УДК 004.22 + 004.93′11

Д.А. РАЧКОВСКИЙ

ОСНОВАННЫЕ НА РАССТОЯНИЯХ ИНДЕКСНЫЕ СТРУКТУРЫ ДЛЯ БЫСТРОГО ПОИСКА ПО СХОДСТВУ

Аннотация. Рассмотрен класс таких индексных структур для быстрого поиска по сходству, при конструировании и применении которых используется только информация о значениях или ранге некоторых расстояний/сходств между объектами. Обсужден поиск как по метрическим расстояниям (для последних выполняется неравенство треугольника и другие метрические аксиомы), так и по неметрическим. Представлены структуры, которые возвращают объекты базы, являющиеся точным ответом на поисковый запрос, а также структуры для приближенного поиска по сходству (они не гарантируют точность, но обычно возвращают близкие к точным результаты и работают быстрее структур для точного поиска). Изложены общие принципы конструирования и применения некоторых индексных структур, а также рассмотрены идеи, лежащие в основе конкретных алгоритмов, как известных, так и предложенных в последнее время.

Ключевые слова: поиск по сходству, поиск ближайшего соседа, индексные структуры, индексирование на основе расстояний, метрическое расстояние, неметрическое расстояние, метрическое дерево, граф соседства, метод ветвей и границ.

ВВЕДЕНИЕ

Зачастую единственным способом найти нужную информацию (объекты) в больших массивах данных является определение ее сходства с уже имеющимися примерами такой информации. Последние используются в качестве запросов к базе (набору, массиву, множеству) объектов. Величина сходства является критерием упорядочивания объектов базы относительно объекта-запроса (не принадлежащего базе). Таким образом, поиск по сходству — это вариант информационного поиска, при котором объекты базы, релевантные объекту-запросу, определяются на основе числовых значений некоторых мер (функций) сходства или несходства (расстояния) между представлениями объектов.

Полученные поиском по сходству данные можно использовать непосредственно или в качестве источника дополнительной информации о входном объекте-запросе. Так, если найденные изображения или тексты удовлетворяют информационную потребность пользователя, они могут быть конечным результатом поиска [1, 2]. Если поставлена задача определить принадлежность объекта-запроса к некоторому классу, для ее решения можно применить метод классификации ближайшего соседа, присваивающий объекту-запросу тот класс, к которому принадлежат сходные с ним объекты [3]. Это простой вариант подхода «рассуждения на основе примеров» (example based reasoning), к его разновидностям относятся рассуждения по прецедентам (case-based reasoning) и по аналогии (analogical reasoning), см. [4–9] и ссылки к ним. Подход использует сходные прецеденты или аналоги для выводов о входном объекте или ситуации и применяется как людьми, так и техническими системами для решения широкого круга задач.

Практическая полезность алгоритмов поиска по сходству определяется скоростью поиска и затратами памяти, а также соответствием результатов поиска потребностям пользователя. Последнее зависит, в частности, от используемых представлений объектов и мер расстояния/сходства, базы и объектов-запросов. Будем считать их заданными.

Проще всего выполнить поиск по сходству вычислением величин сходств объекта-запроса со всеми объектами базы и возвращением объектов, удовлетворяющих условиям запроса. Такой линейный или последовательный поиск часто оказывается недопустимо медленным, особенно для больших баз, сложных многокомпонентных представлений объектов и вычислительно сложных мер сходства.

Один подход к ускорению линейного поиска по сходству состоит в быстрой (хотя и приближенной) оценке величины всех расстояний/сходств. Для этого обычно формируют новые представления объектов в виде вещественных векторов малой размерности (см. обзор в [10]) или бинарных векторов (см. обзор в [11]), позволяющих провести ускоренную оценку. Другой подход заключается в построении по базе такой структуры данных, использование которой при выполнении запроса поиска по сходству приведет к уменьшению количества вычислений сходства объекта-запроса с другими объектами по сравнению с линейным поиском. Отметим, что алгоритмы, разработанные в рамках обоих подходов, зачастую обеспечивают ускорение поиска за счет получения результатов, не полностью совпадающих с результатами (точного) линейного поиска по сходству.

Настоящий обзор посвящен алгоритмам и структурам ускорения поиска по сходству, разработанным в рамках второго подхода, использующим только значения расстояний/сходств между некоторыми объектами и не требующим явного доступа к представлениям объектов. Отметим, что для метрических расстояний выполняются метрические аксиомы, включая неравенство треугольника, которое широко используют для ускорения поиска.

В отличие от классических публикаций по данной теме [12–15] в настоящем обзоре представлены новые подходы и алгоритмы, а также усовершенствования уже известных. Ввиду ограниченного объема статьи в ней изложены лишь базовые идеи, детальную информацию можно найти в приведенных ссылках. В разд. 1 рассмотрены распространенные типы запросов поиска по сходству, основы построения индексных структур, проблемы точного поиска и необходимость приближенного поиска для ускорения выполнения запроса для многомерных данных. В разд. 2 обсуждаются индексные структуры для поиска по метрическим расстояниям, а в разд. 3 — по неметрическим расстояниям, некоторые из последних можно адаптировать для поиска по мерам сходства.

1. ИНДЕКСНЫЕ СТРУКТУРЫ ДЛЯ УСКОРЕНИЯ ПОИСКА ПО СХОДСТВУ. ОСНОВНЫЕ ПРИНЦИПЫ

Объекты базы, которые являются ответом на конкретный запрос поиска по сходству, определяются типом запроса и его параметрами, заданными объектом-запросом (не принадлежащим базе), мерой расстояния/сходства и др. Диапазонный запрос (обозначим его rNN) возвращает объекты базы, расстояния которых от объекта-запроса (по мере расстояния, заданной в запросе) не превышают радиуса запроса r. При некоторых r результатом диапазонного запроса может быть пустое множество объектов или все объекты конкретной базы.

Запрос k ближайших соседей (kNN) возвращает k объектов базы, ближайших к объекту-запросу. Обозначим NN запрос kNN при $k = 1$, а также объект, который является ближайшим соседом. Запрос kNN можно, например, выполнить как запрос rNN при r, равном расстоянию до k-го NN (если такой r известен), или последовательностью запросов rNN с увеличением r до тех пор, пока количество возвращенных объектов не достигнет k [12–15]. Далее рассмотрены и другие варианты выполнения запроса kNN (подразд. 1.2 и 1.5).

Определения типов запросов в терминах сходства аналогичны.

Кроме запросов rNN и kNN, существуют и другие типы запросов поиска по сходству [12–17], однако в настоящем обзоре индексные структуры для них не рассматриваются.

Многие алгоритмы ускорения поиска по сходству используют двухэтапную стратегию фильтрации и уточнения (filter-and-refine, 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). Стратегию F&R иногда применяют многократно при выполнении одного запроса.

1.1. Общие принципы конструирования и применения индексных структур. Индексные структуры (ИС или индексы) для поиска по сходству — это структуры данных, в которых объекты базы организованы таким образом, чтобы ускорялось выполнение запроса по сравнению с линейным поиском за счет уменьшения количества вычисляемых расстояний/сходств.

Конструируют ИС офлайн, т.е. до выполнения запросов. Для построения ИС объектам базы ставятся в соответствие индексные значения (ключи): одно или несколько для каждого объекта. Обычно индексные значения — это вещественные числа или дискретные величины. Например, вещественными индексными значениями могут являться расстояния/сходства объекта с некоторыми другими объектами, а дискретными — идентификаторы областей, которым принадлежит объект.

Индексные значения объектов используют для конструирования ИС, в частности, выполнением следующих операций (некоторых или всех):

— запоминание индексных значений объектов (например, расстояний до других объектов);

— упорядочивание объектов базы (например, по возрастанию или убыванию расстояний до других объектов);

— разбиение (partitioning) объектов базы на множества (например, в одно множество помещают объекты с одинаковыми индексными значениями или с индексными значениями в некотором диапазоне (в частности, в диапазоне расстояний до некоторого объекта); либо объект ассоциируют с его ближайшими соседями из базы).

При конструировании ИС возможно применение и других операций. Отметим, что, если имеется доступ к представлениям объектов (например, объекты являются векторами), индексирование может выполняться на основе не только расстояний/сходств, но и иных характеристик представлений (например, для векторов можно использовать значения отдельных компонентов, результаты операций над векторами и т.д.). В некоторых случаях для одной базы конструируют несколько ИС на основе различных индексных значений.

Существуют статические и динамические ИС. Алгоритм конструирования статических ИС предполагает наличие всей базы объектов, конструирование динамических ИС осуществляется операциями вставки объектов по мере поступления данных.

При выполнении запроса поиска по сходству вычисляют индексные значения объекта-запроса. Далее ИС и алгоритм выполнения запроса заданного типа позволяют определить, какие объекты базы (или их множества) могут являться ответом (или содержать ответ) на запрос и подлежат дальнейшей обработке, а какие — нет и для ускорения поиска их следует отбросить (discard), обрезать (prune) или исключить (eliminate).

В качестве примера рассмотрим ИС для объектов, представленных одним числом, с расстоянием — модулем разности чисел. Конструирование проведем сортировкой элементов массива размерности N, соответствующего N объектам базы. Для выполнения запроса NN применим поиск дихотомией (binary search): объект-запрос x сравним с объектом в серединном (медианном) элементе массива. Если они не совпадают, половину массива исключим из рассмотрения и перейдем к серединному элементу оставшейся половины. Повторение процедуры

находит NN за время $O(\log N)$ в худшем случае. Затраты памяти в этой ИС составляют $O(N)$, а время конструирования $O(N \log N)$. Для выполнения запроса kNN рассмотрим соседние объекты найденного NN. Для запроса rNN выполним два запроса NN с объектами-запросами: $x - r$ и $x + r$; ответом на запрос rNN являются объекты массива между найденными объектами.

1.2. Метод ветвей и границ в поиске по сходству. Конструирование и использование многих ИС можно рассматривать как применение разновидностей метода ветвей и границ (branch and bound, B&B) к поиску по сходству [18]. Объекты базы разбиты на подмножества, ассоциированные с областями индекса. Разбиения могут быть иерархическими (подразд. 1.5), а также динамическими (т.е. в процессе выполнения запроса, например, подразд. 2.1).

При выполнении запроса поиска запоминается текущая верхняя граница расстояния от объекта-запроса до объекта базы, который может являться ответом на запрос (граница решения). Так, для запроса rNN эта граница неизменна и равна r. Для запроса kNN текущая граница решения модифицируется в ходе выполнения запроса и равна расстоянию до k-го объекта-кандидата (из текущего списка обработанных объектов с минимальными расстояниями до объекта-запроса).

Кроме того, определяются нижняя и верхняя границы расстояний от объекта-запроса до еще не обработанных подмножеств объектов базы. Если нижняя граница больше границы решения, подмножество исключают, иначе переходят к его обработке. Если верхняя граница меньше границы решения для запроса rNN, подмножество включают в ответ. Аналогично обрабатывают индивидуальные объекты (это имеет смысл, если границы для них вычисляются быстрее, чем расстояния до объекта-запроса). Среди объектов, которые не удалось исключить, выполняют линейный поиск и принимают решение о включении их в ответ на запрос rNN (или в список текущих кандидатов на kNN). Такие процедуры проводят, пока остаются необработанные подмножества объектов. В результате, если удается исключить часть объектов базы без вычисления расстояний до них или вообще без их рассмотрения, получаем сублинейный поиск (т.е. ускоренный по сравнению с линейным). Конкретные алгоритмы B&B используют различные способы разбиения базы на подмножества, очередности обхода подмножеств и определения границ. Их можно применять как для метрических, так и для неметрических расстояний/сходств, если имеется механизм определения границ.

1.3. Опорные объекты и области индексной структуры и запроса. Для получения границ расстояний/сходств, применяемых в методологии B&B, удобно оперировать областями, которым принадлежат множества объектов базы. В ИС на основе расстояний/сходств области обычно задают разбиением сферами и обобщенными гиперплоскостями, а также их вариантами и комбинациями [12–15]. Для этого используют некоторые выделенные (или опорные) объекты базы. Расстояния до опорных объектов (ОО) также могут непосредственно применяться в качестве индексных значений индивидуальных объектов базы.

Сферическое разбиение задает сферическую область с центром в ОО o_i и радиусом r_i. Ее внутренней части принадлежат все объекты y, для которых $\text{dist}(o_i, y) \le r_i$. (Таким образом, для запроса rNN областью запроса является сферическая область с центром в объекте-запросе x и радиусом r). Внешней части сферической области принадлежат все объекты y, для которых $\text{dist}(o_i, y) > r_i$. Количество объектов в подмножествах можно регулировать значением r_i. Для заключения всего множества объектов в сферическую область выбирают ОО o_i и задают покрывающий радиус $r_{\text{cov}}(o_i)$ как расстояние до самого удаленного объекта множества. В ИС применяют области между сферами с одним центром и отличающимися радиусами, а также области, задаваемые с помощью сфер с различными центрами, и др. (подразд. 2.2).

Гиперплоскостное разбиение задают двумя ОО: o_i и o_j, и параметром η. К области o_i принадлежат объекты y, для которых $\text{dist}(o_i, y) \le \text{dist}(o_j, y) + \eta$, а к области o_j — объекты, для которых $\text{dist}(o_i, y) > \text{dist}(o_j, y) + \eta$ [19]. Выбором η можно регулировать количество объектов в областях. Множество ОО задает области Дирихле (Dirichlet domain) — аналог диаграммы Вороного, используемой в векторных пространствах. Область Дирихле для o_i — это область, где объекты ближе к o_i, чем к другим ОО из множества.

