RESEARCH ARTICLE OPEN ACCESS

A Survey on Multiple Attribute Graph Indexing Methods

Abilasha S [1], Remya G [2], Anuraj Mohan [3]
PG Scholar [1], Assistant Professor [2] & [3]
Department of Computer Science and Engineering
NSS College of Engineering
Palakkad - India

ABSTRACT
Graphs are efficient data structures used for representing connected entities and are very ubiquitously used in real world applications. Graphs with multiple attributes are able to capture the properties of real world data efficiently. Such graphs are called as multiple attribute graphs. Nodes of such graphs have multiple attributes and for each attribute there will be associated non negative weights. Searching for matches on these graphs is complex than traditional graph search. For example users may be interested in searching for social networks to find users who are most active in online activities (ie, at least any of their liking, posting, commenting activities have greater than 50 percent values) and are connected as friends in social network. Manipulation of such query needs efficient way to handle the large set of nodes and their attribute values. Index creation on nodes of multiple attributed graph helps in efficient ordering of this large node data along with attribute values which facilitates for fast and efficient searching and further query processing. Indexing of these nodes can be done using spatial indexing structures where these multiple attributed nodes are mapped as multidimensional points. In this paper multiple attribute graphs are studied and various methods for indexing multidimensional points (multiple attributed node indexing) are addressed and compared.

Keywords: **-** Multiple Attribute Graph, R tree, Graph Indexing.

I. INTRODUCTION

Multiple attribute graphs are those graphs in which for each node there will be multiple attributes associated with it (eg: social network where users have different properties linked to them describing social activities) and having non negative weights assigned to these attributes [1]. Such graphs have various applications in real world which can be used for representing co-authorship graph, social networks, communication networks etc. The most important purpose of multiple attributed graph is that to provide support for fast and efficient querying by considering both structural (finding friends in social network) and attribute constraints (users who are more active in online activities). Searching these graphs are done for identifying sub graphs matching to the query constraints given which can be used for detection of some interesting patterns such as relation between co-authors, presence of some chemicals in chemical compounds, finding closely related friends etc.

Multiple attributed graph search include search on multiple attributes that are weighted, associated with each node (ie, by giving attribute constraints) and also traversal through the edge connectivity. For this type of search, nodes are to be efficiently indexed. Multidimensional data include points, line segments, polygons, rectangles, regions, and volumes in 2D, 3D or higher. Since each node has multiple values associated with them, these are represented as multidimensional points and spatial indexing methods are used for indexing these points. In this paper various indexing trees which deal with spatial objects and multidimensional points are studied and compared.

The rest of the paper is organized as follows. Section 2 includes properties of multiple attribute graph and justifies the requirement of using multidimensional indexing. Also various querying methods are discussed. Section 3 includes overview of various multidimensional indexing methods and specifies the most accurate indexing method. Section 4 compares these methods and section 5 analyses the performance. Section 6 concludes the paper.

II. MULTIPLE ATTRIBUTE GRAPHS

Graphs in general are represented as set of nodes and edges which connect these nodes. Graphs with multiple attributes associated with each node having non negative weights assigned to each attribute are called multiple attribute graphs. Fig.1 shows an example of multiple attribute graph produced from social network.

Users are connected with edges when they are friends. Circles near to users indicate the values of each attribute associated with the user. Here LK defines the likes done and PO postings and CO commenting by user. The query is to find users who are friends and having attribute values as given in the figure having greater or lesser value than a given constraint.

A. Properties of Multiple Attribute Graphs

1) Multiple attribute graphs are complex to represent as the connectivity constraints and attribute values are to be described.

2) Multiple attribute graphs are usually very dynamic in their structure since they represent real world data each time graph grows in size and complexity

3) Unlike normal graphs they are too large, to handle with naïve search and manipulation algorithms. Dynamically adaptable algorithms and techniques are to be used for manipulation.

Multiple Attribute Graphs' node consists of weighted attributes for each nodes and so for making search efficient we build indices upon these nodes which collectively address for attribute and connectivity constraints.

