ГЛОБАЛЬНЫЕ АЛГОРИТМЫ ПОСТРОЕНИЯ R-ДЕРЕВЬЕВ*

А.В. Скворцов

1. Введение

Задача регионального поиска является одной из базовых задач вычислительной геометрии. Так как форма различных объектов может значительно различаться, то для упрощения алгоритмов регионального поиска обычно для каждого из объектов вычисляется минимальный объемлющий прямоугольник со сторонами, параллельными осям координат. Тогда задача регионального поиска определяется как задача нахождения всех прямоугольных объектов, имеющим непустое перекрытие с прямоугольным регионом, являющимся запросом.

В настоящее время существует множество структур для организации данных, позволяющих решать данную задачу регионального поиска в той или иной мере. При этом наиболее эффективной структурой, позволяющей обрабатывать неточечные объекты при эффективном размещении данных во вторичной памяти, считаются R-деревья [1] и ряд их модификаций, в частности, R*-деревья [2]. Практическое сравнение эффективности различных вариантов R-деревьев проведено в работе [3]. Наиболее быстродействующими из них являются R*-деревья, поэтому в дальнейшем в данной работе будет использоваться именно эта разновидность R-деревьев.

Индексная структура R-дерева является сбалансированным по высоте деревом с индексными записями в листьях, содержащими указатели на объекты данных. Узлам дерева соответствуют дисковые страницы, если индекс располагается на диске. Структура является полностью динамической, вставка и удаление объектов может свободно перемежаться с запросами на поиск, при этом не требуется периодической реорганизации данных.

Данная индексная структура предназначена для эффективного выполнения регионального поиска, поэтому основная идея дерева заключается в разбиении пространства в каждом узле дерева на минимально пересекающиеся группы, что позволяет при выполнении поиска эффективно отсекать неверные ветви поиска. Задача разбиения в общем случае, видимо, по меньшей мере NP-полна, так как в настоящее время даже для двух групп не известно полиномиального алгоритма [1]. Поэтому во всех алгоритмах построения R-деревьев используются эвристические подходы, дающие на реальных данных далеко не оптимальные разбиения.

Высокая степень перекрытия входов в узлах дерева происходит из-за динамичности алгоритмов его построения, когда при вставке очередного элемента его необходимо за минимальное время разместить и при необходимости пере-

* Работа выполнена при финансовой поддержке РФФИ (грант № 98-05-03194)

строить дерево. При этом глобальная оптимизация дерева на каждом шаге не производится.

На практике взаимодействие с индексными структурами состоит из двух основных этапов: начального построения дерева и последующей оперативной работы. При этом во многих случаях во время второго этапа производится только поиск объектов без вставки новых и удаления старых.

В данной работе предлагается учесть специфику начального построения структуры, построив вначале более эффективное R-дерево, а на последующих этапах использовать обычные алгоритмы для работы с R-деревьями. В частном случае, когда вначале ничего не строится, получается обычное R-дерево.

2. Глобальные алгоритмы

Во всех существующих структурах R-деревьев основным параметром, определяющим скорость поиска, считается степень взаимного пересечения потомков в узлах дерева [1,2]. Именно поэтому в алгоритмах построения R-деревьев одной из главных целей является минимизация такого пересечения. Это позволяет при выполнении регионального поиска на ранних стадиях эффективно отсекать тупиковые ветви работы алгоритма.

В связи с этим можно ввести следующее определение:

Определение. Построенная для заданного набора объектов структура R-дерева называется *оптимальной*, если сумма площадей попарных перекрытий входов во всех нелистовых узлах дерева является минимальной среди всех возможных структур R-деревьев:

$$\min_{R\text{-деревья}} \sum_{i \neq j} S(R_i \cap R_j),$$ где S – функция площади.

Автором предлагается следующая стратегия построения R-дерева, близкого к оптимальному.

Алгоритм *Построить Дерево.* По заданному множеству из *N* объектов E_i строит R-дерево с параметрами *m, M* [1].

Шаг 1. *Начало*. Построить корень дерева и определить количество уровней дерева по формуле $L(N) = \lceil \log_M N \rceil$.

Шаг 2. *Построение дерева.* Вызвать алгоритм **Построить_Узел**, передав ему в качестве параметра корень дерева и всё множество объектов.

Алгоритм *Построить Узел.* По заданному множеству из *N* объектов E_i и узлу *T* строит потомков узла, имеющего уровень *L* (*L*=1 для листьев).

Шаг 1. *Построение листьев.* Если L=1, то заполнить узел объектами E_i и закончить.

Шаг 2. *Выбор количества потомков узла.* Вычислить количество потомков у данного узла $K = \left\lceil N^{1/L} \right\rceil$.

Шаг 3. *Построение узлов.* Вызвать алгоритм **Разделить_На_Группы** для разбиения всего множества объектов на K групп, минимизируя сумму попарных пересечений групп. При этом в алгоритм необходимо передать номер уровня L для правильной оценки минимального и максимального допустимого количества объектов в группах. Для каждой полученной группы построить пустой узел – потомок текущего узла, и вызвать для него рекурсивно алгоритм **Построить_Узел** с этой группой в качестве множества.

