемая память $O(l_p n k^{l_p})$ для каждой отдельной компоненты связности.

2. Выбор кандидатов. На этом этапе строится сигнатура графа-шаблона и сравнивается с сигнатурой графа данных. Если граф данных состоит из нескольких компонент связности (для каждой из которых мы предположительно построили сигнатуру), то компоненты, в чьей сигнатуре хотя бы одно значение меньше соответствующего значения в сигнатуре шаблона, можно отбросить. Остальные компоненты содержат одно или более вхождение подграфа, соответствующего образцу (шаблону).

3. Поиск соответствующего подграфа. Алгоритм обходит граф-образец в глубину и разбивает ветви обхода на последовательности (называемые *patterns*) накладывающихся друг на друга путей меток. Части графа-кандидата, чьи пути идентификаторов соответствуют этим последовательностям, склеиваются (избавляясь от наложений) для построения соответствующего образцу подграфа. Вычислительная сложность этого шага зависит от числа последовательностей (*patterns*) образца, которое сложно оценить в общем случае. Грубо говоря, оно прямо пропорционально размеру образца и максимальной степени среди узлов в образце и чем больше l_p, тем меньше p.

Если $\overline{n}$ – максимально число узлов с одной и той же меткой, то временная сложность этого шага в худшем случае составляет $O((\overline{n}k^{l_p})^p)$.

Задача сопоставления с образцом, описанная на языке изоморфизмов графа является NP-сложной. Это сильно затрудняет масштабируемость при поиске точных вхождений образца. Более того, требуется нахождение биекций, которые чаще всего ставят строгие ограничения на типы образцов как показано в примере, приведенном в [33].



(*a*) (*b*)

Рис. 3. Организация оборота наркотиков: (*a*) *граф-образец и* (*b*) *граф данных*

Fig. 3. Drug trafficking. Pattern (*a*) *and data graph* (*b*)

Пример. Рассмотрим структуру организации оборота наркотиков, изображенную в виде графа-образца P_0 на рис. 3 (*a*). Глава («boss» - B) осуществляет надзор и ведет контроль операций через группу управляющих помощников (assistant manager - AM). AM контролируют деятельность иерархической структуры работников (field worker – FW) вплоть до трех уровней иерархии, обозначенных меткой дуги «3». FW доставляет наркотики, собирает выручку и выполняет другие поручения. Они отчитываются перед AM напрямую или косвенно, а AM напрямую отчитываются перед боссом. Босс так же может передавать сообщения через секретаря (S) верхнему уровню FW (обозначено меткой дуги «1»).

Рассмотрим группировку по сбыту наркотиков, представленную графом на рис. 3 (*b*), где $A_1,\ldots A_m$ это AM'ы, в то время как A_m является и AM и секретарем (S).

Мы хотим определить всех подозреваемых, которые входят в группировку, с помощью поиска вхождений P_0 в G_0. Однако сопоставление с образцом через изоморфизм подграфов не справится с этой задачей по следующим причинам.

- Узлы AM и S из P_0 должны соответствовать одному и тому же узлу A_m из G_0, что противоречит биекции.
- Узел AM из P_0 соответствует нескольким узлам из G_0 ($A_1,\ldots,A_m$). Это отношение не может быть описано функцией от узлов P_0 в узлы G_0.
- Дуга из AM в FW в P_0 означает, что AM контролирует 3 уровня FW. Это должно соответствовать пути определенной длины в G_0, а не одной дуге. Изоморфизм, при котором дуга переходит в дугу не подойдет в этой ситуации.

К таким выводам пришел B. Фан *et al* [34]. В 2010 г. Он опубликовал работу, в которой пересмотрел практически целиком задачу сопоставления с образцом, введя понятие ограниченной симуляции (*bounded simulation*). Согласно его терминологии:

- *Граф данных* $G = (V, E, f_A)$, где f_A – функция от узлов, при которой

$f_A(u)$ – кортеж ($A_1=a_1,\ldots,A_n=a_n$), где a_i – константа (не обязательно числовая), а A_i – атрибут u, записываемый как $u.A_i=a_i$.

- <u>*Граф-образец*</u> $P = (V_p, E_p, f_v, f_e)$. f_v – функция от узлов образца, сопоставляющая узлу предикат, составленный из объединения атомарных формул вида A op a, где a – константа, A – атрибут, а op - оператор сравнения ($<, \leq, =, \neq, >, \geq$). f_e – функция от дуг образца, такая что $f_e(u,u')$ либо константа k, либо символ «*».

