Drawing Graphs Within Graphs: A Contribution to the Graph Drawing Contest 2003

Daniel Gmach Paul Holleis Thomas Zimmermann

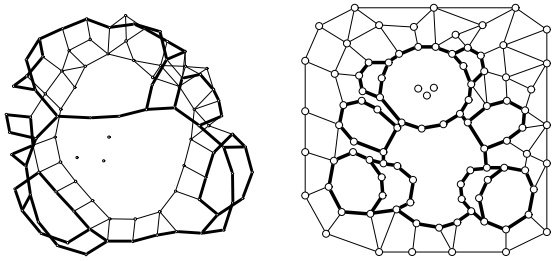
Computer Science, University of Passau
E-mail: {gmach,holleis,zimmerth}@fmi.uni-passau.de

Abstract

This paper is a contribution to the Graph Drawing Contest 2003. Frequently, the drawing of subgraphs is underestimated and subgraphs are simply emphasized using different colors or line styles. In this paper, we present an approach for drawing graphs within graphs that layouts the subgraphs first and therefore increases their locality. In two case studies, we demonstrate how this approach can be used to visualize connections between subgraphs or to highlight many similar subgraphs.

1. Introduction

Graphs play an important role in many fields of research and economy, as well as in day-to-day life. Often graphs are huge, complex buildings that are difficult to understand. A powerful tool for reducing complexity and emphasizing particular aspects are subgraphs. Still the handling of subgraphs is not a prominent topic for graph drawing: Usually, subgraphs are not considered for the layout of the entire graph (even if they are known in advance), and only highlighted afterwards. For simple subgraphs this approach suffices, but for complex subgraphs its structure is lost. For example take Figure 1 which contains a teddy within a web. If the entire graph is layouted first, teddy loses its structure and is hardly recognizable anymore. But, if you bear teddy in mind for layouting, its structure can be preserved.
So the idea is to layout the subgraph before the entire graph. Thus, subgraphs are easier to spot as their locality is increased. For a single subgraph this approach is straightforward, therefore we concentrate on multiple subgraphs that are known in advance. The remainder of this paper is organized as follows. In Section 2 we formalize our approach in general. Section 3 presents a specialized layout algorithm for emphasizing connections between subgraphs. In Section 4 we apply the general approach to many similar subgraphs. We close with a discussion of our approach and future work in Section 5.



(a) Graph first (b) Teddy first

Figure 1. Can You Spot Teddy? Teddy is caught in a web. (a) Without knowledge of teddy, teddy is distorted by the spring embedder. (b) If teddy is layouted first, the web can be arranged around him.

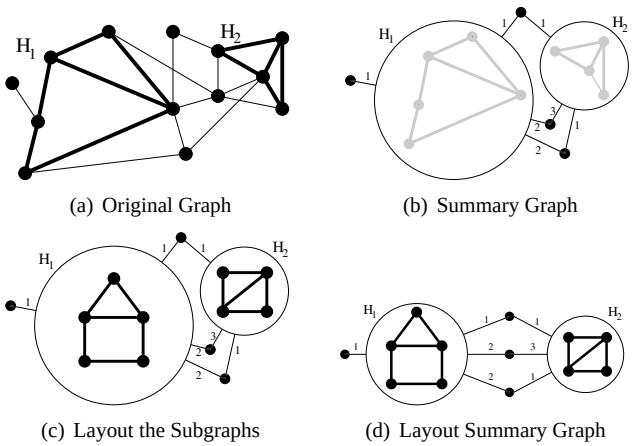
2. General Approach

First, we present a generic framework for emphasizing multiple subgraphs within one graph, so we can deal with different questions and different graphs. Later on we will specialize this framework and introduce applications for it.

