Visualizing dynamic graphs are challenging due to the difficulty to preserving a coherent mental map of the changing graphs. In this paper, we propose a novel layout algorithm which is capable of maintaining the overall structure of a sequence graphs. Through Laplacian constrained distance embedding, our method works online and maintains the aesthetic of individual graphs and the shape similarity between adjacent graphs in the sequence. By preserving the shape of the same graph components across different time steps, our method can effectively help users track and gain insights into the graph changes. Two datasets are tested to demonstrate the effectiveness of our algorithm.

Index Terms: dynamic graph, graph layout algorithm, Laplacian matrix, force directed layout, stress model

1 INTRODUCTION

Dynamic graph visualization, which targets at depicting graphs that evolve over time, has gained increasing attention recently. It is featured in a wide range of applications, such as biochemistry, software engineering, and social network analysis. In a typical application, the data is modeled as an evolving graph and further represented by a sequence of snapshots, which depict the graph structure at different time points. By observing the changed and unchanged graph components in the sequence, analysts discover knowledge in the back-end data, such as how proteins interact with one another and how information flows within a social network.

Previous research has indicated that preserving the mental map is a key to an effective exploration of dynamic graphs. Mental map is to the internal structural cognitive information created by the users, while observing the graph layouts. A well preserved mental map during the exploration can dramatically reduce the cognitive cost and save more mental bandwidth for users to discover evolution patterns. Various dynamic graph layout algorithms attempt to preserve the mental map by keeping the overall shape stable in the graph sequence. The general principle is to first apply a static graph algorithm to generate the layout for the first time step. For the subsequent time steps, previous layout results are used as constraints, in order to preserve the positions of nodes as possible.

Although keeping node absolute positions stable across multiple snapshots can help users tracking individual nodes, we find that it has two limitations. First, it may damage the aesthetic of layouts, such as bringing extensive edge crossings, especially when the graph is changing dramatically. In some cases, it may even mislead users. For example, at a time point, one graph component breaks into two. Since the absolute positions of nodes are strongly restricted by the previous snapshot, the two components may still overlap and appear as one connected component. Second, preserving the absolute positions may not be necessary in some scenarios. Research in human perception and psychology suggested that human have tendency to seek out the whole boundary of the shape rather than position, and they could effectively recognize objects based on silhouettes or components.

Based on these two observations, we propose a new strategy to preserve the mental map by relaxing the absolute position constraint and enforcing the preservation of graph component shapes. Based on Laplacian constrained distance embedding, our strategy contains two steps for each time step. First, a static graph layout algorithm is applied to a snapshot, disregarding the graph layout results before it. After that, we obtain an initial layout with aesthetic criteria such as uniform node distribution and minimal number of crossings. The second step is layout adjustments. By running LCDE algorithm, we refine the initial layout and force nodes to generate overall shapes appeared in the previous snapshot, so that users can easily build connections of the same graph component between adjacent snapshots.

Comparing with existing dynamic graph layout algorithms, our strategy has two distinct advantages. First, our LCDE method is a post-processing step, so that our strategy can fully take advantage of the rich set of existing static layout algorithm. By coupling with different algorithms, our strategy can easily facilitate different applications with the ability to preserve the mental map. Second, our strategy maintains the graph component shapes instead of the absolute positions of nodes. Thus nodes have more freedom to move during the adjustment, and users still can easily recognize the layout result.

2 RELATED WORK

Comparing with static graph visualization, dynamic graph drawing research is relatively new. For example, as one of the first few pioneers, North introduced DynaDag in 1995. In his work, a heuristic method is adopted to draw an evolving directed acyclic graph as hierarchies. At the same time, the concept of preserving the mental map is also proposed as a desirable feature of dynamic graph layout algorithms and draws more and more attention.

Several studies have been conducted to understand the role of mental map in the exploration of dynamic graph layouts. Important conclusions are developed from the studies. For example, suggests that restricting node absolute positions may cause significant node overlap and actually damage the graph readability. On the other hand, the majority of the recently proposed techniques that target at preserving the mental map are based on force-directed placement and animation. In the force calculation, the mental map is often mathematically encoded as a weighting function, which restricts the node absolute movement during the po-
sition adjustment [1]. For example, GraphAEL [16] applies a force directed algorithm to individual snapshots, and connects the same node in adjacent snapshots with virtual links. By controlling the stiffness of the links, users can easily adjust how much of the mental map is preserved. Kumar and Garland [22] develop a stratification-based hierarchical layout algorithm, which not only speeds up the general force directed algorithm but also accommodates time-varying graphs. Once layouts are generated for all snapshots, animation is often used to illustrate the graph evolution. In an animation, the node positions between adjacent snapshots are interpolated, so that analysts do not need to spend much mental bandwidth keeping track of nodes.

