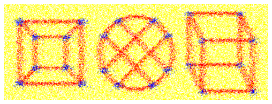

Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 14, no. 1, pp. 97–119 (2010)

Visual Analysis of One-To-Many Matched Graphs

Emilio Di Giacomo[1] *Walter Didimo*[1] *Giuseppe Liotta*[1]
Pietro Palladino[1]

[1]Dipartimento di Ingegneria Elettronica e dell'Informazione,
Università degli Studi di Perugia, ITALY

Abstract

Motivated by applications of social network analysis and of Web search clustering engines, we describe an algorithm and a system for the display and the visual analysis of two graphs G_1 and G_2 such that each G_i is defined on a different data set with its own primary relationships and there are secondary relationships between the vertices of G_1 and those of G_2. Our main goal is to compute a drawing of G_1 and G_2 that makes clearly visible the relations between the two graphs by avoiding their crossings, and that also takes into account some other important aesthetic requirements like number of bends, area, and aspect ratio. Application examples and experiments on the system performances are also presented.

Submitted:	Reviewed:	Revised:	Accepted:
November 2008	July 2009	August 2009	November 2009
	Final:	Published:	
	November 2009	January 2010	
Article type:		Communicated by:	
Regular Paper		I. G. Tollis and M. Patrignani	

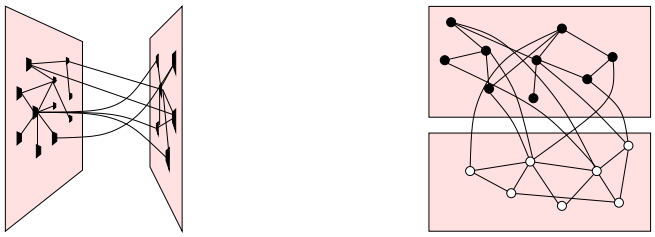
Research partially supported by the MIUR Project "MAINSTREAM: Algorithms for Massive Information Structures and Data Streams".

E-mail addresses: digiacomo@diei.unipg.it (Emilio Di Giacomo) didimo@diei.unipg.it (Walter Didimo) liotta@diei.unipg.it (Giuseppe Liotta) palladino@diei.unipg.it (Pietro Palladino)

1 Introduction

The visual analysis of complex data sets is one of the most natural applications of graph drawing technologies (see, e.g., [11, 15, 20, 24, 25] for some recent works). A typical application scenario consists of a set of data (nodes) and one or more relationships among these data (each relationship is a set of edges); therefore one is given one or more graphs on the same set of nodes. Both each graph must be visualized in a readable way and possible similarities among the different graphs must be easily detected by looking at the different drawings. This scenario has, for example, motivated a rich body of papers and systems about simultaneous graphs embeddings and visualizations of evolving graphs (see, e.g., [4, 10, 12, 13, 14]).

Recently, Collins and Carpendale [5] proposed a new research direction devoted to the visual comparison and analysis of heterogeneous data sets. The input consists of n sets of data $D_1, D_2, \ldots, D_n$, such that for each D_i a distinct set of *primary relationships* (i.e., a distinct graph) is defined; also, there are *secondary relationships* which model semantic connections between data belonging to different sets. The visualization consists of a set of n drawings (one for each graph) on top of which the edges that represent the secondary relationships are displayed. Collins and Carpendale present a system, called VisLink, where each graph is drawn on a distinct plane and the secondary relationships are links between these planes (see Fig. 1(a) for a schematic illustration). The work by Collins and Carpendale extends a previous work by Schneiderman and Aris where multi-plane views with inter-plane edges are used to visualize different semantic substrates of a graph [21](see Fig. 1(b) for an illustration).



(a) (b)

Figure 1: Schematic illustrations of a visualization (a) adopted by VisLink, (b) using different semantic substrates of a network. In both the visualizations the drawing on each plane has been computed without taking into account the relationships with the other. This may cause many crossings between inter-sets relationships.

Motivated by applications of social network analysis and of Web search clustering engines, we elaborate on the concepts of Collins and Carpendale by studying the following problem: We are given two graphs G_1 and G_2 and a function that defines a set of secondary relationships by mapping some of the vertices of G_1 to some other vertices of G_2; we aim at visually analyzing and interacting

both with G_1, G_2 and with their secondary relationships. We observe that the systems described in [5, 21] follow the common approach of drawing each graph independently of each other. As a result, the secondary edges may be difficult to read as they can have many crossings. Our main goal is to design a system where the two drawings are computed by taking into account the edge-crossing minimization of the secondary edges. We focus on *one-to-many relationships* between G_1 and G_2, i.e., vertices of G_1 are associated with disjoint subsets of vertices of G_2. It may be worth noticing that a one-to-many relationship implies a two-level clustering of the vertices of G_2 such that no two clusters share any vertex.

The main contributions of the paper are the following:

- We introduce the concept of *one-to-many matched graphs* and define drawing conventions for these graphs in a *strong* and in a *weak* model. Both drawings require that the secondary relationships between the graphs do not cross each other (Sect. 3).

- We describe a system that computes strong and weak one-to-many matched drawings of the input graphs by also taking into account the optimization of important aesthetic requirements. Furthermore, the system provides the user with several interaction functionalities that make it possible to analyze the drawings at different levels of details by collapsing/expanding clusters and by filtering information with the definition of node/edge thresholds (Sect. 4). Our drawing approach combines orthogonal drawings in the topology driven approach with circular drawing algorithms, and adopts an edge bundling technique to reduce the visual complexity introduced by some links. This technique may be of independent interest since it can be used to draw two-level clustered graphs where no two clusters overlap.

- We show the effectiveness of the system by presenting two application examples, one concerned with social network analysis and the other in the context of Web search clustering engines (Sect. 5). An experimental study on the system performances is also presented, which gives more indications about strength and limits of our approach (Sect. 6).

We finally remark that the problem of drawing two matched planar graphs G_1 and G_2 with one-to-one secondary relationships between them have been originally studied in [9], where it is required that the drawing of each G_i is planar and that the secondary edges are represented as non-intersecting horizontal segments.

2 Basic Terminology

We assume familiarity with basic concepts of graph theory [17]. In the following we recall some terminology of graph drawing and planarity used throughout the

paper (refer to [7, 18] for more details). Let $G = (V, E)$ be a graph. A *drawing* of G is a geometric representation of G in the plane such that each vertex $v \in V$ is drawn as a geometric shape p_v (e.g., a point, a circle, a polygon) and each edge $(u, v) \in E$ is drawn as a simple Jordan curve connecting p_u and p_v. We denote by $\Gamma(G)$ a drawing of G.

A drawing $\Gamma(G)$ is *planar* if there is no edge crossing, no intersection between vertices, and no intersection between an edge (u, v) and a vertex w distinct from u and v. A graph is *planar* if it admits a planar drawing. A planar drawing $\Gamma(G)$ of G partitions the plane into connected regions, called *faces*. Exactly one face is an infinite region, and it is called the *external* face; the other faces are said to be *internal*. The *boundary* of an internal (external) face is the circular clockwise (counterclockwise) sequence of vertices and edges delimiting it. The set of face boundaries of $\Gamma(G)$ is called a *planar embedding* of G. Note that, for a planar graph G there are many infinite planar drawings of G that determine the same planar embedding. Therefore, a planar embedding of G can be described just in terms of face boundaries, without referring to any specific drawing of G. Also observe that a planar embedding of a graph fixes the clockwise order of the edges incident around each vertex. We denote by Ψ a planar embedding of a graph G. An *embedded planar graph* is a graph along with a given planar embedding. If G is an embedded planar graph with embedding Ψ, we say that $\Gamma(G)$ is a planar drawing of G that *preserves* Ψ if the face boundaries of $\Gamma(G)$ are the same as in Ψ.

