EulerMerge: Simplifying Euler Diagrams Through Set Merges

Xinyuan Yan[1][0000−0003−3396−1310], Peter Rodgers[2][0000−0002−4100−3596], Peter Rottmann[3][0000−0003−3926−4938], Daniel Archambault[4][0000−0003−4978−8479], Jan-Henrik Haunert[3][0000−0001−8005−943X], and Bei Wang[1][0000−0002−9240−0700]

[1] University of Utah, USA. {xinyuan.yan,beiwang}@sci.utah.edu
[2] University of Kent, UK. P.J.Rodgers@kent.ac.uk
[3] University of Bonn, Germany. {rottmann,haunert}@igg.uni-bonn.de
[4] Newcastle University, UK. daniel.archambault@newcastle.ac.uk

Abstract. Euler diagrams are an intuitive and popular method to visualize set-based data. In an Euler diagram, each set is represented as a closed curve, and set intersections are shown by curve overlaps. However, Euler diagrams are not visually scalable and automatic layout techniques struggle to display real-world data sets in a comprehensible way. Prior state-of-the-art approaches can embed Euler diagrams by splitting a closed curve into multiple curves so that a set is represented by multiple disconnected enclosed areas. In addition, these methods typically result in multiple curve segments being drawn concurrently. Both of these features significantly impede understanding. In this paper, we present a new and scalable method for embedding Euler diagrams using set merges. Our approach simplifies the underlying data to ensure that each set is represented by a single, connected enclosed area and that the diagram is drawn without curve concurrency, leading to wellformed and understandable Euler diagrams.

Keywords: Euler diagrams · Set visualization · Hypergraph visualization · Scalability.

1 Introduction

Set-based data are found in many real-world examples. In personalized recommendation systems, sets capture multivariate relationships among users, query topics, item features [11], and reasoning [4,20]. Set-based data are also prevalent in biological systems to encode multiway relationships among entities in protein complexes, transcription factor and microRNA regulation networks, protein function annotations, and metabolic processes [36].

An intuitive way to visualize set-based data is through an Euler diagram, which captures sets and their relationships. In an Euler diagram, sets are represented by closed curves that enclose regions. The way the regions overlap reveals the intersections between the sets. Representing sets using an Euler diagram is visually intuitive; however, these approaches can suffer from comprehensibility

issues even with a small number of sets, and scaling is considered to be limited to 10 sets [2]. Recall that a planar graph is a graph that can be drawn on a plane without edge crossings, finding a visualization of a set system as an Euler diagram is equivalent to finding a planar embedding of its dual graph [9]. However, for many set systems, no planar embedding exists. In these cases, previous algorithms to embed Euler diagrams represent a set by splitting it into two or more closed curves [25,32], resulting in the set not being represented by a single *connected enclosed area*. This has the advantage that all instances of set systems are embeddable, but has the disadvantage that the diagram is much harder to understand, as the same set can be represented in different parts of the diagram.

In addition, these prior layout methods typically introduce *concurrency*, where multiple curves share a line segment. Both concurrency and disconnected enclosed areas are violations of important *wellformedness conditions*, which are known to impede understanding [26].

We present a new method that simplifies an Euler diagram via set merges. A set merge takes the union of two or more sets in a set system and represents the resulting set as a single closed curve. This increases the scalability of data that the method can successfully visualize compared to previous methods. Our contributions are as follows:

- We produce Euler diagrams that satisfy a number of wellformedness conditions. In particular, our simplification process ensures that each set (or merged group of sets) is represented by a single connected enclosed area and that there is no concurrency.
- We demonstrate via experiments that, typically, a small number of set merges leads to Euler diagrams that are wellformed and understandable.

2 Related Work

The visualization of set-based data has been an active area of research. We use the terms "set system" and "hypergraph" interchangeably with the understanding that these terms arise from different research communities. Hypergraphs generalize graphs by allowing *hyperedges*, that is, edges that contain more than two nodes. Hence, a hyperedge is a set and a hypergraph is a set system.

Many set visualization techniques use geometric elements such as lines, circles, ovals and closed curves to connect or enclose set elements. Other concepts use tables or matrices (see [2, 8] for surveys). However, in this section, we concentrate on the closely related background in Euler diagrams and simplification.

Euler and Venn Diagrams. Various methods for embedding a set system using closed curves have been developed. These methods vary by the type of shape applied, for instance, constraining the Euler diagram curves to circles [14,33–35] or hexagonal/square grid cells [28]. In these cases, the diagram is represented using only these shapes. However, only a subset of set systems are embeddable with such shape restrictions, a limitation not present in our work.

Other work takes elements that have been previously embedded and superimpose polygons on top of them [6]. Similarly, GMap [10] creates one or more

polygons around areas of a graph that are in the same set. GMap relies on a prior layout of data to be grouped, which our work does not require.

Mäkinen [21] introduced an edge-based and a subset-based approach to draw hypergraphs, where hyperedges are drawn as smooth curves *connecting* or *enclosing* their nodes, respectively. However, an Euler diagram is not embeddable for all set systems [25, 32] with a single closed curve representing each set, as dual graphs derived from these set systems cannot be drawn in a planar way. Other methods, such as SPEULER [13], instead arrange set elements using a circular layout. However, in order to have an embeddable diagram, SPEULER produces overlaps of two curves when there are no elements within them.

Techniques also exist to refine an Euler diagram drawing once it has been generated, improving its readability. For instance, eulerForce [22] uses a force-directed algorithm to refine the set, whereas EulerSmooth [31] uses curve shortening flows to achieve the same objective.