Although animation has been well adopted for dynamic graph visualization, certain patterns are challenging to discover due to the limit of short-term memory [29]. Thus, alternative static visual representations are explored as well. For example, small multiples [6] is a natural alternative to animation, which shows graphs of different snapshots side by side to facilitate comparison tasks.

Comparing with the above techniques, our strategy preserves the mental map by constraining the graph component shapes instead of the node absolute locations. Therefore, our method may avoid unnecessary overlapping and still preserve the mental map. In addition, our method works as a post-processing operation. It can be combined with any existing static graph layout algorithms and enable them the ability to preserve the mental map.

Other alternatives are more adventurous by abandoning the node-link diagram visual representation. For example, EdgeSplatting [5] places vertices for each snapshot on vertical, parallel lines perpendicular to the horizontal time line. Then, the structures of individual graphs are encoded as the texture between neighboring parallel lines. GraphFlow [7] transforms graph evolution patterns into vector fields and adopts flow visualization to analyze them. Although they succeed in certain applications, the unfamiliar visual representations also damage the readability and require certain learning time from users.

3 LAPLACIAN-BASED DYNAMIC GRAPH DRAWING

Given a sequence of graphs \( \{ G_0, G_1, ..., G_n \} \), where \( G_i = (V_i, E_i) \) is a snapshot that represents the structure of graph at time \( i \) \( (0 \leq i \leq n) \). The goal of our algorithm is to produce a sequence of layouts \( \{ L_0, L_1, ..., L_n \} \) that is both aesthetically pleasing and coherent to preserve the mental map, where \( L_i \) is a straight line drawing of \( G_i \).

Since \( G_0 \) is the first graph and users do not possess any mental map yet, we simply adopt a static layout algorithm to generate \( L_0 \). In this paper, we use the simulated annealing force-directed layout algorithm (SAFL) [8] as an example. Once \( L_0 \) is generated, the other layouts \( L_i (i \geq 1) \) can be computed based on the following two steps:

1. Run a force directed layout algorithm for \( G_i \) to get an initial layout \( L_i^\ast \), which we assume has the desired aesthetic criteria. In this paper, we again adopt the SAFL algorithm [8] as an example, due to its popularity.

2. Run our LCDE algorithm, which slightly modifies the node positions in \( L_i^\ast \), based on the overall graph shapes in \( L_{i-1} \), to generate \( L_i \). By tuning a parameter, users can easily balance the aesthetic criteria against the mental map preservation.

In the following section, we describe these two steps in detail.

3.1 Initial Layout

By default, SAFL method used random input locations for the graph. However, in our specific scenario, it is natural to leverage the information in \( L_{i-1} \) to speed up the layout process:

- for node \( v \) in both \( G_i \) and \( G_i-1 \), we directly use its coordinate in \( L_{i-1} \) as the input coordinate;
- for node \( v \) in \( G_i \) but not in \( G_{i-1} \), if it has only one neighbor \( p \) in \( G_{i-1} \), we randomly choose a position on the circle that takes \( p \) as the center and the average edge length in \( L_{i-1} \) as the radius. If it has more than one neighbors in \( G_{i-1} \), we then choose the centroid of the neighbors as its input coordinate.
- for the rest nodes, input coordinates are generated randomly.

3.2 Layout Refinement

We assume that \( L_i^\ast \) represents the ideal layout with great aesthetic criteria and readability. However, it does not preserve the mental map, since it does not leverage any information from \( L_{i-1} \). Inspired by the Laplacian constrained graph layout algorithm [32], we take the subgraph in \( L_{i-1} \) that consists of nodes existing in both \( G_{i-1} \) and \( G_i \), and run the LCDE algorithm on \( L_i^\ast \). Specifically, we take the layout \( L_i^\ast \) as the ideal layout of nodes only existing in \( G_i \), and the layout \( L_{i-1} \) as the ideal layout of nodes in both \( G_i \) and \( G_{i-1} \). Accordingly, a stress energy function can be defined as:

