Generalized Automatic Color Selection for Visualization

Amy Ciavolino and Tim Burke

Abstract—Creating a perceptually distinct coloring for visualizing large data sets with one or more related properties can be difficult, even for a domain expert. To aid in the color selection process, we present a method for selecting an optimized coloring for these data sets by applying a force-directed algorithm. The method we have implemented selects hue values using a constraint graph derived from relationships within the data. This optimal set of hues can then be converted to colors using any color space and used to color the data points in a visualization. This method is demonstrated through a geographical domain problem involving school districts, specifically depicting elementary to middle to high school feeder patterns. Further, the method is applied to two different sets of constraints for creating visualizations of directory structures: file size relationship constraints and dominant file type constraints.

Index Terms—Human Factors, User Interfaces, Optimization, color selection, map visualization, optimization

1 Introduction

This project works towards a domain independent search-based color selection technique for visualizing data with discrete related properties. The goal of the coloring technique described is to generate colors for data points in such a way that the similarities and dissimilarities between the data points are apparent. Creating a coloring by hand that accomplishes this goal can be difficult, particularly for large data sets which is why automatic selection can be useful. Although the relationships in the data are domain specific, we present a process and examples of translating these relationships to a constraint graph with similarity and dissimilarity constraints as edges and data points as nodes.

For data sets which have multiple discrete or categorical properties, there are differing approaches to visualizing the relationships between the data. A common approach is to display patterns (stripes, textures, other region partitioning methods) but others use colors to represent properties in a visualization which are either blended or each property is mapped to a different color component (e.g., hue, saturation, and value). In this work, we we focus on hue to visualize the discrete properties in the data, that way saturation and brightness can be reserved for visualizing other parts of the data, including continuous data.

The color selection produced by the algorithm will be dependent on the constraints derived from the data being visualized. As constraints are dependent on the type of data being visualized, and the precedence of those constraints is domain-dependent, some thought must go into selection and weight of the constraints when applying this coloring technique. As discussed in section 4, the school district problem has very different categorical constraints as compared to the directory structure or file type domains.

Our approach to generating the color sets involves several steps. For each specific domain example, a set of similarity and dissimilarity relationships must be identified between data points, which is then used to create a constraints graph of those relationships (edges) between the data points (nodes). The edges within the constraint graph can also be assigned a weight. Similarity relationships between data points have a force to pull the node colors closer together, and dissimilarity relationships apply a force to push the node colors apart. Once the constraint graph has been constructed, a force-directed algorithm optimizes the forces in the graph to find an optimal distance between node colors that satisfies the push and pull of the constraint forces between nodes. Once the colors have been selected, the original data set can be visualized using the generated colors, in any type of visualization. The result is a visualization of the original data, where colors represent the relationships between the discrete or categorical traits of the data points.

- *Amy Ciavolino is a student at the University of Maryland, Baltimore County, E-mail: aci1@umbc.edu.*
- *Tim Burke is a student at the University of Maryland, Baltimore County, E-mail: tburke2@umbc.edu.*

Manuscript received 31 March 2011; accepted 1 August 2011; posted online 23 October 2011; mailed on 14 October 2011.
For information on obtaining reprints of this article, please send email to: tvcg@computer.org.

2 Related Work

Prior research in this area explores how coloring can be used in different visualization approaches. Much of the available research has focused on using ranges of colors to represent ranges of continuous data. That approach is not always effective as most individuals do not have an intuitive sense of ordering for hues, as such using a rainbow color scale is ineffective for displaying scalar data [2], though our coloring algorithms approach of using color sets is not bound by this constraint as we are exploring discrete traits. Other areas of research have explored ways of using different colors to present multiple data points [7] - the school district coloring domain provides an example of how this approach is useful.

Of interest to our approach, other work has focused on selecting visually distinct colors by using distance between colors in a color space. Physiological factors affect human color perception, and using Euclidean distance to balance colors in a perceptually uniform color space such as the L^*, u^*, v^* color space allows for the selection of effective color sets [3]. None of this work addresses the issue of automatic color set selection and optimization.

