SPEULER: Semantics-preserving Euler Diagrams

Category: Research

Paper Type: algorithm/technique

(a) Original (`https://xkcd.com/2122/`) (b) Reconstructed using our method (c) Well-matched result using our method

Fig. 1: Venn diagrams are often used to highlight complex interactions of sets. This example from `xkcd.com` shows which adjectives can be used in combination (a). Using our method, we can recreate this manually created Venn diagram (b). Here, the diagram contains empty intersection. In these cases, Euler diagrams (c) provide a more faithful representation of the data.

Abstract—Creating comprehensible visualizations of highly overlapping set-typed data is a challenging task due to its complexity. To facilitate insights into set connectivity and to leverage semantic relations between intersections, we propose a fast two-step layout technique for Euler diagrams that are both well-matched and well-formed. Our method conforms to established form guidelines for Euler diagrams regarding semantics, aesthetics, and readability. First, we establish an initial ordering of the data, which we then use to incrementally create a planar, connected, and monotone dual graph representation. In the next step, the graph is transformed into a circular layout that maintains the semantics and yields simple Euler diagrams with smooth curves. When the data cannot be represented by simple diagrams, our algorithm always falls back to a solution that is not well-formed but still well-matched, whereas previous methods often fail to produce expected results. We show the usefulness of our method for visualizing set-typed data using examples from text analysis and infographics. Furthermore, we discuss the characteristics of our approach and evaluate our method against state-of-the-art methods.

Index Terms—Euler diagrams, Venn diagrams, set visualization, layout algorithm

1 INTRODUCTION

Set-typed data is ubiquitous across many different research areas, such as multi-label classification [47] in machine learning, RNA and DNA sequencing [13,19,33] in computational biology, and topic modeling [6] in natural language processing. There are two prominent methods to visualize set relations. Venn diagrams [45] show all possible relations between sets. In contrast, Euler diagrams [15] only depict non-empty relations. When visualizing this kind of data, typical set-related tasks are the number of datapoints per set or overlap between sets. To facilitate these tasks, many special-purpose visualizations have been developed [2]. Still, traditional Venn and Euler diagrams remain an essential tool for showing set intersections because they are easy to read, familiar to most users, and can incorporate data points directly. As such, they are often part of larger systems, such as *UpSet* [23].

Due to their combinatorial nature, the construction of Venn diagrams is straightforward. However, automatically creating Euler diagrams of high quality remains a challenging task, in particular for highly intersecting datasets. An Euler diagram should only include relations that are present in the data and avoid introducing superfluous areas. Further, the diagram should be monotone [5]. We call Euler diagrams that adhere to these properties *semantics-preserving*. In Section 2, we provide a formal description of these characteristics. These do not confuse users with empty intersections, and it increases readability as similar intersections are placed close to each other. An example result of our method and the impact of the above-mentioned properties is shown in Fig. 1c. The Euler diagram on the right has lost the symmetry of the Venn diagram (Fig. 1b) but represents the data faithfully.

First, we introduce and formalize the properties of Euler diagrams. Next, we propose a two-step algorithm for constructing such diagrams efficiently. The first step computes the *Euler dual*, a graph representation of the diagram. The second step creates the *Euler diagram*, whose curves follow guidelines [5] for creating intuitive Euler diagrams. We show the usefulness and characteristics of our algorithm on three examples from different domains and compare our method to previous work. In summary, the main contributions of this paper are:

- SPEULER, a **novel method** for constructing semantics-preserving Euler diagrams that yield fast and reliable results.
- Extensive **analysis of existing construction methods** and how they relate to properties of the Euler diagrams.
- **Three examples** from different domains that show the characteristics and potential of our approach.
- An **extensive evaluation** based on established guidelines of Euler diagrams and direct comparison to state-of-the-art methods.

2 CHARACTERISTICS OF EULER DIAGRAMS

Before we go into the previous work that is related to our method, we want to introduce important properties and concepts of Euler diagrams that will help to understand the subsequent sections. Formally, an Euler diagram is a set of smooth, closed Jordan curves that represent the different sets [11]. Together, these curves comprise various areas in the

drawing that represent the intersections of the sets. All set relations that exist in the data can be described by the *abstract description*—a list of the existing intersections. Euler diagrams can exhibit several different properties that directly influence their appearance and effectiveness in visualizing information. The two most important properties are well-formedness and well-matchedness, as defined by Chow [11].

Properties An Euler diagram is **well-formed**, if it is *simple* (i.e. at most, two curves meet at any given point and there is no concurrency), and exactly a *single curve* represents each set. In a **well-matched** Euler diagram, all intersections are correctly represented, thereby retaining the semantics from the original data: each intersection is represented only once, and the diagram does not contain areas of intersections that are not part of the abstract description. Alsallakh et al. [2] discuss different properties of algorithms for Euler diagrams and their connection to well-formedness. However, there is no such discussion for the well-matchedness and the interplay between both properties, which plays a big role in the effectiveness of the diagram [18]. The two properties are visualized in Fig. 2, which shows a Venn diagram with 4 curves and their 16 intersections. We use uppercase letters to refer to a curve or all nodes that participate in a set, and lowercase letters to refer to specific intersections, which are faces (also called *zones*) in the diagram. We will revisit this simple example throughout the next sections to help showcase our method. Fig. 2 shows the visual differences of adhering to only one or both of these two properties for the same data. Each zone is marked with its respective intersection. As can be observed in Fig. 2a, all four curves intersect on the lower-left corner, resulting in concurrent lines. By creating a well-matched and well-formed diagram, this can be avoided (Fig. 2b). It is important to note that many abstract descriptions exist, for which both properties cannot be satisfied at the same time, requiring a trade-off. However, as analyzed by Chow [11], it is currently not possible to infer for a given abstract description if it is possible to maintain both properties. If a trade-off has to be made, we adhere to the guidance of the work by Chapman et al. [9], which concludes that users prefer well-matched diagrams over well-formed ones. As a result, in these cases, our algorithm always produces well-matched diagrams while minimizing the violations of well-formedness.

Euler Dual A key concept that frequently shows up in construction algorithms is modeling the Euler diagram as a graph. Instead of thinking about the Euler diagram as a set of curves, it can be modeled directly as an edge-labeled graph, called the *Euler graph*. In this representation, each intersection of the curves is represented by a node, and each curve segment is represented by a link, labeled with the respective curve of the underlying original Euler diagram. Instead of creating the Euler diagram directly from the data using curves, it is also possible to indirectly create it by constructing the **Euler dual** of the Euler graph. Each node in the Euler dual represents a face of the Euler graph, and neighboring zones are represented by linked nodes in the Euler dual. However, in theory, all nodes that differ by one set could be linked in the dual—a graph that contains all possible links is therefore called the *super dual*. The *rank* of a node in the Euler dual equals the number of sets participating in that intersection. We can find an ordered representation of the Euler dual by grouping all nodes of the dual that have the same rank. The resulting graph is the *rank-based Euler dual*. Fig. 2c and Fig. 2d show the respective rank-based duals of Fig. 2a and Fig. 2b—the non-pairwise intersection of Fig. 2a is equal to the face *ABCD* in Fig. 2c. In comparison, all the faces of Fig. 2d are quads—we will explain what this means for the diagram in Section 5.

3 Related Work

Many set visualization approaches have been proposed in the past. Good starting points are the survey of Venn diagrams by Ruskey and Weston [38], or Rodgers [34], who focuses on Euler diagrams. Alsallakh et al. [2] offer a comprehensive survey of set visualizations and group the techniques based on their best-suited tasks: Element tasks, set relation tasks, and element attributes tasks.

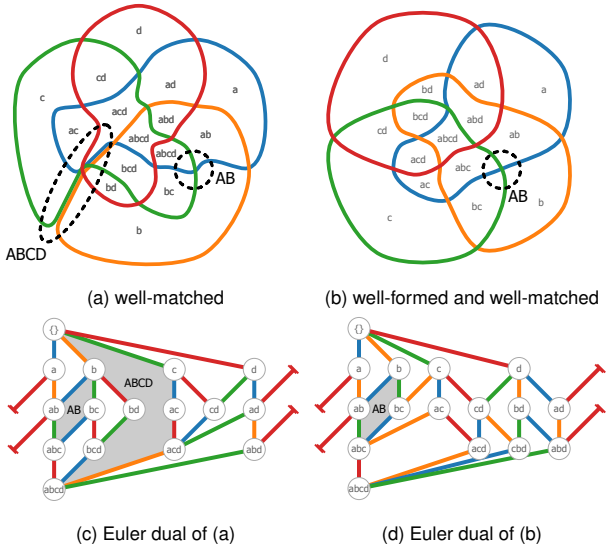


Fig. 2: (a) A well-matched diagram and (b) an additionally well-formed diagram. Well-matched diagrams may exhibit concurrent curves and points where more than two curves intersect, e.g., the intersection of curves *ABCD*. On the other hand, well-formed diagrams do not have these problems and only have pairwise intersections, e.g., *AB*. (c) and (d) show the ranked-based duals for (a) and (b). The concurrency surfaces as face *ABCD* in (c). The well-formed diagram instead only contains faces with 4 surrounding links. We will explain the impact of this in Section 5.

