Intelligent Graph Layout Using Many Users' Input

Xiaoru Yuan, *Member, IEEE*, Limei Che, Yifan Hu and Xin Zhang

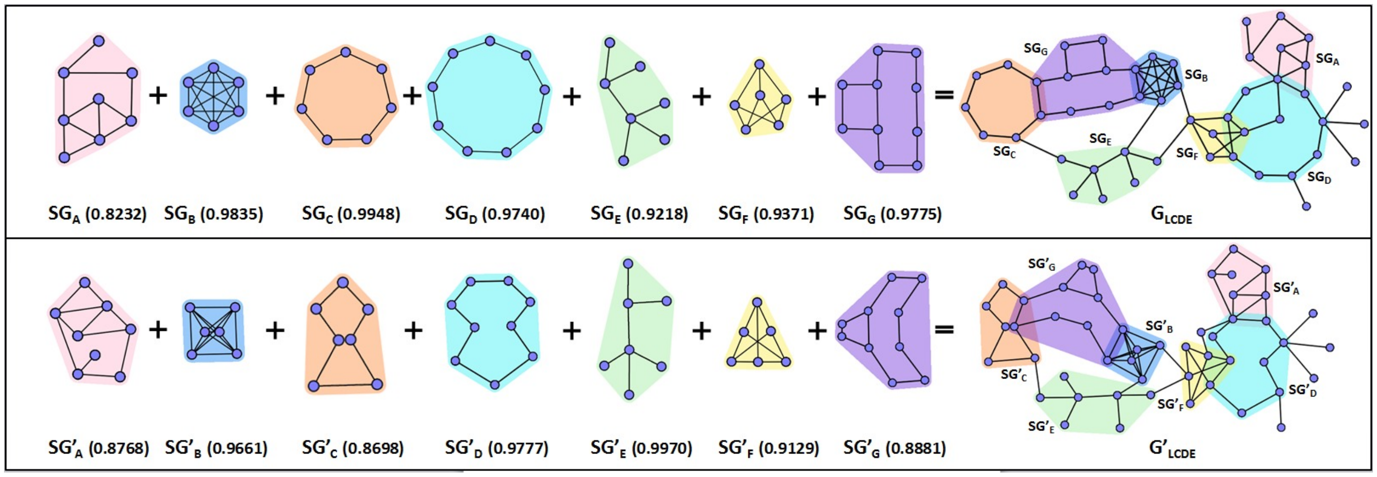


Fig. 1. Two layouts that combine many users' input sub-graphs using our Laplacian Constrained Distance Embedding (LCDE) algorithm. Layout configurations of the input subgraphs are well preserved in the final layouts of the merged graphs. The number under each subgraph is the similarity score between the user layout and the layout of this subgraph in the merging result (G_{LCDE} and G'_{LCDE}).

Abstract—In this paper, we propose a new strategy for graph drawing utilizing layouts of many sub-graphs supplied by a large group of people in a crowd sourcing manner. We developed an algorithm based on Laplacian constrained distance embedding to merge subgraphs submitted by different users, while attempting to maintain the topological information of the individual input layouts. To facilitate collection of layouts from many people, a light-weight interactive system has been designed to enable convenient dynamic viewing, modification and traversing between layouts. Compared with other existing graph layout algorithms, our approach can achieve more aesthetic and meaningful layouts with high user preference.

Index Terms—graph layout, Laplacian matrix, force directed layout, stress model, merging, editing, crowd sourcing

1 Introduction

Graphs are used widely to represent physical networks, social connections, or other abstract relationships. Graph drawing algorithms aim at producing pleasant and readable visual representation of graphs. Traditional graph layout methods, such as force-directed or spectral layout algorithms, are all automatic graph layout techniques that take input data and generate one final layout. Several other methods can produce different styles of layouts, but together these may not satisfy the users' special needs. For example, if we want to force a ten-node circle in the layout, or achieve other specific topological structures when visualizing graphs from applications in chemistry or biology , the aforementioned automatic algorithms cannot retain such user defined local structures.

In this paper we propose algorithms that facilitate user participation and leverage human intelligence in the process of layout generation. Through an interactive system, we allow users to edit part of a graph by changing node positions. We then merge the users' modification with the original layout in a well-founded manner to obtain a layout with user defined subgraphs as constraints (shown in Fig. 1). Here we use the term *users* in a general sense – the users could be human operators, or graph drawing systems designed to handle certain subgraphs well (for instance, a tree drawing algorithm for drawing tree subgraphs). This approach of multi-agents cooperative graph layout combines the ability of human beings in producing good drawings of small subgraphs, with the power of automated algorithms in getting the overall layout structure right. Our algorithms can incorporate drawings of many subgraphs and give an aesthetic layout of the whole graph, while satisfying the practical needs of the underlying application.

The remainder of the paper is organized as follows. In the next section, we review related work on constrained layout, divide and conquer approaches and previous studies comparing automatic and manual graph drawing. This is followed, in Section 3, by a detailed discussion of our algorithms. We then give, in Section 4, case studies that apply our algorithms to incorporate user-generated subgraphs of both abstract and real world networks, followed by discussions in Section 5. We conclude with directions for future work in Section 6.

2 Related Work

The area of graph drawing has progressed considerably to the extent that there are now many layout algorithms for different drawing styles, from circular layout [6, 19], to hierarchical layout [14], orthogonal layout [15], and force-directed layout [13, 17, 24]. Efficiency and effectiveness of force-directed algorithms are further improved by the

- *Xiaoru Yuan, Limei Che and Xin Zhang are with Key Laboratory of Machine Perception (Ministry of Education), and School of EECS, Peking University, E-mail: {limei.che,xiaoru.yuan}@pku.edu.cn.*
- *Xiaoru Yuan is also with Center for Computational Science and Engineering, Peking University.*
- *Yifan Hu is with AT&T Labs, E-mail: yifanhu@research.att.com.*

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.

multilevel approach and fast force approximation [21, 22, 31], as well as by parallelization and the use of GPU [16, 3, 23].

Papadopoulos and Voglis [27] proposed a divide and conquer strategy for drawing graphs via modular decomposition. The graph is decomposed into modules and laid out, and combined using a modified spring embedder. While the decomposition approach can reveal interesting regular graph structures (e.g., cliques), the spring embedder treats each module as a hyper-node occupying a rectangular area, without considering individual nodes and edges it represents. Consequently the orientation of each module, which may not be optimal in the context of the whole graph, is unchanged in the combination process. Archambault et al. [2] proposed TopoLayout which recursively detects topological features inside the graph, and uses the appropriate algorithms for each feature, it then employs an overlap removal algorithm to remove overlap between subgraphs. These previous works differ from our work in that we attempt to harness the intelligence of human beings to help construct aesthetic drawings, and that we propose a principled approach of combining multiple subgraph drawings.

Even with the availability of many layout algorithms, sometimes it is not possible to produce a drawing that satisfies the needs of the end users, in part because aesthetic beauty cannot be captured easily by equations. In addition, there could be constraints relevant to the field of application that current algorithms cannot take care of. In recent years, there have been studies of user generated graphs and how they compare with automatic graph layout. Van Ham and Rogowitz [30] allowed users to generate their own layouts of small graphs. By examining these manually arranged layouts, they verified common aesthetic standards and found a new criterion – a tendency to arrange the clusters inside convex-hulls. This criterion is also reflected by one of our cases (see Section 4). A later work [8] made another comparison of user-generated and automatic graph layouts, and concluded that the best of user-generated layouts performed as well as or better than layouts based on physical models. However, with the exception of a few manual layouts, most of the user-generated layouts were not as good as the model-based layouts. This is because people with less graph drawing experience found it difficult to handle graphs of over 50 nodes.

In order to overcome this barrier, in this paper we allow each user to work on a small part of the graph, which makes it easier for the user to achieve a good layout. We then merge input from multiple users into a layout for the full graph. In this way, we combine the intelligence of users to create a layout which is better than the results from any single automatic algorithm.

Work on constrained graph layout appeared as early as the 1990's [29, 26]. Tim Dwyer et al. [9] proposed the DIG-COLA algorithm applied to directed graph drawing. Graph nodes were divided into different layers, and the constraints were introduced to keep these layers separated. Dwyer et al. [10] attempted to solve constrained graph layout problems with a combination of stress majorization and constrained programming techniques. Constrained optimization also provided the foundation for Dunnart, a constraint-based graph drawing tool [11]. Further work by Dwyer [7, 12, 4] looked at fast handling of simple constraints such as linear or circular constraints. The difference between these approaches and ours is that we are not attempting to satisfy hard constraints, but rather integrate multiple subgraph drawings into one, such that individual drawings are satisfied as much as possible.