Тогда вводится следующее определение ограниченной симуляции:

<u>Опр.</u> *Граф G соответствует образцу P в терминах ограниченной симуляции, если существует бинарное отношение* $S \subseteq V_p \times V$, *т. ч.:*

- *Для каждого* $u \in V_p$ *существует* $v \in V$, *т. ч.* $(u,v) \in S$;
- *Для каждого* $(u,v) \in S$ (*а*) *атрибут узла* v $f_A(v)$ *удовлетворяет предикату* $f_v(u)$ *узла u. То есть для каждой атомарной формулы A* op *a из* $f_v(u)$, *v.A = a′ определено в* $f_A(v)$ *и a′* op *a.* (*б*) *Для каждой дуги* (u,u') *из* E_p *существует непустой путь* $\rho = v/\ldots/v'$ *из G, т. ч.* $(u,u') \in S$ *и длина* len (ρ) $\leq$ *k, если* $f_e(u,u')$ *это константа k.*

Такое бинарное отношение *S* называют вхождением (*match*) *P* в *G* и обозначают как $P \trianglelefteq G$.

Интуитивно, $(u,v) \in S$, если v соответствует критериям поиска, установленным $f_v(u)$, и каждая исходящая из u дуга (u,u') соответствует непустому пути ρ, длины $f_e(u,u')$, если $f_e(u,u') = k$, или неограниченной длины, если $f_e(u,u') = *$.

Поскольку вхождение *P* в *G* может быть несколько, Фан *et al* [30] ставят и доказывают утверждение:

<u>Утв.</u> *Для любого графа G и любого образца P, если* $P \trianglelefteq G$, *то существует единственное максимальное вхождение* S_M, *т. е. для любого вхождения S образца P в граф G верно, что* $S \subseteq S_M$.

На основе введенных определений графа, образца и ограниченной симуляции, авторы пересматриваю задачу сопоставления с образцом следующим образом: *Для данного графа G и образца P требуется найти максимальное вхождение P в G, если* $P \trianglelefteq G$.

Фан *et al* [34] приводят алгоритм, имеющий временную сложность

$$O\left(|V|\,|E| + |E_p|\,|V|^2 + |V_P|\,|V|\right) \qquad (12)$$

Такой алгоритм, в отличие от NP-сложных алгоритмов изоморфизма подграфов, завершается за кубическое время (т. к. в худшем случае $|E|$ это $O(|V^2|)$).

Так как кубическая сложность все еще все еще не подходит для больших графов, Фан *et al* [34] предлагают алгоритмы приращений (*incremental match algorithms*) для поиска вхождений в случае, когда граф G изменяется посредством удаления и вставки дуг. Это позволяет найти вхождение единожды, а затем его эффективно обновлять при изменении G, без нужды запускать весь процесс самого начала.

5. Network motifs.

Множество биологических сетей, представляемых с помощью графов, содержат определенные подсети (подграфы), которые появляются в данной сети с гораздо большей частотой, чем в случайных сетях. Например, в сети взаимодействия протеин-протеин некоторые трех- и четырехузельные подграфы встречаются гораздо чаще, чем в случайной сети со схожими математическими свойствами. В 2002 г. Мило *et al* [35] предложил использовать такие «топологические блоки» для изучения структуры и строения сложных сетей, назвав такие блоки *сетевыми мотивами* (*network motifs*). С тех пор было проведено много исследований на эту тему, некоторые из которых ориентировались на биологической составляющей сетевых мотивов, другие – на теории алгоритмов их поиска. В данной статье не будет обсуждаться само понятие сетевых мотивов, а будут рассмотрены некоторые алгоритмы, которые позволяют анализировать сложные природные и другие сети на предмет нахождения в них сетевых мотивов.

Процесс обнаружения сетевых мотивов обычно состоит из двух шагов: 1) генерация набора случайных сетей, чьи свойства зависят от свойств сети, в которой производится поиск (называемой *настоящая сеть*), 2) подсчет вхождений подграфов в настоящей и случайных сетях. Мило *et al* [35] в своей работе описал несколько сложных графов сетей, встречающихся в природе, содержащих сетевые мотивы. Его работа была ориентирована на обоснование важности задач, которую эти мотивы могут выполнять, а для вычислений использовала метод полного перебора всех возможных подграфов сети с данным количеством узлов. Очевидно, что временная сложность такого алгоритма очень быстро растет с увеличением количества узлов подграфа и ростом самого графа данных. Такой алгоритм справлялся с поиском небольших сетевых мотивов, но нахождение пяти- или шестиузловых мотивов не представлялось вычислительно возможным.

