Multi-level Area Balancing of Clustered Graphs

Hsiang-Yun Wu, Martin Nöllenburg, and Ivan Viola

Abstract—We present a multi-level area balancing technique for laying out clustered graphs to facilitate a comprehensive understanding of the complex relationships that exist in various fields, such as life sciences and sociology. Clustered graphs are often used to model relationships that are accompanied by attribute-based grouping information. Such information is essential for robust data analysis, such as for the study of biological taxonomies or educational backgrounds. Hence, the ability to smartly arrange textual labels and packing graphs within a certain screen space is therefore desired to successfully convey the attribute data . Here we propose to hierarchically partition the input screen space using Voronoi tessellations in multiple levels of detail. In our method, the position of textual labels is guided by the blending of constrained forces and the forces derived from centroidal Voronoi cells. The proposed algorithm considers three main factors: (1) area balancing, (2) schematized space partitioning, and (3) hairball management. We primarily focus on area balancing, which aims to allocate a uniform area for each textual label in the diagram. We achieve this by first untangling a general graph to a clustered graph through textual label duplication, and then coupling with spanning-tree-like visual integration. We illustrate the feasibility of our approach with examples and then evaluate our method by comparing it with well-known conventional approaches and collecting feedback from domain experts.

Index Terms—Graph drawing, Voronoi tessellation, multi-level, spatially-efficient layout

1 Introduction

Over recent decades, graphs have been developed to formulate relationship networks between entities. Social networks, for example, have emerged in recent years and have quickly dominated network data. Graph theory, graph drawing, and graph visualization methods have been identified as effective techniques to analyze this data [13]. Nowadays, the architecture of *knowledge graphs* [10], which are powerful representations of knowledge, can be effectively analyzed by means of graph-related computational- and visual-analytics machinery. Furthermore, the recently established scientific discipline of complexity science [39] studies the complex relationships among entities within particular structures or phenomena.

One typical example for the application of complexity science includes network structures in biology. For instance, life functions are organized in a relationship of interacting elements and chemical compounds that form a supercomplex network of reactions occurring throughout the entire life form. To semantically organize these enormous networks, they can be segmented into network sub-elements, known as *pathways*, to form a graph containing dozens of chemical elements that represent a particular function of life. However, the full collection of pathways is too large to be easily handled using common graph visualization techniques. Therefore, these complex networks are typically organized into clusters, denoted as *subsystems*, to make them easier to visually comprehend. Furthermore, because there is only a limited number of chemical elements that play a role in evolution and other roles in several pathways, molecules of water, oxygen, carbon dioxide, ATP, or NAD are so frequently interconnected that they inevitably form part of almost every pathway, making these networks very tightly connected. Thus, graphical representations of biological networks pose a huge challenge for scientific graph visualization techniques.

Applying state-of-the-art graph-drawing and graph-visualization techniques to such complex scientific networks inevitably becomes hopelessly computationally infeasible. To date, Metabopolis [64] is the only study that has attempted to algorithmically design a network layout for the entire metabolic pathway network. While their algorithm can compute a layout within a reasonable time of just over two hours, it still cannot compete with manually designed metabolic pathway diagrams in terms of visual quality [52].

Perhaps the most notable shortcoming of Metabopolis, as compared to ReconMap (a manual layout that has been compiled over several years; v3 at the time of writing), is its lack of space utilization uniformity. In Metabopolis, some subsystems are given ample space for their pathway layouts with generous space around them, while some other pathways are extremely densely packed, which severely comprises their readability. Some subsystem boxes on the next level of spatial organization are well connected with their neighboring interacting subsystems, but some create inefficient holes in the overall layout. In contrast, ReconMap3 organizes the subsystems more organically for tighter and more uniform space utilization. ReconMap3 also occasionally non-uniformly distributes the network *ink*. This is, however, by design and purposely done to communicate a particular structural motif, such as the citric acid cycle, which is rather sparse, and transport pathways, which are rather dense. Due to the uniqueness of those motifs (recognizable topological structures), some localization of detail can be performed even without recognizing the full details.

We draw inspiration from the manually crafted network design of ReconMap3 for tackling the key shortcomings of

- *H.-Y. Wu is with TU Wien, Austria. E-mail: hsiang.yun.wu@acm.org*
- *M. Nöllenburg is with TU Wien, Austria. E-mail: noellenburg@ac.tuwien.ac.at*
- *I. Viola is with King Abdullah University of Science and Technology (KAUST), Saudi Arabia. E-mail: ivan.viola@kaust.edu.sa*

Manuscript received April 19, 2005; revised August 26, 2015.

the current fully algorithmic network layout of metabolic pathways. In this paper, we address one of the major scientific graph visualization challenges, namely the uniformity of the layout. Our approach is generally applicable, however, and is especially tailored for scientific networks such as metabolic pathways. Moreover, our approach tackles the problem as a multi-layer problem with several scales of conceptual organization. The uniformity, or as we denote it, the *area balancing*, allocates nearly equal area for each textual label over the canvas as an even density of stations across a map [54]. Our approach is conceptually a top-down multi-pass process based on Voronoi tessellations, which results in a well-balanced spatial layout. Our approach also supports motif alignment and vertex duplication to resolve hard hairball-causing cases. When calculating the layout on a detailed level, the approach also takes into consideration the size of the textual labels associated with the vertices. To emphasize this association, we denote a vertex annotated with a textual label as a *vertext* (pl. *vertexts*) in this paper. In summary, our main technical contributions are:

- Multi-level area balancing
- Schematized space partitioning and overlap-free vertext arrangement
- Hairball management via vertext duplication coupled with visual integration

The remainder of this paper is structured as follows: In Section 2, we relate our new algorithm to the existing body of work. We then begin with an explanation of the design criteria for achieving area-balanced networks and provide a high-level algorithm explanation in Section 3. The technical contribution and details are presented in Sections 4-6. Several improvements in our implementation are detailed in Section 7, followed by the implemented results in Section 8, and evaluation and discussion in Section 9. Finally, we conclude this paper and sketch interesting future directions in Section 10.

2 Related Work

Our proposed approach for visualizing clustered graphs in a way that assigns a fair share of the available space to each vertext is related to several studies in the literature. We first cover the layout approach for clustered graphs, and the relevant space partitioning algorithms followed by the layout schematization approaches.

2.1 Layout of clustered graphs

Numerous approaches for drawing graphs with additional vertex grouping or clustering information exist in the literature [12], [17], [23], [38], [70]; however, most of those studies assume proper hierarchical clustering, in the sense that clusters must be either disjoint or one cluster must be contained in the other cluster. Recent surveys by Vehlow et al. [59] on visualizing group structures in graphs and by McGee et al. [46] on visualizing more generally multilayer graphs [39] give a good overview of the state of the art. A recent scalable method for drawing graphs based on stress minimization with additional layout constraints, including constraints for avoiding cluster overlap, has been presented by Wang et al. [61]. In comparison to preserving the geometric structures of node-link diagrams, set visualization techniques can be used to emphasize cluster information. BubbleSets [18], LineSets [9], and KelpFusion [47] are techniques that introduce enclosed regions around pre-placed elements to provide a stronger sense of the grouped elements.

Due to the application of the map metaphor, we are particularly interested in those approaches that use a proper (hierarchical) partitioning of the available space among the different clusters, similar to countries on a map, rather than methods that merely color vertices or use non-space-filling lines or contour overlays on top of node-link diagrams. A well-known example is GMap [31], which first draws the given graph using standard algorithms such as force-directed layout or multidimensional scaling, and then computes a flat clustering of the graph (which could also be given in the input). The vertex positions as well as a large set of additionally placed points are then used as seeds to compute a Voronoi diagram, whose cells are colored according to the cluster information. As a result, a GMap visualization can show a clustered graph in the shape of a political map with countries representing clusters. In the initial paper, vertices always belong to a single cluster, and clusters can sometimes be non-contiguous because the layout and the clustering are computed separately. A more recent journal extension later overcomes this weakness by overlaying semi-transparent clusters over each other [32]. In comparison to GMap, we incorporate a duplication scheme to simplify the graph topological structures to achieve the same goal. Kobourov et al. [41] subsequently developed a method for computing GMap-style layouts with contiguous cluster regions in the setting where either the input embedding or the clustering can be modified. MapSets [25] is another extension for showing cluster membership of vertices in a graph layout by space partitioning. However, regions in MapSets are computed for existing graph layouts that must be preserved and thus may enforce complex cluster shapes. Other works that use a map metaphor for graph layouts similar to GMap draw topographic maps of clustered graphs [33], maps of computer science [28], and GraphMaps [49].

2.2 Space-Partitioning Algorithms

Space-partitioning algorithms for graph layouts typically use a recursive partitioning scheme to assign the required space (or area) of the available drawing area to certain subgraphs, as is often needed when using a map metaphor. Treemaps [56], [57] are among the most prominent space partitioning schemes to visualize (weighted) trees by splitting the region representing a certain interior node into subregions for all its children, each of them proportional to the weight or subtree size. While the original treemaps have used rectangular subdivisions, treemaps have also been studied for non-rectangular regions, e.g., Voronoi cells [11].

A different type of space-filling graph layout represents planar graphs by *dual* planar subdivisions, where each vertex is represented by one cell or face in the subdivision and two vertices are connected by an edge if and only if the cells share a common boundary. Such layouts are related to cartograms [53], i.e., value-by-area maps of countries,

where the area of each country is proportional to some external value. Similarly, floor planning [68] in VLSI designs considers the task of partitioning the space of a circuit board into regions to place electronic components with neighborhood preferences given by a graph model. Floor planning techniques have also been applied in visualization, e.g., for browsing images using a Voronoi tesselation [14], Voronoi-based distribution of overlap-free vertex labels [66], [67] or computing word cloud layouts [19].

2.3 Network Schematization Algorithms

The cluster boundaries in our approach are simplified and schematized into octilinear polygons, where the slope of each edge is a multiple of 45°, as is commonly seen in metro maps, for example [63]. Several algorithms exist for schematizing polylines [20] and also for area- and topology-preserving schematization of polygons [15], [48].

3 Our Balancing Strategies

We use a graph model $G = (V, E)$ to represent a complex relationship network. The model consists of a set of *vertexts* $V = \{v_1, v_2, ..., v_n\}$ representing individual, named entities and the mutual entity connectivity is represented by the edges $E = \{e_1, e_2, ..., e_m\} \subseteq V \times V$. We aim to tackle more generally *clustered graphs*, where each $v \in V$ can belong to one or more clusters $c \in C = \{c_1, c_2, ..., c_k\}$, where each $c_i \subseteq V$. We can consider this as a specific type of multilayer graphs, which is a unified model for complex networks [65].

