Int. J. of Comp. & Info. Tech., (2017) 5(1): 53-58.

IJOCIT

ISBN: 2345-3877
www.ijocit.org
Volume 5, Issue 1

Original Research

A Novel Method to Improve Query in Big Databases Using a Geometric Tree Base Algorithm

Ameneh Eskandari [1]
Zahra Nilforoushan [1,*]
Javad Ranjbar [2]

Received: 30 Nov 2016
Accepted: 08 Jan 2017

Abstract
Nowadays many daily jobs could be done via virtual network as the world of IT and electronics has dramatically expanded. The increase in the number of users of big databases would demand a serious need in storing the information about these users and in particular search in such databases. For example in an identification authentication system, we need to compare a single fingerprint with many known fingerprints. Now if this database is big, the search will be time consuming and the fast methods would be essential. One of the methods for reducing search time is tree structures. In this paper, a new geometric method for search in big databases is proposed where we first construct the search tree and by removing certain branches, we obtain the desired output result. This approach has been compared with KD-Tree and the implementation results have shown that the proposed method is 100 times faster in the search time.

Keywords: Database, query, KD-Tree, split tree, KNN.

Citation: Eskandari, A., Nilforoushan, Z., Ranjbar, J., (2017). A Novel Method to Improve Query in Big Databases Using a Geometric Tree Base Algorithm, *Int. J. of Comp. & Info. Tech. (IJOCIT)*, 5(1): 53-58.

[1] Department of Electrical and Computer Engineering, Kharazmi University
[2] Department of Electrical and Computer Engineering, Yazd University
[*] Corresponding Author: nilforoushan@khu.ac.ir

Introduction

At the first glance, it seems that geometry has no application in database structures but one can interpret a database geometrically. More precisely, records and information in database are considered as points. For example consider a database of personnel of a company. If we want to know the personnel whose ages are between 1980 to 1985 and whose salaries are in the range 1000 to 2000 dollars, this search is called range query [1]. Another example is a police station containing thousands of fingerprints. For identification of a certain person, we need to search in the database for the fingerprint of the person by searching through all fingerprints. This search is called exact query [1]. If the number of points in the database is too large, the search will be very time consuming. One approach to reduce the search time is using the search trees.

Figure 1: interpreting a data base query geometrically [5]

2. KD-Tree

In this section the problem of 2D-rectangular searching is considered which is mostly used in many problems [2]. Let P be the set of points in the plane. The basic assumption is that no two points have the same x-coordinate, and no two points have the same y-coordinate. A 2-D rectangular range query on P asks for the points from P lying inside the query rectangle $[x, x'] \times [y, y']$. A point $p := (p_x, p_y)$ lies inside this rectangle if and only if;

$$p_x \in [x, x'] \text{ and } p_y \in [y, y']$$

The set of (1-D) points is split into two subsets of roughly equal size, one subset contains the point smaller than or equal to splitting value, and the other contains the points larger than splitting value. The splitting value is stored at the root and the two subsets are stored recursively in two subtrees. Each point has its x-coordinate and y-coordinate. Therefore, we first split on x-coordinate and then on y-coordinate, then againon x-coordinate, and so on. At the root we split the set P with vertical line L into two subsets of roughly equal size. This is done by finding the median x- coordinate of the points and drawing the vertical line through it. The splitting line is stored at the root. P_{left}, the subset of points to left is stored in the left subtree, and P_{left} , the subset of points to right is stored in the right subtree. At the left child of the root, we split the P_{left} into two subsets with a horizontal line. This is done by finding the median y-coordinate if the points are in P_{left} . The points below or on it are stored in the left subtree, and the points above are stored in right subtree. The left child itself stores the splitting line. Similarly P_{left} is split with a horizontal line, which is stored in the left and right subtree of the right child. At the grandchildren of the root, we split again with a vertical line. In general, vertical lines separate the nodes with even depth while the horizontal ones separate the nodes with odd depth [2].

