НОВІ ЗАСОБИ
КІБЕРНЕТИКИ, ІНФОРМАТИКИ,
ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
ТА СИСТЕМНОГО АНАЛІЗУ

**Д.А. РАЧКОВСКИЙ**

УДК 004.22+004.93′11

# БЫСТРЫЙ ПОИСК СХОДНЫХ ГРАФОВ ПО РАССТОЯНИЮ РЕДАКТИРОВАНИЯ

**Аннотация.** Дан обзор индексных структур для быстрого поиска по сходству объектов, представленных деревьями и графами. В качестве меры сходства использовано расстояние редактирования. Рассмотрено выполнение запросов точного поиска по сходству. В основном представлены алгоритмы на основе стратегии фильтрации и уточнения, использующие обратное индексирование. Кроме того, рассмотрены алгоритмы точного вычисления расстояния редактирования графов и его нижних и верхних границ.

**Ключевые слова:** поиск по сходству, графы, расстояние редактирования, ближайший сосед, индексные структуры, обратное индексирование.

## ВВЕДЕНИЕ

Поиск объектов базы, сходных с объектом-запросом по некоторой мере расстояния/сходства, называют поиском по сходству. Объекты-запросы могут не принадлежать базе. Будем считать заданной меру (функцию) расстояния/сходства между представлениями объектов, по которой выполняется поиск.

Широко применяются два типа запросов поиска по сходству. Диапазонный запрос (обозначим его rNN) возвращает объекты базы, расстояния которых от объекта-запроса не превышают радиуса запроса $r$. Запрос ближайшего соседа (обозначим его NN) возвращает объект базы, ближайший к объекту-запросу. Запрос $k$ ближайших соседей обозначим kNN. Другие типы запросов см. в [1] и ссылках к ней.

Результаты выполнения запроса all-pair similarity search (APSS), т.е. поиск всех пар объектов базы с $\text{dist} \le r$ (задача similarity join), можно получить, используя в качестве объектов-запросов для rNN все объекты базы.

Выполнение запросов поиска по сходству «линейным» (или «последовательным») поиском заключается в вычислении расстояний/сходств объекта-запроса до всех $N$ объектов базы и возвращении объектов, удовлетворяющих условиям запроса. Поэтому сложность (время) выполнения запроса пропорциональна $N$ и времени вычисления расстояния/сходства между двумя объектами. Такой поиск часто оказывается недопустимо медленным для больших баз, особенно для объектов со сложными многокомпонентными представлениями и вычислительно сложной меры расстояния/сходства.

Один подход к ускорению линейного поиска по сходству состоит в быстрой оценке величины всех расстояний/сходств [2, 3]. Другой подход заключается в построении по базе такой структуры данных (индексной структуры, ИС [4–7]), использование которой при выполнении запроса поиска по сходству позволит сократить количество вычислений сходства объекта-запроса с другими объектами по сравнению с линейным поиском (т.е. обеспечить поиск, сублинейный относительно $N$).

© Д.А. Рачковский, 2019

В обоих подходах при выполнении запроса применяют процедуры, которые можно считать вариантами двухэтапной стратегии фильтрации и уточнения F&R. На первом этапе осуществляется быстрый отбор объектов-кандидатов. Результаты первого этапа уточняются на втором этапе (обычно с использованием линейного поиска среди объектов-кандидатов по мере расстояния/сходства, заданной в запросе). Отметим, что для запроса rNN второй этап устраняет false positives (т.е. объекты-кандидаты, которые не являются ответом на запрос).

Если суммарное количество объектов-кандидатов меньше $N$ и их отбор выполняется достаточно быстро, то можно получить ускорение выполнения запроса относительно линейного поиска. Для точного поиска при выполнении запроса кандидаты выбираются так, что они гарантированно содержат ответ на запрос. Стратегию F&R иногда применяют многократно при выполнении одного запроса для различных подмножеств объектов базы и/или типов фильтров. Последовательное использование нескольких фильтров (обычно от малой сложности и малой эффективности исключения к большой) сокращает количество кандидатов.

На первом этапе фильтрации в стратегии F&R эффективное исключение (или, наоборот, отбор) кандидатов часто выполняют на основе признаков, выделенных из исходных представлений объектов. Например, для бинарных векторов признаком может являться индикатор наличия или отсутствия у некоторой части компонентов вектора определенной комбинации нулевых и единичных значений (что эквивалентно наличию некоторых элементов множества, представленных компонентами такого вектора) [3]. Для символьных строк широко используют $n$-граммы, т.е. комбинации $n$ смежных символов [1]. Для графов признаками могут являться определенные подграфы, обычно с малым количеством вершин и ребер.

Наличие у двух объектов некоторого количества совпадающих признаков связывают с величиной меры сходства между исходными представлениями этих объектов, по которой выполняется поиск по сходству. Если количество совпадающих признаков позволяет ограничить снизу величину расстояния между объектами, то при выполнении запроса нахождение объектов из базы, содержащих требуемое количество признаков, совпадающих с признаками объекта-запроса, позволяет использовать такие объекты в качестве кандидатов и исключить из дальнейшего рассмотрения объекты, у которых меньше совпадающих признаков.

Для быстрого поиска объектов базы с достаточным количеством совпавших признаков часто используют ИС на основе обратного индексирования [5, 7, 8]. В таких ИС для каждого признака запоминают объекты базы, в которых находится этот признак. При выполнении запроса обрабатывают только признаки, имеющиеся в объекте-запросе. Если количество посещенных объектов меньше $N$, то получают сублинейное время поиска (лишь эти объекты подлежат сравнению с порогом для запроса rNN или ранжированию для запроса kNN). Поэтому обратный индекс особенно эффективен для малого количества признаков в объекте и при малом количестве посещенных объектов. Однако в худшем случае сублинейное от $N$ время поиска не гарантируется.

Примерами признаков других типов являются расстояния до опорных объектов [4, 6] или гистограммы некоторых характеристик объекта. Признаки часто представляют как компоненты вектора. С векторами могут применяться ИС для быстрого поиска по соответствующим расстояниям между ними [5–7]. Если расстояние между векторами меньше расстояния между исходными объектами, то результаты запроса rNN для векторов дают объекты-кандидаты, содержащие ответ на запрос rNN для исходных объектов.

В настоящей статье рассмотрены объекты, представленные в виде графов, т.е. множества вершин, связанных ребрами. Вершины и ребра могут быть поме-

ченными. Рассмотрены также объекты-деревья, т.е. связные ациклические графы. Будем называть вершины деревьев узлами. В деревьях могут быть помеченными только узлы.

В качестве меры расстояния, по которой выполняются запросы поиска по сходству, далее рассматривается наиболее распространенная для деревьев и графов мера расстояния, известная как расстояние редактирования. Для деревьев расстояние редактирования $\mathrm{dist}_{\mathrm{TED}}$ [9] — это минимальная стоимость последовательности элементарных операций редактирования узлов, которая преобразует одно дерево в другое. Используются три типа операций редактирования: удаление, вставка, переименование. Расстояние редактирования деревьев предложено в [10] как обобщение расстояния редактирования строк $\mathrm{dist}_{\mathrm{edit}}$ [11].

Для двух графов расстояние редактирования $\mathrm{dist}_{\mathrm{GED}}$ [12], предложенное в [13], — это минимальная стоимость последовательности операций редактирования, которая преобразует первый граф во второй (или в изоморфный второму). В отличие от $\mathrm{dist}_{\mathrm{TED}}$ в $\mathrm{dist}_{\mathrm{GED}}$ три упомянутых типа операций редактирования применяются не только к вершинам, но и к ребрам.

Обзоры быстрой оценки сходства объектов по векторам, в которые они преобразованы, даны в [2, 3]. Обзор ИС для быстрого поиска по сходству объектов, для которых известны только значения расстояний до других объектов, представлен в [4]. ИС для векторов с компонентами — бинарными числами — рассмотрены в [5], а с компонентами — вещественными числами — в [6, 7]. ИС для быстрого поиска символьных строк представлены в [1].

Отметим, что для векторов и даже для строк разработаны алгоритмы и ИС, которые теоретически ускоряют линейный поиск по сходству для данных худшего случая. Некоторые из них уже реализованы на практике или потенциально могут быть реализованы. Однако для веторов/строк большой размерности все ИС с теоретическими гарантиями выполняют запросы не точного, а приближенного поиска по сходству, т.е. возвращают результаты, которые могут отличаться от результатов запросов точного поиска по сходству. Приближенный поиск по сходству востребован на практике, поскольку для многих применений достаточно получать приближенные результаты, но быстро.

