A Meta Heuristic for Graph Drawing

Learning the Optimal Graph-Drawing Method for Clustered Graphs

Oliver Niggemann
University of Paderborn
Warburgerstr. 100
Paderborn, Germany
murray@uni-paderborn.de

Benno Stein
University of Paderborn
Warburgerstr. 100
Paderborn, Germany
stein@uni-paderborn.de

ABSTRACT
The problem of finding a pleasant layout for a given graph is a key challenge in the field of information visualization. For graphs that are biased towards a particular property such as tree-like, star-like, or bipartite, a layout algorithm can produce excellent layouts—if this property is actually detected.

Typically, a graph may not be of such a homogeneous shape but is comprised of different parts, or it provides several levels of abstraction each of which dominated by another property.

The paper in hand addresses the layout of such graphs. It presents a meta heuristic for graph drawing, which is based on two ideas: (i) The detection and exploitation of hierarchical cluster information to unveil a graph’s inherent structure. (ii) The automatic selection of an individual graph drawing method for each cluster.

Categories and Subject Descriptors
I.4.10 [Computing Methodologies]: IMAGE PROCESSING AND COMPUTER VISION; I.2.6 [Computing Methodologies]: ARTIFICIAL INTELLIGENCE

General Terms
Graph-drawing, Learning, Information-Visualization, Clustering

1. INTRODUCTION
Some of the recent graph drawing developments arm up layout algorithms by the exploitation of cluster information: A graph is divided into subgraphs, the so-called clusters, which can be laid out rather independently from each other.\(^1\) Aside from a complexity reduction of the layout process within an order of magnitude, a clustering of the graph can also convey additional structural information. For instance, when rendering the parts graph of a large technical system, subgraphs that stand for assemblies can be identified and accentuated visually.

\(^1\)Note that during the inter-cluster-layout process, links between clusters must also be taken into account.

Notice that the clustering idea entails two subproblems: (i) The identification of appropriate clusters, and (ii) the integration of cluster information into a layout algorithm.

This paper provides new solutions for both problems. With respect to problem (i) the $A$-maximization idea is utilized. This recently developed clustering approach is based on a graph’s partial connectivity (the $A$-value) and is especially suited for the clustering of non-distance graphs. With respect to problem (ii) a strategy radically different to existing research is pursued: Instead of adapting a particular drawing method, we select heuristically, say knowledge-based, the most suited drawing method from a set of methods for each cluster. Since graph drawing methods make also heavy use of heuristics, our approach can be regarded as a meta heuristic for graph drawing.

The meta heuristic approach combines the power of clustering with a differentiation between graph drawing methods: A graph is visualized at different levels of abstraction, and on each level the most suitable layout technique is applied. Note, however, that the selection of a suitable layout technique for a given cluster poses a new problem, which must be solved to make the meta heuristic approach a working concept.

2. A META HEURISTIC APPROACH TO GRAPH DRAWING
The following pseudo code defines the basic steps of the meta heuristic.

\begin{align*}
\text{Input.} & \quad \text{A graph } G = (V, E), \text{ a function } c_q, \text{ mapping from graphs onto graph drawing methods.} \\
\text{Output.} & \quad \text{Positions for all vertices.} \\
\text{function} & \quad \text{meta heuristic}(G) = (V, E) \\
(1) & \quad \text{Find clusters } C_1, \ldots, C_n \text{ in } G. \\
(2) & \quad \text{Create the condensed graph } G' = (V', E') \text{ with} \\
& \quad \quad V' = \{C_1, \ldots, C_n\}, \\
& \quad \quad E' = \{(C_i, C_j) \mid \exists v_i \in C_i, v_j \in C_j : \{v_i, v_j\} \in E\}. \\
(3) & \quad \text{Visualize } G' \text{ by applying drawing method } c_q(G'). \\
(4) & \quad \text{for all } C \in \{C_1, \ldots, C_n\} \text{ with } |C| > 1 \text{ do} \\
& \quad (a) \quad \text{Assign } H \text{ the subgraph of } G \text{ induced by } C. \\
& \quad (b) \quad \text{meta heuristic}(H). \\
(5) & \quad \text{end.}
\end{align*}