A drawing $\Gamma(G)$ is an *orthogonal drawing* if each edge is drawn as a chain of horizontal and vertical segments. A *bend* in $\Gamma(G)$ is a point of an edge shared by a horizontal and a vertical segment of the edge. A drawing $\Gamma(G)$ is a *circular drawing* if there is a circle passing through all vertices and each edge is drawn as a straight-line segment. Figure 2 shows an orthogonal drawing and a circular drawing of the same graph G. In the remainder of the paper, if $G = (V, E)$ is a graph and $V' \subseteq V$ we denote by $G(V')$ the subgraph of G induced by the vertices of V'.

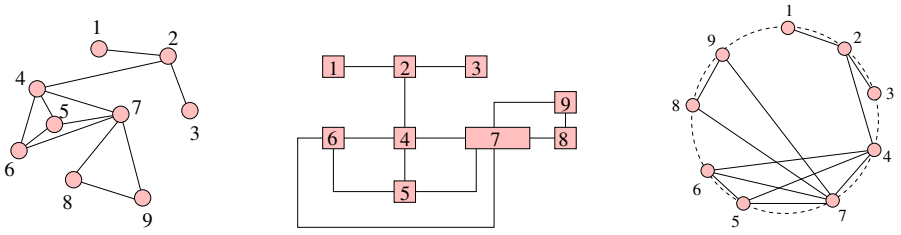


Figure 2: (a) A planar graph G. (b) An orthogonal drawing of G. (c) A circular drawing of G.

3 One-To-Many Matched Graphs and Drawings

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two distinct graphs. We say that $\langle G_1, G_2 \rangle$ is a pair of *one-to-many matched graphs* if: (i) Each vertex u of G_1 is associated with a subset $M(u) = \{v_1, v_2, \ldots, v_k\}$ of vertices of G_2, that we call the *cluster of* u in G_2; (ii) the set of clusters $\{M(u) \subseteq V_2 : u \in V_1\}$ is a partition of V_2, i.e., $\bigcup_{u \in V_1} M(u) = V_2$ and for each pair u, v of distinct vertices of V_1, we have $M(u) \cap M(v) = \emptyset$.

Let $\langle G_1, G_2 \rangle$ be a pair of one-to-many matched graphs, and let $\Gamma(G_1)$, $\Gamma(G_2)$ be drawings of G_1 and G_2, respectively. We say that $\langle \Gamma(G_1), \Gamma(G_2) \rangle$ is a *one-to-many matched drawing* if the following properties hold:

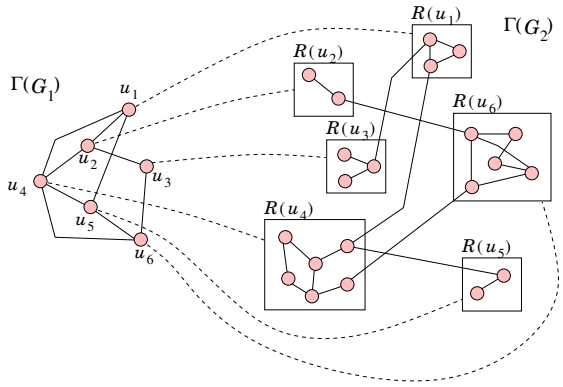
- **(P1)** The bounding boxes of $\Gamma(G_1)$ and $\Gamma(G_2)$ do not intersect.
- **(P2)** For each vertex u of G_1, cluster $M(u)$ in $\Gamma(G_2)$ is bounded by a rectangular region $R(u)$ such that: (i) $G(M(u))$ is completely contained in $R(u)$; (ii) each vertex $v \in V_2 \setminus M(u)$ is outside $R(u)$; (iii) each edge of G_2 intersects the boundary of $R(u)$ at most once.
- **(P3)** For each vertex u of G_1, there exists a simple curve $\ell(u)$ that connects the geometric shape p_u representing u in $\Gamma(G_1)$ to the boundary of $R(u)$ in $\Gamma(G_2)$, in such a way that for each pair u, v of distinct vertices of V_1, we have $\ell(u) \cap \ell(v) = \emptyset$.

In the paper, simple curves $\ell(u)$ are referred to as *matching connections*. Property **(P3)** guarantees that there is no intersection between distinct matching connections. A one-to-many matched drawing is said to be *strong* if the centers of the vertices of $\Gamma(G_1)$ have distinct y-coordinates and regions $R(u)$ are vertically ordered in $\Gamma(G_2)$ according to the positions of the corresponding vertices in $\Gamma(G_1)$. More formally, if $u_1, u_2 \in V_1$ and p_{u_1} is above p_{u_2} in $\Gamma(G_1)$, then $R(u_1)$ is completely above $R(u_2)$ in $\Gamma(G_2)$. In the paper, a one-to-many matched drawing that is not strong will be referred to as a *weak* one-to-many matched drawing. Figure 3 shows two examples of one-to-many matched drawings for the same pair of graphs. The one in Fig. 3(b) is a strong one-to-many matched drawing. Note that in the strong model it is always possible to draw each pair $l(u)$, $l(v)$ of matching connections such that $l(u)$ is above $l(v)$ if $R(u)$ is above $R(v)$. In the remainder we always compute drawings with matching connections that have this property.

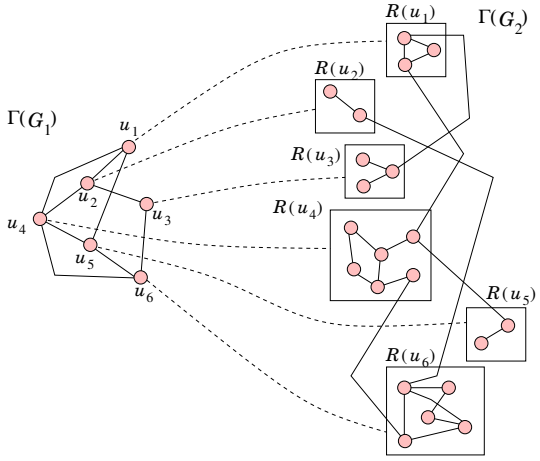
4 The System MOM

In this section we present a system for the display and the visual analysis of one-to-many matched drawings. We call our system MOM[1]. Let $\langle G_1, G_2 \rangle$ be a pair of one-to-many matched graphs to be visualized. MOM displays the drawing of G_1 to the left of the drawing of G_2, according to the following main criteria:

[1] MOM stands for Matched One-to-Many graphs



(a)



(b)

Figure 3: (a) A weak one-to-many matched drawing of a pair of matched graphs. (b) A strong one-to-many matched drawing for the same pair of graphs.

- **(C1)** It assumes that a drawing $\Gamma(G_1)$ is given as part of the input or that it can be computed using some classical graph drawing algorithm.

- **(C2)** It concentrates on the computation of $\Gamma(G_2)$, while trying to optimize a certain number of aesthetic criteria, other than guaranteeing that $\langle \Gamma(G_1), \Gamma(G_2) \rangle$ is a one-to-many matched drawing.

- **(C3)** Once $\Gamma(G_2)$ has been computed, it draws the matching connections and provides the user with a set of interaction functionalities for the visual analysis of the resulting drawing.

