Generating Effective Euler Diagrams

Almas Baimagambetov[1], John Howse[1], Gem Stapleton[1], and Aidan Delaney[1,2]

[1] Centre for Secure, Intelligent and Usable Systems, University of Brighton, UK
{a.baimagambetov, john.howse, g.e.stapleton}@brighton.ac.uk
[2] University of the South Pacific, Fiji
aidan@ontologyengineering.org

Abstract. Euler diagrams are used for visualizing categorized data, with applications including crime control, bioinformatics, classification systems and education. Various properties of Euler diagrams have been empirically shown to aid, or hinder, their comprehension by users. Therefore, a key goal is to automatically generate Euler diagrams that possess beneficial layout features whilst avoiding those that are a hindrance. The automated layout techniques that currently exist sometimes produce diagrams with undesirable features. In this paper we present a novel approach, called iCurves, for generating Euler diagrams alongside a prototype implementation. We evaluate iCurves against existing techniques based on the aforementioned layout properties. This evaluation suggests that, particularly when the number of zones is high, iCurves can outperform other automated techniques in terms of effectiveness for users, as indicated by the layout properties of the produced Euler diagrams.

1 Introduction

Visualizing data can be highly effective due to the human brain processing visual information significantly quicker than textual information [19]. Data items that are grouped into sets can be visualized by a variety of techniques, including LineSets [1] and linear diagrams [9, 18], with by far the most prominent being Euler diagrams [2]. They are widely used for visualizing sets in numerous application areas, including genetics [11], crime control [6], education [10] and classification systems [24]. The Euler diagram in Figure 1 conveys information about modules being studied by students. For instance, those who study Web also study Architecture and those who study Logic all study Maths but not Architecture or Programming. This diagram has features that are known to be effective for cognition, such as circular curves each of which is a unique colour [3]. Empirical studies have identified (in)effective features (further details are given later) including their so-called well-formedness properties [16], that are topological in nature, as well as their graphical properties [3]. In the context of automated layout tools, the difficulty of producing visualizations is compounded by the desire to produce effective drawings.

Given the extensive practical uses of Euler diagrams, it is unsurprising that many automated layout techniques exist for drawing them, with the first one

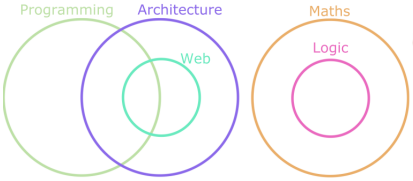


Fig. 1. An Euler diagram.

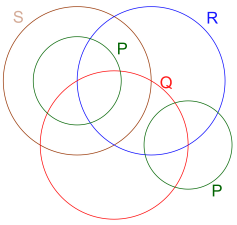


Fig. 2. Venn-4: iCircles.

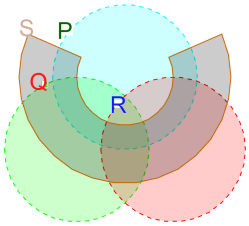


Fig. 3. Venn-4: GED-L.

being produced in 2002 [7]. However, all existing methods have limitations, such as not being able to represent all data sets, producing diagrams that inaccurately represent the data, or breaking well-formedness properties. Indeed, it is impossible to exactly visualize some sets in a completely well-formed way [7]. Consequently, layout methods have to prioritise which desirable features to enforce in their layouts and which to compromise. It is unknown which Euler diagram layout method produces the most effective diagrams.

The contribution of this paper is two-fold. Firstly, we introduce a new general Euler diagram drawing technique, called iCurves, that aims to ensure desirable properties hold (the diagrams are well-formed) at the expense of sometimes introducing extra zones into the diagrams. Secondly, we empirically compare the diagrams produced by iCurves against selected state-of-the-art techniques to gain insight into their relative effectiveness. Section 2 gives a brief summary of effective layout properties and existing techniques for automated Euler diagram layout. In section 3 we cover the concrete (drawn) and abstract Euler diagram syntax. Section 4 presents the theoretical underpinning of our method, iCurves, which we have implemented. A comparative evaluation of iCurves with existing techniques, using real-world data, is given in section 5. We conclude and discuss future work in section 6. The iCurves software and the diagrams on which our evaluation is conducted can be found at https://github.com/AlmasB/d2018.

2 Euler Diagrams: Background

As indicated in the introduction, effective Euler diagrams should be *well-formed* and possess other desirable graphical properties such as the use of circles and colours that are perceptually distinguishable. Five *well-formedness properties* have been identified and established to impact user understanding [16], which are as follows. *Simple curves:* no curve self-intersects. *Non-concurrent curves:* no parts of the curves run along the same path. *Only two-points:* whenever a point is passed through by curves, it is passed through at most twice; points that fail this condition are called *triple points*. *Connected zones:* all of the zones[3] in the diagram are connected components of the plane; a zone which is not

[3] A zone is a maximal region of the plane inside a subset of the curves and outside the remaining curves.

connected is called *disconnected*. *Distinct labels:* no two curves have the same label; curve labels that occur more than once are called *duplicated curve labels*. An example with duplicated curve labels is in Figure 2, which depicts Venn-4 (a Venn diagram representing four sets) using two curves for the set P; it was drawn using iCircles [21], with labels manually added for readability. A diagram which does not break any well-formedness property is called *well-formed*.

Rodgers et al. performed two comparative user studies which revealed that Euler diagrams with duplicated curve labels gave rise to significantly worse performance than those without [16]. They also found that disconnected zones caused significantly worse performance than the remaining properties. Thus, it is particularly important that drawing algorithms avoid disconnected zones and duplicated curve labels. The remaining well-formedness properties were shown to cause significantly worse performance using diagrams as compared to those which are well-formed. So, they should also be avoided where possible, but not at the expense of disconnected zones or duplicated curve labels. Therefore we can see, from the perspective of well-formedness, that Figure 2 is less desirable than Figure 3, which was generated using the General Euler Diagram drawing method in [15] extended to include a library of simple cases [17]; we call the extended method GED-L.