Our work uses previous distance embedding techniques. One of the basic models for distance embedding is the stress model [24], which can be solved with the stress majorization technique [20].

3 Algorithms for Incorporating Multi-user Input

In this section we describe our algorithms for incorporating multi-user inputs. As a starting point, a graph is laid out using a suitable graph layout algorithm. This graph is presented to multiple users/agents, who will inspect the layout, and may choose to change part of the embedding to satisfy their aesthetic preferences or practical needs. The users/agents could be human operators, or graph drawing systems designed to handle certain subgraphs well – for instance, a tree drawing algorithm for drawing tree subgraphs. The challenge is then to incorporate these multi-user inputs so that the subgraph drawings are maintained as much as possible, and at the same time they are pieced together into a good overall layout of the original graph.

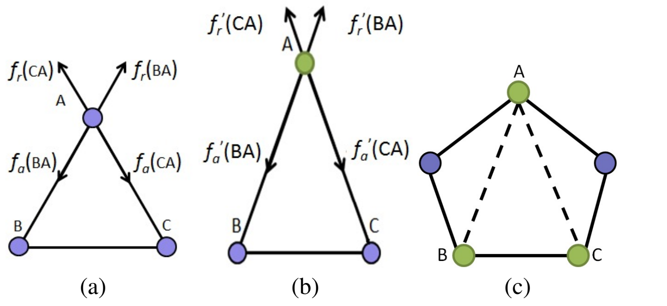


Fig. 2. Force analysis in the FDP algorithm and virtual edges in the CFDP algorithm. (a) A balanced system; (b) an unbalanced system; (c) virtual edges (dashed) added to edited nodes (green).

We first give some definitions. We consider undirected graphs with no loops or multiple edges. Given a graph $G = (V,E)$, with V the set of nodes (vertices) and E the set of edges, we denote the set of nodes connected to node i as $N(i)$. The purpose of graph drawing is to find Cartesian coordinates x_i for each node $i \in V$, so that the resulting drawing is aesthetically pleasing.

Let S be a subset of the vertices of graph G, and G^S the subgraph of G induced by S. In our system, a user can define a subgraph by clicking on nodes, or using the lasso to select a group of nodes. The user can then adjust the subgraph layout based on his/her preference. This user defined subgraph layout is saved, along with layouts from other users. For convenience, from now on we use the term *subgraph* to denote both the subgraph of vertices and edges, as well as the user generated subgraph layout, provided that doing so does not cause ambiguity within the context.

In the following subsections we first recall the force-directed layout algorithm, then propose two algorithms for merging users' subgraphs. Finally we introduce a measure of similarity between a user input subgraph and the merged layout, which allows us to gauge the extent to which inputs from users are obeyed.

3.1 Constrained Force-Directed Placement

This algorithm uses a variant of the Force-Directed Placement [17] (FDP) algorithm. We give each subgraph layout a score (see Section 4.1 for the scoring system) – if a layout is of good quality, it is given a high score. Layouts with the highest score are used for merging. We calculate the distance of each pair of nodes in the subgraphs, and take these as constraints. We call the resulting algorithm Constrained Force-Directed Placement (CFDP). Specifically, in FDP, each node is influenced by attractive forces which act like springs to pull nodes together, and repulsive forces which keep nodes a reasonable distance away from each other. The forces applied to node i by node j are defined as:

$$f_a(j,i) = \frac{\|x_i - x_j\|^2}{K},\ \{i,j\} \in E;$$
$$f_r(j,i) = -\frac{K^2}{\|x_i - x_j\|},\ i,j \in V,$$

where K is a force coefficient representing the ideal edge length. Fig. 2(a) shows a balanced system. For node A, when the distance between nodes A and B equals K, $f_a = K$ and $f_r = -K$. Fig. 2(b) is a user defined graph. Here node A is pulled far away from B and C by the user, so the attractive force on node A is now stronger, $|f_a'(B,A)| > |f_r'(B,A)|$ and $|f_a'(C,A)| > |f_r'(C,A)|$. Therefore, if we use the same force coefficient K for every edge, this is an unbalanced system, in which node A will move away from the user defined position toward its balanced position. In order to maintain the topological

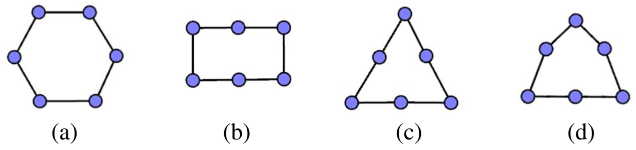


Fig. 3. The CFDP algorithm will combine many input graphs. When a subgraph is edited by more than one user, it contains all user inputs equally by default. (a) Initial layout; (b) a user defined graph1; (c) a user defined graph2; (d) the result of merging graph1 and graph2 by the CFDP algorithm.

structure of the user input, we use the user defined distance to replace the ideal edge length K.

$$f_a(j,i) = \frac{\|x_i - x_j\|^2}{udis(i,j)},\ \{i,j\} \in E^S;$$
$$f_r(j,i) = -\frac{(udis(i,j))^2}{\|x_i - x_j\|},\ i,j \in V^S$$

By replacing force constant K with the user defined distances $udis(i,j)$, if two nodes appear in more than one user input subgraph, the user defined distance will be the average value of all distances in these subgraphs. The CFDP algorithm therefore indirectly maintains multi-user input. When there is no edge between two nodes in a user subgraph, the force coefficient K is replaced by a user defined length only when calculating the repulsive force, since there is no attractive force between the nodes. To further enforce subgraph shapes, we add virtual edges to construct a complete subgraph. In Fig. 2(c) we have a graph with 5 nodes and 5 edges, in which a user edits nodes A, B and C. In this subgraph, there is only an edge between B and C, so we have to add two virtual edges AB and AC to form a complete subgraph. We can create virtual edges with the attractive force coefficients corresponding to the physical node distances in the subgraph placement.

Even though we add additional constraints to CFDP, for small graphs, its speed is still faster than FDP. This is because we first take each subgraph as one hyper node, and build a hyper graph with hyper nodes and edges which are external to the subgraphs. We then run the FDP algorithm. Finally, we unfold nodes represented by each hyper-node with the user defined topological structure, and run the CFDP algorithm. Since the nodes already maintain their user defined relative positions, the iteration process converges relatively quickly.

CFDP is able to incorporate user defined layouts of subgraphs, however, it does so in an indirect way, via the force coefficients. As shown in Fig. 3, the algorithm will produce a compromised version of layout if the same subgraph is edited by multiple users with different layouts. As a result, it does not always maintain the user input very well. We propose a more principled approach which explicitly incorporates multi-user input in the next subsection.

3.2 Laplacian Constrained Distance Embedding

Since we would like to maintain topological information of the user input subgraphs, if we record the relative position of every pair of nodes, these positions will be maintained even if the subgraph is subject to translation.

This idea is used in the scientific visualization literature [1, 25, 28, 32, 33], where the Laplacian vector is used for mesh editing and deformation. In this approach, the Laplace differential between each node and its neighbors is calculated by the formula:

$$\delta_i = v_i - \frac{1}{|N(i)|} \sum_{j \in N(i)} v_j,$$

and the technique seeks to preserve this differential for all nodes. However, this approach is not suitable for our purpose, because the Laplace differential is not preserved if the subgraph is rotated.

Instead, we seek to preserve the distance between each pair of nodes in the user subgraphs, because by the knowing pairwise distances among all nodes in a subgraph, we can exactly realize the layout of the subgraph.

We assume an initial layout of the whole graph. We then collect multi-user inputs for the layout of subgraphs. If two nodes appear in a subgraph, their ideal distance is defined by the subgraph layout. If two nodes appear in more than one subgraph, their ideal distance will be the average length in all the subgraphs involved. Otherwise, their ideal distance is the distance in the initial layout. Thus, we can define a stress energy function as

$$E = \sum_{i,j \in F} w_{i,j}(\|x_i - x_j\| - d_{ij})^2 + \alpha \sum_{i,j \in V^S} w_{i,j}(\|x_i - x_j\| - d_{ij})^2 \quad (1)$$

Here d_{ij} is the ideal distance of nodes i and j, $w_{i,j}$ is the weighting factor for the edges, and typically equals $1/d_{ij}^2$. Note that here the stress for the original graph is calculated over the set of node pairs F, which should be a superset of E. For example, F could be the set of edges in a triangulation of the nodes, plus the original edges. For simplicity, in this paper we take $F = E$. Likewise, the second summation in the stress energy could also be over a subset of node pairs, but for simplicity we take it to be over all pairs of nodes in the subgraphs.

