Эффективность суперузлов многомерного X-дерева в пространствах малой размерности

Андрей О. Трубаков, Евгений О. Трубаков

***Аннотация*—Индексирование многомерных или много атрибутных данных является важной проблемой современности. Такие данные присутствуют практически во всех областях деятельности человека. При этом объемы данных растут очень быстро и без использования специализированных структур и алгоритмов для их индексирования и доступа к ним обойтись практически невозможно. Однако в области исследований алгоритмов работы с многомерными данными до сих пор остается много белых пятен.**

В данной работе приведены результаты исследования одной из структур, используемой для индексирования многомерных данных – *X*-дерево. Данная структура хорошо себя зарекомендовала в области индексирования данных большой размерности. Однако её поведения для пространств малой размерности изучено слабо. Мы провели ряд экспериментов в пространствах до 5 измерений и исследовали эффективность этой структуры по сравнению с традиционно используемым в таких областях *R*-деревом. Результаты исследования и выводы приведены в последней главе данной статьи.

***Ключевые слова*—многомерные данные, индексирование данных, иерархические методы доступа, многомерные деревья поиска.**

I. Введение

Работа с данными большой размерности является крайне востребованной во многих областях, таких как мультимедийные базы данных, системы проектирования, молекулярная биология и т.д. Для работы с ними было разработано большое количество структур данных, таких как *R*-дерево, *R**-дерево, *KD*-дерево [1]-[3] и др. Эти структуры хорошо подходят для двух- и трехмерных случаев. Однако в случае с большим количеством измерений ситуация сильно усложняется.

Для всех иерархических многомерных структур индексирования пространственных данных, построенных по принципу *R*-дереву, характерна проблема пересечение регионов поиска. Во многих случаях невозможно найти такое разбиение узла на части, при котором пересечение этих частей было бы нулевым [4]. Наличие же таких пересечения приводят к снижению скорости поиска из-за того, что при выполнении прохода по дереву появляется необходимость посетить несколько его ветвей. При этом чем больше будет взаимопересечений узлов дерева, тем более выражен будет негативный эффект при поиске.

Данная проблема более существенна для пространств большой размерности. Для борьбы с этой проблемой были предложены разные подходы. Одним из них является использование суперузлов. В рассматриваемом в статье *X*-дереве в случае невозможности найти подходящее разделение пространства с небольшим пересечением, происходит увеличение размера узла – создается суперузел [5]. Так как при больших пересечениях регионов скорость поиска становится ниже, чем в случае линейного поиска, суперузлы позволяют существенно повысить производительность в пространствах большой размерности. Однако в настоящий момент в литературе слабо освещен вопрос, связанный с применением *X*-дерева в пространствах малой размерности. Целью данной работы было проведение исследований и сравнительного анализа алгоритмов *X*-дерева и классической реализации *R*-дерева в пространствах малой размерности.

II. Иерархическая структура R-дерева

R-дерево – одна из первых иерархических структур данных, предназначенная для работы с многомерными данными, предложенная Антонином Гуттманом [6]. Основная особенность этой структуры заключается в том, что пространство с многомерными объектами рекурсивно разбивается на части до тех пор, пока в каждой из частей не останется небольшое количество объектов.

Пусть дано двухмерное пространство, показанное на рис. 1, в котором находятся точечные объекты

$$\{O_1, O_2, O_3, \dots\}.$$

Если не использовать никаких структур данных, то для любой работы с этими объектами потребуется линейный проход по множеству, т.е. получаем даже для простого доступа к данным сложность $O(n)$. Это совсем не приемлемо в случае большого количества объектов O_i.

Для решения этой проблемы в *R*-дереве предлагается разбить пространство на части, сгруппировав объекты в некоторые группы

$$\{G_1, G_2, G_3, \dots\},$$
$$\forall\, O_i\ \exists\, G_j\ O_i \in G_i.$$

Каждая группа G_i заключается в минимальный ограничивающий прямоугольник (*Minimal Bounding Rectangle – MBR*), который вмещает в себя все объекты этой группы:

Статья получена 1 сентября 2022.

А.О. Трубаков, к.т.н., доц. ФГБОУ ВО «Брянский государственный технический университет», Брянск, Россия (email: trubakovao@mail.ru).