Graph and hypergraph simplification. A number of approaches have been proposed that simplify and summarize graphs (e.g., [29]). Visualization of simplified graphs has been explored [7,16,18], which may be applicable to the node-link diagrams of set systems. Our approach is related to graph coarsening driven by visual criteria [1, 3], which aggregates subgraphs into single nodes based on topological properties. Instead of set merges, Euler diagrams may also be simplified via element removal and optimization [27]. Zhou et al. [37] simplified hypergraphs by combining nodes if they belong to almost the same set of hyperedges, and merging hyperedges if they share almost the same set of nodes. However, it is different from our work in the simplification criteria, algorithm, and visualization perspectives. Oliver et al. [24] recently proposed a framework for visualizing scalable hypergraphs, with a convex polygon-based layout. Their approach incorporates an iterative, reversible simplification process and layout optimization. They have several merging operations, including hyperedge merging. As with [37], the simplification criteria, underlying algorithms and resultant visualization differ markedly. Whereas their method attempts to reduce the unwanted overlap of the convex polygons representing hyperedges, our approach guarantees no unwanted overlaps for the curves representing sets.

3 Technical Background

Euler diagrams and zones. A closed curve γ in the plane $\mathbb{R}^2$ is a continuous function $\gamma\colon [0,1] \to \mathbb{R}^2$, where $\gamma(0) = \gamma(1)$. An *Euler diagram* is a pair $\mathcal{E} = (\Gamma, \pi)$, where Γ is a finite set of closed curves in $\mathbb{R}^2$, L is a set of labels, and $\pi\colon \Gamma \to L$ is a mapping that assigns to each curve $\gamma \in \Gamma$ a label in L. A *minimal region* of an Euler diagram is a connected component of $\mathbb{R}^2 - \bigcup_{\gamma\in\Gamma} \text{image}(\gamma)$. A *zone* of an Euler diagram is a set of minimal regions that represent the intersection of sets. The set of zones is called the *abstract description* of the diagram.

Wellformedness conditions. An Euler diagram $\mathcal{E}$ may have a number of desirable properties, referred to as *wellformedness conditions* [34]. These include:

- **Simplicity**: if all curves in Γ are simple curves (i.e., no self-crossings);

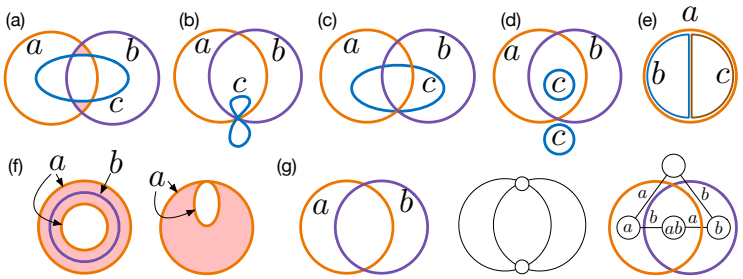

Fig. 1. Key wellformedness conditions: (a) a disconnected zone in pink; (b) a non-simple curve c; (c) a triple point; (d) duplicated curve labels c with an unconnected closed area; (e) concurrency: a and b, a and c, b and c; (f) two examples of duplicated curve label (left and right): label a represents a set with a connected enclosing area that contains a hole. (g) An Euler diagram (left), its Euler graph (middle), and its dual graph (right).

- **No triple points**: if there are no triple points of intersection among the curves in Γ;
- **Transversality**: if two curves in Γ intersect, they intersect transversally;
- **Connected zones**: if each zone of $\mathcal{E}$ is connected.

Our construction method guarantees that the diagram has simple curves and connected zones. There is evidence to show that triple points and transversality have limited impact on user understanding [26]. See Figure 1 for an illustration. Of particular importance to this paper are the two wellformedness conditions:

- **No concurrency**: if no pairs of curves in Γ run concurrently.
- **Unique curve labels**: if π is an injective function.

Unique to this paper, we further refine the condition of unique curve labels into two conditions:

- **Genus free**: if the area enclosed by curves of the same label in Γ does not contain any genus (that is, a hole).
- **Connected enclosed area**: if the area enclosed by curves of the same label in Γ is connected.

For example, in Figure 1(f), the connected enclosed area condition is not violated because the curve "a" is duplicated, but the region represented by "a" is not split into two components. However, the diagram is not genus free as "a" has a hole. However, in Figure 1(d) curve "c" is duplicated and the region represented is split into two components, one inside "a" and "b", with the other region outside them. As a result, "c" does not have a connected enclosed area.

Euler graph and dual graph. An *Euler graph* $G_{\mathcal{E}}$ constructed from an Euler diagram $\mathcal{E}$ has vertices defined at all curve intersection points, and edges defined as the curve segments that connect the vertices. By construction, each face of $G_{\mathcal{E}}$ is a minimal region of $\mathcal{E}$. The *dual graph* of an Euler diagram $\mathcal{E}$ is the standard dual graph of the Euler graph $G_{\mathcal{E}}$. The vertices of the dual graph represent the zones in the Euler diagram and the edges of the dual graph connect

adjacent zones. Vertices are labeled by the curves that enclose their corresponding zones, and edges are labeled by the symmetric difference of their endpoints. See Figure 1(g).

We consider the main cause of poor interpretation with duplicate curve labels to be caused by violations of connected enclosed area rather than genus free. Having genus present in a diagram means that curves with the same label are inside other curves with the same label, whereas the presence of a disconnected enclosed area means that the curves with the same label can be anywhere in the diagram, and therefore users may not spot all such curves when interpreting the diagram. Our approach is designed to ensure that the resulting Euler diagrams have simple curves and connected zones. We then further simplify with set merges to ensure connected enclosed areas and avoid concurrency.

4 Algorithm

In this section, we describe our novel algorithm for simplifying Euler diagrams with set merging. The code is available under a GPL open source license from https://github.com/tdavislab/EulerMerge. We use JGraphT [23], which provides planarity testing.