В данном обзоре рассмотрены (если не оговорено иное), главным образом, реализованные на практике алгоритмы для выполнения запросов точного поиска по сходству по $\mathrm{dist}_{\mathrm{TED}}$ (разд. 1) и $\mathrm{dist}_{\mathrm{GED}}$ (разд. 2), но без гарантий ускорения линейного поиска для данных худшего случая. Рассмотрены также алгоритмы для ускорения вычисления $\mathrm{dist}_{\mathrm{GED}}$ (подразд. 3.1) и его границ (подразд. 3.2).

## 1. ИНДЕКСНЫЕ СТРУКТУРЫ ДЛЯ ПОИСКА СХОДНЫХ ДЕРЕВЬЕВ ПО РАССТОЯНИЮ РЕДАКТИРОВАНИЯ

Для неупорядоченных помеченных деревьев (даже для бинарных деревьев, т.е. с коэффициентом ветвления 2 и с размером 2 алфавита меток) сложность вычисления $\mathrm{dist}_{\mathrm{TED}}$ является NP-полной [9] (в действительности, эта задача MAX SNP-сложная [14]). Однако при поиске по сходству деревьев обычно работают с упорядоченными помеченными деревьями, для которых вычислительная сложность $\mathrm{dist}_{\mathrm{TED}}$ намного ниже [15, 16] и составляет $O(n^3)$, где $n$ — количество узлов. Даже такая сложность очень велика, поэтому поиск по сходству ускоряют, используя стратегию F&R.

Очевидной нижней границей $\mathrm{dist}_{\mathrm{TED}}$, но очень грубой [17], является разность высот двух деревьев. Легко вычислимы и более плотны нижние границы $\mathrm{dist}_{\mathrm{TED}}$ на основе расстояний между некоторыми гистограммами, которые формируются по дереву. (Манхэттенское расстояние $\mathrm{dist}_{\mathrm{Man}}$ между гистограмма-

ми-векторами $\mathbf{a}, \mathbf{b}$ размерности $D$ с компонентами $a_i$, $b_i$ вычисляется как $||\mathbf{a}-\mathbf{b}||_1 = \sum_{i=1}^{D} |a_i - b_i|$.)

Так, в Histogram [17] строят гистограмму расстояний узлов до листьев (расстояние до листового узла определяется как максимальная длина пути из узла в его листья), ее размерность на единицу превышает высоту дерева. Нижней границей $\text{dist}_{\text{TED}}$ являются: расстояние $\text{dist}_{\text{Man}}$ между этими гистограммами $\mathbf{a}$, $\mathbf{b}$ для двух деревьев, а также $\text{dist}_{\text{Man}}/3$ между гистограммами степеней узлов и $\text{dist}_{\text{Man}}/2$ между гистограммами меток узлов [17]. Отметим, что эти фильтры не учитывают порядка узлов. Для поиска использовалась древовидная ИС X-tree из [18].

В BRV/BBD [19] используют гистограмму признаков — бинарных ветвей. Вначале исходное дерево трансформируют в бинарное дерево, оставляя самый левый и самый правый узлы-сыновья каждого узла. Если один из сыновей отсутствует, вместо него добавляют «пустой» узел. Из бинарного дерева извлекают бинарные ветви (узел и два его сына) и формируют вектор, каждый компонент которого является количеством ветвей конкретной конфигурации. Расстояние $\text{dist}_{\text{Man}}/5$ между гистограммами — нижняя граница $\text{dist}_{\text{TED}}$. Учитывается и положение ветвей аналогично подходам к фильтрации данных типа строк [1]. Ускорения фильтрации при выполнении запросов достигают обратным индексированием. Для векторов с неотрицательными компонентами $\text{dist}_{\text{Man}}(\mathbf{a}, \mathbf{b}) = ||\mathbf{a}||_1 + ||\mathbf{b}||_1 - 2\sum_{i=1}^{D} \min(a_i, b_i)$; аналогично выполняется вычисление симметричной разности мультимножеств. Поэтому с помощью обратного индекса достаточно найти $\min(a_i, b_i)$ для компонентов, которые не равны нулю в обоих векторах (гистограммах).

В алгоритме Bounds [20] деревья преобразуют в строки обходом в прямом порядке (узел-родитель посещают до его сыновей) или в обратном порядке (узлы-сыновья посещают до их родительского узла). Для полученных строк выполняется неравенство $\text{dist}_{\text{edit}} \le \text{dist}_{\text{TED}}$ [20]. Сложность вычисления $\text{dist}_{\text{edit}}$ методом динамического программирования составляет $O(n^2)$, т.е. меньше, чем для деревьев. Кроме того, существуют ИС для быстрого поиска по $\text{dist}_{\text{edit}}$ [1]. Менее плотную нижнюю границу $\text{dist}_{\text{TED}}$ дает $\text{dist}_{\text{edit}}$ между строками, которые получают обходом Эйлера [21] (при этом каждое ребро посещается только один раз).

В алгоритме ReferenceSets [20] используют преобразование деревьев в векторы на основе вычисления $\text{dist}_{\text{TED}}$ до опорных объектов, которое сжимает расстояния. Полученные границы расстояний $\text{dist}_{\text{TED}}$ комбинируют с Bounds. Для поиска по векторам применяют R-tree (см. [7]).

В ИС PartSJ [22] используют разбиение дерева на подграфы с учетом величины радиуса запроса $r$. Каждое дерево представляют как бинарное в виде: самый левый сын + правый брат (следующий узел того же уровня, что и сын). Разбиение дерева на $2r+1$ непересекающийся подграф выполняется удалением ребер. Сходные деревья с $\text{dist}_{\text{TED}} \le r$ должны иметь хотя бы один общий подграф. Разбиение на подграфы сбалансированного размера достигается максимизацией минимального размера подграфов.

Ускорение поиска общих подграфов получено конструированием отдельного обратного индекса для деревьев каждого размера. Подграфы делят на группы на основе меток узлов и положения подграфов в дереве (в обратном порядке обхода узлов). Обратный индекс имеет двухуровневую структуру. На первом уровне подграфы относят к группам с учетом положения, на втором — каждую группу разбивают с использованием меток. При выполнении запроса исключают группы, которые не могут содержать деревьев-кандидатов.

## 2. ИНДЕКСНЫЕ СТРУКТУРЫ ДЛЯ ПОИСКА СХОДНЫХ ГРАФОВ ПО РАССТОЯНИЮ РЕДАКТИРОВАНИЯ

В общем случае (т.е. для графов, которые не принадлежат к некоторым узким классам) задача вычисления $\text{dist}_{\text{GED}}$ является NP-сложной [23]. Сложность всех известных алгоритмов вычисления $\text{dist}_{\text{GED}}$ экспоненциальная от количества вершин. Поэтому для быстрого поиска по $\text{dist}_{\text{GED}}$ особенно актуально применение стратегии фильтрации и уточнения F&R на основе быстрого определения границ $\text{dist}_{\text{GED}}$.

При фильтрации используют нижние границы $\text{dist}_{\text{GED}}$, которые определяются, например, по совпадению некоторого количества признаков, извлеченных из графов. Для ассоциирования признаков с графами и быстрого определения графов-кандидатов, прошедших фильтрацию, часто применяют обратное индексирование. Признаки используют как глобальные (например, на основе количества вершин, ребер, их меток (подразд. 2.1.1)), так и локальные (на основе подструктур графов, которые пересекаются (подразд. 2.1.2) или не пересекаются (подразд. 2.1.3)).

Этап уточнения кандидатов, прошедших фильтр, для поиска графов является особенно вычислительно сложным ввиду сложности вычисления $\text{dist}_{\text{GED}}$. Для сокращения списка кандидатов применяют многоступенчатую фильтрацию с использованием различных нижних границ на $\text{dist}_{\text{GED}}$. Фильтрация также востребована на этапе точного вычисления $\text{dist}_{\text{GED}}$ и для проверки, не превышает ли расстояние порогового значения (разд. 3).

**2.1. Фильтрация графов с использованием признаков.** На этапе фильтрации кандидатов используют извлечение признаков, которые позволяют эффективно исключать графы базы, не отвечающие запросу. Некоторые глобальные признаки (подразд. 2.1.1) легко вычислимы, но недостаточно селективны, т.е. позволяют исключать из множества графов-кандидатов только малую часть графов базы. В качестве признаков также применяют подструктуры графа (деревья, пути), которые имеют фиксированный размер и могут пересекаться (подразд. 2.1.2). Более эффективны признаки без пересечений — подграфы нефиксированного размера (подразд. 2.1.3).