In this paper, we primarily focus on the development of (S1) vertext area balancing, which is done by integrating a (S2) schematized space partitioning algorithm together with additional vertexts arrangement. Moreover, a (S3) hairball management via vertext duplication is accommodated with a spanning-tree-like visual integration. We will first give an explanation based on our selected design principles (S1)–(S3) as a whole, followed by a step-by-step example, as shown in Figure 1.

3.1 Vertext Area Balancing (S1)

Note that our approach is decomposed into several levels in a top-down fashion, where area balancing for each level takes into account information complexity or density of the level underneath. This strategy with several refinement steps eventually leads us to a balanced vertext distribution, which takes into account connectivity information on all levels in the progressive composition. Our solution is tailored initially for metabolic networks, although its utility potential reaches beyond this specific application. Based on hand-made graphs and prior experience of box-based clustered graph visualization [64], this approach supports more free-form organic shapes of graphs to ensure the high effectiveness of the layout compaction.

This is achieved by a four-level area balancing approach to allocate appropriate space for each vertext within a cluster. The four distinct phases include (1) category-level (Figures 1(a)-(d)), (2) component-level (Figures 1(e)), (3) topology-level (Figures 1(f)-(g)), and (4) detail-level space partitioning (Figures 1(h)-(i)). This design decision has been made based on the topological properties of networks, where category-level refers to cluster properties, component-level indicates connected-component properties, topology-level represents the abstract form of sub-networks, and detail-level shows the detailed sub-networks. In practice, each level is computed by a force-based layout (see Section 4) followed by a schematization approach for simplifying the shapes of the contours (see Section 5) to accomplish detail-level vertext area balancing.

3.2 Schematized Space Partitioning and Overlap-Free Vertexts Arrangement (S2)

To achieve our goal, we establish a corresponding graph skeleton $g_s \in G_S = \{G_C, G_M, G_T, G_D\}$ for each of the four levels, **C**ategory, co**M**ponent, **T**opology, and **D**etail levels, respectively. The layout of each graph skeleton is used to guide the positioning of its belonging vertices to their expected position, in order to retain a balanced distribution. Each skeleton g_s is built individually based on the structures needed at each level, where we consider the topological properties when forming this skeleton (see definition in Section 5).

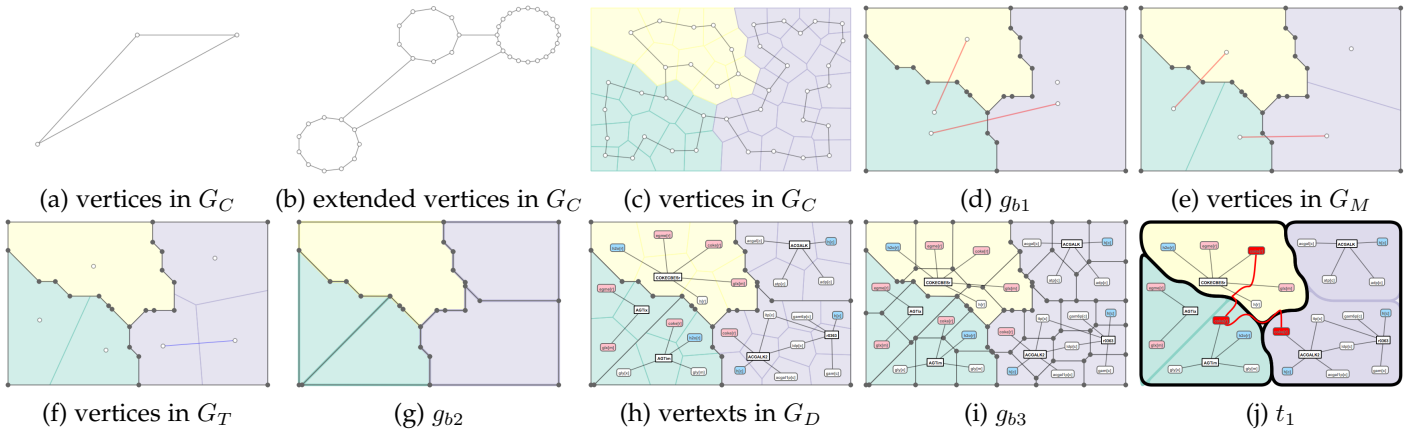
In the category-level, we aim to reserve sufficient space for each category using Voronoi seeds for estimating the appropriate space. For example, each vertex in Figure 1(a) is a representative vertex for a cluster. It consists of elements $m_{G_C} \in M_{G_C}$ that are used as a representative unit for a certain number of detail-level vertexts within a cluster (Figure 1(b)).

This is done by replacing the representative vertex for a cluster with a cycle graph, which enables the flexibility of vertices in the cycle graphs to move during the layout process.

We introduce this strategy because force-directed layout approaches are good at handling sparse and tree-like graphs. In the component-level, we drag components sharing some vertexts close to each other (see red edges in Figure 1(d)-(e)) and align cells containing subgraphs with similar topological structures in their neighborhood. In the topology-level, we again spread representative units in the sub-domains by referring to its abstract topological structures for distribution estimation.

Finally, in the detail-level, we compute the detailed layout by assigning a seed to each vertext for a Voronoi cell computation. Note that we utilize the area computed by Voronoi cells as partitions reserved for each graph skeleton. To improve the layout representation, at the end of each level, we reshape each contour polygon and simplify its boundaries for better shape identification. Adjusting polygon boundaries also allows us to pay particular attention to the textual-annotation of vertexts. Our design takes into account the size of the textual label at each vertext. This is because conventional approaches often introduce textual labels as a post-processing approach by prolonging edges horizontally or vertically to solve the problem [30], which has the drawback that users cannot control the aspect ratio of the final drawing. The algorithm may also produce an unexpected horizontally or vertically long diagram based on the input graph layout.

We therefore define our ideal layout by finding a compromise between the conventional force-directed algorithm



(a) vertices in G_C (b) extended vertices in G_C (c) vertices in G_C (d) g_{b1} (e) vertices in G_M

(f) vertices in G_T (g) g_{b2} (h) vertexts in G_D (i) g_{b3} (j) t_1

Fig. 1. These figures depict the overall pipeline of our approach. (a) A primary graph skeleton for describing a strong neighborhood relationship between clusters. (b) A collection of cycles computed from (a), which are used for estimating the required screen space. (c) The distribution of the cycles after forces are applied. (d) Simplified cluster boundaries. (e) Underlying structures that show the inter-connectivity between packed Voronoi cells, which are used to specify the center of each connected graph component. (f) Abstract graph structures are then used to estimate the area needed for the fine layout. (g) Rearranged cell boundaries. (h) Balanced detailed graph distribution after forces are applied. (i) Boundary for each vertext is then rearranged for better readability. (j) A Steiner tree representation that highlights identical vertexts.

through the blending with Voronoi centroidal forces to retain its initial graph layout. The shapes of the vertext areas are simplified and schematized to better highlight the boundaries of the components. This design is analogous to the boundaries of different districts or countries in a political map. Such a map metaphor has proved its usability in previous approaches [25], [31].

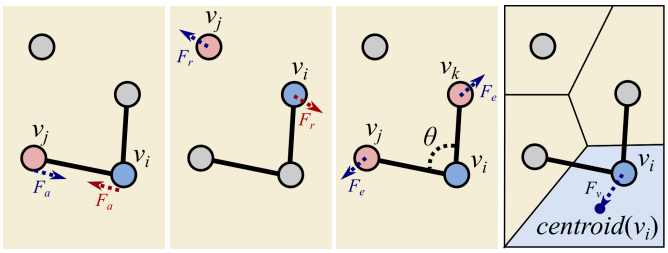
3.3 Hairball Management via Vertext Duplication Coupled with Visual Integration (S3)

In metabolic pathways, some metabolites are omnipresent and connected with almost every elementary subsystem in a pathway category, and this naturally leads to a hairball effect. Our design strategy here is inspired by hand-crafted maps by biologists [52], [55], where we perform a vertext duplication of these metabolites to reduce the graph density and mutual complexity. However, if we duplicate vertexts too much, then the graph components are oversimplified. As a consequence, tracking duplicated vertexts becomes difficult. One challenge is to find the right level of duplication that simplifies the graph just enough, but not more than necessary. The effectiveness of vertex duplication has been demonstrated by Henry et al. [35], who showed that vertex duplication is useful for community-related tasks when exploring grouping structures in a social network. Nielsen et al. [51] also demonstrated its applicability in biological networks. Vertex duplication is helpful here, since it also provides readers with a visual hierarchy of vertices using different visual variables. For example, a high-degree vertex involved in many groups could be essential in social network analysis because it shows the activity of a person who interacts strongly with other persons in the network. On the contrary, it could be less compelling in the case of molecules H_2O or H in biological networks. These molecules frequently join reactions in metabolic pathways, but due to their abundance in the cells do not provide essential biological meaning for interpretation. For this reason, we reduce the hairball effect by duplicating such vertexts. We therefore define two duplication strategies to guide readers to concentrate on important vertexts. Note that duplication also helps in reducing potential edge crossings. The details will be explained in Section 6.1.

Once the vertext duplication is being performed, we construct a graph skeleton G_C representing the connectivity of neighborhood clusters. Each vertex indicates a cluster (see Figure 1(a)), while an edge shows how strongly connected a pair of clusters are, which share vertexts in the dataset. This is done by computing a spanning tree of the clusters based on the number of shared vertexts. For example, in Figure 1(a), there exist three vertices, each representing a differently colored cluster later. We then extend the skeleton G_C (Figure 1(b)), which will be later used as seeds of Voronoi tessellation for estimating the appropriate space to embed vertexts and edges in this category. The number of vertices in a cycle (see Figure 1(b)) of each vertex in Figure 1(a) is proportional to the total pixel size of labels needed to be placed within this category (as shown in Figure 1(c)).

To introduce a vertex in the cycle as a representative of an area bounded by the corresponding Voronoi cell, we define a unit d, which represents a collection of pixels suitable for area approximation. The default value is $d = 40$ since it gives a good approximation.

Unfortunately, the vertext duplication naturally creates many more vertexts, and finding one in the network does not mean that we know all copies of that vertext. We need to formulate the means to be once again able to perceive these as a single node and thus comprehensively understand its role in the entire network. We therefore introduce a visual integration strategy to connect identical duplicated vertexts for their better identification. This is useful in the sense that highlighting the vertexts only shows the distribution of the vertexts, but does not show the connectivity of vertexts between different clusters. Since this is a different type of edge, representing a set of identical vertexts rather than the mutual connectivity of entities in the input data, we adopt a distinct visual representation. This can be modeled as a set visualization problem with a spanning tree that minimizes



(a) (b) (c) (d)

Fig. 2. Our force model, including (a) attractive forces, (b) vertex-vertex repulsive forces, (c) edge-edge repulsive forces, and (d) centroidal forces generated from Voronoi tessellation.

the distance to each of the identical vertexts. Figure 1(j) shows such an example, which allows us to visually discriminate duplicated vertexts in different clusters.