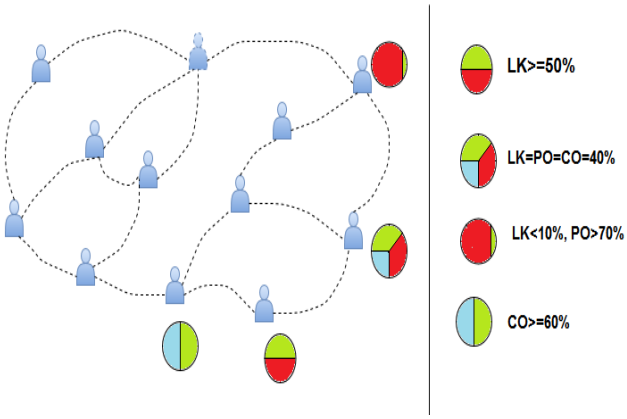

Fig. 1 Multiple attribute graph for social network

B. Applications of Multiple Attribute Graphs

1) Co-authorship graph: Each node represent authors, relation between them shows the co-authorship and is represented by using edges and weights on each attributes show expertise of each author in particular domain.

2) Social networks: Nodes represent users, friends are linked (connected) using edges and their online activity is represented using attributes. Users are defined with attribute values such as liking, commenting, posting etc.

3) Communication network: Hubs, switches are represented as nodes and various ports (registered posts, well known posts, ephemeral posts etc.) associated with each routing device is taken as attributes. Edges between these devices show that they communicate with each other.

4) Chemical compounds: Chemicals present in compounds are given as nodes and the bonding between them is given using edges. Attribute values are properties of each chemical itself such as atomic number, chemical and molecular properties.

5) Biological networks: For representing DNA structure etc graphs can be used where nodes represent the components of these and the edges show the pairing.

C. Requirement of Multidimensional Indexing

1) Weighted multiple attributes of each node can be efficiently represented as multidimensional points on Hilbert space. These capture the properties of Multiple Attribute Graphs attribute values.

2) For searching on large Multiple Attribute Graphs the entire nodes are to be collectively considered on a single structure. Traditional DFS, BFS searches does not work for Multiple Attribute Graphs since we have to consider multiple attributes.

3) As nodes are easily represented as multidimensional points, multidimensional indexing methods can be used for indexing these points.

4) Multidimensional indexing can efficiently prune the whole search space to those data that are relevant to our query.

Spatial indexing methods or multidimensional methods are applied to index these large set of nodes.

D. Queries on Multiple Attribute Graphs

1) Point query: Query includes a particular set of values associated with the attribute defined for Multiple Attribute Graph's nodes, which is considered as description of a single node. It searches for an exact match to the given constraint. Nodes having the same values in its attributes are retrieved as results. For

example in the case of social network as defined above if the query given is to find nodes satisfying the condition LK=PO=CO=40% then it is a point query. This tries to find nodes having attribute values exactly matching to the given constraint.

2) Range query: Query defines a range of value for each attribute defined for Multiple Attribute Graphs nodes. A minimum and maximum value limits are defined for attributes. These may include values for all or some attributes. Searching for nodes matching with range query retrieves approximate matches. All nodes having their attribute value within the limits of given query are returned as result. For example in the case of social network if the query given is like LK>50% then it retrieves all those nodes having LK attribute value higher than 50 regardless of the values of other attributes.

III. MULTIDIMENSIONAL INDEXING METHODS

There are many types of spatial or multidimensional indexing methods. Traditional indexing methods are not suitable for these multidimensional or Multiple Attribute Graphs data. For constructing indices for Multiple Attribute Graphs each nodal attribute weights are to be transformed as multidimensional points.

