Annals of Pure and Applied Mathematics
Vol. 13, No. 1, 2017, 49-62
ISSN: 2279-087X (P), 2279-0888(online)
Published on 31 January 2017
www.researchmathsci.org
DOI: http://dx.doi.org/10.22457/apam.v13n1a6

Annals of
Pure and Applied Mathematics

Labeled Object Treemap: A New Graph-Labeling Based Technique for Visualizing Multiple Hierarchies

Mahipal Jadeja[1] **and** ***Rahul Muthu***[2]

[1]Dhirubhai Ambani Institute of Information and Communication Technology
Gandhinagar– 382007, Gujarat, India. E-mail: jadeja_mahipal@daiict.ac.in

[2]Dhirubhai Ambani Institute of Information and Communication Technology
Gandhinagar – 382007, Gujarat, India. E-mail: rahul_muthu@daiict.ac.in

Received 6 January 2017; accepted 23 January 2017

Abstract. Information visualization is a class of techniques which is used to present data in a graphical or pictorial format. Identification of new patterns as well as an understanding of difficult concepts is possible with the proper use of visualization. Using interactive visualization, various other details related to the information can be obtained. In this paper, we consider trees in which each node is related to a leaf node (object) of taxonomy. We propose a new technique of visualization namely 'Labeled Object Treemap' for the visualization of multiple hierarchies. In our approach, a unique label is associated with each node and this label will convey all the necessary information including adjacency as well as non-adjacency with the other nodes of the tree. The comparison of our proposed technique with already known techniques is also made. Here we develop and use a labeling of trees where vertices represent distinct sets and adjacency coincides with disjointness. The total number of distinct elements used in the underlying labeling is asymptotically minimized. The motivation of selecting set labeling is to use cardinalities of labels to identify level numbers of the underlying tree using which it will be easier to discover adjacency as well as non-adjacency for all vertices. Our motivation to look at disjointness instead of intersection is that several well-known graphs like the Petersen graph and Kneser graphs are expressed in the latter method. Our main contribution is the development of a new visualization technique which solves the issues of edge crossing and continuity of the 'Trees in a treemap' visualization technique while maintaining all the good characteristics of existing methods for visualizing multiple hierarchies. Additional features are also discussed using the modified labeling algorithm.

Keywords: Vertex labelling; graph drawing; taxonomy; visualizing multiple hierarchies; information visualization; treemapping

AMS Mathematics Subject Classification (2010): 05C62, 00A66, 05C78

1. Introduction

Key terms which are related to multiple hierarchies are explained in this section.

Hierarchy: A hierarchy refers to an arrangement of items in which items are highlighted as being "below" or "above" or "at the same level as" one another.

Structure: Structure refers to an arrangement and organization of interrelated elements of the system.

Tree structure: Hierarchical nature of the given structure can be represented using tree structure or tree diagram in a graphical form.

Tree: Simulation of tree structure is possible using trees which are widely used data structures for the implementation of abstract data type.

Taxonomy: Taxonomy refers to the practice and science of classification of concepts/things and it also includes the principles that underlie such type of classification.

Taxonomy Trees: They are special trees in which objects are represented by leaf nodes only, and classification is represented by all other nodes. An example of a taxonomy is shown in Figure 1.

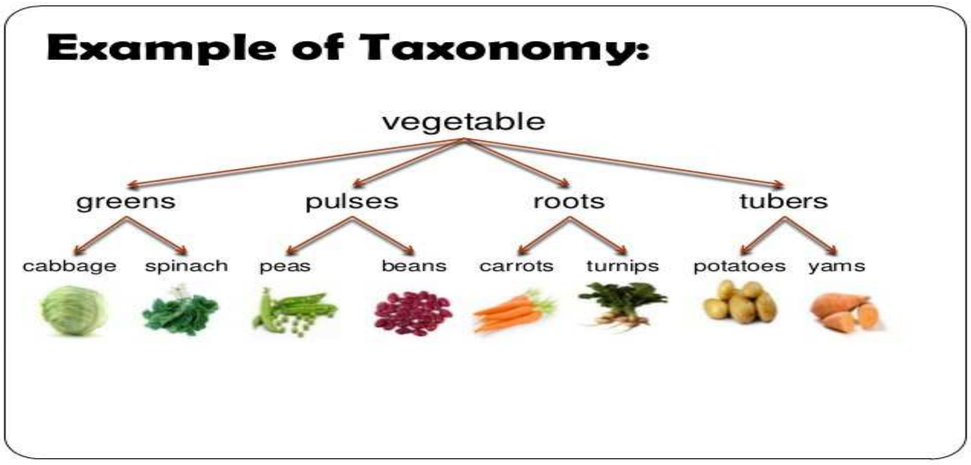


Figure 1: Taxonomy

Treemap: The treemap algorithm offers a practical way of displaying large trees (with millions of nodes) in limited space. The approach of treemapping is recursive. One box is allocated for the parent node and children of the node are represented as boxes within it. Practically using this approach it is possible to render any tree within a standard size display. Treemaps and its variants are studied in detail in the literature [3,4,5,6,7].

For many applications it is necessary to consider two aspects: the relationship between different objects and identification of the object type. One can show these aspects using two different trees: 1) taxonomy tree 2) the other tree where each node is related to leaf nodes (object) of a taxonomy. The problem is to design a visualization technique which effectively conveys both the desirable features i.e. relationships between different objects (object tree) along with the mapping of each object with taxonomy. Generally, in order to represent hierarchical data, directed trees are used.