Наиболее сложной частью предложенной стратегии является алгоритм **Разделить_На_Группы**, от трудоёмкости которого зависит общая сложность алгоритма. В работе предлагается 3 варианта данного алгоритма.

3. Алгоритмы разбиения множества объектов на минимально пересекающиеся группы

Во все существующие алгоритмы построения R-деревьев входят алгоритмы разбиения множества объектов на две части с минимальным пересечением. При этом выбирается два базовых объекта, последовательными присоединениями к которым всех остальных строятся необходимые группы.

Обобщая данный алгоритм на случай нескольких групп, можно выбрать по одному базовому объекту для каждой группы и, используя аналогичные итеративные алгоритмы добавления оставшихся объектов, произвести разбиение. На этом принципе построен следующий базовый алгоритм разбиения.

Алгоритм *Разделить На Группы* (базовый алгоритм). Заданное множество из N объектов E_i делит на K групп с соблюдением ограничения на количество объектов в группах снизу m^{L-1} и сверху M^{L-1}, где L – текущий строящийся уровень дерева.

Шаг 1. *Выбор базовых объектов для групп.* Разбить всё множество объектов вертикальными и горизонтальными равноотстоящими линиями на K групп. После этого в качестве базовых для каждой группы выбрать по одному любому из объектов, попадающих в построенные клетки. Если в некоторые клетки не попадает ни одного объекта, то необходимо раз-

делить любую из клеток с более чем двумя объектами пополам и взять из получившихся частей по объекту. Если базовых объектов по-прежнему недостаточно, то процесс деления клеток повторить. В качестве текущего размера групп взять размер соответствующих им клеток (на рис. 1 приведён пример разбиения и выбора базовых объектов при K=10).

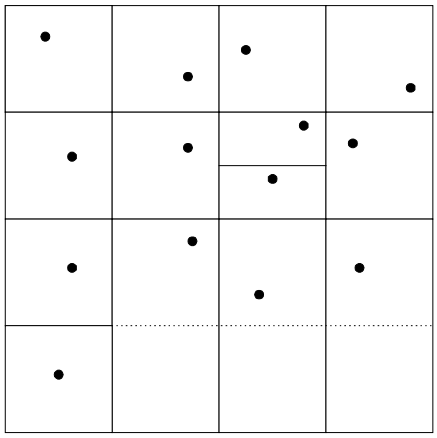

Рис. 1. Разбиение на клетки и выбор базовых объектов.

Шаг 2. *Построение групп.* Все оставшиеся объекты (не базовые) поместить в образовавшиеся группы по принципу минимального увеличения их площадей. При этом добавление объекта в группу не должно привести к числу объектов в группе больше чем M^{L-1}, а также к ситуации, когда оставшихся объектов недостаточно для обеспечения минимально допустимого числа объектов в других группах m^{L-1}.

Одним из недостатков работы такого алгоритма является недостаточно высокое качество разбиения на неравномерных распределениях, а также в случае, когда N незначительно превосходит K (при построении нижних уровней дерева). Это проявляется в ситуациях, когда на втором шаге работы алгоритма некоторые группы уже заполнены, но при этом в соответствии с критерием выбора в них необходимо добавить очередной объект. Но так как это невозможно, то приходится добавлять объекты в некоторые другие группы, что в конечном итоге приводит к сильному перекрытию образовавшихся групп.

В качестве одного из вариантов решения данной проблемы автором предлагается временно удалять объекты, которые нельзя поместить в требуемые группы, из списка участвующих в глобальном алгоритме. При этом после окончания его работы все эти объекты необходимо поместить в дерево, используя уже обычные динамические алгоритмы построения R-дерева.

Как показало проведённое автором экспериментальное исследование работы данного алгоритма, количество объектов, которые нельзя разместить в требуемые группы, чаще всего незначительно и они эффективно размещаются по окончании работы глобального алгоритма динамическим способом.

В следующей теореме устанавливается трудоёмкость работы глобального алгоритма построения R-дерева, использующего предложенный базовый алгоритм разбиения.

Теорема 1 (трудоёмкость базового глобального алгоритма построения R-дерева). Пусть даны N объектов, на которых необходимо построить R-дерево. Тогда трудоёмкость глобального алгоритма построения R-дерева, использующего базовый алгоритм разбиения, составляет $O(N \log N)$.

Доказательство. Вначале установим трудоёмкость работы базового алгоритма разбиения. Сложность работы первого шага этого алгоритма состоит из сложности разбиения на клетки, которая составляет линейное время, а также сложности деления клеток, когда базовых объектов недостаточно. Если при этом в качестве базовых объектов случайно выбрать любые другие объекты, не являющиеся базовыми, то трудоёмкость всего первого шага будет $O(1)$.

Трудоёмкость работы второго шага алгоритма является не более чем линейной, так как в цикле по всем оставшимся объектам (не базовым) происходит сравнение с не более чем K группами, а $K \le M = \text{const}$.

Таким образом, общая трудоёмкость базового алгоритма разбиения составляет $O(N)$.

