Eurographics Conference on Visualization (EuroVis) 2023
D. Archambault, R. Bujack, and T. Schreck
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 42 (2023), Number 3

RectEuler Supplementary

Patrick Paetzold[1], Rebecca Kehlbeck[1], Hendrik Strobelt[2], Yumeng Xue[1],
Sabine Storandt[1], and Oliver Deussen[1]

[1]University of Konstanz, Germany [2]IBM Research AI, US

1. MIP Constraints

We use several constraints to target the wanted properties described in the main publication. To enable the MIP to position the rectangles representing set curves and element groups, we add the coordinates of the upper left corner, denoted as (x_1, y_1), and respectively of the lower right corner (x_2, y_2), as variables to the model. First, we will introduce all used constraints and afterward describe the optimization objective used, subject to the constraints. All variables in the MIP are constrained to be non-negative.

H0 Element Group Size

Each element group has a fixed size depending on the elements it includes. This size has to stay constant throughout the optimization, and only the position of the element group should be changed. Since each element group's upper left corner and lower right corner are variables in the MIP, their values can change independently. Nevertheless, the relative position of both corners and thus the element group's width w and height h has to stay the same. Therefore, we constrain the variables of each element group to its constant width and height.

H1 Positioning Elements in Set

This second constraint enforces each element group e to stay inside all set rectangles they belong to. Each group must be fully enclosed in all rectangular representations of the sets it should be contained in. To include the set's name in the visualization, a margin at the top edge of each rectangle with height s is necessary. In Fig. A1a, the area reserved for the set name is highlighted in light red, and the margin with width d around each element group is indicated in light gray. The four constraints to express the described relations are given in Eq. 1.

$$\begin{aligned} e_{x_1} - d &\geq r_{x_1} \\ e_{y_1} - d &\geq r_{y_1} + s \\ e_{x_2} + d &\leq r_{x_2} \\ e_{y_2} + d &\leq r_{y_2} \end{aligned} \tag{1}$$

The first two constraints enforce the upper left corner of the element group to stay below the area reserved for the set's name and right of the left edge of r. The latter two constraints enforce the lower right corner of e to stay inside of r.

H2 Keeping Elements outside Set

In addition to the constraint enforcing an element group e to lie inside a rectangle r, it is equally important to model the exclusions of element groups from rectangles they do not belong to. We base our definition of the problem using MIP constraints on Kalvelagen [Kal17]. To prevent elements from being inside sets they are not contained in, more constraints are required, as the position of the element group in relation to the set rectangle is not predetermined beforehand and has to be determined during the optimization of the MIP. The element group is positioned left, right, above, or below the sets it is not contained in. Thus a boolean decision variable has to be introduced to represent this *or-relation* of the position. Consider the one-dimensional case that the element group e is positioned left or right of the rectangle r. Based on the technique to formulate either-or constraints in mixed integer programs, a constraint to position the element group e either left or right of the rectangle r is formulated as

$$e_{x_2} + d \leq r_{x_1} + M \cdot b \tag{2}$$

$$e_{x_1} + M \cdot (1 - b) \geq r_{x_2} + d \tag{3}$$

with $e_{x_1}, e_{x_2} \geq 0$ and $b \in \{0, 1\}$.
The constant M has to be chosen big enough to be greater than all other variables. For the sake of simplicity we just assume we know such a M (we will discuss how to find possible values for M in subsection 1.1). To give an insight into the effects the indicator variable b has on the positioning of e in relation to r, we discuss the influences of setting $b = 0$ and $b = 1$ in Eq. 2 and 3.
Firstly, suppose $b = 1$: M is active in Eq. 2 and inactive in Eq. 3. As M is greater than all other possible variables, it satisfies Eq. 2 for any possible value of e_{x_2}. Therefore Eq. 2 is inactive and does not influence the positioning of e. But Eq. 3 is active as $M \cdot (1 - b)$ evaluates to 0 and has therefore no influence on the constraint. To fulfill Eq. 3, e has to be positioned right of r with a margin of at least d.
Secondly suppose $b = 0$: Eq. 2 is activated, as M is not dominating the inequality. Thus to satisfy the constraint, e is positioned left of r with margin d. Eq. 3 is inactive, as the inequality is trivially satisfied by any possible value of e_{x_1}.

Similar constraints can be used to position e above or below r. But as e has to be above or below and right or left of r, not all 8 positions shown in Fig. A1b are possible. Only the four quadrants

e_1, e_3, e_5 and e_7 are feasible positions to the previously defined constraints. Therefore, more constraints and binary variables are necessary to position elements in all eight possible positions relative to r. Eq. 4 to 7 describe the constraints to place e in all possible positions in relation to r.

$$e_{x_2} + d \leq r_{x_1} + M \cdot b_1 \quad e \text{ is left of } r \tag{4}$$
$$r_{x_2} + d \leq e_{x_1} + M \cdot b_2 \quad e \text{ is right of } r \tag{5}$$
$$e_{y_2} + d \leq r_{y_1} + M \cdot b_3 \quad e \text{ is above } r \tag{6}$$
$$r_{y_2} + d \leq e_{y_1} + M \cdot b_4 \quad e \text{ is below } r \tag{7}$$

and furthermore:

$$b_1 + b_2 + b_3 + b_4 \leq 3 \tag{8}$$
$$b_1, b_2, b_3, b_4 \in \{0, 1\} \tag{9}$$

If $b1 + b_2 + b_3 + b_4 = 4$, all constraints from Eq. 4 to 7 would be inactive, as the position is dominated by M and the placement of e in relation to r could be arbitrary. Thus an exclusion couldn't be guaranteed. At least one of the constraints has to be active, by setting one binary variable b_1 to b_4 to 0. The placement of e left of r as shown in Fig. A1b is achieved by setting $b_1 = 0$ and $b_2, b_3, b_4 = 1$. By setting two binary indicators to 0, an element group can be positioned next to a corner of r.