3.1 General Set Visualization

Alternative approaches to visualize set-typed data are matrix and aggregation-based techniques, such as UpSet [23] or RadialSets [1]. These are usually very well suited for element and element attribute tasks. However, they can be verbose to show all set relations at once when the data is complex.

For spatial data, such as maps, there are also techniques that focus on highlighting the connections between sets, such as BubbleSets [12] or KelpFusion [27]. Most methods are not able to directly encode information of the original data points in a unified visualization. For this task, Venn and Euler diagrams are especially well suited and therefore have been combined with glyphs [28], and graphs [31, 39]. Finally, Jacobsen et al. [21] propose using the metro map metaphor to visualize set relations in their MetroSets technique. The visualization can show individual data points for each set relation, and the layout can be fine-tuned according to different optimization strategies.

3.2 Constructing Venn and Euler Diagrams

Venn diagrams always show all possible set relations, with many different methods for their construction [3, 14, 34, 37, 45]. Euler diagrams are more flexible in this regard, but many construction algorithms are limited to specific abstract descriptions and might produce unexpected results [11, 16, 29, 35, 41].

Inductive methods construct diagrams by adding one curve at a time. Venn himself proposed an inductive method to create diagrams for any amount of curves. Edwards later proposed an alternative inductive construction method that creates diagrams by projecting the curves onto a sphere [14]. This method always creates diagrams that are well-formed and well-matched. However, for a larger number of sets, the result becomes hard to understand as the area of new zones becomes smaller and smaller. Other methods focus on the creation of simple, convex Venn diagrams, e.g., Mamakani et al. [25], which are aesthetically more pleasing. Ruskey et al. [37] use a general Venn construction method to analyze methods that create symmetric Venn diagrams. nVenn [32], a recently developed area-preserving Euler-like visualization technique, allows users to get a compact overview, even for larger set counts. They

Table 1: Details of different construction methods for any amount of curves and their properties.

Method	Construction	Any relation	**Well-matched**	**Well-formed**	Monotonicity	simple	Duplicate curves	Non-pairwise intersections
SCD [37] / nVenn [32]	Euler dual	yes	no	no	yes	no	no	yes
Stapleton [43]/ Rodgers [36]	direct	yes	no	no	yes	yes	yes	no
Venn [45]	direct	yes	no	**yes**	yes	yes	no	no
Edwards [14]	direct	yes	no	**yes**	yes	yes	no	no
vennEuler [48]	direct	no	no	**yes**	yes	yes	no	no
eulerr [22]	direct	no	no	**yes**	yes	yes	no	no
Chow-Ruskey [10]	Euler dual	yes	**yes**	no	yes	no	no	yes
Simonetto [42]	Intersection graph	yes	**yes**	no	yes	yes	yes	no
MetroSets [21]	hypergraph	yes	**yes**	no	no	-	-	yes
Flower [16, 17]*	Euler dual	no	**yes**	**yes**	yes	yes	yes	no
Our method	Euler dual	no	**yes**	**yes**	yes	yes	yes	no

*Note: The authors only provide a rough sketch of their method.

use a conventional Venn construction algorithm [37] as its initial layout and adapts it using a force-directed optimization. It heavily relies on the initial positioning and parameters of the force-directed strategy.

If the given dataset does not cover all possible set relations, Venn diagrams produce additional (unwanted) zones, and the diagram is not well-matched. For diagrams that are not well-matched, there is a discrepancy between the semantically correct representation of the abstract description and the visualization. Oftentimes, this problem is solved using shading to mark such additional faces [43,45]. In any case, this encodes unnecessary information that the reader has to process. A solution to this mismatch is well-matched Euler diagrams.

By design, methods that create Euler diagrams are usually well-matched. Their drawback, however, is that they often cannot make any guarantee about the aesthetics of the diagrams, i.e., their well-formedness. Results might contain crossings, concurrent curves, and non-smooth shapes. To alleviate this problem, Stapleton et al. [43] proposed an inductive method to create (semi) well-formed Euler diagrams using circles. Such diagrams weaken the constraints of the well-formedness and allow curve labels to be used multiple times. A current hindrance in the application of Euler diagrams is that most methods only produce expected results for certain datasets. Users do not know beforehand which method will produce well-formed or well-matched diagrams or if it will produce a valid result at all. Existing implementations often fail silently without producing any results or create unwanted zones without communicating this to the user.

It is challenging to create a well-formed and well-matched diagram for any abstract description because of the intricate interplay between the different properties. Therefore, many construction methods that only optimize for one property often cannot make guarantees for the others. This can be seen in Table 1. Usually, an Euler diagram is either *directly* constructed via curves or *indirectly* through an intermediate representation, which is then transformed into the Euler diagram. Examples are constructions using the Euler dual, Euler graph, connectivity graph, closeness graph, or intersection graph. Based on the surveys by Ruskey [38], Rodgers et al. [34], and Alsallakh et al. [2], we created Table 1, in which we compare different properties of Euler and Venn construction methods.

It should be noted that the properties of the final Euler diagram highly depend on the used construction steps as well as the properties of the intermediate representations. As we can observe from the table, direct construction methods usually produce well-formed diagrams, as they directly model the curves. This means the produced curves are usually constrained heavily, for example, by only using circles. As a trade-off, they only produce Venn diagrams or introduce unwanted zones for higher set counts. Alternatively, indirect methods only create the exact intersections needed and then transform the graph to the diagram but fail to create well-formed diagrams from them. Some methods try to transform non-well-formed diagrams into more aesthetic ones, but doing this in hindsight is often not possible. Examples can be found in [35, 42–44]. There is only a single approach that allows for the creation of Euler diagrams of any amount of curves that are both well-matched and well-formed. Flower et al. [16] propose an initial sketch of a solution but do not propose a general implementation. They resort to heuristics to create solutions for less than 5 curves. There are two main differences between our algorithm and the approach by Flower et al. [16]: They do not use the rank-based dual as an intermediate, and they cannot fall back to a sub-optimal solution when no well-formed and well-matched result exists.

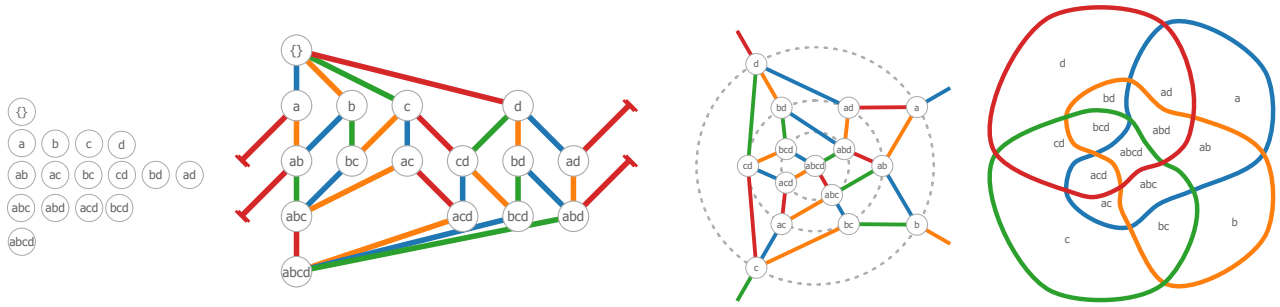
3.3 Evaluation of Euler Diagrams

As mentioned previously, the properties of Euler diagrams can be generally divided into well-matched and well-formed diagrams. However, there are many more properties that influence the semantics (e.g., monotonicity) and the aesthetics (e.g., shape, color, and symmetry). A general overview is given by Blake et al. [5], which introduces different guidelines that good Euler diagrams should adhere to. They directly compare real-word examples with adapted diagrams, which follow their proposed guidelines. Comparing both, such diagrams improve user comprehension. However, it is still unknown which of the guides might have a larger impact, and how they might influence each other. There are several studies that analyze the readability of well-matched vs. well-formed diagrams [9, 46]. Chapman et al. [9] compare various types of set diagrams and found that linear diagrams outperform all other methods, followed by unshaded Euler diagrams. They explain their results by the well-matchedness of those approaches, combined with well-formedness as a secondary influence. Rodgers et al. [36] evaluate methods that combine the Euler diagram with a graph of the datapoints. As their results are not consistent with previous studies of the same methods, they suggest that this might be due to them using datapoint specific tasks, whereas the previous studies used intersection related tasks. They conclude that for graph specific tasks, the properties that we summarize as "semantics preserving" may explain why some methods perform better than others. Wallinger et al. [46] compare Euler diagrams with MetroSets and LineSets for set-related tasks.

