Методы поиска ближайших соседей в задаче анализа графического образа структурированного документа

Д. В. Полевой, В. В. Постников

В статье рассматриваются алгоритмы и вспомогательные структуры данных для эффективного поиска ближайших соседей (Nearest Neighbor Search) в контексте задачи распознавания текстовых документов.

1. Введение

Проблема поиска среди множества информационных объектов наиболее ‘похожего’ на заданный часто возникает во многих задачах искусственного интеллекта. В качестве примеров можно перечислить следующие прикладные задачи: классификация и кластеризация, распознавание текста, образов и речи, поиск информации и сжатие данных, построение баз изображений и видео документов.

Построение эффективных методов поиска требует учета особенностей исходной задачи. Например, строки и вектора должны обрабатываться разными способами, поэтому используемые алгоритмы зависят от природы информационных объектов. Если все необходимые данные могут храниться в оперативной памяти, то методы поиска оптимизируются по количеству операций, в то время как хранение исходных данных на жестком диске требует оптимизации количества обращений к диску. Еще одним важным фактором для векторных данных является размерность пространства, которая существенным образом влияет на выбор конкретного метода.

Данная работа посвящена эффективным методов поиска ближайших соседей, а так же особенностям их использования в задаче анализа изображения структурированного документа. Объекты анализа имеют небольшое количество числовых характеристик, а современные компьютеры позволяют разместить основные данные и вспомогательные структуры полностью в оперативной памяти.

Распознавание образа графического документа начинается с анализа изображения. В ходе анализа оценивается перекос образа при сканировании, выделяются линии и текстовые фрагменты, изображение сегментируется. Если объектами анализа являются фрагменты изображения, то

конфигурации пикселей, интервалов, компонент связности оцениваются на основании геометрических, цветовых и текстурных характеристик. «Близость» конфигураций является основанием для дальнейшей обработки, а поиск ближайших соседей является часто встречающимся элементарным шагом общей схемы. Количество изучаемых на изображении документа объектов достигает десятков и сотен тысяч, что требует специальных алгоритмов обработки для достижения промышленно приемлемых временных характеристик программ.

2. Формальная постановка задачи

Будем использовать следующие обозначения:

$A = \{a_1, a_2, \ldots, a_N\}$ — множество исходных объектов;

$N = |A|$ — мощность множества исходных объектов;

S — множество объектов запроса или поиска (query point);

$dist(\cdot\,, \cdot) \in R$ — мера близости;

$V : v \in R^d$ — векторное пространство;

d — размерность векторного пространства;

$v \in V$, $v = \{v^1, v^2, \ldots, v^d\}$ — вектор (точка, элемент векторного пространства).

2.1. Задача поиска k-ближайших соседей (kNN-задача)

Задача поиска k-ближайших соседей (k-NN search) может быть сформулирована следующим образом: для объекта запроса $s \in S$ найти подмножество $B \subseteq A$ объектов данных такое, что

$$|B| = k, \quad \forall\, b \in B \quad \forall\, c \in A \setminus B : \; dist(s, b) \leqslant dist(s, c).$$

При $k \geqslant |A|$ ответом является все множество исходных объектов данных A, далее этот случай специально рассматривать не будем, считая, что $k < |A|$.

2.2. Задача поиска k^*-ближайших соседей (k^*NN-задача)

Если существует несколько равноудаленных от точки запроса s объектов данных, то не все они попадают в ответ kNN-задачи. Какие именно объекты попадают, а какие нет, зависит от конкретной реализации, что может порождать неоднозначное поведение алгоритмов, опирающихся на результаты поиска соседей. Для разрешения подобных неоднозначностей, возможна следующая формулировка задачи поиска k^*-ближайших соседей, расширяющая задачу поиска k-ближайших соседей:

для объекта запроса $s \in S$ найти минимальное подмножество $B \subseteq A$ объектов данных такое, что

$$|B| \geqslant k, \quad \forall\, b \in B \quad \forall\, c \in A \setminus B : \ dist(s, b) < dist(s, c).$$

2.3. Задача инкрементного поиска ближайших соседей

Наиболее общей задачей поиска соседей может считаться задача инкрементного поиска (Incremental Nearest Neighbor Search). Задача инкрементного поиска состоит в упорядочивании исходных объектов в порядке возрастания расстояния от объекта запроса:

для объекта запроса $s \in S$ найти упорядочение множества объектов данных A такое, что

$$\forall\, a_i, a_j \in A : i < j : \ dist(s, a_i) \leqslant dist(s, a_j).$$

На практике задача инкрементного поиска ближайших соседей часто является частью более сложной задачи. В таких случаях удобно использовать итеративные решения, которые на каждом шаге находят очередного по удаленности соседа.

Отметим, что проблема инкрементного поиска является более широкой, чем поиск k-ближайших, так как последняя лишь частный случай, когда известно точно, сколько соседей необходимо найти.