3 Algorithm

The goal of the color set algorithm is to produce a mapping from data points to colors. These data points can be anything

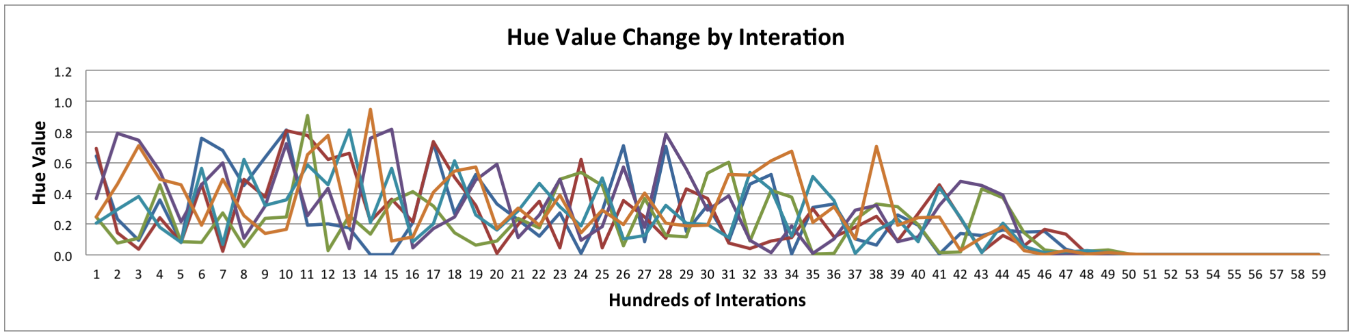

Fig. 1. An example run of the color change algorithm depicting the rate of change in hue values for the individual nodes after every 100 iterations. As the series progresses, the change in hue values falls to zero as each node as the forces between nodes are balanced.

from file types, to map regions, to folders. The colors are selected based the relationships within the data. In section 4 we discuss the methodology for determining these relationships. Once the relationships are been determined, they are represented as a constraint graph where nodes represent data points to be colored in a visualization of the full data set and edges represent the identified relationships. This constraint graph and is used to generate a color for each node by applying a force-directed layout method to balance the constraints in the graph.

Force-directed layout algorithms (often referred to as spring layout algorithms) simulate forces of repulsion and attraction acting on objects. These algorithms are often used in graph drawing where the goal is to find positions for the nodes that locally minimize the forces on the nodes. By representing the node positions as a hue value and using the edges in our constraint graph, we can use these methods to produce optimized color assignments.

3.1 Constraint Graph

Each data point to be colored in the visualization of the full data set translates to a node in the constrain graph, and the identified relationships translate to edges. There are two types of edges to represent two types of relationships within the data: similarity relationships, and dissimilarity relationships. Similarity edges supply a forces that pull hue values closer together, resulting in final colors that are visually related. Dissimilarity edges, conversely, supply a forces that pushes hue values apart, yielding final colors that are visually distinct.

3.2 Color Circle

The position of each node is represented as a hue value in the range (0,1] and is initialized randomly in that range. This method of representing node position effectively creates a one-dimensional, wrapped space in which to execute the force-directed algorithm. Once the hue values have been generated, they can be used to define colors in any color space by using a constant lightness and saturation.

In our implementation, we convert the hues to colors using the HLS color space but a better choice may be to use the L^*, u^*, v^* color space due to its perceptual uniformity. Specifically, the L^*, u^*, v^* color space has the characteristic that similar distances in the space correspond to similar perceptual differences between colors.

3.3 Color Distance

Because the range of all possible hues on the circle is one-dimensional and wrapped, we define a distance metric ρ to measure the differences between the hues. This distance metric is defined for any two colors on the circle, c_1 and $c_2 \in (0,1]$, as:

$$\rho\,(c_1, c_2) = \begin{cases} 2|c_1 - c_2| & : |c_1 - c_2| \leq \frac{1}{2} \\ 1 - 2|c_1 - c_2| & : |c_1 - c_2| > \frac{1}{2} \end{cases} \tag{1}$$

This definition yields distances in the range $(0,1]$, where a distance of 1 corresponds to colors on the opposite sides of the circle ($|c_1 - c_2| = \frac{1}{2}$).

3.4 Forces

There are two types of forces that act on the nodes in the constraint graph: edge forces (similarity, $\vec{f}_{sim}$ and dissimilarity, $\vec{f}_{dis}$ edges) and general color separation forces, $\vec{f}_{sep}$. Each force and edge type is described in more detail below. Additionally, each forces has a corresponding weight associated with it (w_{sim}, w_{dis}, and w_{sep}) that reflects the importance of that force and is used to scale the force corresponding force value.

The force-directed layout is updated iteratively, either until the system stabilizes (i.e., the velocity of all color assignments becomes smaller than a specified constant v_{min}) or until a maximum number of iterations has elapsed. In figure 2, we show the change in hue value after each 100 interactions. Notice that as the number of iterations increase, the change in hue values between iterations falls, and eventually decreases to zero. During each iteration, we compute all of the forces acting upon each node, compute their weighted sum, and apply that force to each node which changes its hue value.