A. Quadtree

Quadtree [2] is a tree data structure which is used to index data that are represented in a two dimensional space (2D). Quadtree was named as so by Raphael Finkel and J.L. Bentley in 1974. In the construction of quadtree the entire two dimensional search space is initially split into four quadrants and again for each of these four partitions the splitting is done repetitively. The entire objects which are placed in the search space is contained in any of the division. These four divisions are also called as quadrants which indicated the splitting of geometrical 2D plane. The partitioned squares are indexed as quadtree where each node points to a square in the search space. Here for each node there will be exactly four children. The splits are usually squared which may be in rectangle or any other shape. Fig. 2. shows an example of quadtree.

The children of root are labelled as NE, NW, SE, SW for indicating the quadrants of the 2-D space. They are used to index point datas, areas, curves, surfaces, and volumes. The decomposition may be into equal parts on each level or according to the input data. At each index level each of the outer level quadrants are given. Inner quadrants appear as children of these outer quadrants. All data points are included in any of the quadrants and appear at the leaf of tree.

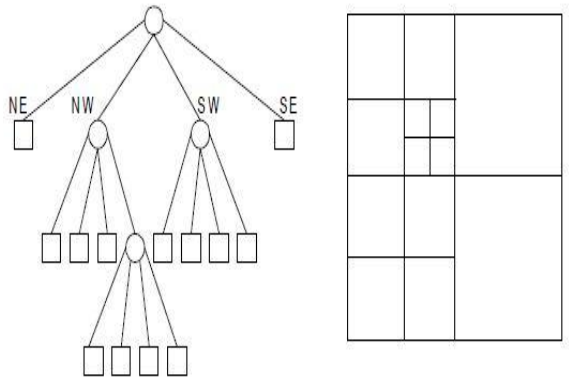

Fig. 2. Quadtree and its divisions

B. k-d Tree

K- Dimensional tree which is abbreviated as k-d tree[3] is a tree data structure which is used to index objects represented in a k- dimensional space. These trees are commonly used in the case where k- dimensional searches are involved (for eg: if a multi dimensional keyword say having k values is to be searched in a search space k-d tree can be used). K-d tree also supports range queries where we assign various values to each of the k dimensions. The search space indexed by k-d tree is a k- dimensional space in which data points have at most k attribute values. These are indexed by drawing hyper planes. K-d tree is defined as an alternative of binary tree. In case of binary tree for each level the search is split or reduced by a factor of two where in this case also for each level the search space is reduced by a factor of two. This is done by drawing a k-1 dimensional hyper plane through one of the data points which splits the search space into two. Through each data point the planes are drawn and these hyper planes occur as nodes of k-d tree. Fig. 3. shows a 2-d tree. Here each object is given as a point and a line is drawn through each of the point. The lines drawn appear as nodes of the tree. Points to the left of splitting hyper plane (which is indexed as internal node) are represented by the left sub tree of that node and points to the right of the hyper plane are represented by the right sub tree of that node.

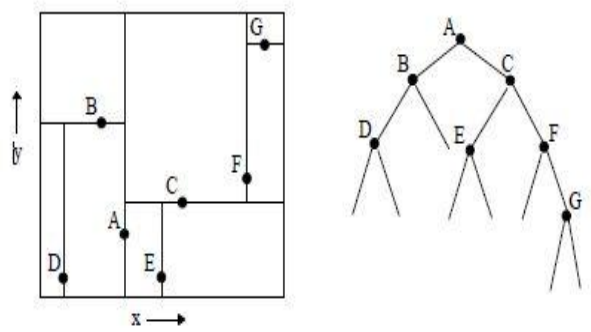

Fig. 3. 2-d Tree

C. Octree

An octree [4] is also a tree data structure which is considered to be a 3- dimensional variant of quadtree. In case of quadtree it indexes 2 dimensional search space where as octree indexes 3 dimensional space. Here each of the nodes of octree has exactly eight sub trees. In case of octree indexing the entire space is initially divided as eight cubes and recursively each of these cubes are divided to smaller cubes and so on. Entire data are indexed by either of the cubes. Octrees are often used in 3D graphics and 3D game engines.