It has been suggested that the use of curves of particular geometric shapes impacts effectiveness [3]: circles were found to be more effective than ellipses, squares or rectangles. The choice of shape is a core part of layout algorithms, as well as routing the curves in order to display the correct set of zones. This is certainly no easy task and the resulting efficacy of the drawn diagrams is inherently intertwined with the design of the algorithms. By contrast, the use of colour applied to a diagram's curves is easily altered post-layout. Therefore, in this paper we will judge the efficacy of diagram layouts, in section 5, by focusing on the five well-formedness properties that impact understanding and the use of circles. We acknowledge that this does not take into consideration diagram aesthetics but there are currently no comprehensive results that indicate users' aesthetic preferences. One of our goals is to provide insight into the relative effectiveness of the diagrams produced by automated layout tools using known results concerning task performance, not user preference.

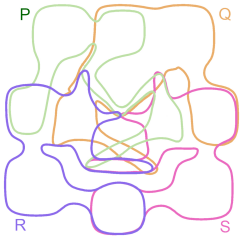


Fig. 4. Venn-4: Bubble Sets.

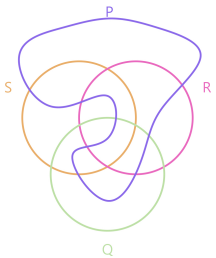


Fig. 5. Venn-4: iCurves.

As well as iCircles and GED-L, a large number of other layout tools for Euler diagrams have been provided, such as [5, 14, 20, 23]; see [2] for a comprehensive overview. All of these techniques have problems. For example, the first Euler diagram drawing method only produces well-formed diagrams but many sets cannot be visualised in this way [7]. Moreover, Bubble Sets [5] produces concurrency and disconnected zones (see Figure 4). Ultimately, there is no 'ideal' technique and it is necessary to accept some ineffective features in Euler diagram layouts. The ambition should be to minimise the ineffective features, whilst ensuring the diagram represents the desired sets. Our technique, iCurves (see Figure 5), sets out to avoid layout features known to be detrimental to task performance.

3 Euler Diagrams

We present prerequisite theory for iCurves. The definitions are the same as, or adaptations of, the material, found in [21–23]. An Euler diagram is a collection of closed curves, each of which has a label from some given set of labels, $\mathcal{L}$.

Definition 1. *An **Euler diagram** is a pair, $d = (C, l)$, where C is a finite set of closed curves drawn in the plane, each curve has a non-empty interior, and $l\colon C \to \mathcal{L}$ is an injective function that returns the label of each curve.*

Note that this definition is, for simplicity, more strict than some given for Euler diagrams, such as [21], which do not require non-empty curve interiors or l to be injective (i.e. when duplicated curve labels are allowed).

The closed curves partition the plane into *minimal regions*, which are connected components of the plane. In Figure 6, the eight minimal regions are enumerated. Another important concept is that of *zones*. A zone is a set of minimal regions that is inside certain curves (possibly none) and outside the remaining curves [23]. In Figure 6, minimal regions 7 and 8 form a single (disconnected) zone, inside P and Q, but outside R, representing the set $(P \cap Q) \backslash R$. In total, there are seven zones. A minimal region is a purely topological notion related to a drawn Euler diagram, whereas zones represent the intersection of sets.

Euler diagram drawing techniques, including iCurves, produce diagrams from *descriptions*. A description includes a list of *abstract zones*, where each abstract zone describes a zone. For example, in Figure 6, the zone formed from minimal regions 7 and 8 is described by $\{P, Q\}$ since it is inside P and Q only.

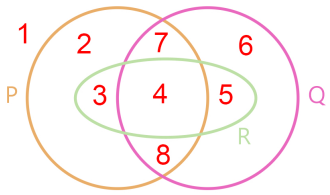


Fig. 6. Minimal regions and zones.

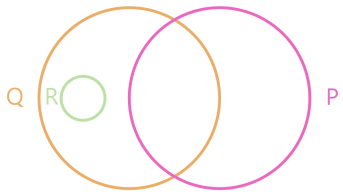


Fig. 7. A nested Euler diagram.

Definition 2. *A **description**, D, is a pair, (Z, L), where L is a subset of $\mathcal{L}$ and $Z \subseteq \mathbb{P}(L)$ such that $\emptyset \in Z$ and each label in L appears in at least one **abstract zone** in Z. We define $Z(D) = Z$ and $L(D) = L$.*

In Figure 7, the diagram, d, has a zone inside the curve P only, described by the abstract zone $\{P\}$, and a zone inside precisely P and Q, described by $\{P, Q\}$ and so on. So the description, D, of d has labels $L = \{P, Q, R\}$ and abstract zones $Z = \{\emptyset, \{P\}, \{Q\}, \{P, Q\}, \{Q, R\}\}$, where $\emptyset$ describes the zone outside all curves. We say that d is a *drawing* of D. Furthermore, we will abuse notation for conciseness and write D as $\{\emptyset, P, Q, PQ, QR\}$.

Some automated layout techniques, such as iCircles, draw diagrams with *additional* zones which are not indicated in the description. Such extra zones can be shaded, following the use of shading in Venn diagrams [25], to show that they represent empty sets. Our technique, iCurves, includes extra zones and uses shading in this way.