The algorithm decomposes a graph $G$ into clusters $C_i$ and visualizes each $C_i$ recursively. If no more clusters are found in $G$, every node becomes its own cluster. Therefore the cluster sizes decrease in each recursion step, i.e. after less than $|V|$ steps only clusters of the size $1$ exist and the algorithm terminates.
The following subsections engage in the key issues of the meta heuristic: quality of a graph layout, graph clustering, and drawing method selection. The function $c_q$, which is responsible for selecting a drawing method, plays a key role in the meta heuristic. Section 3 explains how $c_q$ can be set up by a learning method.

2.1 The Quality of Graph Layouts
The quality of graph layouts and its quantization has been subject to many research projects. In this place we will not pursue such kind of research but fall back onto existing and well accepted concepts. Several criteria for evaluating the quality of a graph-layout exist:

- the number of edge crossings (e.g. in [6],[4],[10])
- the distribution of vertices (e.g. in [1],[2])
- the number of crossings between vertices and edges (e.g. in [8])
- the area required for the layout (e.g. in [16])
- the edge lengths (e.g. in [1],[2])
- combinations of the above

Any of these quality measures can be expressed by a function $q$, mapping from graph layouts into $\mathbb{R}$. The overall goal of the meta heuristic is to optimize a layout with respect to some $q$, which is realized by means of the classification function $c_q$: For a graph $G$ the function $c_q$ is intended to select that graph drawing method that results in the maximum value of $q$. Finding a suitable function $c_q$ for a given $q$ is subject to Section 3.

2.2 Graph Clustering
A model captures a lidded part of reality. When graphs are used as a means of modeling, information is often coded into the structure of the graph. Such structures can be found by applying clustering algorithms (step 1 of the meta heuristic). Clusters are used to reduce the graph sizes handled by the graph drawing algorithms and to further the understanding of complex graphs by underlining its inherent structure.

The left-hand side of Figure 1 shows a clustered graph. On the graph’s top level two clusters $A$ and $B$ have been identified; within cluster $B$ two subclusters $B_1$ and $B_2$ were found. The right-hand side of Figure 1 shows the cluster hierarchy. To detect structures in graphs we have developed the new method of $\Lambda$-maximization, which is primarily based on a graph’s edge connectivity. $\Lambda$-maximization poses a computational expensive problem, however, a good approximation of a graph’s $\Lambda$-value can be computed by the fast algorithm MAJORCLUST (see [12]).

2.3 Drawing Method Selection
One of the original ideas of this paper is the classification of graphs according to the most suitable visualization technique, given a quality measure $q$. Most graph drawing methods implicitly assume a special quality measure and are adapted to special graph classes. Other techniques can cope with all graphs, but their runtime behavior or their quality declines for some types of graphs. Therefore it makes sense to choose for each graph and each quality measure the best layout method (step 3 of the meta heuristic).

Table 1 gives for some graph drawing algorithms the corresponding quality measure and the most suited graph types. The table can neither be complete nor can it take all versions or opinions into consideration. More comprehensive overviews can be found in [5],[3] or in the proceedings of the annual Symposia on Graph Drawing (Springer, Lecture Notes in Computer Science).