When each force is computed, the proper direction of the force needs to be determined because of the wrapped nature of the color space. Given two hue values (c_1, c_2), the proper direction for a force that pushes the nodes apart ($\vec{f}_{dis}$ or $\vec{f}_{sep}$) can be determined as follows:

$$\sigma(c_1, c_2) = \begin{cases} -1 & : c_1 + (1 - c_2) \leq \frac{1}{2} \\ 1 & : c_1 + (1 - c_2) > \frac{1}{2} \end{cases} \tag{2}$$

For forces that pull nodes together ($\vec{f}_{sim}$), this value may simply be negated, or the input order of the nodes may be reversed.

Similarity Edges Similarity edges represent attractive forces, $\vec{f}_{sim}$, that act on nodes to ensure that the colors of those nodes are visually similar. Each similarity edge may also have a weight associated with it representing how closely those nodes are related. Those weights are in addition to the overall weight of each force type. For instance, if we have three nodes representing three file types, $.doc$, $.docx$, and $.pdf$, a $.doc$ file is more closely related to $.docx$ files than a $.pdf$ file. Therefore an edge between $.doc$ and $.docx$ would have a higher weight than an edge between $.doc$ and $.pdf$. The value of the similarity force $\vec{f}_{sim}(c_i)$ for the ith color is defined as:

$$\vec{f}_{sim}(c_i) = \sum_{c_j} \rho(c_i, c_j) w(c_i, c_j) \tag{3}$$

Where $w(c_i, c_j)$ is the weight of the edge between c_i and c_j in the constraint graph. The value of the force increases linearly with the distance between two colors.

Dissimilarity Edges Dissimilarity edges represent repulsive forces that act on nodes to ensure that the colors of nodes are visually distinct. As with similarity edges, each edge may also have a weight associated with it. For example, if coloring a map, the user may want adjacent regions to have easily distinguishable colors so the borders of the regions are clear. To produces that result, adjacent regions could be connected by dissimilarity edges. The value of this repulsive disssimilarity force $\vec{f}_{dis}(c_i)$ for the i color assignment is defined as:

$$\vec{f}_{dis}(c_i) = \sum_{c_j} (1 - \rho(c_i, c_j))^2 w(c_i, c_j) \tag{4}$$

Again, where $w(c_i, c_j)$ is the weight of the edge between c_i and c_j in the constraint graph. For this force, the value decreases with the square of the distance between the two colors.

Separation Forces The default separation forces are not defined within the constraint graph, but can be thought of as edges that connect each node to every other node in the graph and push the nodes apart (much like the dissimilarity edges). These forces are meant to enforce a minimum distance between nodes so that each node color is visually distinguishable. Additionally, this force keeps colors that should be similar from being exactly the same color. In the some domains, it may be desirable to be able to identify each data point in order to make a legend which this force enables. By definition, these forces are explicitly short-distance repulsive forces. The force is applied only if the distance $\rho(c_1, c_2)$ is lower than a predefined boundary value, b_{sepmax}.The separation force for c_i is define as:

$$\vec{f}_{sep}(c_i) = \sum_{c \wedge \rho(c_i, c_j) < b_{sepmax}} 1 - \frac{\rho(c_i, c_j)}{b_{sepmax}} \tag{5}$$

This force decreases linearly with the distance between two colors becoming 0 when $\rho(c_1, c_2) = b_{sepmax}$.

4 Application Domains

We present three example domain applications to demonstrate the color selection algorithm. For each example, we describe the process used to determine the relationships in the data, which translate to the constraint edges graph used by by the color selection algorithm. This process involves identifying the data points which will be colored in a visualization, the ways in which the data point are related, and the types of these relationships. As discussed in section 3, The relationships can have two types, similarity relationships and dissimilarity relationships.