**2.1.1. Фильтрация на основе глобальных признаков графов.** Фильтрация на основе такой глобальной информации, как различия в количестве вершин/ребер [23], основана на нижней границе $\text{dist}_{\text{GED}}(a, b) \geq ||V_a| - |V_b|| + ||E_a| - |E_b||$, где $|V_a|, |V_b|$ и $|E_a|, |E_b|$ — соответственно количество вершин и ребер в графах $a$, $b$. Фильтрация по различию в количестве меток вершин/ребер [24] использует нижнюю границу $\text{dist}_{\text{GED}}(a, b) \geq \Gamma(L_{Va}, L_{Vb}) + \Gamma(L_{Ea}, L_{Eb})$, где $\Gamma(A, B) = \max\{|A|, |B|\} - |A \cap B|$, а $L_{Va}, L_{Ea}$ — мультимножество меток вершин и ребер графа $a$ соответственно. Так как структура графов в этих признаках не используется, такие нижние границы $\text{dist}_{\text{GED}}$ не являются плотными.

Более плотную нижнюю границу $\text{dist}_{\text{GED}}$ на основе глобальных признаков в [25] получают как $2(|V_a| - |V_b|) + ||E_a| - (|E_b| + |V_a| - |V_b|)|$ и как $2(|V_a| - |V_b|) + ||E_a| - (|E_b| + |V_a| - |V_b|)| + (|L_{Va}| - (|L_{Va} \cap L_{Vb}| + |V_a| - |V_b|))$ при $|V_a| > |V_b|$. Учет степеней вершин, вставленных при редактировании графа, позволяет усилить эту нижнюю границу [25].

**2.1.2. Фильтрация на основе признаков в виде пересекающихся частей графов.** Фильтрация на основе нижних границ $\text{dist}_{\text{GED}}$, определяемых по пересекающимся локальным признакам, извлеченным из графов, аналогична подходу для фильтрации строк [1]. Количество одинаковых признаков у двух графов с $\text{dist}_{\text{GED}} \leq r$ составляет не менее $\max\{|A| - rn_{\text{aff}}(V_b), |B| - rn_{\text{aff}}(B)\}$, где $|A|$ — мощность мультимножества признаков графа $a$, $n_{\text{aff}}(A)$ — максимальное количество признаков, модифицируемых одной операцией редактирования.

В ИС к-AT [26] в качестве признаков используют деревья к-AT. Такое дерево для каждой вершины графа состоит из этой вершины и всех других вершин, в которые можно перейти за к шагов обхода графа breadth-first, вершины-сыновья отсортированы по их меткам в графе. Количество совпадающих деревьев к-AT у двух графов с расстоянием $\mathrm{dist}_{\mathrm{GED}}$ составляет не менее $|V_x| - rn_{\mathrm{aff}}$, где $n_{\mathrm{aff}} = 2(\deg(x)-1)^{\kappa-1}$ — максимальное количество деревьев к-AT, модифицируемых одной операцией редактирования, $|V_x|$ — количество вершин графа-запроса $x$, $\deg(x)$ — максимальная степень его вершины. Это дает нижнюю границу $\mathrm{dist}_{\mathrm{GED}}$. Иная нижняя граница для этого случая приведена в [24].

Так как операции редактирования сильно влияют на общие деревья, такая нижняя граница не является плотной и эффективна только для разреженных графов. Выбор к также ограничен малым диапазоном, что приводит к малой селективности и большому количеству кандидатов.

В алгоритме C-star [23] в качестве признаков применяются «звезды», т.е. деревья к-AT с $\kappa = 1$. Предложена нижняя граница $\mathrm{dist}_{\mathrm{GED}}$, вычисляемая как $\mathrm{dist}_{\mathrm{map}}(a, b) / \max\{4, \max\{\deg(a), \deg(b)\}+1\}$, где $\mathrm{dist}_{\mathrm{map}}(a, b)$ — «расстояние отображения» между мультимножествами звезд двух графов (подробнее см. в [23]).

Вычисление величины $\mathrm{dist}_{\mathrm{map}}$ требует решения linear sum assignment problem with error-correction (LSAPE) [27]. Эта задача является расширением линейной задачи о назначениях (LSAP): найти взаимно-однозначное соответствие (биекцию) между узлами двудольного графа, которое минимизирует сумму весов ребер. Задача LSAP решается венгерским алгоритмом (алгоритм Манкреса) [28, 29], сложность которого кубическая от суммарного количества вершин $O(|V|^3)$.

Для применения решения LSAP к решению LSAPE предложены различные подходы [27]. В C-star для вычисления $\mathrm{dist}_{\mathrm{map}}$ звезды двух графов представляют узлами двудольного графа, вес его ребра есть расстояние редактирования между двумя звездами. Чтобы сделать одинаковым количество узлов, добавляют узлы со спецметкой.

Для ускорения работы C-star в ИС SEGOS [30] используют двухуровневое обратное индексирование. Нижний уровень ИС индексирует и позволяет находить признаки (звезды), сходные с признаками графа-запроса. Верхний уровень индексирует графы базы по их признакам (звездам) и при выполнении запроса использует результаты нижнего уровня для получения списка графов базы в порядке убывания сходства с графом-запросом. Недостатком SEGOS является необходимость настройки скрытых параметров [31].

Более плотную, чем признаки-деревья, нижнюю границу $\mathrm{dist}_{\mathrm{GED}}$ дают признаки-пути на графе [31] некоторой длины $n$. Для путей значение $n_{\mathrm{aff}}(x) = \max_{u \in V(x)} |\mathrm{path}(x, u)|$, где $|\mathrm{path}(x, u)|$ — мощность мультимножества путей графа $x$, содержащих вершину $u$. Количество вершин в пути растет линейно от $n$, а в дереве — экспоненциально от к (т.е. от $n$). Поэтому $n_{\mathrm{aff}}$ для путей значительно меньше и нижняя граница плотнее, т.е. позволяет получать более эффективное исключение.

ИС GSimSearch на основе путей [31] использует для повышения эффективности префиксную фильтрацию, аналогичную префиксной фильтрации для множеств и строк [1], в которой требуется совпадение хотя бы одного пути в префиксах длиной $rn_{\mathrm{aff}}(x)+1$. Учет несовпадающих признаков (путей) позволяет уменьшить длину префикса аналогично строкам [1], но более сложно, так как признаки в графах не имеют «начального положения», а также ввиду NP-сложного вычисления «minimum edit operation problem». Для кандидатов применяется глобальная фильтрация по размеру, т.е. разности количества вершин и ребер (см. подразд. 2.1.1).

На этапе уточнения кандидатов используют фильтрацию по количеству совпадающих признаков в полных графах (т.е. не только в префиксах), а также глобальную фильтрацию на основе меток (см. подразд. 2.1.1) и ее локальный вариант. Для оставшихся кандидатов применяют NP-сложное вычисление $\mathrm{dist}_{\mathrm{GED}}$ с помощью алгоритма A* ([32] и подразд. 3.1), усиленного фильтрами на основе несовпадающих признаков. В экспериментах GSimSearch превосходит к-AT и SEGOS.

Признак «ветвь» из ИС Mixed Filter [33] — «сокращенная» звезда с одной вершиной и ребрами (т.е. без вершин, которыми оканчиваются ребра). Это позволяет сокращать $n_{\mathrm{aff}}$ с $\deg(x)$ для звезд до двух для ветвей и получать ряд более плотных границ. Аналогично нижней границе на основе звезд временная сложность вычисления самой плотной границы для ветвей составляет $O(|V|^3)$. Для полученной в [33] менее плотной границы $\mathrm{dist}_{\mathrm{GED}}$ время вычисления удалось сократить до $O(|V|\log|V|)$.

В [33] предложена нижняя граница на основе разбиения графа на непересекающиеся части (подграфы), которые не являются изоморфными другому графу (подразд. 2.1.3), а также ее гибрид с нижней границей на основе ветвей. Кроме того, используется фильтрация на основе нижних границ, полученных для непересекающихся частей, и ряд других границ. Все эти границы сочетают в ИС, которая является аналогом R-tree [7].

Отметим, что расстояние между графами, вычисляемое на основе ветвей, применяют в [34] для быстрой $O(|V|\deg)$ аппроксимации $\mathrm{dist}_{\mathrm{GED}}$, которая, однако, не является нижней границей, т.е. не гарантирует точного поиска (подразд. 3.2).