4 Vertext Area Balancing (S1)

At each designated level, we process the corresponding graph skeletons $g_s \in G_S$, as shown in Figure 1, and move the corresponding vertices or vertexts toward their expected target positions. This is accomplished through the blending of forces generated by force-directed algorithms and forces computed from centroidal Voronoi tessellation [66] for allocating the appropriate area for each vertex in G_S. The challenge of embedding a graph within an arbitrary shape is to keep vertex movement restricted within the region. The nature of *Voronoi Cells* allows us to cleverly avoid this problem by constantly moving the vertex away from the boundary of its Voronoi cell. In the following subsections, we will use detail-level vertexts as examples to describe our force model, since we apply the full set of forces incorporated in our approach to this skeleton graph.

4.1 Conventional Force-Directed Model

Once the initial overlap-free positions of the skeleton G_C have been computed [64], we employ the conventional force-directed algorithm in order to lay out the skeleton. In our implementation, we introduce attractive forces (Eq. (1)) and two types of repulsive forces, including vertex-vertex repulsive forces (Eq. (2)) and edge-edge repulsive forces (Eq. (3)).

The attractive force (see Figure 2(a)) is often formulated using Hooke's law applied on edges $e = (v_i, v_j) \in E$ (we also use the abbreviated form $e_{ij} = (v_i, v_j)$) as follows:

$$F_a(v_i, v_j) = k_a(\|v_j - v_i\| - l_0)(v_j - v_i), \tag{1}$$

where l_0 represents the ideal length of the spring, which is estimated using the ideal average pair-wise distance, as detailed in Section 7. The user-defined constant k_a controls the magnitude of the forces.

The vertex-vertex repulsive force (see Figure 2(b)) is assumed to have electrical charges on vertices so that we can keep minimum distances between them. The force is thus defined as:

$$F_r(v_i, v_j) = -k_r(v_j - v_i)/\|v_j - v_i\|^2, \tag{2}$$

where k_r corresponds to the magnitude of the electric forces and is set to be 1000.0 by default in our system. The value is decided by not adding excessively strong forces relative to the attractive force.

The edge-edge repulsive force (see Figure 2(c)) is similarly defined by adding forces generated from edges (v_j, v_i) and (v_i, v_k) that share the same end vertex v_i in order to push away those edges with a sharp angle θ [45]. This is computed using:

$$F_e(v_i, v_j, v_k) = \left(k_c\left(\arctan\frac{\|v_j - v_i\|}{c_0} + \arctan\frac{\|v_k - v_i\|}{c_0}\right)\right)\begin{pmatrix}0 & -1\\ 1 & 0\end{pmatrix}\cdot u + \left(k_e\cot\frac{\theta}{2}\right)\begin{pmatrix}0 & -1\\ 1 & 0\end{pmatrix}\cdot u \tag{3}$$

where k_c and k_e correspond to the magnitudes of the forces from edge lengths and the forces from the angle θ intersected by edges e_{ij} and e_{ik}, respectively. The value c_0 is the ideal length for edges e_{ij} and e_{ik}, which is computed similarly as l_0. The matrix $\left(\begin{smallmatrix}0 & -1\\ 1 & 0\end{smallmatrix}\right)$ is the rotation matrix of $\pi/2$ and u is the unit vector of $\left(\frac{v_j - v_i}{\|v_j - v_i\|} + \frac{v_k - v_i}{\|v_k - v_i\|}\right)$.

4.2 Centroidal Voronoi Tessellation

Our key idea to balance the area for each vertext is to apply a centroidal Voronoi force to distribute vertexts evenly within a finite polygonal region, while keeping vertexts within this region. This also allows us to potentially eliminate unwanted overlaps between text labels [66]. We accomplish this by partitioning the space through the following steps and move each vertext to its corresponding Voronoi centroid:

1) Compute the Voronoi tessellation using predefined seed points.
2) Crop the Voronoi cell by the polygonal contour.
3) Determine the centroid $\mathrm{centroid}(v_i)$ of each cropped Voronoi cell.

In Step 1, we compute the Voronoi tessellation [40] by referring to the coordinates of each $v_i \in g_s$ as a seed point. Next, we crop the Voronoi cell with the polygonal shape of the cluster one level above in the hierarchy. Alternatively, a hardware-assisted algorithm [37] can be used to approximate the effect of the geometric algorithm by plotting 3D cones of different colors at the given seed points and projecting them onto the frame buffer. In our implementation, we use the first approach for the ease of extracting the polygonal contour from the Voronoi diagram. The centroidal force to each node $v_i \in G_S$ is computed as:

$$F_v(v_i) = -k_v(v_i - \mathrm{centroid}(v_i)), \tag{4}$$

where k_v is a constant that determines the magnitude of the centroidal force. The value is determined by seeking a balance between the centroidal force and other forces to normalize the weights w_f and w_v in Eq.(5). These aforementioned steps are repeatedly computed until all seed points reach their equilibrium position.

Until here, we sum up these attractive and repulsive forces from Sections 4.1 and 4.2 to each vertext v_i as:

$$F_s(v_i) = w_f * (\sum_{e_{ij} \in E} F_a(v_i, v_j) + \sum_{j \in V - \{v_i\}} F_r(v_i, v_j) + \sum_{e_{ij}, e_{ik} \in E} F_e(v_i, v_j, v_k)) + w_v * \sum_{i \in V} F_v(v_i). \quad (5)$$

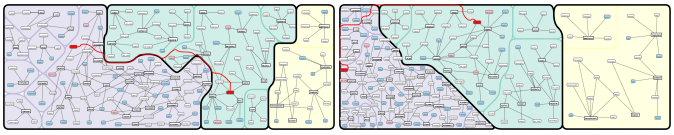
This force is applied to each node until the network layout achieves an equilibrium state. With repulsive forces, we can empirically avoid unexpected visual clutter, such as edge crossings, and with centroidal Voronoi forces, we can balance the area for individual vertices. In our approach, the above steps are iteratively computed until we find a reasonably uniformly partitioned space. Note that the additional force $F_o(v_i, v_j) = -k_o(v_j - v_i)/\|v_j - v_i\|^2$ is adaptively added if the two vertexts unexpectedly overlap. Further note that F_o is only applied on vertexts in the detail-level graph G_D, since we consider the vertex overlaps in other levels will be less influential on the final layout. The weight w_v is increasing toward detail-level since this helps us to find a better vertext distribution.

TABLE 1
A summary of the default parameter settings in our force model.

	w_f	w_v	k_a	k_r	k_c	k_e	k_v	k_o
Fig. 1(c) G_C	0.3	0.7	0.1	1000.0	0.5	0.5	1.0	0.0
Fig. 1(e) G_M	0.1	0.9	0.1	1000.0	0.5	0.5	1.0	0.0
Fig. 1(f) G_T	0.1	0.9	0.1	1000.0	0.5	0.5	1.0	0.0
Fig. 1(h) G_D	0.1	0.9	0.1	1000.0	0.5	0.5	1.0	10.0

Table 1 provides a summary of default parameters used in our system, where we strengthen the centroid forces in the last step to evenly distribute the vertexts.

Practically, F_a and F_r are conventional forces to preserve the distance between connected vertices, and F_v influences the overall balance of the area preserved for each vertex. Figure 3 shows an example of how F_v will influence the final visualization. The forces F_c, F_e, and F_o have a stronger impact on the final rendering by retaining angular resolution and eliminating vertext overlaps. We study the effect of these parameters in the supplementary materials (see Section 11.2).

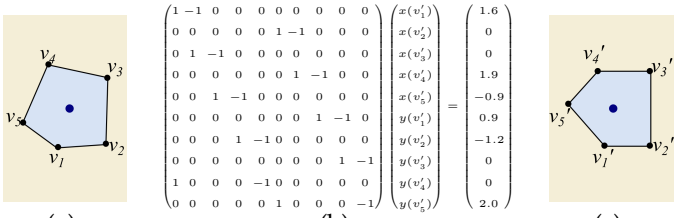


(a) With F_v (b) Without F_v

Fig. 3. An example of how the canvas is partitioned by incorporating (a) with all forces, and (b) without F_v.

5 Schematized Space Partitioning and Vertext Position Arrangement (S2)

As described previously, the constrained forces for vertex area balancing are computed in four levels, including (1) a category-level, (2) a component-level, (3) a topology-level, and (4) a detailed-level computation.



(a) (b) (c)

Fig. 4. An example of (a) an input boundary graph G_B, (b) its associated $\mathbf{AV}' = \mathbf{b}$ with constraints (O1), and (c) the corresponding output.

5.1 Representative Graph Skeletons

Note that at each level, we generate a unique graph skeleton $g_s \in G_S = \{G_C, G_M, G_T, G_D\}$ for guiding corresponding area allocation by applying the technique described in the previous section.

5.1.1 *Category-level Skeleton (G_C)*

Since important vertexts shared between clusters are duplicated to form a hierarchically clustered graph, those clusters sharing vertexts are expected to stay close to each other to avoid scattered vertexts. To achieve this, we prepare a spanning subgraph G_C for maximally retaining the neighborhood relationships of pairs of clusters. We first apply the same strategy as in Metabopolis [64]. This allows us to create a spanning subgraph, which is planar, so that the corresponding planar embedding can be used as the initial positions of the cluster centers (see Figure 1(a)). We then explode the vertices of this spanning subgraph into a sequence of cycle vertices, where the number of vertices here are computed to be proportional to the total pixel size of labels within the corresponding categories (Figure 1(b)).

5.1.2 *Component-level Skeleton (G_M)*

After the vertex duplication, some subgraphs can become disconnected. We first need to distribute these components without overlaps, so that subgraphs will not be drawn over others unexpectedly. Additionally, in order to reduce the readers' workload when tracing duplicated vertexts, we move those components sharing similar important vertexts in different clusters close to each other by adding cross-cluster edges to the graph. Those edges are marked in red in Figure 1(d)–(e).

5.1.3 *Topology-level Skeleton (G_T)*

Here, we aim to shape the area to fit the topological structure of a component. For example, if a graph contains a circular structure, a round area is more suitable for the layout. However, this may not be suitable for a chain graph since it can be embedded better within a long horizontal or vertical rectangular area. A graph skeleton G_T is used to support this idea. For each component, we compute its representative topological structure using the *Markov Cluster Algorithm* [5], [21]. The graph G_T is built upon the connectivity of those clusters, which is used as a representative unit of the entire subgraphs. Of course, we can also apply the *Edge Sparsification* technique [24], or *Motif Simplification* [22] to achieve the same goal.

Additionally, to efficiently recognize some similar topological patterns in the components, we also place subgraphs with similar topological structures close to each other. In our system, we detect and connect pair-wise isomorphic components, but users can also define their own similarity measure for evaluating two subgraphs. Since we consider each component as an independent object, we need to add secondary edges to guarantee that they are expected neighbors in the graph skeleton G_T. This is done by adding random edges between similar vertices while keeping the resulting graph planar.

