Layout Metrics for Euler Diagrams

Jean Flower[1], Peter Rodgers[2], Paul Mutton[2]
[1]University of Brighton, UK
[2]University of Kent, UK

J.A.Flower@brighton.ac.uk, P.J.Rodgers@kent.ac.uk, pjm2@kent.ac.uk

Abstract

We present an aesthetics based method for drawing Euler diagrams. Aesthetic layout metrics have been found to be useful in graph drawing algorithms, which use metrics motivated by aesthetic principles that aid user understanding of diagrams. We have taken a similar approach to Euler diagram drawing, and have defined a set of suitable metrics to be used within a hill climbing multicriteria optimiser to produce "good" drawings. There are added difficulties when drawing Euler diagrams as they are made up of contours whose structural properties of intersection and containment must be preserved under any layout improvements. In this paper we describe our Java implementation of a pair of hill climbing variants to find good drawings, a set of metrics that measure aesthetics for good diagram layout, and issues concerning the choice of weightings for a useful combination of the metrics.

Keywords: *Euler diagrams, graph drawing, layout metrics.*

1: Introduction

Euler diagrams are used in the representation of a range of visual software engineering notations, including constraint diagrams and statechart-like diagrams. They are also a common method for explaining the basics of set theory.

Euler diagrams consist of a set of *contours*, drawn as simple closed curves. If contours meet, then they meet transversely (that is, they may cross but do not touch). At any crossing point, only two contours meet. The contours split the plane into *zones*, as shown in Figure 1. Each zone can be uniquely identified by the contours which contain it. This restriction means that some diagrams become invalid because they have "disconnected zones".

An *atomic* Euler diagram has the property that the contours form a connected subset of the plane. Here we introduce some layout metrics for smoothing atomic Euler diagrams, although the principles of the aesthetics and metrics work equally for non-atomic (*nested*) diagrams, see Figure 2.

An alternative term for these diagrams is "Euler-Venn diagrams" but they are often inaccurately called "Venn diagrams". Venn diagrams often look similar, but must contain all possible intersections of contours. In contrast, Euler diagrams contain any desired combination of intersections between the contours. Visualizations of Venn diagrams are often created by taking advantage of the symmetries present in a Venn structure [12].

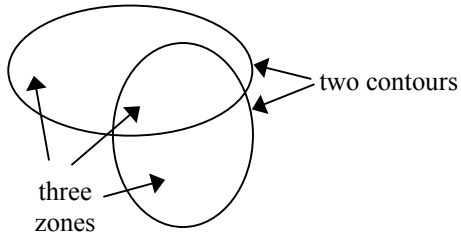


Figure 1

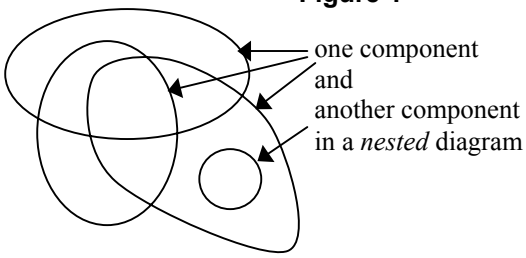


Figure 2

To our knowledge, the automated layout of Euler diagrams to aid understanding has not been investigated before. This paper begins the work needed to introduce aesthetics into the process of Euler diagram creation. To generate our visualizations we have implemented an aesthetics based Euler diagram drawing method which takes an initial Euler diagram and turns it into a more comprehensible representation with the same structure. The resulting diagram is more rounded, smoother and has better spacing, but contains the same contours as before. The areas into which the diagram is divided also remain the same so that if two contours do not meet in the initial diagram, then they will not cross in the final diagram.

In order to visualize Euler diagrams from an initial, poor layout we define a group of metrics that form our aesthetic criteria. These are given individual weights and combined, to produce a total score giving the goodness of a diagram layout. A hill climbing multicriteria optimizing system is then used to change the diagram layout in such a way as to reduce the total score of criteria, and so increase the comprehensibility of the diagram.

One motivation for this work comes from the wish to construct concrete representations of constraint diagrams [6] given an abstract description. These diagrams are used to represent logical statements, particularly as applied in object-oriented program design.

The work described here forms part of the "Reasoning with Diagrams" project, funded by the UK research council, the EPSRC. This project includes in its aims the creation of a software tool for the presentation and manipulation of constraint diagrams, alongside other notations, including UML notations [9]. The application of a reasoning rule will transform one diagram into another, and that transformation is defined only for the "abstract diagram" (which encapsulates the diagram's structural properties but forgets other geometric and topological properties). In order to see the results of an application of a reasoning rule, an algorithm is needed to take an abstract description of a diagram and produce a visual representation of it. Work on reasoning rules with diagrams can be found in [8].

Euler diagrams can also be used to represent statechart-like notations [7]. A further application of Euler diagrams is in teaching where they are widely used to represent the relationships between sets. For example the sets A and B, and their intersection $A \cap B$ could be represented by the diagram given in Figure 1. The work described here could be used in the development of educational software for such set theoretic concepts.

This paper builds on existing work in [4,5] which presented an algorithm for the creation of "correct" Euler diagrams from diagram descriptions. An abstract diagram description is used to create a dual graph. A plane embedding is found for the graph, and from that drawn dual graph, the contours are created using a step called circularisation. During this process, checks are made and a class of abstract diagrams are identified as undrawable. The algorithm is complete in that all undrawable examples are identified and shown to be undrawable, and all drawable examples are given concrete representations. However this work did not address issues of the comprehensibility of diagrams.