H3 Distance of Set outlines

To create well-formed results, we want to avoid concurrent lines in the diagram; thus, the outlines of the rectangles have to keep a minimal distance d. To model the spacing of set outlines, all vertical and horizontal edges of a rectangle must maintain at least the distance d from all other vertical and respectively edges. Fig. A1c illustrates a possible relative position of the red highlighted horizontal edges. Either the bottom edge of r_1 is above r_2 as shown in Fig. A1c or their relative positions are switched. To model the positioning of one edge with at least distance d to a parallel edge we introduce the constraints in Eq. 10 and 11. Since these equations only ensure a desirable placement of one edge of r_1 and one of r_2, we need four binary variables and eight constraints to model the pairwise placement of the horizontal edges. Similarly, four binary variables and eight constraints are necessary for the vertical edges. Thus, for each pair of rectangles, eight binary variables with 16 constraints have to be formulated.

$$(r_{1_{y_2}} - r_{2_{y_2}}) + b_1 \cdot M \geq d \tag{10}$$
$$-(r_{1_{y_2}} - r_{2_{y_2}}) + (1 - b_1) \cdot M \geq d \tag{11}$$

We add these constraints only to rectangles that do share elements, as the margin between edges of sets not sharing elements is enforced by the constraints defined in H6.

H4 Bounding Rectangle

A virtual bounding rectangle br is used to define an objective function keeping all rectangles close together. All rectangles representing a set are enforced to be inside this encompassing bounding rectangle. Indirectly, all element groups are also positioned inside of br as they are included by constraint H1 in at least one rectangle. This bounding rectangle is later used to formulate an objective function to increase the Euler diagram's compactness.

H5 Name of Set Overlap

Names of different sets may overlap if rectangles are positioned close together. This overlap is undesirable, as it renders the names unreadable. Therefore, similar to the elements are positioned outside of sets in H2, overlapping set names are avoided. A bounding box encloses each set name. The same constraint definition as in H2 is used to prevent these bounding boxes from overlapping.

H6 Avoiding Empty Intersections

If sets do not share any elements, they should not intersect; respectively, those rectangles should not overlap. Constraint H3 only forces the sets' edges to keep a minimal distance. The ordering of the edges is not specified. Therefore, undesirable and empty overlaps of rectangles, as shown by the red shaded region in Fig. A1d, can be introduced with the reordering. These empty intersections are not present in the abstract description of the Euler diagram and hinder the diagram's readability. Similar to constraint H2, where an element group is forced outside of all sets it is not contained in, we can avoid undesirable intersections of sets that do not share any elements. We evaluate if two sets contain the same element. If not, we add the constraints described in H2 to keep the minimal distance d between rectangles. However, empty intersections between sets that do share elements cannot be prevented, as they are sometimes required to be able to visualize complex datasets.

H7 Subset Inclusion

If a set represented by r_1 is fully contained in an other set r_2, its rectangle should be enclosed by r_2. Thus we add constraints similar to those of H1. As all elements of r_1 are also in r_2, we can remove the constraints that enforce the element groups of r_1 to lie in r_2.

1.1. Choosing M

To choose the variable M in advance might be tricky, as it has to be big enough to satisfy the constraints [WG04, p. 490]. But too large values of M tend to introduce instabilities while solving the MIP. Thus the documentation of Gurobi[†] suggests choosing a value for M that is not larger than necessary. If an upper bound for M can be estimated in advance, it should be used [CCZ14, p. 67]. In our approach, the necessary size of M highly depends on the dataset. We, therefore, sum up the maximum width and height of each element group. Visually, this represents the positioning of all element groups side by side. As some labels of sets are considerably longer than the element groups, we also add the lengths of all set labels. To be sure that M is large enough, we double the described sum and obtain a rough upper bound for M.

[†] www.gurobi.com/documentation/9.5/refman/dealing_with_big_m_constra.html

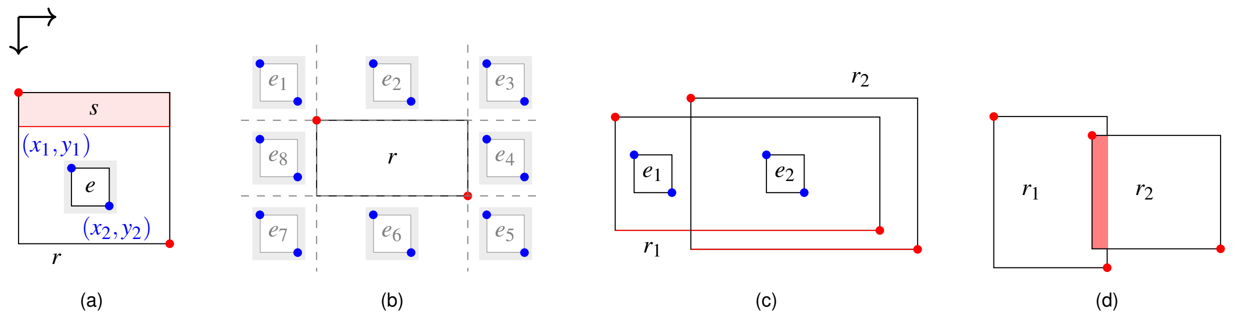


Figure A1: *Overview of constraints: (a) H1: Element e inside of set r (b) H2: Element e outside of set r (c) H3: Maintaining distances between set lines (d) H6: Avoiding empty intersection.*

2. Comparison of splitting strategies

Table A1: *Overview of split dataset properties, runtimes, and empty intersections statistics using the random splitting strategy.*

datasets	#sets	#elements	#zones	first result (s)	first objective	final result (s)	final objective	#sub layouts	#duplicated sets	%area empty	#empty intersections	time used for infeasible solutions (s)
DnD	8	359	108	358.5	69883.0	440.5	58171.0	4.0	8	48.6	140.0	332.9
IMDb	10	90	41	82.9	35823.5	135.4	27153.5	2.0	10	39.7	52.5	42.0
Animals	7	50	29	50.5	15997.0	122.5	12446.5	2.0	7	38.9	40.0	50.0
Marvel	28	24	24	88.9	45597.5	164.7	40627.0	2.0	19	81.8	149.5	50.3
Venn 5	31	5	31	7.1	9127.5	18.0	8143.0	2.0	5	31.4	20.5	6.8
Mosaic Nat. 8	13	178	33	51.0	23616.0	124.5	17276.0	2.0	11	23.8	29.0	50.2