4.1 School Districts

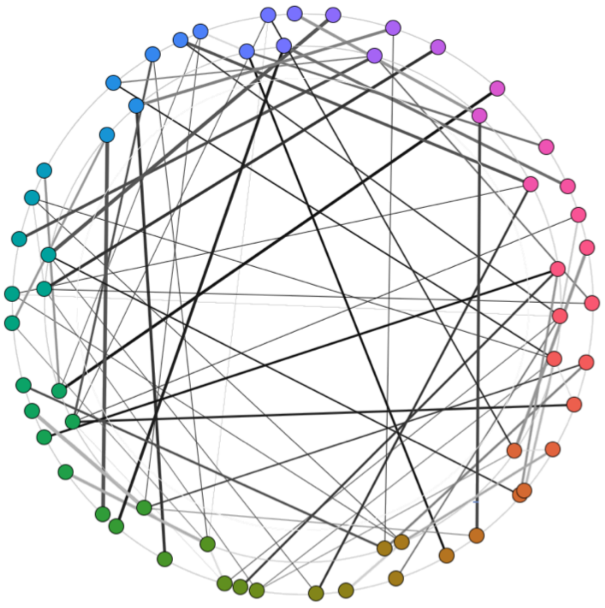

Fig. 2. The constraint graph for the school districts domain problem depicting the edge weights between the school data points. The outer ring represents elementary schools and the inner ring shows middle schools.

Our first domain is a geographical visualization problem involving how neighborhood regions within a suburban Maryland county map to school districts, and how to visually represent the three level feeder patterns of elementary, middle, and high school districts. The result of the visualization is a county map with regions colored by school district. The utility of this example comes in working with a local school board to visualize proposed adjustments to school districts and how it impacts the K-12 school feeder patterns.

Our first consideration is what data points will be colored in our map visualization. The goal is to be able to see school districts and how these districts are related, so the school districtsx become the nodes in our constraint graph. The next consideration is how these nodes are related. In this domain, there are two clear relationships which we explore, feeder patterns and map adjacency, but other relationships such as standardized testing scores or school size could be considered as constraints. As school assignments are discrete categorical values, feeder patterns work well for demonstrating this coloring technique.

Now that we have identified the relationships, we must categorize them as similarity or dissimilarity constraints. In the case of the feeder patterns, we would like feeder schools to be a similar color to the higher level school they feed into, so our feeder relationship is a similarity constraint. Because of the nature of map visualizations, it is easier to see region boundaries when adjacent regions have distinct colors. Therefore, the colors of school districts which are adjacent should be distinct or dissimilar so the map adjacency relationships is a dissimilarity constraint. Note that only school at the same level (elementary, middle or high school) can be adjacent because districts at different levels over lap.

The final constraint graph for this visualization is comprised of school districts as nodes, feeder patterns as similarity constraints and districts adjacency as dissimilarity constraints. In addition to this basic constraint structure, the feeder pattern edges were each assigned a weight proportional to the number of students moving from one school to the other. This resulted in the lower level school with the most students moving to a higher level school having a color closest to the higher level school.

4.2 Directory Structure

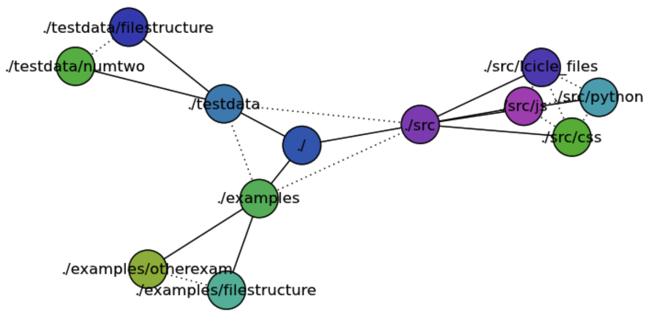

Fig. 3. The constraint graph for the directory structure domain problem. Solid lines between nodes depict parent-child relationships and dashed lines represent neighbor relationships (nodes sharing the same parent node).

The second is a domain problem that involves visualizing directory structures. This depicts similarity relationships of size between parent and child directories along a directory tree hierarchy. For this visualization, the items that will be colored are folders in the directory structure so folders will be the nodes in our constraint graph. The most obvious relationship between folders is a parent-child relationship so we chose to have a similarity constraint between parent folders and their children. This constraint alone is not that interesting, but there is also weight to these constraints based on the percent the childs size is of the parents size. This creates a coloring where it is easy to see which sub-folder contains most of the content of the parent folder.

The second constraint in the graph for this example is similar to adjacent school districts in the previous domain. We would like nodes that are adjacent in our visualization to be visually distinct and in the sunburst visualization we chose (figure 3), folders at the same depth are displayed in a ring. If folders in the same ring were assigned very similar colors, is would be difficult to see the boundaries between their display areas. Therefore, folders at the same depth are assigned dissimilarity constraints.