Create a Summary Graph. In the first step, we create a graph in which each subgraph H_i is represented as a single vertex. We refer to this graph as the *summary graph* $G_S = (V_S, E_S)$. The mapping between the original graph G and G_S is obtained by two functions $f_V : V \to V_S$ and $f_E : E \to E_S$. Note, that f_V and f_E may also be used to combine vertices and edges that are not part of a subgraph.

Layout the Subgraphs and the Summary Graph. After creating the summary graph, we layout the subgraphs first and then the summary graph. All graphs are layouted independently, so different layout algorithms may be used. If possible, the layout algorithm for the summary graph should use information about the size of the subgraphs.

Apply the Layout to the Original Graph. Next, we combine the layouts of the subgraphs and the summary



(a) Original Graph (b) Summary Graph

(c) Layout the Subgraphs (d) Layout Summary Graph

Figure 2. Phases of the General Approach

graph to a layout for the original graph. This is done by replacing vertices that represent a subgraph by the layout of the respective subgraph. The mapping between the original graph and the summary graph can be restored using f_V^{-1} and f_E^{-1}. [1]

Optimize the Layout. The resulting layout is not optimal. So, in the final step fine-tuning we reduce the number of crossings and the required space of the layout.

In Figure 2 we show an example for our approach. First the original graph 2(a) is transformed into the summary graph 2(b). In the summary graph we label edges with the number of edges they represent. Next the subgraphs are layouted 2(c). Finally the summary graph is layouted and the subgraphs are embedded in its layout 2(d). Note that the size of the subgraphs was considered for the layout of the summary graph.

There are two prerequisites or limitations for this framework. First, the subgraphs have to be known before layouting, and second, the subgraphs have to be disjoint.

Subgraphs are Not Known in Advance. In this case, we create an optimized layout *on-the-fly* and use graph animation to transform the initial layout into the optimized one. Additionally, we can *guess* interesting or likely subgraphs, on which we base our initial layout. For "guessing" good graphs, we can use network motifs and the techniques described in [**?**, **?**].

Non-Disjoint Subgraphs are not really a problem for our approach. There are three possible solutions: First, we can *combine* overlapping subgraphs into a new subgraph. Second, we can *partition* those subgraphs in overlapping and non-overlapping parts and consider each of them as a separate subgraph. And finally, we can *duplicate* overlapping vertices, so that every subgraph remains unchanged. Unfortunately this requires duplication of edges as well, which complicates the layout process. [2]

[1] Note that f_V^{-1} and f_E^{-1} are relations and not functions.

3. Case Study: Connection Sets

Given multiple subgraphs, an interesting aspect is the relationship between the subgraphs. We present a layout algorithm that emphasizes these relations by identifying *connection sets*. A connection set for some subgraphs contains all vertices that are reachable from all of those subgraphs without visiting vertices of any subgraph.

Definition Let $G = (V, E)$ be an undirected graph, and $H_1 = (V_1, E_1), \ldots, H_n = (V_n, E_n)$ be subgraphs of G. The remainder sets are $V_R = V - \bigcup V_i$ and $E_R = E - \bigcup E_i$. A *connection set* is defined as follows:

$$C_{\{i\}} = \{\, v \mid u \in V_i,\ \{w_1, \ldots, w_k, v\} \subset V_R,\ \exists \text{ path } u \to w_1 \to \cdots \to w_k \to v\}$$
$$C_I = \bigcap_{i \in I} C_{\{i\}}$$

□

Note, that the definition above allows vertices to be in more than one connection set. For example, if a vertex connects subgraphs H_1, H_2 and H_3 it is contained in seven connection sets $C_{\{1\}}$, $C_{\{2\}}$, $C_{\{3\}}$, $C_{\{1,2\}}$, $C_{\{1,3\}}$, $C_{\{2,3\}}$, and $C_{\{1,2,3\}}$. For some graphs it is more intuitive, if such a vertex is only contained in one set. Therefore we define *exclusive connection sets* in which the vertex is only contained in one set $C^e_{\{1,2,3\}}$.