3. Results

3.1. Simpsons Dataset

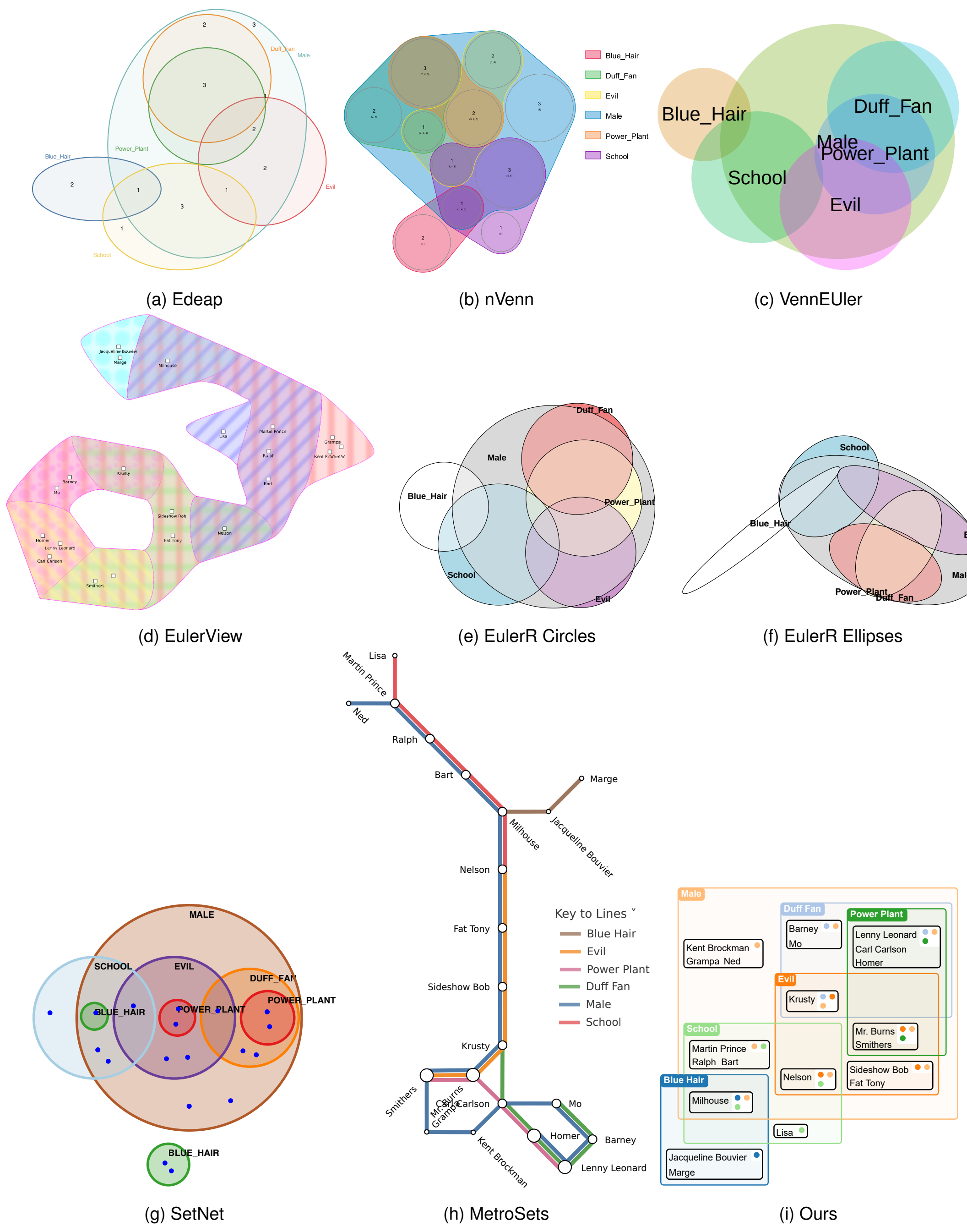


Figure A2: *Simpsons dataset. Almost all techniques using circles/ellipses infringe the well-formedness property (a, b, c, e, f), as some zones might be very small, some zones might not even exist, and some zones are disconnected. An exception to this is SetNet (g), as they duplicate the set curves for powerplant and blue hair. This way, they do not infringe any other well-formedness property. They all do not include individual data elements inside the diagram. EulerView (d) does include the elements, but contains a lot of concurrency. MetroSets also shows all data elements, but due to its layout, set intersections such as (Evil,Male) take up a lot of space in the visualization. Our result displays all set intersections of the abstract description. The only well-formed property that we cannot guarantee is disconnectivity of zones.*