4.3 File Type

The last example is different from our previous examples in that we will use the algorithm generate a legend for a set of

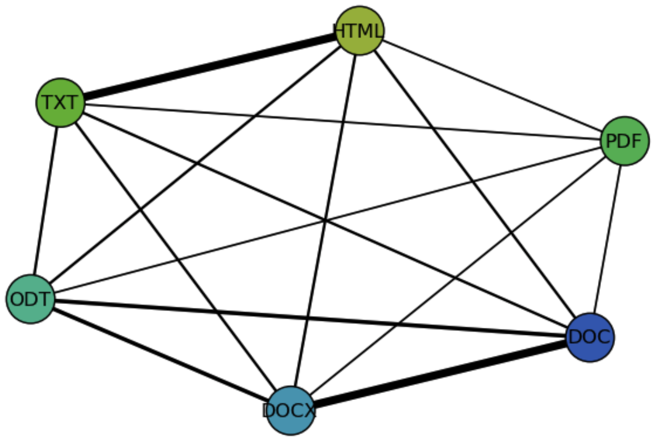

Fig. 5. The constraint graph generated from the file types depicted in figure 4.

values instead of generating colors to be used directly in a visualization. This was accomplished by modeling the relationships between categories in the data rather than the relationships between the data points themselves. The desired result is to generate a coloring for the legend where similar categories are assigned similar colors. For a small number of categories, this could be accomplished by hand. For example, if you were to assign a color to elementary, middle and high schools it would be easy to chose yellow for elementary, orange for middle, and red for high schools, but as the set of categories expands and has more complex relationships, it becomes increasingly difficult to create the color set by hand. To demonstrate this method, we reused the file system domain but added a file type property to each folder which was the most common file type in the given folder. This file type property is our category which can be any file type, but we used a subset of common file types for our legend. Even our subset of file types was to large and complex to color well by hand so we represented the relationships between the file types as a tree which we then converted to a constraint graph.

Figure 4 is a small portion of the relationship tree showing the relationships between document types. Notice that DOC and DOCX are both leaves at the same depth because they are very similar file types and ODT (Open Document Format) is equally as similar to either of those types but not as similar as they are to each other so it is placed as a leaf node one level up from DOC and DOCX. Also notices that PDF is a document type but is not an editable type like the other types so PDF is places one level above the other types. It is useful to think of the tree as a nested list where each list contains types that are similar and the types within that group that are more similar are then put into a sub-list. The tree in figure 4 can be represented by the following nested list structure:

$$(((DOCX, DOC), ODT), (TXT, HTML), PDF)$$

Once this tree has been constructed, a constraint graph is generated by connecting each type, or node, to every other node with a similarity constraint. As with the file size graph, the important part of these constraints is the weight. The weights are

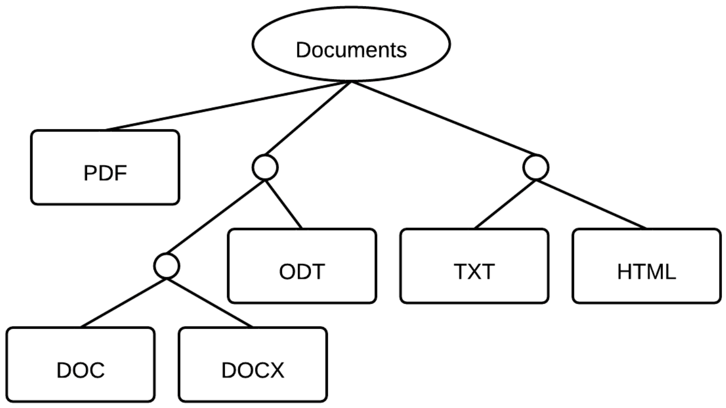

Fig. 4. An example file-type relationship tree displaying the relationship between similar and dissimilar file types.

determined using the tree structure by assign a wight of 1 between nodes at the bottom of a branch. For instance, DOC and DOCX are at the same depth, and there are no sub-trees at the same level, so the weight of the edge between these types is 1. Likewise, the edge between TXT and HTML has a weight of 1. Moving up the tree from these deepest relationships, the edges at shallower levels have the weight $1/(depth+1)$ and are connected to every other node at the same level, as well as nodes in any sub-trees of the branch. By this method, ODT has an edge to both DOC and DOCX, and these edges both have a wight of $\frac{1}{2}$. Following from there, PDF has an edge to all other types shown with weight $\frac{1}{3}$. If there are sub-trees at a given level that have different depths, the maximum depth of the sub-trees is used to calculate the weight.