Note that all ideal distances are physical distances based on the layout of the original and user subgraphs, not graph distances. By minimizing the stress energy, we are balancing two requirements: maintaining the original layout, and maintaining the user subgraph layouts. The magnitude of α controls the balance – a large α attempts to maintain the user layout as much as possible.

To minimize the stress function, we can use the Cauchy-Schwartz inequality to bound the terms in (1), and minimize the resulting quadratic function, as in Gansner et al. [20]. Here we use an easier and equivalent derivation following [18], by using the fact that the minimum for (1) is achieved when its gradient vanishes. Thus, taking the derivative of the function with respect to x_i and setting it to zero gives

$$\sum_{i,j \in E} 2w_{ij}(\|x_i - x_j\| - d_{ij}) \frac{x_i - x_j}{\|x_i - x_j\|} +$$
$$\alpha \sum_{i,j \in V^S} 2w_{ij}(\|x_i - x_j\| - d_{ij}) \frac{x_i - x_j}{\|x_i - x_j\|} = 0.$$

Moving the nonlinear terms to the right hand side, we get

$$\sum_{i,j \in E} w_{ij}(x_i - x_j) + \alpha \sum_{i,j \in V^S} w_{ij}(x_i - x_j)$$
$$= \sum_{i,j \in E} \frac{w_{ij} d_{ij}(x_i - x_j)}{\|x_i - x_j\|} + \alpha \sum_{i,j \in V^S} \frac{w_{ij} d_{ij}(x_i - x_j)}{\|x_i - x_j\|}.$$

In matrix form, this is

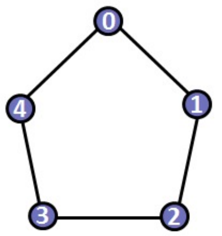
$$(L_w + \alpha L_w^S)x = (L_{w,d} + \alpha L_{w,d}^S)x, \quad (2)$$

where the weighted Laplacian matrices L_w and L_w^S are defined as

$$(L_w)_{ij} = \begin{cases} \sum_{\{i,l\} \in E} w_{il}, & \text{if } i = j \\ -w_{ij}, & \text{if } \{i,j\} \in E \\ 0, & \text{otherwise} \end{cases}$$

and

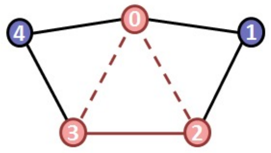
$$(L_w^S)_{ij} = \begin{cases} \sum_{l \in V^S} w_{il}, & \text{if } i = j \\ -w_{ij}, & \text{if } i,j \in V^S \\ 0, & \text{otherwise} \end{cases}$$



(a)

$$\begin{bmatrix} 2 & -1 & & & -1 \\ -1 & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & & -1 & 2 & -1 \\ -1 & & & -1 & 2 \end{bmatrix}$$

(b)



(c)

$$\begin{bmatrix} 6 & -1 & -2 & -2 & -1 \\ -1 & 2 & -1 & & \\ -2 & -1 & 6 & -3 & \\ -2 & & -3 & 6 & -1 \\ -1 & & & -1 & 2 \end{bmatrix}$$

(d)

Fig. 4. An Example of a graph and its associated L_w. Second row: when there is a subgraph and its associated matrix L_w. (a) a small graph; (b) Laplacian Matrix L_w; (c) the graph after editing, pink nodes are the subgraphs nodes; (d) Laplacian Matrix L_w plus a subgraph Laplacian matrix L_w^S.

and the matrices $L_{w,d}$ and $L^S_{w,d}$ have elements

$$(L_{w,d})_{ij} = \begin{cases} \sum_{\{i,l\}\in E} \frac{w_{il}d_{il}}{\|x_i-x_l\|}, & \text{if } i=j \\ -\frac{w_{ij}d_{ij}}{\|x_i-x_j\|}, & \text{if } \{i,j\}\in E \\ 0, & \text{otherwise} \end{cases}$$

and

$$(L^S_{w,d})_{ij} = \begin{cases} \sum_{l\in V^S} \frac{w_{il}d_{il}}{\|x_i-x_l\|}, & \text{if } i=j \\ -\frac{w_{ij}d_{ij}}{\|x_i-x_j\|}, & \text{if } i,j\in V^S \\ 0, & \text{otherwise} \end{cases}$$

We use the stress majorization technique [20] to solve the nonlinear system (2). It works as follows: starting from the initial layout of the whole graph as the $|V|\times 2$ matrix x in the right hand side of the linear system (2), we solve the linear system with the given right-hand-side, and insert the solution again into the right hand side. This process is repeated until the layout stabilizes.

Fig. 4 illustrates the two Laplacian matrices. The Laplacian matrix corresponding to the original layout is sparse, while the Laplacian matrix corresponding to a subgraph is dense. However, the dense part of the Laplacian is small, because subgraphs are relative small due to the inability of human users to handle large graphs. Note that for ease of presentation we assume so far that there is only one subgraph G^S, the case for multiple subgraphs follows the same approach, except that more small dense matrices are added to the Laplacian of the original graph.

As the ideal distances come from both the initial layout and the subgraphs, we first scale the subgraphs to the same proportion as its placement in the original layout.

We call this method Laplacian Constrained Distance Embedding (LCDE). Figure 5 compares LCDE with CFDP on a small example. It is seen that LCDE preserves user input layouts much better. This is because in CFDP, the user specified subgraph only affects the layout indirectly through the force coefficients. On the other hand, in LCDE, the user input layout is used explicitly in the energy minimization. By including all pairwise distances among nodes in the same subgraph, we ensure that the user input is mostly maintained.

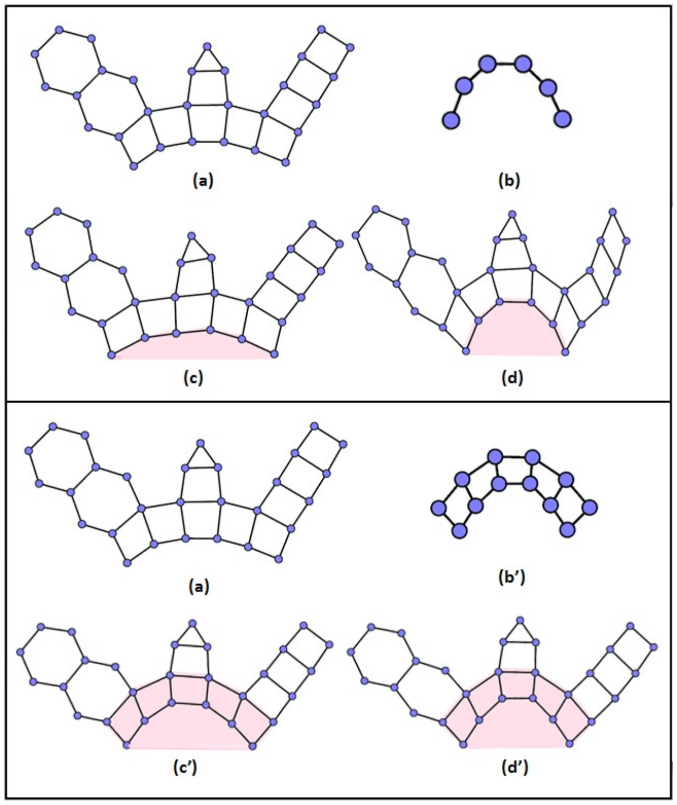


Fig. 5. Comparison of layouts by CFDP and LCDE. (a): An initial layout by FDP; (b) and (b'): user defined subgraphs with different number of nodes and topological shapes; (c) and (c'): layouts generated by CFDP; (d) and (d'): layouts generated by LCDE. Compared with layouts from CFDP, those from LCDE are seen to preserve the "hump" shape of the subgraphs much better.

