Communications in Mobile Computing
a SpringerOpen Journal

ORIGINAL SUBMISSION Open Access

K-NN query algorithm based on PB-tree with the parallel lines division

Jine Tang, ZhangBing Zhou* and Qun Wang

*Correspondence:
zbzhou@cugb.edu.cn
School of Information Engineering, China University of Geosciences (Beijing), Beijing, China

Abstract

Spatial index and query are enabling techniques for achieving the vision of the Internet of Things. K-NN is an algorithm which is used widely in spatial database. Traditional query algorithms use R-tree as the index structure and improve the query efficiency by using the measurement distance and pruning strategy. Based on the study of previous algorithms, this paper proposes a novel K-NN query algorithm based on PB-tree with the parallel lines division. PB-tree index is different from the traditional R-tree index, where PB-tree adopts parallel lines to divide the spatial region and uses parallel lines as the parent node. It is similar to the binary tree index structure and requires to query three small portions nearest to the queried object in each K-NN query. Therefore, the search range is narrowed and the query efficiency is enhanced. Experiments show that PB-tree is better than the traditional R-tree from the aspect of query performance. PB-tree can avoid the deficiency of a large number of overlap and coverage among odes in R-tree and multiple index paths when searching data objects, and hence PB-tree can find K-NN objects meeting the conditions quickly and efficiently in large data sets.

Introduction

Things (or objects) in the Internet of Things [1,2] can be modeled as MBRs [3] in some context, and spatial index and query are enabling techniques for searching objects that a user interests. K-neighbor query method is widely used for querying spatial objects in spatial database. It is different from the point and range query, and is used to find the K-nearest objects near a given point in the space, namely, the k-NN query [4,5].

Current research of spatial index methods attempts to achieve the quick mapping from spatial regions to spatial entities [6]. As an important technology aiming to improve query efficiency, spatial index is a key component of spatial database system structure. R-tree [3,7] proposed by Guttman is a balanced tree which is composed of multi-nested external rectangles. It is another form of B-tree developed towards multi-dimensional space, and divides the space objects according to ranges, and each node corresponds to a region and a disk page. Disk pages of non-leaf nodes store the region scope of all its sub-nodes, and all child nodes of non-leaf nodes fall within the scope of the non-leaf nodes region. Disk pages of leaf nodes store MBRs of all spatial objects within leaf nodes regional scope. R-tree is a dynamic index structure. Queries can be carried out simultaneously with the insertion or deletion algorithm, and this method does not require to reorganize the tree structure regularly.

Many K-NN query algorithms have been proposed based on R-tree index structure. [8] presented a branch and bound method to traverse R-tree, which sorts and prunes the MBR within the nodes in R-tree to carry on K-nearest neighbors query. [9] proposed a spatial k-NN query strategy by using MINDIST and MINMAXDIST to sort and pruning rule to prune tree, and generated Active Branch List (ABL) in leaf-level, namely, rect1, rect2, $\cdots$, rectk. ABL has the nodes needed to be visited currently and the child nodes needed to continue to search. Each level ABL is sorted in R-tree according to the ascending order of MINDIST and MINMAXDIST to carry on K-NN query. [10] proposed a multi-object K-NN query, and used pruning rules to achieve multi-object K-NN query in R-tree. However, R-tree allows overlap and coverage among the sibling nodes, even for the exact match query, R-tree cannot guarantee to visit only one branch when carrying inquiries. This is the factor affecting the searching efficiency of R-tree.

Based on above K-NN algorithms, a novel K-NN query algorithm based on PB-tree with the parallel lines division is proposed in this paper. This method aims to address the problem of overlap and coverage among the nodes in R-tree. The construction of PB-tree index is similar to that of binary tree, which uses the parallel lines dividing the region as parent nodes. The searching path of leaf node is single. There is no overlap and coverage among division regions. No matter the queried objects are stored in the parent nodes or leaf nodes, the K-NN query region of every object contains three small portions only in PB-tree. When the query result does not satisfy the conditions, it needs continue to expand the query scope toward the two sides of the previous region. Due to effectively selecting the searching range, the query efficiency and accuracy are both improved.

The K-NN query algorithm based on the PB-tree with the parallel lines division

Spatial index structure [11,12] organizes and stores index data according to the spatial distributive features of spatial data. It does not traverse data sets when carrying on queries to data sets. Spatial index can completely gain query results or the smaller data set including all query results through visiting the index data, so the good index structure has the advantages of higher storage efficiency and query efficiency.