![Figure 1: A hierarchically clustered graph.](image)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Quality Measure</th>
<th>Suited Graph Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reingold’s algorithm [9]</td>
<td>see [13]</td>
<td>only trees</td>
</tr>
<tr>
<td>hierarchical graph drawing [6]</td>
<td>edge crossings</td>
<td>graphs easy to transform into DAGs</td>
</tr>
<tr>
<td>force-directed approach [11],[7],[2]</td>
<td>optimal edge lengths</td>
<td>small, symmetric graphs</td>
</tr>
<tr>
<td>Tamassia’s algorithm [14]</td>
<td>edge bends</td>
<td>planar graphs</td>
</tr>
<tr>
<td>Wood’s algorithm [16]</td>
<td>edge bends</td>
<td>planar graphs</td>
</tr>
</tbody>
</table>

Table 1: Graph drawing methods with related optimization criterion and graph class.

One possible way to determine the best graph drawing method is of course to test all and then to choose the method resulting in the layout best rated by the quality measure $q$. For runtime reasons this is hardly possible, and a different approach has been chosen here.

<table>
<thead>
<tr>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of connected components</td>
</tr>
<tr>
<td>edge connection</td>
</tr>
<tr>
<td>number of biconnected components</td>
</tr>
<tr>
<td>number of vertices</td>
</tr>
<tr>
<td>number of edges</td>
</tr>
<tr>
<td>maximum distance between two vertices</td>
</tr>
<tr>
<td>diameter</td>
</tr>
<tr>
<td>maximum vertex degree</td>
</tr>
<tr>
<td>minimum vertex degree</td>
</tr>
<tr>
<td>number of clusters as found by MAJORCLUST</td>
</tr>
</tbody>
</table>

Table 2: Important graph features for drawing purposes.

For each graph $G$ a vector $\overrightarrow{v}(G) \in \mathbb{R}^p$ comprising several graph features is calculated. Table 2 shows important graph features. A classification function $c_q: \mathbb{R}^p \rightarrow \{m_1, \ldots, m_k\}$ is assumed to be given, where $m_i$ denotes a graph drawing method. $c_q$ is used to map from a feature vector $\overrightarrow{v}(G)$ onto the best layout technique. Finding an optimal $c_q$ heavily depends on the quality measure $q$. The function $q$ allows for rating features according to their support for a graph drawing method. Section 3 elaborates on how the problem of finding a classification function $c_q$ can be reduced to a standard regression problem, making the automatic learning of $c_q$ possible.
3. LEARNING THE DRAWING METHOD SELECTION

The quality of the meta heuristic depends decisively on the choice of the classification function \( c_q \). As described above, \( c_q \) is used to map from a feature vector \( \varphi(G) \) onto the layout technique best suited to optimize the given quality criterion \( q \). In this section a novel method for learning \( c_q \) by applying standard regression techniques is given.

3.1 The learning process

For non-clustered graphs this learning process is quite simple: A set of typical graphs \( \{G_1, \ldots, G_m\} \) has to be given. Each graph \( G \) is visualized using all graph drawing methods \( \{m_1, \ldots, m_n\} \). The best method according to \( q \) is called \( m(G) \). The feature vector \( \varphi(G) \) is saved together with \( m(G) \). This results in a database \( DB \) of classified feature vectors \( DB = \{< \varphi(G_1), m(G_1)>, \ldots, < \varphi(G_m), m(G_m)>\} \).

Databases like \( DB \) are normal input for standard regression algorithms (see also [15]), i.e. \( c_q \) can be learned by applying regression to \( DB \). \( c_q \) learns which features support or weaken the applicability of a graph drawing method. For runtime reasons it may be reasonable for large databases and complex feature-vectors to use neural-networks as a heuristic to solve the regression problem.