To the best of our knowledge, the best-known visualization technique for representing multiple hierarchies is: Trees in a Treemap technique [2]. The simplest solutions are shown in Figure 2 in which a) both the trees are drawn side by side. Here, it is not possible to identify frequencies as well as locations of different object types quickly in the underlying object tree. b) Connections between the two trees are shown using additional edges. Neither of these solutions is efficient for big data. Other visualization techniques for the representation of the same object tree (which is shown in Figure 2) are discussed briefly in the comparison section. The output of our proposed method for the same object tree is also presented for the sake of completeness in the same section (see Figure 10).

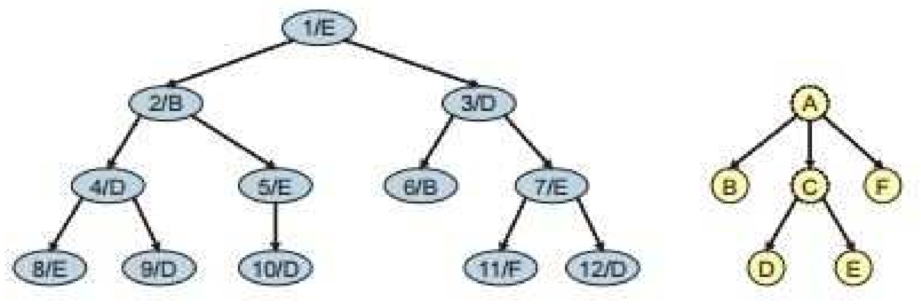
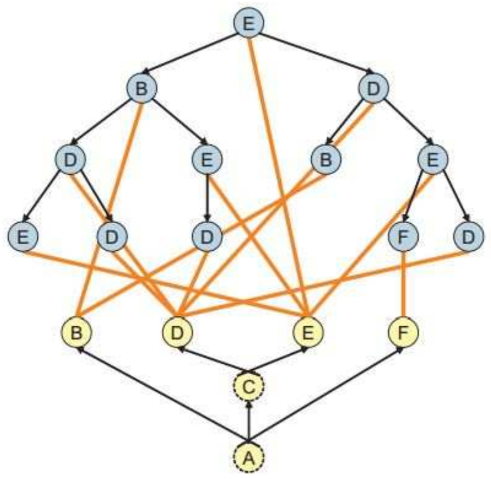


Figure 2: (a) Two separate tree diagrams



(b) Linked tree diagram

1.1 Related work

Trees in a treemap technique belong to the agglomeration class of visualizing multiple trees [9]. Agglomeration: Here single representation is used in order to display multiple parents of a node. One possible option is to replicate nodes with at least 2 parent links across the trees in order to maintain the overall structure more like a tree structure. Node link representations are also used in which multiple tree structure is shown as the graph structure (structure with cycles). For example, GLAD system [10]. Directed placement of various overlapping treemaps is used in CristalView [11] in which shared nodes are used to communicate across multiple hierarchies. The success of the selected visualization technique depends upon various factors including target audience and user interface [12]. Other related visualizations are ZTree [13], Multitrees [14] and animation [15].

Summary of key contributions:

1. Development of a labeling algorithm for trees which uses asymptotically minimal number of labels.

2. Application of this labeling scheme in information visualization domain and solving existing issues of edge crossings and continuity in visualizing multiple hierarchies.

3. Modified labeling algorithm using which it is possible to identify neighbors as well as non-neighbors quickly by reducing the underlying search space for all nodes.

2. Discussion on the proposed labeling scheme

A set labeling of a graph G(V, E) is a function $f : V \rightarrow P(\{1, 2, \ldots, k\}) - \{ \Phi \}$ where $k \in Z^+$ such that

- f is one one. (Two distinct vertices must not get the same set label.)
- $\forall u, v \in V, (u, v) \in E \Leftrightarrow f(u) \cap f(v) = \Phi$

Universe size number (usn) of a graph is the least positive integer k such that a set labeling of G exists.

The closely related concept is intersection graphs [16] for finite sets in which non-adjacency is characterised by disjointness of the corresponding subsets of the underlying universal set. Other types of graph labeling are very well studied in the literature [17,18].

2.1. General results on usn

Here, we present general bounds on usn.

Theorem 2.1. usn(G) ≥ floor($\log_2 n$) + 1where G has n vertices.

Proof: Using floor($\log_2 n$) elements, at most n−1 non-empty subsets can be generated which, due to the requirement of label distinctness, can be used to label at most n−1 vertices. Here total number of vertices is n. In order to assign a non-empty as well as unique label to n^{th} vertex 1 additional element is required in underlying universe. Therefore value of usn is at least floor($\log_2 n$)+1 for any given graph.

3. Results on usn for trees

In this section, we derive results on usn (either exact or asymptotic) on the following classes of graphs: Binary trees, k-ary Trees and Trees.

Theorem 3.1. usn(BT_n) = O(log n), where BT=Complete binary tree and n denotes the total number of vertices of the BT.