Aesthetics based diagram display methods have been previously applied to laying out graph diagrams. As with the work described here the results of the aesthetics for a particular layout are weighted and the total is applied within a multicriteria optimising system. The multicriteria systems used in graph drawing tend to make a much wider search of the problem space than the hill climbing method used here. In particular, simulated annealing [2] and genetic algorithms [11] have been used. Given the differences between graph diagrams and the Euler diagrams described in this paper, many of the aesthetics for graphs, such as edge crossings and subgraph symmetry, are not applicable to our work, however there are some aesthetics that are general across many diagrams, such as diagram total area and diagram aspect ratio that we have adapted for our use. See [1,10] for further discussions of aesthetic criteria for graphs.

Section 2 of this paper describes the iterative process used for manipulating diagrams for smoothing, and in Section 3 we introduce the various metrics used to guide the iteration. In Section 4 we reflect upon the results obtained from the implemented algorithm and consider suitable relative weightings to make optimum use of the metrics to achieve quick and aesthetic results.

2: The Optimising Process

Our Euler diagram drawing method works by attempting to minimize the total score for a group of aesthetic metrics. The total score is calculated as a weighted sum of the metrics, hence the task of diagram drawing forms a multicriteria optimising problem.

There are many techniques for solving such multicriteria optimising problems. We took a view that methods such as genetic algorithms and simulated annealing, which are designed to avoid local minima by providing a wide search of the problem space, were not appropriate to this application. This was firstly because, from our experience, this problem space appears to have a lower incidence of local minima than is found in many applications where wide search is applied and a cooling schedule deals with many of these, and secondly because these methods operate relatively slowly. As a result we decided to search using hill climbing, which only allows improvements to the solution, and so finds a solution in relatively quick time.

Contours are represented by polygons. These polygons are stored as lists of points. A diagram will be changed in two ways: altering the shape of contours by moving individual points; or altering the arrangement of contours by moving all the points that make up a polygon.

Although a complete contour could be moved by all its points moving individually, complete contours are moved in each iteration for two reasons. Firstly, achieving a shift of a contour by moving individual points is slower than moving the whole contour as a single iterative step. Secondly, if only single point steps are allowed, the whole contour may never actually be moved, despite the change of position improving the total score, because the movement of a single point of the contour in that direction may reduce the score and so, with a hill climbing strategy, the individual movement will never be made.

The diagram drawing process involves iterating through a hill climber a set number of times. We implemented two variants of hill climbers, called: "RandomHillClimber" and "FastHillClimber". A single iteration of either hill climber involves going through all the contours in the diagram and testing each point of a contour for a possible improvement, as well as testing whole contour shifts for possible improvements. To find an improvement a move of either a point or contour is made and the total score for the diagram before the move is

compared to the total score for the diagram after the move. If the score is better or the same, then the move is retained, otherwise if the score is worse, then the point or contour moves back to its original position.

The retention of the diagram structure is a key requirement when moving points or contours. The diagram structure represents the "meaning" of the diagram - the information is exactly which zones are present and which are absent. At each iterative step, the new diagram is taken and the set of *abstract zones* is calculated. An abstract zone is a set of contour labels listing the contours which contain the zone. If a zone is in contour A and contour B and no others then its abstract zone is the set $\{A,B\}$. A move is only allowed if the new set of abstract zones matches the initial set.

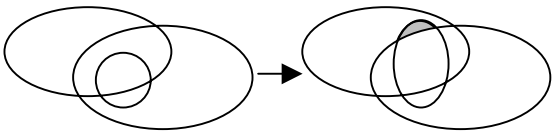


Figure 3

Figure 3 illustrates a diagram manipulation which would be forbidden by the structure checker. A new abstract zone has appeared in the resulting diagram.

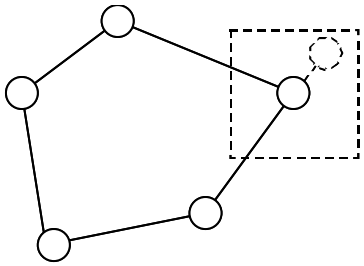


Figure 4

The RandomHillClimber moves a point or contour to a random location within a square area of the starting point, as shown in Figure 4. We use a square area because it is easy to implement, but has a small amount of distortion in deciding the direction because diagonals are slightly favoured. This distortion could be avoided by finding a random location in a circular area about the original location.

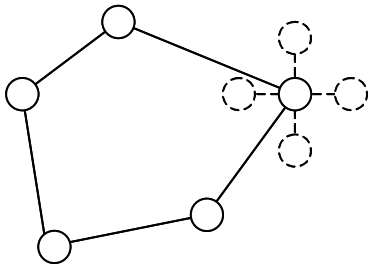


Figure 5

The FastHillClimber moves each point and contour in one of four directions: up, down, left and right, all at the same distance (see Figure 5). For a move, each of the four directions is tried until a step is found to improve the diagram, in which case the move is retained and the remaining directions are not tested.

From initial investigations it was seen that the FastHillClimber generally reached a good diagram in fewer iterations than the RandomHillClimber. This is probably because, although FastHillClimber goes through many more tests on a single iteration, the systematic testing of a variety of directions ensures that the uphill movement for a point or contour is usually found. However for the RandomHillClimber, a point may have to wait many iterations before the random movement direction is an improvement.