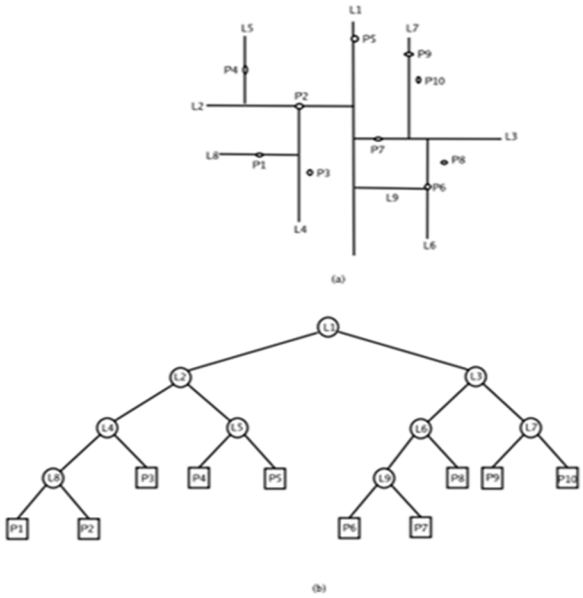

Figure 2: KD-Tree (a) the way the plane is divided (b) corresponding binary tree [2]

2.1. Algorithm of KD-Tree

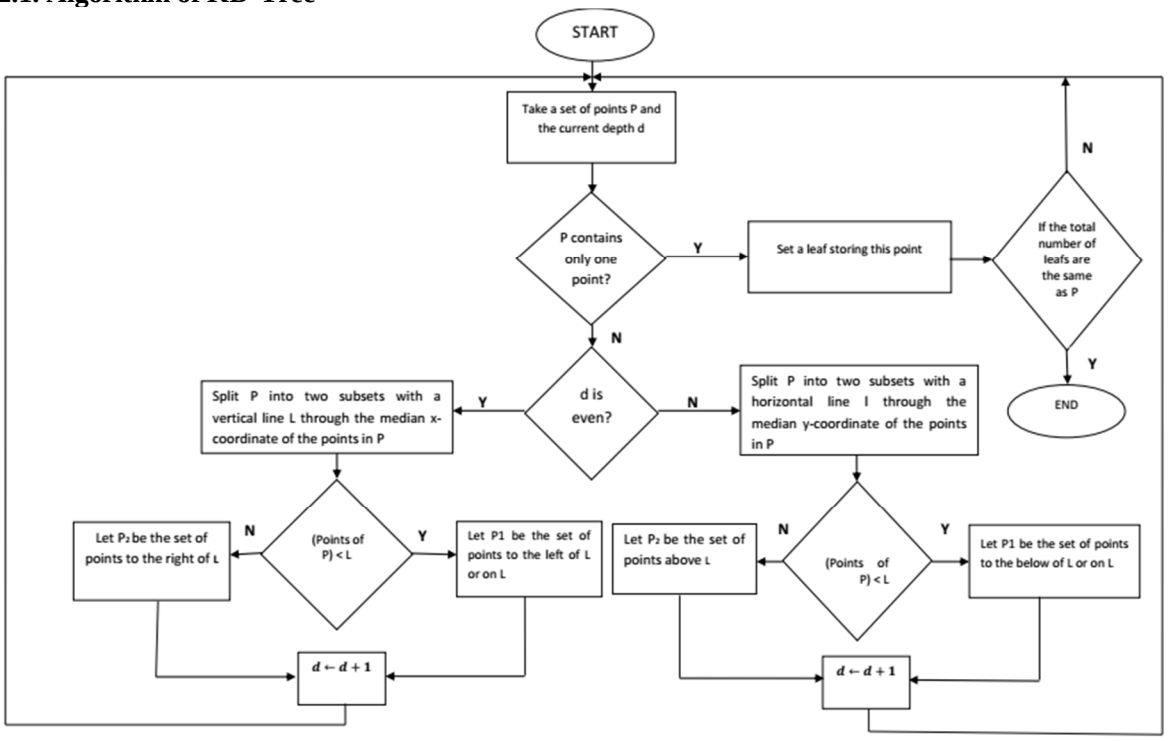

Figure 3: Flow chart of KD-Tree algorithm

In this approach which is very similar to the binary KD-Tree, at first the bounding box is determined. Then width and height of the obtained rectangular is computed and divided into half from the larger side. If the larger side is in the direction of x-axis (y-axis), we will divide the rectangle into half by a vertical (horizontal) line and the left and right (up and down), and the points will make left and right subtrees. If the larger side is in the direction of y-axis, we will divide the rectangle into half by a horizontal line and the up and down points will make left and right subtrees. Then, this process will continue recursively [3].

