Large-Scale Data Visualization with Two-Variate Level-of-Detail Control

1 Introduction

Space-filling approach, such as a series of TreeMaps algorithms, is one of the popular approaches in hierarchical data visualization. The original TreeMaps [1] represents the hierarchy as nested belt chart (so called slice-and-dice), and then it has been improved by squarifying or ordering the subregions representing the nodes of a tree. These algorithms represent quantitative properties of nodes as areas of subregions; however, it is often required that leaf-nodes of a tree is displayed as equally-sized icons. Quantum TreeMaps [2] is a variation of TreeMaps that represents leaf-nodes as equally-sized icons or thumbnails while it attempts to quantize the widths and heights of rectangular subregions so that the icons are well-packed inside them. Against these series of TreeMaps divides display spaces by top-down algorithms, another space-filling technique [3] represents trees as nested rectangular regions and equally-sized icons applying a bottom-up algorithm.

We are applying space-filling hierarchical data visualization techniques to the large-scale table data, as shown in Figure 2, which is supposed as following.

- The data consists of a set of records $R = \{r_1, r_2, .., r_n\}$, where n is the number of records.
- A record r_i in the data has m variables, described as $r_i = \{r_{i1}, r_{i2}, .., r_{im}\}$.

Suppose that the data may contain thousands or even tens of thousands of records, while a record has tens or even hundreds of variables. Our study divides the records according to the selected i-th variable to construct a tree, and displays the tree by coloring the nodes according to j-th ($i \neq j$) variable. Though we would like to basically apply a visualization technique which represents leaf-nodes as equally-sized icons [2] [3], it may be nonsense when the number of leaf-nodes is enormous and therefore they are too small to comprehensively display. Or, it may be often difficult to understand the distribution of branches if there are too many under a parent branch node.

This poster presents a combination of two level-of-detail (LoD) control techniques which adaptively simplifies the visualization of hierarchical data. We call the techniques as two-variate LoD control, because the two techniques realizes the LoD control according to the distribution of the i-th and j-th variables.

The first technique merges branch nodes under a parent branch node, as shown in Figure 2(c), to adaptively reduce the number of branch-nodes to be drawn. Current our implementation prepares two criteria to merge the branches. The first criterion is based on the number of children nodes: the implementation aggressively merges the branch nodes that have smaller number of children nodes. The second criterion is based on distribution of the specified variables: the implementation constructs histogram of the variables of the branch-nodes, and merges them if the distribution of histogram is similar.

The second technique replaces a set of icons corresponding to leaf-nodes under a branch-node as a set of sliced rectangles, as original TreeMaps renders [1], to display overall information. It constructs a histogram of the values of j-th variables, and then slices the rectangular region corresponding to the branch node according to the histogram.

2 Technical Detail

2.1 Hierarchy Construction and Visualization

Our implementation firstly divides the records R according to the i-th variable r_{1i} to r_{ni}, to construct hierarchical data. Here, it is possible to recursively divide the leaf-nodes to construct deep hierarchy, but current our implementation does not support it. It then assigns colors calculated from j-th variable r_{1j} to r_{nj} to the leaf-nodes of the hierarchical data.

Our implementation then visualizes the hierarchical data by applying a space-filling technique which represents leaf-nodes as equally-sized icons. Currently we apply TreeMaps-like space-filling technique [3], but Quantum TreeMaps [2] can be also applied.

Our implementation activates the LoD control technique with users' operations, such as zooming operations, so that it displays detailed information while zooming in, and overall information while zooming out.

2.2 Level-of-Detail Control of Branch-Nodes

We observed the visualization results of the space-filling technique, and determined that we can effectively simplify the visualization based on the following two criteria.

2.2.1 Based on numbers of children nodes

Though small groups of information may be even important in some cases, they are ignorable in other many cases. Our implementation merges branch-nodes which have smaller number of children leaf-nodes. Current our implementation merges the two branch-nodes B_s and B_t if they satisfy the following criterion:

$$C_N(N_s + N_t) < N_{max} \tag{1}$$

Here, C_N is a positive constant value to control the LoD, N_s and N_t are numbers of children leaf-nodes of B_s and B_t, and N_{max} is the maximum number of children leaf-nodes.

Our implementation firstly sorts the branch-nodes in a specific depth of the hierarchy in the ascending order of the numbers of their children leaf-nodes. It then picks up the first two branch-nodes in the sorted list, and merges them if they satisfy the above criterion. It also inserts the merged branch-node at the adequate position of the sorted list. This process is repeated until no pairs of branch-nodes can be merged.

2.2.2 Based on similarity of distributions of values

There may be many groups whose distributions of values look very similar. Our current implementation merges such branch-nodes whose distributions of values are similar. It firstly determines the minimum and maximum values of the j-th variable which are used to calculate colors of leaf-nodes in the whole data, and the divides the range between minimum and maximum value into multiple spans $S = \{s_1, s_2, .., s_{N_s}\}$, where N_s is the number of spans. It then counts the number of children leaf-nodes c_{sk}, whose values of j-th variable belong to k-th span s_k for the branch-node B_s. Consequently, it constructs a histogram of the leaf-nodes consisting of N_s ranks. Current our implementation then calculates the similarity

of statistics of j-th variable between two branch-node B_s and B_t by the following equation:

$$Sim_{st} = \frac{\sum_{k=1}^{N_s} c_{sk} c_{tk}}{\sqrt{\sum_{k=1}^{N_s} (c_{sk})^2} \sqrt{\sum_{k=1}^{N_s} (c_{tk})^2}} \quad (2)$$

Our implementation firstly calculates Sim_{st} for every possible pairs of branch-nodes under the specified depth of the hierarchy. It then sorts the branch-nodes in the descending order of Sim_{st}. It then picks up the first two branch-nodes in the sorted list, and merges them if the Sim_{st} is larger than a predefined value. It also updates the sorted list by calculating Sim_{st} between the merged branch-node and others. This process is repeated until no pairs of branch-nodes can be merged.