Е.О. Трубаков, к.т.н., доц. ФГБОУ ВО «Брянский государственный технический университет», Брянск, Россия (email: trubakoveo@gmail.com).

$$\forall\, O_i \in G_i : MBR(G_i).x^{left} \le O_i.x \le MBR(G_i).x^{right} \,\&\, MBR(G_i).y^{left} \le O_i.y \le MBR(G_i).y^{right},$$

где $MBR(G_i).x^{left}$, $MBR(G_i).x^{right}$, $MBR(G_i).y^{left}$, $MBR(G_i).y^{right}$ – левые и правые границы минимального ограничивающего прямоугольника группы G_i.

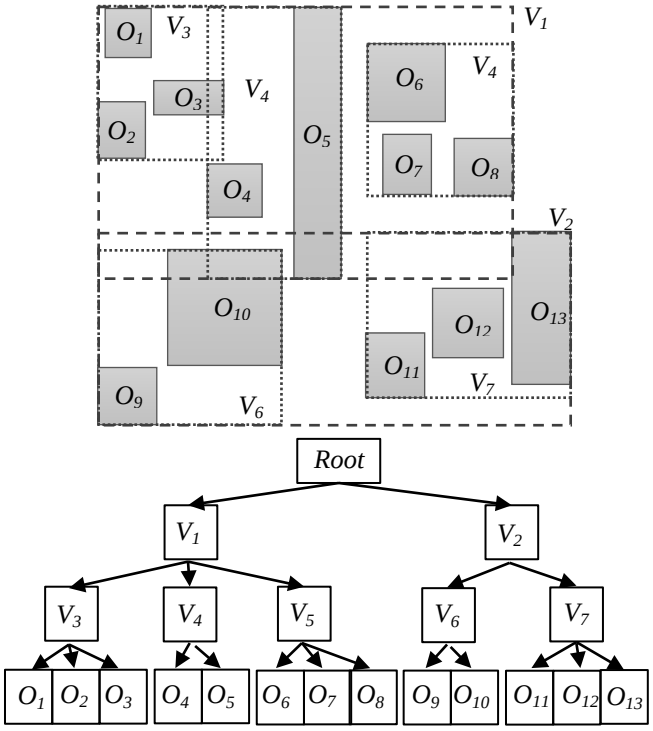

Рис. 1. Иерархическая структура данных *R*-дерево.

Алгоритмы поиска, вставки и удаления объектов работают таким образом, что близко расположенные объекты попадают в одну листовую вершину. При этом процедура обхода заходит только в те вершины дерева, *MBR* которых содержит или пересекается с требуемым объектом или регионом поиска. Благодаря этому при поиске затрагивается только небольшое количество вершин и появляется возможность обрабатывать большие объемы данных с минимальными затратами [8].

Однако стоит заметить, что эффективность процедур поиска будет напрямую зависеть от того, на сколько большие пересечения *MBR* внутренних вершин дерева. Чем больше область пересечений, тем больше вероятность попадания этого региона в *MBR* нескольких узлов. Алгоритм поиска будет рекурсивно вызван для каждого из этих узлов, что приведет к росту числа обращений к внешней памяти [7].

III. Негативный эффект от пересекающихся регионов в R-дереве

Одним из сложных вопросов в реализации *R*-дерева является проблема выбора алгоритма формирования групп объектов при делении пространства на части. Существует большое количество модификаций оригинального *R*-дерева, в которых используются разные алгоритмы построения *MBR*, сильно отличающиеся как по результату, так и по требуемым вычислительным мощностям [9]. Однако идеального алгоритма, который строил бы набор *MBR* абсолютно без взаимопересечения, не существует [10]. Более того, в настоящее время доказано, что в общем случае такого алгоритма не может быть, потому что существуют ситуации, в которых теоретически нельзя разбить пространства на непересекающиеся *MBR*.

Например, рассмотрим ситуацию, показанную на рис. 2(а). У нас дано двухмерное *R*-дерево, минимум и максимум потомков которого равны соответственно 3 и 5. При добавлении 6-го элемента оказывается невозможным найти разделение получившегося набора объектов, не приводящее к появлению пересекающихся областей (см. рис. 2(б), 2(в)).