**2.1.3. Фильтрация на основе признаков в виде непересекающихся частей графов.** Описанные в подразд. 2.1.2 признаки фиксированного размера имеют пересечения — различные признаки включают одни и те же вершины и/или ребра исходного графа. Их недостатком является неполный учет глобальной структуры графов и распределений реальных данных. Фиксированный размер ограничивает селективность признаков, так как не адаптируется к базе и запросам. Кроме того, вследствие пересечения признаки избыточны, что делает условия фильтрации (ориентированные на худший случай) слабыми для вершин с большой степенью и для больших значений $r$ порогов на $\mathrm{dist}_{\mathrm{GED}}$.

Для преодоления этих недостатков разработаны подходы разбиения графов на непересекающиеся части, т.е. на части без общих вершин, по аналогии с разбиением на непересекающиеся части, применяющимся при поиске по сходству бинарных векторов [5] и строк [1]. Если можно найти непересекающиеся части, не совпадающие в двух графах (неизоморфные подграфы), количество которых не меньше $r+1$, то нижняя граница $\mathrm{dist}_{\mathrm{GED}}$ превышает $r$ [33] (см. принцип pigeonhole [35, 5, 1]). Так как части не пересекаются, одна операция редактирования может модифицировать только одну часть. Поэтому не менее $r+1>r$ операций потребуется для преобразования одного графа в другой, т.е. условие запроса rNN не выполняется.

Проблема заключается в том, чтобы определить, существует ли в двух графах достаточное количество непересекающихся и несовпадающих частей. Эта задача, по крайней мере, NP-полная. Эвристическое решение MixedFilter [33] состоит в разбиении графа-запроса на малые части размера 1, 2 и 3 (например, части размера 3 могут состоять из двух вершин и одного ребра или из одной вершины и двух ребер). Затем осуществляется поиск (с помощью обратного индекса всех малых частей, построенного по графам базы) и удаление из графа-запроса малых частей, не совпавших с графом базы. Для оставшегося подграфа графа-запроса поиск несовпавших частей размера, бóльшего 3, выполняется с примене-

нием алгоритма проверки изоморфизма подграфа. Как только найдена $r+1$ несовпавшая часть, граф базы можно исключить.

ИС Pars [36] разбивает графы на непересекающиеся подграфы с возможностью существования полуребер, т.е. ребер, в которых одна вершина определена, а другая не определена (представлена как спецсимвол). Любая операция редактирования графа модифицирует только один такой подграф с полуребрами. Используется принцип pigeonhole: если разбиение графа базы выполнено на $r+1$ непересекающуюся часть, то одна из частей (изоморфный подграф с полуребрами) должна содержаться в графе-запросе.

Разбиение графов базы выполняют с учетом минимизации среднего количества кандидатов, возвращаемых запросом. Для этого применяется эвристический алгоритм: случайное разбиение, которое затем уточняется с помощью стоимостной модели, построенной по объектам-запросам, случайно выбранным из базы. Получают подграфы с различными размерами.

По полученным частям графов строят обратный индекс базы. При выполнении запроса для каждой части (подграфа) индекса проверяется, содержится ли она в графе-запросе (проверка на изоморфизм подграфа). Затем для графов базы с совпавшей частью выполняется глобальная фильтрация по размеру и меткам (см. подразд. 2.1.1). Остальные графы базы являются кандидатами, до которых вычисляется $\mathrm{dist}_{\mathrm{GED}}$ графа-запроса.

Если описанное офф-лайн разбиение графов базы плохо соответствует характеристикам запросов, то в ходе выполнения запроса изменяют разбиение графов базы, прошедших предшествующие этапы фильтрации, чтобы исключить бóльшее количество кандидатов без вычисления $\mathrm{dist}_{\mathrm{GED}}$. На этапе вычисления $\mathrm{dist}_{\mathrm{GED}}$ для ускорения используют найденные на этапе фильтрации части, содержащиеся в графе-запросе. Все это позволяет Pars работать с бóльшими значениями $r$ и выполнять запросы быстрее, чем MixedFilter, GSimSearch, SEGOS.

Для запросов kNN конструируют специальную ИС. Каждый граф разбивают на количество частей от 1 до $r_{\max}+1$, каждое разбиение запоминают на своем уровне иерархии. Вначале создают разбиение самого нижнего уровня $r_{\max}+1$, из него создают вышележащие уровни. Часть на уровне $i$ состоит из двух частей уровня $i+1$, конкретные две части выбирают по стоимостной модели. Затем поуровнево выполняют объединение иерархий графов базы, т.е. по их подграфам строят обратные индексы на каждом уровне.

При выполнении запроса kNN вначале рассматривают «перспективные» графы базы, имеющие много общих частей с графом-запросом. По ним определяют верхнюю границу расстояния до $k$-го ближайшего соседа $\mathrm{dist}_{\mathrm{GED}}(k)$, которую можно использовать для исключения «неподходящих» графов. Для этого, начиная с верхнего уровня ИС $i=1$, ищут графы с совпадающими частями и объединяют их во множества, содержащие графы с одинаковым суммарным количеством $j$ вершин и ребер во всех совпавших частях.

Для некоторого уровня $i$, если $i \geq \mathrm{dist}_{\mathrm{GED}}(k)$, поиск прерывают и возвращают текущие $k$ ближайших соседей. В противном случае, начиная с множества с максимальным $j$, проверяют выполнение $|V(x)|+|V(y)|-j \geq \mathrm{dist}_{\mathrm{GED}}(k)$. Если данное условие выполняется, то графы множества не могут иметь расстояние редактирования до графа-запроса, меньшее $\mathrm{dist}_{\mathrm{GED}}(k)$, и можно перейти к следующему уровню $i$. При невыполнении данного условия для каждого графа множества (в порядке уменьшения количества совпадающих частей) вычисляют $\mathrm{dist}_{\mathrm{GED}}$ до графа-запроса и помещают результат в приоритетную очередь. Затем значение $j$ уменьшают. Для ускорения поиска также используют модификации этой ИС.

Комбинация с pigeonring [35, подразд. “Pars”] ускоряет Pars до 1.04 и 1.9 раз [35]; возможно, это самая быстрая ИС для $\mathrm{dist}_{\mathrm{GED}}$ графов в настоящее время.

ML-Index [37] для повышения плотности нижней границы $\text{dist}_{\text{GED}}$ разбивает граф на $r+m$ частей (подграфов с полуребрами). Это разбиение исчерпывающее, а части непересекающиеся. Чтобы выполнялось $\text{dist}_{\text{GED}} \le r$, должны быть $m$ частей (подграфов), которые изоморфны (любым) подграфам второго графа. С увеличением $m$ растет плотность соответствующих нижних границ $\text{dist}_{\text{GED}}$ и эффективность исключения false positives. Для создания разбиений, обеспечивающих более высокую эффективность исключения, чем Pars, предложена эвристика, в которой предпочтение получают разбиения с бóльшим количеством вершин и ребер и с меньшей частотой встречаемости их меток [37].

Все графы базы разбивают на $r+m$ подграфов с полуребрами. Для каждого подграфа строится обратный список. Это позволяет быстро определить по подграфу запроса, какие графы базы имеют такой же подграф, а также графы базы с $m$ и более совпадающими подграфами, которые являются кандидатами.

Проверка изоморфизма подграфа с полуребрами — трудоемкая задача. Чтобы исключить false positives, используется «профиль» подграфа, который включает частоты меток вершин и ребер. Для каждой метки частота в подграфе не должна превышать частоты в графе-запросе. Эффективность исключения повышает использование $L$ различных разбиений на $r+m_i$ частей. Создаются соответствующие обратные индексы и профили, вместе они образуют ML-Index. При выполнении запроса каждый из $L$ подиндексов дает свое подмножество кандидатов, итоговое подмножество получают как их пересечение (а не объединение, как в ИС LSH [6]). В экспериментах [37] ML-Index превзошел Pars (хотя этот результат не подтвержден).

Отметим работу [38] о выполнении запроса kNN, результаты которой не сравнивались с уже упомянутыми ИС.

## 3. БЫСТРОЕ ВЫЧИСЛЕНИЕ РАССТОЯНИЯ РЕДАКТИРОВАНИЯ ГРАФОВ И ЕГО ГРАНИЦ

При поиске по сходству на этапе уточнения стратегии F&R требуется вычисление $\text{dist}_{\text{GED}}$ или определения точных результатов сравнения с порогом $r$. Как упоминалось в разд. 2, вычисление $\text{dist}_{\text{GED}}$ является NP-сложным и соответствующие алгоритмы требуют времени, экспоненциального от суммарного количества вершин в графах. Поэтому всячески избегают вычисления большого количества расстояний $\text{dist}_{\text{GED}}$ за счет применения многоступенчатой фильтрации кандидатов. Также используют различные эвристики, позволяющие хотя бы немного уменьшить время вычисления.