3.2. Magic dataset

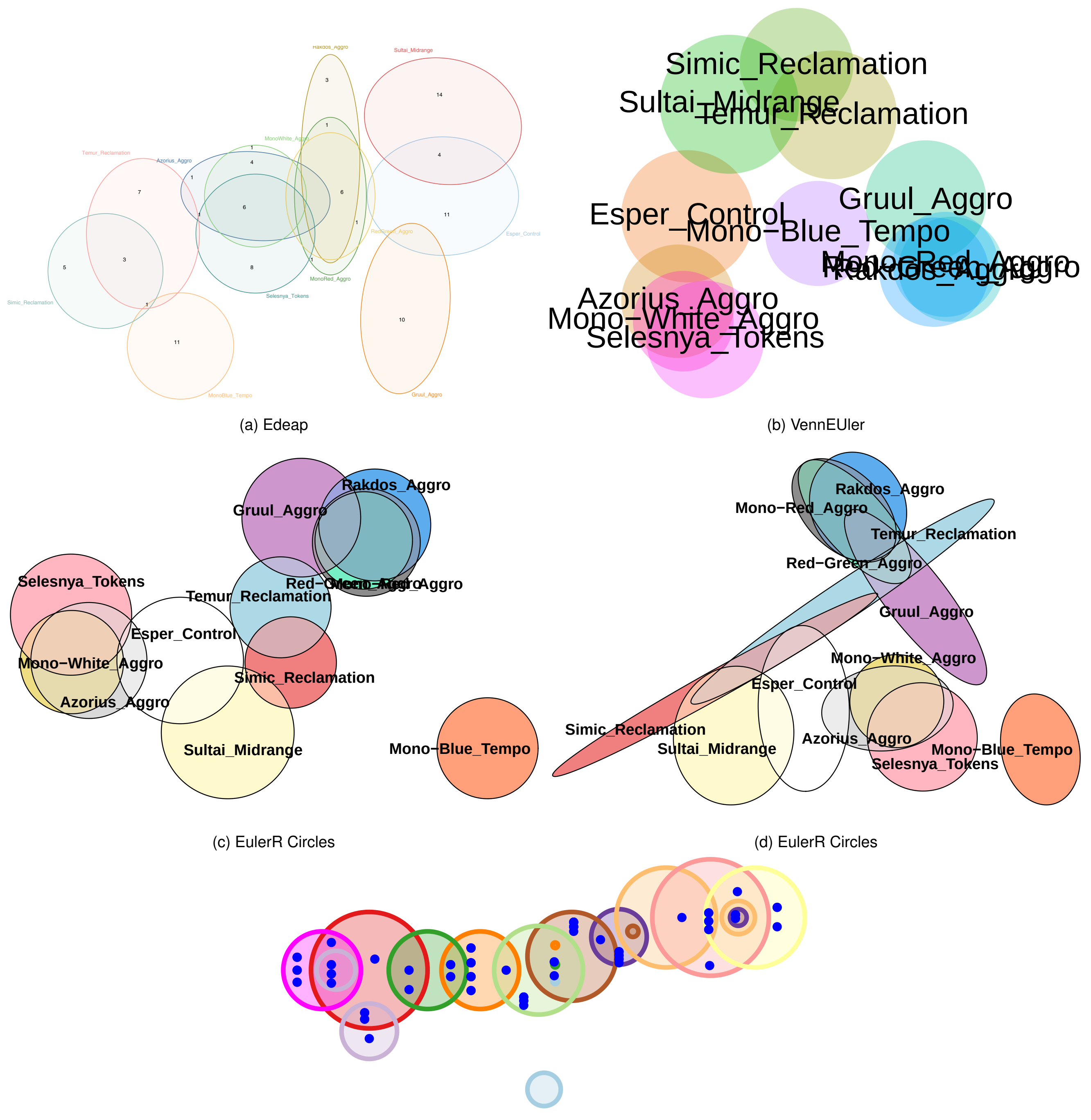
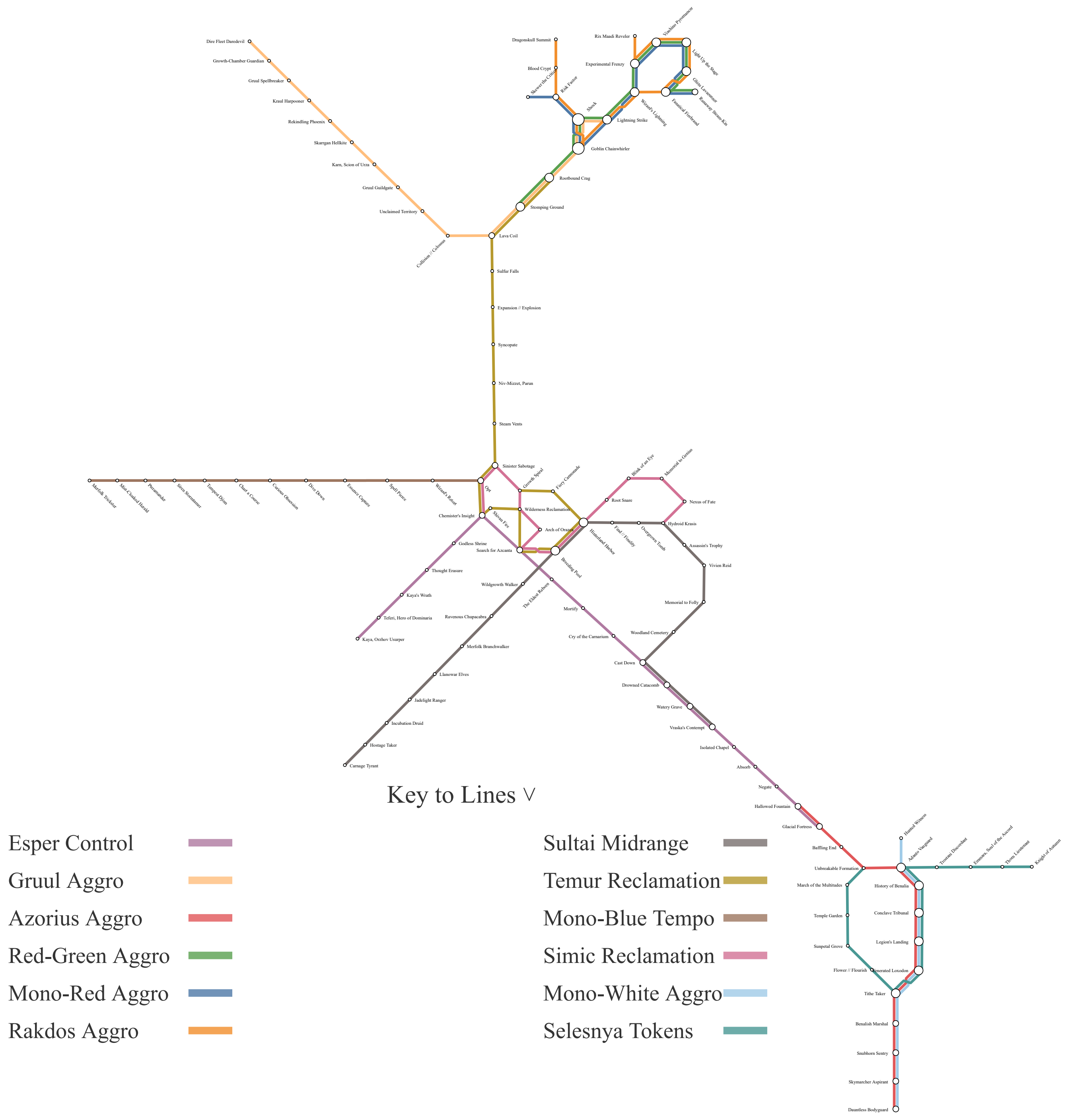
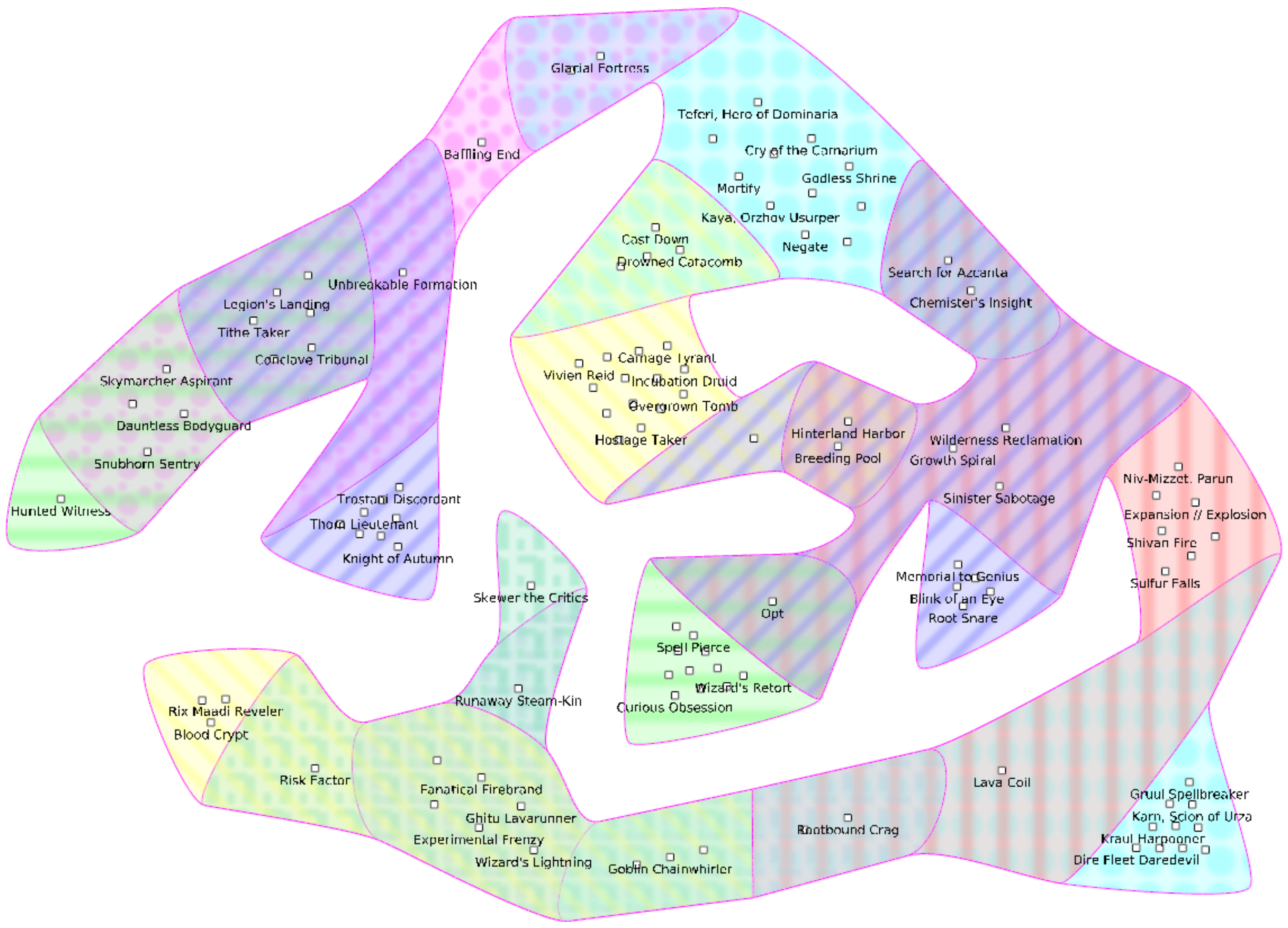


