Visual Graph Comparisons with Bullseyes

Nathaniel Cesario
University of California, Santa Cruz

Alex Pang
University of California, Santa Cruz

Lise Getoor
University of Maryland

Lisa Singh
Georgetown University

ABSTRACT

We propose a Graph Comparison Bullseye as a metaphor to visually display similarities and differences between nodes in a single graph or differences between two graphs. Like an actual bullseye board for darts, different sectors represent different node types, different concentric rings represent different levels of similarity, with the center being an exact match, or bullseye! Thus, the two main parameters are distance from the center and an angle. With this, a simple match or no-match policy can be represented by a bullseye with a single center ring. Nodes that match a particular criterion will be inside the center ring, while everything else will be outside. If varying levels of agreement or disagreement need to be represented, additional concentric rings may be added, or these could be represented implicitly by the distance from the ring center. We examine how this method works for large datasets compared to methods such as graph topology where the visualization is based on the structure of the graph rather than vertex attributes. In addition, we show some of its similarity to parallel coordinates when examining graphics containing vertices of many different types.

1 THE BULLSEYE LAYOUT

Our goal is to create something both useful and easy to understand and use. A bullseye is something many people can look at and intuitively know that marks at different positions of the circle mean something “better” or “worse”. Given a graph $G = (V,E)$ where every vertex $v \in V$ has a type $t_i$ and a set of attributes $A_i$ attached to it, and every edge $e \in E$ has a type $t_e$ and a set of attributes $A_e$ attached to it, we will map a particular attribute to:

- The distance of the vertex from the bullseye.
- The angle within a vertex’s sector.

We note that this method produces patterns that can be used to extract information from a graph in a similar way to looking at graph topology [2].

1.1 Radius Mapping

Using a simple, single ring configuration, we initially assign all vertices to start in the outer ring of the bullseye (everything is a “miss”). In principle, we can simply make a pass over the graph and if certain criteria are met for a particular vertex, we mark that vertex as “matched”. However, the default we use for whether a vertex “matched” or not depends on whether we are comparing multiple graphs or a set of nodes within a graph. This is due to the fact that comparisons within a single graph and comparisons between multiple graphs can be interpreted differently, and we now explain how we make each of those comparisons.

To identify target nodes in multiple graphs we make the following requirement. Each vertex must have a unique identifier within a single graph. So two vertices with ID 12 in the same graph are not allowed. However, two vertices, each in different graphs with ID 12, is valid. If $\Omega = \{\text{a set of graphs}\}$ and $\omega = \bigcap_{G \in \Omega} V(G)$, then all elements in $\omega$ (all vertices common among all graphs) will be marked as “matched” and appear inside the inner circle of the bullseye.

To identify target nodes in a single graph given a set of nodes to examine, we do the following: let $S = \{\text{a set of vertices in G}\}$ and $N(v) = \{\text{set of all vertices adjacent to v}\}$, then $\omega = \bigcap_{v \in S} N(v)$.

Once we know whether a vertex is matched or not we can figure out its proper radius. Given the inner radius of the bullseye to be $r_i$ and the outer radius of the bullseye to be $r_o$ we have the following: For a particular $a_i \in A_i$, we say $\exists a_{i,\text{max}}, a_{i,\text{min}}$ such that $a_{i,\text{max}}$ is the maximum value for $a_i$ over all $V$ and $a_{i,\text{min}}$ is the minimum value for $a_i$ over all $V$. Then the radial distance $r$ of a vertex $v$ from the center of the bullseye is:

$$r = \begin{cases} r_i - \frac{a_i - a_{i,\text{min}}}{a_{i,\text{max}} - a_{i,\text{min}}} (r_o - r_i) & \text{if } v \text{ “matched”}, \\ r_o - \frac{a_i - a_{i,\text{max}}}{a_{i,\text{max}} - a_{i,\text{min}}} & \text{otherwise}. \end{cases}$$

1.2 $\theta$ Mapping

Each type of vertex gets its own sector of the bullseye. That is, if there are $n$ different types of vertices within a single graph and $N$ graphs, the bullseye is divided into $nN$ sectors where each sector occupies $\frac{360}{nN}$ of the bullseye. The start and end of each sector is obtained traveling in a counter-clockwise direction. So, if we place one graph on the bullseye with two types of vertices, the first type will lie in the sector from degree 0 to 180 degrees, and the second type will lie in the sector from 180 to 360 degrees. The angle where a vertex is placed is calculated in the same way as $r_o$ is above, except $r_i$ and $r_o$ would be the starting and ending angles of the node’s sector respectively.

