Drawing Euler Diagrams from Region Connection Calculus Specifications with local search

François Schwarzentruber[1] and Jin-Kao Hao[2]

[1] ENS Rennes, Campus de Ker lann, Av Robert Schumann, 35170 Bruz, France
[2] LERIA, Université d'Angers, 2 Bd Lavoisier, 49045 Angers Cedex 01, France

Abstract. This paper describes a local search based approach and a software tool to approximate the problem of drawing Euler diagrams. Specifications are written using **RCC-8**-constraints and radius constraints. Euler diagrams are described as set of circles.

1 Introduction

Euler diagrams are pictures to understand relations between concepts. Sets (= concepts) are represented as regions in the plane and inclusions or intersections of those regions depict inclusions or intersections of the corresponding sets. Euler diagrams are a very general tool that is used in a wide range of application areas. For instance, Figure 1 shows an Euler diagram representing the relations of the complexity classes as many computer scientists may believe they are. Generally speaking, the user may want to generate automatically Euler diagrams from a knowledge base expressed in description logic [20].

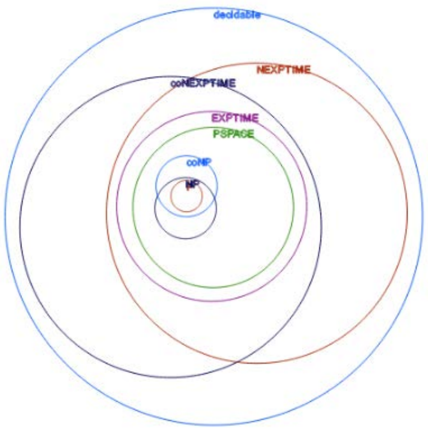

Fig. 1. Euler diagram drawn by our tool

This paper presents a proof-of-concept software tool for drawing Euler diagrams by constraint solving with local search available here: `http://www.irisa.fr/prive/fschwarz/constrainteddrawing`.

Given a set of geometrical objects and a set of constraints over these objects, the objective is to find a drawing that contains the geometrical objects and satisfies the set of given constraints. In this work, objects are circles and our first aim is to draw Euler diagrams from constraints given as an input.

As discussed in Section 6 on related work, there are different approaches to solve this problem of drawing Euler diagrams. One prominent work is to use algorithms coming from graph theory [17]. Nevertheless, when we deal with drawing generation, there is a need which seems essential: the user should be offered the possibility of interacting with the generated drawing. For instance, the user should be allowed to move or resize a circle in the current Euler diagram. Then the system should be able to take into account the input of the user and correct the drawing with respect to the constraints. With the graph theory approach, it seems difficult and even impossible to correct Euler diagrams interactively. That is why, we claim that the local search approach, thanks to its high flexibility, is a suitable method for this problem. Local search both takes into account the input of the user and corrects the picture in order to always satisfy the constraints.

Concerning the constraint specification language, we decided to start from **RCC-8** constraints [11], where **RCC** stands for 'Region connection calculus'. Indeed, **RCC-8** is a desirable formalism for constraint specifications of Euler diagrams. It provides constraints that describe pure set theory concepts. For instance, the proposition 'a and b are disjoint' is expressed as the circles representing the relation 'a and b are disconnected' (DC). This information may come from a knowledge base where the description logic formula $a \sqcap b = \neg\top$ is inferred [20].

Moreover, this tool may be used to draw Euler diagrams representing sets in a topological space (for instance this tool may be used by a math teacher in an introductory course in topology). And **RCC-8** also provides topological concepts. As in illustration, if a and b are sets in a topological space such that interiors of a and b are disjoint and closures of a and b are not disjoint, we may express that the circles representing a and b are externally connected (EC) (see Figure 2).

The paper is organized as follows. In section 2 we describe the language we consider. Section 3 gives the semantics. Section 4 is dedicated to the local search procedure. Section 5 presents the implementation. Section 6 reviews related work. Perspectives are provided in the concluding section. Proofs and experimental results are in [16].

2 Syntax

The syntax of the language $\mathcal{L}$ of constraints is defined by the following rule:

$$\varphi \ ::= \ R(a,b) \ \mid \ radius(a) = r \ \mid \ (\varphi \vee \varphi)$$

where a and b range over a set of constant symbols, r is a rational number and R ranges over the symbol predicates of **RCC-8**. Intuitive meanings of **RCC-8**-relations are given in table 1 and figure 2 gives them in pictures $\mathbb{REL}_{\mathbf{RCC\text{-}8}}$.