В ИС на основе расстояний применяют различные комбинации вариантов сферических и гиперплоскостных разбиений. Согласно методологии B&B, если область запроса не пересекает область ИС, последнюю (и ее объекты) можно исключить без ущерба для точности поиска.

Некоторые достаточные условия исключения областей при использовании метрических расстояний рассмотрены в подразд. 1.4. Для неметрических расстояний и сходств в общем случае не существует универсальных условий исключения, поэтому нужно учитывать специфические свойства конкретных неметрик (см., например, [20] и подразд. 2.7, 3.1), что не всегда возможно, либо использовать ИС не на основе методологии B&B (подразд. 3.1.3 и 3.2–3.4).

1.4. Обработка областей и объектов на основе вариантов неравенства треугольника. В ИС запомнена информация об областях индекса и/или о расстояниях $\text{dist}(o_i, y)$ объектов базы y до ОО. При поступлении запроса rNN известен его радиус r и вычисляется расстояние $\text{dist}(o_i, x)$ от объекта-запроса x до ОО. Для метрических расстояний эта информация в сочетании с вариантами неравенства треугольника [12–15] позволяет исключить некоторые множества объектов в областях ИС и индивидуальные объекты базы без вычисления расстояний $\text{dist}(x, y)$.

Для метрических расстояний и сферического разбиения область внутри сферы с ОО o_i и $r_{\text{cov}}(o_i)$ и (сферическая) область rNN запроса не пересекаются, если

$$\text{dist}(x, o_i) > r + r_{\text{cov}}(o_i). \tag{1}$$

Область вне этой сферы не пересекает область запроса, если

$$\text{dist}(x, o_i) \le r_{\text{cov}}(o_i) - r. \tag{2}$$

Отметим, что при невыполнении неравенств (1), (2) соответствующие области пересекаются.

Для метрических расстояний и гиперплоскостного разбиения, если x принадлежит области o_i, область o_j гарантированно не пересекает область запроса при

$$\text{dist}(o_j, x) - \text{dist}(o_i, x) > 2r + \eta. \tag{3}$$

Отметим, что при невыполнении неравенства (3) и при $\text{dist}(o_i, x) - \text{dist}(o_j, x) > \text{dist}(o_i, o_j)$ области также могут не пересекаться [13, 21].

Таким образом, выполнение неравенств (1)–(3) гарантирует непересечение области запроса соответствующими областями, и объекты базы таких областей можно исключить из дальнейшей обработки. Индивидуальные объекты y можно исключить, если

$$|\text{dist}(x, o_i) - \text{dist}(y, o_i)| > r, \tag{4}$$

так как для них согласно неравенству треугольника $\text{dist}(x, y) > r$.

Если имеется d опорных объектов o_i, $i = 1, \ldots, d$, то y подлежит исключению при выполнении (4) хотя бы для одного o_i. Это условие исключения можно переписать с использованием векторного представления $\boldsymbol{\varphi}(y)$, где значение компонента $\varphi_i(y) = \text{dist}(y, o_i)$, как $\max_i |\text{dist}(x, o_i) - \text{dist}(y, o_i)| = L_\infty(\boldsymbol{\varphi}(x), \boldsymbol{\varphi}(y)) > r$, здесь L_∞ — расстояние Чебышева. Отметим, что это сжимающее преобразование: $\text{dist}(x, y) \ge L_\infty(\boldsymbol{\varphi}(x), \boldsymbol{\varphi}(y))$ [12, 13, 22].

Рассмотренные и другие условия исключения вариантами неравенства треугольника (например, [23]) можно применять совместно, увеличивая вероятность исключения. Некоторые метрические расстояния имеют дополнительные свойства, которые позволяют более эффективное исключение, чем неравенство треугольника (подразд. 2.7).

1.5. Древовидные индексные структуры и алгоритмы их обхода. Для ускорения построения и использования ИС в ряде случаев разбиение базы на множества объектов проводят иерархически: полученные в результате разбиения множества рекурсивно разбивают на более мелкие подмножества. Часто иерархические индексы для поиска по сходству имеют древовидную структуру. Множеству объектов базы (и их области) обычно соответствует узел дерева. Верхний (корневой) узел соответствует всем объектам базы, а узлы-сыновья — подмножествам объектов узла-родителя. В деревьях узел-сын имеет ровно одного родителя. Чем ниже уровень узлов, тем меньше объектов в их подмножестве и тем меньшие их области разбиения. Узел содержит информацию, позволяющую по индексным значениям отнести объект к его узлам-сыновьям и их областям. Объекты базы или ссылки на них содержатся в корзинах. В деревьях корзинами часто являются листья (узлы, не имеющие сыновей). Корзине листа может принадлежать один или более объектов базы.

При выполнении запроса NN на основе метода B&B порядок обхода узлов (областей) существенно влияет на время поиска, так как быстрое нахождение текущего ближайшего соседа с малым расстоянием от объекта-запроса уменьшает область запроса, что может значительно сократить количество подлежащих обработке областей индекса.

В древовидных ИС для быстрого нахождения «хорошего» текущего NN вначале выполняют спуск по дереву из корневого узла в листовой [13]. Для перехода в каждом узле выбирают узел-сын (область) с минимальным расстоянием от объекта-запроса (обычно область, которой принадлежит объект-запрос). В корзине листа линейным поиском находят ближайший объект и используют в качестве текущего NN. Зачастую найденный объект является хорошим кандидатом, однако точный NN может находиться в других корзинах. Поэтому продолжают обход узлов дерева в порядке, который зависит от алгоритма обхода.

При алгоритме обхода depth-first [13] далее выполняют обратное прослеживание (backtracking) — из посещенного листового узла возвращаются на узел-родитель и переходят на другие его листовые узлы, если их области пересекают текущую область запроса. Текущий радиус модифицируют, когда находят более близкий к запросу объект базы. По исчерпании листовых узлов текущего узла выполняется возврат к его узлу-родителю верхнего уровня и рекурсивно продолжается обход непосещенных узлов-сыновей, области которых пересекают текущую область запроса. Фактически указанный порядок обхода определяется порядком извлечения узлов из стека (LIFO), куда заносятся сыновья каждого узла при его посещении.

При алгоритме обхода best-first [13] вычисленные при первоначальном спуске из корня в лист расстояния до узлов (их областей) различных уровней заносят в приоритетную очередь с сортировкой по увеличению. Для дальнейшего обхода из приоритетной очереди извлекают первый узел и обходят его (непосещенные) узлы-сыновья, области которых пересекают текущую область запроса, добавляя в очередь узлы с их расстояниями и модифицируя текущий радиус запроса. Для точного поиска обход прекращают, когда расстояние до первого узла очереди превышает текущий радиус запроса. При приближенном поиске best-first позволяет быстро находить «хороших» соседей и останавливать поиск раньше, чем при точном.

1.6. От точного к приближенному поиску по сходству. Время выполнения запроса в ИС наиболее исследовано для векторных данных. Анализ всех известных алгоритмов точного поиска по сходству показывает, что затраты времени или памяти растут экспоненциально с увеличением размерности данных D [16, 17]. Это явление называют «проклятием размерности» (curse of dimensionality) в поиске по сходству [12, 17, 24, 25]. Для алгоритмов с затратами памяти, близкими к линейным, лучшее известное время $\min(2^{O(D)}, DN)$ для случайного объекта-запроса [17] (т.е. для среднего случая и тем более для худшего). Поэтому для баз с достижимым N с ростом D точный поиск вырождается в линейный, даже при не слишком больших D, что подтверждается и на практике [24]. Для общих метрических пространств с неизвестным представлением объектов проклятие размерности обсуждается в терминах внутренней размерности (intrinsic dimension) [23, 26–33]. Например, в [12] это отношение квадрата среднего значения и дисперсии расстояний между объектами базы.

Преодолеть (точнее, обойти) проклятие размерности для поиска по сходству удается переходом от точного к более быстрому, но приближенному поиску, который допускает отличие результатов от точного поиска. На практике он востребован, поскольку для многих приложений достаточно получать приближенные результаты, но быстро. Многие подходы к ускорению точного поиска можно рассматривать как обработку при выполнении запроса только части тех объектов базы, которые необходимо обработать при точном поиске. В исключенной части могут содержаться объекты, являющиеся точным результатом. При агрессивной обрезке (aggressive pruning) [34] дополнительно к подмножествам, исключаемым точным поиском, исключают некоторые другие области, например, уменьшая текущий радиус запроса. В ряде случаев это позволяет получить гарантированное отношение расстояний объекта-запроса до найденного и точного NN. Ранний останов (early stopping) [34] прекращает поиск до гарантированного достижения точного результата. Критерием останова может являться, например, количество обработанных объектов базы. Количественных гарантий качества результатов поиска обычно не существует. Ранний останов применим для алгоритмов любого типа, которые быстро находят точных ближайших соседей либо их хорошее приближение. Варианты агрессивной обрезки и раннего останова, а также различные типы гарантий качества результатов приближенного поиска обсуждаются в [34].

На практике качество результатов приближенного поиска (степень их отличия от результатов точного поиска) оценивают в экспериментах на конкретных базах. Часто используют меры качества выполнения конкретного запроса, известные как [35] полнота (recall), равная $n1/n2$, и точность (precision), равная $n1/n3$, где $n1$ — количество возвращенных объектов, совпавших с релевантными, $n2$ — количество релевантных запросу объектов в базе, $n3$ — количество возвращенных объектов базы. Для запроса kNN $n2 = n3 = k$, поэтому точность равна полноте.

«Хорошими» соседями могут являться не только точные, но и объекты базы, близкие к ним. Поэтому качество поиска также измеряют отношением суммы (или среднего) расстояний объекта-запроса до возвращенных объектов к соответствующей величине для точного результата.

При выполнении множества запросов полученные для индивидуальных запросов значения мер качества усредняют. Так, для запроса NN полноту (равную точности) измеряют как процент запросов, для которых был возвращен точный NN [36].

Компромисс качества и времени поиска по сходству обычно обеспечивается выбором параметров ИС. Поэтому ИС для приближенного поиска сравнивают по

(усредненному) времени выполнения запроса при фиксированном параметре качества поиска или по значению меры качества поиска при одинаковом времени поиска. Важной характеристикой также являются затраты памяти на ИС (квадратичные затраты обычно неприемлемы).

Для точного поиска определяется лучшая ИС по минимальному времени поиска.

2. ИНДЕКСНЫЕ СТРУКТУРЫ ДЛЯ ПОИСКА ПО МЕТРИЧЕСКИМ РАССТОЯНИЯМ

Большинство рассматриваемых в данном разделе ИС предназначено для точного выполнения запросов поиска по сходству (rNN, а также kNN) вариантами метода B&B (см. подразд. 1.2) с исключением областей и объектов разновидностями неравенства треугольника (см. подразд. 1.4). Поэтому следует ожидать экспоненциальной зависимости времени выполнения запроса от внутренней размерности данных (см. подразд. 1.6). На практике это приводит к широкому диапазону времени поиска $O(N^{\alpha})$ для некоторого $0 \le \alpha \le 1$ [37], где α зависит от особенностей данных, запросов, а также меры расстояния и ИС. Зачастую не имеется аналитических оценок времени поиска, используются экспериментальные оценки на конкретных базах. Во многих случаях возможно ускорение поиска за счет потери гарантий точности такими стандартными приемами, как ранний останов или агрессивная обрезка (см. подразд. 1.6).

В разд. 2 рассмотрены следующие типы ИС: неиерархические на основе предвычисленных расстояний (подразд. 2.1); древовидные с разбиением на области гиперсферами и гиперплоскостями, а также их усовершенствования (подразд. 2.2); на основе ИС для скалярных и векторных данных (подразд. 2.3); деревья соседства (подразд. 2.4); с неиерархическим разбиением на области (подразд. 2.5). В подразд. 2.6 обсуждается выбор ОО, а в подразд. 2.7 — условия исключения без использования неравенства треугольника.

2.1. Неиерархические индексные структуры на основе запоминания расстояний между объектами. Для конструирования ИС AESA [38, 39] вычисляется матрица расстояний между всеми объектами базы. При выполнении запроса вычисляют расстояние $\mathrm{dist}(x, o_i)$ от объекта-запроса x до некоторого ОО (объекта базы o_i, например, выбранного случайно) и исключают объекты базы y, для которых выполняется неравенство (4). Из оставшихся объектов базы выбирают следующий ОО и повторяют процедуру исключения. Процесс продолжают, пока не исчерпается множество ОО либо до заданного количества оставшихся объектов. Среди этих объектов-кандидатов выполняют линейный поиск. Предложен ряд эвристических процедур выбора очередных ОО, повышающих эффективность исключения, например iAESA [40]. Затраты памяти на хранение матрицы расстояний AESA составляют $O(N^2)$, выполнение запроса требует от $O(N)$ до $O(N^2)$ обращений к матрице расстояний. Экспериментально (для данных малой размерности) среднее время выполнения запроса постоянно [38].