3.3 Measuring Similarity

In order to have a more rigorous measure of (dis)similarity between layouts, we use Procrustes Statistic, which is a well known dissimilarity measure [5]. This measure attempts to find the best fit between two inputs, after rotation, translation, and scaling. It then takes the difference between the transformed inputs as their dissimilarity. Let one layout be $y_i,\ i=1,2,\ldots,|V|$, and another $x_i,\ i=1,2,\ldots,|V|$. We would like to find a translation vector b, a scaling value ρ and a rotation matrix T that minimizes:

$$\sum_{i=1}^{|V|} \|y_i - (\rho T x_i + b)\|^2. \tag{3}$$

The solution to this problem is

$$T = (XY^TYX^T)^{1/2}(YX^T)^{-1},\ \rho = \frac{tr((XY^TYX^T)^{1/2})}{tr(XX^T)}, \tag{4}$$

where X is the $2\times|V|$ matrix of x_i's. The translation vector is $b = \frac{1}{|V|}(\sum_{i=1}^{|V|} y_i - \rho T(\sum_{i=1}^{|V|} x_i))$. The minimal value of (3),

$$P(X,Y) = 1 - \frac{(tr((X^TYY^TX)^{1/2}))^2}{tr(X^TX)tr(Y^TY)} \tag{5}$$

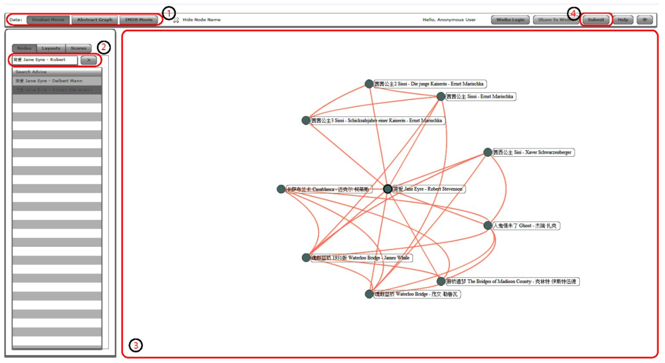
is known as the Procrustes Statistic.

We use $1-P(X,Y)$ to measure the similarity of two layouts: a value of 1 indicates that two layouts are identical, and a value of 0 means that they are completely different. This is used for measuring how well the user input subgraphs are preserved in the merged layout, as well as in Section 4.1 as part of a scoring system for measuring user input.

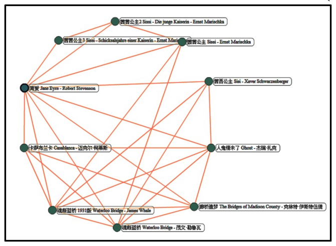
4 Case Studies

We designed and implemented an interactive system[1] to leverage users' drawing capabilities. The system provides a subgraph search

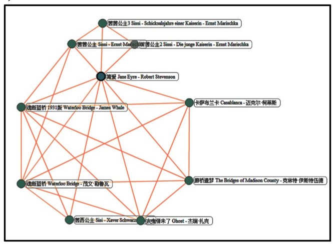
[1]http://vis.pku.edu.cn/graphviz/GraphViz.html



(a)



(b)



(c)

Fig. 6. Interface for editing graph layout. (a) The interface provided to the users. ① Panel to select data sets; ② search box to identify center node to draw; ③ graph layout editing canvas; ④ submit button; (b) a user defined graph layout; (c) alternative graph layout by another user.

function, by which a user can choose a subgraph, containing a few nodes up to the whole graph, and adjust its layout. Fig. 6 shows our interactive graph visualization and manipulation system. The user can search a node by name, or just double click the item listed below the search box, then a subgraph, containing the searched node and its direct neighbor nodes, appears in the window on the right hand side. The users can keep double clicking on the nodes in the graph window to extend the subgraph to include additional nodes connected to the clicked nodes. In this study, we are interested in allowing the user to generate the layout completely unassisted. Thus to prevent users from being influenced by any layout, when they search a node, we only put it in the center of the screen and its neighbors are aligned in a circle (Fig. 6(a)). The user can adjust the layout by dragging the nodes on screen with the mouse. Then, when satisfied with the change (Fig. 6 (b) and (c)), the user can click on the 'submit' button on the upper right corner. The user defined subgraph layout is uploaded to our system via the Internet. In this way, we can collect many user defined subgraphs for our case study.

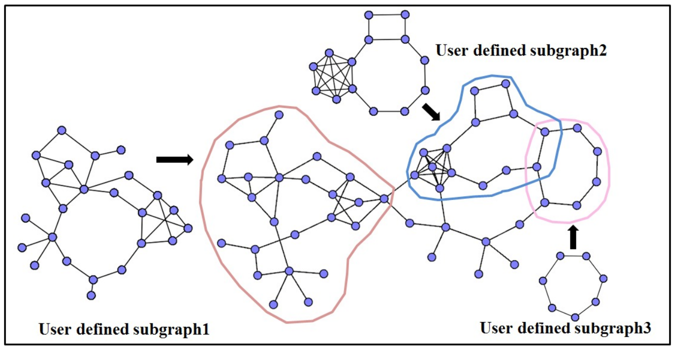
In this section we apply our algorithms to four data sets: a small abstract graph, some graphs representing biochemical metabolic pathways, a co-authorship network, and a movie graph.

4.1 Small abstract graph

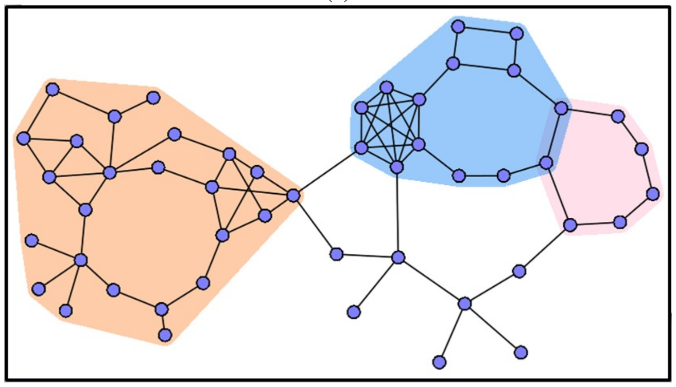
The first experimental data set is a small abstract graph with 50 nodes and 124 edges, taken from a graph layout comparison study by Dwyer et al. [8]. In that work, the authors compared the user-generated layout to automatically generated layouts. They asked users to give an aesthetic layout of the graph. This proved to be a very difficult task for most people, since they were not trained to handle a graph of even 50 nodes. As a result the majority of user generated layouts were found to be inferior to those drawn by a force-directed placement algorithm.

In our study, there are 18 users, among them 13 users are from our visualization group: one teacher and 8 graduate students with visualization background, and 4 undergraduate students with computer science background. The remaining 5 users contributed anonymously after our online announcement.

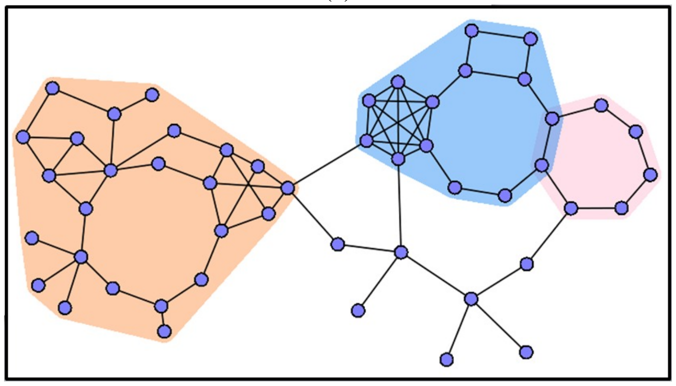
We allow users to select many small subgraphs. We found that when



(a)



(b)



(c)

Fig. 7. CFDP and LCDE algorithms result with 3 user input subgraphs. (a) Initial layout and 3 user defined subgraphs; (b) CFDP layout algorithm result; (c) LCDE layout algorithm result.

there are a few input subgraphs, our algorithms are able to maintain users' input subgraphs well. As seen in Fig. 7(a), we first generated an initial layout by a force-directed layout algorithm. A user searched for three subgraphs and gave them each a nice layout using our interactive system. Two nodes appeared both in subgraph2 and subgraph3. Fig. 7 (b and c) are the result of merging the three user subgraphs using CFDP and LCDE algorithm. For subgraph1, the two algorithms both maintain its shape very strictly, which is not surprising since subgraph1 has only one node connecting to the rest of the graph, so the rest of the graph has less effect on it. For subgraph2 and subgraph3, the CFDP method maintains the general outline with some distortion in the empty circles, this is because subgraph2 and subgraph3 have a common edge with two nodes (conflict nodes), the ideal length is the average of the distance in each subgraph. Due to this change, a perfect circle no longer balances all the forces, so the circles have to deform somewhat to get to a new balanced state. However, the LCDE algorithm keeps the shape of the circles much better, because although there are two ideal lengths of the common edge that appear in the stress function, there are many other virtual edges that act as a scaffolding to make the circles very rigid, and the shape of the circles are mostly maintained.