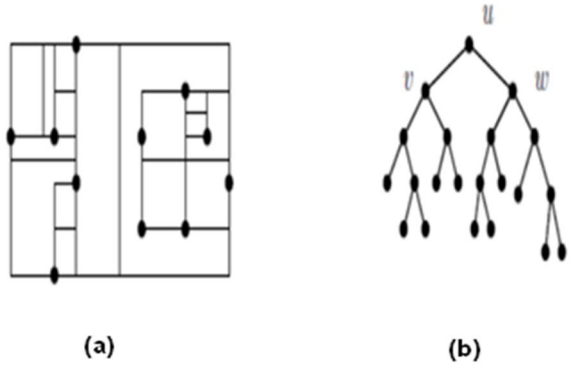

Figure 4: Split -Tree (a) the way the plane is divided (b) corresponding binary tree

2.2. Split Tree Algorithm

Figure 4 (a) & (b) show this algorithm:

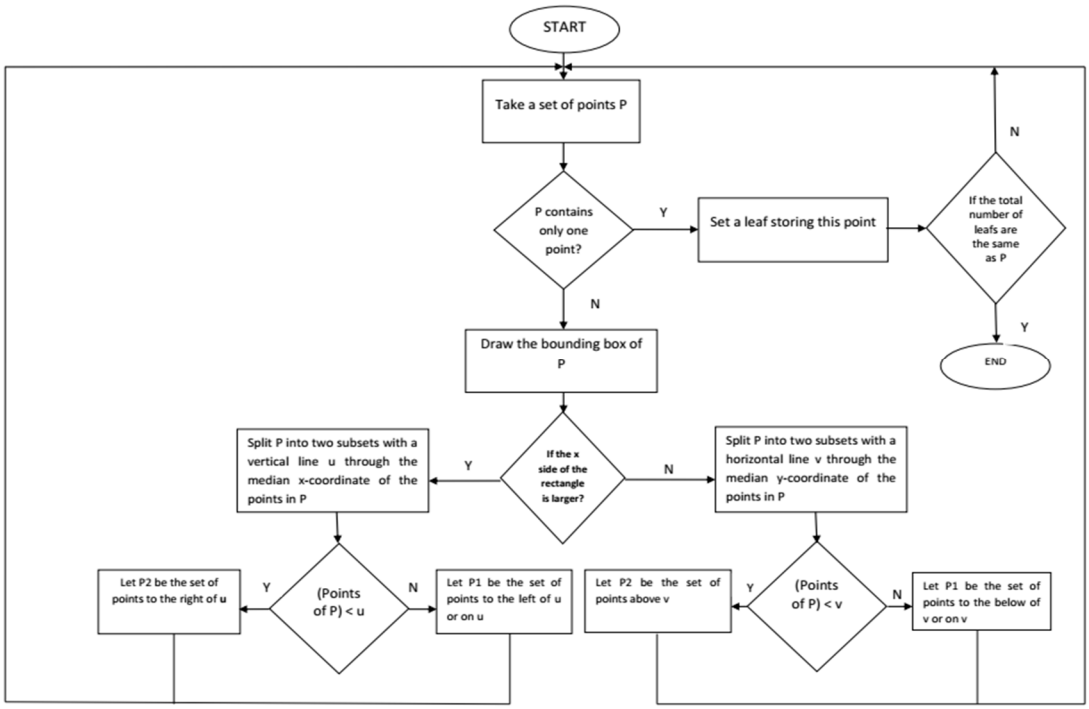

Figure 5: Flow chart of Split-Tree algorithm

3. The proposed method

In this paper both range query and exact query methods are considered where we apply the KNN-method by k value taken from user for the first and k=1 for the second method. The search method will not construct the whole tree in the memory and only the branch which has the output point would be constructed. As a result the search process will be more efficient. The pseudo code of our proposed algorithm is in the following:

Input: set of P point and the point D.
Output: The specified arm of the tree that includes K.

- Until the number of set P is k do(k for KNN)
- find bounding box of P
- If the larger side of the rectangular is in x-dimension
- Then find the median of the x-dimension side (m_X)
- If x-dimension of D is less than m_X
- Find the point in the left side of m_X (P_LEFT)
- else the point in the right side of m_X (P_RIGHT)
- Else find the median of the y-dimension side (m_Y)
- If y-dimension of D is less than m_X
- Find point in the left side of m_Y (P_LEFT)
- else find the point in the right side of m_Y (P_RIGHT)
- Return to 1.