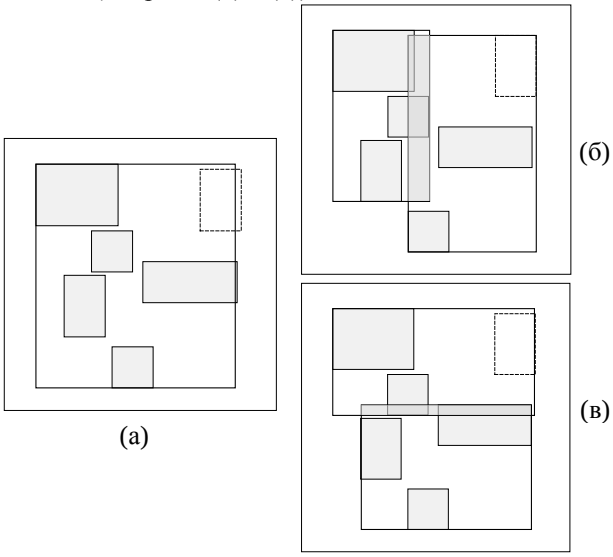

Рис. 2. Проблема пересекающихся регионов в *R*-дереве: (а) – исходные данные; (б), (в) – различные варианты разбиения пространства.

Стоит заметить, что в двух- и трехмерных пространствах общее пересечение всех узлов дерева является не значительным, и им можно пренебречь. Нами были проведены эксперименты, в которых было выявлено, что при равномерно распределенных точечных данных и использовании квадратичного алгоритма поиска разбиений [1] суммарный объем взаимопересечений в двухмерном пространстве не превосходит 1% от общего объема пространства. Это достаточно хорошие результаты, которые позволяют говорить о том, что в пространствах малой размерности взаимопересечения не сильно сказываются на эффективности структуры в целом.

Однако с увеличением количества измерений ситуация кардинально меняется. Количество пересекающихся областей начинает увеличиваться экспоненциально. Это связано с тем, что в пространствах большой размерности становится все сложнее разделить объекты на группы так, чтобы пересечений было как можно меньше. При этом, как было показано выше, чем больше пересечений *MBR* соседних узлов, тем менее эффективным становится процедура поиска.

IV. Суперузлы и уменьшение пересечений MBR узлов

Данную проблему можно решить, сделав некоторое допущение. Допустим, как и ранее, у нас есть некоторое множество объектов $\{O_1, O_2, O_3, \ldots\}$ и нам необходимо разделить эти объекты на k групп $\{G_1, G_2, G_3, \ldots, G_k\}$. Если после выполнения алгоритма разбиения пространства на части минимальные ограничивающие прямоугольники групп будут иметь достаточно большое пересечение (больше некоторого заранее заданного порога P):

$$\Sigma (\cap MBR(G_i), MBR(G_j)) > P,$$

то такой набор объектов не стоит разбивать на отдельные группы, а можно сформировать так называемый суперузел большей вместимости. Такой подход получил название *X*-дерева [11]. В отличии от *TV*-дерева, в котором применяется другая тактика, нацеленная на уменьшение размерности пространства за счет сокращения ряда измерений [12], в данном подходе не теряется часть информации вследствие игнорирования некоторых показателей-измерений. В ряде случаев это может оказаться очень важным фактором при построении информационных систем.

Получившаяся в итоге структура *X*-дерева схожа со структурой *R*-дерева, но помимо обычных узлов в нем используются суперузлы – узлы с увеличенной вместимостью (см. рис. 3). При больших пересечениях *MBR* скорость поиска в *R*-дереве становится хуже линейного поиска. *X*-дерево комбинирует линейный и иерархический способ хранения информации, используя первый только там, где невозможно разделить узлы без образования больших пересечений. При этом количество узлов, которые нужно обойти при поиске, остается таким же, как и в случае разделения узла в *R*-дереве, но на загрузку суперузла из внешней памяти требуется меньше обращений к внешней памяти.