Первым алгоритмом, отличающимся от простого перебора стал *mfinder* – алгоритм, основанный на методе генерации выборки, предложенный Каштан *et al* [36]. Алгоритм оценивает концентрации подграфов направленной или ненаправленной сети, из которых можно сделать вывод о наличии или отсутствии сетевых мотивов. Выборка начинается с выбора произвольной дуги

графа, таким образом порождая подграф размера 2, а затем увеличивает подграф, добавляя в него случайную касающуюся его дугу. Затем, алгоритм продолжает выбирать случайные соседние дуги, пока размер подграфа не достигнет *n*. Наконец, в отобранный подграф добавляются все дуги между полученными *n* узлами. Используя такой подход, необходимо использовать несмещенную выборку, поскольку процедура выборки составляет экземпляры неоднородно. Для этого, Каштан *et al* [36] предложил систему взвешенных подграфов, используя вероятность генерации подграфа в процессе выборки и сопоставляя каждому экземпляру соответствующий вес. Таким образом, вероятные подграфы получают сравнительно меньший вес, чем маловероятные подграфы, обеспечивая тем самым справедливую оценку концентраций подграфов. Так, если положить *P* – вероятность генерации образца подграфа типа *i*, $W = 1/P$, тогда S_i накапливает весовой показатель для подграфов типа *i*: $S_i = S_i + W$. После генерации всех экземпляров, полагая, что было сгенерировано *L* различных типов подграфов, то концентрация подграфа типа *i* вычисляется формулой:

$$C_i = \frac{S_i}{\sum_{k=1}^{L} S_k} \tag{13}$$

Вычислительная сложность такого алгоритма асимптотически независима от размера графа данных и составляет $O(n^n)$ для каждого образца подграфа размером *n*. Но необходимо упомянуть, что процесс выборки может сгенерировать один и тот же подграф несколько раз, тратя время без получения какой-либо информации. Несмотря на это, алгоритм с выборкой является более эффективным подходом, нежели случайный перебор, хоть и оценивает концентрации лишь примерно. Алгоритм может находить мотив вплоть до размера 6 узлов.

За прошедшее десятилетие были предложены такие алгоритмы, как *FANMOD*[37], *Grochow-Kellis*[38], *FPF*[39], *MODA*[40], *G-trie*[41]. Среди упомянутых *g-trie* является самым быстрым, используя новую структуру хранения набора подграфов, предложенную Педро Рибейро и Фернандо Силва в 2010г. *G-trie* это многопутевое дерево, в котором каждая вершина хранит информацию об одном узле графа данных и соответствующих дугах, ведущих к его предкам. Путь от корня к вершине дерева соответствует конкретному графу, а потомки вершины дерева содержат одинаковый подграф. Построение *g-trie* подробно описано в [41]. Основной идеей подсчета количества вхождений подграфа является обратный обход дерева по всем возможным подграфам, при котором одновременно проводятся проверки на изоморфизм подграфов. Важное достоинство такого подхода к хранению при поиске сетевых мотивов заключается в том, что нет необходимости подсчитывать количество вхождений подграфа в случайную сеть, если такой подграф не присутствует в основной сети. Однако главным недостатком *g-trie* является

большой объем используемой памяти, что может ограничить размер находимых мотивов при использовании персонального компьютера со средним объемом памяти.

Стоит отметить еще одно направление развития теории сетевых мотивов – мотивы динамических сетях. С ростом популярности социальных сетей появился и интерес к нахождению сетевых мотивов в динамических сетях. Такие сети чаще всего предлагают временные метки для каждой дуги своего графа. В этом случае усложняется понятие «соседних дуг», так как, если дуга обозначает взаимодействие между двумя пользователями, то соприкасающиеся дуги считаются «соседними» только если их временные метки находятся друг от друга на расстоянии не больше чем некоторый временной порог ΔT. Существующие алгоритмы поиска мотивов статических графов игнорируют временной аспект динамических сетей, поэтому не решают поставленную задачу.