\[
E = \sum_{(j,k) \in E_i} \frac{(||x_j-x_k||-d_{j,k})^2}{(d_{j,k})^2} + \alpha \sum_{j \in V_i, k \notin V_i} \frac{(||x_j-x_k||-d_{j,k})^2}{(d_{j,k})^2}
\]

where \( E_i \) is the edge set of \( G_i \), \( V_i \) and \( V_{i-1} \) are the node sets of \( G_i \) and \( G_{i-1} \). \( d_{j,k} \) is the desired distance between nodes \( j \) and \( k \). Based on the above discussion,

\[
d_{j,k} = \begin{cases} 
\text{distance between } j \text{ and } k \text{ in } L_i^\ast, & \text{if } (j,k) \in E_i; \\
\text{distance between } j \text{ and } k \text{ in } L_{i-1}, & \text{if } j \in V_{i-1} \cap V_i.
\end{cases}
\]

The energy function consists of two terms. The first term emphasizes the layout of \( L_i^\ast \), while the second term emphasizes the layout
of $L_{t-1}$. $\alpha$ is the weight factor that controls how much we want to preserve the mental map. The bigger the $\alpha$, the more likely the shapes in $L_{t-1}$ are preserved in the resulting $L_t$. The small the $\alpha$, the more likely the desired aesthetic features in $L_t$ are kept in the resulting $L_t$.

Figure 1 illustrates the procedure of the above algorithm. We can clearly see that the graph shape at $L_{t-1}$ is successfully kept to $L_t$ after our two step process. In addition, Figure 2 illustrates how $\alpha$ is used to control the mental map preservation. We can see that when $\alpha$ is bigger, the shape in $L_{t-1}$ becomes more prominent in $L_t$.

Assuming Equation 1 achieves the minimum when the gradient vanishes, we need to solve $E' = 0$, which can be represented in the matrix form:

$$(L_w + \alpha L_w^U)x = (L_{w,d} + \alpha L_{w,d}^U)x$$

(2)

where the weighted Laplacian matrices $L_w$, $L_w^U$, $L_{w,d}$, and $L_{w,d}^U$ are defined as:

$$(L_w)_{jk} = \begin{cases} \frac{\sum_{j \in E}(d_{jk})^{-1}}{\alpha}, & \text{if } j = k; \\ -\frac{1}{\alpha}, & \text{if } (j, k) \in E; \\ 0, & \text{otherwise.} \end{cases}$$

$$(L_w^U)_{jk} = \begin{cases} \frac{\sum_{j \in V_1 \cap V_2}(d_{jk})^{-1}}{\alpha}, & \text{if } j = k; \\ -\frac{1}{\alpha}, & \text{if } j, k \in V_1 \cap V_2; \\ 0, & \text{otherwise.} \end{cases}$$

$$(L_{w,d})_{jk} = \begin{cases} \frac{\sum_{j \in E}(d_{jk})^{-1}||x_j - x_k||}{\alpha}, & \text{if } j = k; \\ -\frac{1}{\alpha}||x_j - x_k||, & \text{if } (j, k) \in E; \\ 0, & \text{otherwise.} \end{cases}$$

$$(L_{w,d}^U)_{jk} = \begin{cases} \frac{\sum_{j \in V_1 \cap V_2}(d_{jk})^{-1}||x_j - x_k||}{\alpha}, & \text{if } j = k; \\ -\frac{1}{\alpha}||x_j - x_k||, & \text{if } j, k \in V_1 \cap V_2; \\ 0, & \text{otherwise.} \end{cases}$$

where $x_j$ and $x_k$ are the position coordinates of node $j$ and node $k$, $||x_j - x_k||$ is the real distance between nodes $j$ and $k$.

We use the conjugate gradient method [19] to solve the Equation 2. First, we use $L^*$ as the $|V| \times 2$ matrix $x$ in the right side, and solve the system with it. Then the solution is inserted to the system and solve it again until the layout is stabilized.

### 3.3 Performance Analysis

Conjugate gradient methods running time is evenly distributed among the iterative. Almost the entire solving time is devoted to performing the matrix-vector multiplication. Each such multiplication takes $n^2$ flops (floating point operations) [19]. In practice, for most graphs we have experimented with, CG process is very fast since the total iteration number is typically less than $n/3$. Since CG benefits the fact that we fix $L^*$ as the initial solution of $x$ at first iterative and have an initial approximate solution from the previous iteration. What's more, the overall number of iterations increase very moderately with the size of the graph [19]. Therefore, the time complexity of LCDE algorithm is $O(k \times n^2)$, where $k$ is the iteration number.