To conclude: it is still an open problem to design and implement an algorithm that produces well-matched and well-formed Euler diagrams for any amount of curves if the abstract description allows for it. Generating Euler diagrams with specific properties was also identified as an open problem by Alsallakh et al. [2]. Depending on the existing relations in the data, some properties are impossible to guarantee. We therefore propose a semantics-preserving construction method that generates Euler diagrams for any amount of curves. It creates well-matched and well-formed diagrams if allowed for by the data. If not, we retain the well-matchedness and relax as few individual properties as possible that infringe the well-formedness.

4 Overview

Our method constructs Euler diagrams for a given list of sets and their intersections—the abstract description. For example, the three sets



(a) Abstract description (b) Rank-based Euler dual (c) Circular layout (d) Final diagram

Fig. 3: Overview of our method: After finding all set intersections that exist in the dataset (a), the rank-based Euler dual is created from the abstract description (b). The graph is then transformed to be circular, and nodes are arranged in a well-distributed manner across several rings (c). In (d) we create the final curve for each set using splines.

$\{A,B,C\}$ can have relations $\{\emptyset,a,b,c,ab,bc,ac,abc\}$, where we abbreviate the zone $A \cap B \cap C$ with abc. However, in real-world data usually not all intersections are realized, for example, the intersection ac could be missing. For some abstract descriptions, it is possible to find well-matched and well-formed diagrams. However, there are also many configurations where this is not possible—in these cases our algorithm yields well-matched diagrams, while minimizing the violations of the well-formedness property. We provide further discussion on the influence of the abstract description on these properties in Section 8.

Our algorithm consists of four main steps, see also Section 4. Starting from the **abstract description** (Fig. 3a), we first find the appropriate order in which we place each set. Our algorithm then iteratively grows the graph based on this order while ensuring that new nodes conform to the well-formed property. After finding the connected, planar **rank-based Euler dual** (Fig. 3b), our algorithm arranges the nodes in a **circular layout** (Fig. 3c), which we then use to draw the curves that correspond to each set. Because of the properties of the dual, it is possible to generate a planar **Euler diagram** (Fig. 3d) from this circular layout. We use smooth curves to create compact and simple shapes. In contrast to other techniques, we guarantee a semantic match between the data and the final diagram. In addition, by creating mostly simple set curves and diagrams, we support the readability of the diagram, avoiding unnecessary crossings and concurrency of curves.

To demonstrate the usefulness and evaluate the characteristics of our algorithm, we implemented a prototype in JavaScript and D3[1]. This implementation also allows stepping through the individual steps of our algorithm. The prototype shows exemplary abstract descriptions that can be found in the paper, as well as different interactions that support set-related tasks such as visual identification of subsets and hovering. The implementation of our prototype, together with the example datasets, can be found online[2].

5 Rank-based Euler dual

We already introduced the rank of an intersection in Section 2, which is the number of its involved or participating sets. In the context of the Euler dual, we will call these intersections *nodes*. For example, the node a has rank 1, while node ab has rank 2. In the rank-based Euler dual, there can only be a link from a node with rank r to a node with rank $r+1$. This means that with each link, an additional set gets added to the intersection. In the example above, this is the set b, which we call the *color* of a link. By definition, a link always has a single distinct color. Accordingly, we color the links in the figures containing Euler duals throughout this paper.

Our goal is to draw Euler diagrams with minimal violations of the well-formedness property. As we create our diagram using the dual, we need to find the equivalent property in the dual that guarantees a well-formed result—the faces of the dual. A face in the Euler dual is an area that is enclosed by links and nodes. *Monotone faces* are enclosed by exactly four links that have two alternating colors. It is also important to note that by this definition, monotone faces always span exactly three ranks in the Euler dual. Monotone faces are essential for well-formedness, as they limit how many curves can intersect at a given face—this directly corresponds to how simple the final Euler diagram will be. Fig. 2 shows an example of a monotone and a non-monotone face. Our goal is therefore to *remove* possible links and *reorder* the nodes across all ranks until we are left with a *connected*, *crossing-free* and *monotone* version of the dual. Computing the rank-based dual can be structured into three parts: First, we group the nodes and decide in which order these groups should be placed (lines 2–3). Second, we look at each of the groups and sort the nodes by rank, and improve the sorting using *consecutive ones sequences* (Section 5.2) and distance to the previous set group (lines 5–8). Finally, we place each node so that it maximizes the number of monotone faces while linking them to the already existing nodes in the graph, removing unwanted crossings (lines 13–19).

Algorithm 1: Dual construction algorithm

```
1  function createDual(nodes[])
2      G ← group nodes by extended set and rank
3      G ← sort G by rank(G_{i_0}) and len(G_i)
4      forall S in G do
5          R ← group S by rank
6          forall r of R do
7              cos ← Calculate COs for r
8              r ← sort r by len(cos) and dist_twin
9              forall n of r do
10                 list ← []
11                 forall co of cos_n do
12                     forall possible position p of co do
13                         pp ← {len(co), #mono, #cross}
14                         list.push(pp)
15                     end
16                 end
17                 sort list to maximize monotone faces
18                 insert_node(list[0])
19                 remove_crossings()
20             end
21         end
22     end
```

5.1 Grouping by Participating Sets

As described in the previous section, each node has one or more participating sets. To create the dual, we start by separating these nodes into groups (line 13). We first sort the sets by the lowest rank of each

[1] D3: `https://www.d3js.org`

[2] We will provide a cleaned-up FOSS version of the code upon acceptance.

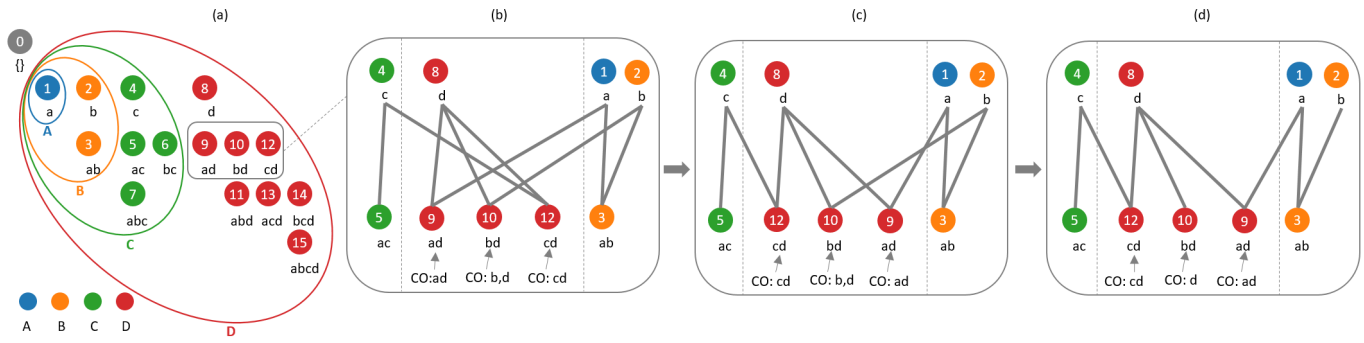


Fig. 4: Creating the Euler dual: (a) shows the initial order of nodes and their respective groups. (b) simply inserting the nodes with this initial order results in a dual that is non-planar. (c) we first remove crossings by changing the insertion order. (d) we finalize the graph by choosing the consecutive ones sequence which does not destroy monotone faces. The final result is a planar graph.

set, and the number of nodes they participate in (line 18). In practice, this means that sets that contain nodes with lower ranks such as *a* are considered first. We then iterate over the individual sets and group nodes that extend the nodes of the previous set with the current set (set extension) in the previously computed order. An example of this can be seen in Fig. 4a: Nodes are arranged vertically by rank and the different colors represent the resulting groups from each extension step. The numbers for each node describe the order in which nodes are added to the groups. Grouping the nodes using the extension of each set gives us a general order in which nodes are inserted into the dual graph. For each group, nodes are inserted in a rank-based order. Meaning nodes that have a lower rank are placed first. However, this on its own is not enough. If we were to insert the nodes in the order determined only by their rank and grouping order, the resulting dual will not be planar. This can be seen in Fig. 4b. Here, we are currently inserting the nodes from the red set with rank 2. If we naively insert and connect nodes, each node of the rank is connected to all nodes in the rank above using all possible links. Therefore, we need to establish the correct order for the nodes within a group, and the correct subset of links, which we will discuss next.