Criterion **(C1)** is motivated by several application scenarios that we had in mind during the design of the system. In these applications G_1 is often a graph whose entities represent geographic locations and therefore their position is either fixed or strongly constrained (examples are given in Sect. 5). About **(C2)**, we focus on well recognized aesthetic criteria like number of crossings, number of bends, drawing area. Since the optimization of these criteria typically leads to an NP-hard problem, we propose some heuristics based on engineered versions of popular graph drawing algorithms, which are able to deal with the constraints of a one-to-many matched drawing. As an additional aesthetic criterion we require that $\langle \Gamma(G_1), \Gamma(G_2) \rangle$ is computed in such a way that the matching connections can be always drawn without intersecting the edges of G_2. When G_2 is a dense graph, $\Gamma(G_2)$ may have a high visual complexity, which makes it difficult to read the drawing at a whole, independent of the drawing strategy applied. This is the motivation for **(C3)**.

4.1 Drawing Algorithm

Our drawing strategy for $\Gamma(G_2)$ combines different drawing conventions. We use orthogonal drawings for the layout of the rectangular regions $R(u)$ and their connections. Circular drawings are used to represent $G(M(u))$ inside $R(u)$. Finally, in order to simplify the visual complexity, we adopt a bundling operation for the edges connecting a vertex inside a region $R(u)$ to vertices outside $R(u)$; to avoid ambiguity, we use a "confluent-like" representation for these edges, as explained later. The algorithms used for the different drawing conventions have been engineered in order to deal with a certain number of constraints. In the following we describe in detail the steps performed by our drawing algorithm. We denote by V_i and E_i the set of vertices and edges of G_i, respectively ($i \in \{1, 2\}$).

Step 1: Planarization. The goal of this step is to compute a suitable planar embedding of the graph consisting of "cluster vertices" and their interconnections, possibly replacing edge crossings with dummy vertices. More precisely, let $u_1, u_2, \ldots, u_n$ be the vertices of G_1 in the top-to-bottom order[2] they appear in $\Gamma(G_1)$, and let G_2' be the graph obtained from G_2 by collapsing each cluster $M(u_i)$ into a single vertex $v(u_i)$ ($1 \leq i \leq n$), called a *cluster vertex*. In G_2' edges connecting vertices in the same cluster $M(u)$ disappear, while an edge connecting a vertex in $M(u_i)$ to a vertex in $M(u_j)$ ($i \neq j$) is transformed to a *corresponding* edge between $v(u_i)$ and $v(u_j)$. We aim at computing a planar embedding Ψ of G_2' that satisfies the following two conditions:

- **(E1)** Cluster vertices $v(u_1), v(u_2), \ldots, v(u_n)$ appear counterclockwise in this order on the external face of Ψ.
- **(E2)** If $v \in M(u_i)$ in G_2 and if $e_1, \ldots, e_k$ are edges of G_2 incident to v, then the edges corresponding to $e_1, \ldots, e_k$ in G_2' appear consecutively (not necessarily in this order) around $v(u_i)$ in Ψ.

[2] If u_i and u_j have the same y-coordinate, they are ordered from right to left.

Condition **(E1)** will guarantee Property **(P3)**, i.e., the possibility of routing the matching connections without crossings among them; it also avoids crossings between matching edges and the edges of G_2. Condition **(E2)** makes it possible to simplify the links between the outside and the inside of each region $R(u_i)$ in the final drawing and to bundle these links as it will be explained in Step 3. To force **(E2)** we further transform G_2' by attaching to $v(u_i)$ a vertex v' for each vertex $v \in M(u_i)$ connected to vertices outside $M(u_i)$, and by replacing the edges $e_1, \ldots, e_k$ that are incident to v with corresponding edges $e_1', \ldots, e_k'$ connected to v'. Vertex v' is called the *image* of v.

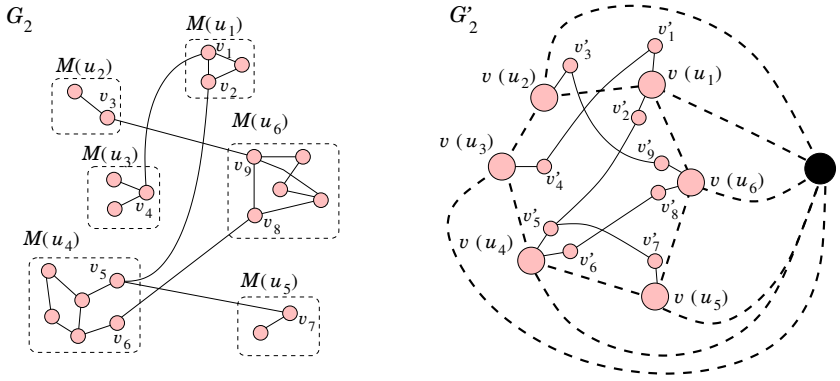


Figure 4: (a) A graph G_2. (b) The graph G_2' used in Step 1 plus the wheel gadget (bold node and dashed bold edges) adopted to guarantee **E1**; the wheel gadget is removed at the end of Step 1.

On G_2' we apply a standard planarization algorithm based on first extracting a maximal planar subgraph and then on iteratively reinserting the discarded edges by computing shortest paths in the dual graph and by replacing edge crossings with dummy vertices [7]. To force **(E1)**, we use a "wheel gadget" of uncrossable edges that will be removed at the end of the planarization phase. Figure 4 shows an example of a graph G_2' and the wheel gadget used to guarantee **(E1)**.

Notice that, quadratic and linear-time algorithms for planarity testing and edge reinsertion within the above described embedding constraints have been also proposed in [1, 16]. Our planarization phase takes $O(|E_2|(c+|V_2|)\log(c+|V_2|))$ time, where c is the number of edge crossings in the final embedding of G_2'. Namely, we spend quadratic time to extract a maximal planar subgraph, and we apply the $O(n \log n)$-time Dijkstra algorithm to compute shortest paths in the dual graph, where n is the number of vertices of the planar graph. We execute this algorithm $O(|E_2|)$ times, and each time dummy vertices are added to the planar structure to replace crossings.

Step 2: Orthogonalization and Compaction. Once a planar embedding Ψ of G_2' (with possible cross vertices) has been found, an orthogonal drawing of G_2'

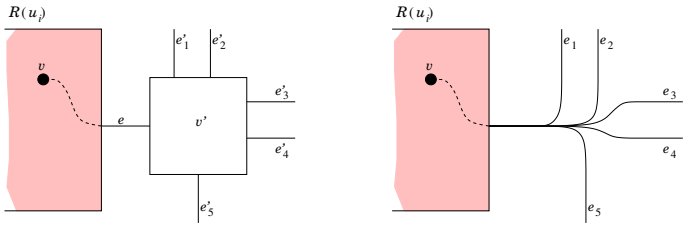
that preserves Ψ is computed. The basic idea is to use an orthogonal drawing algorithm that deals with arbitrary vertex degree and that allows for vertex size customization. Indeed, we want $v(u_i)$ to be drawn as a box big enough to host all vertices of $M(u_i)$. To this aim, the system uses the network flow based drawing algorithm described in [6], which represents a good heuristic both in terms of bend minimization and in terms of area drawing compaction.

More precisely, denote by $B(v(u_i))$ the box representing vertex $v(u_i)$. We will draw $B(v(u_i))$ as a square of a certain size r_i. In the final drawing we place a circle of radius ρ_i inside $B(v(u_i))$ and equi-distribute along its perimeter the vertices of $M(u_i)$. To determine ρ_i, we fix a minimum distance δ we want to guarantee between any two vertices of $M(u_i)$ and we set $\rho_i = \delta \cdot |M(u_i)|/2\pi$. We choose r_i to be larger enough than ρ_i so that it is possible to route the edges connecting vertices inside $B(v(u_i))$ with the outside. Each square $B(v(u_i))$ will correspond to region $R(u_i)$ in the final drawing. Also, in order to guarantee the properties of a one-to-many matched drawing, we add a certain number of constraints as described below.