The concept of *nesting* is of use to us [8]. If the curves of an Euler diagram, d, form more than one connected component of the plane, as in Figure 7 where the curve R is separate from P and Q, then d is *nested*, otherwise d is *atomic* (as in Figure 6). Each connected component is called an *atomic component* [22]. So Figure 7 has two atomic components, whereas Figure 6 has one. Given a description, D, it is possible to identify *abstract atomic components* [22]. These abstract components correspond to atomic components in some drawing of D. In Figure 7, the two atomic components have the following (atomic) descriptions: $\{\emptyset, P, Q, PQ\}$ and $\{\emptyset, R\}$. Whilst this identification is an essential part of our drawing algorithm, we omit the (complex) theoretical details. What is important here is that the abstract atomic components can be computed. Our algorithm draws the atomic components separately and combines them to form the final diagram. We will refer to atomic and nested descriptions in the obvious way.

4 The iCurves Euler Diagram Generation Technique

The goal of the iCurves method is to draw well-formed diagrams, using circles where possible and with extra zones if needed. The drawing process of an abstract atomic component is described by the following sequence of steps:

1. Produce a *decomposition* of the abstract atomic component.
2. Draw the first curve as a circle.
3. Draw the next curve either as a circle or using a *modified dual graph* of the diagram so far. Repeat this step until all curves are drawn.

Given a description, D, a *decomposition* of D is a sequence of descriptions, $(D_n, ..., D_0)$, each with one fewer curve label than the previous, where $D_n = D$ and $L(D_0) = \emptyset$. In addition, $D_i = (Z_i, L_i)$ is formed from $D_{i+1} = (Z_{i+1}, L_{i+1})$ by removing a curve label, λ_{i+1}, from L_{i+1}, (so $L_i = L_{i+1} \setminus \{\lambda_{i+1}\}$) and from each abstract zone in Z_{i+1} to give Z_i; we write $D_i = D_{i+1} - \lambda_{i+1}$. One of our goals is to sensibly choose a decomposition that allows us to build an effective diagram.

A reversed decomposition is called a *recomposition* and corresponds to the order of curve addition described in step 3 above. The use of a decomposition in Euler diagram generation is not new [21, 23], but the order of curve label removal can have a profound impact on the effectiveness of the final diagram, as can the method used to draw the curves. We introduce a new strategy for producing a decomposition that takes into account the process by which we draw curves. We also introduce a new technique for drawing the curves themselves.

4.1 Drawing Curves in Euler Diagrams

Here, we demonstrate how to draw a new non-circular curve given an existing drawing of an atomic Euler diagram, d. This process constructs a *modified dual graph* [23] of d and then finds an *appropriate* cycle for curve addition. To illustrate, given Figure 8, we produce an *Euler graph* [4], shown in Figure 9. From this a dual graph is created, shown in Figure 10. The dual is modified to yield the graph in Figure 11. The cycle highlighted in Figure 11 can be used to add a new curve, S, as shown in Figure 12.

Constructing a Modified Dual Graph First, we obtain an *Euler graph*, $EG(d)$, from an Euler diagram, d: vertices are points at which the curves cross and the edges are the curve segments that connect the vertices. Next, we obtain a (standard) dual, $EGD(d)$, of an Euler graph. Finally, from $EGD(d)$ we obtain a *modified Euler dual*, $MED(d)$: for each edge, e, incident with the vertex, $v_\emptyset$, in the zone outside all of the curves, a new vertex is placed onto e; $v_\emptyset$ is deleted along with its incident edges; new edges are added which join the newly inserted vertices, so that the new vertices together with these new edges form a simple cycle that properly encloses the Euler graph.

Using a Cycle for Curve Addition Cycles in $MED(d)$ are used to add curves. In essence, a cycle's edges become curve segments which combine to form a curve. Our method is to use a *simple cycle*, s, to add new curves: a simple cycle does not pass through any vertex more than once.

Definition 3 (adapted from [23]). *Let $d = (C, l)$ be an atomic Euler diagram with a modified Euler dual $MED(d)$. Let s be a simple cycle in $MED(d)$ and let λ be a label not already used in d. Then d* ***extended by s and λ*** *is an Euler diagram, denoted $d + (s, \lambda)$, where $d + (s, \lambda) = (C \cup \{c\}, l \cup \{(c, \lambda)\})$ such that c is a closed curve, not in C, that traverses the cycle s and has label λ.*

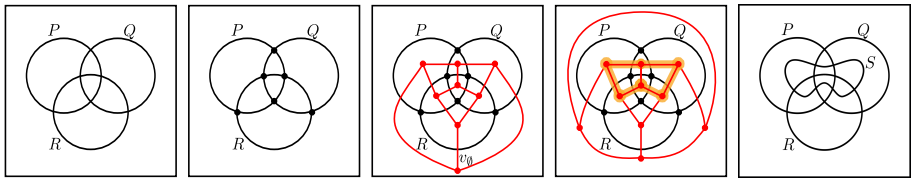


Fig. 8. Venn-3. **Fig. 9.** An Euler graph. **Fig. 10.** A dual. **Fig. 11.** A modified dual. **Fig. 12.** Added S.

Of course we cannot use any simple cycle to add a curve, since we have a description that we are aiming to draw. Consider

$$D = \{\emptyset, P, Q, R, PQ, QR, PR, PQR, PS, PQS, QS, QRS, PRS\}$$

which will be used as a running example; it is drawn, with an extra zone, $PQRS$, in Figure 12. Any decomposition of D is of the form $(D_4, D_3, D_2, D_1, D_0)$, where $D_4 = D$. Suppose that the first label we remove is S, giving $D_3 = \{\emptyset, P, Q, R, PQ, QR, PR, PQR\}$. A drawing, d_3, of D_3 can be seen in Figure 8, alongside its modified Euler dual, Figure 11. We now illustrate how we find a cycle in $MED(d_3)$ to create a drawing (with an extra zone) of D_4:

1. Identify the abstract zones in D_4 that contain S: $\{PS, PQS, QS, QRS, PRS\}$. Remove S from these abstract zones to give $\{P, PQ, Q, QR, PR\}$. These abstract zones necessarily occur in D_3.
2. Identify vertices in $MED(d_3)$ (Figure 11) that are placed in the zones with descriptions $\{P, PQ, Q, QR, PR\}$.
3. Next, seek a cycle that passes through at least these vertices. No cycle passes through exactly these vertices, but the cycle, which we call s, highlighted in Figure 11, passes through them all. Use s to add the curve, seen in Figure 12.