Both hill climbers also have an option to include a cooling schedule. The cooling works by reducing the movement as the iterations progress. For the first iteration the movement is a value specified by the user. For later iterations the movement reduces linearly, until it is nearly zero for the last iteration. The intended effect of this is to move points and contours quite far at the start to quickly produce rough versions of good diagrams. As the number of iterations increase the diagram is fine tuned with the reduced movement allowing layout to be refined within a small range.

The cooling option proved successful when it was tested, as without it the movement distance is constant, and so has to be set to a fairly small number to allow the diagram to be refined. With a small movement distance, moving a contour a long way or changing its shape greatly takes many steps. Fewer steps are needed if a large intial movement is chosen and cooling applied. The cooling schedule may also help avoid local minima, as initial large jumps of contours past troughs are possible, which cannot be passed by the necessarily small movements of the uncooled option.

3: Metrics

Metrics were used to assess the quality of a diagram. Each metric gives a result which is a positive number, where zero represents the best score, and larger numbers reflect lower quality.

A set of metrics have been implemented to reflect different aesthetic requirements. These metrics are used in conjunction by creating a total score built as a weighted sum of the metric scores. The iterative process described in Section 2 uses this combined metric to assess whether or not a change in the diagram is an improvement.

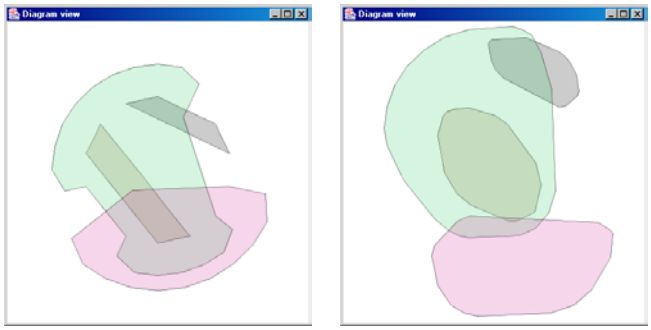
In this section we describe the range of metrics used in the smoothing algorithm, and describe the effects of each metric. The metrics are motivated by a sense of diagram aesthetics. The first two metrics are driven by consideration of the aesthetics of a single contour. A single contour is ideally presented as a "round" and "smooth" curve. Since contours are drawn as polygons, an ideally "round" contour corresponds to a regular polygon. Two metrics can be used to give a quality judgement against this aesthetic.

The *ContourRoundnessAngles* metric measures the variance of the angles within the contours. Let $n(c)$ be the

number of points on contour c, $\overline{\alpha}(c)$ be the mean angle in c and $\alpha_i(c)$ be the i^{th} angle in c, then the metric is given by

$$\sum_{contour\ c}\left(\sum_{i=1}^{n(c)}\frac{(\alpha_i(c)-\overline{\alpha}(c))^2}{n(c)}\right)$$

The metric is invariant under diagram scaling, and takes into account the number of points a contour has. The metric returns zero for a regular polygon. Figure 6 illustrates the effect of this metric in one example. The initial diagram is produced using an algorithm for the creation of "correct" Euler diagrams from diagram descriptions [4,5]. As with all the diagrams in this section, the FastHillClimber with cooling for 30 iterations has been used.



Initial diagram | Roundness using angles

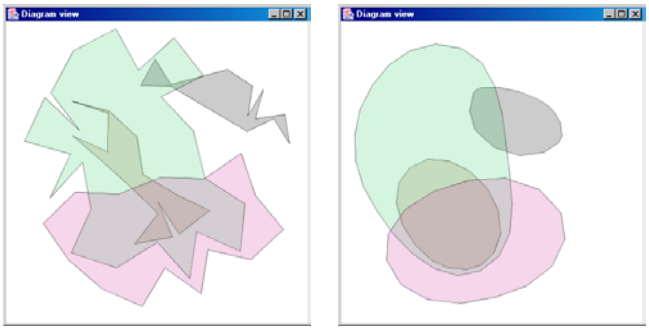
Figure 6

Although the use of one metric has smoothed the contours to a certain extent, we can see that using ContourRoundnessAngles alone is not sufficient to produce iteration towards a regular polygon. Polygons which have (near-)equal angles may still fail the roundness aesthetic, due to the presence of imbalanced edge lengths. A second metric, *ContourRoundnessEdgeLength* is derived from the variance of the edge lengths of contours. Here, d is the diagram, $\bar{l}(c)$ is the mean edge length in contour c, $l(e)$ is the length of edge e in contour c, and $n(c)$ is the number of edges (or vertices) in c.

$$\left(\sum_{contour\ c}\left(\sum_{e\ in\ c}\frac{(l(e)-\bar{l}(c))^2}{n(c)}\right)\right)\Bigg/\left(\sum_{e\ in\ d}l(e)\right)^2$$

This metric gives zero for regular polygons. Again, the metric remains unchanged after scaling because of the division by the square of the sum of edge-lengths.

Fig. 7 uses the same example as figure 6 and illustrates the effects of applying the edge length metric alone and the effects of applying a combination of ContourRoundness-Angles and ContourRoundnessEdgeLength.



Roundness using edges | Roundness using angle and edge approaches combined

Figure 7

Even if a diagram is drawn with “nice” contours, there are other aesthetic qualities we require for good readability. Contours should be of comparable size, otherwise readers may draw misleading conclusions from the different contour sizes. Zones should remain clearly visible. All the diagrams have so far been for the same example, but in the smoothest presentation one zone is disproportionally small. The aesthetics concerned with areas are addressed by the following metrics.