Для больших N квадратичные затраты памяти AESA неприемлемы. Для уменьшения размера матрицы расстояний в ИС LAESA [41] в качестве ОО используют не все, а фиксированное подмножество объектов базы. Скорость исключения повышают применением поиска дихотомией (см. подразд. 1.1) для определения объектов-кандидатов y, для которых $\mathrm{dist}(y, o_i)$ находится в диапазоне $[\mathrm{dist}\,(x, o_i) - r,\ \mathrm{dist}\,(x, o_i) + r]$ [42, 43]. Отметим, что результаты отбора кандидатов, тождественные (L)AESA, дает линейный поиск по расстоянию L_∞ векторных представлений объектов, с компонентами — расстояниями до ОО (см. подразд. 1.4). Ускорение нахождения пересечения множеств кандидатов, полученных для каждого ОО с использованием сжатого представления перестановок [44], предложено в [45]. Приближенный поиск по сходству в (L)AESA рассмотрен в [40] (ранний останов) и [46] (агрессивная обрезка) и ссылках к ним.

В работе [47] ИС EPT состоит из заданного количества групп ОО. В отличие от (L)AESA в каждой группе N объектов базы разбиты на непересекающиеся подмножества со своим «хорошим» ОО для подмножества. Для каждого объекта группы запоминается его ОО и расстояние до него. Для улучшения исключения и соответствующего ускорения поиска количество и состав ОО в группе оптимизируют (подразд 2.6). В экспериментах EPT работает быстрее LAESA, SAT и LC (подразд. 2.4 и 2.5) при меньших затратах памяти.

2.2. Классические метрические деревья и их усовершенствования. Типичные иерархические ИС на основе расстояний с разбиением объектов базы на множества — это метрические деревья (metric trees) [48] и их разновидности. Для разбиения в основном используют сферы, гиперплоскости (см. подразд. 1.3) и их варианты, применяют также запоминание расстояний (объектов базы до ОО, между ОО). Поиск осуществляют вариантами F&R и B&B (см. разд. 1).

Анализ времени выполнения запроса точного поиска обычно проводят для среднего случая. При достаточно малом радиусе области запроса (т.е. при минимальном обратном прослеживании) для сбалансированных деревьев (у которых все пути от корня к листьям отличаются не более чем на 1) время составляет $O(\log N)$ (время спуска из корня в лист). Для данных базы с большой внутренней размерностью время достигает $O(N)$.

2.2.1. Метрические деревья на основе сферических областей. В [49, 50] рекурсивно строят ИС VP-tree как бинарное дерево, разбивая множество объектов базы на подмножества сферическими областями (см. подразд. 1.3). В корне дерева выбирают в качестве ОО один из объектов базы и такой радиус сферы с центром в этом ОО, чтобы половина объектов базы оказалась внутри нее, а половина — вне сферы (радиус равен медианному значению расстояний до ОО). Поддеревья конструируют рекурсивно, выбирая ОО в каждом подмножестве. Дерево получается сбалансированным (по высоте и числу объектов в узлах одного уровня). Время построения составляет $O(N \log N)$, а затраты памяти $O(N)$. Запросы выполняют методом B&B, переход на узлы-сыновья (один или оба) осуществляют согласно (1), (2).

Для уменьшения количества ОО (и вычислений расстояний до них) в [51] для одного ОО строят несколько сфер с различным радиусом, а в MVP-tree [51] используют несколько ОО в каждом узле, что увеличивает количество областей на уровне. В [52] предлагаются VP-tree с одним ОО на уровне и с несколькими ОО и возможностью многократного применения каждого, а также алгоритм запроса kNN и динамическое конструирование.

Чтобы избежать обратного прослеживания (см. подразд. 1.5) в [53] предложено сферическое разбиение на три области с одним центром и двумя радиусами, которые зависят от радиуса запроса. Объекты двух сепарабельных областей (ближней и дальней от центра) рекурсивно подвергаются дальнейшему разбиению в текущем дереве, а оставшиеся после его построения объекты (из кольцевых средних частей) используются для конструирования следующих деревьев. Для запроса rNN с ограниченным радиусом в каждом дереве не проводится обратного прослеживания, однако требуется обработка множества (леса) деревьев, их число мало только при малом r.

2.2.2. Метрические деревья на основе обобщенных гиперплоскостей. В [48] строят ИС GH-tree иерархическим разбиением множеств объектов базы в узлах на два подмножества посредством обобщенных гиперплоскостей (см. подразд. 1.3) при $\eta = 0$. Дерево получается несбалансированное. Время построения составляет $O(N \log N)$, а затраты памяти $O(N)$. Запросы выполняют методом B&B с гиперплоскостным исключением (3).

В ИС BS-tree [54] условия исключения GH-tree усилены введением r_{cov} для поддеревьев. В ИС V-tree [55] улучшают BS-tree за счет использования двух или трех ОО в узле. Для уменьшения количества ОО и вычисляемых расстояний один из ОО выбирается из предшествующего уровня [56].

2.2.3. Метрические деревья с запоминанием расстояний и дополнительными опорными объектами. В некоторых метрических деревьях в дополнение к областям разбиения запоминают некоторые вычисленные при построении индекса расстояния (например, от ОО до некоторых объектов базы или до других ОО). Это позволяет осуществлять более эффективное исключение областей и индивидуальных объектов. В дополнение к сферическим областям MVP-tree [51, 52] запоминает расстояния объектов в листьях до их ОО верхних уровней.

В ИС M-tree [57] используют иерархию сферических областей вокруг нескольких ОО в узле (с их r_{cov}) и запоминают расстояния между ОО (или объектом базы для листового узла) и его родителем. Построение M-tree осуществляется вставкой новых объектов в «лучшее» поддерево (т.е. без изменения или с минимальным увеличением r_{cov}). Поддерживается сбалансированность дерева: при переполнении листа инициируется процесс его расщепления и перепостроения части дерева. Варианты приближенного поиска по сходству с M-tree рассмотрены в [58]. В ИС PM-tree [59] область задается пересечением гиперсферы и набора гиперколец с центрами в глобальных ОО, что позволяет более плотно ограничивать подмножества объектов. Для уменьшения перекрытия областей M-tree (и количества посещаемых областей при поиске) ИС MX-tree [60] использует условное разбиение с помощью суперузлов; кроме того, сокращены затраты на перепостроение дерева при расщеплении, а множество объектов в каждом листе дополнительно индексируется своим VP-tree, расположенным в свободной области памяти MX-tree.

В ИС GNAT [61] объединяют идеи GH-tree, MVP-tree, BS-tree, V-tree: используют иерархию областей Дирихле, несколько ОО в узле, сохранение предвычисленных расстояний от каждой ОО до самого ближнего и дальнего объекта каждой из m областей уровня. За счет затрат памяти $O(mN)$ и сложности построения $O(mN \log N)$ обеспечивается ускорение поиска по сравнению с VP-tree и GH-tree на ряде баз [61]. Модификация GNAT для снижения затрат памяти и повышения скорости поиска предложена в [62]. Динамический вариант GNAT представлен в [63].

Повышение эффективности исключения объектов в работе [64] основано на наблюдении, что вероятность исключения объекта увеличивается с глубиной его уровня в дереве. Для уменьшения количества ОО на верхних уровнях гиперплоскостных деревьев (GH-tree, BS-tree) во всех узлах одного уровня используют одну и ту же пару ОО с разной балансировкой η (3); для исключения объектов в листьях применяют LAESA с этими же глобальными ОО.

2.2.4. Леса метрических деревьев. Для быстрого приближенного поиска по сходству векторов успешно применяют лес (множество) независимых рандомизированных KD-tree [36]. Аналогичная ИС PF с использованием множества рандомизированных VP-tree предложена в [65]. Для построения каждого дерева случайно выбирают подмножество объектов из множества, подлежащего разбиению (вначале это вся база), и один ОО из подмножества. Вычисляют медианное расстояние подмножества объектов от ОО и используют его в качестве радиуса сферы, процесс повторяют, пока число объектов во множествах не станет меньше порогового. При поиске приближенных kNN в каждом дереве спуском из корня находят лист, области которого принадлежит объект-запрос, и в нем k ближайших объектов-кандидатов (т.е. без обратного прослеживания). Из кандидатов всех деревьев выбирают k лучших.

На нескольких базах векторных данных показано превосходство PF (по проценту найденных точных kNN) над такими известными ИС для векторов, как лес

KD-tree и иерархические K-means [36], однако сравнение времени поиска не приводится. В описании алгоритма не используется неравенство треугольника, поэтому не исключено его применение для поиска по неметрическим расстояниям (разд. 3).

2.3. Поиск по метрическим расстояниям на основе древовидных индексных структур для скалярных и векторных данных. Расстояния объекта до множества ОО представляют собой вещественный вектор (набор одномерных значений), см. подразд. 1.4. Для одномерных индексных значений существуют эффективные структуры поиска (см. подразд. 1.1), а также их аналоги с использованием деревьев, такие как ИС B+tree [66]. Для поиска по сходству вещественных векторов малой размерности также существуют эффективные ИС [14, 25], которые можно применять для полученных скалярных и векторных индексных значений.

В работе [31] каждый компонент полученных векторов используют как индексное значение объектов в отдельном B+tree. При выполнении запроса итоговый список кандидатов — это пересечение множеств объектов-кандидатов от каждого B+tree. Таким образом, получают вариант поиска LAESA с применением леса B+tree. Для индексирования векторов в целом (а не отдельных компонентов) в [31] используют ИС R-tree [25].

В ИС M-index [67] применяют области Дирихле (см. подразд 1.3) множества глобальных ОО. Области с большим количеством объектов рекурсивно разбивают на подобласти с использованием ОО, не задействованных для разбиения на верхних уровнях. Объекты листовой области Дирихле дополнительно заключены в кольцо с центром в их ОО верхнего уровня иерархии. Для получения одномерного индексного значения объекта его расстояние до ближайшего ОО суммируют с индексным значением области объекта нижнего уровня иерархии. Все объекты базы запоминают в одном B+tree. С объектом также ассоциированы расстояния до его ОО верхних уровней. Таким образом, M-index можно отнести к ИС, рассмотренным в подразд. 2.2.3. Более компактные области (cut-regions) [68] по аналогии с PM-tree (см. подразд. 2.2.3) позволяют повысить эффективность исключения. Для быстрого приближенного поиска kNN применяют приоритетную очередь и ранний останов.

В ИС SPB-tree [69] векторы с компонентами, которые являются расстояниями до глобальных ОО, преобразуют в одномерные целочисленные индексные значения посредством заполняющих пространство кривых (space filling curves, SFC) [25]. (Большей части последовательных чисел на SFC соответствуют соседние ячейки векторного пространства.) Листовые узлы содержат SFC-значения объектов узла, а нелистовые — информацию об иерархии областей. Области являются минимальными ограничивающими гиперпрямоугольниками (МОП) объектов множества и задаются парой вершин (векторов), которым соответствуют SFC-значения. Для запоминания всех SFC-значений, а также ассоциированной с ними информации используют B+tree. При выполнении запроса преобразуют объект-запрос в векторное представление, определяют гиперпрямоугольную область запроса и исключают МОП-области, не имеющие с ней пересечения. Далее проводят верификацию объектов-кандидатов оставшихся областей.

Ряд других методов формирования векторных представлений объектов на основе их расстояний до ОО (FastMap, MetricMap, SparseMap, см. обзор в [22]) не гарантируют сжатия расстояний или величины их искажения, поэтому их можно применять только для приближенного поиска по сходству без количественных гарантий.

2.4. Деревья соседства для поиска по метрическим расстояниям. В некоторых древовидных ИС каждый узел соответствует объекту базы и связан с множеством своих соседей на следующем уровне иерархии. Поиск проводится расширением множества кандидатов от узлов-родителей к узлам-сыновьям и исключением (с использованием неравенства треугольника) тех из них, которые не могут привести к ответу на запрос. Назовем такие ИС деревьями соседства.

При построении ИС SAT [70] корнем a является случайный объект базы. Его соседи $\mathrm{nbr}(a)$ выбираются следующей эвристикой. Объекты базы сортируют по возрастанию расстояния до a и последовательно добавляют в (первоначально пустое) множество $\mathrm{nbr}(a)$ те объекты, которые ближе к a, чем к любому объекту из $\mathrm{nbr}(a)$. Оставшиеся объекты базы включают в список того объекта из $\mathrm{nbr}(a)$, к которому они ближе. Объекты $\mathrm{nbr}(a)$ становятся узлами следующего уровня дерева, и процедура повторяется для каждого из них. Все узлы также хранят r_{cov} объектов своего поддерева. Затраты памяти составляют $O(N)$.

Если объект-запрос является объектом базы, такая ИС находит его последовательностью переходов на узел-сосед (сын), ближайший к объекту-запросу. При выполнении запроса rNN переходят из корня не на один, а на некоторое подмножество его соседей-сыновей. Вначале переходят на узел-сосед, ближайший к объекту-запросу, и исключают по (3) те узлы-соседи корня, области которых не могут содержать ответ на запрос. Также для исключения по (1) используют r_{cov} узлов-соседей. Для оставшегося подмножества узлов-соседей корня рекурсивно повторяют вариант процедуры исключения на следующем уровне. Запрос kNN выполняется с использованием приоритетной очереди, время поиска $N^{1-\Theta(1/\log\log N)}$.