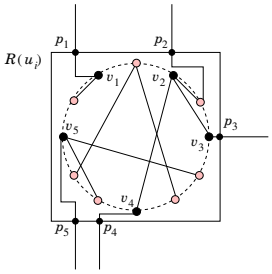
If one wants to compute a strong one-to-many matched drawing, then all vertices $v(u_1), v(u_2), \ldots, v(u_n)$ are temporarily connected in this order to form a simple cycle C that becomes the new boundary for the external face. Then the following angle and bend constraints on the vertices and edges of C are imposed: Each edge of C connecting $v(u_i)$ to $v(u_{i+1})$ $(1 \leq i \leq n-1)$ is constrained to be straight-line in the drawing, while the edge of C connecting $v(u_n)$ to $v(u_1)$ is constrained to turn always in the left direction while moving from $v(u_n)$ to $v(u_1)$. Each angle formed at a vertex $v(u_i)$ on the external face is set to be of 180 degrees. These constraints guarantee that $v(u_1), v(u_2), \ldots, v(u_n)$ are encountered from top-to-bottom in the final drawing and that they are all visible from the left. Once a drawing has been computed the edges of C are removed. If one wants to compute a (not necessarily strong) one-to-many matched drawing, then we still construct cycle C, but we only impose the constraint that the edges of C turn in the left direction or go straight while moving along C counterclockwise. Finally, in order to correctly perform the next step (i.e., the edge bundling operation), we also require that for each image vertex v' attached to a vertex $v(u_i)$, there is no edge incident to v' from the same direction of edge $(v(u_i), v')$.

All the orthogonalization constraints described above are translated into constraints on the flow network of the algorithm in [6]. The orthogonalization and compaction phases take $O((|V_1||V_2| + c)^2 \log(|V_1||V_2| + c))$ time, where c is still the number of cross vertices in the embedding Ψ. This is because the flow techniques used to compute an orthogonal drawing of a planar graph with vertices of size zero (i.e., represented as points) requires $O(n^2 \log n)$ if n is the number of vertices of the graph [23]. The algorithm used for handling vertices of size greater than zero has the same complexity, but in order to model these vertices it adds a number of dummy vertices that is proportional to the maximum size of a vertex. Since we have $O(|V_1|)$ vertices of size greater than zero (i.e., the cluster vertices) and since their maximum size is $O(|V_2|)$, the bound follows.

Step 3: Edge Bundling. This step removes each image vertex v' and creates in its place a "confluent-like" structure for the edges incident to v'. Namely, let v be the vertex of the original graph that has v' as its image and let $M(u_i)$ be the cluster that contains v. Let $e'_1, \ldots, e'_k$ be the edges incident to v' other than edge $(v', v(u_i))$. We want v' to be no longer present in the final drawing and the edges $e'_1, \ldots, e'_k$ to be replaced by the edges $e_1, e_2, \ldots, e_k$ that were originally connected to v. To simplify the final drawing however, we bundle the edges $e_1, e_2, \ldots, e_k$ from v to v'; this edge bundle follows the drawing of e from the boundary of $R(u_i)$ to v' and then it divides in k branches at v' using splines, as shown in Fig. 5(a).



(a)



(b)

Figure 5: (a) Illustration of Step 3. The image vertex v' is removed and its incident edges are replaced by a "confluent-like" structure. The dashed curve is the part of edge bundle that will be drawn in Step 4. (b) Illustration of Step 4. The black vertices inside $R(u_i)$ denote the vertices whose relative circular ordering is fixed according to their corresponding external connections.

It is important to remark that the edge bundling operation guarantees that for each vertex v inside a region $R(u_i)$ there will be at most one link (a bundle of edges) incident to v from the outside of $R(u_i)$. Since these links must be routed around the circular drawing representing $G(M(u_i))$, this property strongly simplifies the visual complexity introduced by these connections. The edge bundling step takes $O(|E_2|)$ time.

Step 4: Circular Drawing Computation. At the end of the previous step, we have a partial drawing of G_2 such that for each cluster vertex $v(u_i)$ there

is a corresponding rectangular region $R(u_i)$ and some edges incident to the boundary of $R(u_i)$ at certain points $p_1, p_2, \ldots, p_k$. To complete the drawing of G_2 we construct a circular drawing for each $G(M(u_i))$, and then connect p_j to its corresponding vertex v_j of $M(u_i)$ $(1 \leq j \leq k)$. See Fig. 5(b) for an illustration.

In order to avoid crossings between links (p_j, v_j), we force the circular order of vertices v_j to be consistent with the circular order of points $p_1, p_2, \ldots, p_k$ around $R(u_i)$, i.e., if $p_1, p_2, \ldots, p_k$ occur clockwise in this order around $R(u_i)$ then we force $v_1, v_2, \ldots, v_k$ to occur clockwise in this order in the circular drawing. Conversely, all vertices of $M(u_i)$ distinct from v_j $(1 \leq j \leq k)$ can be placed everywhere in the circular ordering (these vertices are not connected to vertices outside $R(u_i)$). In other words, if $V_{fix} = \{v_1, v_2, \ldots, v_k\}$ and $V_{free} = M(u_i) \setminus V_{fix}$, we want to find a "good" circular order for the vertices of $M(u_i)$ such that the relative order of the vertices of V_{fix} is fixed; our goal is the minimization of the number of edge crossings, which is however an NP-hard problem [19]. To solve it, we designed a variation of the heuristic described by Baur and Brandes [2], which has been experimentally shown to produce better results in terms of crossing reduction than previous heuristics for computing circular drawings, and that has been successfully adopted for the layout of two-level networks that are similar to the clustered structure of G_2 [3]. We also recall that faster but less effective circular drawing algorithms in terms of edge crossings have been described in [22].

The heuristic by Baur and Brandes computes an ordering of the vertices on a straight line ℓ, assuming that all edges are drawn on the same half-plane determined by ℓ. In terms of edge crossings this model is equivalent to place the vertices on a circle and to draw the edges as straight-line segments. The first vertex to be placed on ℓ is chosen randomly. At the generic step the next vertex to be placed is chosen as the one having the minimum number of unplaced neighbors in the graph, and it is placed on ℓ either before or after all vertices already placed, depending on which choice causes the smallest number of edge crossings. If there are more than one vertex with the (same) minimum number of unplaced neighbors, the one that has the maximum number of neighbors already placed is selected. At the end of this placement greedy heuristic, a post-processing step, called *circular sifting* is applied to further reduce the number of edge crossings if possible. The idea is to iteratively swapping a vertex with its successor vertex in the linear order on ℓ and recording the change in crossing count; the vertex is then placed in the position that corresponds to its local optimum. Denoted by n and m the number of vertices and the number of edges of the input graph, respectively, the placement greedy heuristic can be performed in $O((n+m)\log n)$ time, while repositioning each vertex once in the circular sifting phase can be done in $O(nm)$ time (see [2]).

Our variation of the algorithm in [2] works as follows. The placement greedy heuristic performs analogously to the one described above, but it assumes that the vertices of V_{fix} are already placed on ℓ in a preassigned order; therefore the placement decisions are restricted to the vertices of V_{free}. The circular sifting phase is modified so that swaps between vertices both belonging to V_{fix}

are not allowed. Once the circular ordering of the vertices of $M(u_i)$ has been computed, the algorithm equi-distributes these vertices on a circle inside $R(u_i)$ and rotates this circle in order to reduce the total length of the connections (p_j, v_j) $(1 \leq j \leq k)$, which are routed as polygonal chains of vertical and horizontal segments. The circular drawing computation over all cluster vertices takes $O(|V_1|((|V_2| + |E_2|) \log |V_2| + |V_2||E_2|))$ time (recall that $|V_1|$ corresponds to the number of cluster vertices).