5.2 Consecutive Ones Sequences

To determine the order of nodes within a group we need to introduce the concept of *consecutive ones* (CO). Imagine the following scenario: Given an adjacency matrix of a graph, this graph has the consecutive ones property, if we can reorder the rows of the adjacency matrix so that all 1s in the columns are consecutive. This property was defined by Booth, and is true for graphs that have a planar embedding [7]. A *consecutive ones sequence* is then a group of consecutive nodes. As we want to create planar duals, we can use this property in our construction algorithm. Remember our goal is to insert nodes into the Euler dual so that as many monotone faces as possible are created. We do not need to realize all possible links between nodes. The only thing that we need to ensure is that revisedthe resulting graph is *planar* and *connected*. In the rank-based dual, it suffices to ensure the CO property for neighboring ranks. As there are more links in the abstract description than we need for the Euler dual, there are also multiple potential CO sequences, from which we choose the CO sequence that **maximizes the number of monotone faces**.

If we think back to the overall goal, which is to maximize monotone faces, we can see that the longer the consecutive ones sequence is, the more monotone faces are closed and created, when inserting the corresponding node. Therefore, we change the order of the nodes on each rank, so that nodes with longer CO sequences are placed before nodes with shorter consecutive ones sequences. If the length of COs is equal, we further sort the nodes by their respective *twins* in the previous group, without the current set, and sort them by their distance to the closest CO sequence of length 1 of the current group in the rank above (dist_twin in line 8) In the example of Fig. 4c, we can see that by reordering the nodes so that we first place node 12 and 9, and then node 10, the nodes 12 and 9 will not produce crossings anymore.

However, as shown in Fig. 4c, some crossings still remain. To adhere to the CO property, all possible CO sequences of a node have to be reduced to a single CO sequence. For example node 10 has two possible parents nodes in the rank above—nodes 2 and 8. The latter two are not adjacent, as they generate two CO sequences. So, to insert 10, we have to choose one of the two.

For each CO sequence and for each possible position in the current rank, we calculate a set of attributes that helps to make the decision where to place it. These attributes consider the length of the CO sequence, and the change in (#monotone Faces) and (#crossings)(line 8). We collect these attributes across the CO sequences and the possible positions of the node in a list. As an example, a new node might destroy an existing face, if we place it inside the face. Because the newly inserted node has to be connected to the next rank, a crossing will appear, and the #monotone faces decreases. After we have sorted the list accordingly, we insert the node at the current best position (line 18), resolve crossings (line 19), and move on to the next node.

Returning to our previous example, which can be observed in Fig. 4d, the CO sequence *d* is chosen, because this way no previously created face is destroyed. This is because the node 8 has already been inserted in the rank above, and has created an open space between the nodes 12 and 9. We therefore insert the node 10 into this open space, keeping existing monotone faces intact.

Once all nodes of the current rank have been placed, we continue on to the next rank. If we have placed all nodes of the current group, we move on to the next set, get all its nodes, sort the nodes on each rank, and insert the nodes, rank by rank. Using this method, it is possible to create Euler duals that are *connected*, *planar* and contain only *monotone faces*. We will discuss problematic cases, where this is not possible, in Section 8.

6 Circular Layout of the Euler Diagram

Based on the rank-based Euler dual, we can create the curves of the final Euler diagram. We do this by first removing the empty set and then arranging the nodes in a circular layout (Fig. 3c). At the center of this layout is the intersection with the largest rank, which is usually the full-set. The other nodes are placed on rings around the center, depending on their rank. Using this layout, we then devise a strategy to draw smooth curves that result in the final diagram (Fig. 3d).

6.1 Circular Layout

To guarantee a good distribution of nodes on each ring, we need to place the nodes at well-defined distances to each other. The rank with the largest number of nodes—usually the middle rank—is placed first to guarantee an overlap-free and well-distributed result. Rings are then placed outwards and inwards of this rank. The radii are chosen so that there is still enough space in the inner rings for all nodes, while the outer rings are not too distant from each other. Distributing nodes evenly on each ring can result in clutter in the ranks above and below. Therefore,

we place nodes so that enough space is reserved for their children and parents. Accounting for this, nodes with many children require more space compared to nodes with only a few children. This approach is similar to the layout of *radial trees*, but with the tree growing in both directions. The circular layout is then used to create the final Venn diagram. Fig. 5a shows the circular distribution of the Euler dual from Fig. 2c. On each ring, the nodes are well distributed.

6.2 Drawing Curves

To create an Euler diagram from the dual, the simplest approach would be to use the convex hull of the nodes for each set to create a closed shape. However, such a curve would not consider the nodes outside of the current set. This results in closed curves that create many unwanted zones and a very uneven distribution of areas across the faces. This is clearly not well-matched and decreases readability.

Therefore, we developed an approach to directly control the curve of each set by introducing additional virtual nodes that act as control points its shape. We call these nodes *gate nodes*. They lie on the same circular path as the intersection nodes but are distributed so that they always lie at the midpoint between two nodes on the ring (Fig. 5b, dark gray circles). When we move between ranks, we cross different circular paths in the circular layout, depending on the ranks of the current and following link. As we want to control the shape, we define where this crossing is allowed to happen: only at a gate node position. Due to the properties of the circular graph, which is still a dual of the Euler diagram, we can then create a set curve by finding the order of the links of each set and connecting the midpoints of the links with the gate nodes. This generates a path that moves between the rings, "cutting" the dual graph into two disconnected components.

Using the gate nodes in combination with the midpoints of the links in their respective link order, we create a mostly compact, closed curve for each set. Additionally, we can control the shape of the curve by using different interpolation strategies and adapting the link-mid point. We achieved the best curve results using Catmull-Rom splines. Fig. 5b shows a circular graph with the shape of a set defined by intersection link midpoints and gate nodes. Even though we use splines in our approach to maximize the smoothness, it would be possible to constrain the curve further to generate curves that can only use diagonal, rectangular, or octagonal lines.

6.3 Concurrency

Euler duals that only consist of monotone faces will only have pairwise crossings in the diagram. However, if we have a non-monotone face, this is not the case. Instead, we will create non-pairwise crossings. Curves that use the same gate nodes to cross a ring will produce a concurrent curve segment. To retain a well-matched diagram, we control the curves, which avoids creating unwanted zones. This means that for concurrent segments, each curve is offset according to the order in which they enter the concurrent segment. As the Catmull-Rom interpolation cannot handle straight line segments easily, we instead split the curve into different segments and add additional points to create segments that are concurrent. This can be observed in the bottom left part of Fig. 5b. If the concurrency is not a straight line but instead happens on the outside of the diagram, we create the curve normally but offset the curves as previously explained. These are then combined with normal curve segments to create the final closed smooth shape for each set.

7 Evaluation

We directly compare our method to several other state-of-the-art approaches across three different datasets. Many older set visualization techniques are not made publicly available [42, 46], so it is not possible to compare ourselves directly to them, or they only work on a very limited amount of set curves [30]. We evaluate our method against vennEuler [48], EulerR [22], SetNet [36], nVenn [32], and MetroSets [21]. vennEuler, EulerR, and SetNet only allow circles as curves, whereas nVenn allows for arbitrarily shaped curves. MetroSets are conceptually different from the other techniques as they only produce the Euler graph as an output. Therefore, we only compared them based on criteria that

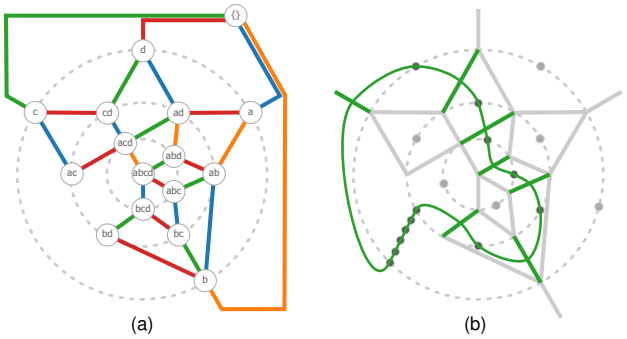


Fig. 5: To define the shape of the Euler diagram, we order the links for each set along the circles and shape them with gate nodes—shown here as grey dots—between the intersection nodes. This enables us to fine tune their shapes.