From our graph visualization system, we collected 45 subgraphs of the abstract graph data set. On inspection, some of these are poorly laid out. Since we open the system to the public, some users did not have graph layout experience, and had to spend time exploring the interaction system and accumulating experience of making a nice layout. Consequently, it is not unusual for users to submit a poor layout, and we have to filter out such subgraphs. To define what kind of subgraphs could be included, we establish a scoring system. First, we give each subgraph a layout score by comparing it with a layout generated by FDP algorithm using Procrustes Statistic (see Section 3.3).

In addition to the similarity comparison, we would also like to see few edge crossings, which is not always achieved by the FDP layout. We rank subgraphs based on each measure, then add the rankings together to derive a score.

Using our scoring system, we choose the top 23 subgraphs as inputs to our algorithms. Wehn we compare the CFDP and LCDE algorithms' layout with the FDP algorithm, our algorithms are found to give a good graph layout, and at the same time, each user subgraph is found to contribute to the whole layout.

In Fig. 8(a) we list 15 reasonable subgraph layouts collected from many different users. We then give three layouts achieved by the FDP algorithm (Fig. 8(b)), the CFDP algorithm (Fig. 8(c))and the LCDE algorithm(Fig. 8(d)). In Fig. 8(c) and (d) the sizes of the pink nodes show how many time the nodes have been edited, larger nodes have more edits. As can be seen, CFDP and LCDE produce good layouts. In the part of the graph where there are more user inputs (left of Fig. 8 (c and d)), nodes and edges are evenly spaced out, unlike the FDP layout where one node is very close to two edges. However, the drawings also reveal a limitation of our algorithms. There is a subgraph shown in $SG10-12$, where six nodes form a clique. Even though many users place the nodes in a circle, in both CFDP and LCDE layouts, this clique is not laid-out in a circular fashion. The reason is that different users align these nodes into a circle in different orders, so the distance of a given pair of nodes varies. Since we only have a few of these layouts, even the average edge length varies. We believe with more user inputs, this variation may eventually even out. However, in the case of limited user input, it is difficult to keep a reasonable shape for these cliques. Perhaps one possible remedy is to identify cliques, and always give the user a subgraph where nodes in a clique are in a pre-arranged circle. We intend to investigate this further in the future.

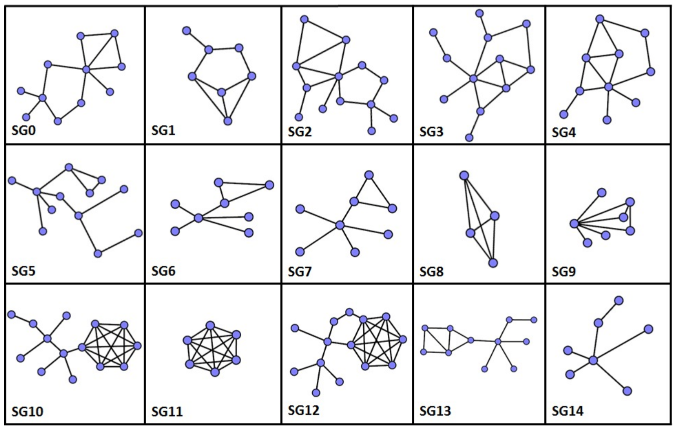
4.2 Biochemical metabolic pathway

We use biochemical metabolic pathway data to demonstrate the proposed graph layout methods in a real-world scenerio. Small metabolic pathways are often published with separated distinct canonical layouts. Though it is meaningful and possible to merge many metabolic pathways into a bigger pathway, it is difficult and tedious to do it by hand. No existing method can automatically achieve this. We use our proposed method in such a scenerio as a first step to such an automatic merging algorithm.

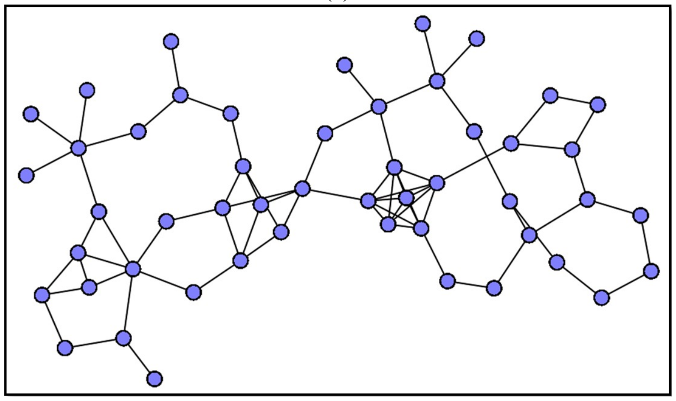
A small set of classical metabolic pathways are used in the experiment: the photosynthesis pathways, cellular respiration pathways and several other related metabolic processes. Photosynthesis is a process used by plants to capture the sun's energy and store it in biochemical form. Cellular respiration is a set of processes that release stored biochemical energy. They are fundamental metabolic pathways widely accessible in biochemical textbooks. There are usually canonical ways to illustrate them, such as in the form of a cycle or a stick. We collected several illustrative images of these pathways from Wikipedia pages[2,3] and manually transformed them into metabolic pathway graphs. Since they have shared chemicals, these pathway graphs can be put together and joined to a bigger connected graph. We also added a few other small pathways related to these chemicals. Then we manually obtained coordinates of chemical nodes from the same

[2]http://en.wikipedia.org/wiki/Citric_acid_cycle

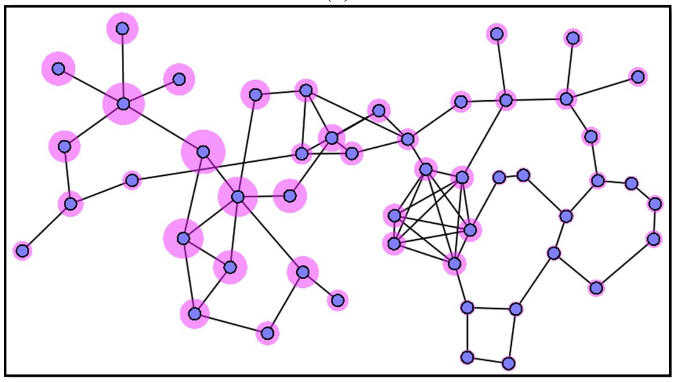
[3]http://en.wikipedia.org/wiki/Cellular_respiration



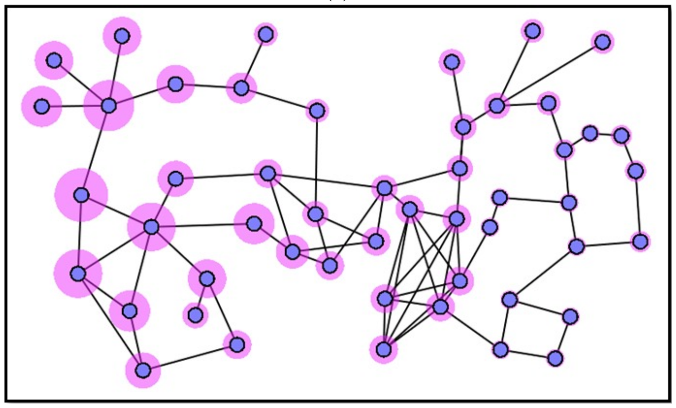
(a)



(b)

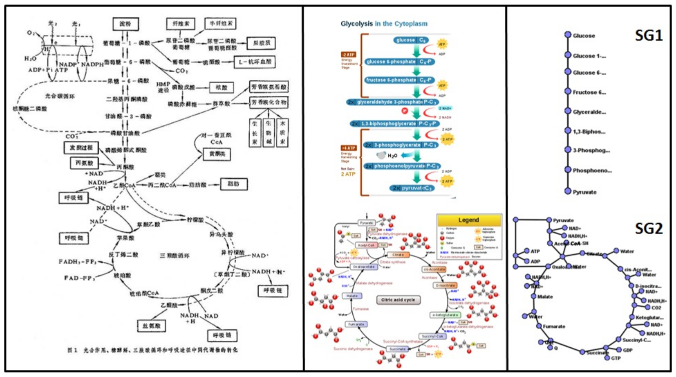


(c)

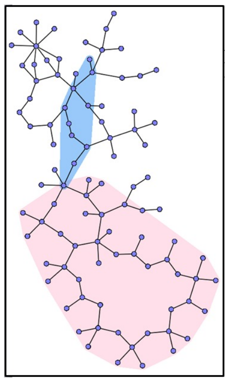


(d)

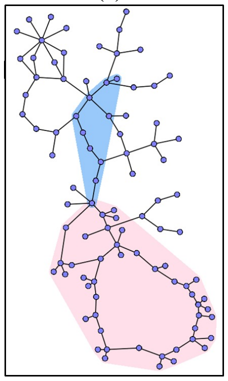
Fig. 8. (a) The top 15 user input subgraphs from all 45 collected subgraphs. The three layouts below are achieved by (b) FDP, (c) CFDP and (d) LCDE algorithms, respectively.