На данный момент, самым современным алгоритмом поиска сетевых мотивов в динамических сетях считается *COMMIT*, представленный Гурукар *et al* [42] в 2015г., который, согласно проведенным экспериментам, на 2 порядка величины быстрее представленных ранее алгоритмов. В целях экономии времени в данной статье *COMMIT* будет представлен только вкратце. За подробным описанием заинтересованный читатель может обратиться к оригинальной статье [42].

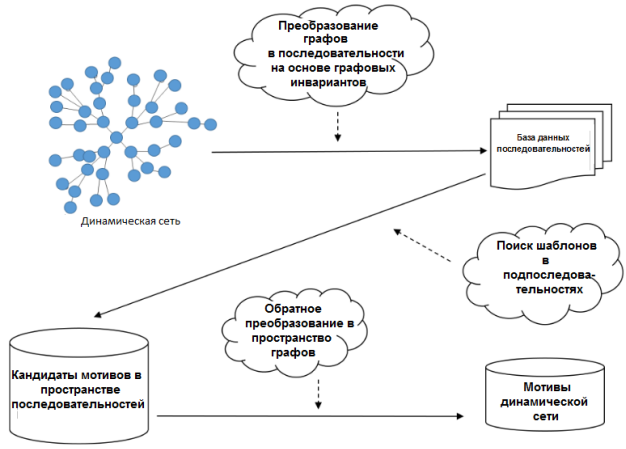


Рис. 4. Конвейер алгоритма COMMIT.

Fig. 4.Pipeline of COMMIT

На рис. 4 представлен конвейер алгоритма COMMIT. Основной идея алгоритма заключается в первом шаге: перевод каждой временно связанной компоненты (компонента связности, в которой метки дуг находятся в пределах одного временного периода, ограниченного порогом ΔT) в математическую последовательность ее взаимодействий. Здесь происходит необходимый анализ

полученной базы данных последовательностей и поиск в ней часто встречающихся шаблонов подпоследовательностей, которые могут представлять мотивы сети. Затем, эти подпоследовательности преобразовываются обратно в пространство графов для проверки и вычисления конечного результата. Такой подход к нахождению мотивов приводит к экономии времени, так как большинство вычислений проходят именно в пространстве последовательностей, чей размер значительно меньше размера пространства графов. Также, проверки на изоморфизм проводятся только на небольшой доле графов-кандидатов на статус мотива сети, полученных в ходе анализа.

Заключение

Таким образом, приведенный анализ показывает, что описанные методы с некоторыми оговорками позволяют решить поставленные задачи. Предложенные алгоритмы можно считать масштабируемыми, однако усовершенствование их временных сложностей как всегда является одним из приоритетных направлений дальнейших исследований. В качестве альтернативного пути развития отметить разработку инкрементальных версий алгоритмов. Такая версия алгоритма уже реализована для решения задачи максимизации влияния.

Далее, помимо рассмотренных в обзоре задач на графовых базах данных серьезный интерес представляют задачи поиска эффективных алгоритмов кластеризации и классификации, задачи статистического и пространственного анализа, визуализация данных. Хотя в настоящее время интерес специалистов к этим задачам менее выражен, их полное решение позволило бы резко повысить эффективность анализа и хранения больших данных.

Список литературы

[1]. Pettey C., Goasduff L. (2011) Gartner, Inc. Gartner Says Solving 'Big Data' Challenge Involves More Than Just Managing Volumes of Data (online publication). Available at: http://www.gartner.com/newsroom/id/1731916, accessed 07.08.2016

[2]. DIS Group, (2014) “Big data.” http://www.dis-group.ru/solutions/data_ management/big_data/, accessed 07.08.2016.

[3]. Бартенев М., Вишняков И . “Использование графовых баз данных в целях оптимизации анализа биллинговой информации,” Инженерный журнал: наука и инновации, no. 11, 2013.

[4]. Manyika J., Chui M., Brown B., Bughin J., Dobbs R., Roxburgh C., Byers A. H . “Big data: The next frontier for innovation, competition, productivity,” 2011.

[5]. Jeh G., Widom J . “Simrank: a measure of structural-context similarity,” in Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 538–543, ACM, 2002.

[6]. Fogaras D. and R´acz B . “Scaling link-based similarity search,” in Proceedings of the 14th international conference on World Wide Web, pp. 641–650, ACM, 2005.