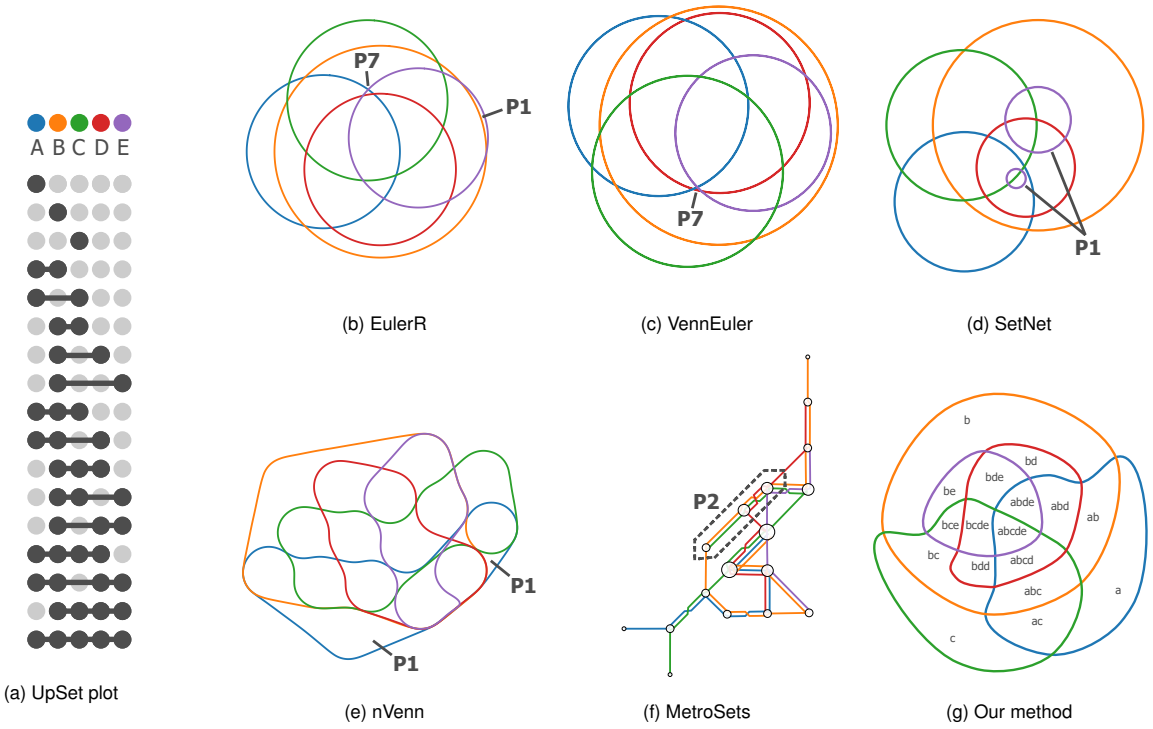
can be applied to the lines of the graph, such as line intersections, concurrency and overall compactness. Some of these approaches also have an additional weight parameter for each intersection that is used to create area-proportionate diagrams. Because such factors skew the comparison, we used an equal weight for all set intersections in these methods to unbias the individual areas. MetroSets allow to directly show data points—examples can be seen in Fig. 7b and Fig. 6f. This is not supported by other methods, including ours. To overcome this, the datapoints in Fig. 7 and Fig. 1 were added manually. As datasets, we chose three examples from different domains: topic modeling and info-graphics. These datasets show a wide variety in their structure, as well as the kind of datapoints that can be overlayed on top of the diagram. We will discuss the results according to well-matchedness, well-formedness, as well as additional properties that we are going to introduce in the next section.

7.1 Guidelines for Euler diagrams

We base our evaluation on the guidelines proposed by Blake et al. [5]. They define 10 different measures which can be used to judge the quality of Euler diagrams, ranked by their importance. There are three properties that we will not discuss in detail: Diverging lines, orientation, and color. Diverging lines are not applicable, and orientation is not considered by any method presented. It can also easily be changed by rotating the visualization. In order to strengthen the comparability of the methods we changed the color and style of all evaluated works to match ours. As Blake et al. [5] found that only outlines are preferred, we refrain from filling the curves. Some of the measures, such as **well-matchedness** (**P1**) and **well-formedness** (**P2**), have already been defined in Section 2. The others will be described briefly. They all relate to the form of the diagram but are not captured in the notion of well-formedness.

Curve Guidelines The **Compactness** (**P3**) defines how close a shape is to a perfect circle. Blake et al. [5] call this property *shape*, as they only consider circles. This is closely connected to the convexity of a shape, and there are further studies on the general understanding of convex vs. non-convex shapes [4, 20, 40]. In general, they conclude that convexity allows users to finish set-related tasks faster, albeit it might make individual curves harder to distinguish. **Smooth curves** (**P4**) are preferred by users and result in diagrams that are easier to read.

Diagram Guidelines **Symmetry** (**P5**) can also be beneficial if the curves are as symmetric as possible while retaining the features that distinguish individual faces. This property measures the similarity across all shapes in a uniform way. Circular approaches will always retain perfect symmetry, while more relaxed shapes might produce symmetric, pseudo-symmetric, or non-symmetric results. If the diagram is symmetric, finding a given intersection face can be challenging because many faces will have a similar shape. Therefore **shape discrimination** (**P6**) is another important property, which defines the uniqueness of individual faces, and allows for effective search tasks. **Zone area equality** (**P7**) measures the area of each face in relation to the other



(a) UpSet plot (b) EulerR (c) VennEuler (d) SetNet (e) nVenn (f) MetroSets (g) Our method

Fig. 6: Comparing relevant previous works for Euler diagrams of a topic modeling dataset. Problems in the results are marked: these can either be not well-formed (**P2**), not well-matched (**P1**), or create zones that only have very little area (**P7**). Our method produces a result that does not destroy the well-formed and well-matched properties. The areas of the zones are distributed evenly and the shape is compact.

faces. In general, for Euler diagrams that are not area-preserving, the area of each zone should be as similar as possible. Area-preserving Euler diagrams, in contrast, try to adapt the size of faces to be equal to a property, for example, to the number of contained data points (cardinality). Infringing this property means that users might misinterpret the difference in size as a difference in the cardinality of the face.

7.2 Topic Modeling

Using *latent Dirichlet allocation* [6], a common topic modeling algorithm, we extracted 5 topics from a political debate. The result of such a topic modeling algorithm is usually a list of keywords that describe each topic, together with their probability of belonging to said topic. We filter keywords to retain words for many combinations of topics while still creating an interesting abstract description that has a well-matched and well-formed diagram.

One common problem of topic modeling results is that it is very hard to visually compare them just using their descriptive keywords. Often words are attributed to multiple topics, but just representing them as a list, one cannot easily discern this. These words, however, might be of special interest to the user. They might describe all the topics very well, in which case the topics might be very similar to each other, or they might be general "common" words that should not be considered by the topic model algorithm, as they reduce the descriptiveness. In this section, we only show the resulting curves; the full diagram including the words can be found in the supplemental material.

Fig. 6 shows the results for the above dataset across all methods. For easier comparison, we have highlighted problematic zones in the respective diagrams, which result from infringements of the well-matchedness (**P1**), well-formedness (**P2**), and area-equality (**P7**) properties. Only a subset of the infringements is shown, as the diagrams might otherwise become unreadable. Some approaches are very similar (vennEuler and EulerR), while others diverge substantially.