В динамических вариантах SAT дерево строят последовательной вставкой новых объектов (как в ИС DSAT [71]) или их кластеров [72]. Комбинация DSAT и DLC (подразд. 2.5) предложена в [73]. В экспериментах DSAT обеспечивают более быстрое построение и поиск, чем SAT. Анализ такого поведения привел к появлению ИС DiSAT [74], где в $\mathrm{nbr}(a)$ вначале добавляют не ближайшие, а дальние объекты. Время поиска DiSAT приблизительно такое же, как у LC (подразд. 2.5), однако при меньшем времени построения.

Быстро работающим на практике метрическим деревом с затратами памяти $O(N)$ является C-tree [75]. Иерархия уровней содержит (например, выбранный случайно) корневой объект на верхнем уровне и все объекты на нижнем. Объект, появившийся на некотором уровне, принадлежит и всем уровням ниже. Каждый объект уровня $i-1$ имеет одного родителя на соседнем верхнем уровне i с расстоянием не более 2^i. Для всех объектов одного уровня расстояние превышает 2^i. Таким образом, соседи-сыновья узлов уровня заключены в непересекающиеся сферические области; уровень корневого узла выбран таким, что его область включает все объекты базы. Возможно статическое и динамическое построение C-tree.

При запросе NN, стартуя с корня, на каждом уровне переходят на узлы (в порядке возрастания расстояний до объекта-запроса x) с расстоянием от x, не превышающем $2^i + \mathrm{dist}\,(x, z)$, где z — текущий NN. Поддерживаются также запросы kNN, rNN и приближенный поиск. Для запроса точного NN и данных с константой расширения $C_e > 2$ (expansion constant), логарифм которой соответствует внутренней размерности данных, в [75] получено время $O(C_e^{12} \log N)$ в худшем случае. Однако в [76] рассмотрены проблемы в доказательстве этого результата.

2.5. Индексные структуры с неиерархическим разбиением на области. Неиерархической ИС, где сферическая область первого ОО содержит m ближайших к ОО объектов базы и r_{cov} области, является ИС LC [77]. Многократное повторение такого разбиения для оставшихся объектов дает $N/(m+1)$ областей-корзин. Структуру LC можно рассматривать как несбалансированное дерево, где только одна из двух ветвей уровня подвергается дальнейшему ветвлению. При выполнении запроса последовательно проверяют условие пересечения области запроса и каждой области LC. Если область запроса принадлежит области LC, поиск останавливают. Эмпирически LC — одна из самых быстрых ИС точного поиска с затратами памяти $O(N)$. Недостатками являются время построе-

ния $O(N^2)$ и необходимость подбора m. Динамический вариант LC (DLC) приведен в [73]. Вариант LC с компактными областями cut-regions предложен в [68]. Сочетание LC с методами подразд. 3.1.3. см. в [78]. Приближенный быстрый поиск выполняется обходом областей в порядке возрастания расстояний до объекта-запроса с ранним остановом [79].

Динамическую многоуровневую ИС D-index [80], как и LC, можно считать неиерархической. На каждом уровне текущую область разбивают на ограниченные сегментами сфер сепарабельные области (см. подразд. 2.2.1) и одну остаточную область, которая становится текущей на следующем уровне. В отличие от [53] на каждом уровне имеется несколько сепарабельных областей с различными центрами, которые являются корзинами. При запросе rNN с ограниченным r переход осуществляется максимум в одну область уровня (и в остаточную область). Возможен и поиск с большим значением r. Индивидуальные объекты исключают по (4), используя ОО уровня.

Неиерархическая ИС RBC [81] позволяет выполнять как точный поиск kNN (с анализом среднего времени), так и приближенный поиск со временем $O(N^{1/2})$ (и с анализом вероятности). Анализ проведен для данных с известной C_e. Для точного поиска каждый (случайный) ОО хранит объекты базы из его области Дирихле и r_{cov}. Запрос выполняется исключением областей вариантами неравенства треугольника и линейным поиском среди оставшихся кандидатов. Для вероятностного поиска ОО хранит заданное число ближайших соседей базы (списки могут перекрываться). При выполнении запроса находят ближайший ОО и из его списка выбирают k ближайших соседей объекта-запроса. Неравенство треугольника применяется только для анализа. Обе ИС позволяют параллельную реализацию.

Различные варианты неиерархических ИС из семейства Singleton [82] конструируют из «блоков», представляющих собой области, ограниченные сферами и гиперплоскостями, а также индексирование объектов расстояниями до ОО, и из различных комбинаций блоков. Задача состоит в создании ИС, обеспечивающей минимальное время поиска. Чем больше блоков в ИС, тем больше времени занимает исключение. Однако количество оставшихся объектов-кандидатов уменьшается и для их верификации линейным поиском требуется меньше времени. Предполагается, что имеется глобальный минимум зависимости сложности поиска от числа блоков. Для базы инкрементно, добавлением блоков, конструируют ИС. Настройка оптимальных параметров выполняется автоматически, стоимость поиска оценивают с использованием модели сложности поиска и эмпирически полученного времени выполнения операции. Скорость поиска ИС семейства приблизительно такая же или в два раза больше, чем у самых быстрых структур (LC, EPT, VP-tree в зависимости от базы) при меньшем в 10 раз времени конструирования.

2.6. Выбор опорных объектов. Эффективность исключения (discarding power) объектов и областей и соответствующее ускорение поиска по сходству значительно зависят от используемых ОО [41, 49, 51, 61]. Последние выбирают из объектов базы. Даже если имеется критерий оценки качества подмножества ОО, комбинаторный перебор всех подмножеств для выбора наилучшего практически не осуществим, поэтому остается рассчитывать на субоптимальные жадные инкрементные эвристики.

Во многих случаях достаточно хорошими ОО являются случайно выбранные объекты базы. Другой хорошей эвристикой является использование в качестве ОО объектов, удаленных от других объектов базы и от других ОО (выбросы, outliers).

В алгоритме FFT [83, 84], стартуя со случайного объекта, в качестве ОО используют самый удаленный от него, затем на каждом шаге выбирают объект, максимально удаленный от множества уже отобранных ОО. В работах [51, 52]

FFT применяют для выбора ОО в VP-tree. В [85] очередной ОО максимизирует сумму расстояний до текущего множества ОО. В алгоритме HOF [31] выбирают объекты базы, близкие к «граничным». Для этого вычисляют расстояние между каждым объектом (не ОО) и всеми ОО из текущего множества. Во множество ОО заносят объект, который минимизирует сумму абсолютного значения разностей вычисленных расстояний и расстояния между первой парой ОО.

В работе [85] экспериментально показано, что, хотя хорошие ОО являются объектами-выбросами, не все выбросы — хорошие ОО; выбросы следует рассматривать лишь как хорошие кандидаты для выбора ОО. В [33] компоненты векторного представления объекта базы (см. подразд. 1.4) получают как расстояния до объектов-выбросов, найденных FFT. Из векторов объектов базы формируют матрицу, по которой находят главные оси PCA ковариационной матрицы, в качестве ОО выбирают объекты-выбросы с максимальной проекцией на главные оси PCA. В работе [69] из объектов-выбросов, полученных алгоритмом HOF [31], инкрементно выбирают в качестве ОО те, которые максимизируют среднее значение отношений расстояний L_∞ между векторами объектов базы и исходных расстояний.

Качество двух множеств ОО одинаковой мощности в [85] предложено сравнивать по величине среднего расстояния L_∞ между векторными представлениями подмножества объектов базы в пространстве расстояний до ОО из сравниваемых множеств (чем больше среднее расстояние, тем больше вероятность исключения). Инкрементный алгоритм получения заданного количества ОО выбирает очередной ОО из случайных подмножеств объектов базы как объект очередного подмножества, который максимизирует среднее расстояние.

В работе [86] новый объект выбирается опорным, если его расстояние до любого имеющегося ОО больше, чем заданный процент максимального расстояния между объектами базы. Это обеспечивает возможность автоматического выбора количества ОО и работы с «растущими» базами. В [87] очередным ОО выбирается объект, обеспечивающий малую (ниже порогового значения) дисперсию разности расстояний между объектами, соседними в упорядоченном списке их расстояний до ОО. Для устранения избыточности ОО оставляют только те из них, для которых значения всех парных корреляций векторов расстояний до объектов базы ниже порогового. В [88] используют инкрементный алгоритм выбора ОО с минимальной суммарной корреляцией с ОО из уже отобранного множества. Корреляцию двух ОО оценивают как среднее геометрическое или гармоническое двух собственных значений ковариационной матрицы векторов расстояний до них объектов базы.

Непосредственно оценить исключающую способность ОО и их множеств позволяет количество исключенных объектов базы, экспериментально полученное на заданном множестве запросов. Это можно использовать для жадных алгоритмов выбора ОО [89–91]. Для снижения вычислительной сложности предлагаются различные стратегии сэмплирования объектов.

Для оценки вероятности исключения объекта базы опорным объектом предложены модели [92, 47], использующие характеристики распределения расстояний между объектами базы [92], а также между ОО и объектами базы [47], которые можно получить на подмножестве базы. Анализ вероятностей исключения показал [92] их большую вариабельность для различных ОО множества. Поэтому для экономии памяти имеет смысл хранить расстояния объекта не до всех ОО, а только до (небольшого) подмножества наиболее эффективных ОО [92, 47]. Обычно это самые дальние либо самые ближние ОО [92, 47].

В EPT [47] (см. подразд. 2.1) формируют несколько групп ОО. Если количества ОО в группе и групп ОО заданы, в каждой группе объекту базы назначают тот единственный ОО из кандидатов, расстояние до которого больше других отлича-

ется от среднего. Это обеспечивает покрытие всех объектов. Чем больше ОО в группе, тем больше вероятность исключения. Оценка дисперсии расстояний между объектами базы, а также между ОО и объектами базы позволяет оценить вероятность исключения. В сочетании с моделью стоимости поиска это дает возможность инкрементно выбрать количество ОО в группе, минимизирующее время выполнения запроса (для заданных базы, количества групп ОО, радиуса запроса).

2.7. Исключение без использования неравенства треугольника. Во всех рассмотренных ИС исключение областей и объектов выполнялось вариантами неравенства треугольника. Как упоминалось в разд. 1.3 и 1.4, для поиска по B&B можно использовать и другие границы.

В [93, 94] рассмотрено семейство границ на основе неравенства Птолемея $\text{dist}(x, v)\,\text{dist}(y, u) \le \text{dist}(x, y)\,\text{dist}(u, v) + \text{dist}(x, u)\,\text{dist}(y, v)$. В общем случае из неравенства Птолемея не следует неравенство треугольника, и наоборот. Для векторных пространств неравенство Птолемея применимо к метрическим расстояниям квадратичной формы $((\mathbf{x}-\mathbf{y})^{\mathrm{T}}\,\mathbf{A}(\mathbf{x}-\mathbf{y}))^{1/2}$, где $\mathbf{A}$ — положительно определенная матрица, включая евклидово. Для любого метрического пространства с dist пространство с $\text{dist}^{1/2}$ удовлетворят неравенству Птолемея. Для расстояний, к которым применимо неравенство Птолемея, оно обеспечивает более эффективное исключение, чем варианты неравенства треугольника. Для метрических птолемических расстояний сочетание двух исключений дает лучшие результаты, чем их отдельное использование. Ряд расстояний близок к птолемическим, что можно использовать для приближенного поиска. В [93] рассмотрены птолемические LAESA, PM-tree, M-index.

Многие метрические пространства имеют свойство четырех точек (the four-point property): любые четыре объекта можно изометрически вложить в трехмерное евклидово пространство [95]. Это свойство присуще любому метрическому пространству, которое можно изометрически вложить в гильбертово (матрица квадратов расстояний должна быть условно отрицательно полуопределенной). К таким «суперметрическим» [95] пространствам относятся, например, векторные с расстояниями Евклида, Йенсена–Шеннона, треугольника, косинусным. Важные пространства без этого свойства включают метрики Манхэттена, Чебышева, Левенштейна. Однако для любой метрики dist пространство с $\text{dist}^{1/2}$ является суперметрическим.

Если суперметрическое пространство разбито на две части гиперплоскостью (см. подразд.1.3), область o_j можно исключить, если $(\text{dist}^2(o_j, x) - \text{dist}^2(o_i, x))/\text{dist}(o_i, o_j) > 2r$ (гильбертово исключение). Это условие позволяет исключать больше объектов, чем (3). Если объекты изобразить на плоскости так, чтобы их расстояния L_2 до ОО совпадали с исходными расстояниями $\text{dist}(y, o_i)$, $\text{dist}(y, o_j)$, то расстояние L_2 между любой парой объектов на плоскости является нижней границей их исходного расстояния $\text{dist}(x, y)$. В экспериментах вариант GH-tree (монотонное BS-tree) работает быстрее DiSAT (оба с гильбертовым исключением). Традиционное исключение (3) для этих деревьев работает медленнее.

3. ИНДЕКСНЫЕ СТРУКТУРЫ ДЛЯ ПОИСКА ПО НЕМЕТРИЧЕСКИМ РАССТОЯНИЯМ

В некоторых задачах поиска по сходству (см. [32]) функция расстояния не является метрикой: для нее не выполняются метрические аксиомы, особенно важно невыполнение неравенства треугольника. Поэтому для точного поиска по неметрическим расстояниям (или мерам несходства) нельзя непосредственно использовать ИС, представленные в разд. 2. Не являются метриками меры сходства, а также многие полученные из них меры несходства.