3. Методы поиска ближайших соседей

Числовое представление характеристик является естественным для ЭВМ, поэтому наиболее интересным является случай, когда множества объектов данных и объектов запросов лежат в одном векторном пространстве $V \subseteq R^d$, т. е. $A \subseteq V$ и $S \subseteq V$. При этом объекты данных часто называют точками или векторами. Мера близости (удаленности) $dist(\cdot, \cdot)$ определена на элементах V, то есть $dist : V \times V \to R$. Сразу отметим, что мера близости не обязана быть метрикой.

Если в ходе решения задачи о поиске соседей множество исходных объектов A может изменяться, то есть объекты добавляются или удаляются, то задача называется динамической. Если множество объектов данных фиксировано и не изменяется, то это статическая задача.

3.1. Обзор методов и структур данных

Для одномерного случая задача поиска ближайших соседей имеет оптимальное решение, которое опираются на бинарный поиск [1]. Для обобщения бинарного поиска на пространства большей размерности используются диаграммы Вороного, случайные выборки (random sampling) и другие методы вычислительной геометрии [2–5]. Однако, увеличение

размерности исходного пространства приводят к неэффективным по скорости или объему требуемой памяти реализации алгоритмов.

Если для исходной практической задачи достаточно поиска соседей не далее, чем внутри фиксированной окрестности, то возможно использование ассоциативных структур. Эти структуры эффективно реализуют поиск в гиперпрямоугольной окрестности точки запроса [6].

Другим подходом к решению задачи является использование вспомогательных структур данных, описывающих рекурсивное разбиение исходного множества точек данных и самого пространства, соответственно. В качестве примеров можно перечислить region quadtrees [7], kd-tree [8], различные варианты R-tree [9] и алгоритмы над этими структурами [10–13]. Далее подробнее будет рассмотрено использовании R-деревьев в задачах поиска соседей.

3.2. R-деревья

Подробное рассмотрение семейства структур данных типа R-дерева можно найти в [14], а здесь напомним основную идею. R-деревья используется для организации геометрических объектов в d-мерном векторном пространстве V. Каждый объект представляется своим минимальным охватывающим прямоугольником (МОП).

МОП множества точек — это гиперпараллелепипед обладающий следующими свойствами:

- ребра параллелепипеда параллельны осям координат;
- множество точек полностью лежит внутри;
- объем минимален.

Формально МОП множества точек A может быть задан следующей системой неравенств

$$a^i \leqslant v^i \leqslant b^i,$$

где $a^i = \min a^i_j$, по всем $a_j \in A$; $b^i = \max b^i_j$, по всем $b_j \in A$.

Каждой вершине R-дерева соответствует МОП, ограничивающий ее потомков. Терминальные вершины дерева вместо ссылок на детей содержат ссылки на исходные объекты. В общем случае МОПы вершин дерева одного уровня могут пересекаться. Более того, МОП в геометрическом смысле может принадлежать многим вершинам, в том числе и не родительским, однако ассоциирован он должен быть только с одной из них.

Для определенности будем считать, что R-дерево имеет следующую структуру данных:

`Node` — вершина дерева;

`Node.rect` — МОП вершины;

`Node.ChildCount` — количество детей данной вершины;

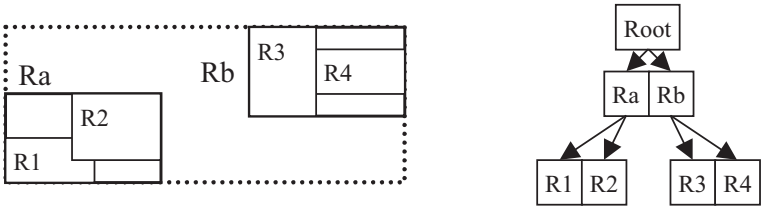

Рис. 1. Пример МОП объектов для R^2 и соответствующей структуры R-дерева

`Node.branch[i]`, `i=1`, `Node.ChildCount` — массив ссылок на детей для внутренних вершин дерева и массив ссылок на исходные объекты;

`Node.Type`, `Node.Type = {INTERNAL, LEAF}` — тип вершины дерева;

`INTERNAL` — для внутренней вершины дерева;

`LEAF` — для терминальной вершины дерева.

Предложенные в [8] R-деревья инициировали целый ряд работ по разработке и использованию новых структур данных. Roussopoulos с соавторами [10] рассмотрел вариант решения задачи поиска k-соседей в n-мерном Евклидовом пространстве с использованием R-дерева, когда k точно известно в начале поиска. Основная идея этого алгоритма состоит в поддержании списка k ближайших кандидатов во время обхода R-дерева в глубину с возвратом. На каждом шаге спуска принимается локальное решение о спуске в ребенка текущего узла на основании оценки расстояния до МОП соответствующих вершин дерева. Если поддерево соответствующее одному ребенку исследовано полностью, а критерий останова не достигнут, то исследуется поддерево следующего по удаленности МОП ребенка из оставшихся необследованными.

Рассмотрим подробнее данный подход для случая $k = 1$ и покажем способ его обобщения. Для выбора направления спуска и отсечения ветвей дерева при обходе используются метрики, которые определяют оптимистичное и пессимистичное расстояние между содержимым R-дерева и точкой запроса.