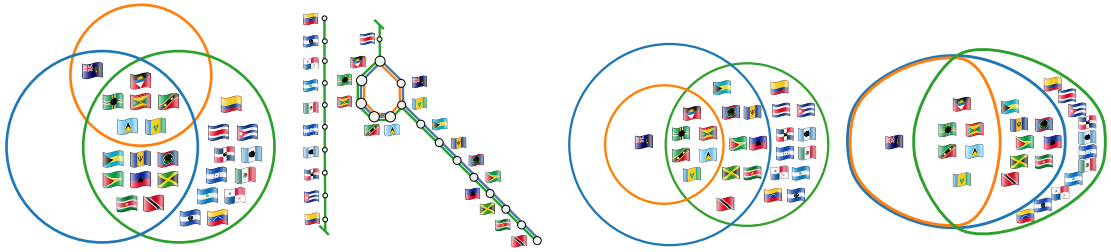
Most methods preserve the abstract description faithfully (**P1**). However, both EulerR and nVenn create intersections that do not appear in the abstract description. nVenn even realizes some relations with multiple faces, that they appear in different parts of the visualization. Regarding well-formedness (**P2**), we can observe that all circular visualizations (b-d) are simple. However, this comes at a cost: SetNet creates duplicate curves for E, while the other two approaches are not well-matched. nVenn and MetrosSets are not simple, as they contain non-pairwise crossings and concurrency. vennEuler, SetNet, and EulerR use circles and are therefore perfectly compact (**P3**). But nVenn also produces relatively compact shapes. MetroSets, on the other hand, produce a very spread out intersection graph that does not fit into a compact shape. All results that produce Euler diagrams create smooth curves (**P4**). Symmetry is not considered in any of the related work (**P5**). Circular methods create zones that are easily distinguishable, whereas the zones produced by nVenn are very similar to each other. As MetroSets only create intersection nodes, no real shape is created that can be considered here (**P6**). All of VennEuler, SetNet, and EulerR create very small areas that are difficult to recognize (**P7**).

Our method produces a both well-matched (**P1**) and well-formed (**P2**) result. The resulting shapes are mostly compact (**P3**). As we use curve interpolation, the produced curves are smooth (**P4**). In our method, some curves retain their symmetry at least partly (**P5**). However, because of this, the zones are also more similar, affecting how easy they are to distinguish (**P6**). The area of the zones remains relatively equal across all ranks (**P7**). In summary, our method retains the guidelines better than all other related works, except for zone discrimination, where we lie between nVenn and vennEuler/EulerR. Most important, the result is a well-matched and well-formed diagram.

7.3 Size Venn Diagram

As a second example, we show a Venn diagram published on `xkcd.com`[3] in Fig. 1a, that describes possible combinations of words in combination with five different adjectives: *little*, *large*, *small*, *great*, and *big*. The original visualization uses a 5-Venn diagram to show which words can occur together with these adjectives. However, there

[3] By RANDALL MUNROE at `https://xkcd.com/2122` CC BY NC



(a) Original (b) MetroSets (c) SetNet (d) Our method

Fig. 7: Here, we show a dataset of infographics concerning different supranational Caribbean bodies and their contained countries (a). While MetroSets is well-matched, the visualization requires a lot of space to show all the data points (b). Therefore, we introduced a discontinuity between *Costa Rica* and *Colombia*. SetNet introduces unwanted zones, as a does not contain any data points (c). Our diagram has only a single concurrency and is well-matched (d).

are some combinations for which no words were specified, such as *little*, *large* and *great*.

We can recreate the symmetric 5-Venn diagram used by the author using our algorithm, as can be seen in Fig. 1b. The words for each relation are added manually on top of the generated layout. This gives us the direct equivalent to the hand-made Euler diagram by the author. Then, we can remove the empty intersections and instead create a well-matched (**P1**) Euler diagram. In this case, the result is not well-formed (**P2**), so we retain minimal concurrency as well as one non-pairwise intersection. The resulting curves are mostly compact (**P3**) and smooth (**P4**). As only a few relations are empty, the diagram retains its high symmetry (**P5**), but in turn, many zones are similarly shaped (**P6**). The area is evenly distributed across the zones (**P7**). A comparison across the related works can be found in the supplemental material.

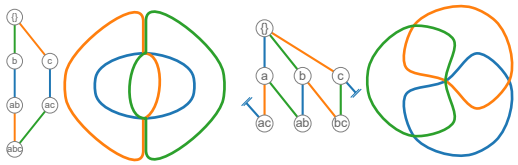
7.4 Supranational Caribbean Bodies

We recreate another info-graphic visualization published on Wikipedia commons[4], where countries are grouped by organizations. In this case, we look at all Caribbean countries that are contained in Supranational Caribbean Bodies. There are three different bodies: the *Association of Caribbean States*, the *Caribbean Community*, and the *Organization of Eastern Caribbean States*. However, not all intersections between the three exist, as there are no countries for some relations. The original visualization uses a 3-Venn diagram to visualize the relations. Existing relations are filled with flags that represent each country. This makes the visualization quite large, as a lot of space is needed to visualize the empty intersections, even though no data is shown.

As an additional comparison, we show how SetNet and MetroSets visualize this data set. In Fig. 7c, we can observe that SetNet does not always preserve well-matchedness (**P1**), as an empty zone is created. SetNet handles this by placing red dots inside faces that are part of the abstract description. Since we already show the flags of the countries that belong to each intersection directly, we chose to omit this in our recreation. MetroSets (Fig. 7b) shows all the data points, in this case countries, directly in the visualization. However, the visualization needs a lot of space, as lines extend outwards, resulting in a non-compact shape (**P3**).

Using our technique, we can visualize the relations as a well-matched (**P1**) Euler diagram. The diagram has concurrency, as some bodies do not contain countries that are only in this body, and is therefore not well-formed (**P2**). Curves are compact (**P3**) and smooth (**P4**). The symmetry is limited (**P5**), but still the zones are similar (**P6**). The area is evenly distributed across the zones (**P7**). Our visualization allows the reader to immediately see that there are four relations in total. The central intersection is shared for all three bodies, while there is a single relation that has outer concurrency.

[4]By WDCF at `https://w.wiki/39HJ`



(a) non-monotone faces (b) no common sink

Fig. 8: Examples with problematic abstract descriptions: (a) non-monotone faces will result in complex Euler diagrams. (b) If there is no shared intersection on the highest rank, a non-pairwise intersection will appear in the center of the diagram.

8 Discussion and Future Work

First, we will discuss runtime, problematic abstract descriptions, and alternative construction methods. Then we will further analyze the influence of design decisions on the aesthetics of the visualization.

8.1 Runtime

We performed experiments to compare the runtime of our approach to two other state-of-the-art methods. Because of its optimization strategy, MetroSets does not scale well with the number of nodes and increases in quadratic time [21]. SetNet extends the *iCircles* [43] algorithm and runs in polynomial time. For lower node counts, our algorithm has similar runtime ($n = 64$, 0.072s) to SetNet ($n = 64$, 0.14s), whereas MetroSets is a bit slower ($n = 64$, 2s)[5]. For larger number of intersection nodes, we can still achieve fast results ($n = 1024$, 27.48s). Our approach is a greedy algorithm that uses grouping and reordering to reduce the search space of possible positions for a new node. This allows us to avoid complex optimization strategies, making the output deterministic. From our experiments, we expect our algorithm to run in polynomial time with the grouping of nodes (Section 5.1) as the limiting factor. However, we hope to prove stronger bounds for this in future work.

8.2 Problematic Abstract Descriptions

As we have discussed before in Section 2, the layout of an Euler diagram strongly depends on the abstract description, and in particular, if there exists a well-formed solution for it. Our method handles this by relaxing the well-formedness properties if otherwise no such diagram can be found while guaranteeing the well-matchedness property. In contrast, other related works, such as eulerR, vennEuler, nVenn, or sometimes even SetNet, fail silently for these abstract descriptions or, arguably worse, create diagrams that are not well-matched. As we rate well-matchedness above all other attributes, this usually means for problematic abstract descriptions that we create non-monotone faces. An example of this is shown in Fig. 8. If there is no common intersection for all sets, all sets will intersect in the center of the diagram

[5]All experiments were run on a desktop PC with an Intel i5-8400 CPU.

and cause a non-pairwise intersection(Fig. 8b). If there is no monotone connection between the empty set and the lowest-ranked nodes, the outer curve will have concurrent curves, as can be seen in Fig. 7d.

There is one aspect of abstract description that we have not considered so far: In some cases, it can happen that the resulting diagram as a whole could be disconnected, or only some sets will have a disconnect, meaning nodes cannot be connected via a strong monotone link to at least one parent node and one child node. These datasets can currently not be visualized in our tool. We plan to remove this limitation in future work, by having special solutions for these disconnected components, for example, separation of the nodes so that each disconnected component is visualized by its own curve or concurrency on the rings, similar to the method of collapsing faces by Chow [10].