PB-tree adopts parallel lines to divide the spatial region. The selected location of parallel lines should be able to make the entire region divided into two almost equal parts in each division until the number of data objects contained in each region is between the pre-specified maximum and minimum values. Data objects intersecting with parallel lines are stored in the corresponding data lists of parent nodes. Data objects completely contained in a region are stored in corresponding leaf nodes. The restrictive parameters M is the maximum number of data objects contained in the leaf nodes of PB-tree, and m is minimum number of entity objects contained in the leaf nodes. The PB-tree has the following properties:

- The root node is the diagonal which divides the entire region into two parts. Root node has two children nodes, the left and right regions of the diagonal (the left and right regions are not divided), or the division parallel lines of the left and right regions (the left and right regions are divided);
- Each non-leaf node is the parallel line dividing the sub-regions. All data objects intersecting with parallel lines are stored in the corresponding data lists. The left and

right child nodes are the adjacent left and right regions or the division parallel lines of the adjacent left and right regions;
- The number of entity objects contained in each leaf node is between m and M.

During the process of carrying on K-NN queries, since the actual space region is very large and without rules, if directly using the parallel lines to divide the area, the distribution of data objects is not uniform. To resolve this issue, before making the inquiries, we carry on clustering on the spatial data objects through using the K-means [13] clustering algorithm, which makes the MBR of each cluster tending to square. The distance among data objects in the same cluster is small, and the distance among clusters is large.

We will search the cluster class of queried objects by adopting the parallel lines to carry on division to the region of above cluster class. In such division, there are two cases depending on whether the queried data object intersecting with the division parallel lines or completely contained in a divided area. The specific query algorithm is as follows:

(1) The queried objects intersecting with the parallel lines

- We first query the pointer which points to the data list where the queried object is, and achieve the corresponding parallel line;
- We search the sub-tree which uses the parallel line as the root node, and gain the most right child node division region from the left sub-tree and the most left child node division region from the right sub-tree. If the left sub-tree or right sub-tree is the division region, we directly return the corresponding left or right division region.
- The neighbor region of the queried object will be the parallel line where the queried object is and the corresponding left and right division regions. We calculate the distance between the data objects in the above three parts and the queried object, sorting and inserting these data objects into the sorting queue.
- If we fail to find all the neighbor objects meeting the conditions, we need continue to expand the query area, merge the searched regions as a new region, and continue to query the left and right parallel lines region of this new area, the corresponding query algorithm being shown in the second case.

(2) The queried object is included in a division region

- We search the region where the query object is, and find the parent node parallel line of the above region. If this area is the left (right) child, then the above parallel line is the nearest right (left) parallel line of the queried object;
- We continue to query the left (right) parallel line of the region where the queried object is, and search the parent node of the found right (left) parallel line. If the right parallel line is the left child, and the region where the query object is does not have left adjacent parallel line, then the neighbor region of the queried data object is the region where the queried object is and the right parallel line. If the right parallel line is the right child, and the parent node parallel line is the left parallel line of the region where the queried object is, then the neighbor region of the queried data object is the region where the queried object is and the left and right parallel lines. If the left parallel line is the left child, and the parent node parallel line is the right parallel line of the

region where the queried object is, then the neighbor region of the queried data object is the region where the queried object is and the left and right parallel lines. If the left parallel line is the right child, we use the middle order traversal algorithm to search for the direct stepfather node of the region where the queried object is, and the parallel line of the direct stepfather node is the right parallel line of the region where the queried object is. We calculate the distance between the data objects in the above searched region and the queried object, sorting and inserting these data objects into the sorting queue.

- If we fail to find the neighbor objects meeting the conditions, it still needs to expand the query area, merge the above queried area as a new area, and continue to find the left and right division regions of the new area, the corresponding query algorithm being shown in the first case.
- When the search is finished in the cluster class where the queried object is, if it still fails to find all the neighbor objects meeting the conditions, we search the next class nearest to the queried object, add the queried object into the new found cluster class to form a new region, and continue to carry on division and query in the new region, until we find all the neighbor objects.