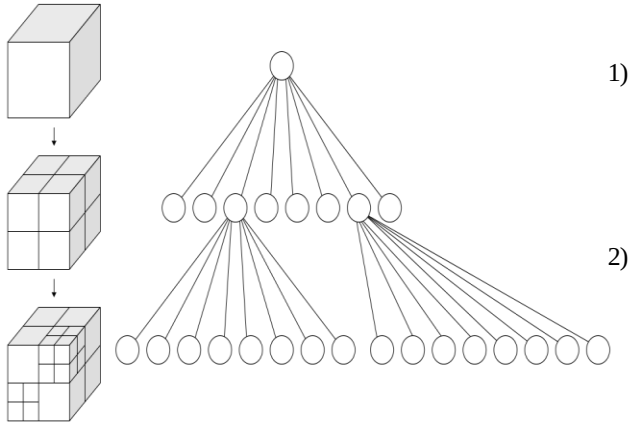

Fig.4 . Octree

D. R-Tree

R-Trees [5] are hierarchical data structures used for indexing multi-dimensional data. Objects usually indexed by R-Tree are spatial objects. The construction of R-Tree involves bounding the spatial objects using minimum bounding boxes (MBBs). R-tree nodes, instead of indexing actual objects, indices these minimum bounding boxes. The leaf level nodes contain pointers to actual objects. Each of the minimum bounding box is again grouped to higher level boxes recursively until entire search apace is bounded to a single large box which appears as the rood node. A minimum bounding box is the one which is the smallest box that can contain those objects and no other smaller box with a lesser dimension can be found to include all those objects. As B-tree (balanced tree) R-trees are also height balanced and has efficient storage utilization. The key idea of this data structure is to group spatial objects that are close to each other in a n-dimensional space. Grouping is done by creating MBBs over these objects. The letter "R" of R-tree stands for rectangle (ie. In case of two dimensional spaces the points are bounded by minimum bounding rectangles). The process of grouping spatial objects into MBBs is known as splitting and they are of many types such as: linear split, quadratic split, etc which are classified according to the way grouping is done. All multi-dimensional objects represented in the search space is include in either of the MBB. For a given query with the range values of n-dimensional attributes query boxes are created and searched in the R-Tree. The node MBBs are checked for overlap with the query box. If no MBBs of tree overlap with query box then there is no possibility that any of the object overlap with the query box. Leaf level boxes only index single spatial objects. R-tree efficiently prunes the entire search space. The R-Tree properties include:

1) Let M be the maximum number of entries that can come in an internal node of R-Tree and m , the minimum number of entries that a node should have. If the node is an internal node of tree it should be within the condition of upper and lower limit, but in case of root or leaf node there can be less than m entries.
2) In case of insertion operation to tree node, if it exceeds the upper limit M then the nodes are split to form the next level of indexing thus increasing the height of R-Tree. If deletion operation makes the number of entries in a node falls below the minimum bound m the rest of its entries are distributed among the sibling nodes by merging.
3)
4) The root contains at least 2 entries, unless it is a leaf.
5) R-tree is a height-balanced tree; every leaf node has the same distance from the root. The height of the tree is at most $\lceil \log_m N \rceil$ for N index records (N > 1).