Отметим, что ранее в настоящей статье рассматривалось «однородное» расстояние редактирования (с одинаковыми стоимостями различных типов операций редактирования). В некоторых случаях более адекватным задаче может быть «неоднородное» расстояние редактирования $\text{dist}_{\text{GED}}$ с неодинаковыми стоимостями операций редактирования. Например, в случае, если метки представлены векторами и стоимостью замены может являться расстояние между векторами [39, 40]. Многие из рассмотренных далее алгоритмов позволяют работать и с такими расстояниями.

**3.1. Точное вычисление расстояния редактирования графов.** Для точного вычисления $\text{dist}_{\text{GED}}$ предложены алгоритмы различных типов. Некоторые из них согласно [41] относятся к типу branch and bound B&B [4].

Первым был разработан алгоритм A*-GED [32, 42, 43]. В нем отображение вершин одного графа на вершины другого (это отображение также задает отображение ребер), при котором стоимость редактирования минимальна, выполняется следующим образом.

В процессе вычислений строят дерево, итеративно создавая его узлы. Уровень дерева соответствует одной вершине первого графа (вершины упорядочены

венгерским алгоритмом по возрастанию перспективности). В узлах первого уровня дерева с первой вершиной первого графа комбинируют каждую вершину второго графа, а также вершину, соответствующую операции удаления. На следующем уровне дерева сыновьи узлы получают добавлением к отображению, соответствующему отцовскому узлу, всех отображений вершины первого графа, соответствующей уровню, и неотображенных вершин второго графа. Таким образом, промежуточные узлы дерева соответствуют частичным отображениям, а листовые — полным.

В A*-GED выполняется обход дерева best-first. На каждой итерации A*-GED выбирает для расширения лучшее частичное отображение. Таким отображением является узел, для которого минимальна величина cost, равная сумме стоимости редактирования уже отображенных вершин графа (и задаваемых этим отображением операций редактирования ребер) и нижней границы стоимости редактирования оставшейся части. Для определения нижней границы могут применяться различные эвристики, в частности, решение LSAPE (см. подразд. 2.1.2). Процедура останавливается по достижении первого листа, так как он соответствует искомому значению $\mathrm{dist}_{\mathrm{GED}}$. Затраты памяти этого алгоритма высоки вследствие необходимости хранения большого количества частичных отображений (промежуточных состояний поиска), так как большинство из них нельзя исключить до самых поздних этапов поиска ввиду возможности их расширения. Кроме того, велика сложность определения частичного отображения с минимальной стоимостью, которое подлежит расширению, а также модификации стоимости для каждого возможного расширения отображения.

Значительно уменьшить затраты памяти и несколько уменьшить время поиска позволяет алгоритм DF-GED [44] на основе обхода дерева depth-first search [4, 7].

В процессе поиска поддерживается верхняя граница $\mathrm{dist}_{\mathrm{GED}}$, которая вначале устанавливается эвристикой LSAPE ([43] и см. подразд. 2.1.2) и впоследствии заменяется реальным значением стоимости редактирования при посещении листа (полный путь редактирования), если это значение меньше текущей верхней границы. Отметим, что при выполнении запроса rNN вычисление точного $\mathrm{dist}_{\mathrm{GED}}$ можно останавливать, как только верхняя граница $\mathrm{dist}_{\mathrm{GED}}$ (при посещении листового узла) становится меньше $r$.

Узлы-сыновья текущего узла сортируют в порядке возрастания cost и заносят в стек те из них, для которых cost меньше текущей верхней границы.

Из стека извлекают верхний узел и выполняют переход на него, если его cost меньше текущей верхней границы. В противном случае из стека извлекают следующий узел и т.д. Это обеспечивает выполнение «обратного прослеживания» (backtracking) [4, 7]. Обход прекращают, когда стек опустошается. При этом текущая верхняя граница дает $\mathrm{dist}_{\mathrm{GED}}$. В этом алгоритме даже в худшем случае в стеке находится небольшое количество частичных отображений — порядка $|V_1||V_2|$.

Параллельный алгоритм depth-first [45] ускоряет алгоритм DF-GED [44].

Ускорение DF-GED для случая одинаковой стоимости операций редактирования достигается в [40] за счет более быстрого вычисления нижней границы $\mathrm{dist}_{\mathrm{GED}}$ (вместо кубической сложности венгерского алгоритма получают $\Theta(\max\{|V_1|,|V_2|\}+\max\{|E_1|,|E_2|\})$.

В CSI-GED [46] также используют обход depth-first, но работают не с отображением вершин графов, а с отображением их ребер и с однородным $\mathrm{dist}_{\mathrm{GED}}$. Расширение CSI-GED на неоднородное $\mathrm{dist}_{\mathrm{GED}}$ предложено в [40]. Вследствие экспоненциальной зависимости времени поиска от количества ребер CSI-GED хорошо работает в основном на разреженных и несходных графах.

Алгоритмы на основе обхода depth-first могут попадать в локальные минимумы (находить субоптимальное решение), преодоление чего требует вычисли-

тельно сложного обратного прослеживания. Кроме того, существует большое количество избыточных отображений (с одинаковой стоимостью редактирования) и неоптимальных отображений.

Для решения этих проблем BSS-GED [47] предлагает алгоритм генерации узлов-потомков в дереве поиска, который значительно уменьшает количество избыточных и неоптимальных отображений. Для обхода применяют beam-stack search [48], который позволяет получать гибкий компромисс между затратами памяти и затратами времени на обратное прослеживание и работает эффективнее, чем обход best-first и depth-first.

Согласно [48] алгоритм BSS-GED для графов с количеством вершин от 12 до (максимального) 21 в 2–95 раз быстрее алгоритма CSI-GED, который в свою очередь быстрее DF-GED и A*-GED. Для малых графов с количеством вершин, меньшим девяти, CSI-GED может быть быстрее BSS-GED.

Отметим, что в задаче поиска по сходству линейный поиск по CSI-GED и BSS-GED быстрее GSimJoin [24] (аналог GSimSearch, см. подразд. 2.1.2), благодаря применению более плотных (и более эффективных) нижних и верхних границ. Для $r \geq 8$ на используемых базах GSimJoin не работал ввиду больших затратах памяти. При «рабочих» $r$ преимущество BSS-GED над GSimJoin составляло 1.6–15000 раз. Алгоритм BSS-GED несколько медленнее CSI-GED при малых $r$, но при $r \geq 6$ быстрее, и преимущество увеличивается с ростом $r$ (максимальные значения $r$ составляли 12–15) от 1.2 до 10000 раз [48].

В [49] предложено обобщение подходов к вычислению $\mathrm{dist}_{\mathrm{GED}}$ на основе обходов best-first и depth-first. Применительно к best-first в AStar+ (модификация A*-GED) частичное отображение хранится в памяти фиксированного размера, для сокращения приоритетной очереди используется верхняя граница $\mathrm{dist}_{\mathrm{GED}}$, пространство поиска значительно уменьшается за счет предложенной плотной нижней границы $\mathrm{dist}_{\mathrm{GED}}$. В результате затраты памяти AStar+ значительно уменьшаются по сравнению с A*-GED, а время поиска уменьшается по сравнению с модификацией DF-GED. Алгоритм AStar+ более чем на четыре порядка [49] быстрее CSI-GED и DF-GED.

Другой тип алгоритмов вычисления точного $\mathrm{dist}_{\mathrm{GED}}$ — это алгоритмы на основе целочисленного линейного программирования [50], когда ищется матрица перестановок, минимизирующая стоимость преобразования одного графа в другой. Алгоритм F2 (BIP-GED) на основе бинарного линейного программирования с $\Theta(|E_1||E_2|)$ переменными и $\Theta(|E_1||V_2|)$ ограничениями [51] лишен недостатков [50], связанных с рассмотрением только невзвешенных графов и без меток на ребрах. В [41] параллельный вариант F2 показал лучшие результаты среди точных алгоритмов. Отметим, что в этом экспериментальном исследовании вычислялись также приближенные результаты (при ограниченном времени счета) или дающие границы точного $\mathrm{dist}_{\mathrm{GED}}$.