Step 5: Drawing of Matching Edges. This step is simply performed by routing the matching edges as polygonal chains from the location of a vertex u_i of $\Gamma(G_1)$ to the boundary of the corresponding region $R(u_i)$ in $\Gamma(G_2)$. Since the circular ordering of the regions on the external face of $\Gamma(G_2)$ is consistent with the top-down ordering of the corresponding vertices in $\Gamma(G_1)$, this can be done without crossing between matching edges. Also, in a strong one-to-many matched drawing, each matching edge can be routed with at most two bends.

Time Complexity. The next theorem summarizes the discussion about the drawing algorithm implemented in MOM. To simplify the time complexity of this algorithm, the statement of the theorem assumes that $|V_1|$ is bounded by a constant. This appears as a reasonable assumption if $|V_1| \ll |V_2|$.

Theorem 1 *Let $\langle G_1, G_2 \rangle$ be a pair of one-to-many matched graphs such that $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. Let $\Gamma(G_1)$ be any drawing of G_1. There exists a polynomial-time algorithm that computes a one-to-many matched drawing $\langle \Gamma(G_1), \Gamma(G_2) \rangle$ (either in the strong or in the weak model) with the additional property that the matching edges can be drawn without intersecting any vertex and edge of $\Gamma(G_2)$. Also, if $|V_1|$ is bounded by a constant, and denoted by N the number $N = |V_2| + c$, where c is the number of inter-cluster edge crossings in $\Gamma(G_2)$, then the time complexity of the drawing algorithm is: $O((|E_2|N + N^2) \log N)$.*

Proof: Consider the drawing algorithm described in the previous sections. From the assumption that $|V_1| = O(1)$, the algorithm time complexity is dominated by Step 1 (Planarization) and Step 2 (Orthogonalization and Compaction). The complexity of Step 1 is $O(|E_2|(|V_2| + c) \log(|V_2| + c))$, while the complexity of Step 2 can be re-written as $O((|V_2| + c)^2 \log(|V_2| + c))$. □

4.2 Interaction Functionalities

In order to facilitate the visual analysis of the computed one-to-many matched drawings, we equipped our system with a certain number of interaction functionalities, other than conventional zooming and translation primitives. We briefly describe in the following the main actions that the user can perform on the drawing.

Cluster Expansion/Contraction: By default, all cluster regions $R(u)$ in $\Gamma(G_2)$ are expanded, i.e., the whole subgraph inside each $R(u)$ is displayed by the system. In order to compact the drawing and/or to hide some details,

the user can decide to contract a certain number of clusters by simply clicking on them. A cluster contraction redraws the cluster as a small box and hides its content. Every cluster can be expanded or contracted an infinite number of times without any restriction. After a cluster expansion/contraction, the drawing is automatically re-compacted by the system, but the orthogonal shape of the drawing remains unchanged, so to avoid the user mental map to be lost. Contracting clusters can be useful to get an overview of the inter-cluster relations before analyzing the intra-cluster ones.

Cluster Filtering: If the user is interested in focusing on some of the clusters, she can select them and hide the remaining clusters and their connections. After such an operation, the user can also decide to re-compact the remaining part of the drawing to save space if possible. When the drawing of $\Gamma(G_2)$ has many clusters and/or many inter-cluster links, the cluster filtering primitive can help to explore the graph structure portion by portion.

Edge Filtering: Our system allows the representation of edge weighted graphs. This means that a weight can be assigned to each edge of G_1 and of G_2. When a graph is too dense, the user can sparsify the links by setting an edge visibility threshold. All links having the weight below the given threshold are not shown by the system. Again, the drawing is re-compacted if required.

Edge/Vertex Highlighting: Moving the mouse over a certain vertex or cluster region, the user can decide to highlight all edges incident to that vertex or to that cluster region. A tooltip with information about the selected vertex is also displayed. This helps to get local information on the drawing. Furthermore, moving the mouse over an edge, a tooltip that displays the labels of its end-vertices is shown. This helps when just a portion of the selected edge fits in the current view.

4.3 Implementation Notes

The user interface of the MOM system has been implemented in Java, using JDK 1.5. The same language has been also used to implement the modified circular drawing algorithm. For the implementation of planarization and orthogonalization algorithms we used the GDToolkit library [1]. Since this library is written in C++, we used the JNI (Java Native Interface) technology to call native GDToolkit methods inside the Java code of MOM.

5 Application Examples

One-to-many matched graphs occur in several applications contexts. Here we briefly present two examples, one on social network analysis and the other in

the field of Web search clustering engines. The drawings of Fig. 6, 7 and 8 were computed using MOM.

The first example focuses on the co-authorship network of the 15^{th} International Symposium on Graph Drawing, GD 2007. In this example, G_1 is the graph having European countries as vertices and edges between countries that cooperated in co-authoring some papers. Each edge has a weight equal to the number of papers resulting from the cooperation of the connected countries. The drawing $\Gamma(G_1)$ is a simple straight-line drawing, where each vertex is placed at a fixed location on a geographic map. Graph G_2 represents authors and their cooperations in the articles. Figure 6 shows a one-to-many matched drawing in the strong model. The drawing gives an overview of the network structure, which reveals the number of contributing authors for each country and a relevant level of cooperation among the different countries. Looking inside a country, it is possible to see its different sub-communities. For example, it is easy to recognize two sub-communities in Greece, in Italy, and in Czech Republic, several communities in Germany, and one big community in Spain. Selecting an author in a country, all her connections with other authors are highlighted by the system. In the figure, author "Kaufmann" inside Germany is selected, and the system highlights (in bold red color) his connections with other authors, three in Greece and one in Italy. Moving the mouse over one of the bold red edges, a tooltip that reports the labels of its end-vertices is displayed. Figure 7 shows an example of edge and vertex filtering on the previous drawing, which makes it easier to focus on specific relationships. Namely, the edges of $\Gamma(G_1)$ has been filtered so that only those edges with a weight greater than 1 are shown. The vertices of $\Gamma(G_2)$ have been filtered in such a way that only the countries having some incident links in $\Gamma(G_1)$ are shown (i.e., Germany, Italy, and The Netherlands). Then, cluster Germany has been contracted to focus on the interplay between Italy and The Netherlands. After the vertex filtering and contraction operations, $\Gamma(G_2)$ is recomputed so to become more compact without destroying the user's mental map. In the figure, the connections of author "Meijer" are highlighted in bold red.

