ABSTRACT

Many real world information can be represented by a graph with a set of nodes interconnected with each other by multiple type of relations called edge layers (e.g., social network, biological data). Edge bundling techniques have been proposed to solve cluttering issue for standard graphs while few efforts were done to deal with the similar issue for multilayer graphs. In multilayer graphs scenario, not only the clutter induced by large amount of edges is a problem but also the fact that different type of edges can overlap each other making useless the final visualization. In this paper we introduce a new multilayer graph edge bundling technique that firstly produces a preliminary edge bundling independently of the different edge layers and then deals with the specificity of multilayer graphs where more than one type of edges can be routed on the same bundle. The proposed visualization is tested on a real world case study and the outcomes point out the ability of our proposal to discover patterns present in the data.

Index Terms: I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation;

1 INTRODUCTION

Nowadays many types of data exhibit complex relational structures. For instance, by considering different social networks spanning over the same set of people, but with different life aspects (e.g. social relationships such as Facebook, Twitter, LinkedIn, etc.), we can get as many relation types as the different aspects. In biology, protein-protein interaction networks can be created considering the pairs of proteins that have direct interaction, physical association or they are co-localised [24]. More examples can be quoted from a gene network where genes are connected by considering the different pathway interactions and recommendation networks [15].

These data require a structure to support the representation of multiple relations among entities. A structure can fit these characteristic is the multilayer graph. A multilayer graph is defined as a graph, with the additional features that more than one edge can exist between the same pair of nodes and each edge may have a different type. Semantically speaking, considering the social network scenario, each edge type (or layer) can be interpreted as a particular social interaction between individuals. For example, a layer can represent interactions coming from the Facebook social network, another layer can represent interactions coming from LinkedIn and so on. Formally, given a set of layers $L = \{ L_1, \ldots, L_d \}$, a multilayer graph $G$ is defined as a tuple $(V, \{ E_i \}_{i=1}^d, L)$ where, $V$ is the set of vertices, $E_i \subseteq V \times V$ is the set of undirected edges over dimension $L_i \in L$.

Such a rich model introduces the possibility to represent more fine-grained information thanks to the multi-layer structure; on the other hand this extra information needs the definition of new visual tasks involving the analysis of the correlation among layers: In which layer(s) a community of nodes appears? What are the common patterns between layers? Which are the specific patterns of a layer w.r.t. the others?

Representing edges from multiple layers for large graphs induces highly cluttered visualizations. Grouping edges into bundles is a successful method to reduce edge cluttering in graphs [5, 10, 14, 13, 23, 8, 16, 11, 3]. Preliminary works on edge bundling also include techniques dealing with various kinds of graph, such as compound graphs [9] or directed graphs [21]. Unfortunately, previous works on multilayer graph visualization [1, 12, 4, 7, 6, 19, 20, 17] did not consider edge bundling approaches.

Applying standard edge bundling to multilayer graphs does not supply suitable results. Edges from different layers can be grouped together reducing the possibility to highlight patterns specific to a particular layer or, conversely, patterns shared among different layers. This fact motivates our research. In this paper, we propose a new edge-bundling technique that avoids to group edges of different layers. The result of our technique is a visualization in which a single map captures similarities and differences of edge distribution among layers.

More in detail, our proposed technique has four main steps: firstly it adapts the technique proposed in [14] to obtain a preliminary edge bundling (Section 2). Secondly, starting from the preliminary edge bundling, our framework smooths the edge visualization (Section 3). Successively, it divides the previously obtained bundle in layer specific bundles enforcing each bundle to contain only edges belonging to the same layer (Section 4). Finally, as the layer specific bundles can cross each others, we reduce the number of inter bundles crossing (Section 5).

2 INITIAL EDGE BUNDLING

We start to define some basic elements we use in the rest of the paper. The terms node and edge refer to multilayer graph node and edge. A control point is a vertex employed to route the edges between two nodes while a segment is a line between two control points over which more than one edge can pass through. A path is a set of consecutive segments between a pair of nodes.

As first step, our approach considers the multilayer graph as a simple graph where no distinction between edge layers is made. We name it flattened graph. The flattened graph has as many nodes as the original multilayer graph and if two nodes are linked in any of the layers of the multilayer graph, then an edge will exist in the flattened graph. The flattened graph is employed to obtain a preliminary edge bundling. To perform such bundling we adapt the algorithm Winding Roads proposed in [14]. Starting from a graph with predefined node positions, this algorithm performs edge bundling discretizing the space around nodes considering a mix of Voronoi diagram and Quad-tree. Instead of employ the algorithm as it was proposed, we only consider Voronoi diagram to perform the space discretization as we observed that directly apply the Winding Roads algorithm, as it is, results in an over-discretization of the space that negatively impacts the final result. The initial edge bundling step supplies a grid, derived by the Voronoi diagram, where the vertex of the grid are the control points and the lines between control
points are the segments over which more than one edge can pass through. The segments are successively used to route the edges of the multilayer graph into bundles.