The *ContourArea* metric measures the variance of contour areas, and factors by the diagram area to ensure the metric is dimensionless. Let $\overline{area}$ be the mean area within a contour of the diagram, $area(c)$ the area within contour c, and $n(d)$ be the number of contours in the diagram d, then the metric is

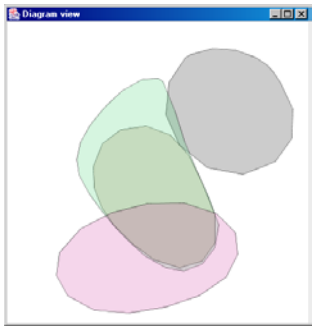
$$\sum_{contour\ c}\frac{(area(c)-\overline{area})^2}{n(d)}\Bigg/\left(\sum_{contour\ c}area(c)\right)^2$$

The *ZoneArea* metric calculation considers each zone area as a proportion of the combined area. In this formula, $area(z)$ is the area of zone z.

$$\left(\sum_{zone\ z}\frac{1}{area(z)}\right)\times\sum_{zone\ z}area(z)$$

The effect of this metric in iteration is to discourage the appearance of disproportionally small zones. ContourArea and ZoneArea are invariant under scaling.

Figure 8 shows output created using roundness metrics in both cases, with ContourArea metric on the left and both area metrics on the right.


After applying roundness and contour area metrics

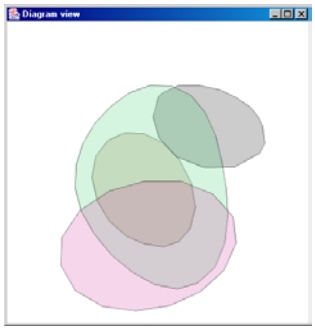

Also including the zone area metric

Figure 8

The aesthetics we have addressed so far require diagrams to have round contours, with the areas of contours, zones and the entire diagram controlled.

The last set of aesthetics to add are concerned with the closeness of contour arcs. The gap between two parts of contours which do not cross should be wide enough that it remains clearly visible. We also want to ensure that where contours cross, the angle at the crossing point is sufficiently great to clearly distinguish the contours. These final aesthetics are addressed by the following two metrics.

The *ContourClosenessPts* metric measures point-to-point between distinct contours, summing the reciprocal of the distances. The effect of this is to encourage points to move apart. Where contours cross, it is inevitable that points on different contours become close, and for this reason, points close to contour crossings are excluded from the summation (this is what's meant by "*some* $v1$ *on* $c1$…" in the formula below). We also divide by the number of points on each contour to ensure that the same effect is achieved, independent of the number of points on a contour. Here $\|v1 - v2\|$ is the distance between vertices $v1$ and $v2$.

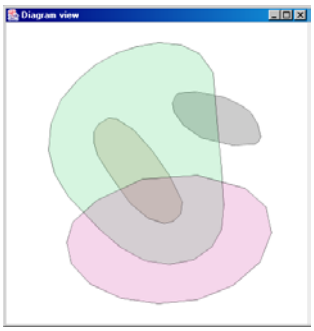
$$\sum_{\substack{\text{contours } c1 \neq c2 \\ \text{some } v1 \text{ on } c1, \\ v2 \text{ on } c2}} \frac{1}{\|v1 - v2\|^2 \times n(c1) \times n(c2)} \times \sum_{\substack{\text{contours } c1 \neq c2 \\ \text{some } v1 \text{ on } c1, \\ v2 \text{ on } c2}} \|v1 - v2\|^2$$

This metric works well for parts of curves where there are many points. If, however, a contour has a long edge (despite the ContourRoundnessEdgeLength metric), then we also need to move points away from that edge.

Some vector algebra is used to determine which points are "relevant" to which edge. To assess whether a point is close to an edge, it is necessary to calculate the perpendicular distance from the point to the edge. To determine whether or not the perpendicular distance should contribute to the metric calculation, project the point onto the line along the edge, so that point is relevant to an edge if its projection lies on the edge itself.

This second contour closeness metric also incorporates the number of points on a contour and the sum of the distances involved, to ensure that it is not biased towards contours with many points, and it is invariant under scaling.

The two contour closeness metrics combine to effectively push contours away from each other and encourage crossings whose angles are closer to 90°. Figure 9 shows the effects of these contour closeness metrics.


Roundness and closeness metrics without area metrics

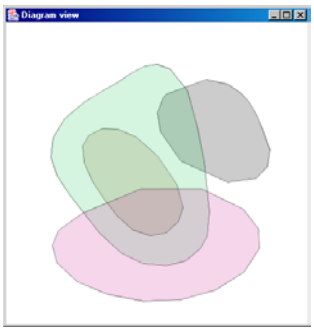

Combination of roundness, area and closeness

Figure 9

The metrics described so far are:

(i) ContourRoundnessAngles
(ii) ContourRoundnessEdgeLength
(iii) ContourArea
(iv) ZoneArea
(v) ContourClosenessPts
(vi) ContourClosenessEdgePt

Other metrics were implemented:

- The DiagramAspectRatio metric can be used to force a diagram to fit into a given space as a subdiagram.
- The DiagramArea metric simply measures the area of the bounding box of the diagram. The effect of this is to encourage diagrams to become smaller.
- ZoneRoundness metrics are similar to metrics about contour roundness but applied to zones.
- The ZoneConvexness metric employed a simpler approach to convexity. Traverse around zone boundaries and determine whether the angles are turning "left" or "right". The metric gives a low score where the orientations are consistent.