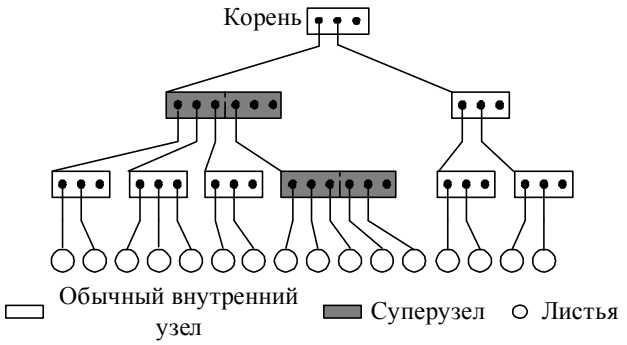

Рис. 3. Структура *X*-дерева и суперузлы.

Стоит отметить, что существуют крайние ситуации. В худшем случае *X*-дерево может выродиться в один линейный суперузел, в лучшем – суперузлы будут отсутствовать, и структура будет эквивалента *R*-дереву. Однако на практике появлении обоих этих случаев является маловероятным.

V. Экспериментальное исследование поведения суперузлов в пространствах малой размерности

Рассмотренный выше подход имеют как сильные, так и слабые стороны. Например, введение суперузлов приводит к тому, что в иерархической структуре (со скоростью $O(log\ n)$) появляются элементы с линейным поиском (со скоростью $O(n)$), что не так однозначно влияет на результат. При этом если в пространствах очень больших размерностей эффект от введения суперузлов очевиден, то в пространствах малых размерностей этот вопрос требует дальнейшего исследования. Именно поэтому нами была проведена серия экспериментов в пространствах размерности до 5 измерений.

Для проведения экспериментов было реализовано консольное приложение на C++. Для хранения данных использовалась внешняя память (при хранении данных в оперативной памяти ответ является очевидным в пользу *R*-дерева). Размер одного блока составлял 32 объекта. Минимальное количество потомков было установлено в 14 штук. Максимальный порог пересечений мы выбрали P = 40%. Для чтения-записи использовался буфер объемом 4 Кб.

Для обеспечения объективности проведенных экспериментов каждый опыт повторялся 30 раз с разным набором данных, результаты усреднялись. При этом в качестве измеряемого параметра выступало как время работы (однако оно сильно зависит от условий запуска приложения, мощности компьютера, используемой операционной системы, мгновенной загруженности процессора т.д.), так и более объективный показатель – количество обращений к внешней памяти (который неподвержен вышеописанным факторам).

VI. Результаты экспериментов

В первом эксперименте деревья заполнялись случайными двухмерными данными, в роли которых выступали прямоугольники с размером от 0 до 100. Координаты прямоугольников варьировались от 0 до 10 000. Аналогичный эксперимент был проведен для данных с размерностью 3, 4 и 5. На рисунках ниже представлены графики зависимости числа обращений к диску и затраченного времени от размерности данных для деревьев с 500 000 записей в разных разрезах.

Первым исследованным алгоритмом был алгоритм добавления новых объектов в дерево. На рис. 4 показан график зависимости числа обращений к диску от размерности данных при вставке нового элемента. Видно, что для *X*-дерева этот показатель выше. Это связано с суперузлами и необходимостью их обработки. Но разница эта не столь значительна. Это связано с тем, что в пространствах малой размерности редки случаи образования суперузлов. На рис. 5 показана зависимость времени вставки от размерности для тех же данных. При этом разница во времени обработки запросов оказалось даже менее выраженной и практически отсутствует.

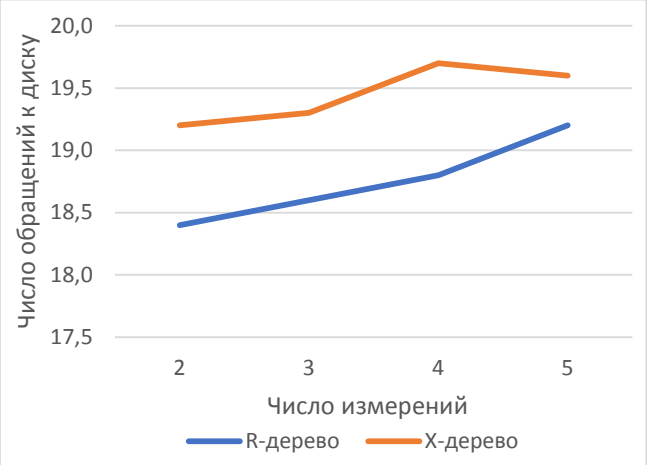

Рис. 4. Зависимость числа обращений к диску от размерности при вставке объектов в дерево.