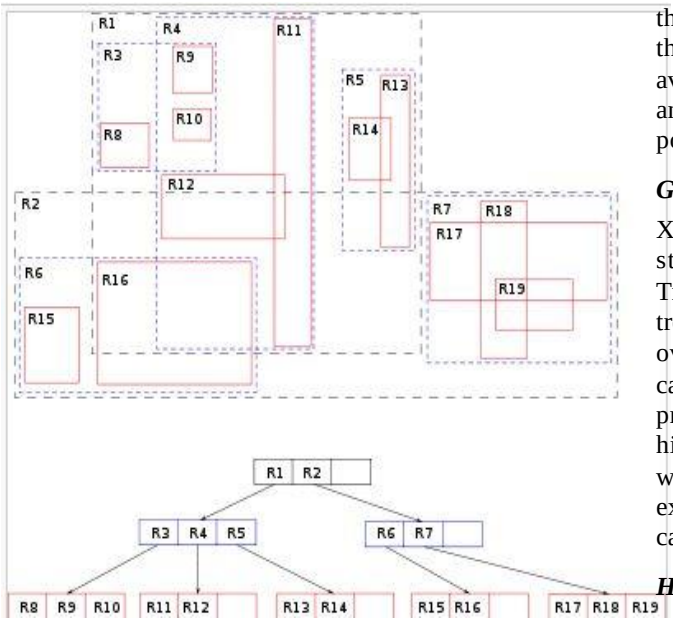

Fig. 5. R-Tree for 2 dimensional objects

E. R+ Tree

The R+-tree was introduced by Sellis et al.[6] in order to overcome the problem caused by the overlapping of child node of same parent. In case of construction of R-Tree if a spatial object shares two internal nodes of same parent the search will not be efficient. In order to avoid this R+-tree indices those spatial objects which share the internal nodes separately under each of them by creating separate leaf nodes. R+ -Tree is compromise between k-d tree and R- tree. By making the nodes indexed to different nodes the search can progress to a single branch as in normal way, but it suffers from redundancy problem.

F. M Tree

M-tree[7] is also a tree data structure which is similar to R-Tree and B-Tree which are efficient, balanced indexing structures. M-tree can index multidimensional spatial objects. It uses a distance metric to construct the nodes and also considers the triangle inequality. Spatial objects are grouped into balls having a particular radius. These balls are indexed as internal nodes of the tree. Let r denote the radius of the ball and N be the internal node of the tree all entries say m coming in the node has a maximum of only r distance from N. M-Tree efficiently works for range queries and k-NN(k nearest neighbour) queries. Each of the ball of M-tree contains a portion of the entire search space. M-Tree works well for many of the conditions but the disadvantage is that there is a high overlapping between the balls and thus the nodes of tree. There is no efficient way found to avoid this problem. M-tree consists of nodes and leaves and each of the nodes has a data object identifier and a pointer to the sub tree where its children appear.

G. X-Tree

X-Tree (eXtended node tree) [8] is an indexing tree data structure which is different from that of R-Tree, R+-Tree and R*-Tree and was appeared on 1996. Both R-tree and R+- tree does not address the problem of overlapping of nodes which becomes a major issue in case of higher dimensions, where X-Tree addresses this problem. In case of R-Tree if the node split results in higher overlapping the split is deferred to a later time which causes in the occurrence of super nodes. In extreme cases tree will linearize which avoids the worst case behaviour seen in other data structure.

H. Hilbert R-Tree

Hilbert R-tree[9], is an R-tree variant, used for indexing multidimensional objects like lines, regions, 3-D objects, or high-dimensional parametric objects. It can be also thought as an extension to B+-tree for indexing multidimensional objects. Hilbert R-trees make use of space-filling curves, and specifically the Hilbert curve, to impose a linear ordering on the data rectangles. Hilbert curves are drawn through data objects to order them.

These are some of the various techniques used for indexing multidimensional points or data. All these indexing methods can also be applied to index multiple attribute graph nodes. Nodes of multiple attribute graphs are represented as k-dimensional points on k-dimensional space and these are indexed using the suitable methods as mentioned.

IV. COMPARISON OF INDEXING METHODS

Many indexing methods have been studied and found all these can be applied for indexing multiple attributed graph nodes also.

TABLE I
WAG INDEXING METHODS

Index	Results and Observations		
	Run Time Complexity	**Advantage**	**Disadvantage**
Quadtree	O(log n) search for point quadtree. 2 dimensional analog of octrees	Equal partitions and efficient , easy way of indexing	Used for 2 dimensional data only.
k-d Tree	O(log n) for search ,insert delete etc	Can index upon multidimensional data	Depends on insertion order. Partition is not balanced since based on the points
Octree	O($\log_8$ n)	Can be used in 3-D graphics, set, state estimation etc.	Used for 3 dimensional data only.
R-Tree	O($log_m n$) for search and O(n) for insertion	Efficiently prunes unwanted search space. Dynamically adapts to insertion and deletion of nodes.	Occurrence of overlapping
M-Tree	O(log n) for insertion and O(n) for querying	Efficient for range and K-NN queries	High overlap Only used for things that satisfy triangle inequality and having distance measures
X-Tree	O(n log n) for build	Avoids overlapping	Not efficiently implemented.