The metrics each have the desired effect when applied in isolation. If a sequence of diagrams is considered which range from "bad" to "good" then the metrics give monotonic results - better diagrams result in lower penalties. Such monotonic behaviour can be derived from a range of different algebraic expressions. For example, in the ContourRoundnessAngles metric, using the absolute value instead of the square would give the same ranking of preference between a range of diagrams.

$$\frac{(\alpha_i(c) - \overline{\alpha}(c))^2}{n(c)} \quad \text{vs} \quad \frac{|\alpha_i(c) - \overline{\alpha}(c)|}{n(c)}$$

We chose to use the first expression because it gives a stronger response for values which stray far from the mean value.

The metrics are applied as a weighted combination. We have achieved one kind of independence between the metrics: if a diagram is scaled, the only metric which is affected is the DiagramArea metric.

For almost all other diagram manipulations, however, the metrics are dependant. When combined in a weighted sum, the metric effects interact. In some examples, the metrics combine to encourage similar changes (positive reinforcement). For example, ContourClosenessPts and ContourClosenessEdgePt collaborate well together. In other examples the metrics encourage contradictory behaviour (negative reinforcement), for example, where one contour is contained within another, the ContourArea and ZoneArea metrics conflict with each other. Balancing these effects is a question of finding a suitable set of weightings to apply and this problem is discussed in the next section.

4: Results

In this section we present the results of running our system on various diagrams. We also include discussions about weighting the criteria, and some observations about how the process proceeds.

As well as using the iterative procedure, we do a final post processing step on the well drawn diagrams to make them more aesthetically pleasing. We add a feature to our display that replaces each straight line segment between two points in a contour with a Bezier curve. The Bezier control points are arranged such that the curve entering a point on the contour is continuous with the curve exiting the same point. When moving points in hill climbing, it is important to check that a curved diagram has the same structure as the original polygon diagram, as it is possible for the curving process to introduce or remove contour intersections. We do this by discovering the maximum area that the Bezier curves can occupy and ensure that when the polygons are extended with these areas they have the same structure, otherwise the drawing cannot be displayed with these curves. An alternative approach to smoothing contours is to use spline drawing algorithms, but this has added complexity when checking diagram structure.

The systematic discovery of a good set of weights for a multicriteria optimisation system is notoriously difficult. In this system, weights serve two functions. The first is to normalize the criteria, as the numerical output can vary greatly. The second is to weight the criteria to indicate their importance in the final diagram so, for example, if contour roundness is more important than zone area equality, a larger number can be assigned to the appropriate weight.

Weight allocation is further complicated as there are interactions between metrics, both positive and negative. Where positive reinforcement occurs (i.e. improvement in one also implies an improvement in the other), such as between the two contour closeness metrics, then the two measures should be seen as combined in some way. In the case of ContourClosenessEdgePt and ContourClosenessPts, which have very coordinated interaction, we regard the overall closeness score as a sum of the two separate metrics. Where negative interaction is present, as is most common between metrics, there is a notion of trade off, as score improvement in one metric is likely to reduce the score awarded by the other metric.

Our technique for normalizing the criteria was to survey the output of metrics from many runs of several diagrams. Although averages could be calculated and used for some diagrams, they were not universally applicable, and so intelligent adjustment of weights was still required.

For weighting by importance, our approach was to start with what we regarded as the single most important metric and apply a fixed weighting. A second metric can then be added and its weighting can easily be adjusted until a desired balance between the two criteria is achieved. Each additional metric is added one by one and its weighting is altered in a similar manner. Weightings in the final set may then be altered slightly to change the overall importance of each metric. Figure 10 shows the interface to the experimental software that was produced to investigate the settings for the criteria, as well as experiment with hill climbers and cooling schedules. The settings for the diagrams in this section are shown. A tick to the left of the metric indicates that it was applied; metrics with no tick were not used. The numbers to the right show the weighting for each metric. Cooling was applied and the FastHillClimber was run for 80 iterations.