Definition Let C_I be connection sets of an undirected graph G. We define an *exclusive connection set* as:

$$C_I^e = C_I - \bigcup_{i \in \{1,\ldots,n\},\ i \notin I} C_{I \cup \{i\}}$$

□

We can now use (exclusive) connection sets to create a layout that emphasizes subgraphs and their relationships to each other. For this we define the summary graph first.

$$f_V(v) = \begin{cases} v_i^H & \text{if } v \in V_i \\ v & \text{else} \end{cases}$$

$$f_E\big(\{u_1, u_2\}\big) = \big\{f_V(u_1), f_V(u_2)\big\} \text{ for } u_1, u_2 \in V$$

[2] Note that duplication is not considered in our f_V and f_E model. To deal with duplication f_V and f_E have to be relations instead of functions

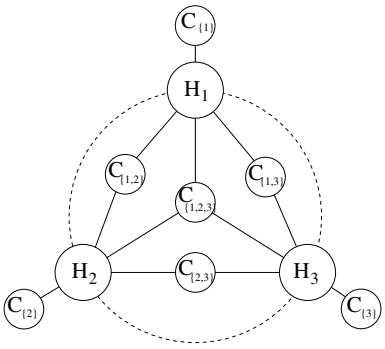


Figure 3. Placement of Connection Sets The connection sets are placed between their subgraphs.

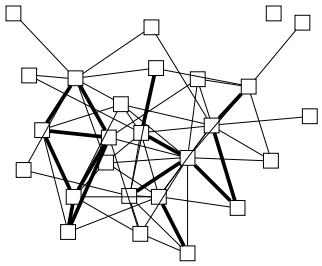
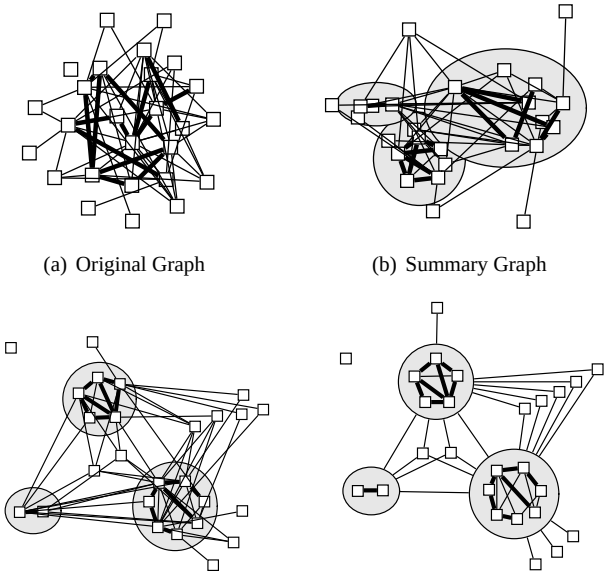


Figure 4. Social Network layouted with a spring embedder. The subgraph is highlighted with bold lines.



(a) Original Graph (b) Summary Graph

(c) Layout the Subgraphs (d) Layout Summary Graph

Figure 5. Social Network and Connection Sets

3.1 Layout of the Summary Graph

For the layout of summary graph we assume that the subgraphs have been layouted in some way and we (optionally) surround those layouts with an ellipse. These are now placed in a circular way, taking their probably different sizes into account. The oval form of the subgraphs makes it particularly easy to place them on the circle in a pleasant way, i.e. with the same distance to each other. Other geometrical objects might fit better for specific layouts of the subgraphs. The connection sets C_I are put around and between the subgraphs, so neither their vertices nor their adjacent edges interfer with each other. For the layout of the connection sets themselves information about edges that lead from them into the subgraphs is used.