The second example shows an interesting integration of an on-line service of the Italian Yellow Pages[3] with recent Web search clustering engine technologies. More precisely, the Italian Yellow Pages has an on-line service that allows users to search for events in a desired geographic area. In our example, we queried the event "concert" in the north of Italy. The service returns a list of results that match the query, every result coming with a brief summary containing additional information on the event. A the same time, it presents a geographic map that displays the location of each event. We passed the results to a clustering engine [8] that is able to group them into distinct thematic categories, each category having a label and a relevance rank automatically computed by the system. Figure 8 shows as G_1 the graph having a vertex for each region involved by our query (in this case G_1 has no edge), while G_2 is the graph whose vertices are the concerts returned by the Italian Yellow Pages and such that two

[3] `http://www.visualpaginegialle.it/`

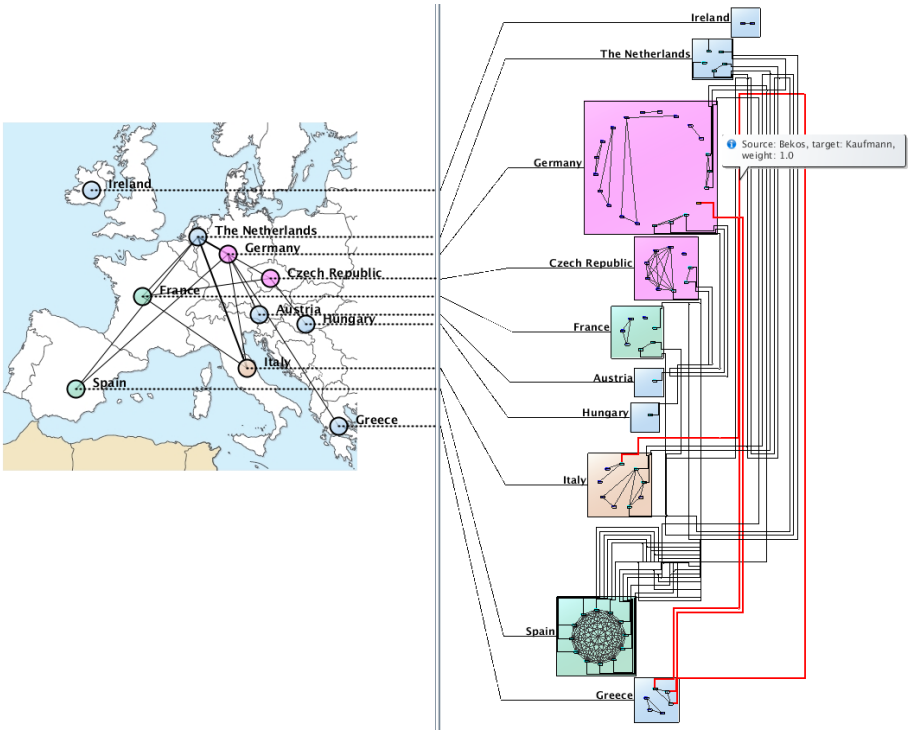


Figure 6: A one-to-many matched drawing showing the European co-authorship network of GD 2007.

concerts are connected by an edge if they belong to the same thematic category. The figure shows only the edges induces by the two most relevant categories. In the figure, cluster "Emilia Romagna" is selected and all its incident links are highlighted using the red color. It is interesting to observe that 90% of the links involves such a region, which means that "Emilia Romagna" hosts most of the concerts related to the first two thematic categories.

6 System Performances

We have tested our system in order to measure its performances. Our main goal was to measure the running time and some important aesthetic requirements, like number of crossings, number of bends, drawing area, and aspect ratio (width/height). We compared the algorithm for strong one-to-many matched drawings against the algorithm for weak one-to-many matched drawings, so to understand the trade-off between the results of the two algorithms. Indeed, a strong drawing greatly helps in the readability of the matching between G_1 and G_2, but we expect that a strong drawing has worse values for some aesthetics

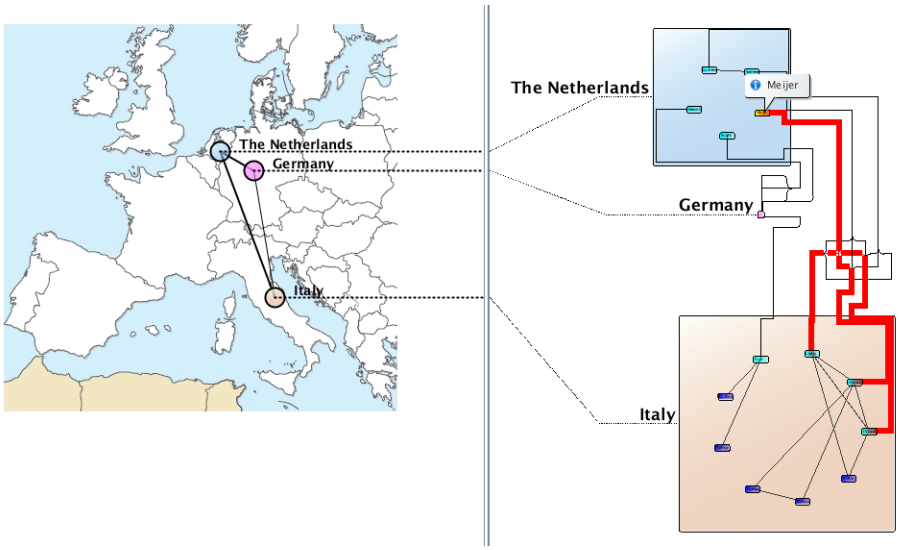


Figure 7: The same one-to-many matched graphs of Fig. 6 after some edge and vertex filtering.

(e.g., aspect ratio and number of bends) than a weak drawing.

The focus is on the drawing of G_2, because we are assuming that a drawing $\Gamma(G_1)$ of G_1 is given as part of the input or that it is computed with some classical drawing algorithm. Therefore, for the experiments we used a test suite of instances of G_2, with given number of cluster vertices. We generated 240 graphs in total, 5 graphs for each *sample*. A sample is obtained by fixing number of vertices, number of clusters, and density (number of edges/number of vertices). The number of vertices is a value in the set $\{100, 400, 700, 1000\}$, the number of clusters is a value in $\{5, 10, 15, 20\}$, and the density is a value in $\{1.0, 1.5, 2.0\}$. Each graph was generated at random, by assuming that 10% of the edges are inter-cluster edges and that 90% of the edges are intra-cluster edges. The experiments have been executed under the Windows 2003 server OS, on an Intel Pentium IV with 3.0GHz and 2GB of RAM.

The charts of the experimental results are shown in Figures 9, 10, and 11. Those about running time, drawing area, number of bends, and aspect ratio, report the average values for the two drawing algorithms as a function of the number of vertices, over all number of cluster values. We used a different chart for each measure and for each density value. Conversely, the number of crossings is independent of the drawing algorithm (the planarization step is applied before imposing the constraints for strong or weak one-to-many matched drawings), and therefore only one curve is shown in the charts.

As for the running time, the algorithm for strong drawings is slightly slower than the algorithm for weak drawings (in the average, it requires about 10%

