UCT JOURNAL OF RESEARCH IN SCIENCE, ENGINEERING AND TECHNOLOGY 2018(03)

Available online at http://journals.researchub.org

An investigation of classification and comparison of R tree construction algorithms with A Set of R* and R+ trees

Goranova Olga, Onufrieva Anna, Dmitrieva Olga, Titov Evgeniy, Atroshenko Larisa*

Department of urban economy and housing law, Moscow Metropolitan Governance University, Moscow, Russian Federation,

ARTICLE INFO

Article history:
Received 24 Apr. 2018
Accepted 01 June 2018
Published 26 Aug. 2018

Keywords:
Algorithms, R Trees, Packets, Classification

ABSTRACT

Given the growth of the internet network and the importance of the quality of different services that must be provided to users by several networks, packet classification is receiving increasing attention as one of the main and sensitive requirements of the network. Such that for the most services provided by the network, the routers must perform the classification with high speed and low memory usage. Packet classification enables the routers to provide modern network services. The classification is performed geometrically in R-tree. The main theory of the spatial index which considered as the most important theory of inquiry is the calculation of proximity theory. The most famous index structure is called the R-tree. The main strategy for R-tree is to collect multidimensional spatial nodes with a minimum boundary rectangle (MBR) that is the smallest inner spatial node of the rectangle. After indexing, the optimization of retrieving information from a spatial database has great importance. Hence for more discussion, a new spatial index belonged to the R^+ tree family called R^{++} tree is investigated in this paper which maintains the overlap of the nodes. The results show that the suggested R^+ tree outperforms the R^+ tree in domains of queries, KNN queries, and Top-k queries.

1. INTRODUCTION

Histograms are important structures that first used in database systems in order to selective estimate the queries. They are used to obtain a fast-approximate response to important dense queries. Most histograms are based on the network and are only applicable to two-point data. An optimal designing for the multidimensional histograms is famed as an NP-hard problem [1-3]. The current situation indicates that during a query execution in a spatial database management system (SDBMS), the query optimizer creates all of the schemes available in query assessment. These schemes are all similar in the eventual result but differ in execution cost and time [4, 5].

The philosophy of the histogram buckets assignment is allocating them to subspaces that correctly detect clusters of objects. Therefore, first, a method for finding the center of each object cluster is proposed. Then an algorithm to create histogram buckets from these centers is suggested. These buckets are initialized from the cluster centers and then expanded to cover the clusters. The best expansion approach is selected based on the concept of increasing skewness [6].

One of the first methods proposed for multidimensional data is called the h tree. This method creates partitions from multidimensional space overlapping based on the frequency as the origin parameter [2]. The authors suggest two strategies:

1) The basic variable acts like follow: In the first phase, the algorithm computes a defined network and the number of subclass spatial objects for each cell. Then base on the network computed, the binary space partitioning (BSP) is used to compute the histograms.

2) To reduce the effect of the second structure strategy, i.e., MinSkew progressive refinement, several networks with different partitions are used. Each network partition is used to create equal segments of histogram linking curves [2].

MinSkew is a known method for creating histograms of spatial data. This method first estimates the main data using a uniform network. Then start with a single bucket consisted of all data objects. The spatial deviation and split point along its dimensions which produces the maximum reduction in spatial deviation are calculated for each bucket. Then MinSkew selects the buckets whose splits lead to the biggest reduction in spatial skewness. Next, divides these buckets into two child buckets, and assigns data from the old bucket to the new one. After MinSkew completed, the created histogram is a set of non-overlapping buckets [7].

The bichromatic bucket creation algorithm can easily be used to receive a new copy for every produced bucket. The set of bichromatic buckets is then reported as the final histogram. This strategy, i.e., converting each bucket to a bichromatic after completing the histogram creation process, can be utilized in any method in which the created buckets are not overlapping [8].

ST-Hist is a method of creating histograms for two/three-dimensional geographical data. Given a dataset with the data space, ST-Hist first partitions the whole data space into a number of data segments. Then for each segment, ST-Hist detects dense points that turn into histogram buckets, recursively. Here, the dense points refer to a data region that meets special conditions in terms of shape, size and body frequency. Every bucket is recognized in a data region which is an organized bucket tree. Considering the bichromatic bucket creation method, it is possible to use this method to convert each produced bucket by ST-Hist during the histogram creation process into a bichromatic version. In another word, for each data segment, once the root bucket is created, the bichromatic

**Corresponding author:* goaolga@rambler.ru
DOI: https://doi.org/10.24200/jrset.vol6iss03pp25-27

bucket creation algorithm is applied to this root bucket. Then, ST-Hist detects the points within the improved root bucket and turns them into child buckets. The bichromatic bucket creation algorithm is applied to each one of these child buckets in order to obtain an enhanced copy. This process continues until no new bucket is left to create [2].