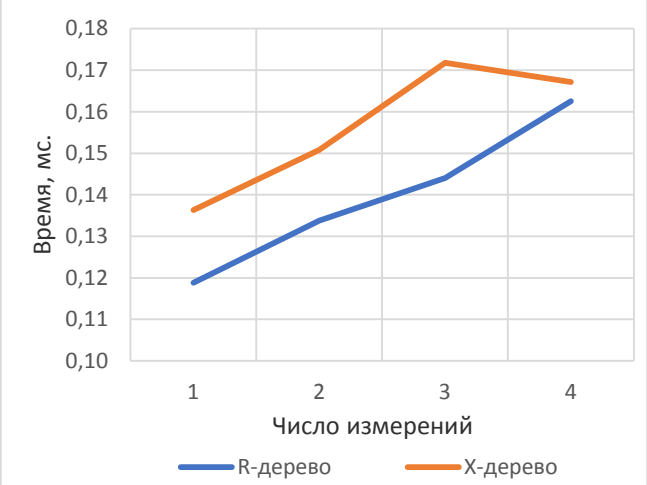

Рис. 5. Зависимость времени вставки объекта от размерности данных.

Однако самым важным алгоритмом для структур индексирования данных является поиск объекта по точному совпадению. В реальных системах это самый часто используемый запрос. Именно от его скорости в большинстве случаев зависит скорость работы информационной системы. Поэтому во второй серии экспериментов основное внимание было уделено именно этому запросу. На рис. 6 и рис. показаны аналогичные показанным выше графики, но для данных, полученных при исследовании операции поиска по точному совпадению.

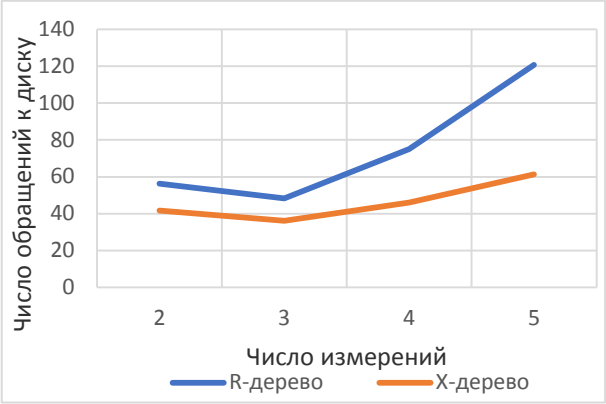

Рис. 6. Зависимость числа обращений к диску от размерности данных при поиске объектов по точному совпадению.

Из приведенных выше данных можно заметить, что при поиске возникает обратная ситуация: благодаря суперузлам снижается число обращений к диску и уменьшается время поиска. Здесь как раз начинает проявляться положительная сторона от внедрения суперузлов. Однако ля размерностей 2-5 разница между *R*-деревом и *X*-деревом по числу обращений к диску невелика. Более глубокий анализ результатов показал, что это связано с тем, что количество суперузлов в пространствах малой размерности невелико.

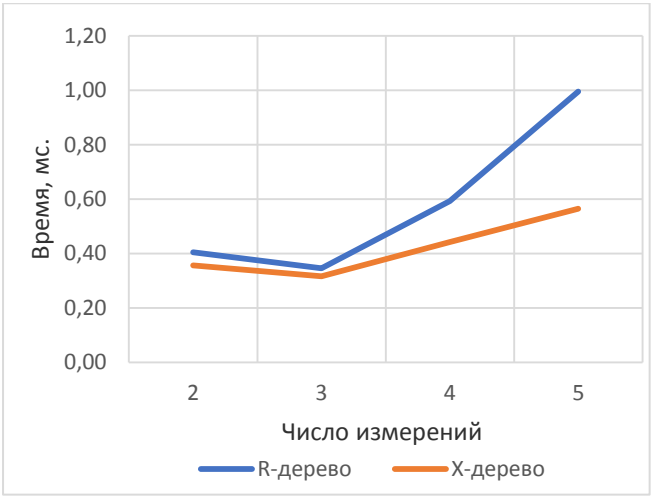

Рис. 7. Зависимость времени поиска от размерности данных при поиске объектов по точному совпадению.