3 Edge Smoothing

As we explained before, to route an edge we employ a set of consecutive segments that link two nodes of the multilayer graph. The same segment can be traversed by more than one edge type and smoothing such set of segments can be helpful to improve the final visualization. In order to better draw the edges between two nodes we propose an Edge Smoothing procedure that firstly determines the amount of free space around a control point and then creates new control points (and segments) to smooth the trajectory of the edges.

3.1 Determine free space around control points

The first step of the Edge Smoothing procedure is dedicated to understand how much free space is available around each control point in order to avoid overlap between edges and nodes in the multilayer graph visualization. This free space is quantified by a radius around the control points and such radius is represented by the distance between the control point and its closest node. For this reason we propose, later, a way to overcome this issue in Section 4.2.

3.2 Create new control points

Given a control point \( x \), once the radius \( d \) corresponding to the free space around it is determined, we can create new control points considering points at distance \( d \) on the adjacent segments to \( x \) (see red points in Figure 1b). Once the new control points are obtained, the curve can be smoothed adding new segments between new control points as depicted in Figure 1c. We distinguished the new control points w.r.t. the previous ones by a different color. More in detail, we used the red color for the new control points and the gray color for the control points that already exist in the initial Voronoi grid. As a result, we obtain smoother bundles than the previous ones.

If the smoothing result needs to be ameliorated, we can repeat the previous process recomputing the radius related to the available space around each control point and, then, create new control points. This procedure ensures that, if there was no overlap before between nodes and edges then the Edge Smoothing step will not introduce any of them. This is due to the locations where the new control points are placed and to the fact that the radius related to each control point is computed considering its closest node. Repeating the smoothing procedure can lead to better visualization but it will increase the computational time.

4 Per Layer Bundle Division

At this point, the different types of edges can be routed through the set of control points and segments. Unfortunately, the obtained bundles are not specific for a layer, this means that edges coming from different layers can overlap each others. To tackle this issue, we propose to divide each bundle of the flattened graph to obtain different bundles, one for each edge type traversing the corresponding bundle of the flattened graph. To do this, firstly we determine the free space around each control points (in the same way as done in Section 3.1). Then, we break the bundle splitting it into as many bundles as the different types of edges it contains. The proposed approach does not really avoid overlap between edges and nodes as shown in Figure 2b where violet segment overlaps green node. For this reason we propose, later, a way to overcome this issue in Section 4.2.

4.1 Bundle Division

In this section we use the similar notation we employ in Section 3.2 to explain how we divide the bundle. Considering Figure 2a, for each new control point (red points) we compute again the distance between it and the closest node to determine the circle corresponding to its available free space (see dotted red circle in Figure 2a).

Then, for each control point we determine a segment that pass through it and intersects the circle determining the free space. For instance, in Figure 2a we can observe that the control point \( cp \) is surrounded by the circle \( c \). Considering \( cp \) and \( c \), we draw the segment \( s \) that intersects \( c \) at the points \( p_1 \) and \( p_2 \). The segment is perpendicular to the segment linking \( cp \) and \( p_2 \) from which it is generated. In the example in Figure 2a, the segment \( s \) is perpendicular to the segment \( [cp, cp'] \). Successively, we duplicate the control point \( cp \) into several control points, one for each edge type of the bundle. Finally, we locate the new control points uniformly on \( s \). Once this operation is done, for each original control point of the bundle, we use the new control points to draw the different edge types without a particular order (see Figure 2b).
cause consecutive control points, traversed by the same type of the multilayer graph visualization. This phenomenon happens be-
induce edge crossing between edges of different types, affecting
As shown in Figure 3a, the previous steps of our framework can
number supplies a good trade off between computational complex-
our approach we fix bundle division step introduced in Section 4.1 (see Figure 4b). In
formally distributed along its length. Such a trick helps to deal with the overlap issue since new control points are added to refine the
the segment incident on control point \( c_1 \) (resp. \( c_2 \)) be the endpoints of \( s_1 \) (resp. \( s_2 \)). Then if a node \( n \) falls into the polygon defined by the set of points \( (p_1, p_2, p'_2, p'_1) \) (see Figure 4a) we over-discretize the segment \([c_1, c_2]\) in \( K \) segments adding \( K - 1 \) control points uniformly distributed along its length. Such a trick helps to deal with the overlap issue since new control points are added to refine the bundle division step introduced in Section 4.1 (see Figure 4b). In our approach we fix \( K = 10 \) as we empirically observe that this number supplies a good trade off between computational complexity and visual result.