[7]. He G., Feng H., Li C., Chen H . “Parallel simrank computation on large graphs with iterative aggregation,” in Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 543–552, ACM, 2010.
[8]. Li C., Han J., He G., Jin X., Sun Y., Yu Y., Wu T . “Fast computation of simrank for static and dynamic information networks,” in Proceedings of the 13th International Conference on Extending Database Technology, pp. 465–476, ACM, 2010.
[9]. Li P., Liu H., Yu J. X., He J., Du X . “Fast single-pair simrank computation.,” in SDM, pp. 571–582, SIAM, 2010.
[10]. Yu W., Lin X., Le J . “Taming computational complexity: Efficient and parallel simrank optimizations on undirected graphs,” in International Conference on Web-Age Information Management, pp. 280–296, Springer, 2010.
[11]. Lizorkin D., Velikhov P., Grinev M., Turdakov D . “Accuracy estimate and optimization techniques for simrank computation,” The VLDB Journal The International Journal on Very Large Data Bases, vol. 19, no. 1, pp. 45–66, 2010.
[12]. Yu W., Lin X., Zhang W., Chang L., Pei J . “More is simpler: Effectively and efficiently assessing node-pair similarities based on hyperlinks,” Proceedings of the VLDB Endowment, vol. 7, no. 1, pp. 13–24, 2013.
[13]. Yu W., Zhang W., Lin X., Zhang Q., Le J . “A space and time efficient algorithm for simrank computation,” World Wide Web, vol. 15, no. 3, pp. 327–353, 2012
[14]. Maehara T., Kusumoto M., Kawarabayashi K.-i . “Scalable simrank join algorithm,” in 2015 IEEE 31st International Conference on Data Engineering, pp. 603–614, IEEE, 2015.
[15]. Fujiwara Y., Nakatsuji M., Shiokawa H., Onizuka M . “Efficient search algorithm for simrank,” in Data Engineering (ICDE), 2013 IEEE 29th International Conference on, pp. 589–600, IEEE, 2013.
[16]. Zhu R., Zou Z., Li J . “Simrank computation on uncertain graphs,” in 2016 IEEE 32nd International Conference on Data Engineering (ICDE), pp. 565–576, May 2016.
[17]. Domingos P., Richardson M . “Mining the network value of customers,” in Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 57–66, ACM, 2001.′
[18]. Kempe D., Kleinberg J., Tardos E . “Maximizing the spread of influence through a social network,” in Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 137–146, ACM, 2003.
[19]. Chen W., Wang C., Wang Y . “Scalable influence maximization for prevalent viral marketing in large-scale social networks,” in Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 1029–1038, ACM, 2010.
[20]. Goyal A., Bonchi F., L. Lakshmanan V . “A data-based approach to social influence maximization,” Proceedings of the VLDB Endowment, vol. 5, no. 1, pp. 73– 84, 2011.
[21]. Jung K., Heo W., Chen W . “Irie: Scalable and robust influence maximization in social networks,” in 2012 IEEE 12th International Conference on Data Mining, pp. 918–923, IEEE, 2012.
[22]. Kim J., Kim S.-K., Yu H . “Scalable and parallelizable processing of influence maximization for large-scale social networks?,” in Data Engineering (ICDE), 2013 IEEE 29th International Conference on, pp. 266–277, IEEE, 2013.
[23]. Kim D., Lee J.-G., Lee B. S . “,”
[24]. Li G., Chen S., Feng J., Tan K.-l., Li W.-s . “Efficient location-aware influence maximization,” in Proceedings of the 2014 ACM SIGMOD international conference on Management of data, pp. 87–98, ACM, 2014.