Для глобального алгоритма построения R-дерева трудоёмкость работы всего алгоритма можно записать следующим образом:

$$S(N) = T(N) + K \cdot S(N / K), \tag{1}$$

где $S(N)$ – общая трудоёмкость глобального построения R-дерева; $T(N)$ – трудоёмкость алгоритма разбиения множества объектов.

Если расписать формулу общей трудоёмкости, то получится

$$S(N) = T(N) + K \cdot T(N / K) + K^2 \cdot T(N / K^2) + \ldots + K^{\lceil \log_K N \rceil - 1} =$$

$$= \sum_{i=0}^{\lceil \log_K N \rceil - 1} K^i \cdot T(N / K^i). \tag{2}$$

Так как в случае базового алгоритма $O(T(N)) = O(N)$, то

$$O(S(N)) = O\left(\sum_{i=0}^{\lceil \log_K N \rceil - 1} K^i \cdot N / K^i \right) =$$

$$= O\left(N \cdot \lceil \log_K N \rceil\right) = O\left(N \cdot \log N\right), \tag{3}$$

что и требовалось доказать.

Как показало экспериментальное исследование, такой алгоритм хорошо ведёт себя на равномерных распределениях непересекающихся объектов. Но на неравномерных распределениях часто происходит переполнение строящихся групп, что приводит к необходимости помещения очередных объектов в неоптимальные группы. В итоге получаются недопустимо большие перекрытия построенных групп. Поэтому в работе предлагается другой алгоритм, в котором вначале на основе клеточного разбиения плоскости строятся группы объектов и

только затем анализируются их количество и наполнение. В случае необходимости полученные группы делятся пополам или объединяются с другими.

Алгоритм *Разделить На Группы* (клеточный алгоритм). Заданное множество из N объектов E_i делит на K групп с соблюдением ограничения на количество объектов в группах снизу m^{L-1} и сверху M^{L-1}, где L – текущий строящийся уровень дерева.

Шаг 1. *Построение базовых групп.* Разбить всё множество объектов вертикальными и горизонтальными линиями на $\left\lceil \sqrt{K} \right\rceil^2$ одинаковых клеток. После этого все объекты распределить по группам, соответствующим полученным клеткам, по принципу попадания центра объектов в построенные прямоугольники (рис. 2).

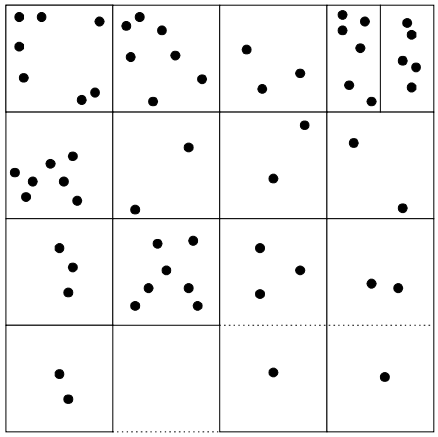

Рис. 2. Клеточное разбиение

Шаг 2. *Уничтожение неполных групп.* Для каждой группы G_0, в которой количество объектов меньше m^{L-1}, выполнить слияние с некоторой другой группой G_i. При этом необходимо выбрать такую группу G_i для слияния, чтобы площадь минимального объемлющего группы G_0 и G_i прямоугольника была минимальной среди всех возможных групп G_i.

Шаг 3. *Уничтожение лишних групп.* Пока общее количество групп больше K, выполнять следующее. Группу с минимальной площадью уничтожить, передав её объекты в некоторую другую группу по критерию, описанному в шаге 2.

Шаг 4. *Создание недостающих групп.* Пока общее количество групп меньше K, выполнять следующее. Группу с максимальным количеством объектов необходимо уничтожить, распределив её объекты по двум новым группам с минимальным пересечением полученных групп с помощью алгоритма **Расщепить_Узел**, входящего в состав классических алгоритмов для манипуляции с R-деревьями [1,2].

Шаг 5. *Распределение переполненных групп.* Для каждой группы G_0, в которой количество объектов превышает M^{L-1}, выполнять шаг 6.

Шаг 6. *Перемещение объектов по группам*. Выбрать ближайшую (например, в смысле расстояний между центрами) к группе G_0 группу G_i, в которой количество объектов меньше M^{L-1}. После этого выбрать из группы G_0 такой объект g, что перемещение его в группу G_i вызовет минимальное увеличение площади последней группы. Переместить найденный объект g из группы G_i в G_0.

Одним из недостатков работы такого алгоритма является увеличение времени работы алгоритма на неравномерных распределениях, так как при этом часто происходит недополнение и переполнение групп, а также приходится выполнять перемещение объектов по группам. В худшем случае алгоритм может иметь квадратичную сложность в случае многократного перемещения объектов по группам в пятом шаге.

В следующей теореме устанавливается трудоёмкость работы глобального алгоритма построения R-дерева, использующего предложенный автором клеточный алгоритм разбиения.