Proof: We prove this result by an iterative algorithm: ValidTreeLabelling. The algorithm is explained below:

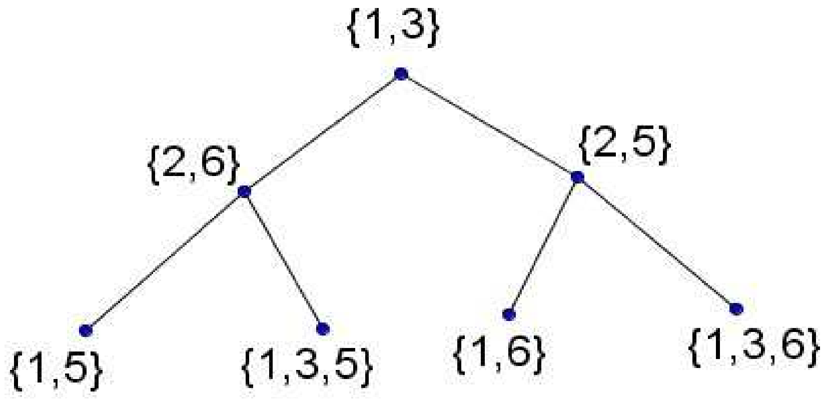


Figure 3: Input

Input: A valid labeling of the complete binary tree of height h using exactly K labels.
Base case: usn(BT_7) = 5. Base case is shown in Fig. 3 with underlying labeling set: {1,2,3,5,6}

Output: A valid labeling of complete binary tree of height h + 1 using exactly K + 4 labels. The four new labels are a, b, c and d.
//During the phase of the algorithm when the number of levels increases from L to L + 1, labels of vertices which are present at level L-2, L and L+1 will be changed. Labels of all other vertices will remain as it is.//

Step 1: For all newly added vertices in level L+1, find their corresponding ancestors in level L-1.
For all v (where v is a level L+1 vertex),
Label(v) = Label(ancestor(v) in level L-1). (The levels L+1 and L-1 are highly similar with respect to adjacency with the remaining levels. Both of these levels are adjacent to level L and non-adjacent to 1, 2, 3, ..., L-3).

Observation after step 1:
1. All the vertex-labels which are present in level L+1 will have non-empty intersection because corresponding ancestors are either same or having non-empty intersection. This is desirable because all vertices of level L + 1 are non-adjacent.
2. Layer L + 1 and L-1 are non-adjacent and they have non-empty intersection after this step.

Step 2:
The level L-2 is adjacent to L-1 but not to L + 1.
For all v_i, (where v_i is a level L-2 vertex)
New-Label(v_i) = Old-Label(v_i) ∪ {a, b, c, d}

Step 3: Consider sequential ordering (left to right) of vertices which are present in level L+1.
For each vertex v_i,
New-Label(vi) = Old-Label(v_i) ∪ {a} if i mod 4 = 1
= Old-Label(v_i) ∪ {b} if i mod 4 = 2
= Old-Label(v_i) ∪ {c} if i mod 4 = 3
= Old-Label(v_i) ∪ {d} if i mod 4 = 0

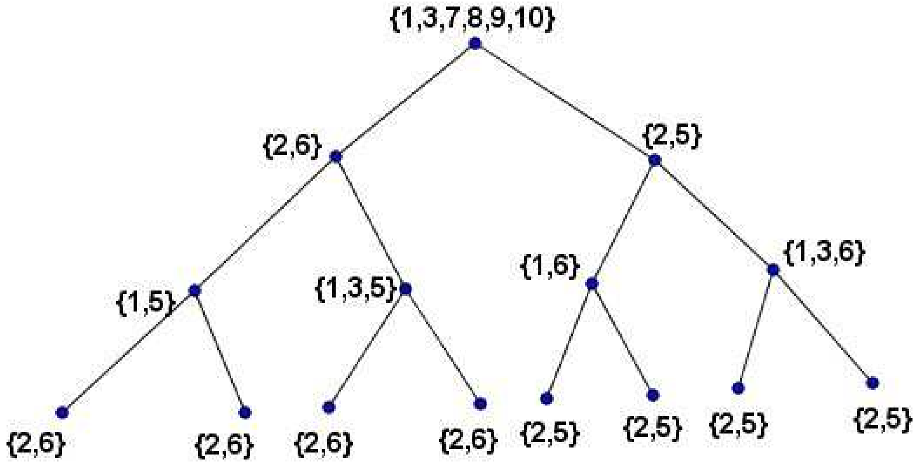


Figure 4: After Step 1 and 2

Vertices of level L − 2 and L + 1 will preserve non-adjacency because the corresponding labels have non-empty intersection after this step.

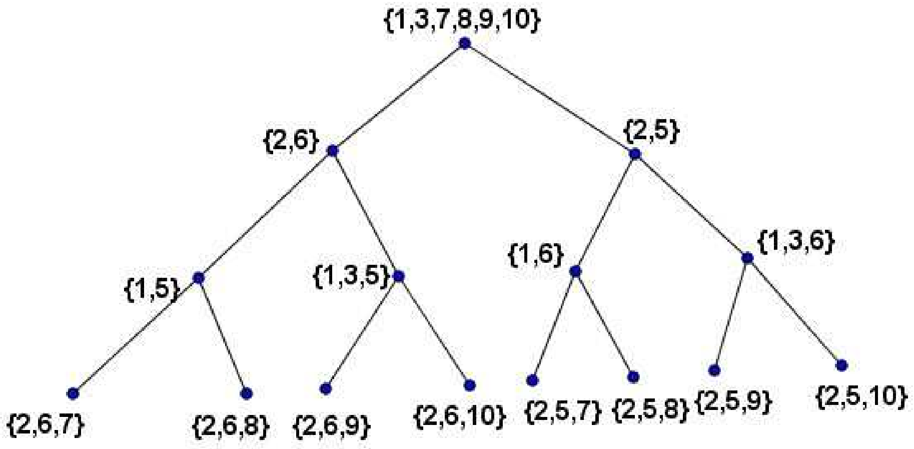


Figure 5: After step 3

Step 4: Consider sequential ordering (left to right) of vertices which are present in level