Пусть дана точка запроса s и объект O, который представлен своим минимальным охватывающим прямоугольником (МОП). Рассмотрим две функции. Первая обозначается MINDIST и соответствует минимально возможному расстоянию между s и O. Более точно, если точка s лежит внутри МОП(O), то MINDIST$(s, \mathrm{O}) = 0$, а если s лежит вне МОП(O), то значение MINDIST(s, O) равно расстоянию от точки s до ближайшей к ней точки границы МОП(O). Вторая обозначается

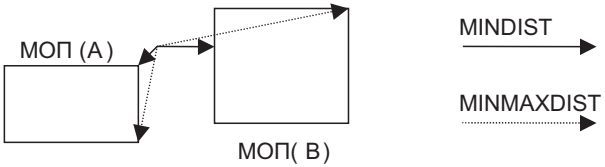

Рис. 2. Пример MINDIST и MINMAXDIST для объектов в R^2

MINMAXDIST и равна минимальному из максимально возможных расстояний от s до объекта внутри МОП(O).

Эти две метрики определяют нижнюю и верхнюю границы расстояния между s и ближайшим соседом из O соответственно. С использованием этих метрик авторы предлагают следующий алгоритм обхода R-дерева с отсечением.

```
01 Procedure NNSearch(Node, Point, Nearest)
02 if (Node.type == LEAF)
03   for i = 1 to Node.ChildCount
04     dist = objectDIST(Point, Node.branch[i])
05     if (dist < Nearest.dist)
06       Nearest.dist = dist
07       Nearest.rect = Node.branch[i].rect
08     endif
09   endfor
10 else
11   genBranchList(Node, branchList)
12   sortBranchList(Node, branchList)
13   last = pruneBranchList(Node,Point,Nearest,branchList)
14   for i = 1 to last
15     newNode = node.brach[branchList[i]]
16     NNSearch(newNode, Pount, Nearest)
17     last = pruneBranchList(Node,Point,Nearest,branchList)
18   endfor
19 endif
20 end
```

Для поиска ближайшего соседа Nearest расстояние до него устанавливается плюс бесконечность, а затем процедуру поиска применяется к корню дерева.

```
Procedure FindNN(Point, Nearest)
  Nearest.dist = +infinity
  NNSearch(Root, Point, Nearest)
end
```

Индексы вершин-кандидатов на исследование хранятся в отсортированном по расстоянию (MINDIST или MINMAXDIST) от точки поиска до этих вершин списке branchList. Во время спуска для текущей вершины R-дерева список кандидатов на исследование пополняется за счет детей (строка 11) и сортируется в порядке возрастания оценки расстояния (строка 12). Далее на основании эвристик из списка кандидатов удаляются заведомо бесперспективные вершины (строка 13), и поиск продолжается до исчерпания списка кандидатов (строки 14–18) в порядке возрастания оценки расстояния. Когда список вершин-кандидатов исчерпан, то Nearest содержит ответ. Если алгоритм поиска доходит до терминальной вершины R-дерева (строка 02), то среди всех точек данных ищется ближайшая к точке запроса (строки 03–09).

Согласно [10] для точки запроса s отсечение поддеревьев поиска во время обхода дерева возможно в соответствии со следующими утверждениями:

(1) если для двух МОП вершин дерева M и M′ расстояние MINDIST(s, M) больше чем MINMAXDIST(s, M′), то поддерево соответствующее M исключается из поиска, так как не может содержать ближайшего соседа;

(2) если текущее расстояние от s до данного объекта O превышает расстояние MINMAXDIST(s, M), где M — МОП некоторого набора объектов, то O может быть исключено из рассмотрения, так как существует O′ лежащий в M и расположенный ближе к s, чем O;

(3) если MINDIST(s, M) для некоторого M превышает текущее расстояние до ближайшего соседа O, то соответствующее M поддерево исключается из рассмотрения, так как не может содержать объект лежащий ближе, чем O.

По словам авторов, MINDIST соответствует более оптимистичному порядку исследования дерева, чем MINMAXDIST, однако, могут существовать наборы данных (в зависимости от расположения и размера МОП вершин R-дерева), для которых последняя эвристика дает более эффективный поиск.

Рассмотренный алгоритм легко обобщается на случай $k > 1$. Принципиальными отличиями является поддержание отсортированного по удаленности от точки поиска s списка ближайших k соседей, и использование для сужения поиска не ближайшего, а k-го соседа.

Дальнейшее исследование Cheung и Fu [15] показало, что для эффективного поиска достаточно только третьего из вышеперечисленных утверждений (это же наблюдение независимо сделано в [12]).

3.3. Алгоритм инкрементного поиска ближайших соседей с использованием R-деревьев

Hjaltson и Samet в работе [12] рассмотрели проблему инкрементного поиска соседей с помощью R-деревьев для произвольных пространственно распределенных объектов. Частным случаем такого поиска является инкрементный поиск соседей для точечных объектов, который мы рассмотрим подробнее.