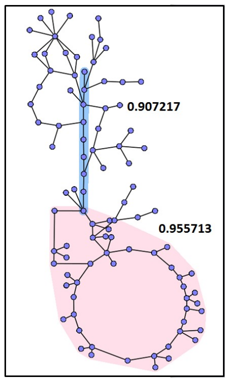
(a)



(b)



(c)



(d)

Fig. 9. Biochemical metabolic pathway. (a) metabolic pathway from a textbook (l), subgraphs from wikipedia (m), the input subgraphs (r). (b) a layout by a force-directed layout algorithm. (c) a layout by the CFD algorithm. (d) a layout by the LCDE algorithm, with numbers indicating similarity scores.

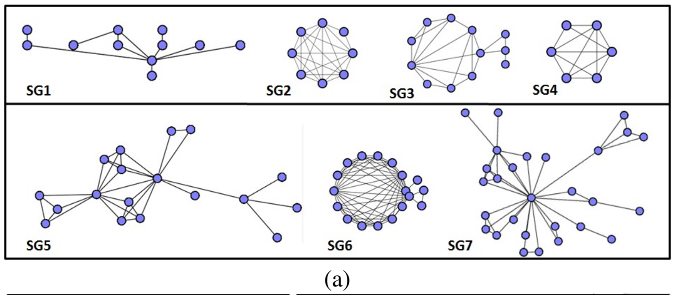
illustrative images from Wikipedia and transformed them into subgraph layouts. Using the whole graph and the subgraph layout data, we are able to calculate a global layout for the whole merged metabolic pathway. The merging layouts are shown in Fig. 9 (a), (b) and (c) after application of FDP, CFDP and LCDE, respectively. The background shadowed hulls in blue and pink indicate subgraphs SG1 and SG2 from Fig. 9 (a), respectively. We easily recognize that the symbolic stick of Glycolysis process and cycles of Citric Acid Cycle are kept.

4.3 Co-authorship network

For the InfoVis 2004 contest, the dataset contains complete metadata for all the papers of 8 years of InfoVis Conference and their references. The metadata includes publication titles, authors, keywords, abstract, references and links to original papers when available in the ACM Digital Library. From this dataset, we extract a co-authorship network, and present its largest connected component, which contains 148 nodes and 349 edges. We produce each subgraph layout by automatic algorithms, SG1 is handled by a hierarchical layout algorithm, SG2-SG4 and SG6 by a circular layout algorithm, SG5 and SG7 by a force-directed layout algorithm. We selected these 7 subgraphs by hand, and left some other nodes in the co-authorship network unedited. From Fig. 10, we can see that layouts of different styles are merged naturally in one layout and each subgraph layout is kept very well.

4.4 Movie graph

We crawled the Douban website for movies and their relations. Douban is a Chinese movie website that provides movie summaries and comments. Douban has a movie recommendation mechanism. This system recommends movies related to any given movie, based on the fact that people who like movie A also like movie B, or C and so on. We collect this relational information to build a graph, where



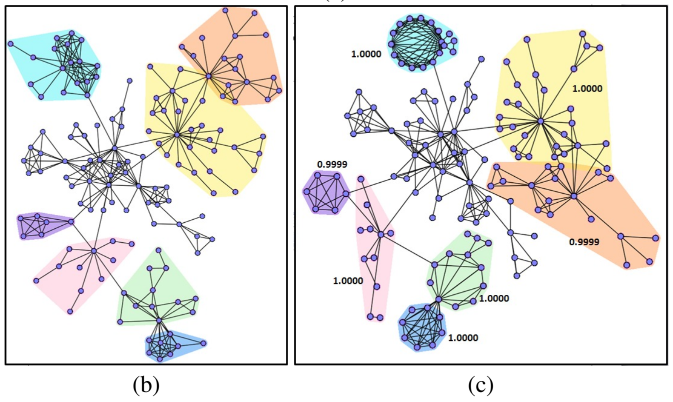


Fig. 10. Co-authorship network. (a) the subgraph layouts produced by automatic algorithms. (b) a layout by force-directed layout algorithm. (c) is a layout by the LCDE algorithm with subgraph constrains. The numbers around the subgraphs are similarity scores to their input layout.

nodes are movies. If movie B is in the recommendation list of movie A and A is also in the recommendation list of movie B, we connect A and B with an edge.

We made an announcement on Sina Weibo, the Chinese equivalent of Twitter, inviting people to use our graph visualization tool by selecting and layouting subgraphs of the movie graph. Eventually, 131 users submitted subgraph layouts; most of them were students, teachers or others interested in movies or visualization. In our web-based system, users can search a movie by its name, and other movies related to it will show up. Then a user can make a layout for these movies if he is familiar with them. We also provide movie summaries and links to help users to get to know the movies. Users can then upload their layouts, and can also share their results with other users.

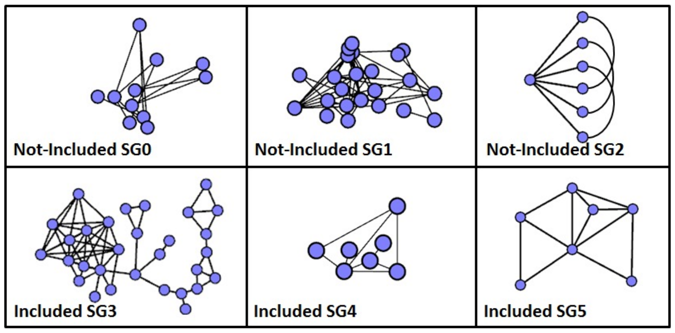


Fig. 11. Examples both not-included and included subgraphs, as determined by our score system.

We collected 178 subgraphs from this system, they cover 772 nodes and 1349 edges. We filtered out about 50% of the layouts with low scores. Fig. 11 lists three examples of excluded subgraphs with very low scores and three included subgraphs with high scores.

After filtering, we achieved 92 reasonable subgraphs, which form a disconnected super-graph. Then we used the CFDP and LCDE algorithms to combine these subgraphs together and get the results in Fig. 12. Fig. 12 (a) shows some of the user input subgraphs; (b) shows a layout generated by the FDP algorithm, (c) and (d) show the layout generated by CFDP and LCDE; (e) shows the layout of corresponding input subgraphs with their movie names for the 3 different layout methods. From the figure, we see that FDP algorithm generated a reasonable layout with a number of clusters. People can see clusters clearly, but it is difficult to see the inner structure of cluster since the attractive forces pull them too close. The CFDP and LCDE algorithms are both able to achieve a nice layout, as well as maintain user inputs. For example, subgraphs SG1-SG4 are well laided-out by CFDP and LCDE. SG2 is a small cluster disconnected from other nodes, while the other 3 subgraphs are part of user defined subgraphs, for example, SG1 in (e) is the left part of SG1 in (a), SG4 in (e) is the right part of SG4 in (a), the left part of SG1 and SG4 in (a) are also edited by many other users, so their structures cannot be seen clearly. In Fig. 12 (e), comparing the layouts of SG2 by the three algorithms, we see that both CFDP and LCDE placed sequels of the same movie close to each other, faithfully preserving the input, since in the mind of the user, these sequels are more similar to each other than to other movies.