Figure A3: *Magic I. Magic is a complex dataset, as it contains many sets and data points. All methods using circles or ellipses struggle here, and often do not even produce all set intersections of the abstract description.*



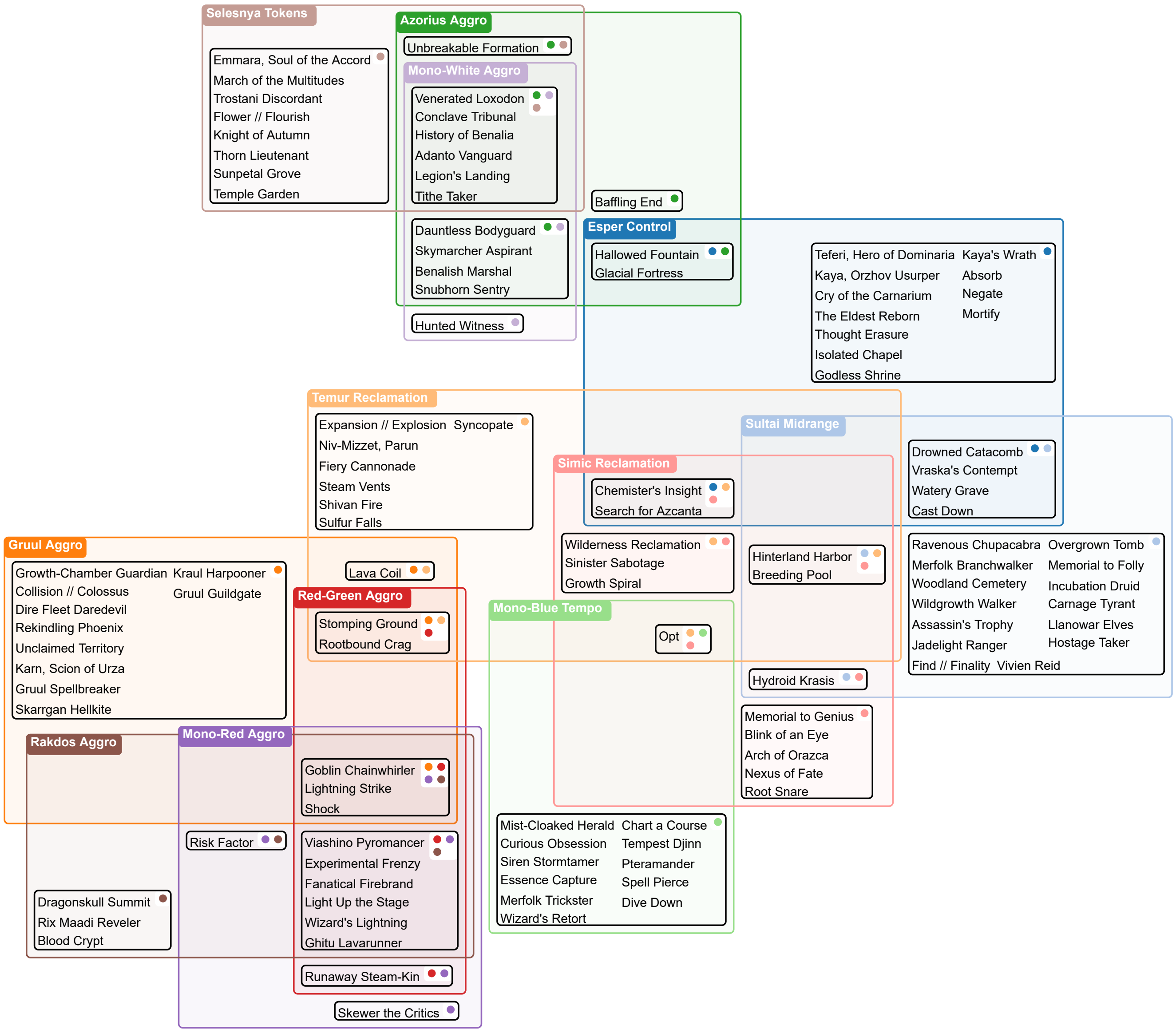
(a) MetroSets

Figure A4: *Magic II. MetroSets can visualize the set intersections of the dataset correctly. However it needs a lot of space to layout all the data points.*