4.1 Algorithm Preliminaries

An Euler diagram is a visual representation of a set system. A *set system* is a set of sets $\mathcal{S} = \{S_1, S_2, ..., S_m\}$, where each set $S_i \in \mathcal{S}$ $(1 \leq i \leq m)$ is a nonempty subset of a universe $U = \bigcup_{i=1}^{m} S_i$. With an abuse of notation, for an element $u \in U$, $\mathcal{S}(u)$ contains sets from $\mathcal{S}$ that contain u, i.e., $\mathcal{S}(u) = \{S_i \in \mathcal{S} \mid u \in S_i\}$. We assume a set system is always given with a label-assigning map l and $l(S_i)$ is the label associated with the set S_i.

Our algorithm merges sets in $\mathcal{S}$ through an iterative process. In each iteration, two sets are selected from $\mathcal{S}$ and replaced in $\mathcal{S}$ by their union. During this process, the algorithm maintains a set of zones Z, which at any time is uniquely defined as follows: Z contains the empty set $z_0 := \emptyset$ and multiple nonempty sets $z_1, z_2, \ldots, z_n$ that partition U, such that any two elements $u, v \in U$ are contained in the same zone if and only if $\mathcal{S}(u) = \mathcal{S}(v)$. In other words, elements of U are in the same zone if they are contained in the same subset of sets from $\mathcal{S}$.

We work with a graphical representation G of the set of zones. We design our algorithm such that, preferably after few iterations, G will be the dual graph of a wellformed Euler diagram of the simplified set system. Although, initially, G may lack the property of a dual graph of a wellformed Euler diagram, for simplicity, we refer to it as *dual graph* throughout the whole merging process.

Let $G = (Z, E, l_Z, l_E)$. With an abuse of notation, the vertices Z represent the zones and the edges E model pairwise relationships among the zones, where l_Z and l_E are (mappings of) zone labels and edge labels, respectively. The zone label $l_Z(z)$ of each $z \in Z$ is formed by a subset of $\mathcal{S}$ that constitutes the zone. The edge label $l_E(e)$ of each edge $e = \{z_i, z_j\} \in E$ is the symmetric difference of

the two zone labels, i.e., $l_E(e) := l_Z(z_i) \triangle l_Z(z_j)$. These labels are updated along with the set system as the algorithm progresses. As noted in Section 3, the set of zone labels is the *abstract description* of the set system.

4.2 A Running Example

We illustrate the set merging process with a running example. This is a set system $\mathcal{S}$ of a director from a movie database, see Section 6. Each set is a movie, and set elements are the actors that appear in the movie. Therefore in an Euler diagram, curves represent movies, and the curves overlap if the movies share at least one actor.

For the running example, the director is *Bonowicz, Brett Ryan*. He has seven movies, forming a set system of seven sets:

(a) Garriage; actors: Caps, Bonowicz, Fox, Kessler, Kostenbaudor, Kozlow.
(b) Last Days of Ki, The; actors: Herbst, Stilwell, Trad-DeStefano, Ashkin, Bonowicz, Chai, Chernyak, Dixon, Harpole, Lindo, Peters, Sawyer, Suppa.
(c) Interview for a Night Job; actors: Dastoli, James, Vergara, Edwin.
(d) Pressing the Public Opinion; actors: DeVries, Yeager, Bonowicz, Chernyak, Coolman, Lindo, Moore (Michael), Nelson.
(e) Baseball and Glory; actors: Caffrey, Dienstag, Seabright, Shults, Chernyak, Coolman, Dastoli, Denniberg, Garcia, Grant, Leery, Myers, Reiber, Sawyer, Shields, Tompkins, Weinstein.
(f) Signs and Voices; actors: Hecht, Moore (Julianne), Shepherd, Bonowicz, Dean.
(g) Banana Shell, The; actors: Baksh, Ashkin, Coolman, Fernandez, Grant, Gunn, Sawyer, Zawacki, Niki.

Its corresponding abstract description (set of zone labels) can be produced by finding the nonempty intersections in the set system. That is, if an actor $u \in U$ is in a collection of movies $\mathcal{S}(u)$ (and no other movies), $\mathcal{S}(u)$ is added to the abstract description, giving: $\{\emptyset, \{a\}, \{b\}, \{c\}, \{d\}, \{e\}, \{f\}, \{g\}, \{b, d\}, \{b, g\}, \{c, e\}, \{e, g\}, \{b, d, e\}, \{b, e, g\}, \{d, e, g\}, \{a, b, d, f\}\}$.

For example, the actor *Dastoli* is in movies "c" and "e": "Interview for a Night Job" and "Baseball and Glory" and no other sets, which gives rise to an element $\{c, e\}$ in the abstract description, and a zone with a label of $\{c, e\}$ (for simplicity, also referred to as "ce").

4.3 An Overview of EulerMerge Algorithm

Our set merging algorithm (Algorithm 1) takes an input set system and generates an initial dual graph. It then selectively merges pairs of sets to produce a planar dual graph and finally applies additional set merges to remove concurrency.

The input to our EulerMerge algorithm is a set system $\mathcal{S}$ equipped with (a mapping of) set labels l, and the output is a dual graph of an Euler diagram G with zone labels l_Z and edge labels l_E; for simplicity, these labels are sometimes omitted in the pseudocode.

Algorithm 1: EulerMerge

Input : Set system $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$
Output: dual graph $G = (Z, E)$
$G \leftarrow$ InitialDualGraph($\mathcal{S}$);
$G \leftarrow$ NonPlanarToPlanar(G);
$G \leftarrow$ ConcurrencyRemoval(G);
return G

The InitialDualGraph algorithm creates an initial dual graph G for an input set system $\mathcal{S}$. First, the algorithm derives the abstract description by computing $\mathcal{S}(u)$ for each $u \in U$. Second, it creates edges and edge labels of G. Two zones z_i and z_j in G are connected with an edge if their zone labels differ by one. Third, the algorithm computes an induced graph for each set $S_i \in \mathcal{S}$. If this is connected then there are no duplicate curve labels for that set in the corresponding Euler diagram. If the induced graph is not connected, the algorithm adds edges to connect it. However, this operation will introduce concurrency wherever the labels of the two connected zones differ by more than one element.