Основной идей алгоритма является обработка исследуемых объектов (МОП вершин R-дерева или исходных объектов) в порядке возрастания расстояний от точки запроса s. Для поддержания порядка обхода используется очередь с приоритетом, элементы которой сортируются в порядке возрастания расстояния (MINDIST). Авторы показывают, что для любого алгоритма поиска соседей с помощью варианта R-дерева в ходе поиска должны быть проверены все узлы дерева, МОП которых пересекаются с областью поиска. Приведенный инкрементный алгоритм является оптимальным в том смысле, что он проверяет только такие узлы.

Для сравнения с алгоритмом NNSearch рассмотрим упрощенную версию алгоритма инкрементного поиска.

```
01 Procedure INNSearch(Point)
02 Node = Root
03 dist = objectDIST(Point, Node)
04 enqueue(PriorityQueue, Node, dist)
05 while PriorityQueue not empty
06   Node = dequeue(PriorityQueue)
07   if (Node.type == POINT)
08     report Node
09   else
10     for i = 1 to Node.ChildCount
11       dist = objectDIST(Point, Node.branch[i])
12       enqueue(PriorityQueue, Node.branch[i], dist)
13     endfor
14   endif
15 endwhile
16 end
```

В этом алгоритме очередь с приоритетом `PriorityQueue` используется для хранения как вершин R-дерева, так и точек исходных данных. Точки данных хранятся в структуре, аналогичной узлам дерева, а для идентификации используется дополнительный идентификатор типа узла `POINT`. То есть `Node.type == POINT` для исходных объектов. В начале работы алгоритма в очередь помещается корень дерева (строки 02–04).

Далее вершины рассматриваются в порядке их извлечения из очереди (строки 05–15). Этот порядок соответствует увеличению расстояния от точки поиска s. Если очередная вершина соответствует точке исходных данных (строка 07), то сообщается ответ (строка 08). Если вершина является элементом структуры R-дерева (строки 07 и 09), то все дочерние вершины помещаются в очередь (строки 10–12).

Приведенный алгоритм вычисляет соседей точки запроса в порядке увеличения расстояния от нее и работает либо до достижения критерия останова (найдено необходимое количество соседей), либо до исчерпания очереди (все точки исходных данных обследованы).

3.4. Теоретические оценки алгоритмов

Важным вопросом практического использования алгоритмов поиска соседей с помощью R-деревьев является правильный выбор конкретного варианта индексной структуры и ее параметров. Теоретическому исследованию характеристик алгоритмов посвящено много работ, например [16–18]. Однако, универсальной методики оценки и оптимизации не существует, а отличные от общепринятых варианты использования рассматриваемых алгоритмов требуют построения специализированных моделей.

Исторически исследователи были вынуждены работать с массивами данных, которые не помещаются в оперативную память компьютера, поэтому общепринятой является модель постраничного хранения терминальных узлов R-дерева на жестком диске. Даже в последних исследованиях [18] в целях возможности масштабирования результатов на действительно большие объемы данных авторы проводят моделирование в условиях искусственных (с точки зрения текущих вычислительных мощностей) ограничений. Например, в качестве индексной структуры данных используют хранимые на диске R-деревья с ограничением на количество (128) хранимых в оперативной памяти вершин. Поскольку случайный доступ к данным на жестком диске значительно медленнее последовательного, задача оптимизации призвана минимизировать общее время работы за счет оптимизации доступа к вершинам R-дерева и возможности перехода к последовательному поиску.

Другой важной проблемой является переход от модельных распределений исходных данных и точек запросов к реальным данным прикладных задачам. В случае оптимизации запросов к БД возможен анализ распределения слабо изменяющихся данных для оптимизации времени исполнения, а использование алгоритмов поиска ближайших соседей для анализа изображений документов требует учета обобщенных особенностей обрабатываемого класса документов на этапе проектирования.

3.5. Обобщение на неметрические пространства

Наиболее популярными в литературе метриками являются функции семейства L_i, и в частности Евклидова метрика. При этом рассмотренные выше алгоритмы и структуры данных не опираются на все свойства метрик и могут быть обобщены на случай неметрических пространств.

Рассмотрим некоторую функцию расстояния, которую можно обобщить на измерение расстояния между точками запроса и МОП подмножеств A так, чтобы она удовлетворяла

$$\forall\, s \in V, \quad \forall\, A_2 \subseteq A_1 \subseteq A \to w(s, A_1) \leqslant w(s, A_2)$$

и для одноточечных множеств совпадала с исходной функцией расстояния

$$\forall\, s \in V, \quad \forall\, A_1 = \{a\} \subseteq A \to w(s, A_1) = w(s, a).$$

Очевидно, что

$$\forall\, s \in A, \quad \forall\, A_1 \subseteq A, \quad \forall\, A_2 \subseteq A \to w(s, A_1 \cup A_2) \leqslant w(s, A_1),$$

$$w(s, A_1 \cup A_2) \leqslant w(s, A_2), \quad \forall\, s \in S_1 \subseteq A \to w(s, S_1) = 0.$$

Функция такого вида может использоваться в качестве MINDIST, то есть быть нижней оценкой расстояния от точки запроса s до исходных точек, лежащих внутри МОП соответствующей вершины R-дерева. Использование такого обобщения необходимо использовать в задачах, где мера близости не является метрикой. Другим важным случаем являются задачи, в которых расстояние от точки поиска s до МОП может быть вычислено существенно быстрее, чем расстояние до накрываемых этим МОП точек. Такая быстрая оценка позволяет эффективно уменьшать общее время работы алгоритма за счет быстрого отсечения при, быть может, даже увеличивающемся количестве исследуемых вершин R-дерева.