Figure 3 illustrates one possible layout for a graph with three subgraphs H_1, H_2 and H_3. For layouts including up to three subgraphs the placement of connections sets does not pose any problems. Although this approach does not seem to scale well with a higher number of subgraphs, there is only a small probability that all 2^n connecting sets really show up in the final layout. The reason is, that for thin graphs, many connection sets will be empty. For dense graphs, we can concentrate on non-exclusive connection sets connecting zero, one, two or all n subgraphs. This reduces the total number of possible connection sets to $\frac{n^2+n}{2} + 2$.

3.2 Layouting a Social Network

We apply the algorithm described above to a social network graph that evolved from an analysis of organizations involved in drug policy making (see Figure 4).

The network is given as an undirected graph where vertices represent the organizations, and the existence of informal communication chains is mirrored by edges between them. Since the collection of such data always implies uncertainty, confirmed relations are given precedence over others. This is reflected by a boolean attribute *important* associated with each edge.

The confirmed relations induce a subgraph, including all edges marked as *important* and their incident vertices. The induced subgraph consists of three connected components. We decided to take these three components as the subgraphs H_1, H_2 and H_3. For the layout we will concentrate on exclusive connection sets. We show the social network at different stages of our layout algorithm in Figure 5:

Original Graph. Figure 5(a) shows the graph with a random placement of vertices. The important relations are emphasized by using bold lines for the edges, hinting at the induced subgraph.

Create the Summary Graph. The nodes of the summary graph are added to the layout (see Figure 5(b)). For the sake of clarity, the vertices belonging to one and the same component are moved closer together. Thus, the three large vertices of the summary graph can be easily spotted and identified with the vertices of the underlying graph that they encircle. The application of the function f_E is postponed to the fourth step. That has no other implications, however.

Layout the Subgraphs. Figure 5(c) shows the layouted subgraphs within the nodes. As those graphs are very small, nearly any algorithm can be used. In fact, we applied one that places nodes in a circular way, trying to reduce the number of crossings. To make the drawing clearer, we adjusted the size of the summary graph nodes and placed the subgraphs on a circle in this step already.

Layout the Summary Graph. The final Figure 5(d) results from redirecting all edges that connect nodes outside the subgraphs with nodes inside a subgraph. Now those edges lead to the summary node instead. This is exactly how function f_E is defined. After that, the remaining graph is layouted taking the fixed position and sizes of the summary nodes into account.

4. Case Study: Network Motifs

The previous case study concentrated on highlighting a few subgraphs within a relatively small graph. This section applies the general approach presented in Section 2 on larger graphs and considerably more subgraphs.

4.1 The Challenge: A Biological Network

We will demonstrate our algorithm on a biological network representing the transcriptional regulation of Escherichia coli. The graph consists of 423 labeled vertices and 578 edges. It is shown, layouted using a standard spring embedder algorithm, in Figure 6. Basically, the vertices represent operons and transcription factors that interact with each other. These interactions are modeled by edges that carry information (via an integer attribute) on the type of interaction. For a detailed description on this graph and its interpretation we refer to [**?**].
As the subgraphs we will take *motifs* – introduced in [**?**]. A motif describes small subgraphs that need not necessarily be isomorphic to each other but all match a specific pattern. In biological research, the frequent recurrence of such motifs gives insight into the order of some events and helps to get an overall view of the system, collapsing information that may be spread throughout the whole network into a few nodes.