ST-Hist-c is an improved version of the ST-Hist method that uses the same framework as ST-Hist. The ST-Hist considers rigid conditions on shape and size to detect region. This is mainly because of an easily related way to find dense regions. However, due to these rigid conditions, ST-Hist sometimes fails to detect the object clusters in datasets. To create an accurate histogram, ST-Hist-c assigns buckets to data clusters locations. First of all, a clustering algorithm is combined with two statistical methods in order to find the number and the location of the object clusters. Then a new algorithm is proposed to create buckets from cluster centers. The bucket region gradually expands from the cluster center to all directions. During a bucket expansion, the best approach among many possible expansion methods is selected based on a new concept of increasing skewness [9].

2. EXPERIMENTS

In this paper, the implementation of different histograms is investigated. The MinSkew is used as a reference method that works well. The second method is called MinSkewProg. For d = 2 a network with 24 cells is used. For $d = 3$, 2^{15} cells are considered in MinSkew. Four networks with $2^6, 2^9, 2^{12}, 2^{15}$ cells are considered in MinSkewProg. Other methods used in experiments are listed in table 1.

Table 1. A review of the cost functions of the studied methods.

Definitions	Cost Function
MBR	C_V
C_V expansion with average, query side length	C_{QP}
K-Uniformity criterion	C_{RK}
Spatial Skew of MBR	C_{SK}
Definition	**Histogram**
MinSkew, stable network	MinSkew
MinSkewProg, filter	MinSkewProg
rkHist with $\alpha = 0.1$	rkHist
Size-invariant Partitioning, Hilbert curve	RTree
Cv, Hilbert Curve	R-V
C_{QP}, Hilbert Curve	R-VQP
C_{RK}, Hilbert Curve	R-RK
C_{SK}, Hilbert Curve	R-SK
ST-Hist Forrest	FST

3. RESULTS

3.1 Construction and estimation time

To reduce the approximate time, the histogram can be provided as the main memory of R-Tree. Figure (1) shows the performance of the size of linking curves and total workload time to provide a histogram. The first two lines belong to R-Tree and the third line relates to an array for linking curves. Main memory adjustments are considered for R-Tree and output capacity is set to 12 entries for each group (the best set in experiments). In addition, an R-Tree is created using the histogram linking curves through an OTP partitioning method with C_V as a cost function.

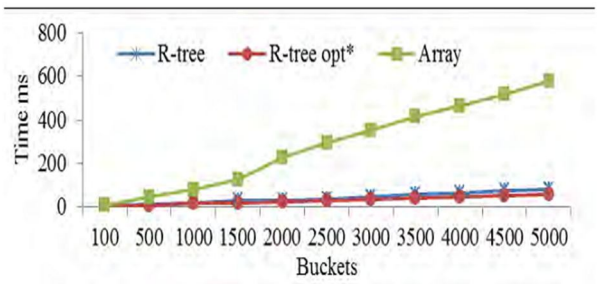

Fig.1 the histogram of R-Tree

Figure (2) show the results of R-Tree methods in comparison with size-invariant partitioning strategies for d=2 data point. It is observed that the methods which use an optimal partitioning framework based on type have more accuracy than the R-Tree method.

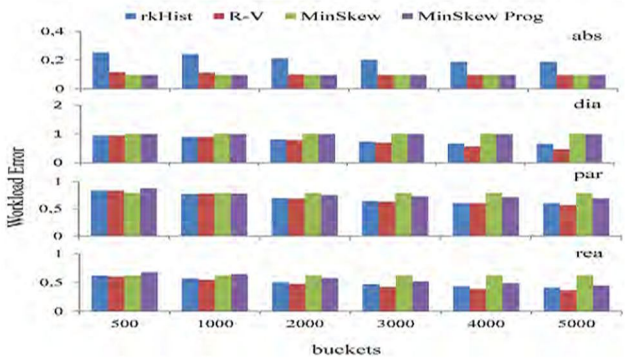

Fig. 2 results of R-Tree methods

A possible solution is to data distribution partitioning with respect to target shape and size and creating a histogram or indexing independently for each category [10].

3.2 Difference between R, R* and R+ Trees

The tree is a method for looking up data with a location that is often represented as (x, y). Searching a tree with one leaf is a solved problem. Searching a tree with two or more leaves, and request for locations near both x and y coordinates require craftier algorithms. R* tree is generally a tree- type data structure that is used to index local information [11].

3.3 Difference between R+ tree and R Tree

R^+ tree is a compromise between R tree and kd-tree. This type of tree avoids the overlap of inner nodes by inserting an object to several leaves if necessary. Two or more nodes have complete overlapping. Minimum overlapping reduces the set of search paths toward a leaf (reduce even more in case of accessing minimum critical overlapping.). An efficient search requires minimum coverage and overlapping [5].

R tree and R^+ tree are different from the following aspects:

At least half of the nodes are not guaranteed. The entries of each inner node are not overlapped. An identifier of an object can be stored in more than one leaf node [12].

Advantages: The nodes have no overlapping with each other. The query of a point within spatial regions is covered at most by a node. A unified path is followed, and fewer nodes are visited compared with R Tree [2].