[25]. Wang X., Zhang Y., Zhang W., Lin X . “Distance-aware influence maximization in geo-social network,”
[26]. Khan A., Zehnder B., Kossmann D . “Revenue maximization by viral marketing: A social network host’s perspective.”
[27]. Li H., Bhowmick S. S., Cui J., Gao Y., Ma J . “Getreal: Towards realistic selection of influence maximization strategies in competitive networks,” in Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, pp. 1525– 1537, ACM, 2015.
[28]. Borgs C., Brautbar M., Chayes J., Lucier B . “Maximizing social influence in nearly optimal time,” in Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 946–957, Society for Industrial and Applied Mathematics, 2014.
[29]. Tang Y., Xiao X., Shi Y . “Influence maximization: Near-optimal time complexity meets practical efficiency,” in Proceedings of the 2014 ACM SIGMOD international conference on Management of data, pp. 75–86, ACM, 2014.
[30]. Mahdian M., Ye Y., Zhang J . “Improved approximation algorithms for metric facility location problems,” in International Workshop on Approximation Algorithms for Combinatorial Optimization, pp. 229–242, Springer, 2002.
[31]. Gallagher B . “Matching structure and semantics: A survey on graph-based pattern matching,” AAAI FS, vol. 6, pp. 45–53, 2006.
[32]. Giugno R., Shasha D . “Graphgrep: A fast and universal method for querying graphs,” in Pattern Recognition, 2002. Proceedings. 16th International Conference on, vol. 2, pp. 112–115, IEEE, 2002.
[33]. Natarajan M . “Understanding the structure of a drug traffcking organization: a conversational analysis,” Crime Prevention Studies, vol. 11, pp. 273–298, 2000.
[34]. Fan W., Li J., Ma S., Tang N., Wu Y., Wu Y . “Graph pattern matching: From intractable to polynomial time,” Proc. VLDB Endow., vol. 3, pp. 264–275, Sept. 2010.
[35]. Milo R., Shen-Orr S., Itzkovitz S., Kashtan N., Chklovskii D., Alon U . “Network motifs: simple building blocks of complex networks,” Science, vol. 298, no. 5594, pp. 824–827, 2002.
[36]. Kashtan N., Itzkovitz S., Milo R., Alon U . “Mfinder tool guide,” Department of Molecular Cell Biology and Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot Israel, Tech Rep, 2002.
[37]. Wernicke S., Rasche F . “Fanmod: a tool for fast network motif detection,” Bioinformatics, vol. 22, no. 9, pp. 1152–1153, 2006.
[38]. Grochow J. A., Kellis M . “Network motif discovery using subgraph enumeration and symmetry-breaking,” in Annual International Conference on Research in Computational Molecular Biology, pp. 92–106, Springer, 2007.
[39]. Schreiber F., Schwobbermeyer H . “Frequency concepts and pattern detection for the analysis of motifs in networks,” in Transactions on computational systems biology III, pp. 89–104, Springer, 2005.
[40]. Omidi S., Schreiber F., Masoudi-Nejad A . “Moda: an efficient algorithm for network motif discovery in biological networks,” Genes & genetic systems, vol. 84, no. 5, pp. 385–395, 2009.
[41]. Ribeiro P., Silva F . “G-tries: an efficient data structure for discovering network motifs,” in Proceedings of the 2010 ACM Symposium on Applied Computing, pp. 1559–1566, ACM, 2010.
[42]. Gurukar S., Ranu S., Ravindran B . “Commit: A scalable approach to mining communication motifs from dynamic networks,” in Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, pp. 475–489, ACM, 2015.

Some problems on graph databases

R.I. Guralnik <guralnikr@gmail.com>
Saint-Petersburg State University,
University Embankment, 7-9, St Petersburg, 199034, Russia

Abstract. Graph databases appear to be the most popular and relevant among non-relational databases. Its popularity is caused by its relatively easy implementation in the problems in which data have big numbers of relations such as protein-protein interaction and others. With the development of fast internet connection, graph database found another interesting application in representation of social networks. Moreover, graph edges are storable which lowers graph traversing calculation costs. Such system appeared to be natural and in-demand in the era of Internet and social networks. The most significant by size and matter section of graph databases problems is data mining. It contains such problems as associative rules learning, data classification and categorization, clustering, regression analysis etc. In this review, data mining graph database problems are considered which are most commonly presented in modern literature. Their popularity is represented by the big number of publications on these problems on several recent years' major conferences. Such problems as influence maximization, motif mining, pattern matching and simrank problems are examined. For every type of a problem we analyzed different papers and described basic algorithms which were offered 10-15 years ago. We also considered state-of-the-art solutions as well as some important in-between versions. This review consists of 6 sections. Besides introduction and conclusion, each section is dedicated to its own type of graph database problem.

Keywords: graph databases; motif mining; influence maximization; pattern matching; simrank.