Figure 5 shows the final constraint graph derived from the file type relationship tree. The nodes are colored with the colors generated by the algorithm and the thickness of the edges shows the derived weight. Note that the edges between TXT and HTML, and DOC and DOCX have the highest weight. This coloring can now be used to color any data which has a file type property, which we discuss an example of below. Although we use file types for this example, a legend for any categorical property could be created using this method.

5 Results

Once the constraints for each domain are generated, it is possible to generate visualizations of the original data using the generated colors for the data points within the original data. With the carefully selected constraints for each application domain, the optimized color set generated can be applied to the data through a visualization toolkit of the users choosing. For the file system examples that follow, we use the JavaScript InfoVis toolkit to generate the visualizations, for the school district visualization a custom Java application was created to display and color the map areas. With the completed visualizations, the viewer can effectively analysis the relationships within the original data because of the optimized color set generated by the coloring algorithm.

5.1 School Districts

In Figure 6, two levels of the school district feeder pattern data is visualized. The map on the top right depicts the school districts for middle schools, where the map on the top left shows elementary school areas. Focusing on the top left area of both maps, the two elementary schools (green and purple) are adjacent and distinctly colored as compared to one another. Those two elementary school districts combined make up the whole area of their parent node, the middle school area of the map depicted in blue. Neither of the elementary schools are colored identically to the parent, as a stronger general separation force was applied when coloring this data. For this visualization, the bottom map depicts both levels of the hierarchy together using colored rings - the inner being the middle school and the outer ring being the elementary school.

The bottom visualization enables the viewer to identify certain anomalies. Because the weight of the similarity constraints is proportional to the feeder pattern, the largest of child nodes (elementary schools) feeding into a parent (middle school) will be most similarly colored to its parent. Where the multiple feeder schools have a similar number of students feeding into the parent, the similarity constraint of the children to the parent will all be similar, yielding uniformly distributed colors. In the case of an elementary school where the feeder ratio is low as compared to its neighbors, that elementary school node will have a low similarity weight as compared to the combined general separation force and the dissimilarity constraint pushing it from its neighbors, yielding a color that is very different from those around it. As schools with low feeder ratios are less desirable, especially when split between multiple regions, being able to quickly identify those regions is very useful to the county school system when considering school district boundary changes.

5.2 Directory Structure

The visualization in Figure 7, depicts a three-level directory hierarchy, and the visualizations in Figures 8 shows a deeper directory hierarchy. Examining the visualizations allows the viewer to identify patters of directories where the parent to child to directories are similar or dissimilar. This is useful in

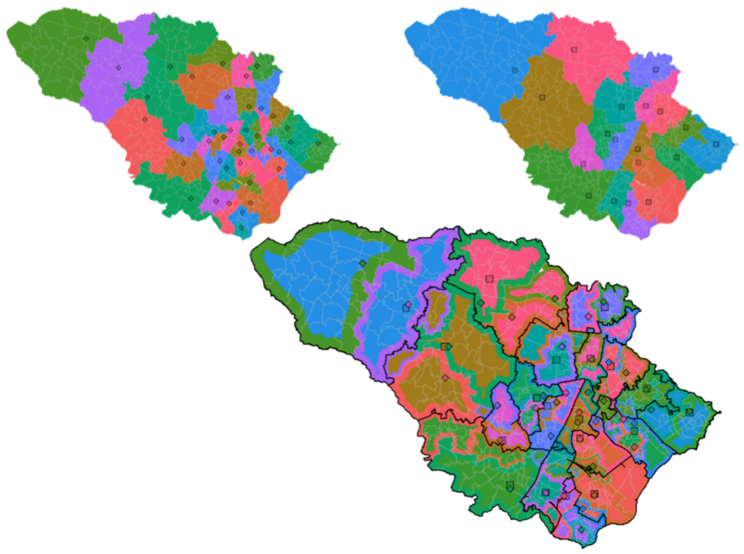

Fig. 6. Example visualization from the school district domain area. *Top:* Maps showing elementary (left) and middle (right) school assignments. *Bottom:* Map showing both elementary and middle school assignments merged on one map using a ring pattern. Within each region (neighborhoods sharing the same elementary and middle school assignment), the outer ring shows the elementary school assignment, and the inside shows the middle school assignment. For added clarity, the boundaries of the middle school assignments are displayed as black lines. The maps also show the locations of schools: diamonds indicate elementary schools and squares indicate middle schools. [4].