3.6. Варианты построения R-дерева

Рассмотрим один из вариантов построения статического R-дерева методом иерархического деления пространства. Построение начинается с корня дерева и соответствующего всем точкам данных МОП. На каждом шаге построения дерева, МОП текущего уровня рассекается гиперплоскостью, перпендикулярной одной из осей координат. Назовем опорной осью сечения ту ось координат, перпендикулярно которой производится разбиение МОП текущего уровня R-дерева. Гиперплоскость делит точки данных на два подмножества, МОП которых приписываются дочерним узлам. Далее процедура повторяется рекурсивно для дочерних узлов до достижения критерия останова.

Практические характеристики алгоритмов построения и использования R-дерева зависят от:

- выбор опорной оси сечения;
- выбор гиперплоскости сечения;
- критерий останова разбиения.

В общем виде опорная ось и секущая гиперплоскость выбираются, как решение оптимизационной задачи для некоторой функции от вариантов разбиения. Однако применение даже простых эвристик позволяет получать хорошие результаты. В качестве примера приведем следующие варианты выбора опорной оси сечения:

- по циклу — МОП корневого уровня разбивается поперек первой координатной оси, а далее, при каждом спуске в дочерний узел, в качестве опорной оси сечения выбирается следующая по номеру ось;
- вдоль наиболее длинного ребра — в качестве опорной оси выбирается ось координат, вдоль которой лежит самое длинное ребро МОП;
- по максимуму дисперсии — в качестве опорной оси выбирается ось координат, проекции точек на которую имеют максимальную дисперсию.

В самом простом случае секущая гиперплоскость может проводиться из чисто геометрических соображений, например, через геометрическую середину ребра, лежащего вдоль опорной оси сечения или через медиану проекций точек на опорную ось. Более сложные варианты требуют дополнительного обоснования, так как усложнение процедуры построения дерева должны компенсироваться улучшениями характеристик его использования для решения конкретной прикладной задачи.

Критерии останова процесса разбиения вершин могут быть самыми разными, приведем несколько самых очевидных:

- максимальное ребро МОП меньше некоторого порогового значения;
- объем МОП меньше некоторого порогового значения;
- количество точек в вершине меньше некоторого порогового значения.

3.7. Применение алгоритмов поиска ближайших соседей в задаче анализа изображения документа

Перечислим некоторые задачи анализа изображений, эффективно решаемые с применением технологии поиска ближайших соседей:

- кластеризация и классификация;
- оценка перекоса изображения при сканировании;
- анализ структуры линий на изображении;

- анализ структуры текста на изображении;
- выделение текстовых фрагментов на изображении.

R-деревья и рассмотренные выше алгоритмы являются хорошей альтернативой ассоциативным структурам [6] без присущих последним ограничения на размер анализируемой окрестности точки поиска. Так же быстрый поиск соседей использовался нами в задаче построения минимального остовного дерева [19] и сегментации изображения документа на его основе [20]. На рис. 3 приведен пример структурированного документа и один из этапов анализа образа с помощью минимального остовного дерева на компонентах связности.

Отдельно отметим, что используемая в качестве индексной структура R-дерева без каких либо изменений и дополнительных накладных расходов может использоваться для эффективного поиска объектов в заданных областях.

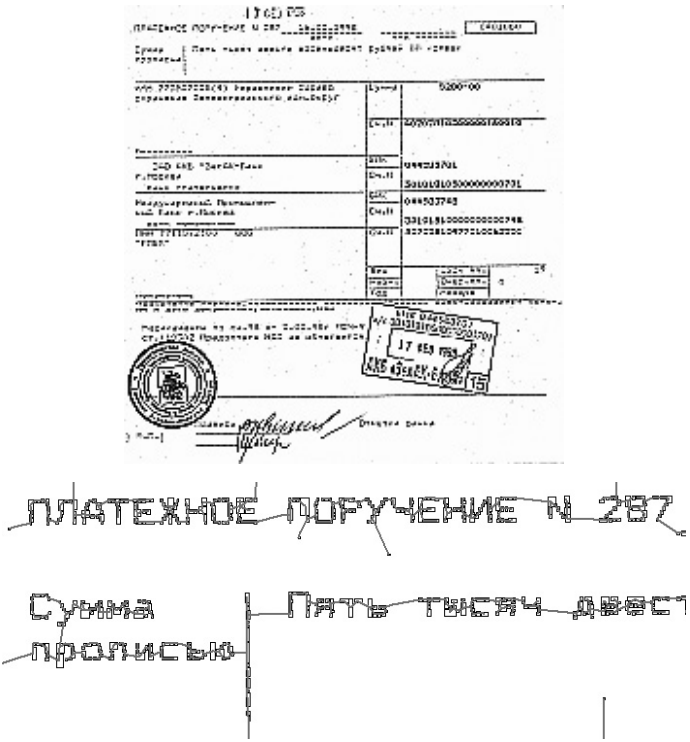

Рис. 3. Пример платежного поручения и построенного с помощью быстрого поиска соседей на компонентах связности изображения минимального остовного дерева