Figure 1: The values $r$ and $\theta$ can be mapped to any attribute attached to that vertex e.g. the vertex’s degree, or the vertex’s local clustering coefficient.

2 USE WITH EXISTING TECHNIQUES

There has been a lot of work done on comparing graphs in a purely mathematical sense [4] and visualizing data with many variables [1, 3]. We attempt to use some of these techniques in unison with the bullseye to alleviate the computation complexity and enhance the visual comparison.
2.1 Pseudo Parallel Coordinates

With the work of Alfred Inselberg in mind, we attempt to use a parallel coordinate-like system. Each type of node lives in its own bullseye along the z-axis i.e. we now have a horizontal stack of bullseye. When viewing the bullseye along the z-axis, we see the normal bullseye representation. However, rotating the view about the y-axis 90 degrees shows the data laid out almost like a parallel coordinate system. There are some important differences. First, there is no restriction that a line connecting two vertices cannot cross a parallel axis. Second, with parallel coordinates, there is a notion of value associated with the vertical axis, that no longer has the same meaning since the vertical axis in this rotated view is just a projection of a node’s y coordinate, which may or may not have significant meaning depending on how the bullseye is rotated about the z-axis. This technique is mainly useful for highlighting associations between different types of nodes. Figure 2.1 is an example of this: we wish to compare an actor’s average rating (mapped to arc length) to a movie’s degree (mapped to radius). We first highlight all edges between green and red nodes (this blends the color of all other edges with the background) and then rotate the bullseye such that every red and green node lie on the y axis. This gives us a more familiar view in terms of parallel coordinates and we can see that actors with an average higher rating played in movies with smaller casts in the 1960s.

2.2 Use of Graph Spectra

Part of the goal of this visualization is that it be interactive. After choosing a visualization, users can manipulate it in 3-dimensional space with a reasonable response time. The limiting factor is usually in the calculation of comparison metrics. Looking at graph spectra may give a solution to some of these situations [4]. Spectral graph theory looks at the eigenvectors of the adjacency matrix for a particular graph to extract information about the structure of a graph. While the adjacency matrix itself is not a good tool for comparing graphs, Zhu and Wilson showed that different forms of the Laplacian (found by taking the difference of the diagonal degree matrix [4] and the adjacency matrix) is a good estimator for comparing graphs. While this option is not implemented in the current application, we can imagine loading several large graphs for comparison with an option given to run a comparison between all the graphs using one of the techniques mentioned in Zhu and Wilson’s paper [4]. This can give an indication of which graphs might be interesting to compare (i.e. very different for example) and which ones may not be worth looking at. Eliminating graphs to compare could help make the visualization more clear and increase performance when manipulating the view of the graph.

Another observation [4] is that the deletion of some edges can have little influence on a graph’s spectrum. If we cannot eliminate any graphs in the comparison, removing some edges might improve performance when trying to achieve real time interaction.

3 Examples and Applications

The initial motivation for this project was to compare graphs of social networks such as facebook, citations, and others. Due to space limitations, we just show some results with the IMDB dataset.

Figure 3: imdb.org data for the period 1940-2009. A node’s arclength ($\theta$) is a measure of their average rating, while a node’s radius is a measure of its degree. We see that the red nodes all lie at the beginning of their sector. This is because they have no rating associated with them in this graph. However, we can get a sense of how big the cast was by the movie’s degree, so red nodes closer to the center of the bullseye indicate movies with high degree (i.e. had many actors/actresses, and others). We can also see that actors (green) get the highest ratings while producer’s get the lowest (purple).

Figure 4: This is every decade from 1940 to present in IMDB compared using using the default matching algorithm (see Radius Mapping section). Here we see that every actor, actress, producer, and director in the 1940s was involved in some movie every decade since the 1940s. A red node in the center ring means that movie title appeared one or more times per decade since 1940.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 0937073.

References