L: $v_1, v_2, \ldots, v_q$

For each vertex v_i,

New-Label(v_i) = Old-Label(v_i) ∪ {c, d} if i mod 2 = 1

= Old- Label(v_i) ∪ {a, b} if i mod 2 = 0

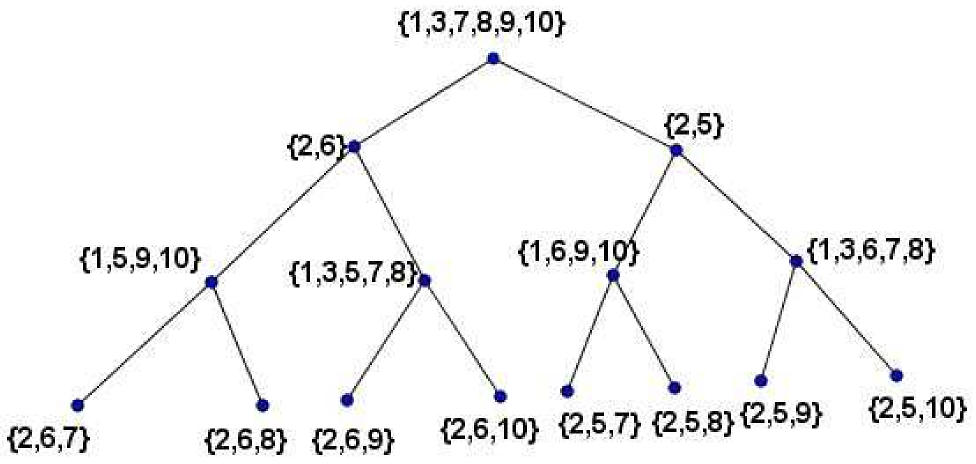


Figure 6: Output

Observation after step 4:
Level L + 1 and L will preserve adjacency as well as non-adjacency. The final labeling is valid and preserves adjacency as well as non-adjacency.

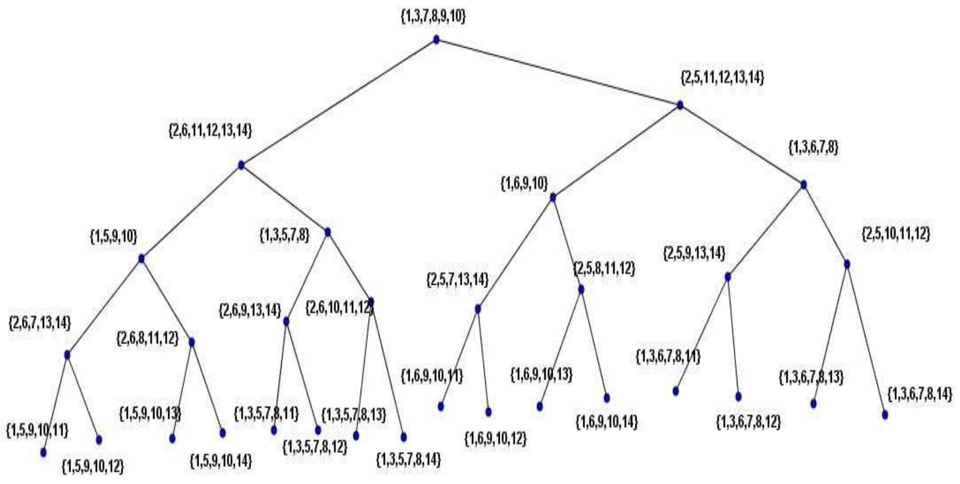


Figure 7: After 2nd iteration

For the complete binary tree, n = 1 + 2 + 4 + ...+ $2^h = 2^{h+1}-1$ i.e. h = O(log n). For valid labeling of each layer exactly 4 additional elements are required. Total number of layers are O(log n). Therefore total number of elements required are 4*O(log n) which is O(log n).

The next two Theorems are consequences of Theorem 3.1.

Theorem 3.2. USN of any k-ary tree is O(log n) where number of vertices present in the T is n.

Proof: Using similar algorithm which is described for complete binary tree, it is possible to do valid labeling of each layer of k-ary tree using exactly k^2 additional elements. k^2 is a constant which is not dependent upon n and total number of layers are O(log n). Therefore total number of elements required are k^2 *O(log n) which is O(log n).

Theorem 3.3. Valid labeling of any given tree can be constructed using O(log n) labels.

Proof: For the given tree, calculate maximum degree a. Consider underlying (a-1)-ary tree, with the same number of levels as the given tree with valid labeling using O(log n) labels. Convert the (a-1) ary tree into the given tree by taking vertex induced sub-graph. For all the vertices which are present in the tree, copy the corresponding labels from the underlying (a-1) ary tree. The resultant labeling is valid and uses at most O(log n) labels.

3.1. Discussion on additional features

It is possible to obtain the following crucial features with the use of modified labeling algorithm using which identification of neighbors as well as non-neighbors will be easier.