Алгоритм F3 [52] на основе смешанного целочисленного линейного программирования MILP улучшает F2 за счет использования набора ограничений, которое не зависит от количества ребер в графах. Это дает ускорение для графов с большой связностью. Алгоритм MIP-GED на основе MILP предложен в [40] и представляет интерес (главным образом, теоретический) в связи с малым количеством переменных и ограничений $\Theta(|V_1||V_2|)$.

В [40] также проведено ускорение различных существующих алгоритмов и их сравнительное тестирование. Для графов с количеством вершин до 10 при однородном $\mathrm{dist}_{\mathrm{GED}}$ ускоренный DF-GED быстрее BIP-GED, CSI-GED, MIP-GED, A*-GED. При неоднородном $\mathrm{dist}_{\mathrm{GED}}$ лидирует модифицированный CSI-GED. Для графов с количеством вершин от 10 до 24 лучшим является BIP-GED. Графы

с бóльшим количеством вершин не исследовались ввиду превышения установленного лимита времени 1000 с.

В [47] отмечается, что при объеме ОЗУ 24 Гб A*GED не работает с графами, в которых более 12 вершин. Результаты [40] показали, что для графов с более чем 16 вершинами время работы исследованных алгоритмов может превышать десятки секунд. В экспериментальных исследованиях [49] для сравнения различных алгоритмов использовали графы с количеством вершин до 30, так как для большего количества существующие алгоритмы не работали. Отметим, однако, что некоторые варианты AStar+ на компьютере с ОЗУ 16 Гб работали с количеством вершин до 60 и даже до 1024 (при времени до 1000 с).

Таким образом, несмотря на интенсивные исследования, практические алгоритмы вычисления точного расстояния редактирования между графами не могут обеспечить такого вычисления за разумное время или с разумными затратами памяти. Поэтому перспективным направлением ускорения вычисления $\mathrm{dist}_{\mathrm{GED}}$ является использование более сложных и плотных, но быстрых нижних и верхних границ. Плотные границы также ускоряют выполнение запросов поиска по $\mathrm{dist}_{\mathrm{GED}}$.

**3.2. Быстрое вычисление границ расстояния редактирования графов.** Быстрое вычисление нижних границ $\mathrm{dist}_{\mathrm{GED}}$ полезно для: быстрой фильтрации кандидатов в стратегии F&R при выполнении запросов точного поиска по сходству; быстрого выполнения запросов приближенного поиска по сходству (по полученной границе вместо $\mathrm{dist}_{\mathrm{GED}}$); ускорения вычисления точного значения $\mathrm{dist}_{\mathrm{GED}}$ (для каждого неисключенного частичного пути редактирования вычисляется нижняя граница стоимости его продления до полного пути). Верхняя граница может применяться для быстрого включения объектов в результат поиска (например, при выполнении запроса rNN) или для дальнейшего уточнения $\mathrm{dist}_{\mathrm{GED}}$ (например, при вычислении расстояния).

Границы $\mathrm{dist}_{\mathrm{GED}}$ через количество совпадений в графах признаков различного типа (подструктур графов) рассмотрены в [26, 24, 33, 36, 37] и подразд. 2.1.

Эвристики для быстрого определения верхних границ $\mathrm{dist}_{\mathrm{GED}}$ на основе решения LSAPE (см. подразд. 2.1.2) предложены в [23, 42, 43]. В общем случае графы разбивают на локальные подструктуры вокруг вершин. Определяется мера расстояния между подструктурами. Она используется для формирования требующейся в задаче LSAPE матрицы, в которой строки и столбцы соответствуют вершинам графов. Затем решается задача LSAPE. Вычисленное решение интерпретируется как последовательность операций редактирования, и его стоимость является верхней границей $\mathrm{dist}_{\mathrm{GED}}$. В [42, 43] задачу назначения называют bipartite graph matching.

В C-star [23] в качестве подструктур применялись звезды (см. подразд. 2.1.2) и аппроксимировался однородный вариант $\mathrm{dist}_{\mathrm{GED}}$. Определялась как верхняя, так и нижняя границы $\mathrm{dist}_{\mathrm{GED}}$. Верхние границы $\mathrm{dist}_{\mathrm{GED}}$ на основе ветвей (т.е. «сокращенных» звезд с ребрами без вершин, которыми оканчиваются ребра, см. подразд. 2.1.2) предложены в алгоритме BP [42, 43]. Вычислительная сложность составляет $O(n^2\Delta^3 + n^3)$, где $\Delta$ — максимум из максимальных степеней двух графов. Ветви также используют в [33] (см. подразд. 2.1.2).

Меры расстояния между ветвями позволяют получать не только верхние, но и нижние границы $\mathrm{dist}_{\mathrm{GED}}$, а также работать с неоднородными стоимостями операций редактирования.

Branch [39] расширяет верхние границы на основе ветвей алгоритма BP на нижние границы различной плотности для неоднородного $\mathrm{dist}_{\mathrm{GED}}$. Их вычислительная сложность приблизительно $O(n^2\Delta^3 + n^3)$. В экспериментах границы

Branch показали лучшее соотношение время/точность по сравнению с известными нижними границами [50], а также лучшие результаты фильтрации в поиске по сходству при последовательном применении границ.

Границы для неоднородного $\mathrm{dist}_{\mathrm{GED}}$, очевидно, можно использовать как границы для однородного $\mathrm{dist}_{\mathrm{GED}}$. Граница BranchTight [39], которая итеративно улучшает нижнюю границу Branch (замедляя вычисления), оказалась самой плотной из исследованных в [39] границ для однородного $\mathrm{dist}_{\mathrm{GED}}$ и вычисляется быстрее других. В задаче фильтрации последовательность границ типа Bound также имела преимущество.

Использование подструктур с бóльшим учетом топологической информации, чем ветви, дает более плотные верхние границы за счет более крупных локальных структур, но увеличивает затраты времени.

В Subgraph [53] в качестве подструктур применяют локальные подграфы в окрестностях вершин графа, что улучшает верхнюю границу BP. Однако в качестве расстояния между ними используется $\mathrm{dist}_{\mathrm{GED}}$, что позволяет получать полиномиальное время только для входных графов с ограниченной степенью. В Ring [54] применяют локальные структуры графа, известные как кольца; получена верхняя граница $\mathrm{dist}_{\mathrm{GED}}$, самая плотная для алгоритмов LSAPE.

Алгоритмы BP, Branch, Subgraph, Ring могут работать с неоднородным $\mathrm{dist}_{\mathrm{GED}}$.

Более плотную верхнюю границу получили в [55] за счет приближенного решения формулировки $\mathrm{dist}_{\mathrm{GED}}$ как задачи квадратичного программирования QAP (точное решение NP-сложное). Однако это привело к увеличению времени счета. В [56] используют несколько решений LSAPE для инициализации такого решения, что позволяет получать плотную верхнюю границу $\mathrm{dist}_{\mathrm{GED}}$, причем алгоритм легко распараллеливается.

Алгоритм локального ветвления LocBra [57] предложен для приближенного вычисления $\mathrm{dist}_{\mathrm{GED}}$ на основе MILP [51], которое итеративно модифицируется добавлением дополнительных ограничений (локальный поиск в пространстве решений MILP). Точность LocBra выше, чем у некоторых других известных приближенных алгоритмов, использующих beam search (ссылки в [57]) и QAP [55]. Хотя LocBra изначально предназначался для графов без атрибутов на ребрах, он может иметь и более широкое применение.

Отметим верхнюю границу [58] на основе различных иерархических рассмотрений графов обходом breadth-first и нового иерархического обхода и поиска соответствия для отображения графов. Она сравнима по качеству с верхними границами BP [43] и C-star [23] и их улучшениями в [23, 46] (ошибка изменяется в пределах почти от нуля до 10–20%), но вычисляется на три порядка быстрее и от двух до пяти порядков быстрее точного метода CSI-GED [46]. Вычислительная сложность составляет $O(|E||V|^2)$.

## ЗАКЛЮЧЕНИЕ

Если стоимостная функция элементарных операций редактирования является метрикой, то расстояния редактирования $\mathrm{dist}_{\mathrm{TED}}$ и $\mathrm{dist}_{\mathrm{GED}}$ также являются метриками [9, 12]. Поэтому с ними могут использоваться ИС на основе расстояний [4], т.е. ИС, не специализированные именно для деревьев и графов. Однако это вычислительно сложно, особенно для $\mathrm{dist}_{\mathrm{GED}}$, так как требует вычисления $\mathrm{dist}_{\mathrm{GED}}$ как при конструировании ИС, так и при выполнении запроса. Поэтому возможно использование модификаций этих ИС на основе быстрых алгоритмов для вычисления границ $\mathrm{dist}_{\mathrm{GED}}$ (см. подразд. 3.2).