(a) EulerView

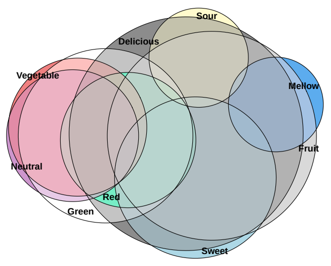
Figure A5: *Magic III. EulerView can visualize the set intersections of the dataset correctly. However, It it very hard to infer the depth of a set intersection, as the graph is packed very tightly. The outline of the set shapes is also somewhat complex.*



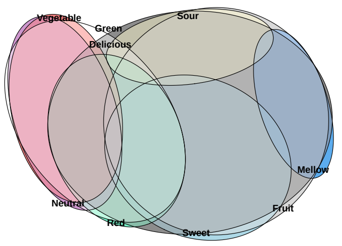
(a) Ours

Figure A6: *Magic III. Our method can visualize all set intersections of the dataset, although we do create some empty intersections, e.g. (Grul Aggro,Rakdos Aggro) in the top left. The layout is compact while still showing the structure a little bit, without having any direct concurrency of lines.*

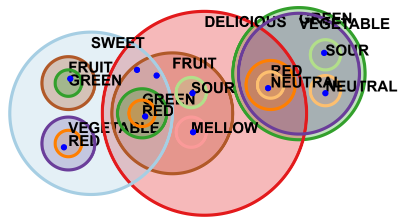
3.3. Fruit Dataset



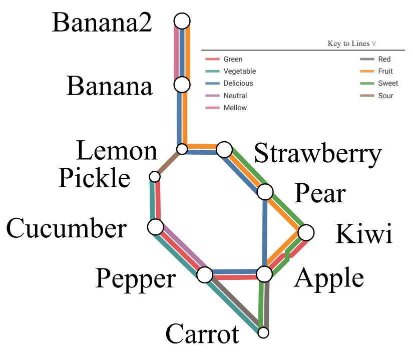
(a) EulerR Circles



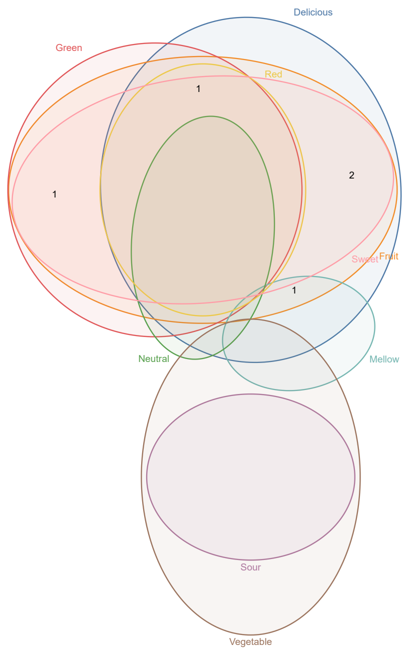
(b) EulerR Ellipses



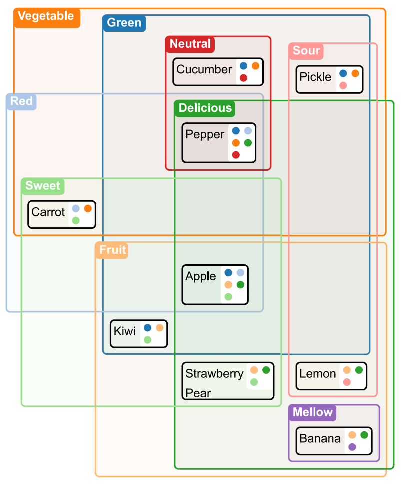
(c) SetNet



(d) MetroSets. Note that we manually added Banana2, as otherwise would not have produced a result (sets with one element cannot be drawn)