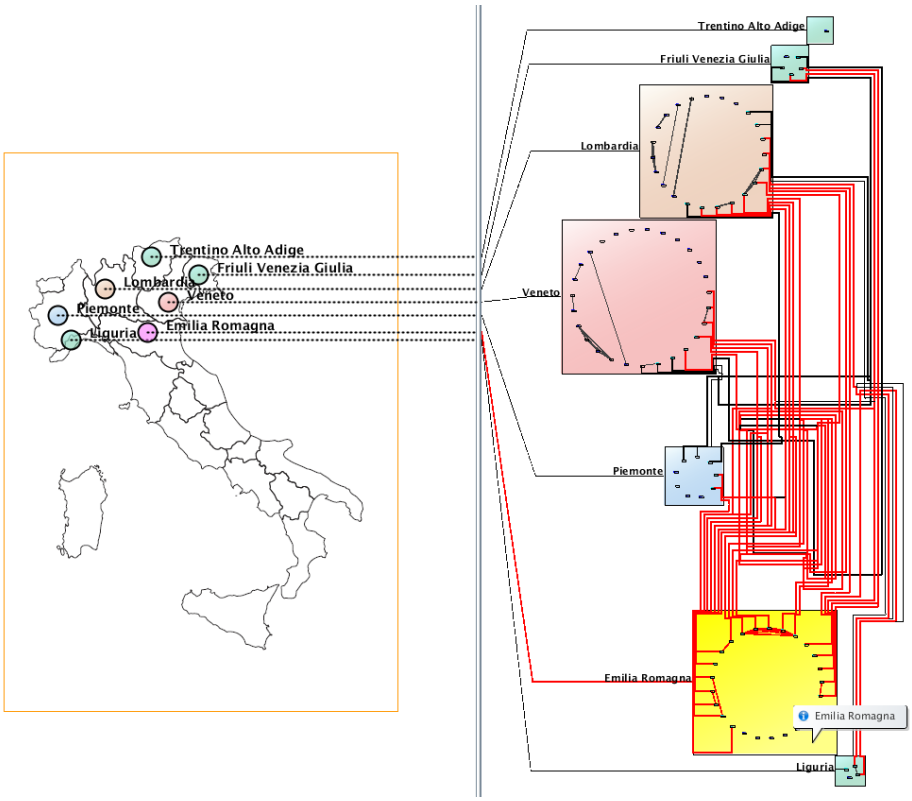


Figure 8: A one-to-many matched drawing in the contexts of Web clustering engines.

more time). In general, both types of computations take a few seconds for graphs up to 400 vertices and low density values. Graphs with the highest density and 700 vertices are computed in a few minutes, while the computations may require up to 30 minutes for the hardest instances of our test suite, i.e., graphs with 1000 vertices and density 2.0.

Concerning the area and the aspect ratio, since in a strong one-to-many matched drawing every two cluster regions are constrained to stay one below the other, strong drawings have a worst aspect ratio but smaller area than weak drawings which have typically aspect ratio close to 1.

About the number of bends, strong drawings present in the average $11-12\%$ bends more than weak drawings, which are caused by their greater number of constraints. Finally, as already observed, the number of crossings is independent of the two drawing algorithms, and as expected it rapidly increases with the graph density.

7 Conclusions and Open Problems

We presented a new system for the visual analysis of one-to-many matched graphs, i.e., graphs having secondary one-to-many relationships between their vertices. The system computes a visualization of the two graphs such that secondary relationships do not cross each other, and other important aesthetic requirements are taken into account. Furthermore, the system provides the user with several interaction functionalities for the visual analysis of the drawing. We also described the results of an experimental analysis that measures the system performances, both in terms of running time and in terms of drawing quality.

Our drawing approach combines orthogonal drawings in the topology driven approach with circular drawing algorithms, and adopts an edge bundling technique to reduce the visual complexity introduced by some links. In the near future we plan to explore other visualization paradigms for the studied problem. For example, to speed-up the system on larger instances it could be interesting to design an algorithm that combines the well-know Sugiyama algorithm with circular drawings.

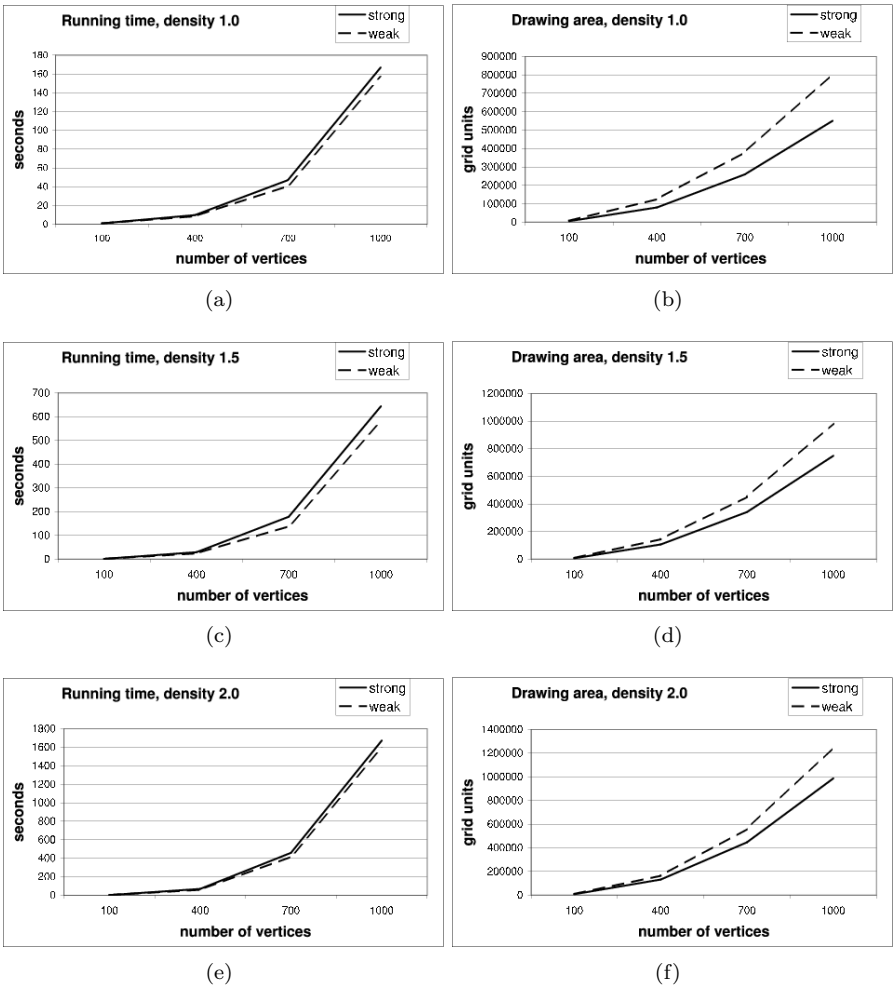


Figure 9: Experimental comparison of the strong and weak one-to-many matched drawing algorithms: Running time ((a), (c), (e)) and Area ((b), (d), (f)).