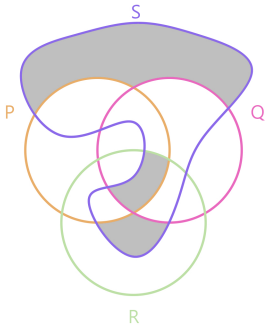
We call the diagram, d_4, in Figure 12, a *relaxed drawing* of D_4, as it contains an extra zone; iCurves will shade this non-required zone.

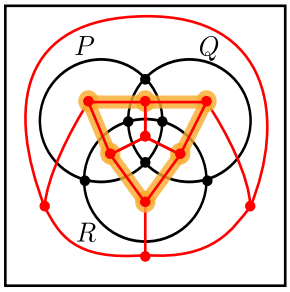
Definition 4. *Let d be an atomic Euler diagram. Let ZON be a non-empty set of zones in d. Let s be a simple cycle in $MED(d)$. If (i) vertices that arise from zones in ZON are in s and (ii) all vertices in s that are located in the zone outside all curves form one path in s then s is a **curve-adding cycle** for d **respecting** ZON.*

Importantly, curves added using curve-adding cycles ensure that *the resulting diagram is well-formed*, provided the diagram to which the curve is added is also well-formed. In our running example, the cycle used to create d_4 (Figure 12) respects ZON, where ZON contains the five zones with descriptions $\{P, PQ, Q, QR, PR\}$. In Figure 11, in addition to the cycle we used to add the curve S, we observe that there exist other curve-adding cycles for d_3 respecting $\{P, PQ, Q, QR, PR\}$. The choice of cycle profoundly impacts the layout of the final diagram. For example, using an alternative cycle to add the new curve could result in the diagram d_4', in Figure 13, which contains three extra zones. We note that the more vertices included in the chosen cycle, the more extra zones the resulting diagram will have. Therefore, iCurves strategically chooses a cycle respecting ZON which has few vertices. We now formalize the notions of a super description and a relaxed drawing.

Definition 5. *Let $D_1 = (Z_1, L)$ and $D_2 = (Z_2, L)$ be descriptions. If $Z_1 \subseteq Z_2$ then D_2 is a **super description** of D_1.*

Definition 6. *Let D_1 be a description with super description D_2. If d_2 is a drawing of D_2 then d_2 is a **relaxed drawing** of D_1.*





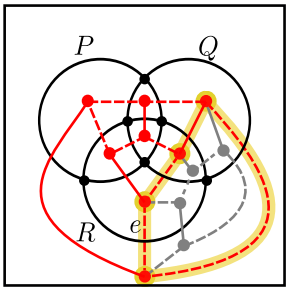


Fig. 13. A relaxed drawing. **Fig. 14.** A non-valid cycle. **Fig. 15.** A Hamiltonian dual.

H-extendible Euler diagrams It is important that iCurves can draw any description. Yet some curve-adding cycles respecting a given *ZON* set need not ensure we can add the next curve to the resulting diagram. We observe that given a pair of descriptions, (D_i, D_{i+1}), in a recomposition, the existence of a Hamiltonian cycle in the Euler graph dual of d_i (a relaxed drawing of D_i) ensures we can find a cycle in the modified Euler dual to add a curve that splits each zone in two *and* ensures the resulting Euler graph dual is Hamiltonian. Once a curve is added using a Hamiltonian cycle, we can shade any extra zones to give a relaxed drawing of D_{i+1}. Clearly, we want to use our curve adding strategy to minimize the number of extra zones. So the chosen cycle need not be Hamiltonian. However, the presence of a Hamiltonian cycle guarantees that there exists at least one curve-adding cycle and, thus, ensures the drawability of any D_{i+1} given (D_i, D_{i+1}). We want to ensure that the Hamiltonian property is preserved.

Definition 7. *Let d be an atomic Euler diagram. If an Euler graph dual of d is Hamiltonian then d is* ***H-extendible****.*

In this context, we place an additional constraint on curve-adding cycles. Such cycles partition the plane into two subspaces: bounded and unbounded. A vertex, v, is *inside* s if v is in the bounded subspace. It is possible for cycles that have vertices inside them not to give rise to diagrams with Hamiltonian duals. For example, in Figure 14, if we use the highlighted cycle to add a new curve then the resulting diagram will not have a Hamiltonian dual.

In addition, we place a stronger condition on edges of curve-adding cycles, which guarantees that, after adding a curve, the resulting diagram has a Hamiltonian dual. In Figure 15, the Euler graph dual includes a Hamiltonian cycle, shown using dashed (red) edges. A curve-adding cycle, s, in the Euler graph dual is highlighted by a semi-translucent path (in yellow). If s is used to add a new curve, the Euler graph dual would need to be updated for the new diagram. The resulting new edges and vertices are shown in grey. The dashed grey edges together with the dashed red edges, excluding e, form a Hamilitonian cycle in the new Euler graph dual. Notice the edge e and how its incident vertices are

used to join the red and grey paths to form the new Hamiltonian cycle thus extending the Hamiltonian property to the new dual. The presences of e in a Hamiltonian cycle is important: it ensures that we can join the dashed paths together using its incident vertices to create a new Hamiltonian cycle. In general, the existence of an edge in both a Hamiltonian cycle in $EGD(d)$ *and* in the curve-adding cycle, is sufficient to ensure that a Hamiltonian cycle exists in the new dual after adding a curve.