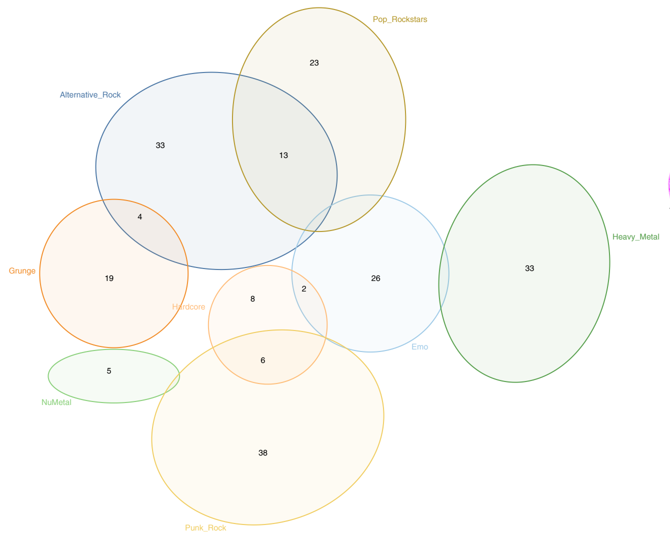
(e) Edeap



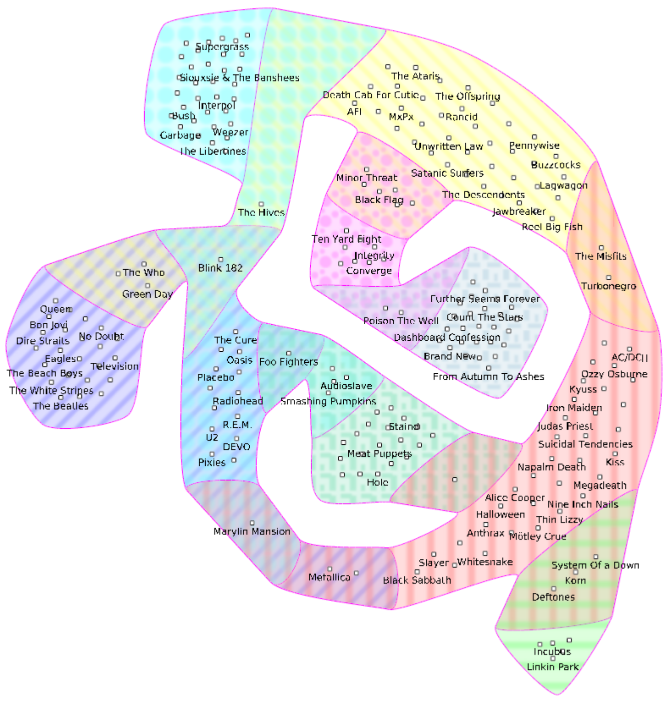
(f) Ours

Figure A7: *Fruit dataset*

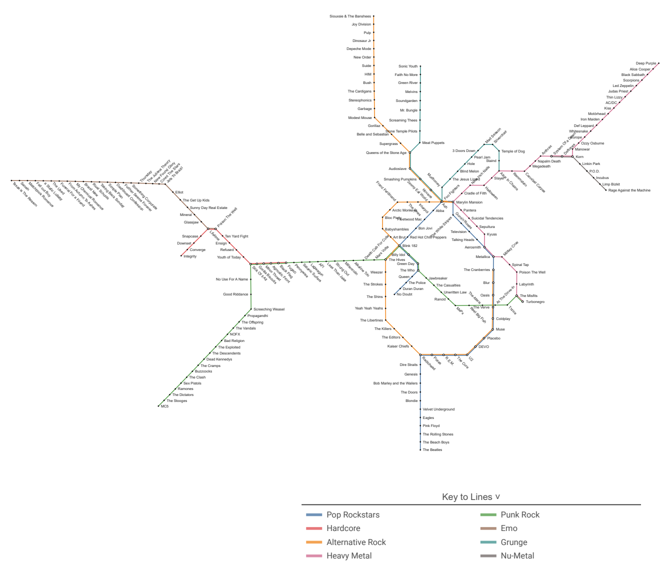
3.4. RocknRoll dataset



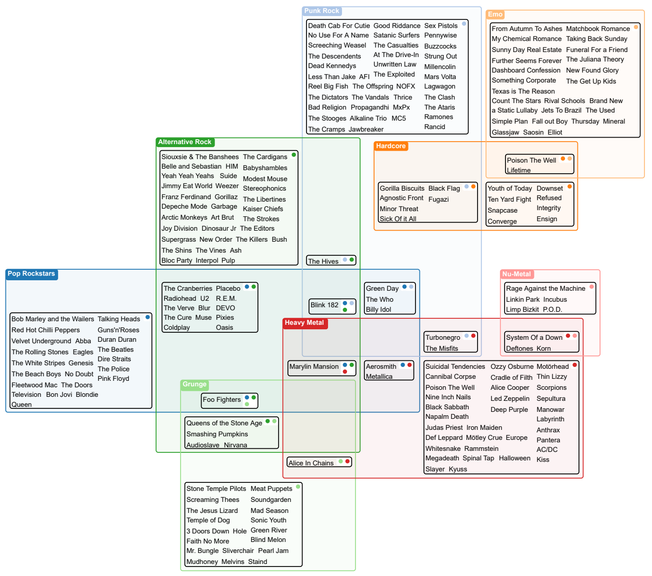
(a) Edeap



(b) EulerrView



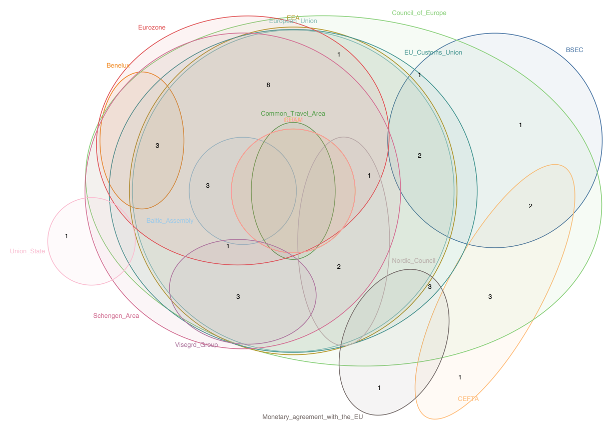
(c) MetroSets



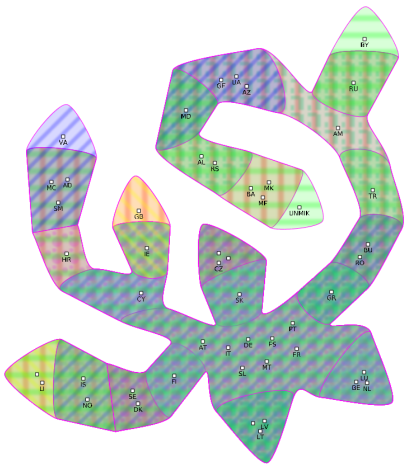
(d) Ours

Figure A8: *RocknRoll*

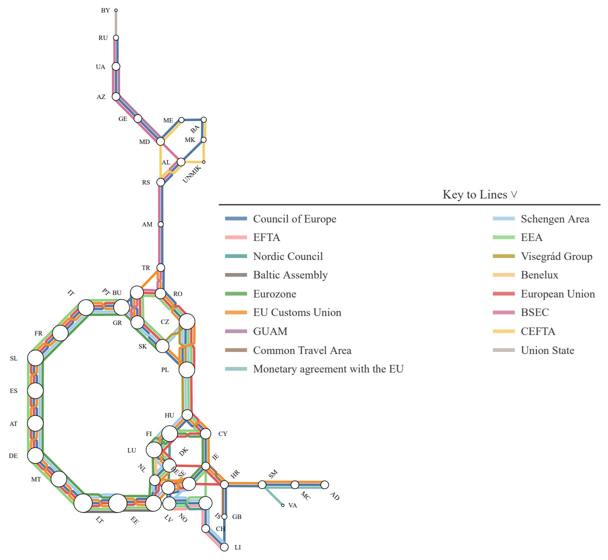
3.5. EU Dataset



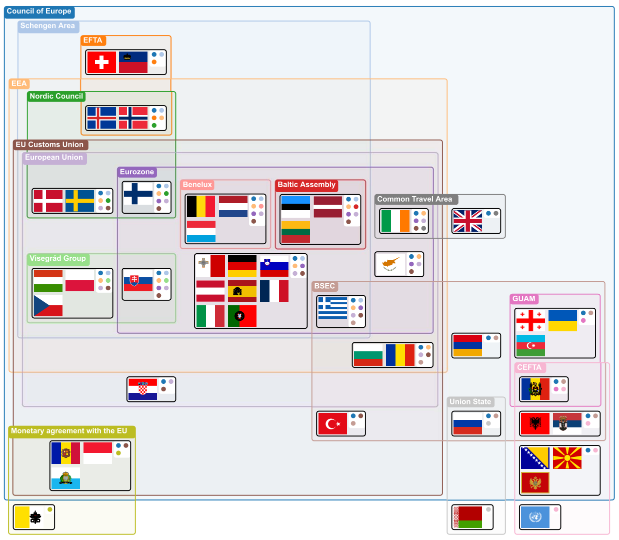
(a) Edeap



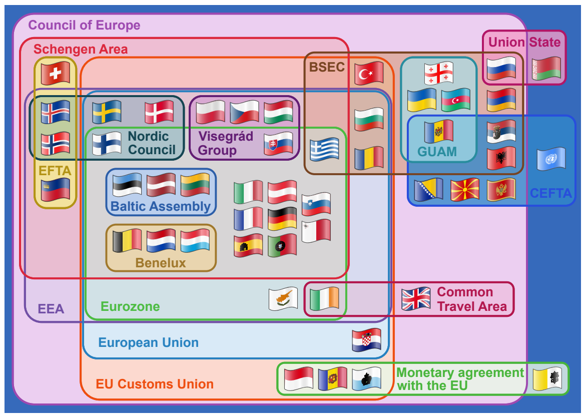
(b) EulerView



(c) MetroSets



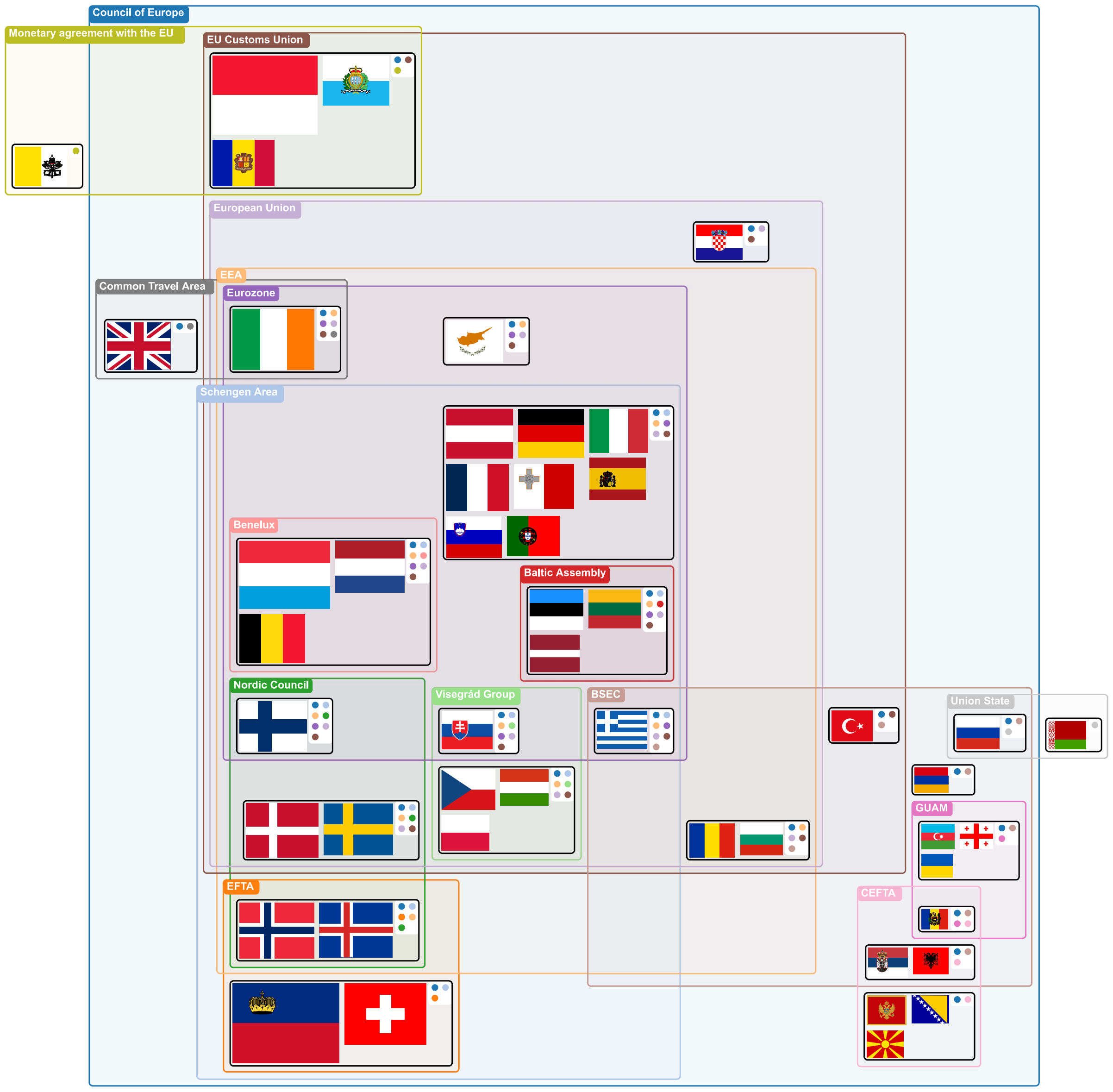
(d) Ours



(e) EU from WikiMedia

Figure A9: *EU dataset*

3.6. EU GDP



(a) EU

Figure A10: *EU data set with flags scaled according to the country's GDP.*

References

[CCZ14] Conforti M., Cornuéjols G., Zambelli G.: *Integer Programming*. Springer International Publishing, Cham, 2014. doi:10.1007/978-3-319-11008-0_9. 2

[Kal17] Kalvelagen E.: Rectangles: no-overlap constraints, 2017. [Online; accessed 11-February-2022]. URL: http://yetanothermathprogrammingconsultant.blogspot.com/2017/07/rectangles-no-overlap-constraints.html. 1

[WG04] Winston W. L., Goldberg J. B.: *Operations Research: Applications and algorithms*. Thomson Brooks/Cole Belmont, 2004. 2