The initial dual graph G may not correspond to a wellformed Euler diagram. First, there may not be a planar dual graph for the initial set system. Second, the constructive process for initializing a dual graph is heuristic and may not produce a planar dual even if one exists. However, using a heuristic process is justifiable as deciding whether a given set system can be drawn as an Euler diagram is NP-complete [12]. Additionally, G may not correspond to a wellformed Euler diagram because the diagram may have concurrent edges.

For the running example, the initial dual graph can be seen in Figure 2(a). The InitialDualGraph procedure ensures that zones that have labels with single symmetric difference are connected by edges. For example, "b" and "bd" are connected by an edge. However, this leaves the subgraph induced from the set "a" disconnected. Hence "a" and "abdf" are linked with an edge. This dual graph is nonplanar, so set merges via NonPlanarToPlanar (Section 4.4) are applied.

4.4 Set Merging for Planarity

Once we have an initial dual graph, we then find a planar dual by merging sets. We prioritize the planarity objective because we cannot embed an Euler diagram without a planar dual. Furthermore, we can move toward planarity and reduce concurrency simultaneously.

Every nonplanar graph contains a Kuratowski subdivision [17] (i.e., a subdivision of K_5 or $K_{3,3}$) as a subgraph, denoted G^K. Moreover, such a subgraph can be found in linear time [5]. As shown in Algorithm 2, we merge two sets present in such a subgraph until G becomes planar.

Our process for merging a pair of sets is shown in Algorithm 2. We first replace any zone label and edge label containing $l(S_2)$ with $l(S_1)$. We then merge vertices in the dual graph with identical labels.

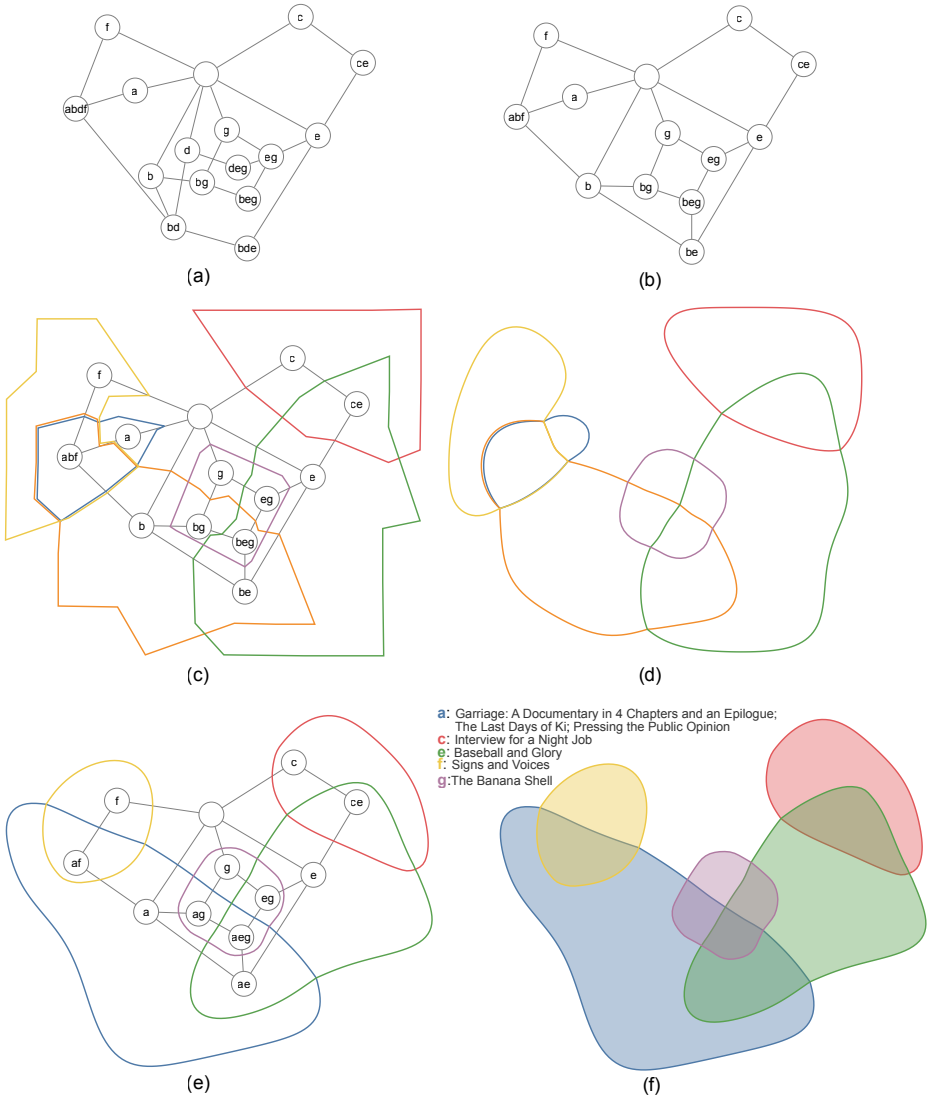

Fig. 2. Steps in the merging process: (a) The initial dual graph. (b) The first planar dual graph after merging sets "b" and "d". (c) The first planar dual graph and Euler diagram without smoothing. (d) The first Euler diagram with smoothing. (e) The dual graph and concurrency free final Euler diagram after merging sets "a" and "b". (f) The final Euler diagram with set labels.