Все вышеперечисленные задачи анализа изображения структурированного документа характеризуются следующими особенностями:

- количество точек исходных данных порядка $N \in [10^2, 10^5]$;
- размерность пространства исходных данных мала, $d \in [2, 8]$, $d \ll N$;
- количество точек поиска сравнимо с количеством точек исходных данных;
- количество ближайших соседей $k \in [2, 50]$, то есть $k \sim d$, $k \ll N$;
- распределение точек исходных данных заранее неизвестно и может сильно меняться от документа к документу.

Таким образом, современная вычислительная техника позволяет хранить все исходные и вспомогательные данные в оперативной памяти. При практической реализации необходимо учитывать общую сложность построения вспомогательных структур и решения задачи поиска, так как для каждого изображения вспомогательные структуры строятся динамически. Последним рассматриваемый нами случай отличается, например, от популярных в литературе БД, где индексные структуры могут сохраняться и использоваться повторно при большом количестве запросов.

Поскольку существующие теоретические оценки обладают существенными ограничениями, выбор оптимальных параметров было решено произвести на основе эксперимента. В качестве примера рассматривалось решение задачи поиска k^*-соседей с использование бинарного статического R-дерева и инкрементного алгоритма (`INNSearch`). Экспериментальному исследованию подверглись изображения различных документов, при этом точки поиска были распределены следующим образом:

- случайно;
- в ячейках регулярной решетки;
- совпадали с исходными точками.

Измерения временных характеристик для различных распределений множества точек поиска показали, что зависимость несущественная и не влияет на характерные особенности поведения алгоритма. Далее все эксперименты проводились для множества точек поиска совпадающих с исходными точками.

В ходе экспериментов измерялись следующие характеристики:

- V2V — количество измерений расстояния между точкой поиска и исходными точками данных;
- V2R — количество измерений расстояния между точкой поиска и МОП вершин дерева;
- Sum — общее количество измерений расстояния, $\mathrm{Sum} = \mathrm{V2R} + \mathrm{V2V}$.

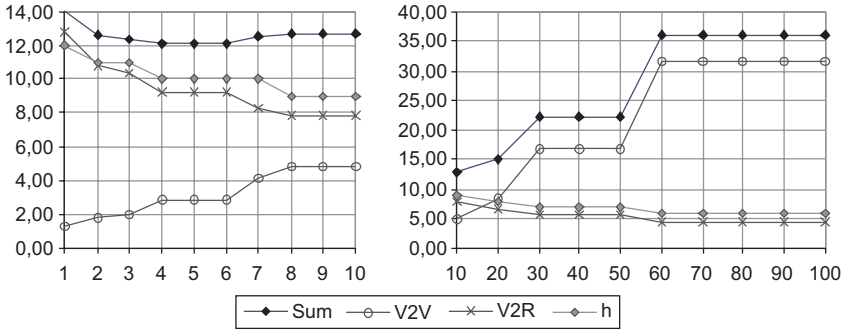

Рис. 4. Зависимость удельного числа элементарных операций измерения расстояний $\mathrm{Sum}(f)$, $\mathrm{V2R}(f)$ и $\mathrm{V2V}(f)$ от емкости терминальной вершины f, $k = 5$, $d = 3$

Количество измерений нормировалось на общее количество найденных соседей, то есть вычислялось среднее (удельное) количество измерений расстояния для каждого найденного соседа.

На следующем графике (рис.4) представлен типичный вид зависимости удельного числа элементарных операций измерения расстояний от емкости терминальной вершины R-дерева.

Общее количество измерений (Sum) складывается из двух составляющих. Первая часть (V2R) связана со спуском по дереву и зависит от его высоты $h = \log_2(N/f)$, что хорошо видно на графике. При увеличении емкости терминального узла высота дерева уменьшается, поэтому с ростом f абсолютный и относительный вклад V2R тоже уменьшается. Вторая часть (V2V) зависит от емкости терминального узла и характеризует «избыточность». Если $f = 1$, то V2R дает достаточно точную оценку для расстояния до терминальной вершины и единственного соседа в ней, соответственно. При увеличении f доля точек исходных данных в терминальной вершине, которые попадают в ответ запроса, уменьшается.

При малых f основной вклад дает V2R. С ростом f, за счет уменьшения V2R и слабого роста V2V, достигается субоптимальное значение. При дальнейшем росте f скорость уменьшения V2R падает и V2V дает основной вклад в общее количество измерений.

При увеличении k и сохранении $k \ll N$ характер зависимости не изменяется (теоретическое обоснование и методику оценки границы применимости можно найти в [17]). Для примера рассмотрим график той же зависимости при $k = 100$ (рис.5).