Definition 8. *Let d be an atomic Euler diagram. Let ZON be a non-empty set of zones in d. A curve-adding cycle, s, for d respecting ZON is **valid** if there are no vertices inside s and there is an edge, e, in s such that there exists a Hamiltonian cycle in $EGD(d)$ that contains e.*

Lemma 1. *Let d be an atomic Euler diagram that is H-extendible. Let ZON be a non-empty set of zones in d. Then there exists a valid curve-adding cycle for d respecting ZON.*

It follows that at each step in our diagram generation process we want to generate an H-extendible diagram, so that there exists at least one valid curve-adding cycle for the next step.

Theorem 1. *Let (D_i, D_{i+1}) be a pair of descriptions in a recomposition. Let $d_i = (C, l)$ be an H-extendible Euler diagram which is a drawing of D_i. There exists an H-extendible relaxed drawing, d_{i+1}, of D_{i+1} and obtained from d_i by adding a curve using a valid curve-adding cycle.*

We can readily show that given any description, D, there exists a relaxed drawing of D by appealing to Venn diagrams, which are known to have Hamiltonian duals [13]. For example, in Figure 16 we can shade any zone in a Venn-5 diagram and obtain a relaxed drawing of any description with five curve labels.

Lemma 2. *Let D be a description. There exists an atomic Euler diagram that is H-extendible which is a relaxed drawing of D.*

In summary, to guarantee drawability using iCurves, we select the *shortest* valid curve-adding cycles respecting ZON which ensures the resulting diagram is H-extendible. Shortest cycles lead to fewer extra zones being introduced when producing a relaxed drawing. The validity and H-extendible conditions ensure that we can continue adding curves to create a well-formed relaxed drawing of the required description.

4.2 Producing a Decomposition

We have now seen how iCurves adds curves, taking into account extra zones and drawability. The choice of decomposition also impacts on the quality of the final diagram. We present our strategy that leads to a particular choice for the order of curve label removal. Note that given an atomic description, D, iCurves draws an atomic diagram. Our strategy ensures that all descriptions in the decomposition of D are atomic. This is achieved by always choosing to remove a label that results in an atomic description, i.e. a *non-disconnecting* label.

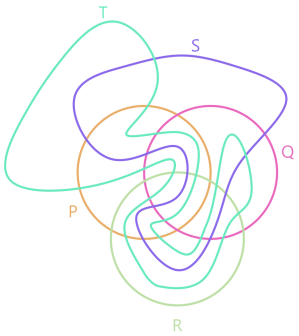


Fig. 16. Venn-5.

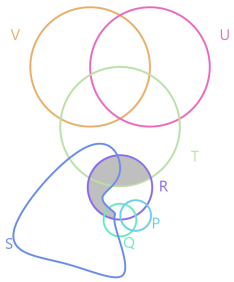


Fig. 17. A bad choice of decomposition.

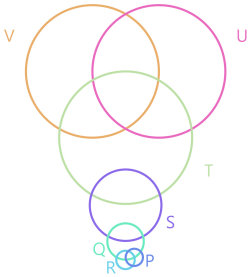


Fig. 18. A good choice of decomposition.

Lemma 3. *Let D be an atomic description. There exists a label, λ, in D that, when removed, results in an atomic description.*

This is an important part of our strategy, since removing disconnecting labels, yielding a nested description, necessarily leads to extra zones being included when drawing the diagram. For example, in Figures 17 and 18, the diagrams were generated from the same description. In Figure 17 the diagram has four extra zones and was drawn using a decomposition which first removed label S, giving a nested diagram; iCurves will not produce this layout as the decomposition violates our strategy. Figure 18 was drawn with iCurves using only atomic descriptions in the decomposition and has no extra zones. Our decomposition strategy to remove a label from D_{i+1} to give D_i is as follows:

1. If there is only one non-disconnecting label in D_{i+1}, remove it.
2. Else, see if any non-disconnecting label might be drawable as a circle that 'cuts across' one curve: for each non-disconnecting label, λ, see if $\{az \in Z(D_{i+1}) : \lambda \in az\} = \{az, az \cup \{\lambda_1\}\}$ for some $az \in Z(D_{i+1})$ and $\lambda_1 \in L(D_{i+1})$. If there are such labels, randomly choose one of them to remove.
3. Else, see if any non-disconnecting label might be drawable as a circle that 'cuts across' two curves, possibly adding one or two extra zones: for each non-disconnecting label, λ, see if $\{az \in Z(D_{i+1}) : \lambda \in az\} \subseteq \{az, az \cup \{\lambda_1\}, az \cup \{\lambda_2\}, az \cup \{\lambda_1, \lambda_2\}\}$ for some $az \in Z(D_{i+1})$ and $\lambda_1, \lambda_2 \in L(D_{i+1})$. If there are such labels, randomly choose one of them to remove.
4. Else, remove a label whose addition is likely to introduce the fewest extra zones: for each non-disconnecting label, λ, first compute $n = |\{az \in Z(D_{i+1}) : \lambda \in az \wedge az \setminus \{\lambda\} \notin Z(D_{i+1})\}|$, which gives the number of abstract zones that contain λ and do not have *neighbours*[4]. Next, compute m by finding the smallest set of abstract zones, Z', where $Z' \subseteq Z(D_{i+1} - \lambda)$ such that the graph (V, E), defined by $V = \{az \in Z(D_{i+1} - \lambda) : az \cup \{\lambda\} \in Z(D_{i+1})\} \cup Z'$ and $E = \{\{az, az'\} : az = az' \cup \{\lambda'\}\}$ for some $\lambda' \in L(D_{i+1} - \lambda)$, is a

[4] Two abstract zones are neighbours if their symmetric difference has one element.

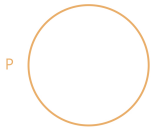


Fig. 19. d_1.