Рассмотрим ИС для ускорения поиска по неметрическим расстояниям. В отличие от ИС, описанных в разд. 2, многие ИС, представленные в разд. 3, не обеспечивают точного поиска, находят ближайших соседей с некоторой вероятностью, часто без ее количественных оценок. В то же время на практике ряд ИС работает быстро и демонстрирует хорошую точность поиска

3.1. Поиск с применением индексных структур для метрических расстояний и векторов.

3.1.1. Использование ИС для метрических расстояний. В ИС QIC-M-tree [96] для поиска по неметрическому расстоянию предлагают конструировать ИС для ограничивающего его снизу метрического расстояния, которое и используют для исключения областей ИС при выполнении запроса. Этот подход требует разработки сжимающего метрического расстояния для каждой конкретной неметрики, что не всегда возможно.

В ИС LCE [97] базу разбивают на непересекающиеся подмножества, для каждого из которых вводится свое метрическое расстояние (путем сложения исходного неметрического расстояния с выбранной для каждого подмножества константой), что позволяет эффективно исключать объекты каждого подмножества с применением метрической ИС (в [97] используют ИС на основе B+tree). Однако LCE требует получения матрицы расстояний между объектами базы и гарантирует точный поиск только для объектов-запросов из базы.

Алгоритм TriGen [32] позволяет изменять процент троек объектов базы, для которых выполняется неравенство треугольника. При увеличении этого процента, однако, растет внутренняя размерность базы и снижается скорость поиска, что приводит к необходимости подбора параметров для конкретных баз и не гарантирует точного поиска для новых объектов. В ИС NM-tree [98] используют МТ для расстояний, преобразованных в метрические посредством TriGen. Компромисс скорость/точность поиска регулируется при выполнении конкретных запросов.

Для выполнения запроса rNN по неметрической дивергенции Брегмана в [20] использован вариант сферического дерева и метод B&B с предложенными условиями включения и пересечения сферических областей Брегмана.

Для поиска (точного и приближенного) по максимальному значению сходства, задаваемого ядерной функцией, значения которой соответствуют скалярному произведению векторных представлений объектов в некотором гильбертовом пространстве, в [99] индуцированное ядерной функцией расстояние используют для конструирования C-tree (см. подразд. 2.4). Для поиска применяют метод B&B с предложенной верхней границей значения ядра между объектом-запросом и любым объектом в поддереве.

3.1.2. Использование ИС для векторов. Методы формирования векторных представлений объектов на основе их расстояний до ОО можно применять к неметрическим расстояниям, что позволяет использовать ИС, предназначенные для поиска векторов [25, 14]. Однако для такого преобразования (см. подразд. 1.4) в этом случае уже не гарантируется сжатия расстояний. Ряд других методов формирования векторных представлений объектов на основе их расстояний до ОО (FastMap, MetricMap, SparseMap [22]) не гарантируют не только сжатия расстояний, но и значений их искажения. Это также относится к алгоритмам формирования векторных представлений с применением обучения (см. ссылки в [32]). Поэтому качество результатов поиска по полученным векторам можно оценить только экспериментально.

Для некоторых конкретных неметрических расстояний/сходств можно получить плотные границы исходных расстояний через расстояния между полученными векторами. Например, для поиска по неметрическому расстоянию dynamic

time warping между временными рядами в [100] формируют векторные представления и используют плотную границу исходного расстояния для эффективного точного поиска в ИС для векторов на основе R-tree. Использование векторных ИС для поиска по дивергенции Брегмана см. в [101, 102].

3.1.3. Представления на основе ближайших опорных объектов. В работах [103–109] формируют «сигнатуры» объектов на основе информации об их ближайших ОО. Расстояния/сходства сигнатур в некоторой степени соответствуют исходным расстояниям/сходствам объектов. При выполнении запроса получают сигнатуру объекта-запроса и рассматривают в качестве кандидатов только те объекты базы, сигнатуры которых близки к сигнатуре объекта-запроса. Ускорение поиска достигается за счет более эффективного поиска сходных сигнатур, чем линейный поиск по исходным мерам расстояния/сходства объектов. Сигнатура объекта может включать различную информацию (например, множество ближайших ОО объекта, ранги их сходств, значения сходств и др.). По-разному можно определять меры сходства между сигнатурами, их параметры и алгоритмы обработки сигнатур для выполнения запросов, включая применение ИС для поиска сходных векторов и множеств. Поиск не имеет гарантий точности, не использует в явном виде неравенства треугольника для исходных расстояний и его можно применять как для метрических, так и для неметрических расстояний/сходств [110]. Алгоритмы показывают хорошие результаты в экспериментах и обобщены в [109].

В работах [103, 104] сигнатура объекта y является вектором, компоненту вектора соответствует ОО из фиксированного множества, значение компонента равно номеру ОО в списке ОО, отсортированном по величине расстояния от y. Расстояние между сигнатурами определяется по L_1 или L_2 и используется для отбора кандидатов линейным поиском [103].

Бинарный вектор в [105] получают сравнением с порогом разности значения и номера компонента вектора. Далее выполняют линейный поиск по расстоянию Хэмминга между бинарными векторами или применяют поиск LSH (подразд. 3.4).

В ИС MIF [103] для y запоминают только K ближайших ОО. Для сравнения сигнатур используют модификацию расстояния L_1, где при отсутствии соответствующего ОО к значению расстояния добавляется штраф. Ускорения сравнения сигнатур запроса и объектов базы достигают применением обратного индекса, в котором для каждого ОО запоминают список объектов базы с их рангами. Индексирование префиксным деревом предложено в [107]. Оценивать сходство объектов по проценту их общих ближайших ОО (без учета их ранга) предложено в алгоритме NAPP [108]. Поиск ускоряют за счет применения эффективного обратного индекса. В [109] представлены и другие варианты реализации этого подхода.

3.2. Поиск на основе сравнений. Этот подход, также известный как «комбинаторный» [111], использует при поиске по сходству только информацию о том, какой из двух объектов ближе к третьему, что позволяет ранжировать объекты. Неравенство треугольника не используется. Подход можно применять как для метрических, так и для неметрических расстояний/сходств. Отметим, что многие сигнатуры (см. подразд. 3.1.3) тоже формируются на основе ранжирования.

В работе [111] предложен аналог неравенства треугольника для рангов сходств с использованием константы беспорядка C_d (disorder constant), $\log C_d + 1$ является разновидностью внутренней размерности: $\text{rank}_y(x) \le C_d(\text{rank}_z(x) + \text{rank}_z(y))$, где $\text{rank}_y(x)$ — положение объекта x в списке объектов базы, отсортированных по сходству с y.

Практический алгоритм RanWalk [111] для поиска NN использует случайные блуждания. При построении для каждого объекта базы сортируют по сходству с ним все остальные объекты и запоминают. При выполнении запроса, стар-

туя со случайного объекта, случайно выбирают $3C_d(\log\log N+1)$ объектов из его «окрестности» с заданным числом объектов. Вычисляют их расстояния до объекта-запроса и переходят на ближайший. Повторяют такой шаг $\log N$ раз с уменьшением окрестности на каждом шаге и оказываются в объекте y. Если ранг объекта-запроса по отношению к y больше C_d, процедуру повторяют, иначе выполняют поиск в окрестности y с числом объектов C_d^2. Точный NN возвращается RanWalk за среднее время $O(C_d \log N \log\log N + C_d^2)$, однако требует квадратичных времени построения и памяти.

Нижняя граница среднего времени поиска NN в рандомизированных алгоритмах комбинаторного подхода и иерархическая ИС с использованием сэмплирования для поиска NN и rNN приведены в [113]. Почти линейным временем построения и затратами памяти и почти логарифмическим временем запроса точного NN отличаются детерминированные combinatorial nets [112], где строится иерархическая структура покрытий объектов с экспоненциально убывающим радиусом. Однако алгоритмы [112, 113] не реализованы.

Практической ИС на основе сравнений является SASH [114]. При конструировании случайно выбирается половина объектов текущего множества (вначале все объекты базы), которые становятся узлами на текущем уровне иерархии. Процедура повторяется, пока не останется один корневой объект. Сходные объекты соседних уровней соединяют ребрами. Каждый узел (кроме корня) может соединяться с несколькими родителями, поэтому SASH является графом, а не деревом. Для поиска kNN, стартуя с корня, переходят по ребрам на ближайшие к объекту-запросу узлы следующего уровня (количество узлов определяется квотой уровня). Эти узлы являются кандидатами, среди которых выбирают k ближайших к запросу объектов. В отличие от деревьев соседства (см. подразд. 2.4) неравенство треугольника не используется. Нахождение точных kNN не гарантируется, но часто наблюдается на практике.

К комбинаторному подходу относится практический алгоритм RC-tree [115] со структурой, подобной C-tree и SASH. При поиске переход осуществляется на узлы, наиболее сходные с объектом-запросом (по рангу), число которых задается квотой уровня. Анализ показывает, что RC-tree с высокой вероятностью возвращает точные kNN при времени выполнения запроса с меньшей степенью полиномиальной зависимости от C_e, чем в CT, и сублинейной зависимостью от N. Экспериментально RC-tree быстрее C-tree и SASH на большинстве исследованных баз. К поиску на основе сравнений относятся также ИС, описанные в подразд. 3.3.

3.3. Индексные структуры на основе графов соседства. В графе соседства (proximity graph или neighborhood graph) объекты базы соответствуют узлам, а ориентированные ребра направлены к «соседним» (в разном смысле) объектам. Алгоритм жадного поиска стартует со случайного узла и ранжирует его соседей относительно объекта-запроса x. Если текущий узел ближе к x, чем все соседи узла («локальный минимум»), то он и является ответом на запрос. В противном случае переходят к ближайшему к x соседу [70, 13].

Простейшим графом, для которого такой алгоритм жадного поиска находит точного ближайшего соседа, является полносвязный граф, однако в каждом узле уже требуется линейный поиск по всей базе. Для векторных пространств граф с минимальным числом ребер, позволяющий жадно находить NN, — это граф Делоне (Delaunay), но для общих метрических расстояний его нельзя построить по матрице расстояний между объектами [70]. Однако для поиска приближенного NN не требуется точного графа Делоне, поэтому на практике используют графы, подграфы которых являются его аппроксимацией.

Алгоритм для соединения каждого объекта с соседями в [116] такой же, как для корня SAT ([70] и см. подразд. 2.4), и жадный поиск находит объект базы, совпадающий с объектом-запросом x при старте из произвольного узла. Для x не из базы этого не гарантируется и применяют поиск best-first. В [117] в качестве аппроксимации графа Делоне предложено использовать граф KNN, в котором каждый узел имеет ориентированные связи с K ближайшими соседями ($K \neq k$ в общем случае). Чем меньше степень узлов K, тем меньше число посещаемых на каждом шаге узлов, однако уменьшается и вероятность нахождения точного NN. Усилия по развитию этого подхода (подразд. 3.3.1 и 3.3.2) направлены на повышение скорости поиска и качества найденных соседей, уменьшение сложности построения графа KNN.

3.3.1. Поиск ближайших соседей. Для точных запросов rNN и kNN в [118] применяют модификацию графа KNN, в которой запомнены расстояния на ребрах, а также r_{cov} узлов. В сочетании с неравенством треугольника это используется для исключения объектов и выбора для обработки непосещенных узлов. Ускорение достигается для данных малой размерности.

В других работах не гарантируется нахождения точных kNN. В [119] для поиска одного NN применяют однократный спуск жадного алгоритма в локальный минимум, стартуя со случайно выбранного узла; расстояния вычисляют до $E < K$ соседей узла в графе ($K = 1000$). Для поиска kNN предусмотрен рестарт из нескольких случайных узлов, останов поиска — по заданному числу жадных переходов на ближайший к объекту-запросу x узел из E соседей текущего узла (без проверки на локальный минимум). Несколько рестартов применяют в [117, 120, 121]; объекты, полученные жадным поиском локального минимума от всех рестартов, сортируют и выбирают k наилучших. В [121] останов выполняют при отсутствии изменения в списке текущих kNN; чтобы избежать дублирования, используют единый список непосещенных узлов для всех рестартов.

Многие алгоритмы поиска по графам соседства используют для обхода приоритетную очередь непосещенных узлов, упорядоченных по расстоянию до объекта-запроса (см. подразд. 1.5). На каждом шаге извлекают лучший узел и помещают в очередь его непосещенных соседей по графу. Вместо останова в локальном минимуме из очереди извлекают следующий узел и исследуют его соседей (аналог обратного прослеживания).

В очередь заносят узлы, получаемые обходом графа из одного узла (например, [122]) или от нескольких рестартов (например, [117]). В [117] в очередь заносят узлы из «расширенной окрестности» текущего узла, т.е. связанные с ним через узлы, расстояние которых до объекта-запроса не превышает порогового. Останов поиска выполняют по количеству вычисленных расстояний [122] или по соотношению расстояний до ближайшего узла и до текущего NN [117].

3.3.2. Конструирование графов KNN и их аппроксимаций. Конструирование точного графа KNN в общем случае имеет сложность $O(N^2)$, что неприемлемо на практике. При метрических расстояниях ускорения можно достичь применением ИС для точного поиска (см. разд. 2). В сочетании с оптимизацией за счет совместного выполнения N запросов и использованием уже построенной части графа для получения границ на некоторые расстояния это позволяет снизить эмпирическую сложность конструирования, особенно для данных невысокой размерности [123] (до $O(N^{1.1})$ при $D = 4$ и до $O(N^{1.96})$ при $D = 24$).