Еще одним немаловажным показателем является то, как ведет себя структура с увеличением количества записей. Поэтому в дополнение к описанным выше нами также были проведены эксперименты, направленные на анализ поведения структуры при разном количестве добавленных объектов. В качестве исследуемого было выбрано 4D пространство. Такая размерность была выбрана, потому что в 2D пространстве разница не столь заметна, как было показано выше. В пространствах большей размерности будет явно намечаться преимущество *X*-дерева.

На рис. 8-11 показаны графики зависимости числа обращений к диску и времени от количества записей в дереве. Исследования, как и ранее, проводились отдельно для операций модификации данных структуры (операции вставки), так и для операций поиска данных по точному совпадению.

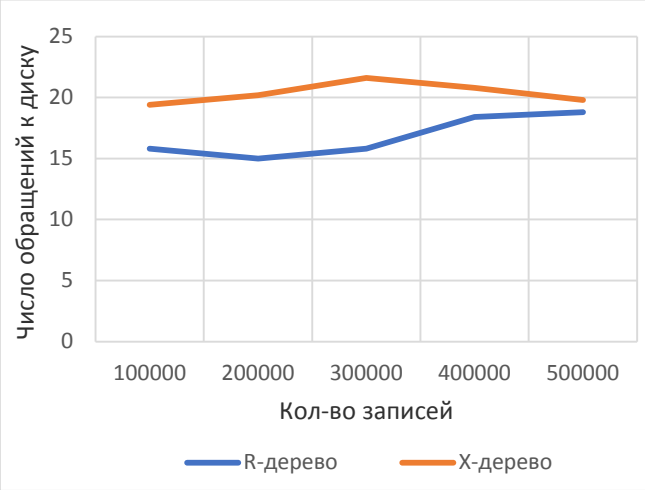

Рис. 8. Зависимость числа обращений к диску от количества записей при вставке объектов

Из приведенного графика видно, что с ростом количества записей скорость запросов и обращения к внешней памяти в *X*-дереве снижается медленнее. Это связано с тем, что записи добавлялись в ограниченной области и это приводит к множественным пересечениям, что дает возможность суперузлам X-дерева проявить себя в полную силу.

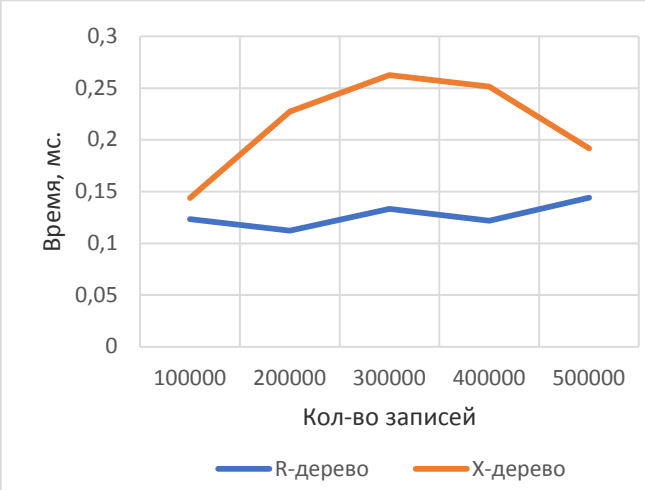

Рис. 9. Зависимость времени вставки от количества записей.

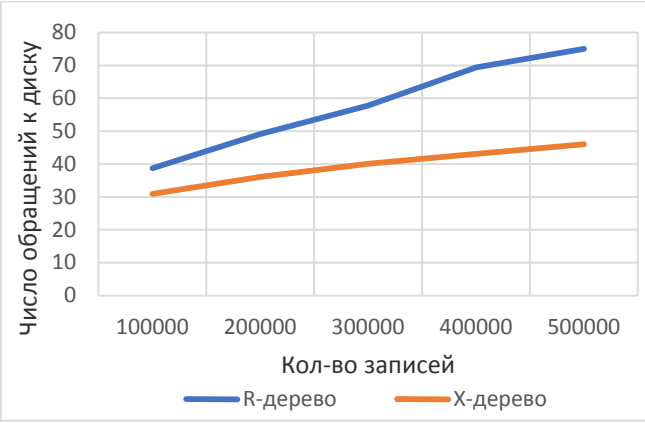

Рис. 10. Зависимость числа обращений к диску от количества записей при поиске объектов по точному совпадению.