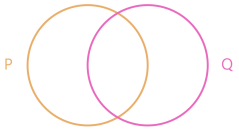


Fig. 20. d_2.

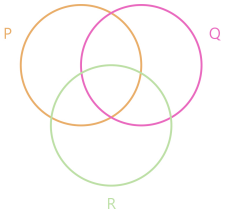


Fig. 21. d_3.

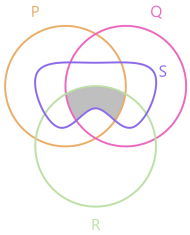


Fig. 22. d_4.

simple cycle; set $m = |Z'|$. Remove the label with the smallest $n+m$; if there is a choice, randomly choose one of them to remove.

A decomposition of our running example, D, can be formed by removing S, then R, followed by Q and P. The drawings of D_1-D_3 are in Figures 19-21. The diagram d_4, in Figure 22, is a relaxed drawing of D_4 and, therefore, of D. In D, all four labels are non-disconnecting and cannot be identified as (possibly) drawable with circles. Hence, we compute $n + m$ for each label. The value of n for P is 2 since there are two abstract zones, PS and PRS, where corresponding abstract zones, S and RS, are not in D. To compute m for P, we obtain the set V, which is formed from two sets $V' \cup Z'$, where V' arises from abstract zones in D that contain P. So, we have $V' = \{QR, R, \emptyset, Q, QS, S, RS\}$. The smallest set that can be added to V' to give a simple cycle is $\{QRS\}$. Thus, m for P is $|\{QRS\}| = 1$ and we have $n+m$ for P is 3. Similarly, $n+m$ for Q, R and S is 3, 2 and 1 respectively. S has the smallest $n+m$ and, so, is removed from D. Next, P, Q and R can be removed in any order as they are all drawable as circles.

4.3 Drawing Algorithm

To generate an atomic Euler diagram from an atomic description, D, using iCurves, call Algorithm 1 (page 12) with D as input.

Theorem 2. *Let D be an atomic description. Applying Algorithm 1 to D produces an atomic Euler diagram, d, that is a relaxed drawing of D.*

To draw a nested description, D, iCurves starts by splitting D into its atomic parts, say D_1, D_2, ..., D_n. Each D_i is then drawn using Algorithm 1. The diagrams are then displayed appropriately, reflecting the nested properties of D; the formal details are omitted due to space constraints. An example of a nested diagram drawn by iCurves can be seen in Figure 23. We are now in a position to state our main theorem:

Theorem 3. *Let D be a description. Then iCurves produces an Euler diagram, d, that is a relaxed drawing of D.*

Algorithm 1: Draw atomic component.

Input : An atomic description, D.
Output: An atomic Euler diagram, d, which is a relaxed drawing of D.
begin
  Set $dec(D) = (D_n, ..., D_0)$, removing labels using the strategy given above.
  D_0 has no labels and D_1 has a single label, λ_1. Draw the first curve as a circle with label λ_1. Call the resulting Euler diagram d_1. Trivially, d_1 is atomic, H-extendible and a drawing of D_1. Set $i = 1$.
  while $i < n$ **do**
    Set λ_{i+1} to be the label such that $D_i = D_{i+1} - \lambda_{i+1}$.
    if $i == 1$ **then**
      Add a circle with label λ_{i+1} to d_i, resulting in d_{i+1}.
    else
      Set $ZON = F_i(IN(D_i, D_{i+1}))$, where $IN(D_i, D_{i+1})$ is the set $\{az \in Z(D_i)\colon az \cup \{\lambda_{i+1}\} \in Z(D_{i+1})\}$ and F_i maps abstract zones of D_i to zones of d_i in the obvious way.
      Construct $MED(d_i)$.
      if *$|ZON| = 2$ and the zones in ZON are adjacent and the edge between the zones is in a Hamiltonian cycle in $EGD(d_i)$ or $|ZON| = 2$ and we can add two extra zones to ZON to form a valid curve-adding cycle or $|ZON| = 3$ and we can add one extra zone to ZON to form a valid curve-adding cycle or $|ZON| = 4$ and the zones in ZON form a valid curve-adding cycle* **then**
        Add a circle with label λ_{i+1} to d_i, resulting in d_{i+1}.
      else
        Set s_i to be a valid curve-adding cycle for d_i respecting ZON, selected using the strategy given above.
        Set $d_{i+1} = d_i + (s_i, \lambda_{i+1})$.
      end
    end
    d_{i+1} is atomic and a relaxed drawing of D_{i+1}.
    Increment i by 1.
  end
  To finish, set $d = d_n$.
end

5 Evaluation

We now set out to evaluate the layouts produced by iCurves against competing state-of-the-art techniques. It is not possible to evaluate against all implemented techniques as there are a large number of them. We selected techniques such that (a) the software is freely available, and (b) they could theoretically draw any description, even though the implementation may fail. So, the methods selected for comparison with iCurves are: Bubble Sets [5], GED-L [17], and iCircles [21].

We selected a stratified random sample of 40 data sets from the SNAP (Stanford Network Analysis Project) collection, which are derived from Twitter ego-networks [12]. The 40 data sets had between four and eight sets. The number of

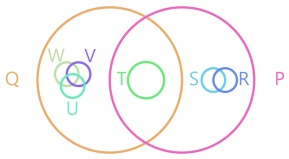


Fig. 23. A nested Euler diagram with 8 curves.

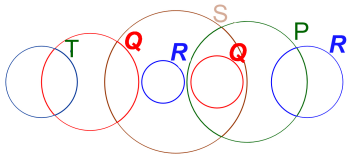


Fig. 24. iCircles.

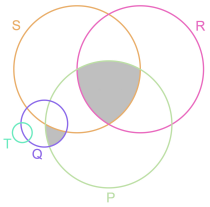


Fig. 25. iCurves.