Граф приближенных KNN можно сконструировать эффективнее как быстрым поиском приближенных KNN, так и более простыми средствами. Для произвольных мер сходства (неметрических) итеративным улучшением случайно выбранных кандидатов [124] эмпирически получена сложность конструирования, приблизительно равная $O(N^{1.14})$ для баз различной размерности.

Чтобы уменьшить время поиска при сохранении высокой вероятности нахождения точного NN, в [125] удаляют из графа KNN ребра, без которых алгоритм жадного поиска может достичь KNN узла по существующим ребрам. В [122] аппроксимируют минимальный граф, который позволяет жадно находить NN для объектов-запросов из базы. Среднее число ребер у таких графов-аппроксимаций невелико для данных с невысокой размерностью.

Для преодоления проблемы возможной потери связности графа KNN используют увеличение K и двунаправленные ребра ([125, 126] и подразд. 3.3.3). Другой недостаток — отсутствие в общем случае свойств, позволяющих находить NN за poly log N шагов на основе локальной информации (жадного поиска), можно преодолеть с помощью графов, описанных в подразд. 3.3.3.

3.3.3. Графы тесного мира. В графах со структурой «тесного мира» (small world graph, SWG) минимальное число ребер в пути между узлами медленно (логарифмически) растет от количества узлов N [127, 128]. Отметим, что для любого графа с постоянной степенью узлов нижняя граница диаметра (величины самого длинного кратчайшего пути между узлами) составляет $\Omega(\log N)$ [129, 130]. Графы SWG имеют свойство навигации (navigability, NSWG), если жадный алгоритм поиска переходит из любого исходного в любой целевой узел за среднее число шагов (время) $O(\text{poly log } N)$. Не все SWG имеют свойство навигации: жадный алгоритм поиска не находит короткого пути, хотя он существует [128, 131]. Для превращения некоторых типов графов в NSWG в [128] предложено дополнять их длинными (long-range) однонаправленными ребрами из определенного распределения. Отметим, что произвольный граф нельзя таким образом превратить в NSWG: нижняя граница количества шагов для дополненного графа субполиномиальная $\Omega(2^{\sqrt{\log N}})$ [132]. Верхняя граница среднего числа шагов $\widetilde{O}(N^{1/3})$ [133] улучшена до $2^{\sqrt{\log N}+o(1)}$ [131].

В [130] показано, что свойство NSWG появляется в широком классе сетей, конструируемых на основе естественных и довольно общих принципов. Отметим, что в графе KNN нет механизма формирования «длинных ребер», соединяющих различные части графа.

Для данных с известной C_d в [112] предложен алгоритм построения visibility graph — варианта NSWG, обеспечивающего жадный точный поиск NN за не более чем log N шагов. Так как степень узлов графа достигает $O(D^4 \log N)$, время поиска $O(D^4 \log^2 N)$, сложность построения $\text{poly}(D) N \log^2 N$. Однако алгоритм не исследован на практике.

Практические алгоритмы [120, 121] конструируют NSWG инкрементно, выполняя запрос KNN со вставляемым объектом в качестве объекта-запроса и связывая его с найденными соседями двунаправленными ребрами. Вставленные в начале конструирования ребра узлов впоследствии становятся «длинными ребрами». Поиск выполняют жадными процедурами (подразд. 3.4.2) с несколькими рестартами из непосещенных узлов. Эмпирически процедуры поиска и вставки имеют полилогарифмическую сложность. В [134] останов поиска выполняют при неулучшении расстояния до запроса в течение нескольких итераций поиска, используют также общую приоритетную очередь изменяемой длины.

Для повышения скорости поиска алгоритма из [121] (особенно в больших базах при низкой размерности данных [135, 110]) иерархический алгоритм HNSW [136] инкрементно строит иерархическую структуру графов соседства различных уровней для вложенных подмножеств объектов базы (на верхнем уровне малое число объектов, на нижнем — все объекты). Максимальный уровень объекта выбирается случайно с экспоненциально убывающей вероятностью.

Поиск стартует с верхнего уровня, жадно находит узел локального минимума и продолжается с этого узла (или с нескольких ближайших к запросу) на нижнем уровне. На самом нижнем уровне используют более длинную приоритетную очередь, чем на остальных, а найденные на нем объекты являются ответом на запрос. Для конструирования HNSW выполняется последовательность процедур вставки объектов. На каждом уровне, начиная с максимального уровня объекта и ниже, вставляемый объект соединяют с K объектами, которые находят модифицированной процедурой поиска.

3.4. Индексные структуры на основе сохраняющего сходство хэширования. В локально-чувствительном хэшировании (locality-sensitive hashing, LSH) в отличие от обычного LSH-функции генерируют хэш-значения так, что вероятность их совпадения у сходных объектов выше, чем у несходных ([139, 16, 17], см. обзор в [11]). Объекты с одинаковым хэшем образуют корзину. При поступлении объекта-запроса генерируют его LSH-значение, извлекают из соответствующей корзины объекты базы и используют их в качестве кандидатов для линейного поиска по исходному расстоянию. Хэш-значения сходных объектов, сгенерированные одной LSH-функцией, могут оказаться различными, поэтому для повышения вероятности нахождения ближайших к запросу объектов-кандидатов при поиске объединяют списки кандидатов из корзин нескольких независимых LSH-функций (аналогично лесам деревьев, см. подразд. 2.2.4).

При надлежащем выборе параметров ИС LSH позволяют гарантировать для худшего случая сублинейное время выполнения некоторых типов запросов приближенного поиска по сходству (с рассчитываемой вероятностью). Семейства LSH-функций с известными характеристиками, позволяющими рассчитать параметры ИС, разработаны для некоторых мер расстояния/сходства векторных представлений; характеристики не зависят от конкретного набора объектов базы.

Для невекторных представлений ИС LSH можно использовать для приближенного поиска с некоторыми гарантиями, если имеются векторные представления, расстояния которых аппроксимируют исходные с известным искажением, и для этих векторов существует LSH-семейство. Например, для расстояния Левенштейна между строками известны вложения в векторы с расстоянием L_1 (см. ссылки в [10]). Использование LSH-структур для приближенного поиска по расстоянию Левенштейтна рассмотрено в [137, 138].

Аналоги LSH-функций для общих метрических и неметрических расстояний используют расстояния до ОО. Однако поиск не имеет строгих гарантий LSH, так как для этих аналогов неизвестны количественные характеристики, позволяющие рассчитать ИС ввиду того, что неизвестна структура пространства, а также вследствие зависимости от выбора ОО. Кроме того, LSH-функции существуют только для метрических расстояний [139].

В работе [140] в качестве LSH-значений используют случайно выбранные компоненты бинарных векторов [105] (см. подразд. 3.1.3). В хэшировании DBH [141] применяют FASTMAP для получения (вектора) вещественных индексных значений объекта. Отдельные значения бинаризуют двумя порогами (значение между порогами дает 0, иначе 1). В экспериментах на базах с неметрическими мерами расстояния получено преимущество над VPT в скорости при той же точности. В [142] для оптимизации точности и времени поиска DBH подстраивают к базе путем выбора ОО, их пар, а также количества хэш-функций.

Алгоритм RBBF [143] использует представление объекта битами, каждый их которых кодирует положение объекта относительно обобщенной гиперплоскости (см. подразд 1.3).

Корзины DFLSH [144] включают объекты базы из своих областей Дирихле со случайно выбранными ОО. В работе [145] ОО и области получают варианта-

ми алгоритмов кластеризации на основе расстояний K-medoids. Для ускорения поиска применяется параллелизация. Полнота поиска kNN достигается на уровне DFLSH, однако время запроса несколько меньше вследствие лучшей сбалансированности корзин.

В работе [146] в качестве областей LSH используют области M-index [67] — «виртуальные корзины», размер которых может изменяться в ходе выполнения запроса.

ЗАКЛЮЧЕНИЕ

Преимущество рассмотренных в обзоре ИС заключается в том, что явно не используются представления объектов как при конструировании индекса, так и при выполнении запросов поиска по сходству. Это позволяет применять такие структуры для поиска по сходству объектов с различными типами представлений. Меры расстояния/сходства, по которым выполняется поиск, также достаточно разнообразны.

Индексные структуры, представленные в разд. 2, работают с мерами расстояния, которые удовлетворяют метрическим аксиомам. При конструировании используется информация о расстоянии объектов базы до некоторых опорных объектов. При выполнении поискового запроса эта информация в сочетании с вариантами неравенства треугольника применяется для быстрого исключения из обработки объектов и множеств объектов базы, которые не могут являться ответом на поисковый запрос (метод ветвей и границ, см. подразд. 1.2), что дает возможность ускорить поиск по сравнению с линейным при сохранении точности результатов. Некоторые метрические расстояния имеют дополнительные свойства, позволяющие исключать из обработки большее количество объектов (см. подразд. 2.7).

Представленные в разд. 2 ИС имеют следующие недостатки. Скорость поиска существенно зависит от выбора опорных объектов, который осуществляется субоптимальными эвристическими алгоритмами (см. подразд. 2.6). Как и для всех ИС точного поиска по сходству, время выполнения запроса экспоненциально зависит от внутренней размерности данных: с ее ростом поиск вырождается в линейный. Ускорение поиска возможно только за счет потери гарантий точности — путем обработки объектов и их множеств, ближайших к объекту-запросу, а также исключения не только тех же объектов, что при точном поиске, но и других, которые могут являться ответом на запрос, но с меньшей вероятностью.

Индексные структуры, рассмотренные в разд. 3, предназначены для поиска по неметрическим мерам расстояния. Так как не существует универсальных, применимых ко всем неметрическим расстояниям условий исключения нерелевантных объектов при выполнении запроса, ИС для точного поиска методом ветвей и границ требуют учета специфических свойств конкретных неметрик или их классов, которые позволяют разработать условия точного исключения, что возможно далеко не всегда (см. подразд. 3.1.1 и 3.1.2).

Для приближенного поиска используют преобразование неметрических расстояний между объектами множества в метрические (см. подразд. 3.1.1), а также формирование векторов или множеств признаков объектов (см. подразд. 3.1.2 и 3.1.3), сходство которых отражает исходное сходство объектов (но без количественных гарантий искажения сходства). Далее используют ИС для быстрого поиска по метрическим расстояниям, а также для векторов и множеств. Однако точный поиск в этих ИС в общем случае не гарантирует получения результатов точного поиска по исходным расстояниям.

Приведенные в подразд. 3.2, 3.3 ИС вместо исключения объектов и областей методом ветвей и границ используют последовательность переходов к объектам базы и их множествам, сходным с объектом-запросом и с текущими кандидатами. Ограничение количества рассматриваемых на каждом шаге объектов-канди-

датов позволяет ускорить выполнение запроса. В ИС, рассмотренных в подразд. 3.4, кандидатами являются объекты базы, имеющие то же значение хэша, что и объект-запрос, причем хэш-значения генерируются так, что велика вероятность совпадения хэшей сходных объектов. Точность результатов при сублинейном времени выполнения запроса в большинстве методов не гарантируется, хотя эмпирически получают хорошие результаты.

Универсальные ИС, разработанные для поиска по общим неметрическим расстояниям/сходствам, можно использовать для поиска по общим метрическим расстояниям, а также по специальным расстояниям (включая различные расстояния/сходства между векторами).

Отметим, что для некоторых типов запросов поиска по сходству объектов, представленных векторами, разработаны ИС, обеспечивающие приближенный поиск с гарантиями сублинейного времени, а также качества результатов для худшего случая, такие как LSH [16, 17] и LSF [147] (см. также подразд. 3.4). Экспериментальное сравнение [136, 148] реализаций ряда из рассмотренных в настоящем обзоре ИС показывает, что в настоящее время наилучшие результаты поиска демонстрирует иерархическая структура на основе графов тесного мира со свойством навигации HNSW (см. подразд. 3.3.3). Для HNSW на больших базах эмпирически получена логарифмическая сложность поиска. Показано преимущество HNSW над многими лучшими ИС для поиска по невекторным данным: VP-tree, NAPP, NSW, граф KNN [124]. Подбор параметров позволяет для некоторых алгоритмов, включая HNSW, на некоторых базах получать точные результаты поиска (за счет увеличения времени выполнения запроса по сравнению с приближенным поиском). Отметим, что для поиска по векторным данным HNSW дает результаты выше, чем лучшие структуры, специализированные для векторных представлений [136].