Feature 1: Identification of non-neighbors quickly

Vertices having same label cardinalities are in the same level. So no edge between them. (after ith iteration, this observation is true for all levels except level i + 1,i + 2 and i + 3.)

Feature 2: Identification of neighbors quickly by reducing the search space

In trees, edges are present only between adjacent layers. After applying the modified labeling algorithm, it is possible to assign level number correctly to each vertex. This will make search easier and the user can easily identify object connections.

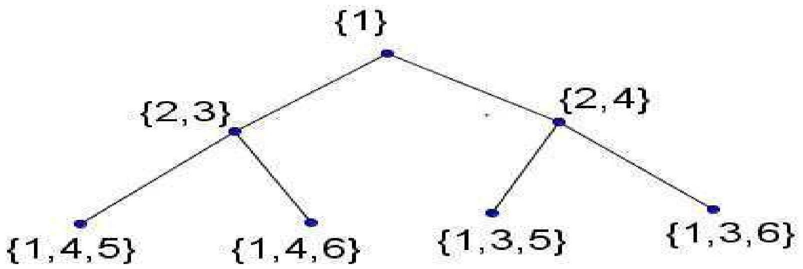


Figure 8: Input for the modified algorithm

Explanation of modified labeling algorithm. The objective of the modified algorithm is to generate labels of unique cardinalities in increasing order for all levels (except the last 3 levels) and assign same cardinality labels to all vertices which belong to the same level. i.e. level 1 must have the minimum cardinality label whereas level i must have labels with the maximum cardinality. In order to obtain the desired features, the following two modifications are required.

1. Input tree. The modified input tree with valid labeling is shown in Figure 8.
2. After applying Valid Tree Labelling algorithm, add {e} to all vertex-labels which are present at level L + 1. So overall use of 5 new elements instead of 4.

Let C_i denote cardinalities of all vertex labels which belong to level i. The algorithm adds exactly 1 level after each iteration. The input complete binary tree has 3 levels. Therefore after i iterations, total i + 3 levels will be generated. Before applying the modified algorithm, the cardinalities of labels are 1, 2, 3 (i.e. $C_1= 1$, $C_2= 2$ and $C_3=3$).

After first iteration, new cardinalities: 5,2,5,4. After second iteration: 5,6,5,6,7. After third iteration: 5,6,9,6,9,8. In general after ith iteration first i entries will be sorted in the increasing order and $C_i= C_{i-2}+ 4$ (true for all values of j such that $1 \leq j \leq i$), $C_{i+1}= C_{i-1}$, $C_{i+2}= C_i$ and $C_{i+3}= C_{i+1}+ 2$. Using these details, it is possible to quickly identify the location of each vertex in the underlying tree hierarchy.

4. Description of our proposed method and its advantages

Labeled object treemap technique is very similar to the 'Trees in a treemap' technique [2]. In the 'Trees in a treemap' technique, nodes are represented by small circles which are placed in the boxes that represent the objects associated with the nodes. Edges are represented by lines which connect these small circles.

According to us, the 'Trees in a treemap' method has the following problems:

1) Edge crossing is present in both the variants of the technique.

2) Continuity: For very large data sets, it is very difficult to follow the edge in order to identify whether those two objects are connected or not.

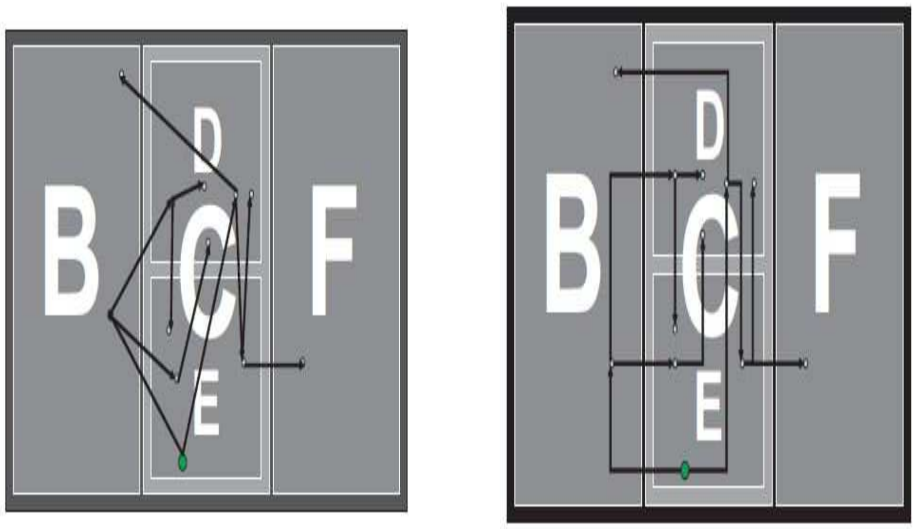


Figure 9: Trees in treemap visualization technique (from [2])

Our proposed method solves these problems by constructing an equivalent labeled representation which does not contain edges and the total number of labels which are used in the representation are asymptotically minimal. i.e. O(log n). If the labels of two objects are disjoint then they are adjacent. i.e. edge is present between them. The size of each label is at most O(log n). Hence, it is possible to represent the information about edges using an O(n) $*$ O(log n) matrix. Each row in the matrix corresponds to the label of a node. If element j is present in the label of the i^{th} node then entry (i, j) = 1 else 0. In the worst case O(log n) comparisons are required in order to determine whether two nodes are adjacent or not. In the best case, only constant time is required for the same case because it may possible that the 1^{st} element is present in both the labels.