Для ускорения поиска по $\mathrm{dist}_{\mathrm{GED}}$ можно использовать не только ИС (см. разд. 2), но и линейный поиск с фильтрацией по границам $\mathrm{dist}_{\mathrm{GED}}$ (см. подразд. 3.2). Это обусловлено тем, что этап уточнения стратегии F&R непосредственным вычислением $\mathrm{dist}_{\mathrm{GED}}$ чрезвычайно сложен ввиду экспоненциального времени вычисления $\mathrm{dist}_{\mathrm{GED}}$, что делает невозможным применение таких ИС на практике. В значительной степени это относится и к $\mathrm{dist}_{\mathrm{TED}}$, а также к аппроксимациям $\mathrm{dist}_{\mathrm{GED}}$ со значительной (кубической) сложностью. Поэтому актуальны более быстрые методы аппроксимации $\mathrm{dist}_{\mathrm{GED}}$ и исследование других мер сходства графов, которыt дают хорошие результаты в применениях.

Преобразование в векторы позволяет использовать меры сходства векторов, имеющие вычислительную сложность, линейную от размерности векторов. В [39] отмечают, что потеря локальной информации об исходных графах при этом допустима для больших графов, а для малых графов зачастую недопустима и следует использовать меры сходства непосредственно исходных графов.

Ядерные меры сходства графов (см. ссылки в [2, 3]) часто могут быть аппроксимированы или точно вычислены через скалярное произведение векторов [2, 3]. Отметим представление графов эпизодов из баз знаний в виде разреженных бинарных векторов [8], которые дают результаты оценки сходства эпизодов, близкие к результатам психологических экспериментов с человеком. Представление в таком же формате вершин графов рассмотрено в [59, 60]. Возможно снижение размерности векторов, например, случайным проецированием [2, 3, 61, 62] — рандомизированным алгоритмом, который применяется и в других задачах [63–65]. Иные подходы к вложению графов и их вершин в векторы рассмотрены в [66–69]. Перспективным направлением дальнейших исследований также является приближенное вычисление $\mathrm{dist}_{\mathrm{GED}}$ посредством вложения графов в векторное пространство на основе обучения нейронной сети [70]. При этом вычислительная сложность вложения составляет всего лишь $O(|E|)$, а оценки сходства — $O(|V|^2)$ в худшем случае.

СПИСОК ЛИТЕРАТУРЫ


1. Rachkovskij D.A. Index structures for fast similarity search for symbolic strings. C*ybernetics and Systems Analysis.* 2019. Vol. 55, N 5. P.
2. Rachkovskij D.A. Real-valued vectors for fast distance and similarity estimation. *Cybernetics and Systems Analysis.* 2016. Vol. 52, N 6. P. 967–988.
3. Rachkovskij D.A. Binary vectors for fast distance and similarity estimation. *Cybernetics and Systems Analysis.* 2017. Vol. 53, N 1. P. 138–156.
4. Rachkovskij D.A. Distance-based index structures for fast similarity search. *Cybernetics and Systems Analysis*. 2017. Vol. 53, N 4. P. 636–658.
5. Rachkovskij D.A. Index structures for fast similarity search for binary vectors. *Cybernetics and Systems Analysis.* 2017. Vol. 53, N 5. P. 799–820.
6. Rachkovskij D.A. Index structures for fast similarity search for real-valued vectors. I. *Cybernetics and Systems Analysis*. 2018. Vol. 54, N 1. P. 152–164.
7. Rachkovskij D.A. Index structures for fast similarity search for real-valued vectors. II. *Cybernetics and Systems Analysis*. 2018. Vol. 54, N 2. P. 320–335.
8. Rachkovskij D.A., Slipchenko S.V. Similarity-based retrieval with structure-sensitive sparse binary distributed representations. *Comp. Intelligence.* 2012. Vol. 28, N 1. P. 106–129.
9. Bille P. A survey on tree edit distance and related problems. *Theoretical Computer Science*. 2005. Vol. 337, N 1–3. P. 217–239.
10. Tai K.-C. The tree-to-tree correction problem. *Journal of the Association for Computing Machinery (JACM)*. 1979. Vol. 26. P. 422–433.

11. Levenshtein V.I. Binary codes capable of correcting deletions, insertions, and reversals. S*oviet Physics — Doklady*. 1966. Vol. 10, N 8. P. 707–710.
12. Gao X., Xiao B., Tao D., Li X. A survey of graph edit distance. *Pattern Analysis and Applications.* 2010. Vol. 13, N 1. P. 113–129.
13. Sanfeliu A., Fu K.S. A distance measure between attributed relational graphs for pattern recognition. *IEEE Trans. Syst. Man. Cybern.* 1983. Vol. 13, N 3. P. 353–362.
14. Zhang K., Jiang T. Some MAX SNP-hard results concerning unordered labeled trees. *Information Processing Letters*. 1994. Vol. 49. P. 249–254.
15. Pawlik M., Augsten N. Rted: a robust algorithm for the tree edit distance. *Proceedings of the VLDB Endowment.* 2011. Vol. 5, N 4. P. 334–345.
16. Pawlik M., Augsten N. Tree edit distance: Robust and memory-efficient. *Information Systems*. 2016. Vol. 56. P. 157–173.
17. Kailing K., Kriegel H.-P., Schonauer S., Seidl T. Efficient similarity search for hierarchical data in large databases. *Proc. EDBT'04.* 2004. P. 676–693.
18. Berchtold S., Keim D., Kriegel H.P. The X-tree: An index structure for high-dimensional data. *Proc. VLDB'96.* 1996. P. 28–39.
19. Yang R., Kalnis P., Tung A.K.H. Similarity evaluation on tree-structured data. *Proc. SIGMOD'05*. 2005. P. 754–765.
20. Guha S., Jagadish H.V., Koudas N., Srivastava D., Yu T. Integrating XML data sources using approximate joins. *ACM Trans. Database Syst.* 2006. Vol. 31, N 1. P. 161–207.
21. Akutsu T., Fukagawa D., Takasu A. Approximating tree edit distance through string edit distance. *Algorithmica*. 2010. Vol. 57, N 2. P. 325–348.
22. Tang Y., Cai Y., Mamoulis N. Scaling similarity joins over tree-structured data. *Proc. VLDB Endowment*. 2015. Vol. 8, N 11. P. 1130–1141.
23. Zeng Z., Tung A.K.H., Wang J., Feng J., Zhou L. Comparing stars: On approximating graph edit distance. *Proc. VLDB Endowment.* 2009. Vol. 2, N 1. P. 25–36.
24. Zhao X., Xiao C., Lin X., Wang W. Efficient graph similarity joins with edit distance constraints. *Proc. ICDE'12.* 2012. P. 834–845.
25. Gouda K., Arafa M. An improved global lower bound for graph edit similarity search. *Pattern Recogn. Lett.* 2015. Vol. 58. P. 8–14.
26. Wang G., Wang B., Yang X., Yu G. Efficiently indexing large sparse graphs for similarity search. *IEEE Trans. Knowledge and Data Engineering*. 2012. Vol. 24, N 3. P. 440–451.
27. Bougleux S., Gauzere B., Blumenthal D.B., Brun L. Fast linear sum assignment with error-correction and no cost constraints. *Pattern Recogn. Lett.* https://doi.org/10.1016/j.patrec.2018.03.032.
28. Kuhn H.W. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*. 1955. Vol. 2, N 1–2. P. 83–97.
29. Munkres J. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics.* 1957. Vol. 5, N 1. P. 32–38.
30. Wang X., Ding X., Tung A.K.H., Ying S., Jin H. An efficient graph indexing method. *Proc. ICDE'12.* 2012. P. 210–221.
31. Zhao X., Xiao C., Lin X., Wang W., Ishikawa Y. Efficient processing of graph similarity queries with edit distance constraints. *VLDB J.* 2013. Vol. 22. P. 727–752.
32. Riesen K., Fankhauser S., Bunke H. Speeding up graph edit distance computation with a bipartite heuristic. *Proc. MLG'07.* 2007. P. 21–24.
33. Zheng W., Zou L., Lian X., Wang D., Zhao D. Efficient graph similarity search over large graph databases. *IEEE TKDE.* 2015. Vol. 27, N 4. P. 964–978.
34. Li Z., Jian X., Lian X., Chen L. An efficient probabilistic approach for graph similarity search. *Proc. ICDE'18*. 2018. P. 533–544.
35. Qin J., Xiao C. Pigeonring: a principle for faster thresholded similarity search. *Proc. VLDB Endow*. 2018. Vol. 12, N 1. P. 28–42.