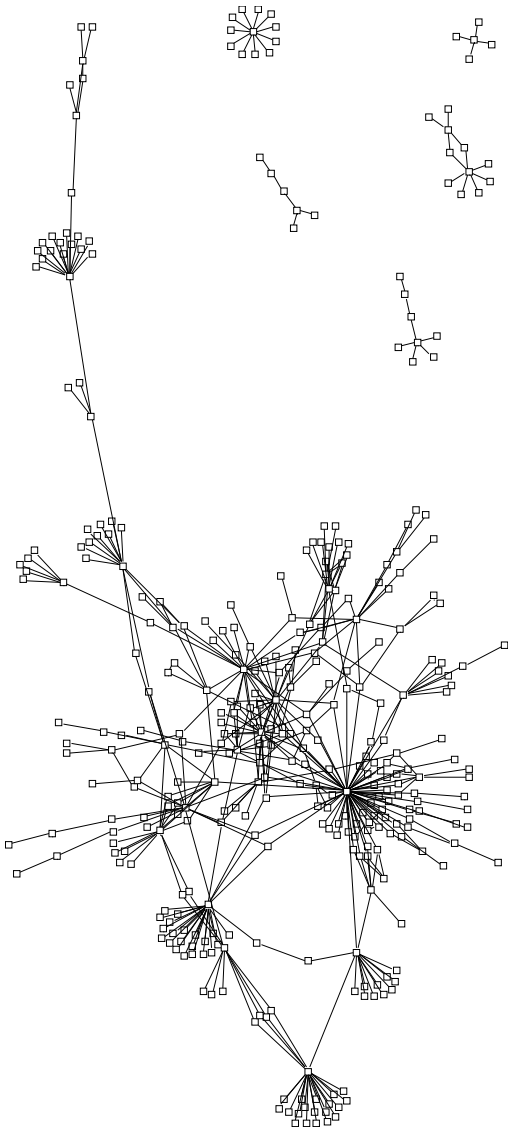


Figure 6. Biological Network This layout was created with a Spring Embedder. Note that only components of size five or more are shown. The SIMs can be recognized by their fan-outs.

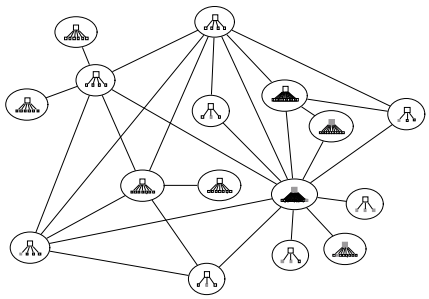


Figure 7. SIMs in Biological Network

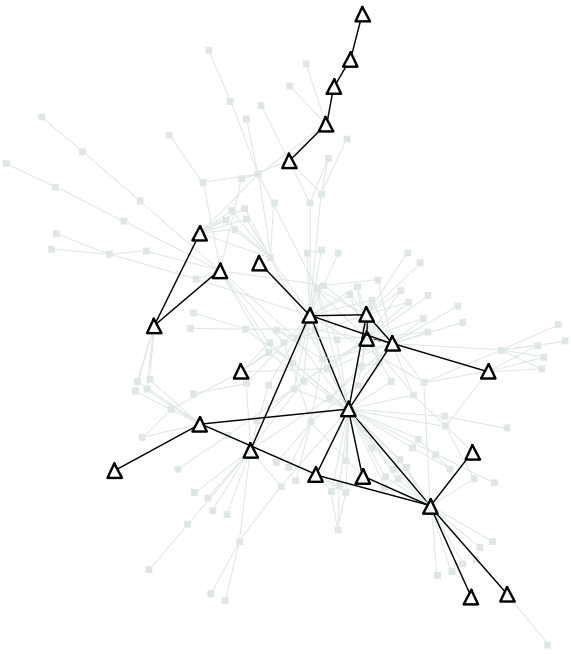


Figure 8. Biological Network showing SIMs

In this section we will focus on the large component of the biological network (see Figure 6). Additionally, we will concentrate on the *single input module* motif (SIM in short) which we adopted from [**?**]. The idea behind this motif is that several *operons* are controlled by one single operon, called a *transcription factor*. A search for this pattern in our component of the biological network identified 25 appearances of SIM which we will take as subgraphs.

We show 16 connected appearances of the SIM motif in Figure 7. One can clearly see the common pattern of a SIM, one transcription factor controlling several operons. Duplicates of vertices appearing in multiple motifs are highlighted. An operon appearing in one motif can, by definition, not be an operon in another motif (as it has exactly one input), but it can play the role of the transcription factor, controlling other operons.