Although achieving planarity is our top priority, we decide on the order of pairwise set merges based on the reduction of concurrency because the two sets are in a Kuratowski subdivision and merging them very likely also removes the subdivision. As a result we can remove concurrency whilst finding a planar dual. In a limited number of cases (for instance, when the sets to be merged are both

Algorithm 2: PairwiseSetMerge

Input : dual graph $G = (Z, E)$, Sets S_1 and S_2
Output: dual graph $G' = (Z', E')$
$G' \leftarrow G$;
for $z \in Z'$ **do**
  if $l(S_2) \in l'_Z(z)$ **then**
    $l'_Z(z) \leftarrow l'_Z(z) \cup l(S_1) \setminus l(S_2)$
for $e \in E'$ **do**
  if $l(S_2) \in l'_E(e)$ **then**
    $l'_E(e) \leftarrow l'_E(e) \cup l(S_1) \setminus l(S_2)$
for $z, z' \in Z'$ **do**
  if $l'_Z(z) = l'_Z(z')$ **then**
    $G' \leftarrow \text{ZoneMerge}(G', z, z')$
return G';

in exactly the same zones), the subdivision remains, whereas concurrency is greatly reduced by the set merges and so the second aim of the merging process is satisfied. In this case, planarity will be achieved through subsequent merges.

We introduce a measure that quantifies the concurrency in a dual graph:

$$\text{Concurrency}(G) = \sum_{e \in E} |l_E(e)| - |E|.$$

Recall that concurrent curve segments appear in the Euler diagram because there are multiple sets on an edge label. Concurrency(G) measures the overall size of edge labels. Concurrency$(G) = 0$ means that G has no concurrency as all edges are labeled with a single set. In Algorithm 3, we merge two sets in G^K that cause the most reduction in concurrency.

Algorithm 3: NonplanarToPlanar

Input : dual graph $G = (Z, E)$
Output: dual graph $G' = (Z', E')$
$G' \leftarrow G$
while !IsPlanar(G') **do**
  $G^K = (Z^K, E^K) \leftarrow \text{KuratowskiSubdivision}(G')$
  $R \leftarrow \bigcup_{z \in Z} l^K_Z(z)$;
  $G^M \leftarrow G'$;
  for $R_i \in R$ **do**
    for $R_j \in R$ **do**
      $G^S \leftarrow \text{PairwiseSetMerge}(G', R_i, R_j)$;
      if Concurrency$(G^S) <$ Concurrency(G^M) **then**
        $G^M \leftarrow G^S$;
  $G' \leftarrow G^M$;
return G'

In our running example, the initial dual graph is nonplanar and so we must apply Algorithm 3. In this case, we only need a single iteration as merging sets "b" and "d" results in a planar dual, shown in Figure 2(b). We retain the set label with the lowest lexicographical order during set merges, in this case, "b".

This set merge also leads to reduced concurrency. The dual graph in Figure 2(a) has a Concurrency of 6, whereas the dual graph in Figure 2(b) has a Concurrency of 2. Once we have a planar dual graph, it can be used to embed an Euler diagram as shown in Figure 2(c) and with improved layout in Figure 2(d).

4.5 Set Merging to Remove Concurrency

We can apply additional set merges to our planar dual to remove the remaining concurrency, see Algorithm 4. We apply a greedy approach, that is, by merging pairs of sets that reduce the most amount of concurrency at each step. We note that alternative strategies for ordering pairwise set merges are also possible, as discussed in Section 7. The final if statement, which ensures planarity by calling Algorithm 3, is for rare cases where nonplanar duals have been produced by set merging. We have not encountered any example that contains such a rare case.

Algorithm 4: ConcurrencyRemoval

```
Input  : dual graph G = (V, E)
Output: dual graph G' = (V', E')
G' ← G
while Concurrency(G') > 0 do
    R ← ⋃_{z∈Z'} l_{Z'}(z);
    G^M ← G';
    for R_i ∈ R do
        for R_j ∈ R do
            G^S ← PairwiseSetMerge(G', R_i, R_j);
            if Concurrency(G^S) < Concurrency(G^M) then
                G^M ← G^S;
    G' ← G^M;
if !IsPlanar(G') then
    G' ← NonPlanarToPlanar(G')
return G'
```

In the running example, the dual graph in Figure 2(c) still contains concurrency, e.g., between the connected zones "a" and "abf", as seen in the Euler diagram curve where segments "b" (orange) and "f" (yellow) run concurrently. To remove all concurrency in this case, we need only a single iteration by merging sets "a" and "b", shown in Figure 2(e). The merge renames "abf" as "af" removing the concurrency as they have a single symmetric difference with "a". With no concurrency, the merging process is complete. The final Euler diagram is given in Figure 2(f). Here, one curve represents the merging of three movies. We have

now simplified the abstract representation and the Euler diagram, embedding an Euler diagram without concurrency, at the cost of losing some detail.

Once we have formed an embeddable dual, we apply existing algorithms for embedding the dual graph [25], followed by smoothing using EulerSmooth [31] to refine the diagram boundaries.

Finally, we compare against the prior state-of-the-art general Euler diagram embedding [25], which we refer to as EulerGeneral. As shown in Figure 3, EulerGeneral produces an Euler diagram where sets "a" and "f" are represented by disconnected enclosed areas. It has Concurrency of 5.

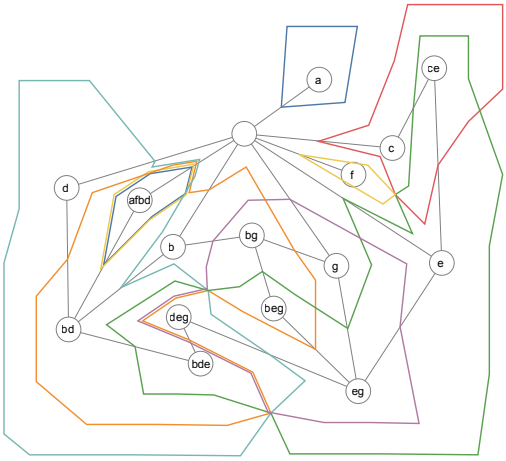

Fig. 3. The result of the running example with prior state-of-the-art EulerGeneral [25].