Advantages: It offers a single representation in order to visualize object tree as well as taxonomy. Crossings are not present in the representation. Continuity is not visible but one can still obtain the required information by a careful observation of labels. It is possible to detect clusters and outliers using our proposed method. The visualization technique is highly compact because it requires just O(n) $*$ O(log n) space in order to store information about edges. This is better as compared to sorted/unsorted adjacency matrix technique because they require $O(n^2)$ space for the representation. Taxonomy is represented using treemap.

From the previous section, it is clear that it is possible to give a valid labeling of any given tree using O(log n) labels. By using theorem 3.3 on the output of the algorithm (see figure 6), it is possible to construct a valid labeling of the given input tree. The final visualization is shown in the figure 10.

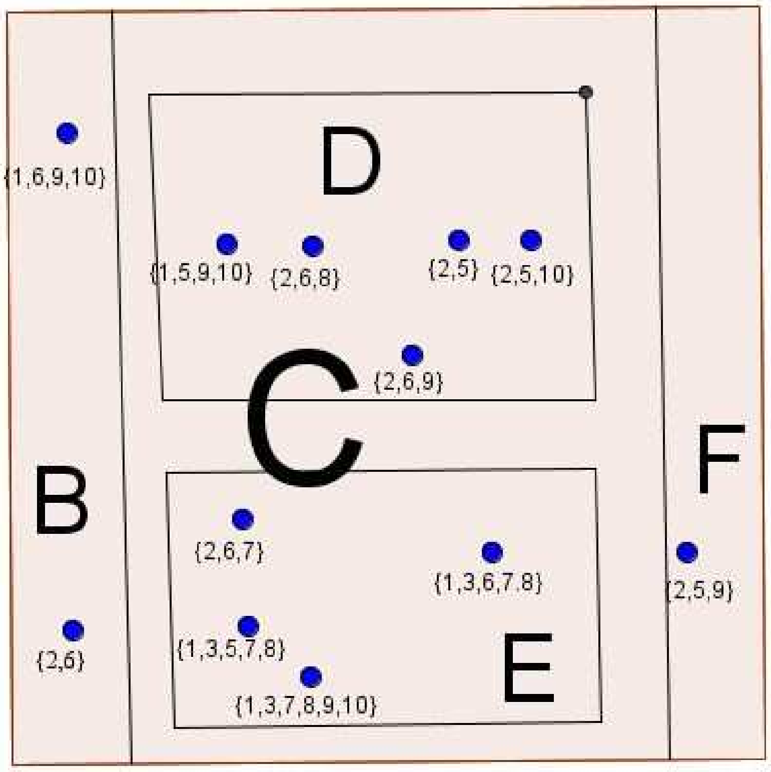


Figure 10: Our proposed method: labeled object treemap

5. Comparison with existing methods

Calculation of leaf word of the taxonomy:

lw(t) = t if t is a leaf node

= concatenation of $lw(t_1)$, $lw(t_2)$,...,$lw(t_k)$ otherwise

Here $t_1 \ldots t_k$ are subtrees of t.

For the given taxonomy, valid leaf words are BDEF, FEDB and DEBF.

In the space filling visualization technique, adjacency matrices can be used. In Figure 11, adjacency matrix representations are shown for sorted as well as unsorted dimensions. For sorting, a leaf word is used.

In the colored tree diagrams, colors are assigned to the leaf nodes. This assignment is based upon the order of occurrence of a leaf node in the leaf word. Similar colors are assigned to objects which are close to one another in the underlying taxonomy (See Figure 12). Leaf word is also used for sorting objects in Parallel coordinate views technique (See Figure 12).

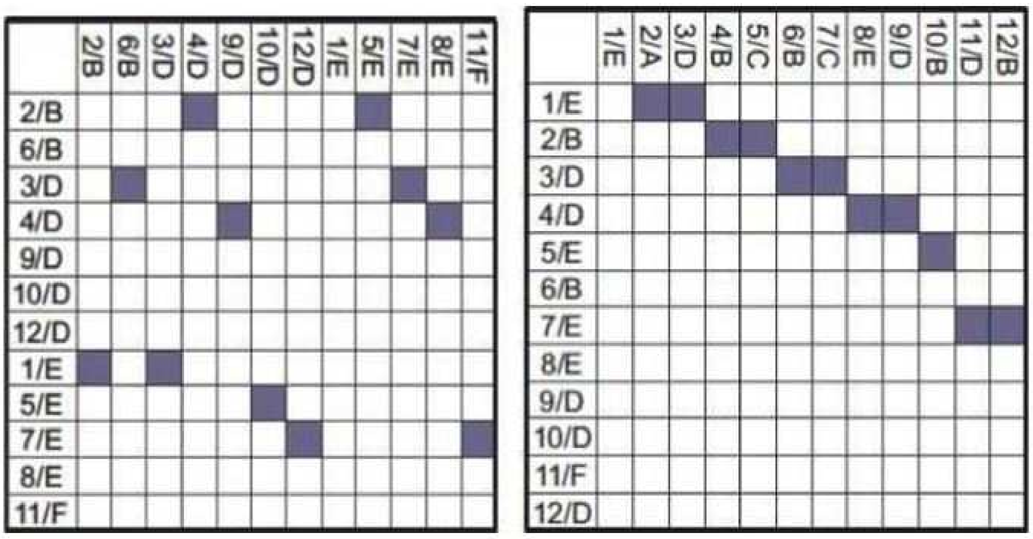
We are going to consider following criteria for different visualization techniques (for multiple hierarchies) in order to compare them with our proposed technique.