Для значений емкости f терминальной вершины дерева наблюдается интервал субоптимальных значений. Для различных изображений точное оптимальное значение f может быть разным, однако, выбор

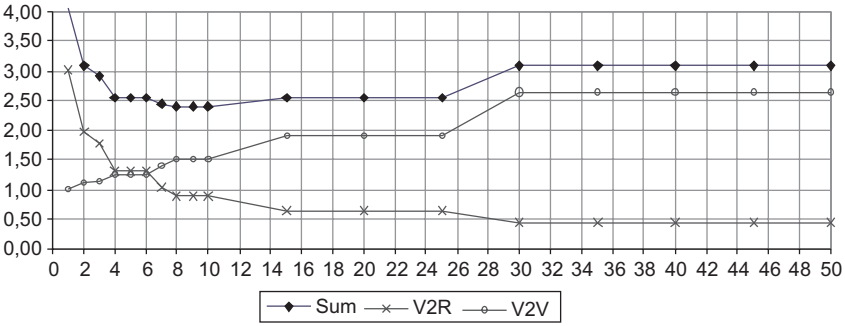

Рис. 5. Зависимость удельного числа элементарных операций измерения расстояний $\mathrm{Sum}(f)$, $\mathrm{V2R}(f)$ и $\mathrm{V2V}(f)$ от емкости терминальной вершины f, $k = 100$, $d = 3$

значения f из субоптимального интервала дает почти оптимальное (минимум удельного числа измерений расстояния) решение для различных изображений. Этот интервал с ростом k плавно смещается в сторону увеличения, однако исследования на образах различных документов с количеством исходных точек данных в широком диапазоне $(10^2–10^5)$ показало, что в рассматриваемом классе прикладных задач оптимальное значение f лежит в диапазоне 4–8.

На следующем графике (рис.6) показаны результаты измерения зависимость общего удельного числа элементарных операций измерения

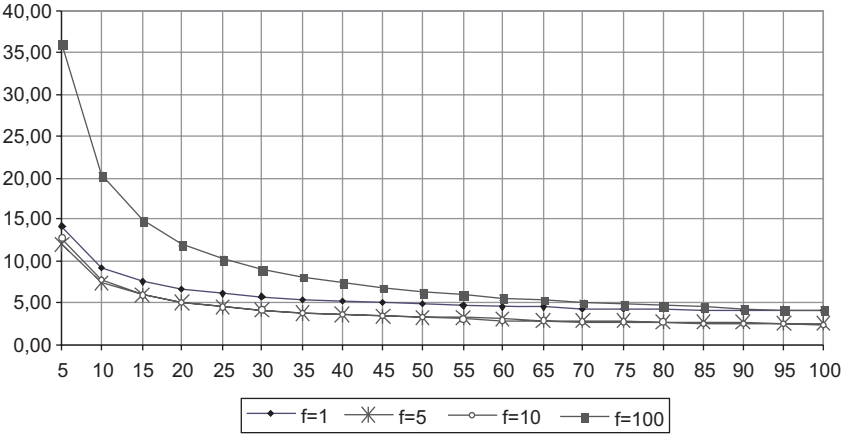

Рис. 6. Зависимость удельного числа элементарных операций измерения расстояний $\mathrm{Sum}(k)$ от числа соседей k при различных значениях емкости терминальной вершины f, $d = 3$

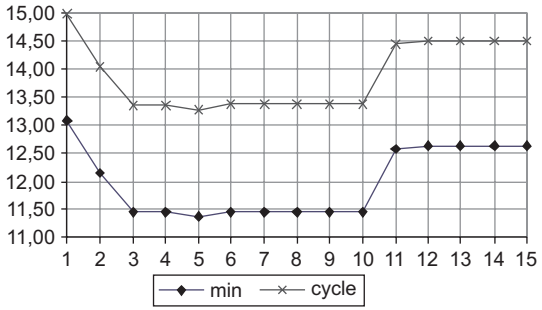

Рис. 7. Зависимость удельного числа элементарных операций измерения расстояний $\mathrm{Sum}(k)$ от емкости терминальной вершины f для разных вариантов выбора опорной оси, $k = 5$, $d = 3$

расстояний $\mathrm{Sum}(k)$ от числа соседей k при различных значениях емкости терминальной вершины f.

Другим параметром алгоритмов, влияющим на результат и слабо исследованным теоретически, является выбор способа построения R-дерева. Для примера сравним следующие правила выбора опорной оси:

- min — выбор оси вдоль самого длинного ребра МОП;
- cycle — поочередный циклический выбор одной из координатных осей.

График, изображенный на рис.7, является примером общего наблюдения, что характер изучаемых зависимостей сохраняется для различных вариантов построения R-дерева.

3.8. Обсуждение и дальнейшая работа

Алгоритмы быстрого поиска ближайших соседей с помощью вспомогательных иерархических структур данных позволяет строить эффективные метода анализа изображения структурированного документа. Выбор разных вариантов построения R-дерева приводит к реализациям поиска ближайших соседей с разными характеристиками. Для достижения максимальной эффективности способ построения R-дерева должен быть согласован с используемой функцией расстояния, а так же учитывать особенности обрабатываемого класса изображений. Теоретическое обоснование выбора конкретного алгоритма и структуры данных, а так же оценка средних и худших временных характеристик для произвольно функции близости остаются открытыми вопросами.