5.1.4 Detailed-level Skeleton (G_D)

At this final level, we simply treat vertexts in the clustered graph as individual vertexts in the skeleton G_D, which allows us to finalize the vertext position of each vertext through the blending of Centroidal Voronoi force.

5.2 Layout Frame Schematization

Although Voronoi tessellation allows us to find a partitioning of a region, where each cell consists of all points closer to its seed than to any other seed, the boundary of such a cell has a unique, irregular shape. This increases the visual complexity, since it is still difficult to recognize the polygonal contours extracted from a group of cells. In this section, we aim to rearrange and simplify the boundary formed by Voronoi cells to provide the readers with clean and easily traceable region patterns for better memorability. We incorporate three constraints here. This includes (O1) octilinearity, to arrange edge orientation to a limited set of angles, (O2) relative positioning, to maximally retain the initial boundary positions, and (O3) overlap removal, to remove overlaps between vertexts and the schematized boundary. To achieve this, we first construct a boundary graph G_B. We take boundary edges extracted from the Voronoi cells into consideration and build a simple network that covers all edge segments along the cluster boundary. This graph G_B is necessary since it will then later be used to bundle the edges that are used to show the set of duplicated vertexts without occluding text labels. We use the following three constraints.

5.2.1 (O1) Octilinearity

Our primary goal here is to rearrange the edges in G_B so that the edge orientation is constrained to either horizontal, vertical, or diagonal at 45 degrees, also known as octilinear (or octolinear) orientations. This is achieved by minimizing the angle difference between the current edge angle and the target octilinear angle θ_e. Note that the target angle $\theta_e \in \{0, 0.25\pi, 0.5\pi, 0.75\pi, \pi, 1.25\pi, 1.5\pi, 1.75\pi\}$ is precomputed from the input edge $e_{ij} \in G_B$. The constraint can be formulated as:

$$\Omega_{o_1} = \sum_{e_{ij}=(v_i,v_j)\in E} |(v'_i - v'_j) - \mathrm{Oct}(v_i - v_j)|^2, \tag{6}$$

where $v'_i, v'_j \in V$ represents the unknown ideal output coordinates that need to be computed, and Oct is a function that rotates the edge $e_{ij} = (v_i, v_j)$ in the input graph to its closest octilinear direction θ_e.

5.2.2 (O2) Relative Positioning

To maximally retain the balanced partitioning from the Voronoi cell computation, we want to avoid a drastic change from this partitioning after schematizing G_B. Therefore, the relative position of vertexts should play a role in determining where to place a vertext. This is done by minimizing the distance between the input vertext v_i and the expected output vertext v'_i, and the energy term is defined as follows:

$$\Omega_{o_2} = \sum_{v'_i \in V'} (v'_i - v_i)^2. \tag{7}$$

5.2.3 (O3) Overlap Removal

The final constraint aims to remove the overlaps between vertexts and boundary edges $e \in G_B$. The idea is to retain a minimal distance between a vertext and its corresponding Voronoi cell. This is done by giving a soft penalty to the term:

$$\Omega_{o_3} = \sum_{v_i \in G, e(v_j,v_k)\in G_B} |(v'_i - p'_{jk}) - \delta(v_i - p_{jk})|^2, \tag{8}$$

where $\delta = \frac{\epsilon}{v_i - p_{jk}}$. The value ϵ is the minimum distance between vertext v_i and edge e_{jk}. We set this value to be equal to half the text label width, in order to maintain some distance between a vertext and its cell boundary. The point p_{jk} is the closest point on the edge e_{jk} to vertext v_i. This is computed at each iterative step.

In summary, the objective function is defined as:

$$\Omega = w_{o_1}\Omega_{o_1} + w_{o_2}\Omega_{o_2} + w_{o_3}\Omega_{o_3}, \tag{9}$$

where w_{o_i} indicates the weight for each energy term. Based on our empirical experiences, we assign the weights as $w_{o_1} = 10.0$, $w_{o_2} = 0.00001$, and $w_{o_3} = 20.0$ by default. Readers can refer to [58], [62] for some similar implementation details.

During this optimization process, our system tends to find the ideal aligned position of vertext $v \in V(G_B)$. Figure 4 gives an example of how the optimization is computed, where $v_1 = (1.0, 1.0)$, $v_2 = (-0.5, 1.6)$, $v_3 = (-1.2, -0.2)$, $v_4 = (-0.2, -1.0)$, and $v_5 = (1.0, -1.0)$ are our input vertices. The constraints in Eqs. (6), (7), and (8) can be transformed into a linear system $\mathbf{AV'} = \mathbf{b}(\mathbf{V'})$, where A is a coefficient matrix, and V' collects all vertext output coordinates. The constraints we have here are more than the number of variables, so the matrix is overdetermined. Therefore, the problem is then solved by using $\mathbf{V'} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b'}$. In our system, we incorporate the conjugate gradient optimization technique [36] to minimize our objective function Ω. Another benefit of using a conjugate gradient is that it allows us to iteratively solve this problem. We therefore check the overlaps of text labels by examining the distance between label centers to edges e in G_B, and introduce necessary constraints to penalize (O3) at each iterative step. This allows us to not include all constraints from the beginning, thus avoiding unnecessary computational complexity.

6 Hairball Management via Vertext Duplication Coupled with Visual Integration (S3)

In this section, we explain how we transform a real-world relationship into a clustered graph so that we can create

multiple graph skeletons at different levels for finding appropriate vertext areas as described in the previous subsections.

6.1 Vertext Duplication and Spanning Subgraphs

The goal of visualizing a clustered graph is to arrange vertices in the same cluster close to each other for better identification. Clusterings can be automatically generated [50], or predefined by domain experts in biology, medicine, or finance etc. Clusters, often highlighted using colors associated with the underlying subgraph, provide additional insight into the data. For instance, in biology, a reader can associate relationships with corresponding functional groups or cellular compartments. However, clusters in clustered graphs are by definition either disjoint or one cluster contains the other [27]. Such clusters sometimes cannot cover the properties of real-world relationships [65]. Vertext duplication is an intuitive way to transform overlapping clusters into disjoint clusters and is an alternative scenario to remove mutual edge crossings for better visual quality. This strategy is commonly applied in biological [52] and social network visualization [35]. We apply two strategies for vertext duplication by defining the importance of vertexts in the datasets.

Besides umimportant vertexts, the remaining high-degree vertexts are denoted as important. We only duplicate an important vertext if it belongs to multiple clusters. Moreover, each important vertext is unique to each component. On the other hand, a less informative vertext, such as H_2O in metabolic pathways, will be fully duplicated for each incident edge to reduce the visual complexity induced by edge crossings. The importance of a vertext can be either determined from a list given by the user or derived from a threshold on the degree.

6.2 Spanning-Tree Visual Integration

Since a vertext in our input graph can belong to multiple categories, we declutter the visual complexity originating from this property by duplicating vertexts to set up a clustered graph for better readability. Once the vertext duplication is applied as described in the previous subsection, we naturally increase the number of vertexts, which also complicates reading in the sense that a vertext is not unique anymore. To alleviate the problem of finding all copies of a vertext, our system allows users to connect all vertext instances using a spanning sub-tree computed from G_B, beyond simply highlighting vertexts using colors. With the use of the graph G_B, we can avoid drawing edges over any vertext, and with a spanning sub-tree, we can also show the connectivity of the vertext between different clusters. The underlying problem is formulated as a *Steiner tree problem*, which finds an optimal tree of minimum weight for a given set of vertexts in a graph. This problem differs from the *minimum spanning tree problem* in the sense that not all vertexts will be included and finding such a set with minimum weight is an NP-complete problem. To solve this problem, we use a greedy algorithm for computing the Steiner tree [42] for all instances of an identical vertext. When a user selects a target vertext, the system will compute the corresponding set of duplicates and highlight them using user-specified or determined colors (see Figure 1(j)).

7 Implementation and Force Approximation

In this section, we present experimental results of the proposed approach on synthetic and real-world datasets, together with discussions of the present approach. Our prototype system has been implemented on a desktop PC with Quad-Core Intel Xeon CPUs (3.7GHz, 10MB cache) and 12GB RAM, and the source code was written in C++ using GSL for numerical computation, OpenGL for graphics, and GLUI library for the user interface. The source code was written in C++, and the graphics rendering and user interface were implemented using the Qt library [6]. The primary graph data structure was developed on the Boost Graph Library [1], and CGAL [2] was used for computational geometry algorithms, such as computing Voronoi diagrams. Eigen [3] is used to perform matrix computation for optimization and the Micans package is used for fast graph clustering algorithms [5], [21]. The source code for our system is available on github https://github.com/yun-vis/KeiRo.

7.1 Estimation of Screen Size and Initial Settings

It is often tricky to decide how big a canvas we need for embedding a graph with vertexts and their specified aspect ratios. Our approach estimates the canvas size as $D = (\sum_{v_i \in G} f(v_i)) * (|e|^R/|v|+1)$, where $f(v_i)$ represents the number of pixels of a vertext. R is a user-defined constant that determines the size of a region dedicated to drawing edges, by default set to 1.3. Users can also specify an aspect ratio $r:1$ to fully control the diagram in our approach. Since we use Voronoi cells to allocate a balanced area for each vertext, all vertexts are expected to be uniformly distributed over the entire screen space. Based on this assumption and the aspect ratio r, we compute our ideal edge length of a graph skeleton G_S using $l_0 \approx \sqrt{0.5 * \text{region}(G_S)/n}$, where $\text{region}(G_S)$ returns the bounding region of G_S.

Since our graph is not simply embedded inside a rectangular domain but an arbitrary polygonal domain, we have to guarantee that no vertext will move outside of the domain. Our Voronoi centroidal force directly solves this problem since it always provides forces to move the vertext away from the boundary. However, we are not this lucky when assigning the initial positions of vertexts since we cannot guarantee that the centroid of a polygon is always located inside the polygon. We thus try to perturb the centroid several times (100 by default), and select the one with the maximum radius to the boundary of the polygon. In this case, we can initialize the vertext positions within this circle using conventional layout approaches.

7.2 Perturbation in Simulated Annealing

Nevertheless, conventional force-directed algorithms are known for their low computational efficiency and less flexibility to escape poor local minima. To avoid this, we also implemented a spatial quadtree subdivision technique and a temperature function to enable vertext perturbation when applying the force-directed algorithm. As described previously in Section 4, the force-directed layout is simulated by finding an equilibrium state of a physical system, where the attractive and repulsive forces applied on vertexts are balanced. Even though in our formulation we do not place vertexts fully randomly at the initial steps, but rather slightly

move the vertexts to their preferred boundary edges, some vertexts could nevertheless still fall into local minima. To avoid this, a temperature function from simulated annealing is introduced to perturb vertexts to escape this situation. This temperature constrains the dynamics of moving vertexts as the temperature decreases, which is captured as the simulated iteration counter iter is increased:

$$\mathrm{Decay} = 1.0 - \mathrm{MinTemperature}^{1.0/\mathrm{iter}}, \tag{10}$$

where $\mathrm{MinTemperature}$ indicates the lower limit of the system temperature.