Finally, we want to point out that although our algorithm always produces a valid well-matched result, so far we have not proven that this result is optimal with respect to minimal violations of the well-formedness property. We also aim to pursue this in future work.

8.3 Alternative Construction Methods

Initially, we also tried alternative construction methods for Euler diagrams. We adapted the backtracking algorithm that Mamakani et al. [25] proposed for finding symmetric Venn diagrams to handle arbitrary Euler diagrams. The basic idea behind this technique is to find suitable crossings by permuting the order of curves. We used this approach to investigate the proportion of abstract descriptions for which well-formed diagrams exist. The number of possible intersections grows drastically with each additional set: For $n = 4$ there are 65K+ combinations—for $n = 5$ there are already 4M+ possible combinations. Furthermore, we only consider descriptions where each node has at least one incoming and one outgoing link, all sets have a common source as well as sink, and the empty set always exists. As described in Section 8.2, these are the abstract descriptions for which the diagram is connected. This lowers the combinations significantly to 3152 for $n = 4$. Using the modified backtracking, we found that for $n = 4$, only 125 out of the 3152 combinations have monotone diagrams (3,96 %). In the rest of the cases, the backtracking approach would find no solution at all. The reason for this is simple: Backtracking cannot readily find sub-optimal solutions, which motivates our proposed method. By allowing non-monotone faces as the last resort, our approach always yields a valid Euler diagram.

8.4 Design Considerations

Curves When we create the diagram using our method, we use customized splines to create a single, smooth curve for each set. We have also performed experiments using convex hulls and linear polyhulls for drawing curves, but these approaches cannot guarantee well-formedness, which is why we chose Catmull-Rom curves. In particular, we segment the curve into parts and use different strategies depending on what kind of curve segment occurs. There are three different types of segments: regular segments, U-turn segments, and concurrent segments. This differentiation allows us to be flexible in our choice of interpolation strategies, and we can control the smoothness of the curve by adapting the number of control points. For instances where concurrency cannot be avoided, we tried different approaches to mitigate it, e.g., concurrent lines or thinner lines so that equal line width is retained. Another approach would be dashed stroke segments with alternating colors.

Another interesting consideration is how to visualize the individual intersections. We show the curves without filled-in areas [5]. The problem with filling the areas of the curves is that, because of blending, each intersection will have a unique color that is not part of the original color set. With increasing number of sets, the visual difference between these colors decreases, which makes it hard to distinguish the intersections. Methods have been developed to alleviate these effects [2], but we consider their application out of the scope for this paper.

A limitation of using Catmull-Rom curves is that for large abstract descriptions, curves might become complex and non-convex, which might have a negative impact on the readability of the diagram. This effect can already be observed Fig. 1c, and it increases for highly-intersecting datasets, which can also be seen in the supplementary.

Area-proportionate Diagrams Cardinality is an important characteristic that is inherent to the data, if it exists in a given dataset. Currently, our method does not incorporate information about the cardinality of set intersection nodes. In the future, we plan to integrate a method to adapt the zones to given data point weights, creating area-proportionate diagrams, and fill these zones automatically with data points. This will also remove a current limitation of our tool, as it does not scale with large number of datapoints in a single zone.

Symmetry Another factor that influences the aesthetics of the diagram is symmetry, which many approaches do not retain. However, we believe that symmetry is an important aspect to investigate with regard to user engagement. Symmetric objects are often perceived as more aesthetic [8], especially so rotational symmetry [24]. However, striving for symmetry goes against the guidelines by Blake et al. [5], which we introduced in Section 7.1, as symmetry leads to zones that are visually very similar to another. These violate the shape discrimination property (**P6**) and can hinder user-understanding. An extreme example of this is nVenn. Ultimately, we believe that this a trade-off that has to be made depending on the goal of the visualization. For efficiency, the diagram should be simple and easy to read. If, on the other hand, the goal is to achieve an aesthetic representation, a more symmetric result can be chosen, based on user preference. By default, our method creates efficient diagrams. But our method also allows us to create symmetric results for some configurations, by simply incorporating the intersections that are missing from the abstract description—essentially creating a Venn diagram instead of an Euler diagram. Depending on the task, it can also make sense to draw empty intersections to emphasize the absence of instances. An example of this can be seen in Fig. 1b.

Using our method, it is also possible to create well-matched and well-formed Venn diagrams for any number of curves very fast. This is because of the inductive nature of Venn diagrams. Because each new set in a Venn diagram doubles the number of nodes, we can simply for each rank inverse the order of the nodes in the rank before, extend the nodes using the new set, and insert the nodes and links. However, currently some artifacts appear at higher set counts ($n \geq 8$). This is because the space of the individual faces is compressed a lot. We plan to improve our curve interpolation to handle Venn diagrams with more curves. The current problems can be observed in the supplementary material. A more difficult problem is creating symmetric Venn diagrams. Currently, we are able to create symmetric Venn diagrams for 5 and 7 sets by constricting the number of children a node can have. This influences the insertion strategy so that CO sequences are preferred that have fewer children while still maximizing monotone faces. It is still an open problem to find simple symmetric Venn diagrams for any prime number of sets, and the largest to be produced is a 13-Venn diagram [26].

9 Conclusion

We have presented spEuler, a novel approach to create well-matched Euler diagrams that focuses on creating mostly simple, planar, and connected solutions. The visualization of highly connected sets is a challenging problem, as the number of possible intersections increases exponentially with the number of sets. Our solution is fast and can scale to a large number of intersection nodes. We imagine our technique and accompanying visualization to be used as a part of larger a system that gives a brief and intuitive overview of set-typed data.

References

[1] B. Alsallakh, W. Aigner, S. Miksch, and H. Hauser. Radial sets: Interactive visual analysis of large overlapping sets. *IEEE Trans. Vis. Comput. Graph.*, 19(12):2496–2505, 2013. doi: 10.1109/TVCG.2013.184

[2] B. Alsallakh, A. Hanbury, H. Hauser, S. Miksch, and A. Rauber. Visual methods for analyzing probabilistic classification data. *IEEE Trans. Vis. & Comp. Graphics*, 20(12):1703–1712, 2014. doi: 10.1109/TVCG.2014.2346660

[3] A. Bannier and N. Bodin. A new drawing for simple Venn diagrams based on algebraic construction. *Journal of Computational Geometry*, Vol 8:No 1 (2017), 2017. doi: 10.20382/JOCG.V8I1A8

[4] M. Bertamini and J. Wagemans. Processing convexity and concavity along a 2-d contour: figure–ground, structural shape, and attention. *Psychonomic Bulletin & Review*, 20(2):191–207, 2012. doi: 10.3758/s13423-012-0347-2

[5] A. Blake, G. Stapleton, P. Rodgers, and J. Howse. The impact of topological and graphical choices on the perception of Euler diagrams. *Information Sciences*, 330:455–482, 2016. doi: 10.1016/j.ins.2015.05.020

[6] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[7] K. S. Booth and G. S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using pq-tree algorithms. *J. Comput. Syst. Sci.*, 13(3):335–379, 1976. doi: 10.1016/S0022-0000(76)80045-1

[8] N. Cawthon and A. V. Moere. The effect of aesthetic on the usability of data visualization. In *2007 11th International Conference Information Visualization (IV '07)*, 2007. doi: 10.1109/iv.2007.147

[9] P. Chapman, G. Stapleton, P. Rodgers, L. Micallef, and A. Blake. Visualizing sets: An empirical comparison of diagram types. In *Diagrammatic Representation and Inference*, vol. 8578, pp. 146–160, 2014. doi: 10.1007/978-3-662-44043-8_18

[10] S. Chow and F. Ruskey. Towards a general solution to drawing area-proportional Euler diagrams. *Electron. Notes Theor. Comput. Sci.*, 134:3–18, 2005. doi: 10.1016/j.entcs.2005.02.017

[11] S. C. Chow. *Generating and drawing area-proportional Euler and Venn diagrams*. PhD thesis, University of Victoria, 2007.