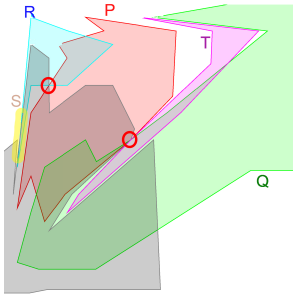


Fig. 26. GED-L.

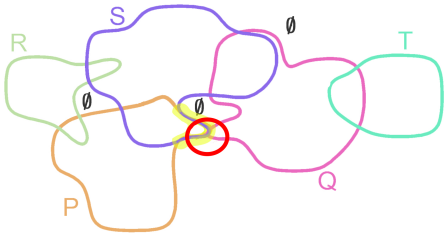


Fig. 27. Bubble Sets.

zones varied too: for each number of sets, we randomly selected two data sets from each quartile based on the number of zones. For example, the eight-set data sets in the SNAP collection had between 8 and 48 zones, with 'number of zones' quartiles $8-12$, $13-14$, $15-18$ and $19-48$; we randomly chose two data sets from each quartile. This approach to sampling was designed to provide insight into how the effectiveness of the layouts changed as the complexity of the data changed, as measured by the number of sets and the number of zones.

Each selected data set was drawn using each of the four techniques. We counted the number of times each well-formedness property was broken. For each curve label, we counted the number of 'extra' times it was used. For each disconnected zone we counted the number of extra minimal regions it comprised (i.e. one fewer than the number of minimal regions of which the zone comprised) on the basis that one minimal region per zone is needed. For concurrency, we counted the number of cases where two or more curve segments run along the same path and we also counted the number of triple points. There were no occurrences of non-simple curves. We also counted the number of non-circular curves. These counts provide a measure of the relative effectiveness of the diagrams. Lastly, we counted the number of extra zones present in the diagrams (i.e. zones whose abstraction is not in the drawn description). Sometimes the techniques failed to draw a given description, so our discussion below accounts for that.

The diagrams generated from the data sets, using all four techniques, are available from https://github.com/AlmasB/d2018. They are marked to show how we counted violations of the well-formedness properties. For example, Figures 24-27 illustrate the same data set drawn using each of the four techniques

Techniques	iCircles (40)			Bubble Sets (40)			GED-L(29)			iCurves (38)		
	c	a	n	c	a	n	c	a	n	c	a	n
Duplicated Labels	45	1.1	17									
Disconnected Zones				146	3.7	21	2	0.1	2			
Concurrency				51	1.3	12	12	0.4	5			
Triple Points				36	0.9	15	20	0.7	7			
Non-circles				240	6.0	40	46	1.6	7	11	0.3	8
Extra Zones	5	0.1	2	87	2.2	20				63	1.7	15

Table 1. Total counts and averages for the diagram features broken by the techniques.

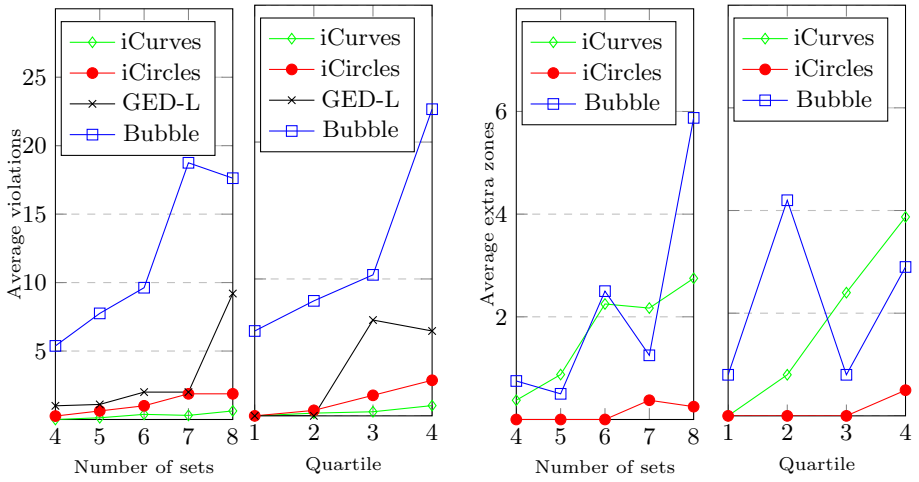


Fig. 28. The averages for the diagram features broken by the techniques.

(the labels are manually added for readability). In Figure 24 there are two duplicate curve labels. In Figure 25 there are two extra zones. In Figure 26, there is one case of concurrency and two triple points. In Figure 27, there are two cases of disconnected zones, one triple point and one case of curve concurrency.

The data we collected are summarised in Table 1. The column headings show how many of the 40 data sets the technique successfully drew; GED-L failed to generate 11 diagrams out of 40 and iCurves failed to generate 2. In most of the failed drawing attempts the number of zones was high. For each property, the total count, c, is given, together with the average number, a, of violations for each *drawn* diagram, and the number of diagrams, n, in which the violation occurs. When the count is 0, no entry is given to avoid clutter.

The data are also displayed in line charts in Figure 28. The two charts on the left show the average number of violations of the five well-formedness properties plus the average number of non-circular curves; one graph is by number of sets, the other by quartile. The other two charts show the number of extra zones by number sets and quartile respectively. Bubble Sets often produces diagrams with

disconnected zones and extra zones. Furthermore, none of the curves are circles. In iCircles all of the curves are circles, but as the number of zones increases iCircles produces duplicated labels and GED-L produces curve concurrency and triple points. Unsurprisingly, as the diagrams increased in complexity, as indicated by number of sets or zone quartile, the diagrams produced by all four techniques exhibited an increasing number of bad properties.