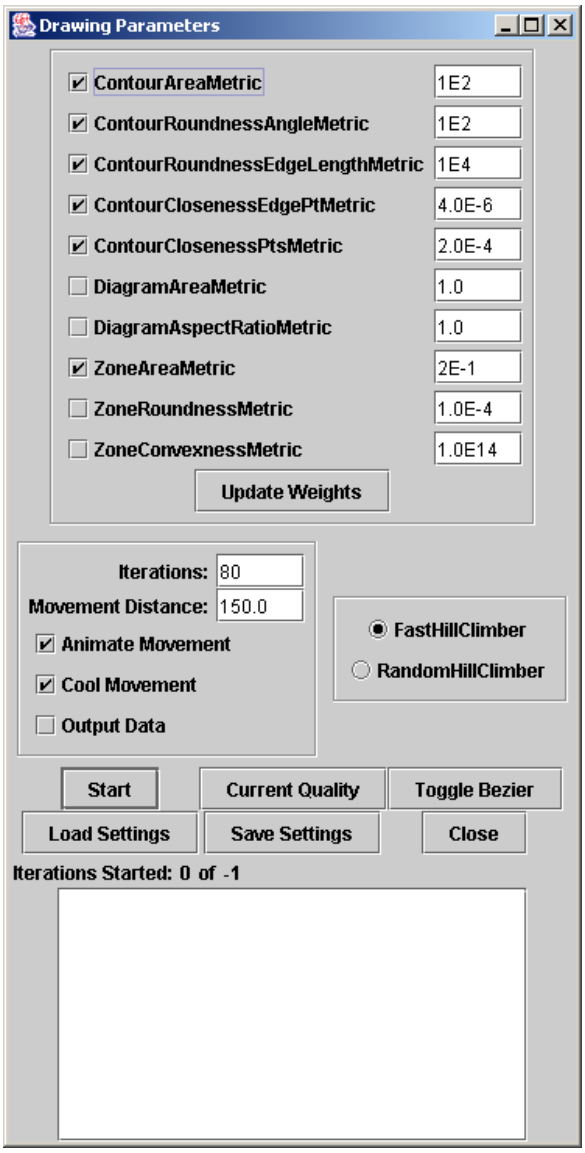
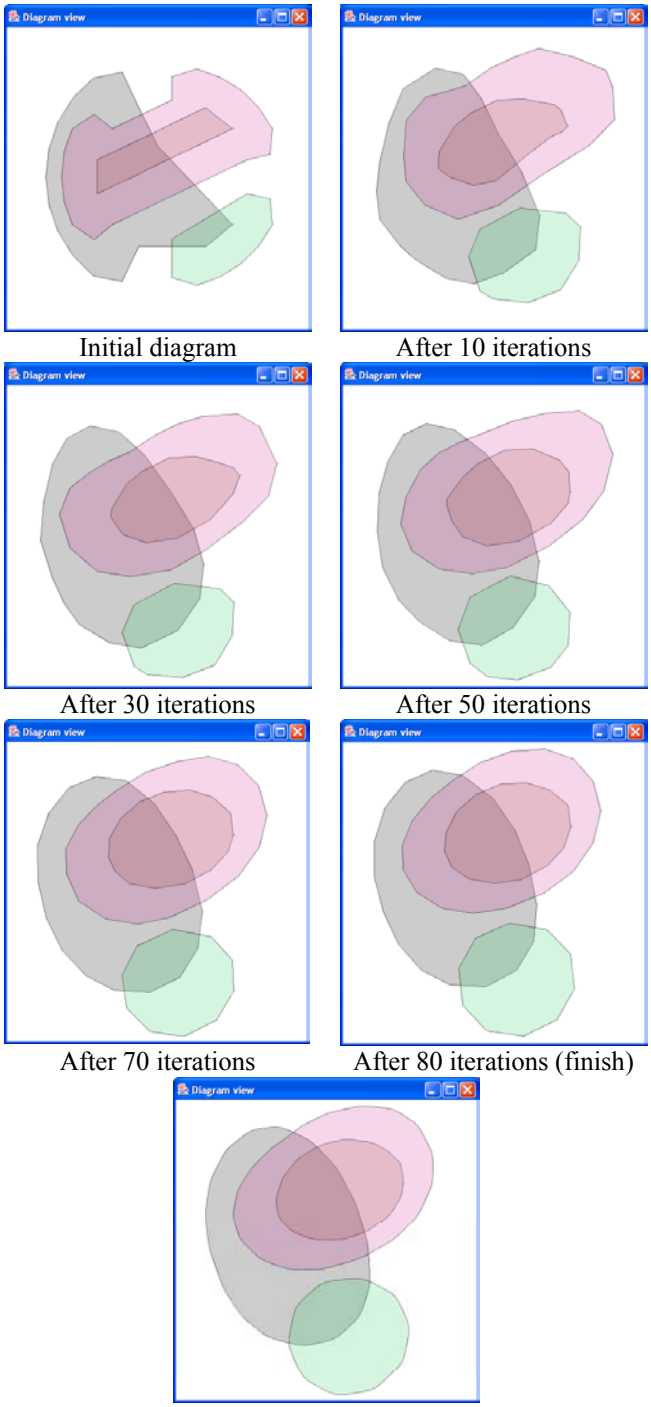


Figure 10

The variation in numbers mainly reflects the need to normalize the output of the metrics. However there is also a strong weighting for ContourRoundness and Contour-RoundnessEdgeLength, so that in a typical diagram before drawing, each of these measures might be 10 times greater than, for example, ZoneArea. This reflects our view that roundness is the primary positive aesthetic in the diagrams. The other criteria are broadly speaking given around equal weighting. It should be noted that all these comparisons vary by diagram as well as during the drawing process.



Initial diagram | After 10 iterations

After 30 iterations | After 50 iterations

After 70 iterations | After 80 iterations (finish)

Finished diagram with Bezier curves

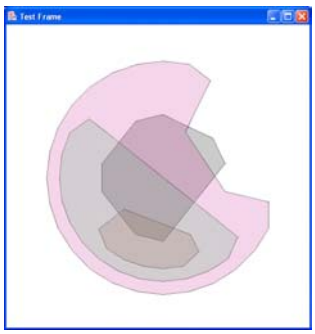
Figure 11

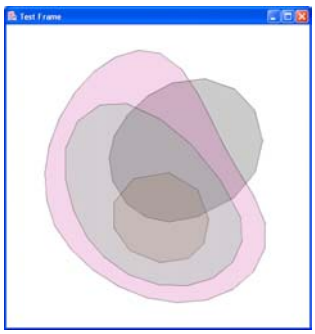
Iteration	0	10	30	50	70	80
ContourRoundness Angle	111.10	18.70	12.40	5.70	2.90	2.20
ContourRoundness EdgeLength	85.50	14.70	11.20	7.80	3.50	3.00
ContourCloseness EdgePt	3.50	3.00	2.80	2.80	3.00	2.80
Contour ClosenessPts	4.60	5.00	5.10	5.60	5.50	5.40
ZoneArea	17.40	12.20	8.90	11.20	11.20	11.20
ContourArea	9.60	6.70	6.70	5.90	5.50	4.90
Total Score	**231.90**	**60.40**	**47.10**	**39.00**	**31.60**	**29.60**