.

5 Evaluation

We compare our EulerMerge algorithm with the previous general Euler diagram embedder EulerGeneral [25]. Our algorithm produces Euler diagrams that have connected enclosed areas and no concurrency (see Section 3). It reduces the number of sets to be visualized. The EulerGeneral algorithm [25] visualizes all sets at the cost of containing disconnected enclosed areas and concurrency.

For EulerMerge, we need to consider the number of set merges required to produce a planar dual graph as well as those required to remove concurrency. We therefore count the number of set merging operations in Algorithm 3 and Algorithm 4. EulerGeneral might not obtain a wellformed diagram as duplicated curve labels and concurrency appear in many cases. Hence we quantify the number of duplicated curve labels as well as the amount of concurrency in the diagram. We note that in some complex cases, EulerGeneral does not produce an embedding at all. These cases have been removed from the data.

We use two collections of real-world set systems for evaluation (Table 1). First, the MovieDB data from the 2007 InfoVis contest [15]. A set system is formed from movies directed by a director with the movies as sets and the actors that appear in a movie as elements in the set. Second, a set system from

the Twitter Circles [19] collection contains sets that are interests groups formed by Twitter users. We include only diagrams that contain duplicated curve labels or concurrency.

For EulerMerge, Table 2 shows the average number of set merges required to produce a planar dual graph as well as those required to achieve concurrency. Concurrency reduction is over ten times more common than planarity reduction for both collections.

Collection	#Set Systems	Mean Sets	Mean Zones
MovieDB	225	4.4	7.78
Twitter Circles	59	5.92	8.24

Table 1. Real-world data summary, reporting the number of set systems per collection, the mean number of sets (Mean Sets) and mean number of zones (Mean Zones).

Collection	Planarity	Concurrency	Both
MovieDB	0.11	1.23	1.33
Twitter Circles	0.07	1.97	2.04

Table 2. EulerMerge: the average number of merges for achieving planarity (Planarity) and removing concurrency (Concurrency), together with the both merges (Both).

Collection	#Duplicated Curve Labels	Concurrency
MovieDB	0.35	4.63
Twitter Circles	0.41	6.19

Table 3. With EulerGeneral, the average number of duplicated curve labels (#Duplicated Curve Labels) and the average Concurrency count (Concurrency).

For EulerGeneral, Table 3 shows the average number of duplicated curves and the average Concurrency count. Duplicated curve labels occur in less than half of the diagrams generated, which may be artificially reduced by the failure of the algorithm to visualize some set systems, particularly complex ones.

In Table 2 and Table 3, it is shown that a smaller number of set merges are required to remove concurrency with EulerMerge, compared to the amount of concurrency in EulerGeneral.

6 Examples of Use Cases

Twitter Data Example. Figure 4 illustrates the process of applying our method to an example group of users from the Twitter data, where each user may belong to multiple interest groups. The set system consists of 13 sets and 32 intersections. The names of sets are shortened to single letters.

We first construct an initial dual graph, as shown in Figure 4(a) where vertices represent zones, namely, the 32 nonempty intersections of sets. However, the initial dual graph is nonplanar and, thus, it is not possible to generate an Euler diagram from it. We therefore employ Algorithm 3 to merge sets until the graph can be embedded without edge crossings. The first iteration merges

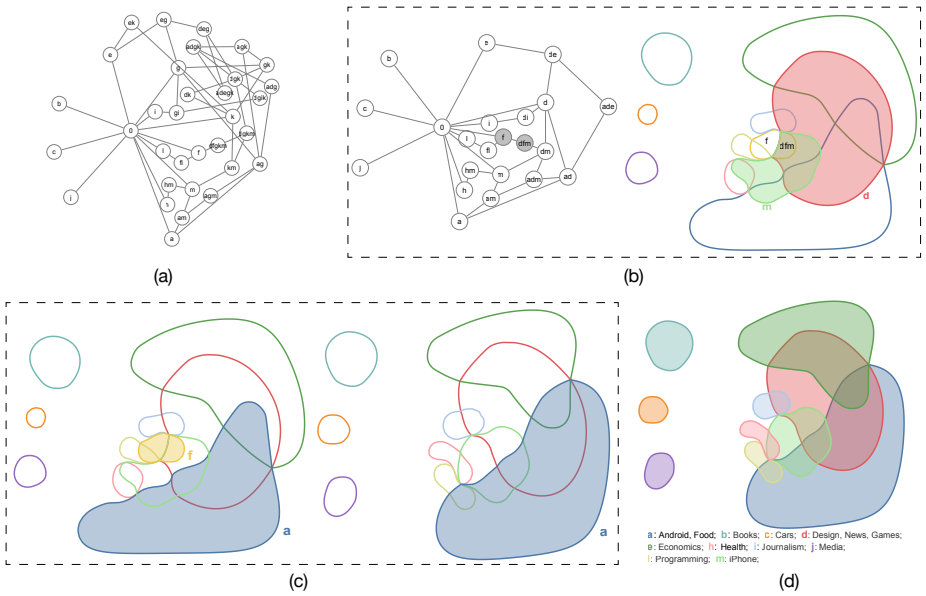

Fig. 4. The set merging process of a Twitter data set. (a) The original dual graph. (b) The first planar dual graph (left) and its corresponding Euler diagram (right) with concurrency. (c) From left to right: sets "a" and "f" merge into "a". (d) The final Euler diagram without concurrency.

sets "d" and "k". A second merge is required before reaching planarity, so "d" and "g" are merged. The two merges produce a planar dual graph with 11 sets and 21 intersections, see Figure 4(b). However, the result exhibits concurrency. For example, the vertices "f" and "dfm" are connected, but differ by two sets. To eliminate concurrency, we continue to merge sets using Algorithm 4, which merges "a" and "f", as shown in Figure 4(c). The resulting Euler diagram has no concurrency.