Мы видим дальнейшие развитие данного подхода в разработке единого теоретического и практического подхода к выбору оптимального

сочетания функции оценки расстояния, алгоритма поиска, вспомогательной структуры данных и ее ключевых параметров. Другим направлением дальнейших исследований является использование динамического поиска ближайших соседей для анализа изображения структурированного документа.

4. Заключение

В данной работе мы рассмотрели метод поиска ближайших соседей с помощью R-деревьев. Особенности эффективного использования этого метода в задаче анализа графического образа структурированного документа исследовали на примере изображений реальных документов.

Программная реализация методов анализа изображения с использованием алгоритмов быстрого поиска ближайших соседей вошла в состав системы массового ввода стандартных форм документов Cognitive Forms [20], которая широко используется для ввода стандартных форм документов.

Литература

1. *Кормен Т.*, *Лейзерсон Ч.*, *Ривест Р.* Алгоритмы: построение и анализ. М.: МЦНМО, 2001.
2. *Dobkin D.*, *Lipton R. J.* Multidimensional searching problems // SIAM Journal of Computing, 5(2). 1976. P. 181–186.
3. *Clarkson K.* A randomized algorithm for closest-point queries // SIAM Journal of Computing. 1988. V. 17. P. 830–847.
4. *Meiser S.* Point location in arrangements of hyperplanes. Information and Computation. 1993. V. 106(2). P. 286–303.
5. *Agarwal P. K.*, *Mutusek J.* Ray shooting and parametric search // In Proccedings of the 24^{th} Symposium on Theory of Computing. 1992. P. 517–526.
6. *Кляцкин В. М.*, *Котович Н. В.* Применение методов вычислительной геометрии для поиска линейных объектов // В сборнике трудов ИСА РАН «Управление информационными потоками». 2002.
7. *Samet H.* The Quadtree and Related Hierarchical Data Structures, ACM Computing Surveys. 1984. V. 16(2). P. 187–260.
8. *Friedman J. H.*, *Bently J. L.*, *Finkel R. A.* An algorithm for finding best matches in logarithmic expected time // In ACM Transaction on Mathematical Software. 1977. V. 3. P. 209–226.
9. *Guttman A.* R-tree: a dynamic index structure for spatial searching // Proceedings of the ACM SIGMOD International Conference on Management of Data. 1984. P. 71–79.

10. *Roussopoulos N.*, *Kelley S. and Vincent F.* Nearest Neighbor Queries // Proceedings of the ACM SIGMOD International Conference on Management of Data. 1995. P. 71–79.
11. *Bently J. L.* Multidimensional binary search trees used for associative searching. Commun. ACM 18, 9 (Sept.). 1975. P. 509–517.
12. *Hjaltason G. R.*, *Samet H.* Ranking in Spatial Databases // Proceedings of the 4^{th} International Symposium on Large Spatial Databases. 1995. P. 83–95.
13. *Hjaltason G. R.*, *Samet H.* Distance Browsing in Spatial Databases // ACM Trans. Database Systems. 1999. V. 24(2). P. 265–318.
14. *Yannis Manolopoulos*, *Alexandros Nanopoulos*, *Apostolos N. Papadopoulos*, *Yannis Theodoridis.* R-trees Have Grown Everywhere. Unpublished technical report, 2003.
15. *Cheung K. and Fu A.* Enhanced nearest neighbour search on the r-tree // In ACM SIGMOD Record. 1998. V. 27:3. P. 16–21.
16. *Apostolos Papadopoulos*, *Yannis Manolopoulos.* Performance of Nearest Neighbor Queries in R-Trees Source // Proceedings of the 6th International Conference on Database Theory. 1997. P. 394–408.
17. *Berchtold S. Bohm C.*, *Keim D.*, *Krebs F. and Kriegel H. P.* On optimizing nearest neighbor queries in high-dimensional data spaces // Proc. 8^{th} ICDT Conference. 2001. P. 435–449.
18. *Tao Y.*, *Zhang J.*, *Papadias D.*, *Mamoulis*, *N.* An Efficient Cost Model for Optimization of Nearest Neighbor Search in Low and Medium Dimensional Spaces. IEEE Transactions on Knowledge and Data Engineering (TKDE). 2004. V. 16(10). P. 1169–1184.
19. *Полевой Д. В.*, *Постников В. В.*, *Усков А. В.* Алгоритм быстрого построения минимального охватывающего дерева для множества точек в конечномерном псевдометрическом пространстве // В сборнике трудов ИСА РАН. «Интеллектуальные информационные технологии. Концепции и инструментарий». 2005. Т. 16.
20. *Полевой Д. В.*, *Постников В. В.* Сегментация графического образа документа с использованием алгоритма MSTKD.
21. *Арлазаров В. В.*, *Постников В. В.*, *Шоломов Д. Л.* Cognitive Forms — система массового ввода структурированных документов // В сборнике трудов ИСА РАН «Управление информационными потоками». 2002.