DOI: 10.15514/ISPRAS-2016-28(4)-12

For citation: R.I. Guralnik. Some problems on graph databases. Trudy ISP RAN/Proc. ISP RAS, vol. 28, issue 4, 2016, pp. 193-216 (in Russian). DOI: 10.15514/ISPRAS-2016-28(4)-12

References

[1]. Pettey C., Goasduff L. (2011) Gartner, Inc. Gartner Says Solving 'Big Data' Challenge Involves More Than Just Managing Volumes of Data (online publication). Available at: http://www.gartner.com/newsroom/id/1731916, accessed 07.08.2016

[2]. DIS Group, (2014) "Big data." http://www.dis-group.ru/solutions/data_ management/big_data/, accessed 07.08.2016.

[3]. Bartenev M.V., Vishnyakov I.E. "Graph database usage to optimize analysis of billing information", Engineering Journal: Science and Innovations [Inzhenernyi zhurnal: nauka i innovatsii], issue 11, 2013. Available at: http://engjournal.ru/catalog/it/hidden/1058.html

[4]. Manyika J., Chui M., Brown B., Bughin J., Dobbs R., Roxburgh C., Byers A. H . "Big data: The next frontier for innovation, competition, productivity," 2011.

[5]. Jeh G., Widom J . "Simrank: a measure of structural-context similarity," in Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 538–543, ACM, 2002.

[6]. Fogaras D. and R´acz B . “Scaling link-based similarity search,” in Proceedings of the 14th international conference on World Wide Web, pp. 641–650, ACM, 2005.
[7]. He G., Feng H., Li C., Chen H . “Parallel simrank computation on large graphs with iterative aggregation,” in Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 543–552, ACM, 2010.
[8]. Li C., Han J., He G., Jin X., Sun Y., Yu Y., Wu T . “Fast computation of simrank for static and dynamic information networks,” in Proceedings of the 13th International Conference on Extending Database Technology, pp. 465–476, ACM, 2010.
[9]. Li P., Liu H., Yu J. X., He J., Du X . “Fast single-pair simrank computation.,” in SDM, pp. 571–582, SIAM, 2010.
[10]. Yu W., Lin X., Le J . “Taming computational complexity: Efficient and parallel simrank optimizations on undirected graphs,” in International Conference on Web-Age Information Management, pp. 280–296, Springer, 2010.
[11]. Lizorkin D., Velikhov P., Grinev M., Turdakov D. “Accuracy estimate and optimization techniques for simrank computation,” The VLDB Journal The International Journal on Very Large Data Bases, vol. 19, no. 1, pp. 45–66, 2010.
[12]. Yu W., Lin X., Zhang W., Chang L., Pei J . “More is simpler: Effectively and efficiently assessing node-pair similarities based on hyperlinks,” Proceedings of the VLDB Endowment, vol. 7, no. 1, pp. 13–24, 2013.
[13]. Yu W., Zhang W., Lin X., Zhang Q., Le J . “A space and time efficient algorithm for simrank computation,” World Wide Web, vol. 15, no. 3, pp. 327–353, 2012
[14]. Maehara T., Kusumoto M., Kawarabayashi K.-i . “Scalable simrank join algorithm,” in 2015 IEEE 31st International Conference on Data Engineering, pp. 603–614, IEEE, 2015.
[15]. Fujiwara Y., Nakatsuji M., Shiokawa H., Onizuka M . “Efficient search algorithm for simrank,” in Data Engineering (ICDE), 2013 IEEE 29th International Conference on, pp. 589–600, IEEE, 2013.
[16]. Zhu R., Zou Z., Li J . “Simrank computation on uncertain graphs,” in 2016 IEEE 32nd International Conference on Data Engineering (ICDE), pp. 565–576, May 2016.
[17]. Domingos P., Richardson M . “Mining the network value of customers,” in Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 57–66, ACM, 2001.´
[18]. Kempe D., Kleinberg J., Tardos E . “Maximizing the spread of influence through a social network,” in Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 137–146, ACM, 2003.
[19]. Chen W., Wang C., Wang Y . “Scalable influence maximization for prevalent viral marketing in large-scale social networks,” in Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 1029–1038, ACM, 2010.
[20]. Goyal A., Bonchi F., L. Lakshmanan V . “A data-based approach to social influence maximization,” Proceedings of the VLDB Endowment, vol. 5, no. 1, pp. 73– 84, 2011.
[21]. Jung K., Heo W., Chen W . “Irie: Scalable and robust influence maximization in social networks,” in 2012 IEEE 12th International Conference on Data Mining, pp. 918–923, IEEE, 2012.
[22]. Kim J., Kim S.-K., Yu H . “Scalable and parallelizable processing of influence maximization for large-scale social networks?,” in Data Engineering (ICDE), 2013 IEEE 29th International Conference on, pp. 266–277, IEEE, 2013.
[23]. Kim D., Lee J.-G., Lee B. S . “Topical influence modeling via topic-level interests and interactions on social curation services,”