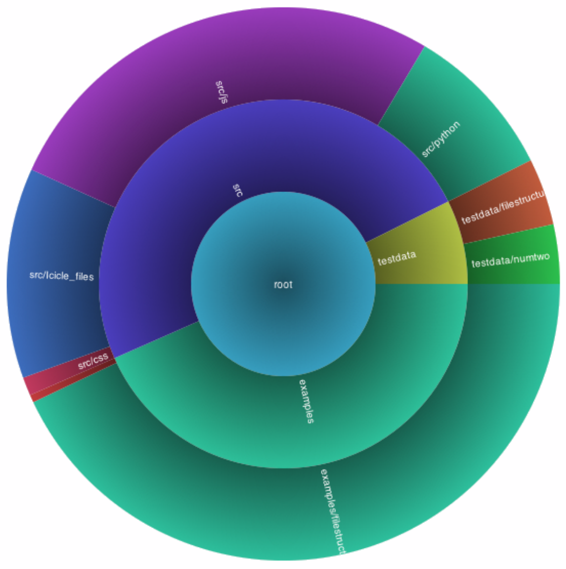

Fig. 7. This final visualization from the directory structure domain problem displays a directory hierarchy in concentric rings out from the root directory in the center. The directories have been colored with the automatic coloring algorithms, and those directories that have a similar size will have a similar coloring as the parent of that directory.

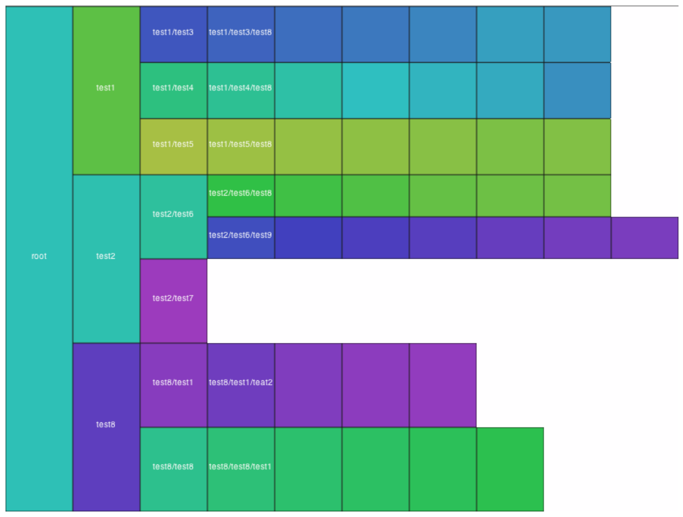

Fig. 8. This visualization depicts the Directory Structure constraint set as applied to a deeper file system hierarchy than that of Figure 7.

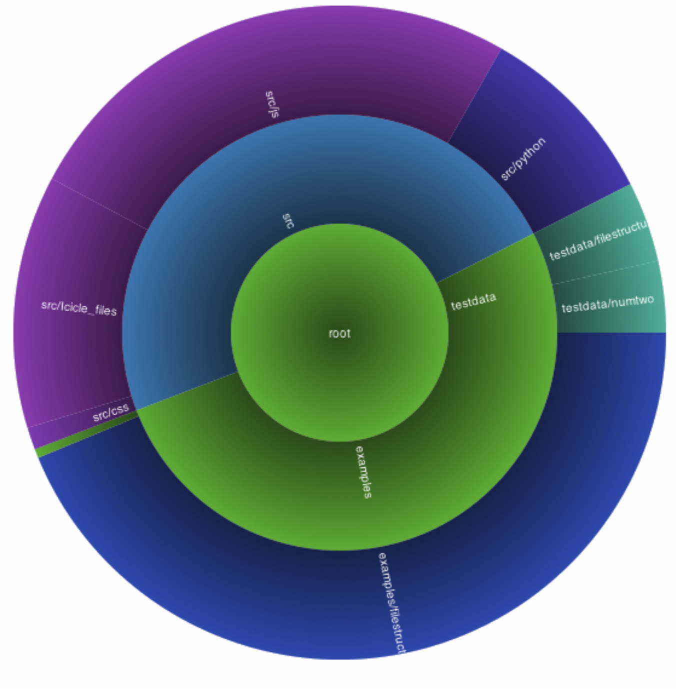

Fig. 9. This final visualization depicts the same data as displayed in Figure 7, but colored using the file type constraints rather than the directory size.

finding outliers in the hierarchy, most notably directories that are empty or that contain significantly larger files than those in the hierarchy around it.