In summary, we can see that iCircles and Bubble Sets have particularly problematic features, such as duplicated curve labels and, respectively, disconnected zones. These data suggest that GED-L and iCurves strongly outperform iCircles and Bubble Sets based on these two well-formedness properties. GED-L often breaks other well-formedness properties (although not as many times as Bubble Sets) unlike iCurves. Considering effectiveness based just on the well-formedness properties, these data suggest that iCurves should be used rather than GED-L. In addition, iCurves more often used circular curves than GED-L, again pointing favourably towards iCurves. Of course, we must be mindful that the effective features of iCurves come at the expense of extra zones being present.

6 Conclusion and Future Work

Our technique, iCurves, generates well-formed diagrams, in contrast with other techniques. The evaluation has shown that iCurves can outperform other techniques with respect to layout features that are detrimental to task performance.

In the future, it would be beneficial to determine the effects of extra zones on diagram comprehension. Understanding the effects would allow us to gain deeper insight into the trade-off between duplicated labels and extra zones, for example in Figures 24 and 25. Moreover, whilst GED-L did not break any particularly problematic properties, the overall aesthetic qualities of the diagram were questionable. As can be seen from Figure 26, use of jagged lines makes the diagram convoluted. Currently there are no clear aesthetic criteria we can use to evaluate the diagrams, but properties such as symmetry and curve smoothness can be considered. Finally, we plan to extend iCurves to include graphs, so we can visualize grouped network data (elements joined with lines). Extending the technique in a way that primary spatial rights [5] are not assigned to either the sets or the network represents a considerable challenge.

References

1. Alper, B., Henry Riche, N., Ramos, G., Czerwinski, M.: Design study of LineSets, a novel set visualization technique. IEEE Transactions on Visualization and Computer Graphics 17(12), 2259–2267 (2011)
2. Alsallakh, B., Micallef, L., Aigner, W., Hauser, H., Miksch, S., Rodgers, P.: Visualizing sets and set-typed data: State-of-the-art and future challenges. In: Eurographics conference on Visualization STAR, pp. 1–21. Wiley. (2014)
3. Blake, A., Stapleton, G., Rodgers, P., Howse, J.: The impact of topological and graphical choices on the perception of Euler diagrams. Information Sciences 330, 455 – 482 (2016)

4. Chow, S.: Generating and Drawing Area-Proportional Euler and Venn Diagrams. Ph.D. thesis, University of Victoria (2007)
5. Collins, C., Penn, G., Carpendale, M.S.T.: Bubble sets: Revealing set relations with isocontours over existing visualizations. IEEE Transactions on Visualization and Computer Graphics 15(6), 1009–1016 (2009)
6. Farrell, G., Sousa, W.: Repeat victimization and hot spots: The overlap and its implication for crime control and problem-oriented policing. Crime Prevention Studies 12, 221–240 (2001)
7. Flower, J., Howse, J.: Generating Euler diagrams. In: Diagrams. pp. 61–75. Springer, (2002)
8. Flower, J., Howse, J., Taylor, J.: Nesting in Euler diagrams: syntax, semantics and construction. Software and Systems Modelling 3, 55–67 (2004)
9. Gottfried, B.: A comparative study of linear and region based diagrams. Journal of Spatial Information Science 2015(10), 3–20 (2015)
10. Ip, E.: Visualizing multiple regression. Journal of Statistics Education 9(1) (2001)
11. Kestler, H., Muller, A., Liu, H., Kane, D., Zeeberg, B., Weinstein, J.: Euler diagrams for visualizing annotated gene expression data. In: Euler Diagrams. (2005)
12. Leskovec, J., Krevl, A.: SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data (Jun 2014), accessed September 2017
13. Pruesse, G., Ruskey, F.: All Simple Venn Diagrams are Hamiltonian. ArXiv e-prints (Apr 2015)
14. Riche, N., Dwyer, T.: Untangling Euler diagrams. IEEE Transactions on Visualization and Computer Graphics 16(6), 1090–1099 (2010)
15. Rodgers, P., Zhang, L., Fish, A.: General Euler diagram generation. In: Diagrams. pp. 13–27. Springer (2008)
16. 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 (2012)
17. Rodgers, P.: General embedding method (with diagram library). http://www.eulerdiagrams.com/Library.htm, accessed October 2017
18. Rodgers, P., Stapleton, G., Chapman, P.: Visualizing sets with linear diagrams. ACM Trans. Comput.-Hum. Interact. 22(6), 27:1–27:39 (Sep 2015), http://doi.acm.org/10.1145/2810012
19. Sato, Y., Masuda, S., Someya, Y., Tsujii, T., Watanabe, S.: An fMRI analysis of the efficacy of Euler diagrams in logical reasoning. In: IEEE Symposium on Visual Languages and Human-Centric Computing (2015)
20. Simonetto, P.: Visualisation of Overlapping Sets and Clusters with Euler Diagrams. Ph.D. thesis, Université Bordeaux (2012)
21. Stapleton, G., Flower, J., Rodgers, P., Howse, J.: Automatically drawing Euler diagrams with circles. Journal of Visual Languages and Computing 23(3), 163–193 (2012)
22. Stapleton, G., Rodgers, P., Howse, J.: A general method for drawing area-proportional Euler diagrams. Journal of Visual Languages and Computing 22(6), 426 – 442 (2011)
23. Stapleton, G., Rodgers, P., Howse, J., Zhang, L.: Inductively generating Euler diagrams. IEEE Transactions on Visualization and Computer Graphics 17(1), 88–100 (2011)
24. Thièvre, J., Viaud, M., Verroust-Blondet, A.: Using Euler diagrams in traditional library environments. In: Euler Diagrams. ENTCS, vol. 134, pp. 189–202. (2005)
25. Venn, J.: On the diagrammatic and mechanical representation of propositions and reasonings. Phil.Mag 10(59), 1–18 (1880)