Single representation: We are supposed to display relationships between two distinct structures: object tree and taxonomy. Therefore it is desirable to have a single representation for the provided multiple hierarchies.

Crossing: There is no restriction on the size of the data.(object tree) So edge crossing in the drawing must be reduced in order to see the connections between objects more clearly.

Continuity: This parameter reflects the difficulty level in the following the lines which are representing edges in the underlying visualization technique.

Compactness: Visualization technique should be as compact as possible.



	2/B	6/B	3/D	4/D	9/D	10/D	12/D	1/E	5/E	7/E	8/E	11/F
2/B				■					■			
6/B												
3/D		■								■		
4/D					■						■	
9/D												
10/D												
12/D												
1/E	■		■									
5/E						■						
7/E							■					■
8/E												
11/F												

	1/E	2/A	3/D	4/B	5/C	6/B	7/C	8/E	9/D	10/B	11/D	12/B
1/E		■	■									
2/B				■	■							
3/D						■	■					
4/D								■	■			
5/E										■		
6/B												
7/E											■	■
8/E												
9/D												
10/D												
11/F												
12/D												

Figure 11: Adjacency matrices

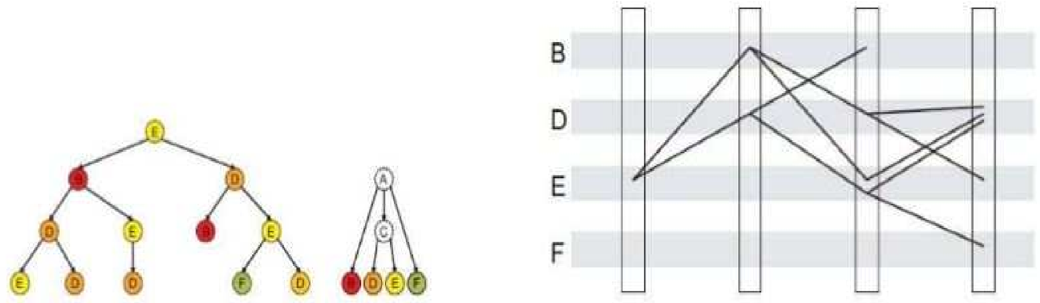
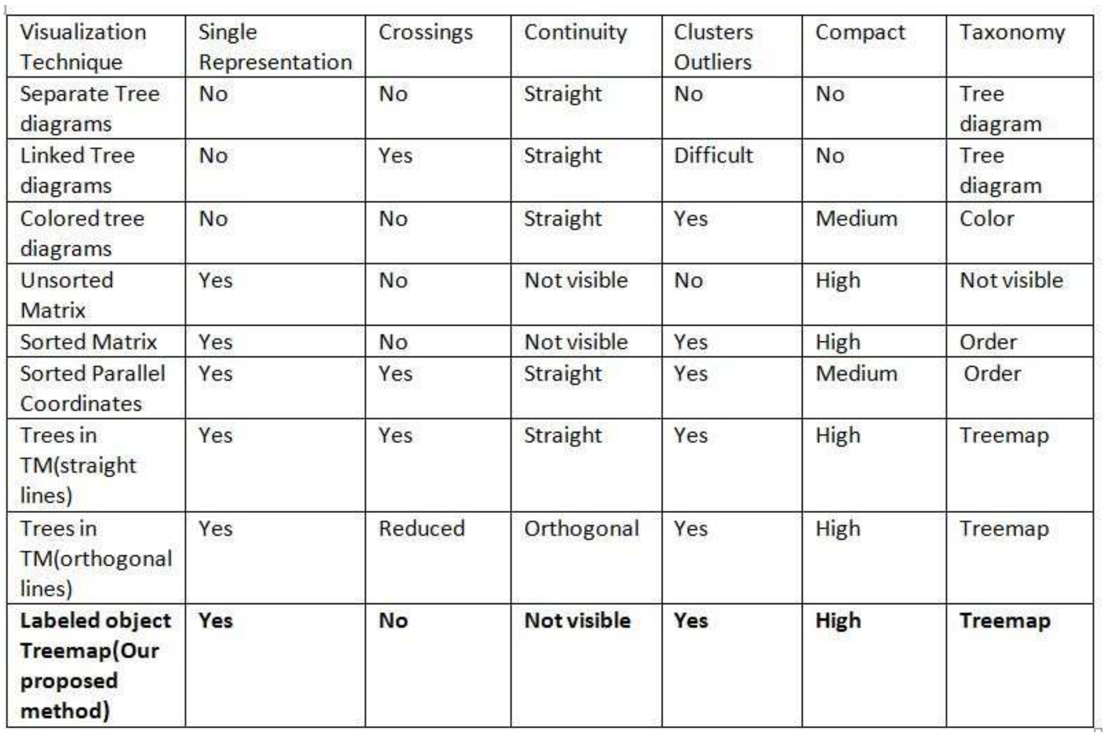


Figure 12: Colored tree diagrams and sorted parallel coordinates

In order to compare different techniques, representations of the same given trees (shown in Figure 2) are displayed using various methods. All the figures presented in this section are taken from [2].