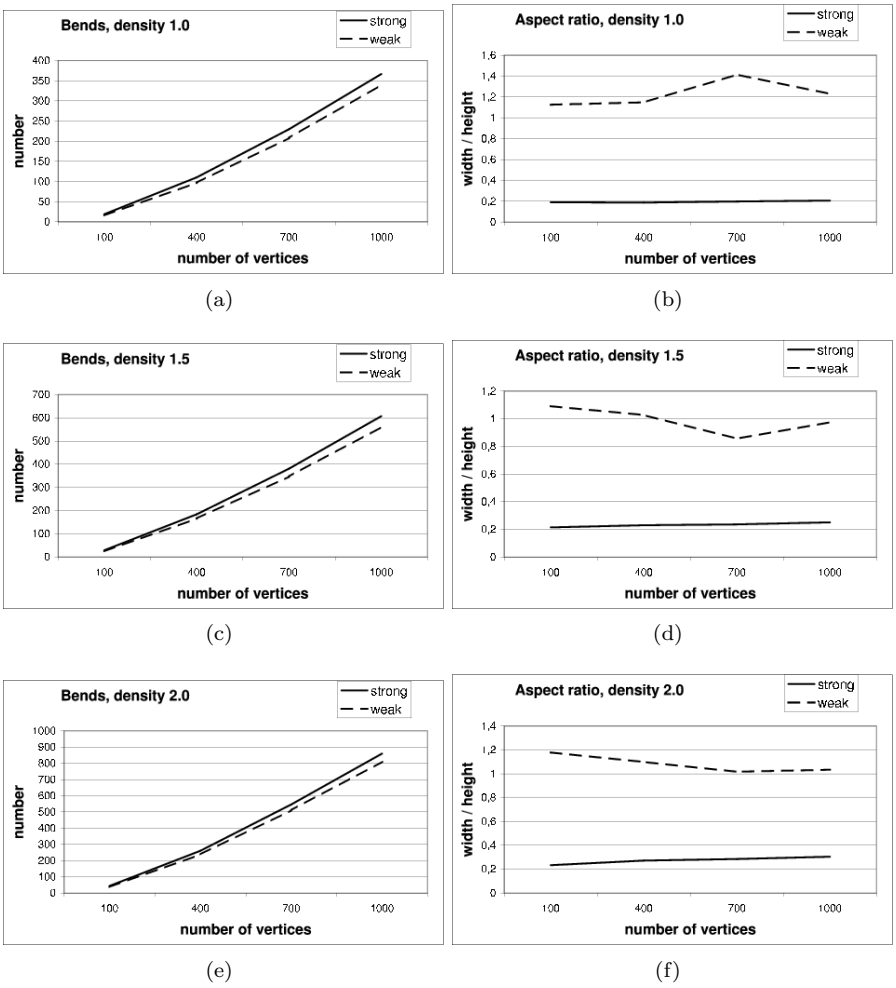
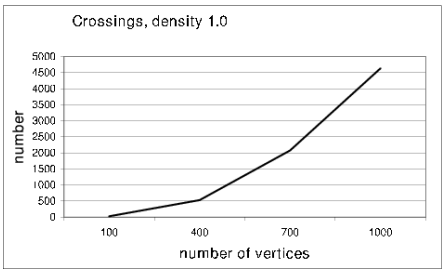
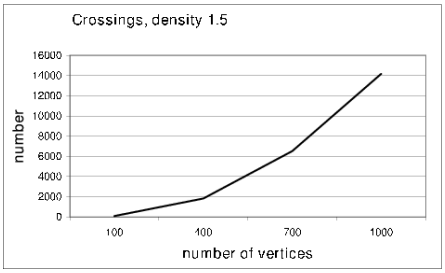


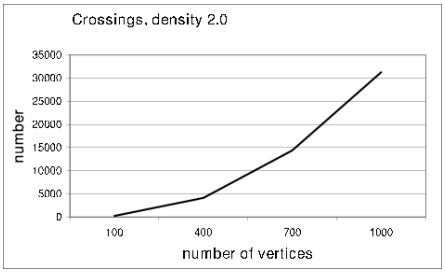
Figure 10: Experimental comparison of the strong and weak one-to-many matched drawing algorithms: Number of bends ((a), (c), (e)) and Aspect Ratio ((b), (d), (f)).



(a)



(b)



(c)

Figure 11: Experimental results of both the strong and weak one-to-many matched drawing algorithms: Crossings.

References

[1] GDToolkit: Graph drawing toolkit.
URL: `http://www.dia.uniroma3.it/~gdt/`.

[2] M. Baur and U. Brandes. Crossing reduction in circular layouts. In *WG 2004*, volume 3353 of *LNCS*, pages 332–343, 2004.

[3] M. Baur and U. Brandes. Multi-circular layout of micro/macro graphs. In *Graph Drawing (GD'07)*, volume 4875 of *LNCS*, pages 255–267, 2007.

[4] P. Braß, E. Cenek, C. A. Duncan, A. Efrat, C. Erten, D. Ismailescu, S. G. Kobourov, A. Lubiw, and J. S. B. Mitchell. On simultaneous planar graph embeddings. *Comput. Geom.: Theory and Appl.*, 36(2):117–130, 2007.

[5] C. Collins and M. S. T. Carpendale. VisLink: Revealing relationships amongst visualizations. *IEEE Trans. Vis. Comput. Graph.*, 13(6):1192–1199, 2007.

[6] G. Di Battista, W. Didimo, M. Patrignani, and M. Pizzonia. Orthogonal and quasi-upward drawings with vertices of prescribed sizes. In *Graph Drawing (GD'99)*, volume 1731 of *LNCS*, pages 297–310, 1999.

[7] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing*. Prentice Hall, 1999.

[8] E. Di Giacomo, W. Didimo, L. Grilli, and G. Liotta. Graph visualization techniques for web clustering engines. *IEEE Trans. Vis. Comput. Graph.*, 13(2):294–304, 2007.

[9] E. Di Giacomo, W. Didimo, M. J. van Kreveld, G. Liotta, and B. Speckmann. Matched drawings of planar graphs. In *Graph Drawing*, volume 4875 of *LNCS*, pages 183–194, 2007.

[10] E. Di Giacomo and G. Liotta. Simultaneous embedding of outerplanar graphs, paths, and cycles. *Intern. Journ. of Comput. Geom. and Appl.*, 17(2):139–160, 2007.

[11] N. Elmqvist, T.-N. Do, H. Goodell, N. Henry, and J.-D. Fekete. ZAME: Interactive large-scale graph visualization. In *IEEE VGTC Pacific Visualization Symposium 2008 (PacificVis 2008)*, pages 215–222, 2008.

[12] C. Erten, P. J. Harding, S. G. Kobourov, K. Wampler, and G. V. Yee. GraphAEL: Graph animations with evolving layouts. In *Graph Drawing (GD'03)*, volume 2912 of *LNCS*, pages 98–110, 2003.

[13] C. Erten and S. G. Kobourov. Simultaneous embedding of a planar graph and its dual on the grid. *Theory Comput. Syst.*, 38(3):313–327, 2005.

[14] C. Erten, S. G. Kobourov, V. Le, and A. Navabi. Simultaneous graph drawing: Layout algorithms and visualization schemes. *Journ. Graph Alg. and Appl.*, 9(1):165–182, 2005.

[15] R. Görke, M. Gaertler, and D. Wagner. LunarVis - analytic visualization of large graphs. In *15^{th} Intern. Symposium on Graph Drawing, (GD 2007)*, pages 352–364, 2007.

[16] C. Gutwenger, K. Klein, and P. Mutzel. Planarity testing and optimal edge insertion with embedding constraints. *Journ. of Graph Alg. and Appl.*, 12(1):73–95, 2008.

[17] F. Harary. *Graph Theory.* Addison-Wesley, 1972.

[18] M. Kaufmann and D. Wagner. *Drawing Graphs.* Springer Verlag, 2001.

[19] S. Masuda, T. Kashiwabara, K. Nakajima, and T. Fujisawa. On the NP-completeness of a computer network layout problem. In *20th IEEE Int. Symposium on Circuits and Systems*, pages 292–295, 1987.

[20] C. Muelder and K.-L. Ma. A treemap based method for rapid layout of large graphs. In *IEEE VGTC Pacific Visualization Symposium 2008 (PacificVis 2008)*, pages 231–238, 2008.

[21] B. Shneiderman and A. Aris. Network visualization by semantic substrates. *IEEE Trans. Vis. Comput. Graph.*, 12(5):733–740, 2006.

[22] J. M. Six and I. G. Tollis. A framework and algorithms for circular drawings of graphs. *Journ. of Discr. Alg.*, 4(1):25–50, 2006.

[23] R. Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM Journ. Comput.*, 16(3):421–444, 1987.

[24] F. B. Viégas, M. Wattenberg, F. van Ham, J. Kriss, and M. McKeon. Many eyes: A site for visualization at internet scale. In *IEEE Symposium on Information Visualization (InfoVis 2007)*, pages 1121–1128, 2007.

[25] Y. Wu and M. Takatsuka. Visualizing multivariate networks: A hybrid approach. In *IEEE VGTC Pacific Visualization Symposium 2008 (PacificVis 2008)*, pages 223–230, 2008.