3.1 Implementation results

Our implementation has been done in MATLAB using a laptop with 4GB of RAM and DUAL CORE 3.4 GHz CPU over windows 7 operating system. In figure 6 , at first the range search with KNN with k=5 in a database of 1000 points is done and at the second phase one of these points would be chosen as a test point and the exact search with KNN with k=1 is applied (figure 7).

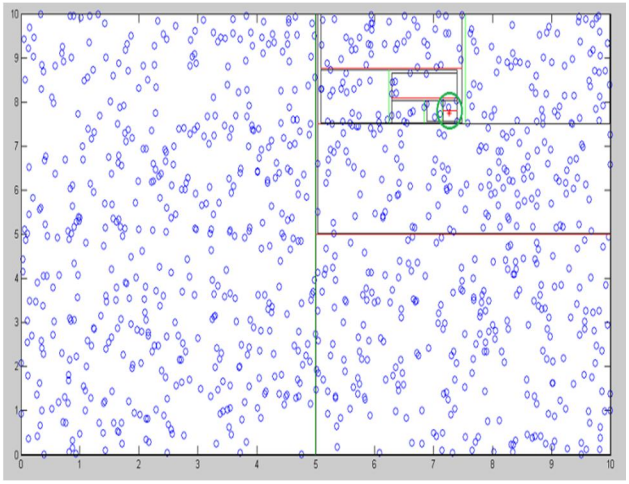

Figure 6: Range query in proposed method with k=5

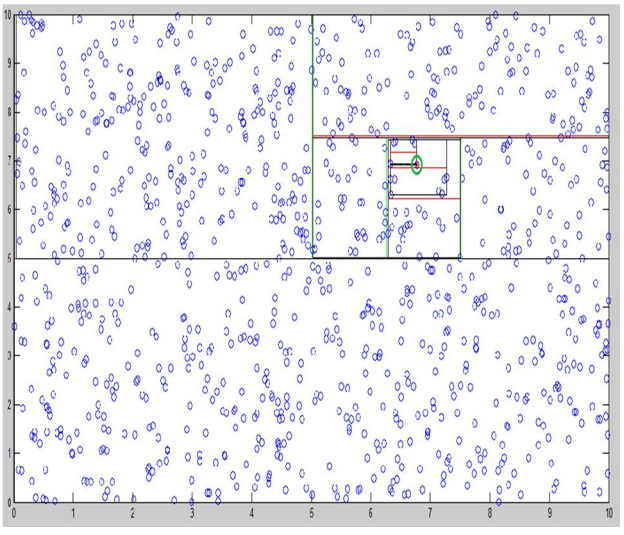

Figure 7: Exact query in proposed method with k=1

In order to show the performance of this algorithm we compare it with KD-Tree.

We repeat the search process for similar database but with different points and the elapsed time of each method is shown in table 1.

Table 1: Numerical comparison of elapsed time in proposed method and KD-Tree in database query

Number of points	Elapsed time in proposed method (s)	Elapsed time in KD-Tree (s)	Number of points	Elapsed time in proposed method (s)	Elapsed time in KD-Tree (s)
20	0.000582	0.004667	520	0.001212	0.09222
40	0.000551	0.008884	540	0.001035	0.096011
60	0.000475	0.015554	560	0.001091	0.1052
80	0.000465	0.016003	580	0.001095	0.106841
100	0.000534	0.019479	600	0.001139	0.109288
120	0.000701	0.022158	620	0.001096	0.117218
140	0.000536	0.026929	640	0.001002	0.118015
160	0.000488	0.030524	660	0.001014	0.122392
180	0.000584	0.035514	680	0.000967	0.127769
200	0.000692	0.042037	700	0.000958	0.13209
220	0.000553	0.041129	720	0.000963	0.13394
240	0.000578	0.044308	740	0.001096	0.137489
260	0.000666	0.046428	760	0.001243	0.141615
280	0.000643	0.052551	780	0.001149	0.144513
300	0.000735	0.056387	800	0.001019	0.148844
320	0.000632	0.060729	820	0.001155	0.151518
340	0.000843	0.064485	840	0.001101	0.164905
360	0.000749	0.07092	860	0.001203	0.158842
380	0.000815	0.071647	880	0.001152	0.158924
400	0.000882	0.076666	900	0.001223	0.162881
420	0.000909	0.078128	920	0.0012	0.165303
440	0.001004	0.080135	940	0.001371	0.167491
460	0.00083	0.083748	960	0.001488	0.169391
480	0.000926	0.088798	980	0.001089	0.172748
500	0.000933	0.088698	1000	0.001296	0.17319

In figure 8 it is seen that the speed of search has been dramatically increased. In order to better understanding the numerical difference between two algorithms, both run times of the two algorithms are presented. As a conclusion we can say that the speed of proposed method is at least 104 times faster than that of in the KD-Tree.