### 4 Experiment

We use two datasets to demonstrate our algorithm's effectiveness. One dataset is adopted from Frishman's work [17], the other one is an academic collaboration graph. Both datasets confirm our algorithm's distinctive performance on preserving mental map.

Figure 3: Comparison of the layout results of the new fraternity data [26]: (a) the results generated by our method; (b) the results generated by the online dynamic graph drawing algorithm [17]
4.1 Newcomb’s Fraternity Dataset

The first case is Newcomb’s fraternity data [26], which represents friendship relations between college students. This data is first used by the peer-influence (PI) algorithm of SoNIA. Later, Frishman et al. [17] also visualized the same dataset to demonstrate their online dynamic graph drawing algorithm. A green subgraph can generate a aesthetic result by reducing the movements of nodes. Figure 3 is the comparison between our method (Figure 3(a)) and Frishman et al.’s algorithm (Figure 3(b)).

This data has six snapshots (T0 to T5). At T0, our method and Frishman et al.’s both use simulated annealing force-directed algorithm and generate similar layouts. Due to different parameter settings, the graph shapes are distorted a little. However, the node relative positions are nearly identical in the two layouts.

To demonstrate the effectiveness of our method, we outline two subgraphs (in orange and green) in both sequences and track their change over time. From the figure, we can see that our layout is significantly more stable than Frishman et al.’s result. For example, in the whole sequence, the green subgraph is kept very stable by our algorithm, despite all the edges added to it or removed from it in contrast, the layout changes dramatically in Frishman et al.’s result. For example, node 8 and node 13 suddenly exchange their locations at T1. And the total structure is completely changed at T1, T3, and T4. For the orange subgraph, it is clear that the T3 is the turning point for our algorithm and Frishman et al.’s. However, the change in our layout is considerably smoother, as the contour shrinks smoothly. In contrast, the orange subgraph in Frishman et al.’s is abrupt and hard to track. For further comparison, we mark node 4 with a red cycle, we can see that it makes big jumps at T1, T3, and T4, which possibly cause the abrupt shape change.

4.2 Siggraph Collaboration Dataset

The second case shows the evolution of collaboration relations between affiliations of publications. We collect SIGGRAPH and SIGGRAPH Asia papers from 2000 to 2012 (13 snapshots), which totally consists of 46 affiliations and 74 collaboration relationships. The dynamic graph is created as follows: if two different affiliations appear in one paper, they have an edge in the year of the paper published. To enlarge graphs, sliding window technique is applied to accumulate nodes and edges for each year.

Four layouts (from 2004 to 2007) are displayed in Figure 4. We can see that some sub-graph structures can be tracked easily. For example, the five red nodes appeared in the first layout, and the star shape is kept during the whole time. Similarly, the triangle shape of the nodes in blue is also kept well in the sequence. In particular, some structures are also kept even after translation or rotation adjustments, such as the green nodes in the right down corner. At start, the green component is isolated from the main component. However, in 2006, a new edge connecting “microsoft research” and “microsoft research asia” appears, which makes the green component translate and rotate to keep balance. Our algorithm successfully preserve the green shape so that users still can easily recognize it. This case demonstrates our algorithm’s good performance on preserving mental map, especially its flexible structure shape keeping ability.

5 Conclusion

In this paper, we proposed a novel approach to achieve highly stable and aesthetic layouts for dynamic graph visualization. Our algorithm has two highlights. The first one is universality. Our algorithm works as a post-processing step and can be coupled with any static layout algorithms, which may be designated specifically for different applications, and enable them the ability of mental map preservation. By tuning one parameter, users can easily decide exactly how much mental map need to be preserved. Second, instead of restricting the absolute positions of nodes, our algorithm more focuses on preserve the graph component shapes, which gives layout algorithm more freedom to adjust the node positions and achieves better aesthetic results.

There are several avenues for the future work. For example, the top one in our waiting list is to make this parameter selection automatically, if the algorithm can self-tuning mental map preserving degree, the aesthetic of the layout will be improved accordingly. Also, we plan to add the editing function to allow users to interactively adjust the layouts to better facilitate the readability.

Acknowledgements

The authors wish to thank the anonymous reviewers for their valuable comments. This work is supported by NSFC No. 61170204 and the National Program on Key Basic Research Project (973 Program) No. 2015CB352503.

References