Теорема 2 (трудоёмкость клеточного глобального алгоритма построения R-дерева). Пусть даны N объектов, на которых необходимо построить R-дерево. Тогда трудоёмкость глобального алгоритма построения R-дерева, использующего клеточный алгоритм разбиения, составляет $O(N^2)$.

Доказательство. Вначале установим трудоёмкость работы алгоритма клеточного разбиения. Его трудоёмкость состоит из трудоёмкости работы шагов 1-4 алгоритма, которая линейна ($T_{1-4}(N) = O(N)$), что следует из описания алгоритма при условии использования линейного по трудоёмкости алгоритма **Расщепить_Узел** [1], а также из трудоёмкости работы шагов 5-6. При выполнении шага 5 возможна ситуация, когда во всех группах, кроме одной, находится минимально допустимое количество объектов m^{L-1}. Если при этом необходимо разбить всё множество объектов на группы по M^{L-1} объектов (в случае, если $N = K \cdot M^{L-1}$), то в одной из групп находится $K \cdot M^{L-1} - (K-1) \cdot m^{L-1}$ объектов. Поэтому на шестом шаге алгоритма будет перемещено всего $R(N) = (K-1) \times \times (M^{L-1} - m^{L-1})$ объектов. Так как $m \le \frac{M}{2}$ и $N \le K \cdot M^{L-1}$, то общее количество перемещаемых объектов становится в худшем случае:

$$R(N) = (K-1) \cdot (M^{L-1} - m^{L-1}) \ge (K-1) \cdot (M^{L-1} - \frac{M}{2}^{L-1}) =$$

$$= (K-1) \cdot \frac{M^{L-1}}{2} = \frac{K-1}{2K} K \cdot M^{L-1} \ge \frac{K-1}{2K} N \,. \qquad (4)$$

С другой стороны, так как $N > (K-1)\cdot M^{L-1}$, то

$$R(N) = (K-1)\cdot (M^{L\text{-}1} - m^{L\text{-}1}) \le (K-1)\cdot M^{L\text{-}1} < N .$$

Таким образом, $O(R(N)) = O(N)$. А так как в шестом шаге для каждого перемещаемого объекта выполняется цикл по некоторой группе G_i, то общая трудоёмкость работы шагов 5-6 может составлять:

$$O(T_{5-6}(N)) = O(R(N)\cdot M^{L\text{-}1}) = O\left(R(N)\cdot \frac{1}{K} K\cdot M^{L\text{-}1}\right) =$$

$$= O(N)\cdot O(K\cdot M^{L\text{-}1}) = O(N^2). \qquad (5)$$

Таким образом, общая трудоёмкость клеточного алгоритма разбиения составляет $O(T(N)) = O(T_{1-4}(N)) + O(T_{5-6}(N)) = O(N) + O(N^2) = O(N^2)$.

Для установления общей трудоёмкости клеточного глобального алгоритма построения R-дерева воспользуемся формулой (2):

$$O(S(N)) = O(\sum_{i=0}^{\lceil \log_K N \rceil - 1} K^i \cdot T(N/K^i)) = O(\sum_{i=0}^{\lceil \log_K N \rceil - 1} K^i \cdot O(N^2/K^{2i})) =$$

$$= \sum_{i=0}^{\lceil \log_K N \rceil - 1} O(N^2/K^i) = O(N^2\cdot(1 + \frac{1}{K} + \frac{1}{K^2} + \ldots + \frac{1}{K^{\lceil \log_K N \rceil - 1}})) =$$

$$= O(N^2 \cdot F(\mathrm{K})).$$

Так как $2 \le m \le K$, то

$$F(K) = 1 + \frac{1}{K} + \frac{1}{K^2} + \ldots + \frac{1}{K^{\lceil \log_K N \rceil - 1}} \le 1 + \frac{1}{K} + \frac{1}{K^2} + \ldots =$$

$$= \frac{1}{1-\frac{1}{K}} \le \frac{1}{1-\frac{1}{2}} = 2.$$

С другой стороны, $F(K) = 1 + \frac{1}{K} + \frac{1}{K^2} + \ldots + \frac{1}{K^{\lceil \log_K N \rceil - 1}} \ge 1$. Поэтому

$$O(S(N)) = O(N^2 \cdot F(\mathrm{K})) = O(N^2), \qquad (6)$$

что и требовалось доказать.

Теперь рассмотрим другой вариант алгоритма разбиения, основанный на стратегии «Разделяй и властвуй», в котором производится последовательное разбиение всего множества на две части до тех пор, пока всё множество не окажется разбито на требуемое число частей:

Алгоритм *Разделить На Группы* (алгоритм «Разделяй и властвуй»). Заданное множество из N объектов E_i делит

на K групп с соблюдением ограничения на количество объектов в группах снизу m^{L-1} и сверху M^{L-1}, где L – текущий строящийся уровень дерева.

Шаг 1. *Разделение на две группы*. Вызвать алгоритм **Разделить_На_Две_Группы**, передав в качестве аргументов всё множество объектов E_i и соотношение разделения ([K/2]):(K-[K/2]) для получения групп G_1 и G_2 (рис. 3).