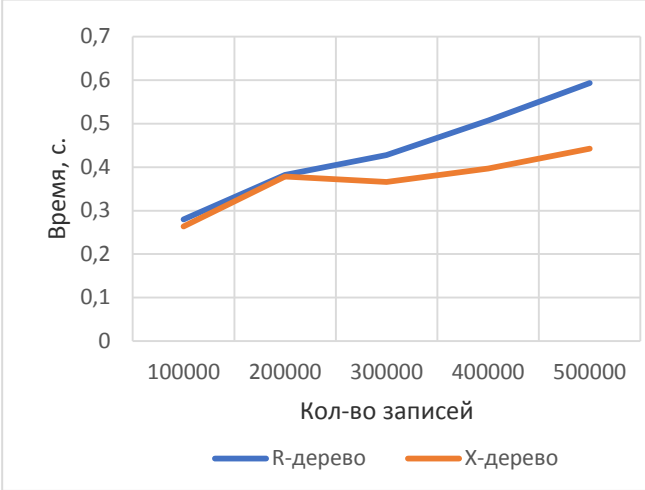

Рис. 11. Зависимость времени поиска от количества записей при поиске объектов по точному совпадению.

VII. Заключение

Проведенные исследования и эксперименты позволяют сделать следующие выводы: *X*-дерево сохраняет высокую производительность даже при малых размерностях данных. Это позволяет использовать его в любых многомерных применениях. Однако стоит заметить, что использование суперузлов дает колоссальные преимущества только в пространствах большой размерности. В пространствах же малой размерности эффект от их использования не столь велик.

Стоит так же заметить, что скорость работы структуры при вставке объектов в *X*-дерево ниже, чем у *R*-дерева. Алгоритм вставки всегда проходит ровно по одной ветке дерева. При этом в *X*-дереве на пути алгоритма встречаются суперузлы, вследствие чего увеличивается количество данных, считанных из файла.

При поиске, напротив, скорость *X*-дерева оказалась немного выше. Однако разница в производительности в обоих случаях невелика, так как суперузлы в пространствах малой размерности образуются не часто.

Библиография

[1] В.К. Гулаков, А.О. Трубаков, Е.О. Трубаков. Структуры и алгоритмы обработки многомерных данных : монография. – 2-е изд. – Санкт-Петербург : Лань, 2021. – 356 с. – ISBN 978-5-8114-7965-8.

[2] H. Samet. Foundations of multidimensional and metric data structures. – San Francisco : Morgan Kaufmann Publishers Inc., 2006. – 1024 p. – ISBN 0-12-369446-9.

[3] K. Markov, K. Ivanova, I. Mitov, S. arastanev. Advance of the access methods. // International Journal «Information Technologies and Knowledge», Vol.2, 2008.

[4] S. Brakatsoulas, D. Pfoser, Y. Theodoridis. Revisiting R-tree construction principles. // Advances in Databases and Information Systems, 2002. – P. 149-162.

[5] А.О. Трубаков, Е.С. Молодьков. Алгоритмы построения суперузлов при индексировании многомерных данных в пространствах большой размерности с использованием X-дерева // Программные продукты, системы и алгоритмы. – Тверь, 2018. – N 4. – С.37-47.

[6] A. Guttman. R-Trees: A dynamic index structure for spatial searching // In Proceedings of the ACM SIGMOD International Conference on Management of Data, 1984. – P. 47-57.