СПИСОК ЛИТЕРАТУРЫ

1. Datta R., Joshi D., Li J., Wang J. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys*. 2008. Vol. 40, N 2. P. 1–60.
2. Manning C., Raghavan P., Schütze H. Introduction to information retrieval. New York: Cambridge University Press, 2008. 506 p.
3. Duda R.O., Hart P.E., Stork D.G. Pattern classification. 2nd Edition. New York: John Wiley & Sons, 2001. 680 p.
4. Lopez De Mantaras R., Mcsherry D., Bridge D., Leake D., Smyth B., Craw S., Faltings B., Maher M.L., Cox M.T., Forbus K., Keane M., Aamodt A., Watson I.. Retrieval, reuse, revision and retention in case-based reasoning. *Knowledge Engineering Review*. 2005. Vol. 20, N 3. P. 215–240.
5. Voskoglou M.G., Salem A.-B.M. Analogy-based and case-based reasoning: Two sides of the same coin. *IJAFSAI*. 2014. Vol. 4. P. 5–51.
6. Wharton C.M., Holyoak K.J., Downing P.E., Lange T.E., Wickens T.D., Melz E.R. Below the surface: Analogical similarity and retrieval competition reminding. *Cognitive Psychology*. 1994. Vol. 26. P. 64–101.
7. Gentner D., Smith L. Analogical reasoning. In: Encyclopedia of human behavior. V.S. Ramachandran (Ed.) (2nd ed.). Oxford, UK: Elsevier, 2012. Vol. 1. P. 130–136.
8. Rachkovskij D.A., Slipchenko S.V. Similarity-based retrieval with structure-sensitive sparse binary distributed representations. *Computational Intelligence*. 2012. Vol. 28, N 1. P. 106–129.
9. Forbus K., Ferguson R., Lovett A., Gentner D. Extending SME to handle large-scale cognitive modeling. *Cognitive Science*. DOI: 10.1111/cogs.12377.
10. Rachkovskij D.A. Real-valued embeddings and sketches for fast distance and similarity estimation. *Cybernetics and Systems Analysis*. 2016. Vol. 52, N. 6. P. 967–988.
11. Rachkovskij D.A. Binary vectors for fast distance and similarity estimation. *Cybernetics and Systems Analysis*. 2017. Vol. 53, N 1. P. 138–156.
12. Chavez E., Navarro G., Baeza-Yates R., Marroquin J.L. Searching in metric spaces. *ACM Computing Surveys*. 2001. Vol. 33, N 3. P. 273–321.
13. Hjaltason G.R., Samet H. Index-driven similarity search in metric spaces. *ACM Transactions on Database Systems*. 2003. Vol. 28, N 4. P. 517–580.

14. Samet H. Foundations of multidimensional and metric data structures. San Francisco: Morgan Kaufmann, 2006. 1024 p.
15. Zezula P., Amato G., Dohnal V., Batko M. Similarity search: The metric space approach. New York: Springer, 2006. 220 p.
16. Andoni A., Indyk P. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM.* 2008. Vol. 51, N 1. P. 117–122.
17. Andoni A., Indyk P. Nearest neighbors in high-dimensional spaces. In: Handbook of discrete and computational geometry. 3rd edition. 2017 (to appear). Chap. 43.
18. Fukunaga K., Narendra P.M. A branch and bound algorithm for computing k-nearest neighbors. *IEEE Trans. Comput*. 1975. Vol. C-24, N 7. P. 750–753.
19. Lokoc J., Skopal T. On applications of parameterized hyperplane partitioning. *Proc. SISAP'10*. 2010. P. 131–132.
20. Cayton L. Efficient Bregman range search. *Proc. NIPS'09.* 2009. P. 243-251.
21. Connor R., Vadicamo L., Cardillo F.A., Rabitti F. Supermetric search with the four-point property. *Proc. SISAP'16*. 2016. P. 51–64.
22. Hjaltason G.R., Samet H. Properties of embedding methods for similarity searching in metric spaces. *IEEE Trans. PAMI.* 2003. Vol. 25, N 5. P. 530–549.
23. Clarkson K. Nearest-neighbor searching and metric space dimensions. In: Nearest-neighbor methods for learning and vision: Theory and practice. MIT Press, 2006. P. 15–59.
24. Weber R., Schek H.J., Blott S. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. *Proc. VLDB'98*. 1998. P.194–205.
25. Böhm C., Berchtold S., Keim D.A. Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. *ACM Com. Surv*. 2001. Vol. 33, N 3. P. 322–373.
26. Beyer K., Goldstein J., Ramakhrishnan R., Shaft U. When is "nearest neighbor" meaningful? *Proc. ICDT'99*. 1999. P. 217–235.
27. Shaft U., Ramakrishnan R. Theory of nearest neighbors indexability. *ACM Trans. Database Syst*. 2006. Vol. 31. P. 814–838.
28. Volnyansky I., Pestov V. Curse of dimensionality in pivot based indexes. *Proc. SISAP'09*. 2009. P. 39–46.
29. Pestov V. Indexability, concentration, and VC theory. *Journal of Discrete Algorithms*. 2012. Vol. 13. P. 2–18.
30. Camastra F. Data dimensionality estimation methods: a survey. *Pattern Recogn.* 2003. Vol. 6, N 12. P. 2945–2954.
31. Traina C., Santos Filho R.F., Traina A.J.M, Vieira M.R., Faloutsos C. The Omni-family of all-purpose access methods: A simple and effective way to make similarity search more efficient. *VLDB Journal*. 2007. Vol. 16, N 4. P. 483–505.
32. Skopal T., Bustos B. On nonmetric similarity search problems in complex domains. *ACM Comput. Surveys.* 2011. Vol. 43, N 4. P. 34:1–34:50.
33. Mao R., Mirankerb W.L., Mirankerc D.P. Pivot selection: Dimension reduction for distance-based indexing. *J. Discrete Algorithms*. 2012. Vol. 13. P. 32–46.
34. Patella M., Ciaccia P. Approximate similarity search: a multi-faceted problem. *J. Discrete Algorithms.* 2009. Vol. 7, N 1. P. 36–48.
35. Powers D.M.W. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *J. of Machine Learning Tech.* 2011. Vol. 2, N 1. P. 37–63.
36. Muja M., Lowe D.G. Scalable nearest neighbor algorithms for high dimensional data. *IEEE TPAMI.* 2014. Vol. 36, N 11. P. 2227–2240.
37. Navarro G. Analyzing metric space indexes: What for? *Proc. SISAP'09*. 2009. P. 3–10.
38. Vidal E. An algorithm for finding nearest neighbors in (approximately) constant average time. *Patt. Recog. Lett.* 1986. Vol. 4, N3. P. 145–157.
39. Vidal E. New formulation and improvements of the nearest-neighbour approximating and eliminating search algorithm (AESA). *Patt. Recog. Lett*. 1994. Vol. 15, N 1. P. 1–7.
40. Figueroa K., Chavez E., Navarro G., Paredes R. Speeding up spatial approximation search in metric spaces. *ACM Journal of Experimental Algorithmics*. 2009. Vol. 14. P. 3.6.1–3.6.21.
41. Mico L., Oncina J., Vidal E. A new version of the nearest-neighbor approximating and eliminating search (AESA) with linear preprocessing-time and memory requirements. *Patt. Recog. Lett*. 1994. Vol. 15, N 1. P. 9–17.
42. Nene S., Nayar S. A simple algorithm for nearest neighbor search in high dimensions. *IEEE Trans. PAMI.* 1997. Vol. 19, N 9. P. 989–1003.
43. Chavez E., Marroquín J., Baeza-Yates R. Spaghettis: an array based algorithm for similarity queries in metric spaces. *Proc. SPIRE'99*. 1999. P. 38–46.
44. Munro I., Raman R., Raman V., Rao S.S. Succinct representations of permutations and functions. *Theor. Comput. Sci*. 2012. Vol. 438. P. 74–88.

45. Chavez E., Ruiz U., Tellez E. CDA: Succinct spaghetti. *Proc. SISAP'15*. 2015. P. 54–64.
46. Tokoro K., Yamaguchi K., Masuda S. Improvements of TLAESA nearest neighbour search algorithm and extension to approximation search. *Proc. ACSC'06*. 2006. P. 77–83.
47. Ruiz G., Santoyo F., Chavez E., Figueroa K., Tellez E. Extreme pivots for faster metric indexes. *Proc. SISAP'13*. 2013. P. 115–126.
48. Uhlmann J.K. Satisfying general proximity/similarity queries with metric trees. *Information Processing Letters*. 1991. Vol. 40, N4. P. 175–179.
49. Yianilos P.N. Data structures and algorithms for nearest neighbor search in general metric spaces. *Proc. SODA'93*. 1993. P. 311–321.
50. Chiueh T. Content-based image indexing. *Proc. VLDB'94*. 1994. P. 582–593.
51. Bozkaya T., Ozsoyoglu M. Indexing large metric spaces for similarity search queries. *ACM Trans. Datab. Syst*. 1999. Vol. 24, N 3. P. 361–404.
52. Fu A.W.-C., Chan P.M.-S., Cheung Y.-L., Moon Y.S. Dynamic vp-tree indexing for n-nearest neighbor search given pair-wise distances. *VLDB Journal*. 2000. Vol. 9, N 2. P. 154–173.
53. Yianilos P. Excluded middle vantage point forests for nearest neighbor search. In: DIMACS Implementation Challenge, ALENEX'1999. URL: http://citeseer.ist.psu.edu/.
54. Kalantari I., Mcdonald G. A data structure and an algorithm for the nearest point problem. *IEEE Trans. Softw. Eng*. 1983. Vol. 9, N 5. P. 631–634.
55. Dehne F., Noltemeier H. Voronoi trees and clustering problems. *Information Systems*. 1987. Vol. 12, N 2. P. 171–175.
56. Noltemeier H., Verbarg K., Zirkelbach C. Monotonous bisector* trees — A tool for efficient partitioning of complex scenes of geometric objects. *LNCS*. 1992. Vol. 594. P. 186–203.
57. Ciaccia P., Patella M., Zezula P. Mtree: An efficient access method for similarity search in metric spaces. *Proc. VLDB'97*. 1997. P. 426–435.
58. Zezula P., Savino P., Amato G., Rabitti F. Approximate similarity retrieval with M-trees. *VLDB Journal*. 1998. Vol. 7, N 4. P. 275–293.
59. Skopal T., Pokorny J., Snasel V. PM-tree: PM-tree: pivoting metric tree for similarity search in multimedia databases. *Proc. ADBIS'04*. 2004. P. 99–114.
60. Jin S., Kim O., Feng W. MX-tree: A double hierarchical metric index with overlap reduction. *Proc. ICCSA'13*. 2013. P. 574–589.
61. Brin S. Near neighbor search in large metric spaces. *Proc. VLDB'95*. 1995. P. 574–584.
62. Fredriksson K. Geometric near-neighbor access tree (GNAT) revisited. arXiv:1605.05944. 20 May 2016.
63. Navarro G., Uribe R. Fully dynamic metric access methods based on hyperplane partitioning. *Information Systems*. 2011. Vol. 36, N 4. P. 734–747.
64. Connor R. Reference point hyperplane trees. *Proc. SYSAP'16*. 2016. P. 65–78.
65. O'Hara S., Draper B.A. Are you using the right approximate nearest neighbor algorithm? *Proc. WACV'13*. 2013. P. 9–14.
66. Comer D. The ubiquitous B-tree. *ACM Comput. Surv*. 1979. Vol. 11. P. 121–138.
67. Novak D., Batko M. Metric Index: An efficient and scalable solution for precise and approximate similarity search. *Information Systems*. 2011. Vol. 36, N 4. P. 721–733.
68. Lokoc J., Mosko J., Cech P., Skopal T. On indexing metric spaces using cut-regions. *Information Systems*. 2014. Vol. 43. P. 1–19.
69. Chen L., Gao Y., Li X., Jensen C.S., Chen G. Efficient metric indexing for similarity search. *Proc. ICDE'15*. 2015. P. 591–602.
70. Navarro G. Searching in metric spaces by spatial approximation. *VLDB Journal*. 2002. Vol. 11, N 1. P. 28–46.
71. Navarro G., Reyes N. Dynamic spatial approximation trees. *Journal of Experimental Algorithmics*. 2009. Vol. 12. Article 1.5. 68 p.
72. Barroso M., Reyes N., Paredes R. Enlarging nodes to improve spatial approximation trees. *Proc. SISAP'10*. 2010. P. 41–48.
73. Navarro G., Reyes N. New dynamic metric indices for secondary memory. *Information Systems*. 2016. Vol. 59. P. 48–78.
74. Chavez E., Luduena V., Reyes N., Roggero P. Faster proximity searching with the distal SAT. *Information Systems*. 2016. Vol. 59. P. 15–47.
75. Beygelzimer A., Kakade S., Langford J.C. Cover trees for nearest neighbor. *Proc. ICML'06*. 2006. P. 97–104.
76. Curtin R.R. Improving dual-tree algorithms: Ph.D. thesis. Georgia Inst. Tech. 2015. 246 p.
77. Chavez E., Navarro G. A compact space decomposition for effective metric indexing. *Pattern Recognition Letters*. 2005. Vol. 26, N 9. P. 1363–1376.
78. Roggero P., Reyes N., Figueroa K., Paredes R. List of clustered permutations in secondary memory for proximity searching. *J. of Com. Science Tech*. 2015. Vol. 15, N 2. P. 107–113.