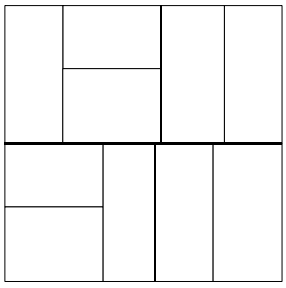

Рис. 3. «Разделяй и властвуй».

Шаг 2. *Рекурсивное разделение*. Если [K/2]>1, то вызвать алгоритм **Разделить_На_Группы**, передав в качестве аргументов полученные группу G_1 с параметром деления [K/2], иначе выдать в качестве результата G_1.

Если K-[K/2]>1, то вызвать алгоритм **Разделить_На_Группы**, передав в качестве аргументов полученную группу G_2 с параметром деления K-[K/2], иначе выдать в качестве результата G_2.

Алгоритм *Разделить На Две Группы.* Заданное множество из N объектов E_i делит на 2 группы в заданном соотношении площадей k:l с соблюдением ограничения на количество объектов снизу $k \cdot m^{L-1}$ и сверху $k \cdot M^{L-1}$ для первой группы, а также снизу $l \cdot m^{L-1}$ и сверху $l \cdot M^{L-1}$ для второй, где L – текущий строящийся уровень дерева.

Данный алгоритм аналогичен алгоритму *Расщепить_Узел* обычного R-дерева, за исключением проверок, при которых сравниваемые значения взвешиваются по величинам k и l.

Шаг 1. *Выбор оси координат для сортировки*. Выбрать такую ось координат i, на которой достигается максимальное значение выражения

$$\max_{\text{оси } i}\left(\frac{\max\limits_{\text{фигуры } j} R_a^{j,i} - \min\limits_{\text{фигуры } j} R_b^{j,i}}{\max\limits_{\text{фигуры } j} R_b^{j,i} - \min\limits_{\text{фигуры } j} R_a^{j,i}}\right),$$

где $R_a^{j,i}$ и $R_b^{j,i}$ – соответственно меньшее и большее значение i-й координаты минимального объемлющего j-ю фигуру прямоугольника.

Шаг 2. *Разделение по группам*. Разделить все объекты на три части по значению $P^i(j) = (R_a^{j,i} + R_b^{j,i})/2$ для выбранного i по квантилям уровня α и $1-\alpha$ ($\alpha \in [0; 0{,}5]$ – параметр разделения по группам): из наименьшей но значениям части образовать первую группу объектов, а из наибольшей – вторую группу. Оставшиеся объекты распределить по группам в шагах 3 и 4.

Шаг 3. *Проверка завершения*. Если все объекты распределены, то закончить. Если одна из групп в результате дальнейших операций будет недополнена (количество объектов в группах $p_1 < k \cdot m^{L-1}$ или $p_2 < l \cdot m^{L-1}$), то вставить в неё все оставшиеся объекты и закончить. Если одна из групп в результате дальнейших операций будет переполнена (количество объектов в группах $p_1 > k \cdot M^{L-1}$ или $p_2 > l \cdot M^{L-1}$), то вставить в другую группу все оставшиеся объекты и закончить.

Шаг 4. *Распределение объектов*. Взять любой нераспределённый объект и поместить его в группу, размер которой увеличится в минимальной степени при взвешивании по k и l. Перейти на шаг 3.

В следующей теореме устанавливается трудоёмкость работы глобального алгоритма построения R-дерева, использующего предложенный алгоритм разбиения «Разделяй и властвуй».

Теорема 3 (трудоёмкость глобального алгоритма построения R-дерева «Разделяй и властвуй»). Пусть даны N объектов, на которых необходимо построить R-дерево. Тогда трудоёмкость глобального алгоритма построения R-дерева, использующего алгоритм разбиения «Разделяй и властвуй», составляет $O(N \cdot \log^2 N)$.

Доказательство. Вначале установим трудоёмкость работы алгоритма разбиения «Разделяй и властвуй». Его трудоёмкость можно записать следующим образом:

$$T(N) = R(N) + 2 \cdot T(N/2), \tag{7}$$

где $R(N)$ – трудоёмкость алгоритма разбиения множества объектов на две части.

В алгоритме разделения на две части наиболее трудоёмким этапом является второй шаг, в котором производится разделение множества объектов на три части по заданным квантилям. Если использовать линейные по сложности алгоритм разделения по квантилям Хоара или алгоритм цифровой сортировки,

то получается $O(R(N))=O(N)$, так как другие шаги обсуждаемого алгоритма не более чем линейны, что следует из описания алгоритма.