Figure 12

To give an idea of how the drawing progresses and how the criteria metrics alter, Figure 11 shows some selected stages during the drawing of a diagram. The scores for the relevant stages are given in Figure 12. Here the values for each metric are shown after they have been weighted by the multipliers shown in Figure 10. In these diagrams the closeness metrics, even when added together, give a relatively low result. This is due to the natural separation of the contours in this particular diagram. For Figure 12 it can be seen that the total score falls rapidly in the first 10 iterations, from 213.9 to 60.4, as easy improvements are discovered, and the effect of large moves are applied. This is reflected in the top two diagrams in Figure 11, where a great difference in the pictures can be observed. The later progress is much more gradual, where only a slight improvement in the roundness measures, a difference of 2.0, between iterations 70 and 80 is observed. The relevant diagrams in Figure 11 show two diagrams that are very similar, however they are much more refined than previous diagrams. We conjecture that the small difference in the two diagrams for 70 and 80 iterations is both because of the reduction in movement due to cooling and because the diagrams are close to an optimum layout.

The relative values of the individual metrics at the end of the optimisation process are markedly different from their starting values. In particular, ZoneArea is much higher than the rest. It started with a value of 17.4 and finished with a value of 11.2, and so is making a large contribution to the final total score. This is particularly interesting considering other metrics started out much higher, for instance, ContourRoundnessAngle has fallen from 111.1 to 2.2. It is difficult to understand why some metrics may reduce less than others, but the interactions between metrics could be one factor, so that there may be strong negative factors between ZoneArea and the other metrics, so when it reduces in value all of the other five metrics may increase in value.


Initial diagram


Finish position with polygons

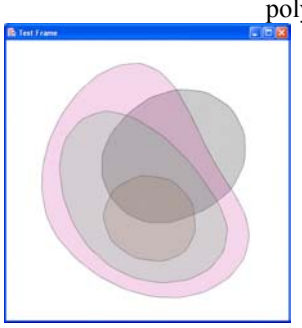
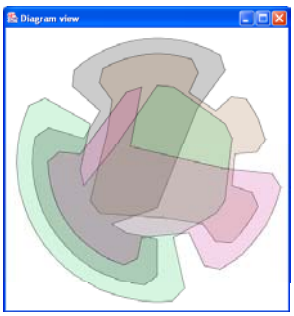

Finish position with Bezier curves

Figure 13

Figure 13 shows an Euler diagram, both before and after smoothing, and the final drawing with Bezier curves applied. Compared to the values in Figure 12, the starting metric values are relatively high for the contour closeness metrics: ContourClosenessEdgePt gives 11.56 and ContourClosenessPts gives 11.92, so that the combined total is 23.48, which is greater than the value of ZoneArea at 18.11. In this case, their combined total is also greater than the ContourArea outcome of 12.60. The initial value of ContourRoundnessAngles is 92.88 and ContourRoundnessEdgeLength is 43.12. The total score is 190.18. So although there is a great difference in the relative initial values of the metrics for the diagrams in Figures 11 and 12, the result in both cases is a rounded, aesthetically pleasing diagram.


Initial Diagram

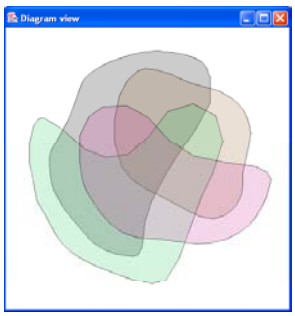

Final Position - Polygons

Figure 14: A Venn diagram with 4 contours

Figure 14 shows the before and after diagrams for a Venn diagram with four contours. A typical human based drawing for this diagram is to take the Venn diagram with three contours and add a banana shaped contour to it, resulting in three very rounded contours and one misshapen one. Our optimisation method has compromised the drawing of all four contours, with elements of roundedness, as well as slight negative curves in most of the contours.

5: Conclusions and Further Work

We have been able to show that by defining a suitable set of metrics it is possible to automate the enhancement of the layout of Euler diagrams. Our choice of metrics was motivated by a set of aesthetics, and the resulting diagrams measure well against these aesthetics.

Future work to extend this approach could include more metrics, if new aesthetic criteria can be determined. Existing metrics could be tuned by non-linear scaling functions, motivated by studies of how the metrics interact with each other in a large range of examples.

The cooling of iteration was an essential step to help avoid locally good but globally bad results. Without cooling, diagrams would converge to a layout which can't be improved using the same step size, but could be improved with a smaller step size. The cooling used was a linear reduction in step size over the iteration time. There is potential for investigating different kinds of cooling, with non-linear alterations in step size.