79. Ponomarenko A., Avrelin N., Naidan B., Boytsov L. Comparative analysis of data structures for approximate nearest neighbor search. *DATA ANALYTICS 2014*. 2014. P. 125–130.
80. Dohnal V., Gennaro C., Savino P., Zezula P. D-index: Distance searching index for metric data sets. *Multimedia Tools and Applications*. 2003. Vol. 21, N 1. P. 9–33.
81. Cayton L. Accelerating nearest neighbor search on manycore systems. *Proc. IPDPS'12*. 2012. P. 402–413.
82. Tellez E.S., Ruiz G., Chavez E. Singleton indexes for nearest neighbor search. *Information Systems*. 2016. Vol. 60. P. 50–68.
83. Rosenkrantz D.J., Stearns R.E., Lewis P.M. II. An analysis of several heuristics for the traveling salesman problem. *SIAM Journal on Computing*. 1977. Vol. 6, N 3. P. 563–581.
84. Gonzalez T.F. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*. 1985. Vol. 38. P. 293–306.
85. Bustos B., Navarro G., Chavez E. Pivot selection techniques for proximity searching in metric spaces. *Pattern Recogn. Lett*. 2003. Vol. 24. P. 2357–2366.
86. Brisaboa N.R., Farina A., Pedreira O., Reyes N. Similarity search using sparse pivots for efficient multimedia information retrieval. *Proc. ISM'06*. 2006. P. 881–888.
87. Van Leuken R.H., Veltkamp R.C. Selecting vantage objects for similarity indexing. *ACM Trans. Multimedia Comput. Commun. Appl.* 2011. Vol. 7. P. 16:1–16:18.
88. Kim S.-H., Lee D.-Y., Cho H.-G. An eigenvalue-based pivot selection strategy for improving search efficiency in metric spaces. *Proc. BigComp'16*. 2016. P. 207–214.
89. Berman A., Shapiro L.G. Selecting good keys for triangle-inequality-based pruning algorithms. *Proc. CAIVD'98*. 1998. P. 12–19.
90. Venkateswaran J., Kahveci T., Jermaine C.M., Lachwani D. Reference-based indexing for metric spaces with costly distance measures. *VLDB Journal*. 2008. Vol. 17, N 5. P. 1231–1251.
91. Mao R., Zhang P., Li X., Xi L., Lu M. Pivot selection for metric-space indexing. *Int. J. Mach. Learn. Cybern*. 2016. Vol. 7, N 2. P. 311–323.
92. Celik C. Effective use of space for pivot-based metric indexing structures. *Proc. SISAP'08*. 2008. P. 113–120.
93. Hetland M.L., Skopal T., Lokoc J., Beecks C. Ptolemaic access methods: Challenging the reign of the metric space model. *Information Systems*. 2013. Vol. 38, N 7. P. 989–1006.
94. Hetland M.L. Ptolemaic indexing. *JoCG*. 2015. Vol. 6, N 1. P. 165–184.
95. Connor R., Vadicamo L., Cardillo F.A., Rabitti F. Supermetric search with the four-point property. *Proc. SISAP'16*. 2016. P. 51–64.
96. Ciaccia P., Patella M. Searching in metric spaces with user-defined and approximate distances. *ACM Database Systems*. 2002. Vol. 27, N 4. P. 398–437.
97. Chen L., Lian X. Efficient similarity search in nonmetric spaces with local constant embedding. *IEEE TKDE*. 2008. Vol. 20, N 3. P. 321–336.
98. Skopal T., Lokoc J. NM-tree: Flexible approximate similarity search in metric and non-metric spaces. *Proc. DEXA'08*. 2008. P. 312–325.
99. Curtin R.R., Ram P., Gray A.G. Fast exact max-kernel search. *Proc. SDM'13*. 2013. P. 1–9.
100. Keogh E., Ratanamahatana C. Exact indexing of dynamic time warping. *Knowledge and Information Systems*. 2005. Vol. 7, N 3. P. 358–386.
101. Zhang Z., Ooi B.C., Parthasarathy S., Tung A.K.H. Similarity search on bregman divergence: towards non-metric indexing. *Proc. VLDB Endowment*. 2009. Vol. 2. P. 13–24.
102. Abdullah A., Moeller J., Venkatasubramanian S. Approximate Bregman near neighbors in sublinear time: Beyond the triangle inequality. *Proc. SCG'12*. 2012. P. 31–40.
103. Amato G., Savino P. Approximate similarity search in metric spaces using inverted files. *Proc. InfoScale'08*. 2008. P. 28:1–28:10.
104. Chavez E., Figueroa K., Navarro G. Effective proximity retrieval by ordering permutations. *IEEE TPAMI*. 2008. Vol. 30, N 9. P. 1647–1658.
105. Tellez E.S., Chavez E., Camarena-Ibarrola A. A brief index for proximity searching. *Proc. CIARP'09*. 2009. P. 529–536.
106. Amato G., Gennaro C., Savino P. Mi-file: using inverted files for scalable approximate similarity search. *Multimed. Tools Appl*. 2014. Vol. 71, N 3. P. 1333–1362.
107. Esuli A. Use of permutation prefixes for efficient and scalable approximate similarity search. *Information Processing & Management*. 2012. Vol 48, N 5. P. 889–902.
108. Tellez E.S., Chavez E., Navarro G. Succinct nearest neighbor search. *Information Systems*. 2013. Vol. 38, N 7. P. 1019–1030.
109. Chavez E., Graff M., Navarro G., Tellez E. Near neighbor searching with K nearest references. *Information Systems*. 2015. Vol. 51. P. 43–61.
110. Naidan B., Boytsov L., Nyberg E. Permutation search methods are efficient, yet faster search is possible. Proc. *VLDB Endowment.* 2015. Vol. 8, N. 12. P. 1618–1629.

111. Goyal N., Lifshits Y., Schutze H. Disorder inequality: A combinatorial approach to nearest neighbor search. *Proc. WSDM'08*. 2008. P. 25–32.
112. Lifshits Y., Zhang S. Combinatorial algorithms for nearest neighbors, near-duplicates and small world design. *Proc. SODA'09*. 2009. P. 318–326.
113. Tschopp D., Diggavi S.N., Delgosha P., Mohajer S. Randomized algorithms for comparison-based search. *Proc. NIPS'11*. 2011. P. 2231–2239.
114. Houle M.E., Sakuma J. Fast approximate similarity search in extremely high-dimensional data sets. *Proc. ICDE'05*. 2005. P. 619–630.
115. Houle M.E., Nett M. Rank-based similarity search: Reducing the dimensional dependence. *IEEE TPAMI*. 2015. Vol. 37, N 1. P. 136–150.
116. Arya S., Mount D.M. Approximate nearest neighbor queries in fixed dimensions. *Proc. SODA'93*. 1993. P. 271–280.
117. Sebastian T., Kimia B. Metric-based shape retrieval in large databases. *Proc. ICPR'02*. 2002. Vol. 3. P. 291–296.
118. Paredes R., Chavez E. Using the k-nearest neighbor graph for proximity searching in metric spaces. *Proc. SPIRE'05*. 2005. P. 127–138.
119. Hajebi K., Abbasi-Yadkori Y., Shahbazi H., Zhang H. Fast approximate nearest-neighbor search with K-nearest neighbor graph. *Proc. IJCAI'11*. 2011. P. 1312–1317.
120. Malkov Y., Ponomarenko A., Logvinov A., Krylov V. Scalable distributed algorithm for approximate nearest neighbor search problem in high dimensional general metric spaces. *Proc. SISAP'12*. 2012. P. 132–147.
121. Malkov Y., Ponomarenko A., Logvinov A., Krylov V. Approximate nearest neighbor algorithm based on navigable small world graphs. *Information Systems*. 2014. Vol. 45. P.61–68.
122. Harwood B., Drummond T. FANNG: fast approximate nearest neighbour graphs. *Proc. CVPR'16*. 2016. P. 5713–5722.
123. Paredes R., Chavez E., Figueroa K., Navarro G. Practical construction of k-nearest neighbor graphs in metric spaces. *Proc. WEA'06*. 2006. P. 85–97.
124. Dong W., Charikar M., Li K. Efficient K-nearest neighbor graph construction for generic similarity measures. *Proc. WWW'11*. 2011. P. 577–586.
125. Aoyama K., Saito K., Sawada H., Ueda N. Fast approximate similarity search based on degree-reduced neighborhood graphs. *Proc. KDD'11*. 2011. P. 1055–1063.
126. Li W., Zhang Y., Sun Y., Wang W., Zhang W., Lin X. Approximate nearest neighbor search on high dimensional data — experiments, analyses, and improvement. arXiv:1610.02455. 8 Oct 2016.
127. Watts D.J., Strogatz S.H. Collective dynamics of small-world networks. *Nature*. 1998. Vol. 393, N 6684. P. 440–442.
128. Kleinberg J. The small-world phenomenon: an algorithmic perspective. *Proc. STOC'00*. 2000. P. 163–170.
129. Chung F.R.K. Diameters of graphs: old problems and new results. *Congr. Numer.* 1987. Vol. 60. P. 295–317.
130. Achlioptas D., Siminelakis P. Navigability is a robust property. *Proc. WAW'15*. 2015. P. 78–91.
131. Fraigniaud P., Giakkoupis G. On the searchability of small-world networks with arbitrary underlying structure. *Proc. STOC'10.* 2010. P. 389–398.
132. Fraigniaud P., Lebhar E., Lotker Z. A lower bound for network navigability. *SIAM Journal on Discrete Mathematics*. 2010. Vol. 24, N 1. P. 72–81.
133. Fraigniaud P., Gavoille C., Kosowski A., Lebhar E., Lotker Z. Universal augmentation schemes for network navigability: Overcoming the $\sqrt{n}$-barrier. *Proc. SPAA'07*. 2007. P. 1–7.
134. Ruiz G., Chavez E., Graff M., Tellez E.S. Finding near neighbors through local search. *Proc. SISAP'15*. 2015. P. 103–109.
135. Ponomarenko A., Avrelin N., Naidan B., Boytsov L. Comparative analysis of data structures for approximate nearest neighbor search. *Proc. Data Analytics'14.* 2014. P. 125–130.
136. Malkov Yu.A., Yashunin D.A. Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. arXiv:1603.09320. 21 May, 2016.
137. Sokolov A. Vector representations for efficient comparison and search for similar strings. *Cybernetics and Systems Analysis*. 2007. Vol. 43, N 4. P. 484–498.
138. Sokolov A. Investigation of accelerated search for close text sequences with the help of vector representations. *Cybernetics and Systems Analysis.* 2008. Vol. 44, N 4. P. 493–506.
139. Charikar M. Similarity estimation techniques from rounding algorithms. *Proc. STOC'02.* 2002. P. 380–388.
140. Tellez E.S., Chavez E. On locality sensitive hashing in metric spaces. *Proc. SISAP'10.* 2010. P. 67–74.
141. Athitsos V., Potamias M., Papapetrou P., Kollios G. Nearest neighbor retrieval using distance-based hashing. *Proc. ICDE'08.* 2008. P. 327–336.
142. Jangyodsuk P., Papapetrou P., Athitsos V. Optimizing hashing functions for similarity indexing in arbitrary metric and nonmetric spaces. *Proc. SDM'15*. 2015. P. 828–836.

143. Andrade J.M., Astudillo C.A, Paredes R. Metric space searching based on random bisectors and binary fingerprints. *Proc. SISAP'14.* 2014. P. 50–57.
144. Kang B., Jung K. Robust and efficient locality sensitive hashing for nearest neighbor search in large data sets. *Proc. BigLearn'12*. 2012. P. 1–8.
145. Silva E.S, Teixeira T.S.F.X, Teodoro G., Valle E. Large-scale distributed locality-sensitive hashing for general metric data. *Proc. SISAP'14.* 2014. P. 82–93.
146. Novak D., Kyselak M., Zezula P. On locality-sensitive indexing in generic metric spaces. *Proc. SISAP'10.* 2010. P. 59–66.
147. Becker A., Ducas L., Gama N., Laarhoven T. New directions in nearest neighbor searching with applications to lattice sieving. *Proc. SODA'16*. 2016. P. 10–24.
148. ANN benchmark. http://github.com/erikbern/ann-benchmarks. Accessed 12 Apr. 2017.

Надійшла до редакції 07.07.2016

Д.А. Рачковський

ОСНОВАНІ НА ВІДСТАНЯХ ІНДЕКСНІ СТРУКТУРИ ДЛЯ ШВИДКОГО ПОШУКУ ЗА СХОЖІСТЮ

Анотація. Розглянуто клас таких індексних структур для швидкого пошуку за схожістю, при конструюванні та застосуванні яких використовують тільки інформацію про значення або ранг деяких відстаней/схожостей між об'єктами. Обговорено пошук як за метричними відстанями (для яких виконується нерівність трикутника та інші метричні аксіоми), так і за неметричними. Наведено структури, які повертають об'єкти бази, що є точною відповіддю на запит, а також структури для наближеного пошуку за схожістю (вони не гарантують точності, але зазвичай повертають близькі до точних результати та працюють швидше структур для точного пошуку). Викладено загальні принципи конструювання і застосування деяких індексних структур, а також розглянуто ідеї, на яких базуються конкретні алгоритми (відомі та запропоновані останнім часом).

Ключові слова: пошук за схожістю, пошук найближчих сусідів, індексні структури, індексування на основі відстаней, метрична відстань, неметрична відстань, метричне дерево, граф сусідства, метод гілок і меж.

D.A. Rachkovskij

DISTANCE-BASED INDEX STRUCTURES FOR FAST SIMILARITY SEARCH

Abstract. In this survey paper we consider the class of index structures for fast similarity search that uses for index construction and application only information about the values or ranks of some distances/similarities between objects. We discuss the search by metric distances (for which the triangle inequality and other metric axioms are valid), as well as by non-metric ones. Considered index structures include those returning the objects of the base that are exact results to the similarity search query, and index structures for approximate similarity search, which do not guarantee the accuracy, but usually return close to accurate results and work faster than the structures for exact search. Some general principles for construction and usage of index structures as well as some ideas of specific algorithms, including recently proposed ones, are discussed.

Keywords: similarity search, nearest neighbor search, index structures, distance-based indexing, metric distances, non-metric distances, metric trees, proximity graph, branch and bound.

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.