Таким образом, общая трудоёмкость алгоритма разбиения «Разделяй и властвуй» составляет

$$O(T(N))=O(N)+2\cdot O(T(N/2))=$$
$$=O(N)+2\cdot(O(N/2)+2\cdot O(T(N/4)))=$$
$$=O(N)+2\cdot(O(N/2)+2\cdot(O(N/4)+2\cdot O(T(N/8))))=\ldots=$$
$$=O(N)\cdot O(\lceil\log_2 N\rceil)=O(N\log N). \quad (8)$$

Для установления общей трудоёмкости глобального алгоритма построения R-дерева «Разделяй и властвуй» воспользуемся формулой (2):

$$O(S(N))=O\left(\sum_{i=0}^{\lceil\log_K N\rceil-1} K^i\cdot T(N/K^i)\right)=$$
$$=O\left(\sum_{i=0}^{\lceil\log_K N\rceil-1} K^i\cdot O(N/K^i\cdot\log(N/K^i))\right)=$$
$$=\sum_{i=0}^{\lceil\log_K N\rceil-1} O(N\cdot\log N)=O(N\cdot\log N)\cdot O(\lceil\log_K N\rceil)=$$
$$=O(N\cdot\log^2 N), \quad (9)$$

что и требовалось доказать.

На втором шаге работы алгоритма разделения «Разделяй и властвуй» присутствует параметр α разделения по группам. Для установления его влияния на качество получаемого разбиения автором было проведено экспериментальное моделирование работы глобального алгоритма построения R-дерева, использующего алгоритм разбиения «Разделяй и властвуй». При этом было установлено, что уменьшение значения α приводит к некоторому улучшению структуры R-дерева на неравномерных распределениях и распределениях со значительным перекрытием объектов. С другой стороны, это приводит к снижению скорости работы алгоритма и уменьшению качества разбиения на точечных и регулярных наборах данных.

Суммируя достоинства и недостатки поведения алгоритма при разных значениях α на различных распределениях объектов, автором было выбрано эмпирическое значение параметра разделения для универсального применения, равное 0,45. При этом если на практике имеются некоторые сведения о реальном распределении объектов, то значение параметра α, вероятно, стоит оценить дополнительно в зависимости от требований конкретной задачи.

4. Уточнение разбиения

Повышение качества получаемого разбиения множества объектов на группы является главным направлением совершенствования алгоритмов работы с R-деревьями. Поэтому для повышения качества разбиения множества объектов на группы предлагается алгоритм уточнения полученных групп, позволяющий в ряде случаев значительно улучшить качество получаемого разбиения. Такой подход может использоваться как в обычных динамических алгоритмах манипуляции с R-деревьями (в алгоритме **Расщепить_Узел** при вставке объектов в дерево), так и в алгоритмах разбиения, применяемых в глобальных алгоритмах. Для реализации данного подхода предлагается следующий алгоритм уточнения разбиения:

Алгоритм *Уточнить Разбиение*. Уточняет заданное разбиение из N объектов E_i на K групп с соблюдением ограничения на количество объектов в группах снизу m^{L-1} и сверху M^{L-1}, где L – текущий строящийся уровень дерева.

Шаг 1. *Выбор группы с объектом для перемещения*. Выбрать группу G_0 максимального размера, у которой количество объектов больше m^{L-1}.

Шаг 2. *Выбор объекта для перемещения*. Выбрать из группы G_0 максимальный по площади объект g, попадающий целиком внутрь какой-либо другой группы G_i (не касается границы группы). Если таких объектов нет, то выбрать из группы G_0 объект, перемещение которого в некоторую другую группу G_i вызывает минимальное увеличение площади группы G_i. При выборе G_i проверять только те группы, в которых число объектов меньше M^{L-1}.

Шаг 3. *Проверка завершения*. Если на втором шаге не выбрано ни одного объекта либо выбран объект, который уже перемещался в предыдущей итерации, то закончить работу алгоритма.

Шаг 4. *Перемещение объекта*. Переместить выбранный объект g из группы G_0 в группу G_i. Вычислить минимальные объемлющие группы прямоугольники.

Шаг 5. *Переход на следующую итерацию*. Если ещё выполнено не более M итераций, то перейти на шаг 1 для выполнения следующей, иначе закончить работу алгоритма.

В следующей теореме устанавливается трудоёмкость работы алгоритма уточнения разбиения.

Теорема 4 (трудоёмкость алгоритма уточнения). Трудоёмкость алгоритма уточнения разбиения составляет $O(N)$.

Доказательство. В соответствии с условием цикла на шаге 5 всего может быть выполнено M операций перемещения объектов. Внутри цикла на втором шаге может выполняться перебор из $M^{L-1} \cdot K \approx N$ объектов, а на четвёртом – пересчёт прямоугольников, требующий K операций. Таким образом, весь цикл локального уточнения имеет трудоёмкость

$$O(S(N)) = O(M \cdot N) = O(N), \tag{10}$$

что и требовалось доказать.

Следствием данной теоремы является то, что порядок трудоёмкости алгоритмов построения R-деревьев, использующих алгоритм уточнения, не изменяется по сравнению с алгоритмами без уточнения.

5. Практический анализ глобальных алгоритмов

Для количественного анализа построенных алгоритмов автором было проведено моделирование работы различных алгоритмов построения R-деревьев на различных распределениях объектов и поисковых регионах. При этом анализировались такие параметры, как [1]:

1. *Время построения* R-дерева.
2. *Процент попадания при поиске*. Отношение количества узлов, в которых найден хотя бы один объект, к их общему количеству, проверенному при выполнении регионального поиска. Все объекты были распределены в единичном квадрате, центр региона поиска (прямоугольника) выбирался случайно равномерно и независимо в единичном квадрате, его размеры – также равномерно и независимо в [0; 0,02).
3. *Процент перекрытия потомков*. Отношение суммы площадей попарных пересечений входов в узлах к общей площади всех входов в процентах.