R-tree is considered as an efficient data structure for indexing multidimensional points. It is fast in splitting the points and grouping them to minimum bounding boxes efficiently. A good split helps in avoiding the overlapping of nodes in internal nodes without using redundant multiple entry method. Splits that form MBBs are efficient for indexing closely related multidimensional points together by reducing the overlapping and providing a high pruning data structure. In case of search it reduces the search space significantly. Build and search time complexity of R-tree is much less than any other indexing methods. The balancing property of R-Tree makes it dynamically adaptable to inputs given.

V. EXPERIMENTAL EVALUATION

Datasets used in the experiments for evaluating the working time of R-Tree are DBLP dataset, online social network data extracted from You tube, communication network dataset from LBL (Lawrence Berkley National Laboratory) which was modified to reduce the number of vertices and edges to make them smaller to compute. All these dataset consists of thousands of nodes which are linked to each other. For each of these dataset nodes attribute values are calculated. Experiments are done to evaluate the runtime and build time of R-Tree. Results are summarized as follows:

Dataset	**#Vertices**	**#Edges**	**Build time(sec)**	**Runtime (sec)**
DBLP dataset	100	563	4	1
You tube dataset	100	621	5	1
LBL dataset	150	638	5	2

VI. CONCLUSION AND FUTURE WORK

Analysis of complex structures like graphs with multiple attributes for nodes, has gained importance with the growth of real world networks. This work aims to study the properties and applications of weighted multiple attribute graphs and its representation in computer memory. Executing complex queries on such graphs demands efficient look ups of nodes which satisfies the query constraints. By indexing multiple attribute graphs, the search process can be made efficient. In this work, various indexing methods were analyzed and observed that R-Tree is an efficient spatial indexing method that can be used for multiple attribute graph

indexing. Performing sub graph search and graph search based on structural and attribute constraints are two interesting directions of future work.

REFERENCES

[1] Senjuti Basu Roy, Tina Eliassi-Rad, and Spiros Papadimitriou,"Fast Best-Effort Search on Graphs with Multiple Attributes" *IEEETran Know. Data. Eng*. vol 27. No.3, pp 755 March 2015.

[2] Hee Kap Ahn ,Nikos Mamoulis and Ho Min Won "A Survey on Multidimensional Access Methods", May 2001.

[3] Bentley, J. L."Multidimensional binary search trees used for associative searching". *Communications of the ACM , pp,509*, 1975.

[4] Meagher, Donald. "Octree Encoding: A New Technique for the Representation, Manipulation and Display of Arbitrary 3-D Objects by Computer", oct 1980

[5] A. Guttman, "R-trees: A dynamic index structure for spatial searching," *in Proc. ACM SIGMOD Int. Conf. Manage. Data*, 1984,pp. 47–57.

[6] Sellis et al, "The R+-Tree: A Dynamic Index for Multi-Dimensional Objects", *in Proc. ACM Int. Conf. on Very Large Data*, 1987.

[7] Ciaccia, Paolo; Patella, Marco; Zezula, Pavel "M-tree An Efficient Access Method for Similarity Search in Metric Spaces". Proceedings of the 23rd VLDB Conference 1997.

[8] Berchtold, Stefan; Keim, Daniel A.; Kriegel, Hans-Peter (1996). "The X-tree: An Index Structure for High-Dimensional Data". Proceedings of the 22nd VLDB Conference (Mumbai, India): 28–39.

[9] I. Kamel and C. Faloutsos. Hilbert R-tree: An improved R-tree using fractals. In Proc. of VLDB Conf., pages 500–509, Santiago, Chile, September 1994.

[10] https://en.wikipedia.org/wiki/R-tree

[11] https://en.wikipedia.org/wiki/Octree