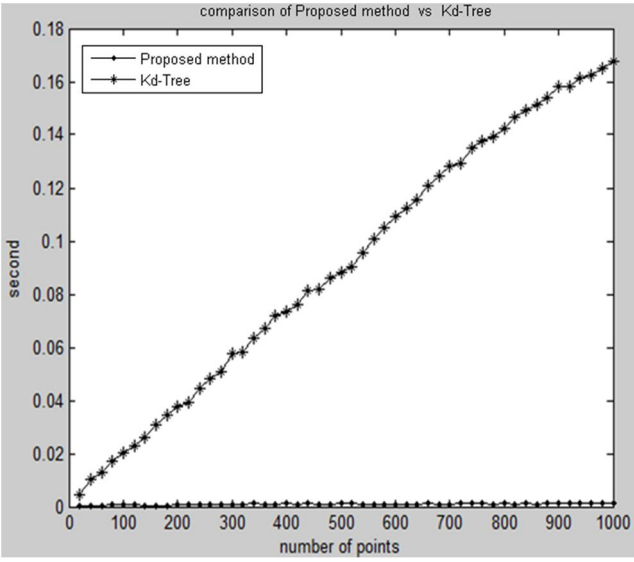

Figure 8: Comparison of elapsed time in proposed method vs. KD-Tree

4. Conclusion

In this paper a new geometric algorithm for search in big databases is proposed. We applied the algorithm for range and exact query in a database. In order to show the efficiency of our algorithm, we compared it with the KD-Tree. The simulation results show the efficiency of our algorithm and reduction of searching time which is about 104 times faster of it.

References

[1] Shivangi Surati; Devesh C Jinwala; Sanjay Garg, (2014). "A Peer-to-Peer multi way tree network with efficient range query search using multidimensional indexing ", Fourth International Conference on Digital Information and Communication Technology and its Applications (DICTAP), pp: 1–6.

[2] Vandana Dixit K., Deepti Singh, Parul Raj, M. Swathi and P. Gupta, (2008). "kd-tree Based Fingerprint identification System", 2008 2nd International Conference on Anti-counterfeiting, Security and identification, pp: 5–10.

[3] Narasimhan, Giri; Smid, Michiel, (2007). Geometric Spanner Networks, Cambridge University Press, ISBN 0-521-81513-4., pp: 155 – 163.

[4] Hyeong-Il Kim1 and Jae_woo Chang, (2013). "K-Nearest Neighbor Query Processing Algorithms for a Query Region inside the Road Networks," *Journal of Computer Science and Technology*, Vol. 28, no. 4, pp. 585–596.

[5] Hemant M. Kakde, (2005). " Range Searching using KD-Tree", from the citeseerx database on the World Wide Web: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.122.5818.

Authors Profile

Ameneh Eskandari is a MSc student of Computer Science, in Kharazmi University, Tehran, Iran from 2014. Her research interests are Computational Geometry and Computer Graphics. Email: eskandari@gmail.com

Zahra Nilforoushan received her MSc in Pure Mathematics-Algebraic Geometry and PhD in Computer Science - Computational Geometry from Amir kabir University of Technology, Tehran, Iran in 2003 and 2009 respectively. She served as a lecturer at Dept. of Computer Science, Faculty of Mathematical Sciences and Computer, Kharazmi University, Tehran, Iran from 2009 to 2011. Since 2012 she is with Dept. of Electrical and Computer Engineering, Faculty of Engineering at Kharazmi University as an Assistant Professor. Her research interests are Computational Geometry, Computer Graphics, Computer Vision, Differential Geometry, and Robotics. Email: nilforoushan@khu.ac.ir

Javad Ranjbar; received his MSc in communication engineering from Amir kabir University of Technology, Tehran, Iran in 2010 and is a PhD student in communication-wireless system in Yazd university, Yazd, Iran from 2013. His research interests are wireless systems, WSN, and pattern recognition. Email: jranjbar@yahoo.com