Disadvantages: Since the rectangles are repeated, the size of an R^+ tree might become so bigger than an R tree over the same dataset. Construction and maintenance of an R^+ tree are more complex than the construction and maintenance of other types of R trees [13].

R^* tree is a type of R tree that is used to index local information. The cost of creating an R^* tree is slightly higher than the standard R trees. Although data might require replacement, this tree normally has better query performance. This tree just like standard R tree can be stored in both two points and spatial data.

3.4. Difference between R^* tree and R tree

Minimizing the overlap and coverage, both rely on R tree performance. The overlapping means that during data query or insertion, more than one branch of the tree is required for expansion. Minimum overlapping leads to improve pruning operation. In most cases, it allows excluding entire pages from the search, especially for negative domain queries.

R* tree tries to reduce both (overlapping and coverage). This tree uses the combination of a revised splitting algorithm and the concept of forced reinsertion in order to avoid the overflow of a node. It depends on the R tree structure observations and is very sensitive to the order of input insertion. Therefore, the structure of insert construction is probably suboptimal. Insertion and removal of the inputs allow finding a location in the trees that is more suitable than the original place [2]. In table 2, a comparison between R, R^* and R^+ trees is presented.

Table. 2. A comparison between R, R^* and R^+ trees.

R	Comparison
R+	There are fewer nodes in R^+ compared to R.
R*	The structure of insert construction is probably suboptimal.

4. CONCLUSION

For future works, researchers can use the R^+ tree for packet classification. The R^+ tree is a method to lookup data with a location like coordinates (x, y), for places on the ground. The R^+ tree is a compromise between R tree and kd-tree and avoids the overlap of inner nodes by inserting an object to several leaves if necessary. This tree reduces the minimum overlap of a set of search paths toward a leaf. An optimum search requires a minimum of coverage and overlapping. One advantage of this algorithm is non-overlapped nodes, which leads to higher performance of queries. Among the disadvantages is that since rectangles are repeated, the tree becomes bigger than a regular R tree over the same dataset. Moreover, the construction and maintenance of the R^+ tree are more complex than a standard R tree. Experimental results show that ST-Hist is better than other proposed methods. In this paper, R^{++}, R^+, and R^* trees are compared as common indicators of the R tree family. The R^{++} tree is proposed as an improved version of the R^+ tree. The results show that the suggested R^{++} tree outperforms the R^+ tree in domains of queries, KNN queries, and Top-k queries. However, the performance of the R^{++} tree decreases by the dimension growing, because repetition values are also increased. Finally, it is shown that the efficiency of the search time of the R^{++} tree is so much better than the R^+ tree, especially when the duplicate data point is considered.

REFERENCES

[1] Z. Houshmandian, "Presentation of a Packet Classification Algorithm Using R Tree", MSc thesis, Islamic Azad University, Kermanshah Branch, Kermanshah, Iran, 2015.

[2] N.M. BalouchZehi, M. Fathi, S. Yousefi, "Internet Packet Classification using Prefix-Based Bitmap Intersection", Iranian Journal of Science Research, Vol. 3, Num. 1 (a), 2005.

[3] B. Roshan. R, "A Packet Classification using Bit Matrices", 8th Annual Iranian Computer Society Conference, 2002.

[4] M. Fathi, N.M. BalouchZehi, "Presentation of a Method to packet Classification based on Bit Matrices", 9th Annual Iranian Computer Society Conference, Sharif University of Technology, Tehran, 2003.

[5] M. V, Mohammadi, "Present a Hybrid Method of IP Traffic Classification with Help of Feature Mapping and Genetic Algorithms", 15th International Conference of Iranian Computer Society, Energy Technology development Center, Tehran, 2009.

[6] A. M. Abedi, V. Dehghan, M. Sabaei, "A New Algorithm for Detecting and Repairing Interferences Available between Dominated principles of Firewalls ", Conference of Information Technology and Economic Jihad, Kazerun Higher Education Complex, Kazerun, Iran, 2011.

[7] S. VahabZadeh, "Packet Classification using six-bit trees", 12th Annual Iranian Computer Society Conference, 2006.

[8] H. Abdoli, H. Saeedi, "packet Classification Algorithm in Non-uniform Separators", 14th Iranian Conference on Electrical Engineering, 2006.

[9] F. Baker, editor. “Requirements for IP version 4 routers,” RFC 1812, http://www.ietf.org/rfc/rfc2676.txt, June 1995.

[10] Pankaj Gupta, “Algorithms for Routing Lookups and Packet Classification”, PhD dissertation, Computer Science Department, Stanford university, 2000

[11] C. Hedrick. “Routing information protocol,” RFC 1058,http://www.ietf.org/rfc/rfc1058.txt, June 1988.

[12] P. Gupta, and N. McKeown, "Algorithms for Packet Classification," IEEE Trans. Network, vol. 15, no. 2, pp. 24- 32, 2001.

[13] D. E. Taylor, "Survey and Taxonomy of Packet Classification Techniques," Journal of ACM Computing Surveys, vol. 37, no. 3, pp. 238-275, 2005.