36. Zhao X., Xiao C., Lin X., Zhang W., Wang Y. Efficient structure similarity searches: A partition-based approach. *The VLDB Journal*. 2018. Vol. 27, N 1. P. 53–78.

37. Liang Y., Zhao P. Similarity search in graph databases: A multilayered indexing approach. *Proc. ICDE'17*. 2017. P. 783–794.

38. Abu-Aisheh Z., Raveaux R., Ramel J.-Y. Efficient k-nearest neighbors search in graph space. *Pattern Recognition Letters*. https://doi.org/10.1016/j.patrec.2018.05.001.

39. Blumenthal D.B., Gamper J. Improved lower bounds for graph edit distance. *IEEE TKDE*. 2018. Vol. 30, N 3. P. 503–516.

40. Blumenthal D.B., Gamper J. On the exact computation of the graph edit distance. *Pattern Recognition Letters*. 2018. https://doi.org/10.1016/j.patrec.2018.05.002/.

41. Abu-Aisheh Z., Gauzere B., Bougleux S., Ramel J.-Y., Brun L., Raveaux R., Heroux P., Adam S. Graph edit distance contest: Results and future challenges. *Pattern Recognition Letters*. 2017. Vol. 100. P. 96–103.

42. Riesen K., Neuhaus M., Bunke H. Bipartite graph matching for computing the edit distance of graphs. *Proc. GbRPR'07*. 2007. P. 1–12.

43. Riesen K., Bunke H. Approximate graph edit distance computation by means of bipartite graph matching. *Image and Vision Computing*. 2009. Vol. 27, N 7. P. 950–959.

44. Abu-Aisheh Z., Raveaux R., Ramel J.Y., Martineau P. An exact graph edit distance algorithm for solving pattern recognition problems. *Proc. ICPRAM'15*. 2015. P. 271–278.

45. Abu-Aisheh Z., Raveaux R., Ramel J.-Y., Martineau P. A parallel graph edit distance algorithm. *Expert Systems with Applications*. 2018. Vol. 94. P. 41–57.

46. Gouda K., Hassaan M. A novel edge-centric approach for graph edit similarity computation. *Information Systems*. 2019. Vol. 80. P. 91–106.

47. Chen X., Huo H., Huan J., Vitter J.S. An efficient algorithm for graph edit distance computation. *Knowledge-Based Systems*. 2019. Vol. 163. P. 762–775.

48. Zhou R., Hansen E.A. Beam-stack search: Integrating backtracking with beam search. *Proc. ICAPS'05*. 2005. P. 90–98.

49. Chang L., Feng X., Lin X., Qin L., Zhang W. Efficient graph edit distance computation and verification via anchor-aware lower bound estimation. arXiv:1709.06810. 1 Oct 2017.

50. Justice D., Hero A. A binary linear programming formulation of the graph edit distance. *IEEE Trans. Pattern Anal. Mach. Intell*. 2006. Vol. 28, N 8. P. 1200–1214.

51. Lerouge J., Abu-Aisheh Z., Raveaux R., Heroux P., Adam S. New binary linear programming formulation to compute the graph edit distance. *Pattern Recognition*. 2017. Vol. 72. P. 254–265.

52. Darwiche M., Raveaux R, Conte D., T'Kindt V. Graph Edit Distance in the exact context. *Proc. S+SSPR'18*. 2018. P. 304–314.

53. Carletti V., Gauzere B., Brun L., Vento M. Approximate graph edit distance computation combining bipartite matching and exact neighborhood substructure distance. *Proc. GbRPR'15*. 2015. P. 188–197.

54. Blumenthal D., Bougleux S., Gamper J., Brun L. Ring based approximation of graph edit distance. *Proc. S+SSPR'18*. 2018. P. 293–303.

55. Bougleux S., Brun L., Carletti V., Foggia P., Gauzere B., Vento M. Graph edit distance as a quadratic assignment problem. *Pattern Recognition Letters*. 2017. Vol. 87. P. 38–46.

56. Daller E., Bougleux S., Gauzere B., Brun L. Approximate graph edit distance by several local searches in parallel. *Proc. ICPRAM'18*. 2018. P. 149–158.

57. Darwiche M., Conte D., Raveaux R., T'Kindt V. A local branching heuristic for solving a graph edit distance problem. *Comp. & Oper. Res*. https://doi.org/10.1016/j.cor.2018.02.002.

58. Gouda K., Arafa M., Calders T. A novel hierarchical-based framework for upper bound computation of graph edit distance. *Pattern Recognition*. 2018. Vol. 80. P. 210–224.

59. Slipchenko S.V., Rachkovskij D.A. Analogical mapping using similarity of binary distributed representations. *Information Theories & Applications*. 2009. Vol. 16, N 3. P. 269–290.

60. Rachkovskij D.A. Some approaches to analogical mapping with structure-sensitive distributed representations. *Journal of Experimental & Theoretical Artificial Intelligence.* 2004. Vol. 16, N 3. P. 125–144.

61. Rachkovskij D.A. Formation of similarity-reflecting binary vectors with random binary projections. *Cybernetics and Systems Analysis.* 2015. Vol. 51, N 2. P. 313–323.

62. Рачковський Д.А., Гриценко В.І. Розподілене подання векторних даних на основі випадкових проекцій. Київ: Інтерсервіс, 2018. 216 с.

63. Rachkovskij D.A., Revunova E.G. A randomized method for solving discrete ill-posed problems. *Cybernetics and Systems Analysis*. 2012. Vol. 48, N 4. P. 621–635.

64. Revunova E.G. Model selection criteria for a linear model to solve discrete ill-posed problems on the basis of singular decomposition and random projection. *Cybernetics and Systems Analysis.* 2016. Vol. 52, N 4. P. 647–664.

65. Revunova E.G. Averaging over matrices in solving discrete ill-posed problems on the basis of random projection. *Proc. CSIT'17*. 2017. P. 473–478.

66. Riba P., Llados J., Fornes A., Dutta A. Large-scale graph indexing using binary embeddings of node contexts for information spotting in document image databases. *Pattern Recognition Letters*. 2017. Vol. 87. P. 203–211.

67. Narayanan A., Chandramohan M., Venkatesan R., Chen L., Liu Y., Jaiswal S. Graph2vec: Learning distributed representations of graphs. *Proc. MLG'17.* 2017. P. 21:1–21:8.

68. Goyal P., Ferrara E. Graph embedding techniques, applications, and performance: A survey. *Knowledge Based Systems*. 2018. Vol. 151. P. 78–94.

69. Wu Z., Pan S., Chen F., Long G., Zhang C., Yu P.S. A comprehensive survey on graph neural networks. arXiv:1901.00596. 10 Mar. 2019.

70. Bai Y., Ding H., Bian S., Chen T., Sun Y., Wang W. SimGNN: A neural network approach to fast graph similarity computation. *Proc. WSDM'19*. 2019. P. 384–392.

*Надійшла до редакції 25.04.2019*

## Д.А. Рачковський

### ШВИДКИЙ ПОШУК СХОЖИХ ГРАФІВ ЗА ВІДСТАННЮ РЕДАГУВАННЯ

**Анотація.** Наведено огляд індексних структур для швидкого пошуку за схожістю об'єктів, поданих деревами та графами. Як міру схожості використано відстань редагування. Розглянуто виконання запитів точного пошуку за схожістю. В основному описано алгоритми на основі стратегії фільтрації та уточнення, які використовують обернене індексування. Крім того, розглянуто алгоритми точного обчислення відстані редагування графів та її нижніх і верхніх меж.

**Ключові слова:** пошук за схожістю, графи, відстань редагування, найближчий сусід, індексні структури, обернене індексування.

## D.A. Rachkovskij

### FAST SEARCH FOR SIMILAR GRAPHS BY EDIT DISTANCE

**Abstract.** This survey article considers index structures for fast similarity search for objects represented by trees and graphs. Edit distance is used as a measure of similarity. The execution of exact similarity search queries is considered. Algorithms based on the filter-and-refine strategy using inverted indexing are mainly presented. Algorithms for exact calculation of the graph edit distance and its lower and upper bounds are also considered.

**Keywords:** similarity search, graphs, edit distance, nearest neighbor, index structures, inverted indexing.

**Рачковский Дмитрий Андреевич,**
доктор техн. наук, ведущий научный сотрудник Международного научно-учебного центра информационных технологий и систем НАН Украины и МОН Украины, Киев, e-mail: dar@infrm.kiev.ua.