For simplicity, we assume the mass of all charged vertexts is $m = 1$, so the acceleration of a vertext at time t is then computed by Newton's law $a_t = F_s/m$. In our implementation, we use velocity Verlet integration [60] to calculate the next position of a vertext $v_t(i)$ as:

$$v^{(t+1)}(i) = 2v^{(t)}(i) - v^{(t-1)}(i) + a^{(t)}\,\Delta t^2, \tag{11}$$

where $v^{(t+1)}$, $v^{(t)}$, and $v^{(t-1)}$ represent the next, current, and previous time steps, respectively. After introducing velocity attenuation Decay, the equation becomes

$$v^{(t+1)}(i) = v^{t(i)} + \mathrm{Decay} * (v^{t(i)} - v^{(t-1)}(i)) + a^{(t)}\,\Delta t^2. \tag{12}$$

The time step of simulation is defined as $\Delta t = 1$. The idea of the simulated annealing algorithm allows us to gradually control vertext movement during the process. The system has a higher temperature initially, which enables faster movement to escape local minimum. This magnitude of the movement decreases as the iteration counter increases, until the final layout is obtained.

7.3 Force Approximation using Quadtree Subdivision

Computing repulsive forces is also computationally expensive because it requires examining all pairs of vertices. Thus, our solution is to subdivide the screen space into a quadtree for faster computation using Barnes-Hut approximation [71], as done in earlier force-directed approaches [29], [34]. The idea is to aggregate vertices with large distances and compute repulsive forces from these aggregated vertices. This will improve the time complexity to $O(n \log n)$ for well-distributed vertex positions.

7.4 Continuous Curvy Contours and Curvy Paths

To improve the visual quality of the layout, we incorporate Chaikin's corner cutting algorithm [16] to generate curvy contours and curvy paths in the final visual representation. The idea of this algorithm is to smoothen the corners of a polygon or a polyline by cutting the corners off the original one. The smoothness of Chaikin curves depends on the sample points on the contours and the recursive steps that add intermediate points. We first re-sample the points on a contour, so that sample points are evenly distributed to avoid sharp artifacts. Then we use Chaikin's algorithm to refine the corner points by adding intermediate points. The underlying process is based on iterative improvement until the distance between two adjacent sample points is below a predefined threshold.

8 Results

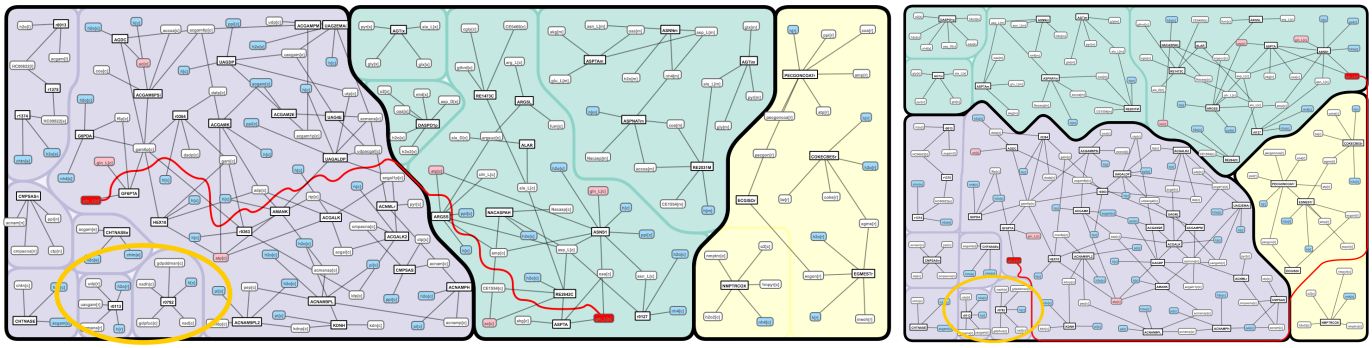
In this section, we demonstrate the feasibility of our approach by visualizing the four datasets **Small Pathway**, **Recipe**, **Metabolic Pathway**, and **KEGG Overview Pathway**. The scale of the datasets ranges from small to relatively large, as shown in Table 2.

Small Pathway. In Figure 5, we present a small set of metabolic pathways to demonstrate the balanced layout generated using our approach. Biological pathways are chains of biochemical reactions within cells. Such pathways often include proper classifications (e.g., subsystems) of reactions, such as *Transferase* together with their associated metabolites, such as *ATP* or *glucose*. Some metabolites are often involved in multiple subsystems, which visually complicates the diagram and thus leads to an unwanted hairball effect.

Our technique allows us to focus on important metabolites, for example, *ADP* (pink), while fully duplicating less informative molecules such as H_2O and H (cyan). Thus, users can select and trace in which reactions the *Glutamate* (red) is involved. In both examples in Figure 5, we see that the *Glutamate* connects to *ASNS1*, *ASPTA*, and *GF6PTA* in *Alanine and aspartate metabolism* and *Aminosugar metabolism*, respectively. Glutamate is shared, so that its containing subsystems are positioned as neighbors. Figures 5(a) and 5(b) show the results with a different aspect ratio of $r = 8/3$ and $r = 4/3$, respectively. While similar in both images, vertexts *r0113* and *r0782* (highlighted in yellow in Figure 5) are placed side by side within the purple region due to their identical topological structure.

Metabolic Pathway. Figure 7(b-4) gives an example of major pathways as a subset of human metabolism in ReconMap [52]. To generate a clustered graph, we first performed our duplication scheme to guarantee that each vertext appears in one cluster only, which are highlighted in differently colored regions. This successfully relaxes the topological structure as well as the hairball effect induced by the H_2O molecules that connect to almost all reactions in this dataset. After computing our layout, the vertexts are nearly uniformly distributed over the screen space and small connected components are pushed and aligned at the corner (see top-right corner in Figure 7(b-4)). *Glutamate* is again highlighted here and demonstrates that those subsystems involving *Glutamate* are placed in the neighborhood. Since our spanning subtree is computed based on the boundary generated by Voronoi cells, we successfully avoid drawing paths over other vertexts and avoid introducing further unwanted visual complexity.

Recipe. This dataset originates from the Graph Drawing Contest 2019 [4] and includes 151 popular food recipes extracted from the *TheMealDB database* [7]. The extracted recipes come from 11 countries, including USA, Britain, China, etc., and in total include 297 ingredients, such as *Flour*, *Onion*, *Egg*, and so on. We construct our input graph by connecting the recipes with their associated ingredients and assign a country id as clusters to each of the recipes. In the end, the graph includes 448 vertexts and 1377 edges (Table 2). Figure 7(c-4) shows the resulting diagram generated by our approach. The white rectangular labels are recipe vertexts and the rounded rectangular labels are ingredient



(a) (b)

Fig. 5. A collection of pathways in human metabolism, including *Alanine and Aspartate Metabolism*, *Alkaloid Synthesis*, and *Aminosugar Metabolism*. Each of the clusters, or so-called subsystems in metabolism, is highlighted in different colors in the diagram. White rectangular labels represent biochemical reactions, and rounded labels are metabolites involved in the reactions. Pink vertices indicate important vertices, such as the metabolite *ATP* carrying energy in *Alanine and Aspartate Metabolism*, and cyan vertices are the duplicated less important metabolites, such as H_2O or H_2, which are involved in most of the reactions in human metabolism. The **Small Pathway** includes 52 reactions and 117 different metabolites before duplication, where each reaction belongs to one of the three subsystems. The red route here indicates a highlighted metabolite appearing as a duplicate in multiple subsystems. Our system allows users to specify an input aspect ratio such as (a) $r = 8/3$ and (b) $r = 4/3$.

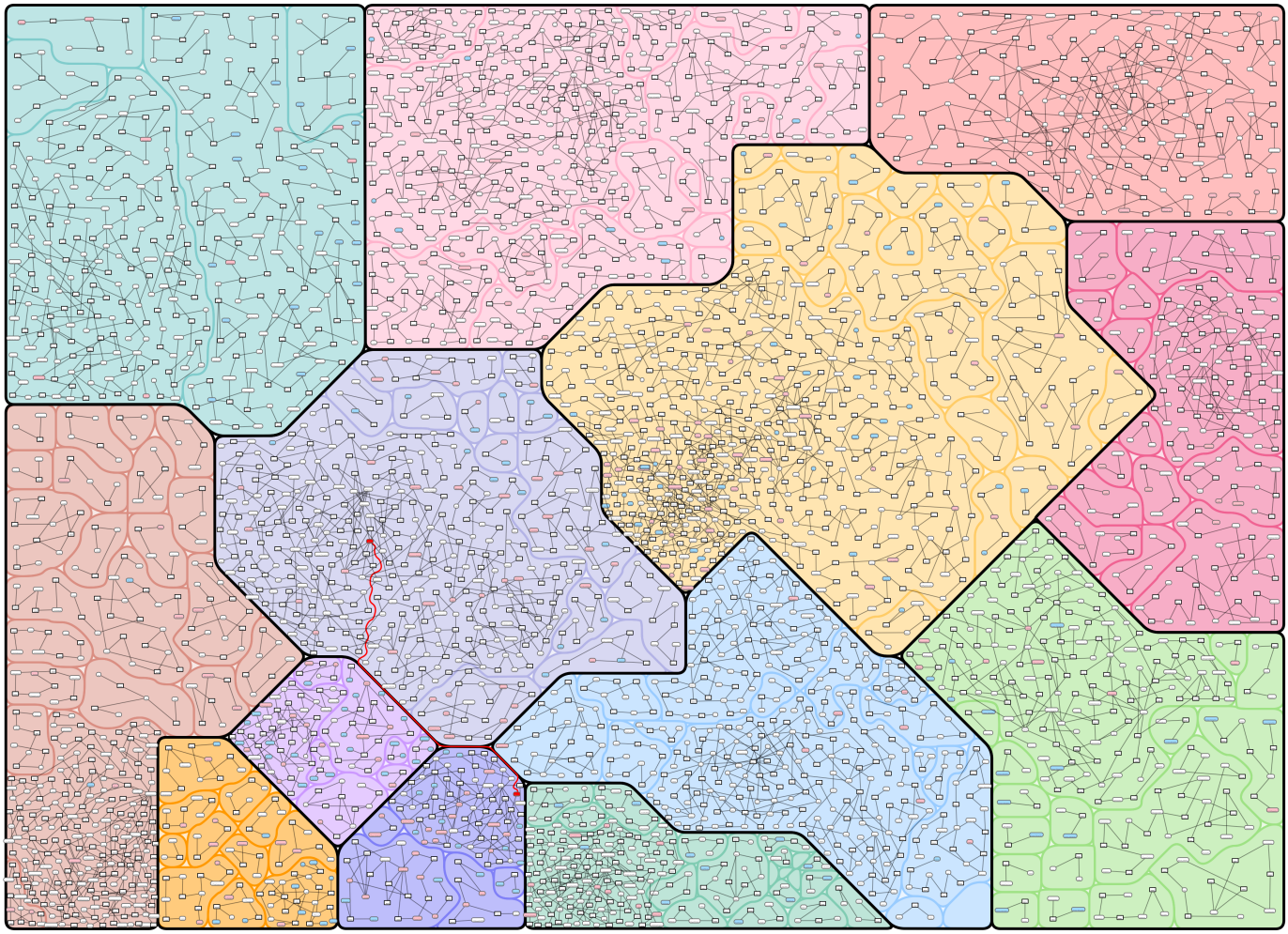


Fig. 6. Redrawing the human metabolic pathway map of KEGG [43] using our approach in comparison to the diagram in Metabopolis [64]. The present aspect ratio is $r = 4/3$.

vertexts, similarly to the biological pathways. The cyan vertexts indicate fully duplicated common vertexts and the pink ones are important vertexts that users may want to focus on. We assume that frequently used ingredients are less informative since they can be easily found in several local areas.