[24]. Li G., Chen S., Feng J., Tan K.-l., Li W.-s . “Efficient location-aware influence maximization,” in Proceedings of the 2014 ACM SIGMOD international conference on Management of data, pp. 87–98, ACM, 2014.
[25]. Wang X., Zhang Y., Zhang W., Lin X . “Distance-aware influence maximization in geo-social network,”
[26]. Khan A., Zehnder B., Kossmann D . “Revenue maximization by viral marketing: A social network host’s perspective.”
[27]. Li H., Bhowmick S. S., Cui J., Gao Y., Ma J . “Getreal: Towards realistic selection of influence maximization strategies in competitive networks,” in Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, pp. 1525– 1537, ACM, 2015.
[28]. Borgs C., Brautbar M., Chayes J., Lucier B . “Maximizing social influence in nearly optimal time,” in Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 946–957, Society for Industrial and Applied Mathematics, 2014.
[29]. Tang Y., Xiao X., Shi Y . “Influence maximization: Near-optimal time complexity meets practical efficiency,” in Proceedings of the 2014 ACM SIGMOD international conference on Management of data, pp. 75–86, ACM, 2014.
[30]. Mahdian M., Ye Y., Zhang J . “Improved approximation algorithms for metric facility location problems,” in International Workshop on Approximation Algorithms for Combinatorial Optimization, pp. 229–242, Springer, 2002.
[31]. Gallagher B . “Matching structure and semantics: A survey on graph-based pattern matching,” AAAI FS, vol. 6, pp. 45–53, 2006.
[32]. Giugno R., Shasha D . “Graphgrep: A fast and universal method for querying graphs,” in Pattern Recognition, 2002. Proceedings. 16th International Conference on, vol. 2, pp. 112–115, IEEE, 2002.
[33]. Natarajan M . “Understanding the structure of a drug traffcking organization: a conversational analysis,” Crime Prevention Studies, vol. 11, pp. 273–298, 2000.
[34]. Fan W., Li J., Ma S., Tang N., Wu Y., Wu Y . “Graph pattern matching: From intractable to polynomial time,” Proc. VLDB Endow., vol. 3, pp. 264–275, Sept. 2010.
[35]. Milo R., Shen-Orr S., Itzkovitz S., Kashtan N., Chklovskii D., Alon U . “Network motifs: simple building blocks of complex networks,” Science, vol. 298, no. 5594, pp. 824–827, 2002.
[36]. Kashtan N., Itzkovitz S., Milo R., Alon U . “Mfinder tool guide,” Department of Molecular Cell Biology and Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot Israel, Tech Rep, 2002.
[37]. Wernicke S., Rasche F . “Fanmod: a tool for fast network motif detection,” Bioinformatics, vol. 22, no. 9, pp. 1152–1153, 2006.
[38]. Grochow J. A., Kellis M . “Network motif discovery using subgraph enumeration and symmetry-breaking,” in Annual International Conference on Research in Computational Molecular Biology, pp. 92–106, Springer, 2007.
[39]. Schreiber F., Schwobbermeyer H . “Frequency concepts and pattern detection for the analysis of motifs in networks,” in Transactions on computational systems biology III, pp. 89–104, Springer, 2005.
[40]. Omidi S., Schreiber F., Masoudi-Nejad A . “Moda: an efficient algorithm for network motif discovery in biological networks,” Genes & genetic systems, vol. 84, no. 5, pp. 385–395, 2009.
[41]. Ribeiro P., Silva F . “G-tries: an efficient data structure for discovering network motifs,” in Proceedings of the 2010 ACM Symposium on Applied Computing, pp. 1559–1566, ACM, 2010.

[42]. Gurukar S., Ranu S., Ravindran B . “Commit: A scalable approach to mining communication motifs from dynamic networks,” in Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, pp. 475–489, ACM, 2015.