The new K-NN query algorithm based on PB-tree will traverse the tree no matter searching the left and right division regions or the left and right division parallel lines. In the process of traversing, there are two cases according to the region where the object is. Because of partitioning out the data objects intersecting with the parallel lines, there are no overlap and coverage among the division regions. When searching the corresponding regions and parallel lines, the query path is single, and it can directly find the left and right neighbor regions. Compared with the R-tree, the width and depth of PB-tree is reduced, and all the division regions are not at the same level. The traversal query is very quick when carry on querying the left and right parallel lines of the division regions in the low levels, and the query efficiency is greatly improved. If the most left and right division areas of the sub-tree of a certain parallel line are in the low levels, the time spent on querying is also significantly reduced, and the query efficiency of traversing from top to bottom or from bottom to top is significantly enhanced, thus it can greatly improve the K-NN query efficiency.

As the division region shown in Figure 1 and the corresponding PB-tree index shown in Figure 2, if we want to find the six neighbor objects of P_{12}, we search the data list of the storage area where the P_{12} is, that is the parallel line d, according to the search algorithm of the first case. We find the left and right division regions R_4 and R_5 of d, calculate the distance between P_{12} and the data objects in the above three regions, sort and insert these data objects into the sorting queue, and achieve the former three neighbor object P_5, P_8, P_9. Then we merge the three regions, search the left and right parallel lines according to the query algorithm of the second case, and gain the remaining three neighbor objects P_1, P_2, P_6.

Experiment and Evaluation

In order to demonstrate the K-NN query algorithm feasibility of the improved PB-tree data structure, we adopt the new K-NN query algorithm *a* based on the PB-tree with the parallel lines division and the K-NN query algorithm *b* proposed in literature [8] to carry

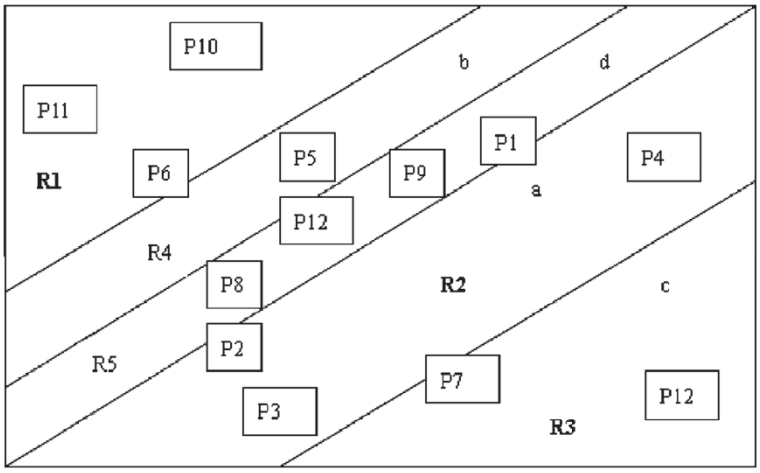

Figure 1 The region division of spatial objects.

on comparison of the query performance. The experimental data come from the real two-dimensional geospatial data of a city. The data set size is 500000, including points, lines and polygons and other spatial entities. We use c++ language to carry on programming, and all programs are run on windows XP operating system. Figure 3 shows the CPU running time comparison of the two algorithms by adjusting the number of the queried objects (fixing K). Figure 4 presents the changes of the used disk access times with k when using the two algorithms (fixing the number of the queried objects). Figure 5 shows the accuracy when query the 250 neighbor objects of the given 100, 200, 300, 400 data objects.

The experimental results show that the new query algorithm outperforms the traditional K-NN query algorithm based on the R-tree in performance. The CPU time and disk

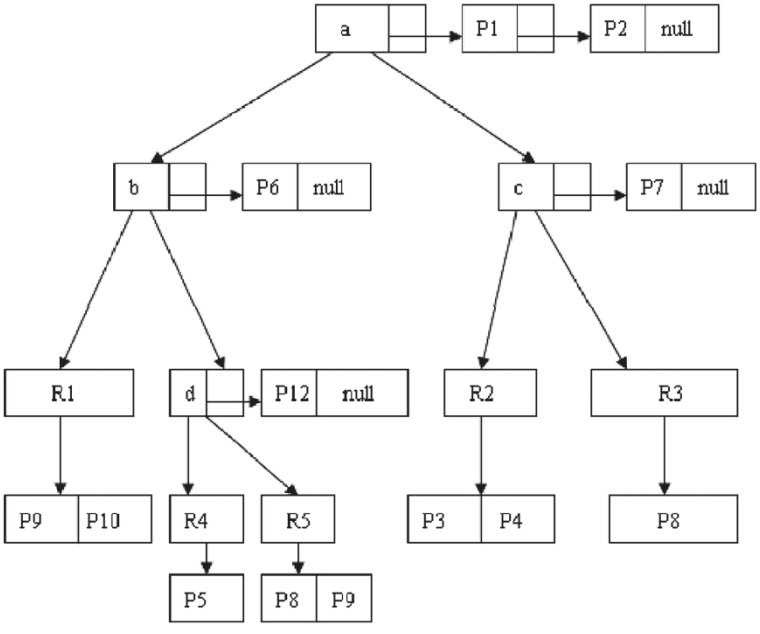

Figure 2 PB-tree index structure.