4.2 Layouting the Biological Network

We again start by creating the summary graph for the 25 subgraphs. Since the biological network is quite large, we replace the individual subgraphs with a motif-specific geometric shape. This allows us to distinguish between original vertices and vertices representing motifs as well as between several motifs. For the SIM motif, a triangle has been chosen, representing the transcriptional factor on top that controls several operons below. Figure 8 shows a version of the summary graph layouted using the same spring embedder as before.

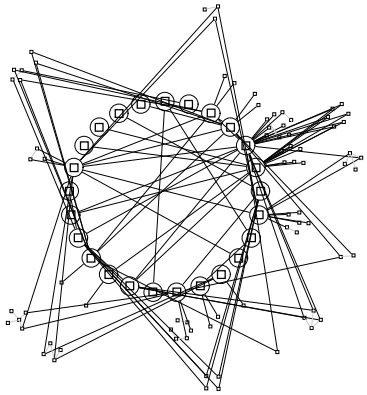


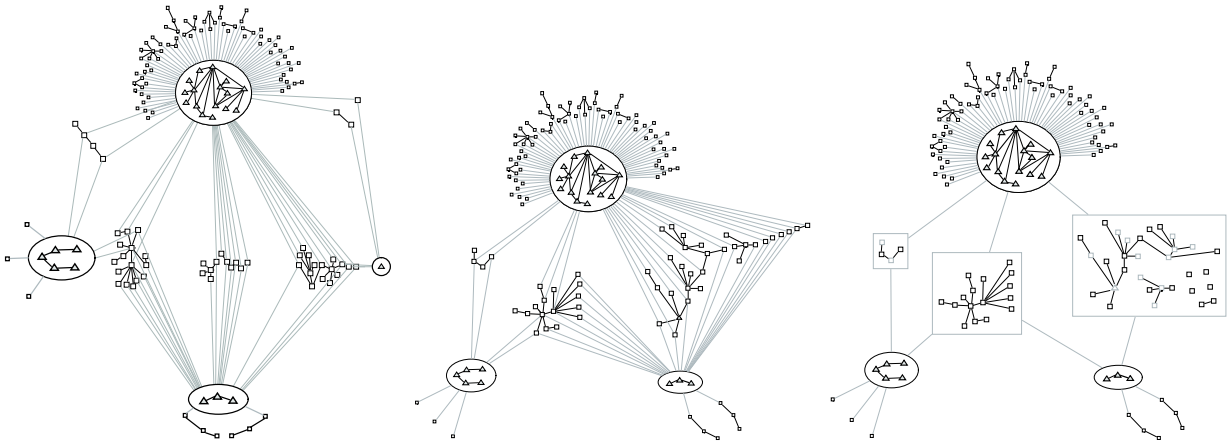
Figure 9. SIMs and their Connection Sets

Applying the connection set method described in Section 3 turned out not to be feasible. In theory, the number of possible connection sets is 2^{25}. As mentioned before, the actual number of non-empty connection sets is much smaller: In our case there have been 36 non-empty exclusive connections sets. Eleven of those have been connected to exactly one SIM, and another eleven to exactly two SIMs and 13 to more than two SIMs. There have been no vertices connected to all SIMs. In Figure 9 we show a layout where we concentrated on vertices connecting *exactly* one or two SIMs (therefore we took *exclusive* connection sets). We did not concentrate on the final optimization step, since we decided to pursue another direction to cope with that large graph.

We now apply the ideas used for layout of the social network in Section 3 to our biological network. As Figure 8 indicates the SIMs induce a subgraph. This subgraph consists of four components, one of which consisting of one vertex only. We will now visualize the connections between these four components. The justification is that an analyst who is interested in the occurrence of motifs will very probably be interested in the connections between those subgraphs as well. Figure 10(a) shows one possible layout resulting from that scenery.