To produce authentic food, we can mainly focus on country specific ingredients (white, rounded vertexts) in this case. The threshold between important and less important vertexts is set as 9 here, based on the frequency of ingredient usage in the dataset. From the results, we can see that many British recipes are contained in the database. Compared to other countries, ingredients in British recipes (yellow) are relatively strongly connected, since after vertext duplication, apart from *Vegetarian Chili* (bottom-right of yellow region), the subgraph is still a single connected component.
Besides commonly used ingredients, ingredients of Italian recipes (green) are relatively independent from each other. In Figure 7(c-4), readers can see that there are more individual star-liked structures. Those recipes have a tendency to be considered as home cooking, for which we do not need specific ingredients. As an example, the red highlighted Steiner tree here connects the common ingredient *Soy Sauce* vertexts in different countries (if any). It is not surprising that Chinese (purple) and Japanese (pink) recipes often use *Soy Sauce* as primary ingredients, while it is absent from most other countries. Nonetheless, one outlier (*Mushroom and Chestnut Rotolo*) exists in British cuisine. The map we created using this dataset won the 1st Place Award in the 28th Annual Graph Drawing Contest [44].

KEGG Overview Pathway. Figure 6 is the result of reproducing the KEGG overview pathway map using our approach. The color coding of the category here is directly retrieved from the original KEGG database [43], as similarly incorporated in Metabopolis [64] (see figures in supplementary materials). We also set the same threshold for specifying unimportant vertices as in Metabopolis, so that readers can refer to the paper for comparison (Figure 6). The advantage of this technique allows us to arrange the vertexts in a balanced fashion by pushing vertexts away from each other. This initially gives users an idea of how big each category is, and explicitly shows which reaction is classified under which category. Users can also identify sub-components effectively since those components with identical topological structures are aligned as neighbors. This also helps users to comprehend which structures are associated with certain types of pathways, such as small chains, stars, etc.

9 Evaluation and Discussion

In this section, we measure and evaluate the space coverage of the results generated by our implementation, and we report the running time for each image shown in this paper. Table 2 summarizes the properties of our datasets, where the number of vertexts, edges, clusters, and the corresponding graph densities are noted as V, E, C, and *Den*, respectively. The notation with subscript $_D$ refers to these numbers after vertext duplication. We use the same graph density function for *Den* and Den_D, which is defined as $Den = |E|/(|V|^2 - |V|)$ [69]. We multiply this ratio by 100 for simplicity in Table 2.

TABLE 2
The number of nodes ($|V|$), edges ($|E|$), and density (*Den*) before and after node duplication, while $|C|$ shows the number of clusters.

	Before duplication/After duplication			
	$\|V\|/\|V_D\|$	$\|E\|/\|E_D\|$	Den/Den_D	$\|C\|$
Fig. 5	169/211	223/260	0.79/0.59	3
Fig. 7(b-4)	593/948	1244/1635	0.35/0.18	11
Fig. 7(c-4)	448/1377	1618/3236	0.80/0.17	11
Fig. 6	3679/3832	4008/4010	0.0296/0.0273	13

9.1 Measuring Space Coverage and Time Complexity

To the best of our knowledge, none of the existing works visualizes clustered graphs by balancing vertext areas within an arbitrary shape by fully utilizing the entire screen space. However, Bubble Sets [18] is a pioneering technique to visualize set information over point clouds. The advantage of this approach is that it shows the connectivity of components in the same clusters. GMap [31] and MapSets [25] are also relevant techniques, which use Voronoi tessellation to visualize clustered graphs with a map metaphor. We therefore compare our results together with these three conventional approaches, since they all utilize filled-in arbitrary regions to emphasize cluster information. The figures give the results from Bubble Sets, GMap, and MapSets that visualize the same datasets from Section 8. Since the region computation of Bubble Sets, GMap, and MapSets relies on the same initial layout algorithms, all three approaches cannot fully control the number of split clusters (see the purple clusters in Figure 7), and the screen size.

One property of Bubble Sets, GMap, and MapSets is that space usage is fragmented with empty white spaces and the fragmented empty space is not fully utilized. Our approach relaxes this constraint and finds a balanced layout that fully uses the screen as preferred by the biologists [52]. We thus introduce two space coverage measures M_N and M_V to evaluate the proper distribution of vertexts in the layout. We define M_N as the coefficient of variation ($CV = \sigma_n/\mu_n$) of distances of the vertexts to their k nearest neighbors, to examine if each vertext has equal distances to its neighbors. The value $\sigma_n = \sqrt{\frac{1}{|v|-1}\sum_{i=1}^{|v|}(X_i - \mu_n)^2}$ corresponds to the standard deviation of average distances X_i of a vertext to its k nearest neighbors and μ is the corresponding mean value of X_i. The other measure, $M_V = \sigma_v/\mu_v$, is defined similarly to M_N, while each vertext value X_i corresponds to the number of pixels of its corresponding Voronoi cell. Table 3 gives the summarized CVs. Both measures show that the area assigned to each vertext is more balanced in our approach, and this tendency increases as the data size increases.

We show the running times for Bubble Sets, GMap, MapSets, and our approach in Table 4, and Figure 7 shows the corresponding layouts. In general, Bubble Sets and GMap are faster than our approach since they only compute the Voronoi tessellation once to generate the contour. The bottleneck of our algorithm is that we need to iteratively compute Voronoi tessellations in order to guide a vertext to its proper position. Nonetheless, our approach runs comparably to MapSets, since it also has an iterative process to glue separate clusters into an aggregated one. We observe that the MapSets strategy sometimes fails to link all identical clusters, because the positions of points are fixed initially (see Figures 7(a-3)-(c-3)).

9.2 Interview with Experts in Biology

To validate the usability of our approach, we shared our results with six domain experts who are experienced with

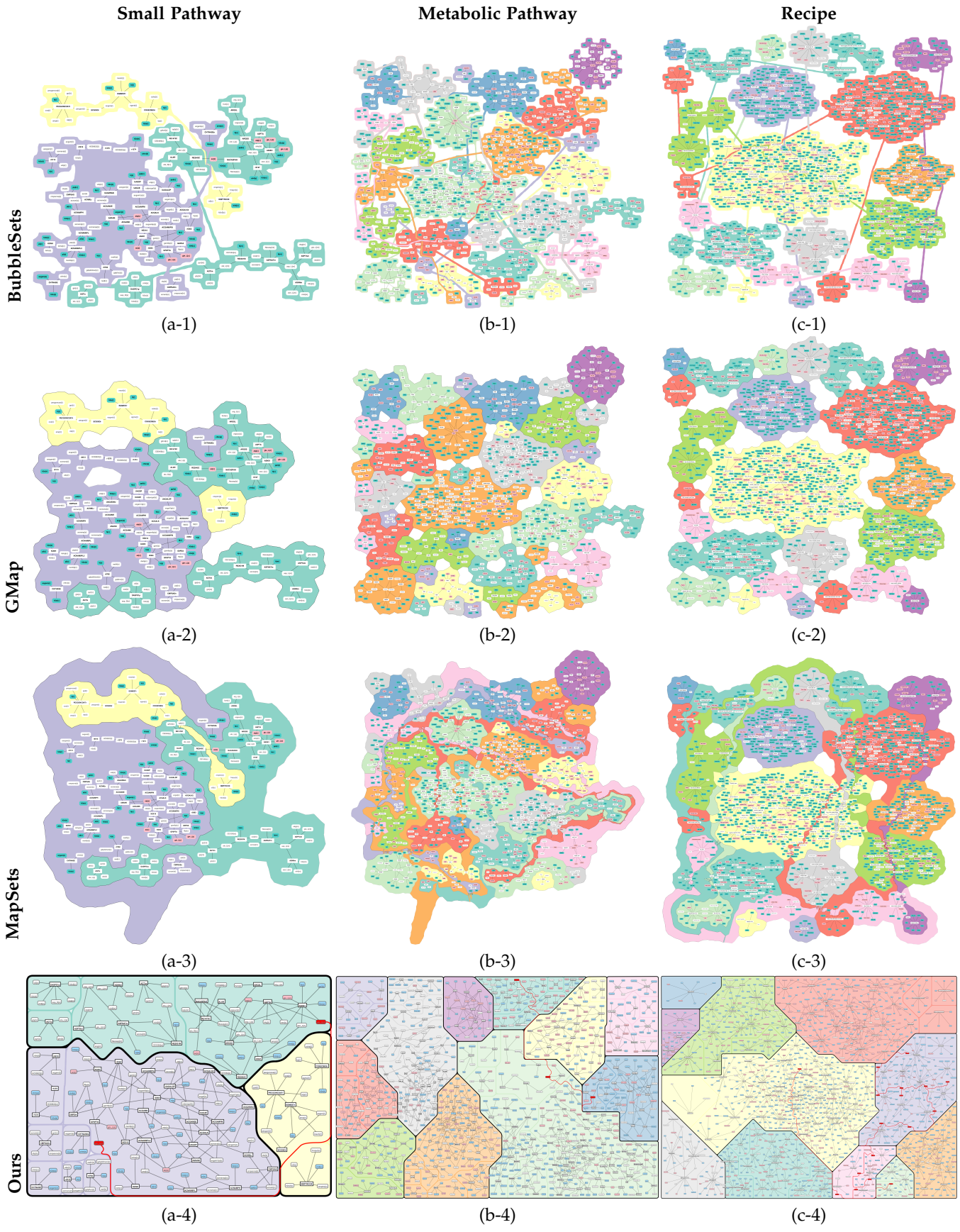


Fig. 7. Results generated using the Bubble Sets [18], GMap [31], MapSets [25] and our area balancing algorithms. The initial layout has been computed using the graphviz library (version 2.40.1), which generate better results in comparison to previous versions. Readers can refer to supplementary materials (Section 11) for larger figures. Note that each color here indicates a single cluster, even if it is non-contiguous in some cases.