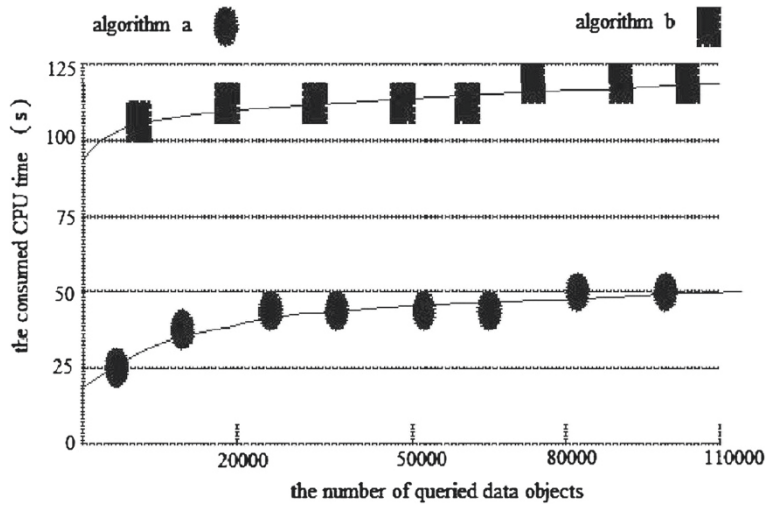

Figure 3 The CPU time comparison of the different number of queried objects.

access times required for querying are greatly reduced. After clustering the regions, when carry on searching in each cluster class, due to the number of data objects is less, it is easy to manage, and the distance between the data objects in the same class is small. After dividing the region by using the parallel lines, the length and width of the small region in each class are proportional to the distance between data objects. It will not appear the situation of making the data objects in the neighbor area not the former neighbor objects of the queried object, at the same time, the data objects not in the neighbor region are the former neighbor objects due to the length and width of the neighbor area overlarge. If we carry on the division to the whole area, this situation will occur frequently and the query efficiency and accuracy are obviously reduced. Because of conducting the region division to the cluster where the queried object is and establishing the PB-tree index in the above cluster, it can quickly find the neighbor query region through simply traversing

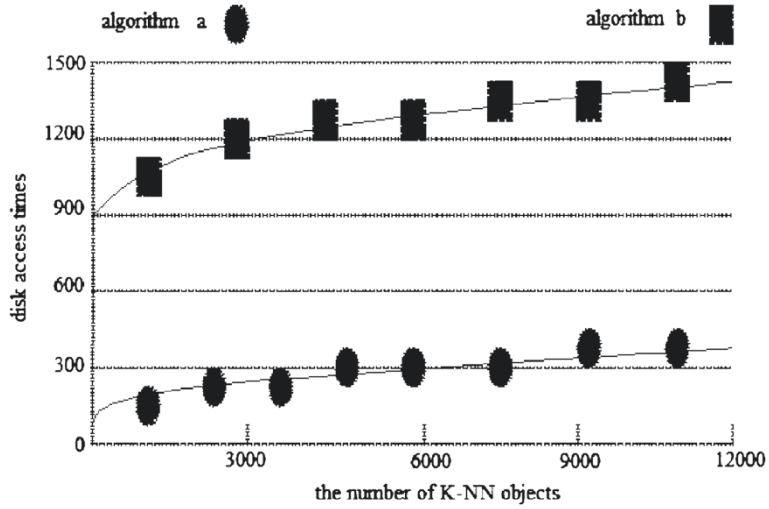

Figure 4 The disk access times comparison of the different number of k-NN objects.