2.3 Level-of-Detail Control of Leaf-Nodes

After merging the branch-node, our implementation switches representation of leaf-nodes. When a user would like to simplify the visualization result, our implementation represents the branch-node as a set of slices of rectangular regions, as the original TreeMaps does [1], instead of rendering the set of icons.

3 Result

Figure 1 shows an example of the visualization result by using the presented technique. Figure 1(Upper) is a result of space-filling visualization of hierarchical data containing small groups in the red rectangular regions. Figure 1(Center) is a result after applying the LoD control of branch-nodes. The small groups disappeared by the merge process, and consequently the visualization result is effectively simplified. Figure 1(Lower) is a result after applying the LoD control of leaf-nodes. Here, pink rectangular regions in Figure 1(Center)(Right) shows that there are characteristic groups whose distributions of the j-th variable are much different from those of other groups. We think that these kinds of features can be easily discovered after the LoD control of leaf-nodes. Also, we think these kinds of branch-nodes should not be merged by the LoD control of branch-nodes.

We can integrate the two criteria for merging branch-nodes described in Section 2.2, though current our implementation alternatively applies them. We would like to integrate them and test it as a future work. Other future issues include experiments with various data, numerical and subjective evaluations, development of more sophisticated user interface to effectively apply the LoD control, and development and experiments of LoD control with three or more variables.

This work has been partially supported by Japan Society of the Promotion of Science under Grant-in-Aid fir Scientific Research (B) 13900008.

References

[1] B. Johnson, B. Shneiderman, Tree-Maps: A Space Filling Approach to the Visualization of Hierarchical Information Space, IEEE Visualization '91, pp. 275-282 (1991).

[2] B. Bederson, B. Schneiderman, Ordered and Quantum Treemaps: Making Effective Use of 2D Space to Display Hierarchies, ACM Transactions on Graphics, Vol. 21, No. 4, pp. 833-854 (2002).

[3] T. Itoh, Y. Yamaguchi, Y. Ikehata, Y. Kajinaga, Hierarchical Data Visualization Using a Fast Rectangle-Packing Algorithm, IEEE Transactions on Visualization and Computer Graphics, Vol. 10, No. 3, pp. 302-313 (2004).

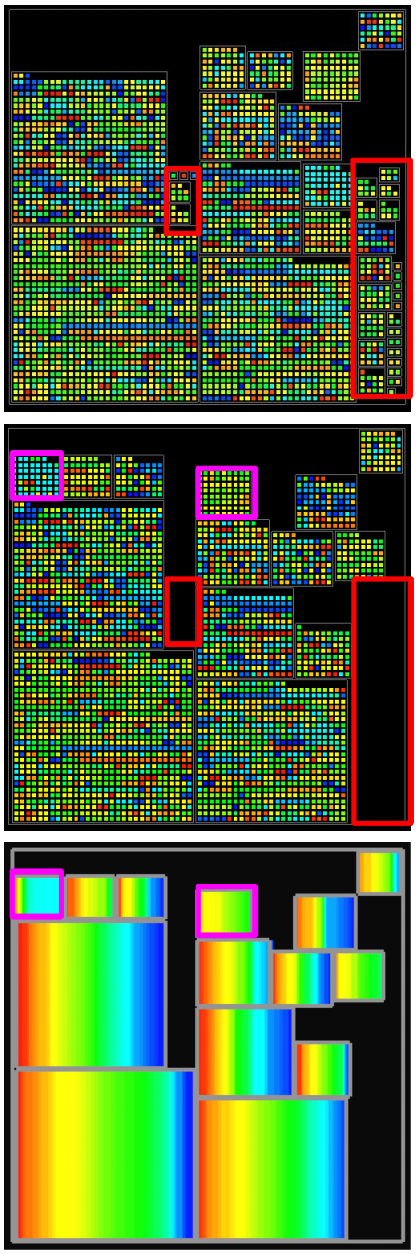

Figure 1: Example. (Upper)Before applying LoD control. (Center) After applying LoD control of branch-nodes. (Lower) After applying LoD control of leaf-nodes.

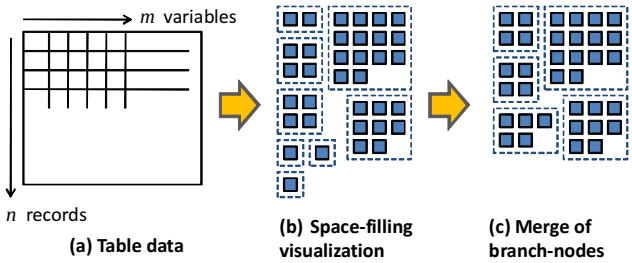

Figure 2: Processing flow.