[12] C. Collins, G. Penn, and S. Carpendale. Bubble sets: Revealing set relations with isocontours over existing visualizations. *IEEE Trans. Vis. & Comp. Graphics*, 15(6):1009–1016, 2009. doi: 10.1109/TVCG.2009.122

[13] A. D'Hont, F. Denoeud, J.-M. Aury, F.-C. Baurens, F. Carreel, O. Garsmeur, B. Noel, S. Bocs, G. Droc, M. Rouard, and et al. The banana (musa acuminata) genome and the evolution of monocotyledonous plants. *Nature*, 488(7410):213–217, 2012. doi: 10.1038/nature11241

[14] A. Edwards. Venn diagrams for many sets. *New Scientist*, 121(1646):51–56, 1989.

[15] L. Euler. Lettres a une princesse d'Allemagne. *St. Petersbourg*, 1768.

[16] J. Flower, A. Fish, and J. Howse. Euler diagram generation. *J. Vis. Lang. Comput.*, 19(6):675–694, 2008. doi: 10.1016/j.jvlc.2008.01.004

[17] J. Flower and J. Howse. Generating Euler diagrams. In *Diagrammatic Representation and Inference, Second International Conference, Diagrams 2002, Callaway Gardens, GA, USA, April 18-20, 2002, Proceedings*, vol. 2317, pp. 61–75, 2002. doi: 10.1007/3-540-46037-3_6

[18] C. A. Gurr. Effective diagrammatic communication: Syntactic, semantic and pragmatic issues. *J. Vis. Lang. Comput.*, 10(4):317–342, 1999. doi: 10.1006/jvlc.1999.0130

[19] M. W. Hentze, A. Castello, T. Schwarzl, and T. Preiss. A brave new world of rna-binding proteins. *Nature Reviews Molecular Cell Biology*, 19(5):327, 2018.

[20] J. Hulleman, W. te Winkel, and F. Boselie. Concavities as basic features in visual search: Evidence from search asymmetries. *Perception & Psychophysics*, 62(1):162–174, 2000. doi: 10.3758/bf03212069

[21] B. Jacobsen, M. Wallinger, S. Kobourov, and M. Nollenburg. MetroSets: Visualizing sets as metro maps. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1257–1267, 2021. doi: 10.1109/tvcg.2020.3030475

[22] J. Larsson. *eulerr: Area-Proportional Euler and Venn Diagrams with Ellipses*, 2020. R package version 6.1.0.

[23] A. Lex, N. Gehlenborg, H. Strobelt, R. Vuillemot, and H. Pfister. Upset: Visualization of intersecting sets. *IEEE Trans. Vis. Comput. Graph.*, 20(12):1983–1992, 2014. doi: 10.1109/TVCG.2014.2346248

[24] A. D. Makin, M. M. Wilton, A. Pecchinenda, and M. Bertamini. Symmetry perception and affective responses: A combined EEG/EMG study. *Neuropsychologia*, 50(14):3250–3261, 2012. doi: 10.1016/j.neuropsychologia.2012.10.003

[25] K. Mamakani, W. J. Myrvold, and F. Ruskey. Generating simple convex Venn diagrams. *J. Discrete Algorithms*, 16:270–286, 2012. doi: 10.1016/j.jda.2012.04.013

[26] K. Mamakani and F. Ruskey. New roses: Simple symmetric Venn diagrams with 11 and 13 curves. *Discrete & Computational Geometry*, 52(1):71–87, 2014. doi: 10.1007/s00454-014-9605-6

[27] W. Meulemans, N. H. Riche, B. Speckmann, B. Alper, and T. Dwyer. Kelpfusion: A hybrid set visualization technique. *IEEE Trans. Vis. Comput. Graph.*, 19(11):1846–1858, 2013. doi: 10.1109/TVCG.2013.76

[28] L. Micallef, P. Dragicevic, and J. Fekete. Assessing the effect of visualizations on bayesian reasoning through crowdsourcing. *IEEE Trans. Vis. Comput. Graph.*, 18(12):2536–2545, 2012. doi: 10.1109/TVCG.2012.199

[29] L. Micallef and P. Rodgers. eulerforce: Force-directed layout for Euler diagrams. *Journal of Visual Languages & Computing*, 25(6):924–934, 2014. doi: 10.1016/j.jvlc.2014.09.002

[30] L. Micallef and P. Rodgers. EulerForce: Force-directed layout for Euler diagrams. *Journal of Visual Languages & Computing*, 25(6):924–934, 2014. doi: 10.1016/j.jvlc.2014.09.002

[31] P. Mutton, P. Rodgers, and J. Flower. Drawing graphs in Euler diagrams. In *Proceedings of Diagrammatic Representation and Inference*, vol. 2980, pp. 66–81, 2004. doi: 10.1007/978-3-540-25931-2_9

[32] J. G. Pérez-Silva, M. Araujo-Voces, and V. Quesada. nVenn: generalized, quasi-proportional Venn and Euler diagrams. *Bioinform.*, 34(13):2322–2324, 2018. doi: 10.1093/bioinformatics/bty109

[33] F. Ramírez, V. Bhardwaj, L. Arrigoni, K. C. Lam, B. A. Grüning, J. Villaveces, B. Habermann, A. Akhtar, and T. Manke. High-resolution tads reveal dna sequences underlying genome organization in flies. *Nature communications*, 9(1):1–15, 2018.

[34] P. Rodgers. A survey of Euler diagrams. *J. Vis. Lang. Comput.*, 25(3):134–155, 2014. doi: 10.1016/j.jvlc.2013.08.006

[35] P. Rodgers, L. Zhang, and A. Fish. General Euler diagram generation. In *Diagrammatic Representation and Inference, 5th International Conference*, vol. 5223, pp. 13–27, 2008. doi: 10.1007/978-3-540-87730-1_6

[36] P. J. Rodgers, G. Stapleton, B. Alsallakh, L. Micallef, R. Baker, and S. J. Thompson. A task-based evaluation of combined set and network visualization. *Inf. Sci.*, 367-368:58–79, 2016. doi: 10.1016/j.ins.2016.05.045

[37] F. Ruskey, C. D. Savage, and S. Wagon. The search for simple symmetric Venn diagrams. *Notices of the AMS*, 53(11):1304–1311, 2006.

[38] F. Ruskey and M. Weston. A survey of Venn diagrams. *Electronic Journal of Combinatorics*, 4:3, 1997.

[39] M. Sathiyanarayanan, G. Stapleton, J. Burton, and J. Howse. Properties of Euler diagrams and graphs in combination. In *IEEE Symposium on Visual Languages and Human-Centric Computing*, pp. 217–218, 2014. doi: 10.1109/VLHCC.2014.6883063

[40] G. Schmidtmann, B. J. Jennings, and F. A. A. Kingdom. Shape recognition: convexities, concavities and things in between. *Scientific Reports*, 5(1), 2015. doi: 10.1038/srep17142

[41] P. Simonetto, D. Archambault, and C. Scheidegger. A simple approach for boundary improvement of Euler diagrams. *IEEE Trans. Vis. & Comp. Graphics*, 22(1):678–687, 2016. doi: 10.1109/TVCG.2015.2467992

[42] P. Simonetto and D. Auber. An heuristic for the construction of intersection graphs. In *13th International Conference Information Visualisation*, pp. 673–678, 2009. doi: 10.1109/IV.2009.30

[43] G. Stapleton, J. Flower, P. J. Rodgers, and J. Howse. Automatically drawing Euler diagrams with circles. *J. Vis. Lang. Comput.*, 23(3):163–193, 2012. doi: 10.1016/j.jvlc.2012.02.001

[44] G. Stapleton, P. Rodgers, J. Howse, and L. Zhang. Inductively generating euler diagrams. *IEEE Trans. Vis. Comput. Graph.*, 17(1):88–100, 2011. doi: 10.1109/TVCG.2010.28

[45] J. Venn. On the diagrammatic and mechanical representation of propositions and reasonings. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 10(59):1–18, 1880.

[46] M. Wallinger, B. Jacobsen, S. Kobourov, and M. Nöllenburg. On the readability of abstract set visualizations. *preprint arXiv:2101.08155*, 2021.

[47] Y. Wei, W. Xia, M. Lin, J. Huang, B. Ni, J. Dong, Y. Zhao, and S. Yan. Hcp: A flexible cnn framework for multi-label image classification. *IEEE transactions on pattern analysis and machine intelligence*, 38(9):1901–1907, 2015.

[48] L. Wilkinson. Exact and approximate area-proportional circular Venn and Euler diagrams. *IEEE Transactions on Visualization and Computer Graphics*, 18(2):321–331, 2012. doi: 10.1109/tvcg.2011.56