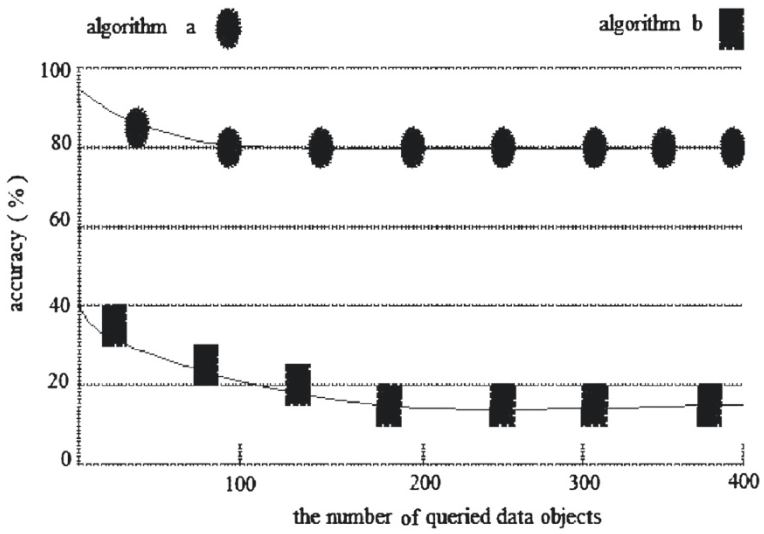

Figure 5 The accuracy comparison when querying the different number of the queried objects.

a certain part of the tree or querying a certain single path. During the process of regional expansion, it is the alternation of the above two cases. The entire search algorithm makes K-NN query efficiency greatly increased.

Conclusions

To support spatial query in the context of the Internet of Things, this paper proposed a novel K-NN query algorithm based on PB-tree index. Through dividing a large region into small regions by clustering algorithm, our method can achieve the effective management of spatial data objects. The experiments show that the proposed query algorithm has a good performance, and can be widely used in the large spatial data set.

Acknowledgements
This work was supported by the Fundamental Research Funds for the Central Universities (China University of Geosciences at Beijing).

Received: 9 August 2012 Accepted: 13 August 2012
Published: 30 November 2012

References

1. Ashton K (2011) That 'Internet of Things' Thing. RFID J
2. Doytsher Y, Galon B, Kanza Y (2012) Querying Socio-spatial Networks on the World-Wide Web. In: Proceedings of the 21st International World Wide Web Conference (WWW), Lyon, France, 329–332
3. Guttman A (1984) R-trees: a Dynamic Index Structure for Spatial Searching. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Boston, Massachusetts, 47–57
4. Li B, Pan M, Wu Z (2011) Effective Reverse K-nearest Neighbor Query Based on Revised R∗-tree in Spatial Databases. In: Proceedings of the 19th International Conference on Geoinformatics, Shanghai, China, 1–5
5. Lu J, Lu Y, Cong G (2011) Reverse Spatial and Textual k Nearest Neighbor Search. In: Proceedings of the 2011 international conference on Management of data (SIGMOD '11), Athens, Greece, 349–360
6. Al-Badarneh A, Al-Alaj A (2011) A Spatial Index Structure Using Dynamic Recursive Space Partitioning. In: Proceedings of the International Conference on Innovations in Information Technology, Abu Dhabi, United Arab Emirates, 255–260
7. He Z, Wu C, Wang C (2008) Clustered Sorting R-tree: An Index for Multi-Dimensional Spatial Objects. In: Proceedings of the 4th International Conference on Natural Computation, Jinan, China, 348–352
8. Lai L, Liu Z, Yan L (2002) K-nearest neighbor search algorithm by using R-tree. In: Computer Engineering and Design, vol 23(9)
9. Liu Y, Zhu Z, Shi S (2001) A new space k-nearest neighbor query strategy. Journal of Shanghai Jiao Tong University 35(9)
10. Liu Y, Bo S, Zhang Q, Hao Z (2004) Multi-object nearest neighbor queries. In: Computer Engineering, vol 30(11), 66–68

11. Xie G, Su J (2005) Spatial data indexing technology and its application in GIS software. In: Hainan Normal University, vol 18(4), 372–376
12. Wu X, Zang C (2009) A New Spatial Index Structure For GIS Data. In: Proceedings of the 3rd International Conference on Multimedia and Ubiquitous Engineering, Qingdao, China, 471–476
13. Pettinger D, Fatta GD (2010) Space Partitioning for Scalable K-Means. In: Proceedings of the 9th International Conference on Machine Learning and Applications, Washington, DC, USA, 319–324

doi:10.1186/2192-1121-1-10
Cite this article as: Tang *et al.*: K-NN query algorithm based on PB-tree with the parallel lines division. *Communications in Mobile Computing* 2012 **1**:10.