The meta heuristic combines the learning process as described above and the recursive clustering approach: Not the original graphs but all graphs created by the recursive clustering are used for parameterizing the classification function. This will now be formally described.

| Input. | Graphs \( \{G_1, \ldots, G_m\} \). |
| Output. | A function \( c_q \) mapping from graphs onto graph drawing methods. |

**learning step I**

1. Choose an optimality criterion \( q \).
2. For all graphs \( G \in \{G_1, \ldots, G_m\} \), \( G = (V, E) \) do
   1. learn(\( G \))

**function learn(\( G = (V, E) \))**

3. Find clusters \( C_1, \ldots, C_n \) in \( G \).
4. Create the condensed graph \( G' = (V', E') \) with \( V' = \{C_1, \ldots, C_n\} \), \( E' = \{ (C_i, C_j) | \exists v_i \in C_i, v_j \in C_j : \{v_i, v_j\} \in E \} \).
5. Visualize \( G' \) by applying all implemented graph drawing methods.
6. Create the database \( DB \) by calculating the feature vector \( \varphi(H) \) and saving it together with the best graph drawing algorithm.
7. For all \( C \in \{C_1, \ldots, C_n\} \) with \( |C| > 1 \) do
   1. Assign \( H \) the subgraph of \( G \) induced by \( C \)
   2. learn(\( H \))
7. od

**learning step II**

9. Learn a classifier \( c_q \) by applying regression to \( DB \).

Note that the resulting function exclusively relies on the graphs used by the learning method, i.e. by choosing graphs from a special domain, the algorithm specially adapts to this domain.

Learning the function \( c_q \) can be seen as a preprocessing step for the meta-heuristic. Since it is applied usually only once, runtime considerations are less important here.

4. SOME EXPERIMENTAL RESULTS

The meta heuristic has been applied to several domains, two are presented here.

4.1 Envisioning Configuration Knowledge-Bases

The resource-based configuration paradigm is a successful approach in the field of automatic configuration of technical systems. It allows for a local modeling of distributed systems.

From a visualization point of view, knowledge-bases for resource-based configuration establish bipartite, undirected graphs. Because most conventional graph drawing techniques fail with respect to a purposeful representation, we have tested the meta-heuristic approach here.

The left-hand side of Figure 2 shows a part of a configuration knowledge-base. The subgraphs, say, clusters, in the center were visualized using a spring-embedder; for the other clusters a hierarchical layout algorithm has been employed. The right-hand side of the figure shows the graph of clusters, i.e. an abstract view onto the graph.

20 Graphs with 500 vertices and 5 graphs with 1000 vertices have been visualized. The quality measure applied for the described experiments was the number of edge crossing; other measures such as the layout symmetry gave similar results. Two graph drawing algorithms have been implemented, namely, a spring-embedder [1],[7],[2] and a hierarchical layout method [6],[10]. The meta heuristic resulted in 39% less edge crossings than the hierarchical approach and in 27% less edge crossing than the spring-embedder solution.

4.2 Network Traffic Analysis

Network traffic analysis is substantial for administrating and analyzing computer networks. The amount of traffic between all pairs of computers in the network is recorded in the so-called traffic matrix. Each node in the computer network becomes a vertex and all communication between two nodes results in an weighted edge. The edge-weight is proportional to the amount of communication.

Figure 3 shows a network from its physical setup (left), a related communication graph (middle), and a nicely drawn version of the communication graph (right). Its two top clusters were visualized by an spring-embedder, the bottom cluster graph shows the typical hierarchical layout.
5. SUMMARY

The presented paper introduces a new meta-heuristic for drawing large graphs. The meta-heuristic works by firstly clustering a graph and then using for each subgraph the optimal drawing method. The dynamic choice of a suited drawing algorithm allows for the visualization of heterogeneous graphs as created by many applications, whereof two are explained in more detail.

Two mains paradigms are defined and combined by the authors:

1. When visualizing graphs from real world applications, there exists no drawing method that is suited for all graphs. Thus, the graphs should be analyzed and the best layout algorithm chosen dynamically.

2. The graph-inherent structure has to be emphasized by the layout algorithm. To accomplish this job special clustering algorithms are needed.

To find natural clusters within a graph the novel approach of $\Lambda$-maximization is applied. Clearly, clustering is a generic means to cope with the complexity when layouting large graphs. Note however, that a natural clustering does also support the understanding of complex graphs.

6. REFERENCES