TABLE 3
Comparison of space coverage with Bubble Sets [18], GMap [31], MapSets [25], and ours. We chose $k = 5$ for the k nearest neighbors.

	(Neighbor M_N)/(Voronoi M_V)			
	Bubble Sets	GMap	MapSets	Ours
Fig. 5	0.195/0.966	0.195/0.966	0.315/4.779	0.181/0.202
Fig. 7(b-4)	0.233/1.210	0.233/1.210	0.282/5.549	0.201/0.382
Fig. 7(c-4)	0.293/0.965	0.293/0.965	0.328/7.836	0.262/0.595
Fig. 6	0.280/0.743	0.280/0.743	0.421/8.276	0.392/0.791

TABLE 4
Comparison of running times (in seconds) with Bubble Sets [18], GMap [31], MapSets [25], and ours.

	Bubble Sets	GMap	MapSets	Ours
Fig. 5	7.03	3.88	7.87	22
Fig. 7(b-4)	27.6	4.59	242.38	245
Fig. 7(c-4)	42.32	7.62	247.08	237
Fig. 6	610.72	8.76	1090.27	1136

manually creating pathway diagrams, and discussed our selected aesthetic criteria and the quality of the results with them. The interviews involved one professor (P1) from Scripps Research in the USA, one professor (P2) from the University of Vienna specializing in biology, two postdoctoral researchers (P3 and P4), and two Ph.D. candidates (P5 and P6) from the Research Center for Molecular Medicine of the Austrian Academy of Sciences (CeMM), who specialize in bioinformatics. The professors had more than seven years experience manually working with the pathway layout, and the participants had, on average, about three years of experience. The process began by first explaining how to read the visualization (see Section 11 for a complete reference), including the content of the datasets and the corresponding color coding. We gave all participants enough time (10-15 minutes) to investigate, question, and understand the results, until they did not have any further questions about the visual representation.

The two senior professors (P1 and P2) expressed that (S1) has evenly distributed vertexts with fewer overlaps, making the map easier to read because the vertexts retain mutual distance. P6 expressed that singular clusters help to quickly spot biological functions, and complex shapes do not provide additional contextual information (S2). Two postdoctoral researchers (P3 and P4) considered the visual integration in (S3) to be more important than (S1) and (S2), since it helps readers to focus on the network connectivity.

P3 suggested we could eliminate unimportant duplicated vertexts (e.g., H_2O) completely from the diagram, and P4 mentioned that balanced distribution is nice only if it helps to remove vertext overlaps. This constraint also untangles high-degree vertices, which eliminates the nature of highly connected vertices that are often gathered in a dense region. P5 stated that the diagram with less vertext occlusion (S1) gives the best readability. All participants agreed that duplicating the vertexts improves the visual quality; however, completely duplicating the vertexts helps to untangle visual clutter, even though it requires more effort to find all the connected neighbors.

P3 and P5 considered our visual integration helpful for tracking duplicated vertices; nonetheless, we could add many visual integrations, which again would complicate the readability of the diagrams. Four participants appreciated the adjustment of the input aspect ratio when using our approach, while the other two thought that it was good to have but not critical. All participants expressed their interest in using our algorithm. In the end, we also discussed what they considered are the other important factors for pathway diagrams. P4 preferred to have the possibility to adjust parameters to retrieve her preferred results and P5 preferred that the usage should be as simple as possible. P1 strongly recommended to integrate interaction techniques into the computed diagrams, especially when the diagram is large. P6 suggested that we could visualize pathways beyond referring to topological patterns.

P1 and P3-P6 agreed that Bubble Sets, GMap, and MapSets preserve the network structure well, while our approach provides contiguous clusters and a more balanced area to follow. As a trade-off, non-separate and contiguous clusters were more appreciated, and a more balanced area improved label readability. We did not receive specific feedback from P2 regarding this comparison.

9.3 Limitations and Potential Extensions

Since the present approach is a force-based balancing technique, it also inherits the same limitations from the conventional force-based approach. One significant drawback is that the initial position has a strong influence on the final layout. In our current implementation, we use the sfdp package, a multi-scale version of Kamada and Kawai's approach in GraphViz [26], to compute the initial layout of the decomposed subgraphs. Another common limitation is that our technique has the potential to fall into a local minimum, where a vertext is constrained by the positions of other vertexts. This could potentially generate unpleasant octilinear boundaries. Perturbation on vertexts has been introduced to experimentally avoid a vertext being trapped by other surrounding forces. As shown in Figure 6, large graphs require a large canvas for proper investigation. A possible approach for a fixed canvas size would be to consider a hierarchical representation, but this in turn would require careful design and evaluation of its interpretability.

Moreover, as mentioned previously in Section 3, the structure of the graph has been transformed from a general graph to a clustered graph through vertext duplication, so that clusters in the diagram are disjoint. This property also allows us to apply the approach to visualize multilayer graphs, since we can compress each layer of the graph and embed each layer into a polygonal region and can still utilize the same spanning-tree visual integration to highlight vertext instances in different layers. We do not claim that our technique can fully solve multilayer graph visualization. However, our method provides an alternative solution because our original idea mimics the design principles from domain experts in biology [52], where they manually adjust pathway diagrams as a multilayer graph.

10 Conclusion and Future Work

This paper presents a pioneering approach that takes vertext area into consideration in order to embed a clustered graph inside arbitrary polygonal areas in a space-balanced fashion. We achieve this by incorporating a multi-level scheme to balance vertext area using a top-down model, where we utilize structure motifs to guide the partitioning in order

to approximate the regions effectively. Hairball effects are controlled via vertext duplication, which is coupled with a visual integration using a Steiner tree algorithm.

As a future research direction, we plan to extend the same concept by introducing features other than the points, and integrate features to multiple hierarchy layers. Such features may include line features or area features to enable more complex visual representations together with a graph, because some experimental analysis requires spatial information together with relationship information. For example, biologists often investigate which pathways occur inside which portion of a cell. Establishing a motif library is also a future goal to develop a standard language to convey and link domain knowledge to representative topological structures. This can be done through topological structure analysis to partition structures into certain limited disjoint sets. A more sophisticated vertext duplication technique [51], such as minimizing the duplication number or the best scheme to split a vertext, will also be investigated further. Last but not least, we will make the source code for our system readily available to the community [8].

ACKNOWLEDGMENTS

The project has received funding from the European Union Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No. 747985, from the Vienna Science and Technology Fund (WWTF) through projects VRG11-010, from the Austrian Science Fund (FWF) through project P31119, and from King Abdullah University of Science and Technology (KAUST) through award BAS/1/1680-01-01. The authors would like to thank Michael Cusack from Research Communication and Publication Services at KAUST for proofreading.

REFERENCES

[1] Boost C++ Libraries. http://www.boost.org/. Accessed: 2018-02-15.
[2] CGAL: The Computational Geometry Algorithms Library. https://www.cgal.org. Accessed: 2018-08-09.
[3] Eigen: A C++ template library for linear algebra. https://eigen.tuxfamily.org/. Accessed: 2018-08-09.
[4] Graph drawing contest 2019. http://www.graphdrawing.de/contest2019/topics.html. Accessed: 2019-03-11.
[5] MCL: a cluster algorithm for graphs. https://micans.org/mcl/index.html. Accessed: 2018-12-09.
[6] Qt 5.8: Cross-platform software development for embedded & desktop. https://www.qt.io/. Accessed: 2018-02-15.
[7] Themealdb database. https://www.themealdb.com/. Accessed: 2019-03-11.
[8] Keiro: A package for visualizing graphs, 2020.
[9] B. Alper, N. Riche, G. Ramos, and M. Czerwinski. Design study of linesets, a novel set visualization technique. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2259–2267, 2011.
[10] S. Auer, V. Kovtun, M. Prinz, A. Kasprzik, M. Stocker, and M. E. Vidal. Towards a knowledge graph for science. In *Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics*, 2018.
[11] M. Balzer and O. Deussen. Voronoi treemaps. In *Proc. IEEE Symposium on Information Visualization (InfoVis'05)*, pp. 49–56, 2005.
[12] F. Bertault and M. Miller. An algorithm for drawing compound graphs. In J. Kratochvíyl, ed., *Graph Drawing*, pp. 197–204, 1999.
[13] S. P. Borgatti, A. Mehra, D. J. Brass, and G. Labianca. Network analysis in the social sciences. *Science*, 323(5916):892–895, 2009.
[14] P. Brivio, M. Tarini, and P. Cignoni. Browsing large image datasets through Voronoi diagrams. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1261–1270, 2010.
[15] K. Buchin, W. Meulemans, and B. Speckmann. A new method for subdivision simplification with applications to urban-area generalization. In *GIS*, pp. 261–270, 2011.
[16] G. M. Chaikin. An algorithm for high-speed curve generation. *Computer Graphics and Image Processing*, 3(4):346 – 349, 1974.
[17] S. Chaturvedi, C. Dunne, Z. Ashktorab, R. Zachariah, and B. Shneiderman. Group-in-a-box meta-layouts for topological clusters and attribute-based groups: Space-efficient visualizations of network communities and their ties. *Computer Graphics Forum*, 33(8):52–68, 2014.
[18] C. Collins, G. Penn, and S. Carpendale. Bubble sets: Revealing set relations with isocontours over existing visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1009–1016, 2009.
[19] W. Cui, Y. Wu, S. Liu, F. Wei, M. X. Zhou, and H. Qu. Context preserving dynamic word cloud visualization. In *IEEE Pacific Visualization Symposium*, pp. 121–128, 2010.
[20] D. Delling, A. Gemsa, M. Nöllenburg, T. Pajor, and I. Rutter. On d-regular schematization of embedded paths. *Comput. Geom. Theory Appl.*, 47(3A):381–406, 2014.
[21] S. Dongen. A cluster algorithm for graphs. Technical report, CWI (Centre for Mathematics and Computer Science), Amsterdam, The Netherlands, The Netherlands, 2000.
[22] C. Dunne and B. Shneiderman. Motif simplification: Improving network visualization readability with fan, connector, and clique glyphs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pp. 3247–3256, 2013.
[23] P. Eades and Q.-W. Feng. Multilevel visualization of clustered graphs. In *Graph Drawing*, pp. 101–112, 1997.
[24] P. Eades, Q. Nguyen, and S.-H. Hong. Drawing big graphs using spectral sparsification. In *Graph Drawing and Network Visualization*, pp. 272–286, 2018.
[25] A. Efrat, Y. Hu, S. G. Kobourov, and S. Pupyrev. MapSets: Visualizing embedded and clustered graphs. *J. Graph Algorithms Appl.*, 19(2):571–593, 2015.
[26] J. Ellson, E. Gansner, L. Koutsofios, S. C. North, and G. Woodhull. Graphviz— open source graph drawing tools. In *Graph Drawing*, pp. 483–484, 2002.
[27] F. Frati. *Clustered Graph Drawing*, pp. 326–331. Springer, 2016.
[28] D. Fried and S. G. Kobourov. Maps of computer science. In *2014 IEEE Pacific Visualization Symposium*, pp. 113–120, 2014.
[29] T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software—Practice & Experience*, 21(11):1129–1164, 1991.
[30] E. R. Gansner and Y. Hu. Efficient node overlap removal using a proximity stress model. In *Graph Drawing*, pp. 206–217, 2009.
[31] E. R. Gansner, Y. Hu, and S. Kobourov. Gmap: Visualizing graphs and clusters as maps. In *2010 IEEE Pacific Visualization Symposium (PacificVis)*, pp. 201–208, 2010.
[32] E. R. Gansner, Y. Hu, and S. Kobourov. Visualizing graphs and clusters as maps. *IEEE Computer Graphics and Applications*, 30(6):54–66, 2010.
[33] M. Gronemann and M. Jünger. Drawing clustered graphs as topographic maps. In *Graph Drawing (GD'12)*, vol. 7704 of *Lecture Notes in Computer Science*, pp. 426–438. Springer, 2013.
[34] S. Hachul and M. Jünger. Drawing large graphs with a potential-field-based multilevel algorithm. In J. Pach, ed., *Graph Drawing*, pp. 285–295, 2005.
[35] N. Henry, A. Bezerianos, and J. Fekete. Improving the readability of clustered social networks using node duplication. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1317–1324, 2008.
[36] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6):409–436, 1952.
[37] K. E. Hoff, III, J. Keyser, M. Lin, D. Manocha, and T. Culver. Fast computation of generalized Voronoi diagrams using graphics hardware. In *Proceedings of SIGGRAPH '99*, pp. 277–286, 1999.
[38] M. L. Huang and Q. V. Nguyen. A space efficient clustered visualization of large graphs. In *Fourth International Conference on Image and Graphics (ICIG 2007)*, pp. 920–927, 2007.
[39] M. Kivelä, A. Arenas, M. Barthelemy, J. P. Gleeson, Y. Moreno, and M. A. Porter. Multilayer networks. *Journal of Complex Networks*, 2(3):203–271, 2014.
[40] R. Klein. *Concrete and Abstract Voronoi Diagrams*, vol. 400 of *Lecture Notes in Computer Science*. 1989.
[41] S. Kobourov, S. Pupyrev, and P. Simonetto. Visualizing graphs as maps with contiguous regions. In *EuroVis - Short Papers*, 2014.