5.3 File Type

Lastly, the image in Figure 9 depicts the same directory hierarchy as the one displayed in Figure 7, but with a different set of constraints. The colors selected by the algorithm are not based on the data being visualized, rather, the coloring algorithm was applied to the constraint graph generated from a tree of file type and sub-type associations. Each node in the file system hierarchy is bound into a single file type based on the count of the most common file type within the directory. This allows the viewer to see groups of folders which all have similar dominant file types.

In this case, as previously mentioned, being able to quickly find folders buried in a directory hierarchy that have an obviously dissimilar type as compared to those around it. Of interest here is the field of digital forensics in law enforcement where it may be necessary to search a suspects computer files for illegal materials. Searching an arbitrarily nested directory structure by hand is a slow process for a human, and having visualization tools to quickly locate areas of potential interest are useful in finding evidence.

6 Future Work / Conclusions

We have presented a method for automatically generating a constraint-based, optimized color set for use in visualizing data points with one or more related discrete characteristics. Further, we have demonstrated that the method is useful when applied to two very different application domains. Further work towards applying this technique to additional domains will continue, as

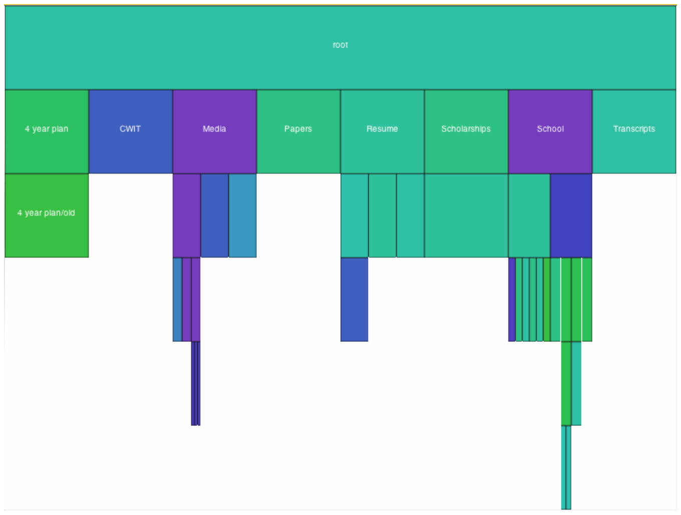

Fig. 10. A larger and deeper example data set colored using the file type domain.

well as exploring using the lightness and saturation components of the HLS color space for visualizing continuous attributes of data points colored with the algorithm presented.

Additionally, there are opportunities to optimize and simplify the algorithm by consolidating the forces. This could involve combining the similarity and dissimilarity constraints into a single relationship with positive and negative values.

Acknowledgments

This project was based heavily on the unpublished paper *Color Selection for Visualizing Multiple Related Categorical Properties*, by Blazej Bulka, Penny Rheingans, and Marie desJardins, and we would like to thank them for providing the code for the coloring algorithm. We would also like to extend thanks to the University of Maryland, Baltimore County Center for Women in Technology, who provided the funding for this research to happen.

References

[1] L. BERGMAN, B. ROGOWITZ, and L. TREINISH. Color selection for visualizing multiple related categorical properties. *In Proceedings of IEEE Visualization 1995*, pages 118 – 125, 1995.

[2] D. BORLAND and R. TAYLOR. Rainbow color map (still) considered harmful. *IEEE Computer Graphics and Applications*, pages 27, 2, 14 – 17, 2007.

[3] C. A. BREWER. Color use guidelines for data representation. *In Proceedings of the Section on Statistical Graphics.*, pages 55 – 60, 1999.

[4] B. BULKA, P. RHEINGANS, and M. desJARDINS. A rule-based tool for assisting colormap selection. *Unpublished.*, 2011.

[5] D. FARNSWORTH. The farnsworth-munsell 100 hue test for the examination of color discrimination. 1957.

[6] C. G. HEALEY. Choosing effective colours for data visualization. *In Proceedings of IEEE Visualization 1996 (San Francisco).*, pages 263 – 270, 1996.

[7] P. RHEINGANS. Task-based color scale design. *In Proceedings of Applied Image and Pattern Recognition 1999.*, pages 35 – 43, 1999.

[8] N. ROBERTSON, D. P. SANDERS, P. SEYMOUR, and R. THOMAS. A new proof of the four-color theorem. *Electronic Research Announcements of the American Mathematical Society*, pages 2, 17 – 25, 1996.

[9] C. WARE. Color sequences for univariate maps. *IEEE Computer Graphics and Applications*, pages 8, 5, 41 – 49., 1988.