One method of allocating weights for the criteria could be adapted from genetic algorithms, which often define a scale and range for each weight to ensure that they are all within a certain range [11]. However, this relies on developing a large population of varied attempted solutions. It might be possible to generate a test set and method for randomising diagrams which could be used within such a automatic weighting system. An alternative approach to weight allocation is dynamic, using the outcome from each metric in previous iterations. For example, if one metric is reducing in very large steps, it might be sensible to decrease its weighting to give the other metrics some influence over the diagram changes.

Cooling seems to avoid many local minima during the improvement of diagrams in the current test set. Further example diagrams or the introduction of new aesthetics may require the development of a more sophisticated multicriteria optimisation method such as simulated annealing or a genetic algorithm. Simulated annealing introduces random negative movements and a genetic algorithm would generate populations of diagrams, merging and mutating selected members to produce later generations. Research questions for these methods include representing members of the population for a genetic algorithm approach, which are usually represented in a linear string based form, and investigating the trade off of wide searches against performance.

From looking at human based drawing of Euler diagrams its clear that there is a tendency to draw some contours very well and fit the rest in afterwards. This may be a limitation of the way they are drawn, or it may be aesthetic preference. If it is because of the latter, an automatic Euler diagram drawing mechanism that mimicked this approach has the potential to produce good diagram visualizations quickly by taking drawings of preprepared patterns of common Euler diagrams and

incrementally fitting in new contours based on user defined rules of drawing.

Variations on well known graph drawing algorithms may also be used to generate drawing of Euler diagrams. For instance, with the contours represented as a cyclic graph, force directed techniques could easily draw rounded contours. The challenge is to a develop a force model that effectively deals with interactions between contours and so produce good zones. An alternative force approach, which might be combined with other methods, would be to overlay the diagram with a second graph, the planar dual of the diagram, and apply a force model upon it. This should distribute the zones evenly.

The work described here is in the context of *atomic* Euler diagrams. The same aesthetics would hold in the context of *nested* Euler diagrams, and most metrics would work in this extended context. The metrics which refer to zones of a diagram would have to be reconsidered - in a nested diagram, a zone may have internal as well as other bordering contours.

It would be more efficient to smooth atomic components of a nested diagram, then reassemble the whole, but then accommodation must be provided in zones which will contain a subdiagram. The ZoneArea metric can be easily adapted to allow different weightings for different zones - guiding some zones to increase in size more than others. If a zone is to contain a subdiagram, it would be useful to include a ZoneRoundness or ZoneConvexness metric. Again, these metrics can be weighted to affect some zones more than others. Moreover, the subdiagram must be made to fit within the zone, and here a rectangle needs to be found in the zone, and a strongly weighted aspect ratio metric applied to the subdiagram.

Future implementations of the smoothing work should allow for nested diagram examples, and this will increase the number of suitable examples against which we can assess the smoothing approach.

The application of this work in constraint diagrams, and in particular the application of reasoning rules, gives another extension of this work. This will require a dynamic approach, because if a user requests from a tool that a reasoning rule is applied to an existing diagram, this existing diagram will be already drawn and suitably laid out. To maintain the users' mental map of the diagram the application of the rule should endeavour to preserve as much of the original layout as possible. If some new components are required, they can be added and manipulated under the guidance of the diagram metrics, while the original contours remain unchanged. This would be a relatively simple extension to the hill climbers in our current implementation.

Acknowledgements

This work has been partially supported by UK EPSRC grant GR/R63516.

References

1. G. Battista, P. Eades, R. Tamassia and I. Tollis. Graph Drawing: Algorithms for the Visualisation of Graphs. Prentice Hall. 1999.
2. R. Davidson, D. Harel. Drawing Graphs Nicely Using Simulated Annealing. ACM Trans. Graphics, 15(4):301-331, 1996.
3. P. Eades. A Heuristic for Graph Drawing. Congressus Numerantium 42. pp. 149-60. 1984.
4. J. Flower, J. Howse, J. Taylor, Nesting in Euler Diagrams. GT-VMT-02, International Workshop on Graph Transformation and Visual Modeling Techniques, October, pp. 99-108. 2002.
5. J. Flower and J. Howse, Generating Euler Diagrams, proc. Diagrams 2002, Springer Verlag, 61-75.
6. Y. Gil, J. Howse, S. Kent, Constraint Diagrams: a step beyond UML, Proc. TOOLS USA 1999, IEEE Computer Society Press, 453-463.
7. D. Harel. On Visual Formalisms. In J. Glasgow, N. H. Narayan, and B. Chandrasekaran, editors, Diagrammatic Reasoning, pages 235–271. MIT Press, 1995.
8. J. Howse, F. Molina, J. Taylor, S. Kent, J. Gil, Spider Diagrams: A Diagrammatic Reasoning System, Journal of Visual Languages and Computing. Vol. 12, No. 3, Jun 2001, 299-324.
9. Kent Modelling Framework (KMF). Home Page www.cs.ukc.ac.uk/kmf
10. M. Kaufmann and D. Wagner. Drawing Graphs: Methods and Models, LNCS 2025. 2001.
11. M.H.W. Hobbs and P.J. Rodgers Representing Space: A Hybrid Genetic Algorithm for Aesthetic Graph Layout. In FEA'98 Frontiers in Evolutionary Algorithms, Proceedings of JCIS'98 The Fourth Joint Conference on Information Sciences, volume 2, pages 415-418, October 1998.
12. F. Ruskey. A Survey of Venn Diagrams. The Electronic Journal of Combinatorics. March 2001.