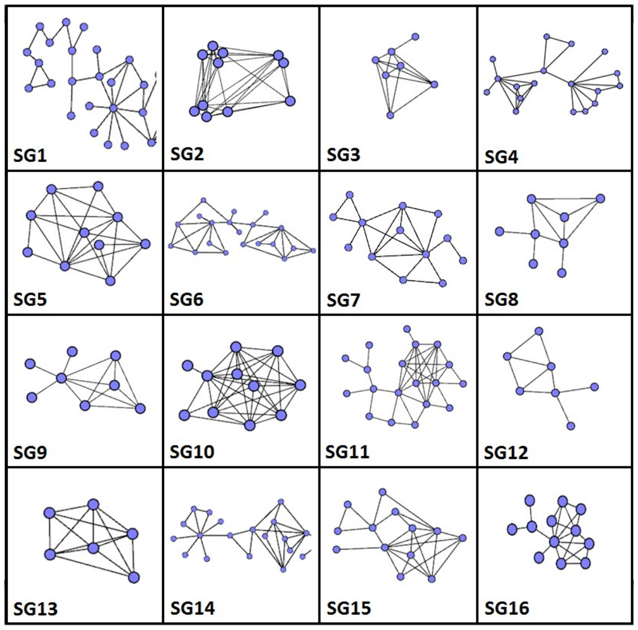
Our system provides a platform to help users visually and interactively express their understanding and feeling of movies and their relationship. The CFDP and LCDE algorithms are able to maintain user input layouts, and enable us to collect and integrate users' knowledge to form a knowledge database with visual representations.

5 Discussion

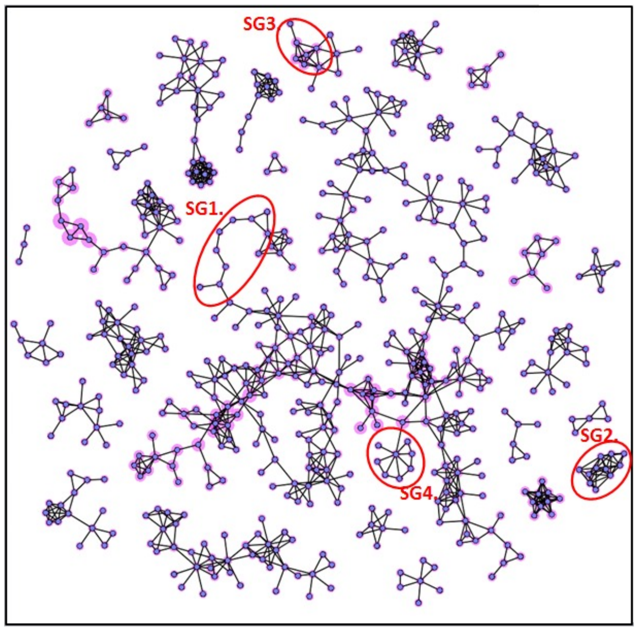
From the four case studies, we see that both CFDP and LCDE algorithms can maintain, to a large extent, the topological structure of users' input subgraphs. CFDP maintains user input indirectly through force coefficients. It is found to be good at maintaining the structures of relatively independent subgraphs, for which forces within a subgraph are less affected by those from other nodes in the whole graph. On the other hand, the LCDE algorithm preserves the topological structure of user subgraphs directly through distance preservation, and is found to preserve subgraph topological structures very well, even when some subgraphs overlap.

Whichever method is used, a key ingredient of our approach is to leverage humans' innate sense of order, symmetry and aesthetics in drawing small subgraphs, and to utilize automated algorithms to incorporate these desired characters. In cases where layouts from users are good and consistent, we have achieved satisfactory results. Furthermore, in some application areas, such as biology and chemistry, scientists are used to seeing illustrations of certain biochemical reactions in a canonical way. However, even though each reaction graph may be small, a whole biological pathway network can be very large, making it hard for scientists to construct an overview of the whole graph while maintaining the shape of each small graph. Our algorithms are shown to be able to achieve this difficult task successfully. Another application we demonstrated is to combine outcomes from different automatic layout algorithms into one combined layout. For a graph with many clusters, a dense cluster may be arranged more suitably by a circular layout, while a sparse cluster benefits from a hierarchical layout. Taking advantage of the strength of each layout algorithm helped us in producing a better layout, which should allow users to understand and explore the graph more easily.

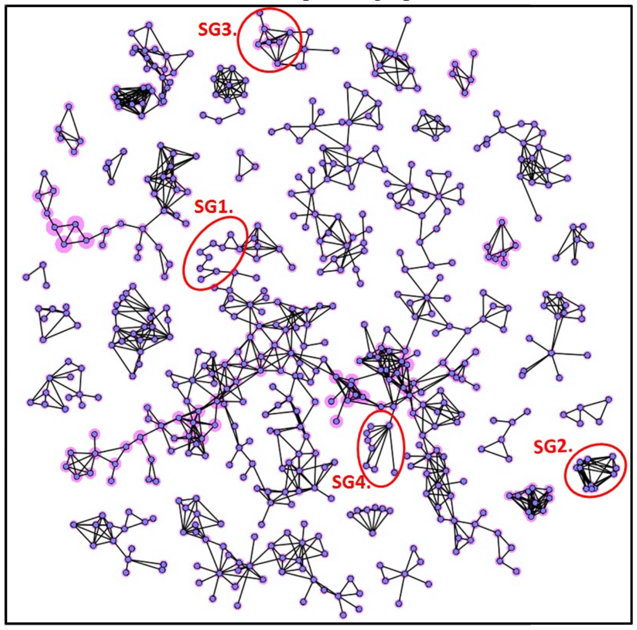
In our study, we have implemented a system for collecting many users' inputs. While the system does not support a collaborative multi-user input function, with which many users can work together in one interface at the same time, our web-based user input system does allow a user to edit his own subgraph and then submit it, knowing that his subgraph is eventually merged with other inputs to give a layout of the whole graph. From observing user inputs, we find that the the quality of the input subgraphs can be inconsistent. This represents a challenge to our idea of combining multiple good layouts into a whole. We proposed a scoring system to help weed out poor input. Nevertheless, we would like to investigate better ways to gauge the quality of user input. In addition, as we collect more and more user subgraphs,



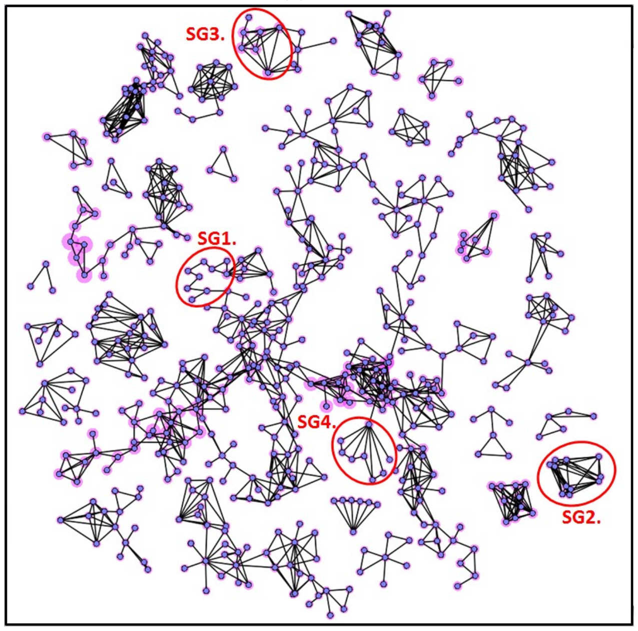
(a) User input subgraphs



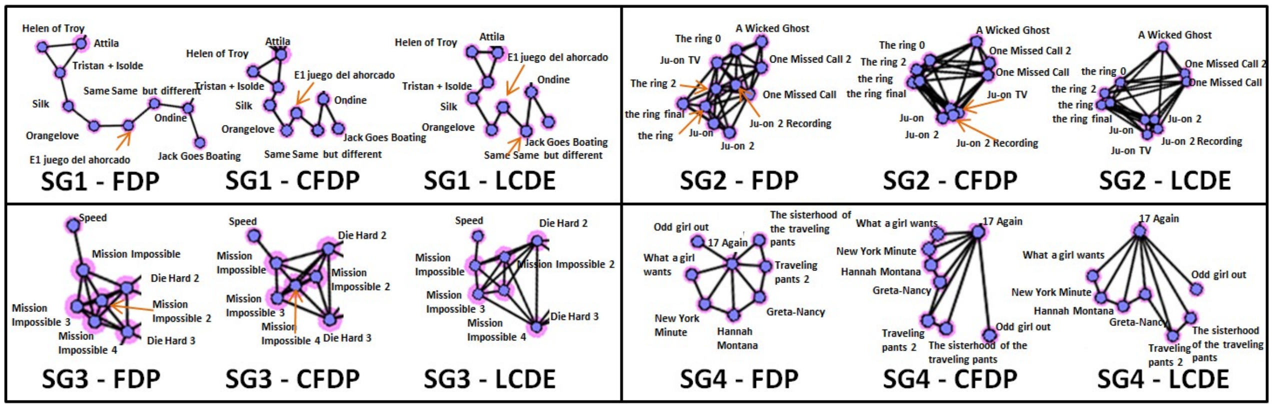
(b) FDP



(c) CFDP



(d) LCDE



(e) Zoom in subgraphs comparison with different algorithms

Fig. 12. A movie graph with 588 nodes and 1167 edges. (a) Collection of 16 subgraphs among overall 92 subgraph layouts submitted by the users; (b) layout generated by FDP algorithm; (c) layout generated by CFDP algorithm; (d) layout generated by LCDE algorithm; (e) comparison of subgraph layout in the large graph generated with different algorithms.