[42] L. Kou, G. Markowsky, and L. Berman. A fast algorithm for steiner trees. *Acta Informatica*, 15(2):141–145, 1981.
[43] K. Laboratories. KEGG PATHWAY Database: Metabolic pathways. https://www.kegg.jp/kegg-bin/show_pathway?map01100, 2018. Accessed: 2018-07-20.
[44] G. Li, S. Nickel, M. Nöllenburg, I. Viola, and H.-Y. Wu. World map of recipes, 2019. 1st Place Award, Creative Topic-"Meal Ingredients", of the 28th Annual Graph Drawing Contest.
[45] C.-C. Lin and H.-C. Yen. A new force-directed graph drawing method based on edge–edge repulsion. *Journal of Visual Languages & Computing*, 23(1):29 – 42, 2012.
[46] F. Mcgee, M. Ghoniem, G. Melançon, B. Otjacques, and B. Pinaud. The state of the art in multilayer network visualization. *Computer Graphics Forum*, 38(6):125–149, 2019.
[47] W. Meulemans, N. H. Riche, B. Speckmann, B. Alper, and T. Dwyer. Kelpfusion: A hybrid set visualization technique. *IEEE Transactions on Visualization and Computer Graphics*, 19(11):1846–1858, 2013.
[48] W. Meulemans, A. van Renssen, and B. Speckmann. Area-preserving subdivision schematization. In *Proceedings of the 6th International Conference on Geographic Information Science*, pp. 160–174, 2010.
[49] L. Nachmanson, R. Prutkin, B. Lee, N. H. Riche, A. E. Holroyd, and X. Chen. Graphmaps: Browsing large graphs as interactive maps. In *Graph Drawing and Network Visualization*, Springer Lecture Notes in Computer Science, pp. 3–15, 2015.
[50] M. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 103 23:8577–82, 2006.
[51] S. S. Nielsen, M. Ostaszewski, F. McGee, D. Hoksza, and S. Zorzan. Machine learning to support the presentation of complex pathway graphs. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, pp. 1–1, 2019.
[52] A. Noronha, A. D. Danóelsdóttir, P. Gawron, F. Jóhannsson, S. Jónsdóttir, S. Jarlsson, J. P. Gunnarsson, S. Brynjólfsson, R. Schneider, I. Thiele, and R. M. T. Fleming. ReconMap: an interactive visualization of human metabolism. *Bioinformatics*, 33(4):605–607, 2017.
[53] S. Nusrat and S. Kobourov. The state of the art in cartograms. *Computer Graphics Forum*, 35(3):619–642, 2016.
[54] M. J. Roberts, H. Gray, and J. Lesnik. Preference versus performance: Investigating the dissociation between objective measures and subjective ratings of usability for schematic metro maps and intuitive theories of design. *International Journal of Human-Computer Studies*, 98:109 – 128, 2017.
[55] A. Roy, A. Noronha, A. Puente, A. Žagare, A. Heinken, A. D. Daníelsdóttir, B. Garcia, D. Merten, D. A. Ravcheev, E. Guerard, E. John, G. Preciat, H. S. Haraldsdóttir, J. Modamio, L. Heirendt, L. Wiltgen, L. Friscioni, M. Prendergast, M. Krueger, M. Rodrigues, M. Krecke, M. Rouquaya, P. Gawron, R. Schneider, S. Magnúsdóttir, S. Sahoo, Y. Jarosz, I. Thiele, R. M. Fleming, A. Zinovyev, I. Kuperstein, and N. Sompairac. The Virtual Metabolic Human database: integrating human and gut microbiome metabolism with nutrition and disease. *Nucleic Acids Research*, 47(D1):D614–D624, 2018.
[56] B. Shneiderman. Tree Visualization with Tree-maps: 2-d space-filling approach. *ACM Transactions on Graphics*, 11(1):92–99, 1992.
[57] B. Shneiderman and M. Wattenberg. Ordered treemap layouts. In *IEEE Symposium on Information Visualization*, pp. 73–78, 2001.
[58] O. Sorkine and D. Cohen-Or. Least-squares meshes. In *Proceedings of the Shape Modeling International*, pp. 191–199, 2004.
[59] C. Vehlow, F. Beck, and D. Weiskopf. Visualizing group structures in graphs: A survey. *Computer Graphics Forum*, 36(6):201–225, 2017.
[60] L. Verlet. Computer "experiments" on classical fluids. i. thermodynamical properties of lennard-jones molecules. *Physical Review Journals Archive*, 159:98–103, 1967.
[61] Y. Wang, Y. Wang, Y. Sun, L. Zhu, K. Lu, C. Fu, M. Sedlmair, O. Deussen, and B. Chen. Revisiting stress majorization as a unified framework for interactive constrained graph visualization. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):489–499, 2018.
[62] Y.-S. Wang and M.-T. Chi. Focus+context metro maps. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2528–2535, 2011.
[63] H.-Y. Wu, B. Niedermann, S. Takahashi, M. J. Roberts, and M. Nöllenburg. A survey on transit map layout – from design, machine, and human perspectives. *Computer Graphics Forum*, 39(3):619–646, 2020.
[64] H.-Y. Wu, M. Nöllenburg, F. L. Sousa, and I. Viola. Metabopolis: Scalable network layout for biological pathway diagrams in urban map style. *BMC Bioinformatics*, 20(1), 2019.
[65] H.-Y. Wu, M. Nöllenburg, and I. Viola. Graph models for biological pathway visualization: State of the art and future challenges. In *1st workshop on the Visualization of Multilayer Networks*, 2019. https://www.cg.tuwien.ac.at/research/publications/2019/wu-2019-visworkshop/wu-2019-visworkshop-paper.pdf.
[66] H.-Y. Wu, S. Takahashi, and R. Ishida. Overlap-free labeling of clustered networks based on voronoi tessellation. *Journal of Visual Languages & Computing*, 44:106 – 119, 2018.
[67] H.-Y. Wu, S. Takahashi, C.-C. Lin, and H.-C. Yen. Voronoi-based label placement for metro maps. In *Proceedings of the 17th International Conference on Information Visualisation (IV2013)*, pp. 96–101, 2013.
[68] K.-H. Yeap and M. Sarrafzadeh. Floor-planning by graph dualization: 2-concave rectilinear modules. *SIAM J. Comput.*, 22(3):500–526, 1993.
[69] V. Yoghourdjian, D. Archambault, S. Diehl, T. Dwyer, K. Klein, H. C. Purchase, and H.-Y. Wu. Exploring the limits of complexity: A survey of empirical studies on graph visualisation. *Visual Informatics*, 2(4):264–282, 2018.
[70] V. Yoghourdjian, T. Dwyer, G. Gange, S. Kieffer, K. Klein, and K. Marriott. High-quality ultra-compact grid layout of grouped networks. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):339–348, 2016.
[71] Y. Zhonghua and W. Lingda. Accelerated layout for large-scale network based on quadtree. In *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, pp. 422–425, 2017.

Hsiang-Yun Wu is Postdoctoral Research Fellow at the Institute of Visual Computing & Human-Centered Technology, TU Wien, Austria. Her research interests include the algorithm development of customized graph representations and she has been working on map labeling, railway map design, and complex network visualization. She received her PhD from The University of Tokyo, Japan in 2013.

Martin Nöllenburg is an associate professor for graph and geometric algorithms in the Algorithms and Complexity Group, TU Wien, Vienna, Austria. He received his PhD and habilitation degrees in computer science from Karlsruhe Institute of Technology (KIT), Germany, in 2009 and 2015, respectively. His research interests include graph drawing algorithms, computational geometry, and information visualization with a focus on applications in network and geovisualization.

Ivan Viola is an associate professor in Visual Computing at King Abdullah University of Science and Technology (KAUST), Saudi Arabia. His research interest is scalable technology for interactive visualization with the ultimate goal of constructing, visualizing, and modeling the entire complex biological organism at an atomistic or molecular detail. This technology will allow people to interact, explore, study, and understand the life at nanoscale.