5 Bundle crossing reduction
As shown in Figure 3a, the previous steps of our framework can induce edge crossing between edges of different types, affecting the multilayer graph visualization. This phenomenon happens because consecutive control points, traversed by the same type of edge, could not have the same order on their segment. For instance, we can observe in Figure 3a that control points employed to draw the orange bundle do not have the same relative position along the different segments. To deal with this issue, we adapt the barycenter heuristic, usually employed to draw DAGs [22]. More in detail, in our case we want to find the order of the control points on the line \( d_i \) that minimizes the number of crossing edges. This problem is closely related to the metro-line crossing minimization task [18]. Alternative heuristics have been proposed in [18] and they could be also adapted to our problem.

Given the line \( d_i \) and a set of control points lying on it, for a control point of a particular edge type we compute the barycenter of the projections of its neighbors on \( d_i \). We repeat the same procedure for all the control points lying on \( d_i \). Considering our example in Figure 3a, Figure 3b shows the barycenter computed for each control point on \( d_i \) considering its neighbors. The barycenters are highlighted by points on \( d_i \) with red border. Then, the control points of \( s \) are reordered according to the computed barycenter obtaining a new order of the original control points as depicted in Figure 3c.

4.2 Avoiding Edge-Node Overlap

In a more general scenario, if we consider only the space around the control points to draw the different edge types we can fall in a situation similar to the one reported in Figure 2b (violet segment overlaps green node). As we can observe, one or more of the new segments can overlap the nodes inducing ambiguity in the visual-
ization. In order to address this issue, we adopt the following strategy, named Edge-Node overlap heuristic: let \( s_1 \) (resp. \( s_2 \)) be the segment incident on control point \( c_1 \) (resp. \( c_2 \)) as explained in Section 4.1 and, \( p_1 \) and \( p'_1 \) (resp. \( p_2 \) and \( p'_2 \)) be the endpoints of \( s_1 \) (resp. \( s_2 \)). Then if a node \( n \) falls into the polygon defined by the set of points \( (p_1, p_2, p'_2, p'_1) \) (see Figure 4a) we over-discretize the segment \([c_1, c_2]\) in \( K \) segments adding \( K - 1 \) control points uniformly distributed along its length. Such a trick helps to deal with the overlap issue since new control points are added to refine the bundle division step introduced in Section 4.1 (see Figure 4b). In our approach we fix \( K = 10 \) as we empirically observe that this number supplies a good trade off between computational complexity and visual result.

6 Case Studies
In this section we illustrate a case study that shows the practical benefits of our method to visualize multilayer graph data. The case study investigates social interaction among people who communicate through different media. We employ the Reality Mining dataset\(^1\). This multilayer graph contains human interaction data collected by the MIT Media Lab. The experiment was carried out on a total of 94 people and this also represents the number of nodes in the corresponding multilayer graph. The different layers offered by the dataset pertain to the means of interaction between a pair of people. Namely, CALL layer refers to subjects calling each other, FRIEND layer contains friendship claims, SMS layer builds on text message exchanges (SMS) and DEVICE layer contains Bluetooth device scans. For our purpose we consider the first three layers (CALL, FRIEND and SMS) discarding the DEVICE layer as it is a quasi-clique and it does not provide useful information in the context of edge bundling visualization. The three considered layers have, respectively, 177, 82 and 113 edges. Experiments are carried out on a Desktop Computer with Intel Core i7-3770 CPU @ 3.40 GHz x 8, with 8 Gb of RAM.

Figure 5 shows the result of our multilayer graph edge bundling on the portion of Reality Mining dataset we consider. The violet edges correspond to the CALL layer, the green edges correspond to the FRIEND layer and SMS layer is depicted in orange. Curved edges that are not parts of bundles are artifacts of the initial bundling algorithm (see Section 2). Computation time is of 2.40 seconds with 2 iterations of the edge smoothing process.

Analyzing the visual result, we can note that the graph has three cluster structures: \( C_1, C_2 \) and \( C_3 \). We can observe that \( C_1 \) and \( C_2 \) contain edges of different types while \( C_3 \) contains only people that

\(^1\)http://realitycommons.media.mit.edu/realitymining.html [Online; accessed 12-March-2015]
Figure 5: Result of the multilayer graph edge bundling approach on the Reality Mining dataset with three focus on different interaction patterns our method helps to highlight.

Figure 6: Visualization of the subsample of BIOGRID multilayer graph: a) the visualization before applying our approach b) the result obtained with the multilayer graph edge bundling strategy.

7 CONCLUSION AND FUTURE WORK

In this paper, we have presented a novel and intuitive technique to route different types of edges into bundles to visualize multilayer graphs. Our approach reduces edge clutter at both global and per layer level. As future work, we plan to consider the number of edges passing through a bundle to determine its width and manage weighted multilayer graph.
REFERENCES