we can build a subgraph layout database. The most direct way to utilize this database is to apply a user defined layouts of a subgraph to all isomorphic subgraphs in a new graph. Finding isomorphic subgraphs, however, is an NP-hard problem. However, we can learn, for example layout styles, from the database. Even though two subgraphs are not exactly isomorphic, if we can extract same characteristics from them, then one graph can learn from the layout style of the other. This is an interesting direction to pursue for the future.

6 Conclusion and Future Work

In this paper, we proposed two new methods to combine many users' input subgraphs into a consistent whole. We believe that human beings have a natural sense of order and symmetry, and are able to lay out small graphs well. The algorithms we proposed represent our attempt to harness the abilities of the users. Furthermore, users of graph drawing systems understand their areas of application better, and often like to see certain parts of the graph drawn in a particular way to conform to the conventions of the field. Our algorithms offer a general way to incorporate users' input, and the interactive system we implemented provides a convenient tool for the users to specify their preference and constraints.

Our approach has its limitations. When the input is of low quality, or in conflict with each other (as in the case of cliques seen in Section 4.1), we would like to find a better way to exclude poor layouts, resolve the conflicts, and understand users' intentions. Another area of interesting research is in extending our interactive system into an online game, where inputs are scored based on quality measures of the layouts, and users gain points for better layouts. This would encourage mass participations, and with better and more scalable merging algorithms, could provide a powerful layout engine for complex networks.

Finally, our approach is not restricted to graph layout. The same methodology we proposed can be used for crowd-sourcing multidimensional data analysis, where users can help find low-dimensional embeddings for items (e.g. movies) with multiple attributes.

Acknowledgments

The authors wish to thank the anonymous reviewers for their valuable comments.

References

[1] M. Alexa. Mesh editing based on discrete laplace and poisson models. In *ACM SIGGRAPH 2006 Courses*, pages 51–59, 2006.

[2] D. Archambault, T. Munzner, and D. Auber. Topolayout: Multi-level graph layout by topological features. *IEEE Trans. Vis. Comput. Graph.*, 13(2):305–317, 2007.

[3] D. Auber and Y. Chriricota. Improved efficiency of spring embedders: Taking advantage of gpu programming. In *Visualization, Imaging, and Image Processing - 2007*, pages 169–175, 2007.

[4] M. Baur and U. Brandes. Multi-circular layout of micro/macro graphs. In *Proceedings of the 14th international conference on Graph drawing*, volume 4875, pages 255–267, 2007.

[5] T. F. Cox and M. A. A. Cox. *Multidimensional Scaling*. Chapman and Hall/CRC, 2000.

[6] U. Dogrusöz, B. Madden, and P. Madden. Circular layout in the graph layout toolkit. In *Proceedings of the Symposium on Graph Drawing*, GD '96, pages 92–100, 1996.

[7] T. Dwyer. Scalable, versatile and simple constrained graph layout. *Computer Graphics Forum*, 28(3):991–998, 2009.

[8] T. Dwyer, B. L. Danyel Fisher, K. Inkpen Quinn, P. Isenberg, G. Robertson, and C. North. A comparison of user-generated and automatic graph layouts. In *IEEE Symposium on Information Visualization*, pages 961 – 968, 2009.

[9] T. Dwyer and Y. Koren. Dig-cola: Directed graph layout through constrained energy minimization. In *Proceedings of the 2005 IEEE Symposium on Information Visualization, (Infovis'05)*, pages 65–72, 2005.

[10] T. Dwyer, Y. Koren, and K. Marriott. Constrained graph layout by stress majorization and gradient projection. *Discrete Mathematics*, 309(7):1895–1908, 2009.

[11] T. Dwyer, K. Marriott, and M. Wybrow. Dunnart: A constraint-based network diagram authoring tool. In *Proceedings of the 16th international conference on Graph Drawing*, GD'08, pages 420–431, 2008.

[12] T. Dwyer and G. Robertson. Layout with circular and other non-linear constraints using procrustes projection. In *Proceedings of the 17th international conference on Graph Drawing*, volume 5849/2010 of *GD'09*, pages 393–404, 2009.

[13] P. Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42:149–160, 1984.

[14] P. Eades, Q. Feng, X. Lin, and H. Nagamochi. Straight-line drawing algorithms for hierarchical graphs and clustered graphs. In *Algorithmica*, pages 113–128, 1996.

[15] M. Eiglsperger and M. Kaufmann. Fast compaction for orthogonal drawings with vertices of prescribed size. In *the 9th International Symposium on Graph Drawing*, GD '01, pages 124–138, 2001.

[16] Y. Frishman and A. Tal. Multi-level graph layout on the gpu. *IEEE Trans. Vis. Comput. Graph.*, 13(6):1310–1319, November 2007.

[17] T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software: Practice and Experience*, 21:1129–1164, 1991.

[18] E. R. Gansner, Y. Hu, and S. North. A maxent-stress model for graph layout. In *Proceedings of IEEE Pacific Visualization Symposium*, pages 73–80, 2012.

[19] E. R. Gansner and Y. Koren. Improved circular layouts. In *Proceedings of the 14th international conference on Graph drawing*, GD'06, pages 386–398, 2006.

[20] E. R. Gansner, Y. Koren, and S. North. Graph drawing by stress majorization. In *Proceedings of the 12th international conference on Graph Drawing*, GD'04, pages 239–250, 2004.

[21] S. Hachul and M. Jünger. Drawing large graphs with a potential field based multilevel algorithm. In *Proceedings of the 12th international conference on Graph Drawing*, GD'04, pages 285–295, 2004.

[22] Y. Hu. Efficient and high quality force-directed graph drawing. *Mathematica Journal*, 10:37–71, 2005.

[23] S. Ingram, T. Munzner, and M. Olano. Glimmer: Multilevel mds on the gpu. *IEEE Trans. Vis. Comput. Graph.*, 15(2):249–261, 2009.

[24] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31:7–15, 1989.

[25] Y. Li, E. Zhang, Y. Kobayashi, and P. Wonka. Editing operations for irregular vertices in triangle meshes. *ACM Trans. Graph.*, 29(6):153:1–153:12, 2010.

[26] J. Marks, K. Ryall, and S. Shieber. An interactive constraint-based system for drawing graphs. In *Proceedings of the 10th annual ACM symposium*

on User interface software and technology, pages 97–104, 1997.

[27] C. Papadopoulos and C. Voglis. Drawing graphs using modular decomposition. In *Proceedings of the 13th international conference on Graph Drawing*, GD'05, pages 343–354, 2005.

[28] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, SGP '04, pages 175–184, 2004.

[29] R. Tamassia. Constraints in graph drawing algorithms. *Constraints*, 3(1):87–120, 1998.

[30] F. van Ham and B. Rogowitz. Perceptual organization in user-generated graph layouts. *IEEE Trans. Vis. Comput. Graph.*, 14(6):1333–1339, 2008.

[31] C. Walshaw. A multilevel algorithm for force-directed graph drawing. In *Proceedings of the 8th International Symposium on Graph Drawing*, GD '00, pages 171–182, 2000.

[32] Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, and H.-Y. Shum. Mesh editing with poisson-based gradient field manipulation. *ACM Trans. Graph.*, 23(3):644–651, 2004.

[33] D. Zorin, P. Schröder, and W. Sweldens. Interactive multiresolution mesh editing. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '97, pages 259–268, 1997.