Для сравнения алгоритмов в различных условиях в экспериментах строились наборы данных с 10 000 объектами, располагавшимися в единичном квадрате $[0,1)^2$, со следующими характеристиками (базовые типы распределений взяты из работы [2]):

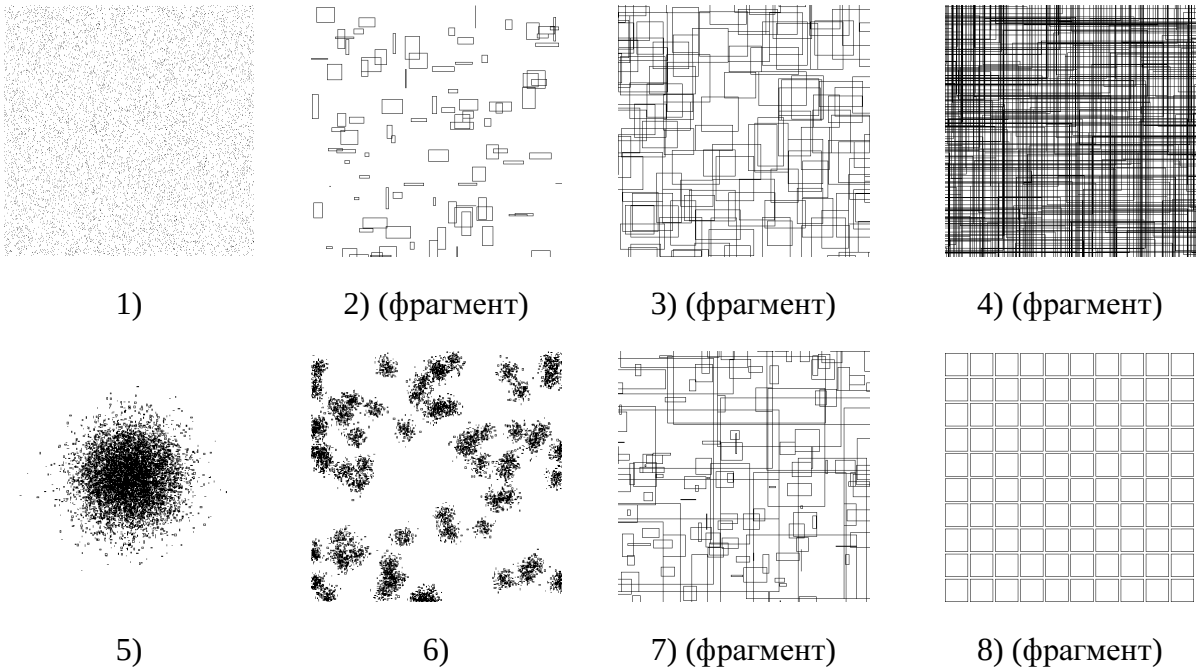

Рис. 4. Примеры тестовых данных для моделирования работы алгоритмов построения R-деревьев.

1. *Точечное распределение*. В этом наборе данных все объекты имели нулевые размеры, при этом их положение выбиралось в единичном квадрате равномерно и независимо по обеим координатам (рис. 4-1).
2. *Равномерное распределение (малые объекты)*. Центры всех объектов были распределены равномерно и независимо в единичном интервале. Линейные размеры выбирались так же случайно и независимо в интервале $[0; 0{,}01)$ (рис. 4-2).
3. *Равномерное распределение (средние объекты)*. Центры всех объектов были распределены равномерно и независимо в единичном интервале. Линейные размеры выбирались так же случайно и независимо в интервале $[0{,}01; 0{,}02)$ (рис. 4-3).
4. *Равномерное распределение (большие объекты)*. Центры всех объектов были распределены равномерно и независимо в единичном интервале. Линейные размеры выбирались так же случайно и независимо в интервале $[0{,}05; 0{,}1)$ (рис. 4-4).
5. *Нормальное распределение*. Центры всех объектов размещались в единичном квадрате по нормальному распределению с центром в точке (0,5; 0,5) и среднеквадратическим отклонением 0,1. Точки, попадавшие за пределы единичного квадрата, отбрасывалась. Размеры объектов выбирались случайно равномерно и независимо в интервале $[0; 0{,}01)$ (рис. 4-5).

6. *Кластерное распределение*. Внутри единичного квадрата выбирались случайно (равномерно и независимо по обеим координатам) 64 точки, которые становились центрами кластеров, содержащих в среднем 156 объектов. Внутри кластеров точки распределялись по двумерному нормальному распределению со среднеквадратическим отклонением 0,02 (рис. 4-6).
7. *Смешанное распределение*. Данная выборка состояла из двух частей размерами 9000 и 1000, в каждой из которых центры всех объектов были распределены равномерно и независимо в единичном интервале. Линейные размеры выбирались так же случайно и независимо в интервалах $[0; 0{,}01)$ и $[0; 0{,}1)$ соответственно для первой и второй частей (рис. 4-7).
8. *Регулярные данные*. Данная выборка получалась в результате разбиения единичного квадрата на 10000 одинаковых квадратов с помощью равноотстоящих горизонтальных и вертикальных линий (рис. 4-8).