This example demonstrates that EulerMerge has achieved a desirable Euler diagram: after three set merges, two for planarity and one for concurrency, we can visualize this as an Euler diagram with connected enclosed areas and without concurrency, see Figure 4(d).

Movie Data Example (Director: Hooker, Keith). This data set consists of five sets in total. Each set represents a movie. We show set intersections where one or more actor appears in those films and no other films. As shown in Figure 5(a), the initial dual graph is not planar. The simplification process merges once for planarity (sets "a" and "c" merge into "a") and merges once for concurrency removal (sets "a" and "b" merge into "a"). Figure 5(b) left shows the first planar dual graph. Figure 5(b) right shows the corresponding Euler diagram, which exhibits concurrency. Figure 5(c) shows the concurrency removal step, where sets "a" and "b" in the left Euler diagram merge into "a" in the right Euler diagram. Figure 5(d) gives the final Euler diagram with the original set names.

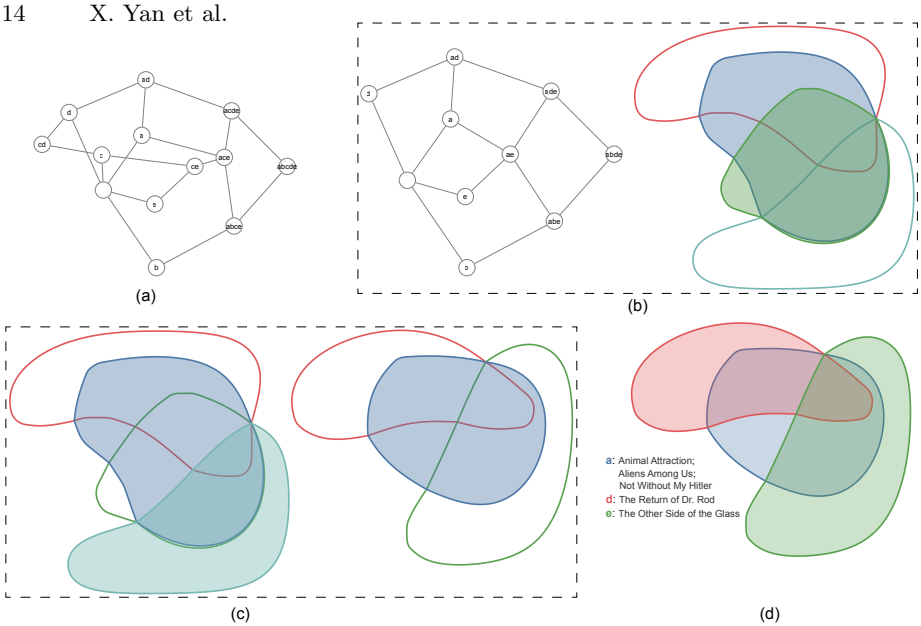

Fig. 5. The set merging process of a movie data set involving the director Hooker, Keith. (a) The initial dual graph. (b) Planarity merges: merging sets "a" and "c" into "a" and then merging "a" and "b" into "a". (c) Concurrency merges: merging "a" and "b" in to "a". (d) The final diagram with the original set names.

7 Conclusion

This paper provides, for the first time, an algorithm that uses set merges to simplify the layout of Euler diagrams to meet multiple wellformedness conditions. Merging just a few sets (often three or less) using EulerMerge results in a diagram without split sets or concurrency, thus producing a simplified diagram that is easier to comprehend than the non-wellformed alternative. Merging two sets into one reduces the amount of detail available to the user, which may be seen as a disadvantage. However, we see the simplification of a complex Euler diagram as a potential benefit. If needed, with the integration of a suitable user interface, the accepted visualization approach of "overview first, zoom and filter, then details-on-demand" [30] can be employed to reveal the missing details. We further perceive a simplified, wellformed diagram as a basis for hierarchical exploration via interactively expanding/collapsing merged sets, thus reflecting the precise set relations required by well-matchedness [2]. Finally, a future direction is to study the balance between visual readability and information loss, built upon additional set simplification approaches via element removal [27].

Acknowledgements We thank Dagstuhl Seminar 22462 "Set Visualization and Uncertainty" (November 13-18, 2022) for initializing this research. This research was partially supported by grants DOE DE-SC0021015 and NSF IIS-2145499. For the purpose of open access, the author has applied a CC BY public copyright license to any Author Accepted Manuscript version arising from this submission.