For demonstrating a final optimization step we first needed to complicate the layout. So, we decided to ”ignore” the one-vertex component and treated this SIM as a normal vertex (though it kept its triangular form for identification). That leads to three components containing three, five and 16 vertices respectively.

The summary graph is roughly layouted as shown in Figure 3. In the next step, the subgraphs and the connection sets are layouted. This is exactly where one of the powers of the presented algorithm shows up: All those parts of the graph can be layouted using individually chosen algorithms. In fact, we applied four different methods to create the final layout which we present in Figure 10(b):



(a) All Connected Components of SIMs (b) Only large Connected Components (c) Optimized Connection Sets

Figure 10. Layouts of the Biological Network using Connection Sets

- The small components (up to five vertices) have been placed *randomly*, forcing them not to use too much space.
- A *spring embedder* followed by a *crossing-reduction algorithm* has been applied to the subgraph on top and the central connection set.
- The connection set on the right hand side was layouted in some kind of *interactive, left-to-right algorithm* with the aim to reduce the number of edge crossings (neglecting the size of the layout).
- Finally, for the connection set topping the large component, an algorithm has been chosen that works particularly well for thin graphs where most of the vertices are connected to one central vertex. Since there are 73 vertices loosely connected with each other but nearly all of sharing an edge with the large summary node, this *pincushion* algorithm (the drawing should explain the name) was judged ideal.

All those layouts had to take into account the fixed position of the summary nodes.

4.3 Optimization for Large Connecting Sets

For large graphs, with many edges between connection sets and summary nodes, the notion of the summary graph can be extended: The connection sets are considered components and put into (or replaced by) a summary node themselves. Edges incident to a vertex within a connection set are redirected to the corresponding summary node. This considerably reduces the number of edges in the final layout and therefore increases readability. Additionally, the connection sets, having lost many edges, can now be layouted taking up much less space. The optimized layout of our biological network is shown in Figure 10(c).

5. Conclusion

We presented an approach for emphasizing subgraphs within graphs. Additionally, we described two applications for our approach: One for visualizing connections between subgraphs, and one for highlighting many similar subgraphs. In a final step we combined both techniques. We implemented all techniques presented in this paper and integrated them within a graph editor called GRAFFITI. The GRAFFITI platform has been developed at University of Passau, Chair for Theoretical Computer Science.
There are situations where our approach produces undesired results, because considering subgraphs for the layout of entire graphs is not always reasonable. As an example take a subway network as the graph, and a connection between two places as the subgraph. Layouting the subgraph first, and the subway network afterwards is not a good idea. First, the vertices of the connection will probably layouted on a single line or in a circle. And second, the viewer will lose orientation, because the layout does not match his mental map, which is influenced by the geographical distribution of the vertices. Still the mental map could be restored by enriching the subgraph with additional information about its environment or using graph animation.

6. Future Work

Our future work will include the following topics:

Consider Labels. For the graphs to be useful for analysis, some way must be provided to display the labels

(names) of the vertices. That can be done using tooltips (not taking up any space in the drawing) or by putting labels besides vertices.

Graph Animation. In case subgraphs are not known in advance, we could use graph animation to transform the initial layout into our optimized one. For this we have to adopt existing animation techniques and probably create new ones.

Vertex Duplication. When vertices are duplicated, the connection between the vertex and its clone gets lost. This connection should be preserved and displayed in some way or the other. Possible methods include labeling, coloring pairs the same way, etc.

Non-Disjoint Motifs. Still vertex duplication is not the only possibility to handle non-disjoint motifs or subgraphs. Some of them have been addressed in Section 2.

Several Motifs. It might be interesting to use several motifs instead of one. Since one can use different shapes and sizes for the summary vertices, they should be clearly distinguishable. Obviously the vision is to combine motif detection with graph layouting.

Acknowledgments. Holger Cleve and Richard Kuntschke provided useful help and constructive comments on earlier revisions of this paper.