В табл. 1-2 приведены результаты моделирования работы различных алгоритмов построения R-деревьев на различных типах экспериментальных данных. Эксперименты проводились на машине с процессором Pentium 166, время построения дерева представлено в миллисекундах. Во всех таблицах данные верны с относительной точностью 1% при доверительной вероятности 0,95.

Таблица 1. Время построения R-дерева различными алгоритмами.

Алгоритм	Тип экспериментальных данных							
	№1	*№2*	*№3*	*№4*	*№5*	*№6*	*№7*	*№8*
R*-дерево	70	71	71	71	72	72	71	70
R*-дерево с уточнением	246	257	234	272	226	253	249	252
Базовый	141	141	142	142	148	149	143	141
Базовый с уточнением	342	332	354	367	342	360	355	346
Клеточный	90	92	91	99	98	97	93	90
Клеточный с уточнением	252	256	251	258	269	260	261	251
«Разделяй и властвуй»	306	305	309	310	313	314	305	303
«Разделяй и властвуй» с уточнением	482	517	521	520	522	525	519	451

Таблица 2. Процент попадания в нужные узлы при поиске в R-деревьях.

Алгоритм	Тип экспериментальных данных							
	№1	*№2*	*№3*	*№4*	*№5*	*№6*	*№7*	*№8*
R*-дерево	22	25	28	41	44	35	24	26
R*-дерево с уточнением	25	27	31	44	46	37	26	30
Базовый	37	38	38	36	18	14	20	41
Базовый с уточнением	37	39	38	37	19	14	20	43
Клеточный	40	41	41	40	20	18	22	44
Клеточный с уточнением	40	42	41	40	21	18	22	45
«Разделяй и властвуй»	45	46	47	55	58	49	36	48
«Разделяй и властвуй» с уточнением	45	47	47	55	58	49	36	49

Таблица 3. Процент перекрытия потомков в узлах R-деревьев.

Алгоритм	Тип экспериментальных данных							
	№1	*№2*	*№3*	*№4*	*№5*	*№6*	*№7*	*№8*
R*-дерево	73	75	76	142	64	57	83	74
R*-дерево с уточнением	65	69	69	141	58	54	79	62
Базовый	7,6	22	77	364	261	213	144	5,9
Базовый с уточнением	7,6	21	77	355	254	210	140	4,8
Клеточный	6,2	18	72	290	225	187	129	5,5
Клеточный с уточнением	6,2	17	73	281	218	186	128	4,6
«Разделяй и властвуй»	0	5,2	22	175	10	7,2	54	2,3
«Разделяй и властвуй» с уточнением	0	5,0	22	174	10	7,1	53	1,3

Сравнение данных в табл. 2-3 подтверждает правильность выбранной стратегии на минимизацию взаимных перекрытий потомков узлов в R-дереве, так как практически везде меньшему проценту перекрытия потомков соответствует больший процент попадания в нужные узлы при поиске.

Как видно из приведённых таблиц, предложенные в работе базовый и клеточный алгоритмы наиболее хорошо работают на равномерных распределениях, когда объекты мало пересекаются между собой (наборы данных 1,2,8). Но на неравномерных распределениях заметно ухудшение работы алгоритмов по сравнению с существующими вариантами R-деревьев. В то же время алгоритм

«Разделяй и властвуй» одинаково хорош как на равномерных, так и на неравномерных распределениях. Практически во всех тестах он обходит остальные алгоритмы по качеству получающегося построения. Только при построении R-дерева на четвёртом наборе данных (большие объекты) этот алгоритм строит дерево, уступающее по проценту перекрытия классическому алгоритму построения R-дерева.

Из представленных результатов экспериментального моделирования видно, что предложенный алгоритм уточнения разбиения позволяет несколько повысить качество получаемой структуры R-дерева. При использовании данного подхода в применении к обычному R*-дереву достигается увеличение попадания на 5-10% и сокращение перекрытия потомков на 7-15%. Применение же локального уточнения в составе глобальных алгоритмов практически не даёт никакого эффекта, за исключением уменьшения перекрытия потомков на регулярном наборе данных (набор 8).

ЛИТЕРАТУРА

1. Guttmann A. R-trees: A Dynamic Index Structure For Spatial Searching // Proc. ACM SIGMOD Int. Conf. on Management of Data, 1984, p. 47-57.
2. Beckmann N., Kriegel H., Schneider R., Seeger B. The R* tree: An Efficient and Robust Access Method for Points and Rectangles // ACM, 1990, p. 322-331.
3. Kriegel H., Schiwietz M., Schneider R. Seeger B. Performance comparison of point and spatial access methods // Proc. Symp. On the Design and Implementation of Large Spatial Databases, Santa Barbara, 1989, Lecture Notes in Computer Science.