References

1. Abello, J., van Ham, F., Krishnan, N.: ASK-GraphView: A large scale graph visualization system. IEEE Transactions on Visualization and Computer Graphics **12**(5), 669–676 (2006)
2. Alsallakh, B., Micallef, L., Aigner, W., Hauser, H., Miksch, S., Rodgers, P.: The state-of-the-art of set visualization. Computer Graphics Forum **35**(1), 234–260 (2016)
3. Archambault, D., Munzner, T., Auber, D.: Tugging graphs faster: Efficiently modifying path-preserving hierarchies for browsing paths. IEEE Transactions on Visualization and Computer Graphics **17**(3), 276–289 (2010)
4. Bourou, D., Schorlemmer, M., Plaza, E.: Euler vs Hasse diagrams for reasoning about sets: A cognitive approach. In: International Conference on Theory and Application of Diagrams. pp. 151–167. Springer (2022)
5. Boyer, J.M., Myrvold, W.J.: On the cutting edge: Simplified $O(n)$ planarity by edge addition. Journal of Graph Algorithms and Applications **8**, 241–273 (2004)
6. Collins, C., Penn, G., Carpendale, S.: Bubble Sets: Revealing set relations with isocontours over existing visualizations. IEEE Transactions on Visualization and Computer Graphics **15**(6), 1009–1016 (2009)
7. Dunne, C., Shneiderman, B.: Motif simplification: improving network visualization readability with fan, connector, and clique glyphs. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems pp. 3247–3256 (2013)
8. Fischer, M.T., Frings, A., Keim, D.A., Seebacher, D.: Towards a survey on static and dynamic hypergraph visualizations. In: 2021 IEEE Visualization Conference (VIS). pp. 81–85 (2021)
9. Flower, J., Howse, J.: Generating Euler diagrams. In: International Conference on Diagrammatic Representation and Inference (DIAGRAMS). pp. 61–75 (2002)
10. Gansner, E.R., Hu, Y., Kobourov, S.: GMap: Visualizing graphs and clusters as maps. In: 2010 IEEE Pacific Visualization Symposium. pp. 201–208 (2010)
11. Huang, X., Liu, X.: Incorporating a topic model into a hypergraph neural network for searching-scenario oriented recommendations. Applied Sciences **12**(15), 7387 (2022)
12. Johnson, D.S., Pollak, H.O.: Hypergraph planarity and the complexity of drawing Venn diagrams. Journal of Graph Theory **11**(3), 309–325 (1987)
13. Kehlbeck, R., Gortler, J., Wang, Y., Deussen, O.: SPEULER: Semantics-preserving Euler diagrams. IEEE Transactions on Visualization and Computer Graphics **28**(1), 433–442 (2022)
14. Kestler, H.A., Müller, A., Kraus, J.M., Buchholz, M., Gress, T.M., Liu, H., Kane, D.W., Zeeberg, B.R., Weinstein, J.N.: VennMaster: Area-proportional Euler diagrams for functional go analysis of microarrays. BMC Bioinformatics **9** (2008)
15. Kosara, R., Jankun-Kelly, T.J., Chlan, E.: IEEE InfoVis 2007 contest: InfoVis goes to the movies. https://eagereyes.org/blog/2007/infovis-contest-2007-data
16. Koutra, D., Kang, U., Vreeken, J., Faloutsos, C.: VoG: Summarizing and understanding large graphs. Proceedings of the 2014 SIAM international conference on data mining pp. 91–99 (2014)
17. Kuratowski, C.: Sur le probleme des courbes gauches en topologie. Fundamenta mathematicae **15**(1), 271–283 (1930)
18. Lee, K., Jo, H., Ko, J., Lim, S., Shin, K.: SSumM: Sparse summarization of massive graphs. Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining pp. 144–154 (2020)

19. Leskovec, J., Krevl, A.: SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data (Jun 2014)
20. Linker, S.: Intuitionistic Euler-Venn diagrams. In: International Conference on Theory and Application of Diagrams. pp. 264–280. Springer (2020)
21. Mäkinen, E.: How to draw a hypergraph. International Journal of Computer Mathematics **34**(3-4), 177–185 (1990)
22. Micallef, L., Rodgers, P.: eulerForce: Force-directed layout for Euler diagrams. Journal of Visual Languages & Computing **25**(6), 924–934 (2014)
23. Michail, D., Kinable, J., Naveh, B., Sichi, J.V.: JGraphT—a java library for graph data structures and algorithms. ACM Transactions on Mathematical Software **46**(2) (2020)
24. Oliver, P., Zhang, E., Zhang, Y.: Scalable hypergraph visualization. IEEE Transactions on Visualization and Computer Graphics **30**(1), 595–605 (2024)
25. Rodgers, P., Zhang, L., Fish, A.: General Euler diagram generation. In: Stapleton, G., Howse, J., Lee, J. (eds.) Diagrammatic Representation and Inference. pp. 13–27. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
26. Rodgers, P., Zhang, L., Purchase, H.: Wellformedness properties in Euler diagrams: Which should be used? IEEE Transactions on Visualization and Computer Graphics **18**(7), 1089–1100 (2011)
27. Rottmann, P., Rodgers, P., Yan, X., Archambault, D., Wang, B., Haunert, J.H.: Generating Euler diagrams through combinatorial optimization. Computer Graphics Forum **43**(3) (2024)
28. Rottmann, P., Wallinger, M., Bonerath, A., Gedicke, S., Nöllenburg, M., Haunert, J.H.: MosaicSets: Embedding set systems into grid graphs. IEEE Transactions on Visualization and Computer Graphics **29**(1), 875–885 (2023)
29. Shin, K., Ghoting, A., Kim, M., Raghavan, H.: SWeG: Lossless and lossy summarization of web-scale graphs. In: WWW Conference. pp. 1679–1690 (2019)
30. Shneiderman, B.: The eyes have it: A task by data type taxonomy for information visualizations. In: Proceedings of the IEEE symposium on visual languages. pp. 336–343 (1996)
31. Simonetto, P., Archambault, D., Scheidegger, C.: A simple approach for boundary improvement of Euler diagrams. IEEE Transactions on Visualization and Computer Graphics **22**(1), 678–687 (2016)
32. Simonetto, P., Auber, D.: Visualise undrawable Euler diagrams. In: 12th International Conference Information Visualisation. pp. 594–599 (2008)
33. Stapleton, G., Flower, J., Rodgers, P., Howse, J.: Automatically drawing Euler diagrams with circles. Journal of Visual Languages & Computing **23**(3), 163–193 (2012)
34. Stapleton, G., Rodgers, P., Howse, J., Zhang, L.: Inductively generating Euler diagrams. IEEE Transactions on Visualization and Computer Graphics **17**(1), 88–100 (2011)
35. Wilkinson, L.: Exact and approximate area-proportional circular Venn and Euler diagrams. IEEE Transactions on Visualization and Computer Graphics **18**(2), 321–331 (2012)
36. Zhou, W., Nakhleh, L.: Properties of metabolic graphs: biological organization or representation artifacts? BMC Bioinformatics **12**(132) (2011)
37. Zhou, Y., Rathore, A., Purvine, E., Wang, B.: Topological simplifications of hypergraphs. IEEE Transactions on Visualization and Computer Graphics (TVCG) **29**(7), 3209–3225 (2023)