Visualization Technique	Single Representation	Crossings	Continuity	Clusters Outliers	Compact	Taxonomy
Separate Tree diagrams	No	No	Straight	No	No	Tree diagram
Linked Tree diagrams	No	Yes	Straight	Difficult	No	Tree diagram
Colored tree diagrams	No	No	Straight	Yes	Medium	Color
Unsorted Matrix	Yes	No	Not visible	No	High	Not visible
Sorted Matrix	Yes	No	Not visible	Yes	High	Order
Sorted Parallel Coordinates	Yes	Yes	Straight	Yes	Medium	Order
Trees in TM(straight lines)	Yes	Yes	Straight	Yes	High	Treemap
Trees in TM(orthogonal lines)	Yes	Reduced	Orthogonal	Yes	High	Treemap
Labeled object Treemap(Our proposed method)	**Yes**	**No**	**Not visible**	**Yes**	**High**	**Treemap**

Figure 13: Comparison with existing techniques

6. Conclusion and future work

Our proposed data visualization technique shows all the data (multiple hierarchies) without any information loss. Using our technique, it is possible to display large data sets coherently. The method encourages the human eye to compare different objects because each object is represented by a unique label. It is possible to identify objects which obey certain rules with the use of the underlying treemap structure. As shown in Figure 13, our proposed method offers all the good characteristics of existing methods including single representation, identification of clusters, compactness as well as visible taxonomy using treemap.

In future, we plan to design a more efficient interactive version of the proposed technique in which we would incorporate more features. In the interactive version, we want to highlight edges which are present within certain rectangle regions of the treemap whenever the user clicks on it. Using this feature, users can understand whether those nodes which belong to the same object are related to each other or not. To improve efficiency in computation, we plan to assign an extra label to each node which represents the level number. Edges are present only between adjacent layers so this idea may reduce the search space. Within one rectangle region of treeemap, we will put disjoint labels i.e. adjacent vertices within the alpha-neighborhood of them. This will help users to identify connections rapidly. Whenever the user clicks on a particular node, we will highlight all the neighbors of the node by doing real time computation on labels and we will also highlight the corresponding edges so that users can find out the neighbors. To provide more information related to adjacency among the nodes, we will align adjacent nodes across different rectangles of treemaps in the same horizontal/vertical line. In order to provide more information about the underlying hierarchical structure, we may plan to use other variants of treemap: ordered/cushion treemap.

REFERENCES

1. P.Erdos and A.W.Goodman and L.Pósa, The representation of a graph by set intersections, *Canad. J. Math.*, 18 (1966) 106-112.
2. M.Burch and S.Diehl, Trees in a treemap: Visualizing multiple hierarchies, *Electronic Imaging 2006*. International Society for Optics and Photonics, 2006.
3. J.Mahipal and K.Shah, Tree-map: a visualization tool for large data, *GSB'15* (2015) 9.
4. S.Ben and M.Wattenberg, Ordered treemap layouts, *Proceedings of the IEEE Symposium on Information Visualization 2001*, Vol. 73078. 2001.
5. V.Wijk, J.Jarke and Huub Van de Wetering, Cushion treemaps: Visualization of hierarchical information, *Information Visualization, 1999.(Info Vis' 99) Proceedings. 1999 IEEE Symposium*, 1999.
6. B.Mark, K.Huizing and J.J.Van Wijk, Squarified treemaps, *Data Visualization 2000*. Springer Vienna, 2000. 33-42.
7. B.Benjamin, B.Shneiderman and M.Wattenberg, Ordered and quantum treemaps: Making effective use of 2D space to display hierarchies, *ACM Transactions on Graphics*, 21(4) (2002) 833-854.
8. T.Ying and H.-W.Shen, Visualizing changes of hierarchical data using treemaps, *IEEE Transactions on Visualization and Computer Graphics,* 13(6) (2007) 1286-1293.
9. G.Martin and J.Kennedy, A survey of multiple tree visualization, *Information Visualization,* 9(4) (2010) 235-252.

10. F.Bastian and T.Muecke, Unification and evaluation of graph drawing algorithms for different application domains, *Tenth International Conference on Information Visualisation*, IEEE, 2006.
11. F.W.M.Mank, Cristal View-The visualization of a Cristal, *Master's Thesis, Mathematics and Computer Science, Technische Universiteit Eindhoven, Eindhoven, Netherlands* (2005) 115.
12. A.G.García, Paulo, et al., A usability study of taxonomy visualisation user interfaces in digital repositories, *Online Information Review*, 38 (2) (2014) 284-304.
13. L.Bartram, A.Uhl and T.Calvert, Navigating complex information with the ZTree, *Graphics Interface*, 2000.
14. G.W.Furnas and J.Zacks, Multitrees: enriching and reusing hierarchical structure, *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1994.
15. G.Robertson, et al., Animated visualization of multiple intersecting hierarchies, *Information Visualization,* 1(1) (2002) 50-65.
16. M.Pal, Intersection graphs: an introduction, *Annals of Pure and Applied Mathematics,* 4(1) (2013) 43-91.
17. S.K.Vaidya and N.B.Vyas, Antimagic labeling of some path and cycle related graphs, *Annals of Pure and Applied Mathematics*, 3(2) (2013) 119-128.
18. S.K.Vaidya and N.H.Shah, Further results on divisor cordial labeling, *Annals of Pure and Applied Mathematics*, 4(2) (2013) 150-159.