[7] R.N. Brisaboa, M.R. Luaces, G. Navarro, D. Seco. Space-efficient representations of rectangle datasets supporting orthogonal range

querying // Information Systems, Volume 38, Issue 5, 2013. – P. 635-655.
[8] Y. Manolopoulos, A. Nanopoulos, A.N. Papadopoulos, Y. Theodoridis.R-Trees: Theory and Applications. // Springer Publishing Company, Incorporated, 2005. – 194 p. – ISBN:978-1-85233-977-7.
[9] N. Athanasiou, M. Vassilakopoulos, A. Corral, Y. Manolopoulos. Use-based Optimization of Spatial Access Methods // In Proceedings of the 9th International Conference on Management of Digital EcoSystems, New York, 2017. – P. 65–72.
[10] C. Torres, P. Perez-Lantero, G. Gutierrez. Linear separability in spatial databases // Knowl Inf Syst 54, 2018. – P. 287-314.
[11] S. Berchtold, D.A. Keim, H. Kriegel. The X-tree: An Index Structure for High-Dimensional Data. – 1996. – 12 p.
[12] K. Lin, H.V. Jagadish, C. Faloutsos. The TV-Tree: An Index Structure for High-Dimensional Data. – 1994.

Efficiency of multidimensional X-tree supernodes in low-dimensional spaces

Andrey O. Trubakov, Evgeniy O. Trubakov

***Annotation*— Indexing multidimensional or multi-attribute data is an important problem today. Data volumes are growing very fast and it is necessary to use specialized structures and algorithms to index them. However, in the field of research of algorithms for working with multidimensional data there are still many blank spots.**

In this paper, we present the results of research on one of the structures used to index multidimensional data - X-tree. This structure has proved itself well in the field of indexing high-dimensional data. However, its behavior for low-dimensional spaces has been poorly studied. We have conducted a number of experiments in spaces of up to 5 dimensions and investigated the effectiveness of this structure compared to the R-tree traditionally used in such areas. The results of the study and conclusions are given in the last chapter of this paper.

***Key words*— multidimensional data, data indexing, hierarchical access methods, multidimensional search trees.**

REFERENCES

[1] V.K. Gulakov, A.O. Trubakov, E.O. Trubakov. Structures and algorithms of multidimensional data processing : a monograph. - 2nd ed. - Saint-Petersburg : Lan, 2021. – 356 c. –- ISBN 978-5-8114-7965-8.

[2] H. Samet. Foundations of multidimensional and metric data structures. – San Francisco : Morgan Kaufmann Publishers Inc., 2006. – 1024 p. – ISBN 0-12-369446-9.

[3] K. Markov, K. Ivanova, I. Mitov, S. arastanev. Advance of the access methods. // International Journal «Information Technologies and Knowledge», Vol.2, 2008.

[4] S. Brakatsoulas, D. Pfoser, Y. Theodoridis. Revisiting R-tree construction principles. // Advances in Databases and Information Systems, 2002. – P. 149-162.

[5] A.O. Trubakov, E.S. Molodkov. Algorithms of building supernodes when indexing multidimensional data in high-dimensional spaces using X-tree // Software Products, Systems and Algorithms. – Tver, 2018. – N 4. – C.37-47.

[6] A. Guttman. R-Trees: A dynamic index structure for spatial searching // In Proceedings of the ACM SIGMOD International Conference on Management of Data, 1984. – P. 47-57.

[7] R.N. Brisaboa, M.R. Luaces, G. Navarro, D. Seco. Space-efficient representations of rectangle datasets supporting orthogonal range querying // Information Systems, Volume 38, Issue 5, 2013. – P. 635-655.

[8] Y. Manolopoulos, A. Nanopoulos, A.N. Papadopoulos, Y. Theodoridis.R-Trees: Theory and Applications. // Springer Publishing Company, Incorporated, 2005. – 194 p. – ISBN:978-1-85233-977-7.

[9] N. Athanasiou, M. Vassilakopoulos, A. Corral, Y. Manolopoulos. Use-based Optimization of Spatial Access Methods // In Proceedings of the 9th International Conference on Management of Digital EcoSystems, New York, 2017. – P. 65–72.

[10] C. Torres, P. Perez-Lantero, G. Gutierrez. Linear separability in spatial databases // Knowl Inf Syst 54, 2018. – P. 287-314.

[11] S. Berchtold, D.A. Keim, H. Kriegel. The X-tree: An Index Structure for High-Dimensional Data. – 1996. – 12 p.

[12] K. Lin, H.V. Jagadish, C. Faloutsos. The TV-Tree: An Index Structure for High-Dimensional Data. – 1994.