3 Semantics

Usually in logic, semantics is given in terms of truth values. A formula φ is either true or false in a given model. But, for the local search algorithm, we need the semantics to be soft and we measure how much a formula φ is true (or false).

Table 1. Intuitive meanings of **RCC-8**-relations

Construction	Intuitive meaning
$DC(a, b)$	a and b are disconnected
$EC(a, b)$	a and b are externally connected
$PO(a, b)$	a and b partially overlap
$TPP(a, b)$	a is a tangential proper part of b
$TPP^{-1}(a, b)$	b is a tangential proper part of a
$NTPP(a, b)$	a is a non-tangential proper part of b
$NTPP^{-1}(a, b)$	b is a non-tangential proper part of a
$EQ(a, b)$	a and b are equal
$radius(a) = r$	the radius of the circle a is r
$(\varphi \vee \psi)$	φ or ψ

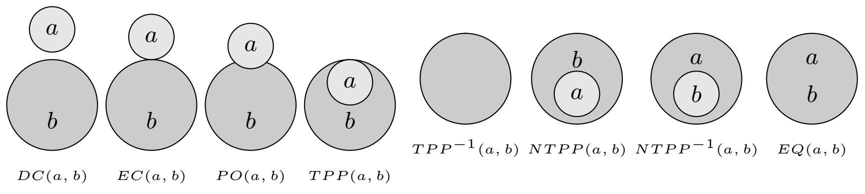

Fig. 2. The eight **RCC-8**-relations in pictures

First we define the hard semantics of our language $\mathcal{L}$. Second we define the soft semantics and we make a correspondence between them.

Models are pairs $\mathcal{M} = \langle C, i \rangle$ where:

- C is a non-empty set of circles of non-zero radius in the plane (for all $c \in C$, we respectively denote $c.x$, $c.y$ and $c.r > 0$ the abscissa, the ordinate and the radius of the circle c ; we note $c.c$ the center of c);
- i assigns to each constant symbol an element in C.

To ease the readability, a constant symbol a also designates $i(a)$, that is the circle represented by a in a model $\mathcal{M}$.

3.1 Hard Semantics

Let us define the truth conditions as follows.

Definition 1. *Let $\mathcal{M} = \langle C, i \rangle$ a model. We define the relation $\mathcal{M} \models \varphi$ by induction on $\varphi \in \mathcal{L}$ as follows:*

$\mathcal{M} \models DC(a,b)$	*iff* $d(a.c, b.c) > a.r + b.r$;
$\mathcal{M} \models EC(a,b)$	*iff* $d(a.c, b.c) = a.r + b.r$;
$\mathcal{M} \models PO(a,b)$	*iff* $d(a.c, b.c) \in]\|a.r - b.r\|, a.r + b.r[$;
$\mathcal{M} \models TPP(a,b)$	*iff* $d(a.c, b.c) = b.r - a.r$ *(and* $a.r \leq b.r$*)*;
$\mathcal{M} \models TPP^{-1}(a,b)$	*iff* $d(a.c, b.c) = a.r - b.r$ *(and* $b.r \leq a.r$*)*;
$\mathcal{M} \models NTPP(a,b)$	*iff* $d(a.c, b.c) < b.r - a.r$ *(and* $a.r < b.r$*)*;
$\mathcal{M} \models NTPP^{-1}(a,b)$	*iff* $d(a.c, b.c) < a.r - b.r$ *(and* $a.r > b.r$*)*;
$\mathcal{M} \models EQ(a,b)$	*iff* $a.c = b.c$ *and* $a.r = b.r$;
$\mathcal{M} \models radius(a) = r$	*iff* $i(a).r = r$;
$\mathcal{M} \models (\varphi \vee \psi)$	*iff* $\mathcal{M} \models \varphi$ *or* $\mathcal{M} \models \psi$.

The problem we tackle here is defined as follows:

- input: a finite set $I = \langle \varphi_1, \ldots, \varphi_n \rangle$ of constraints in $\mathcal{L}$;
- output: a model $\mathcal{M}$ such that for all $i \in \{1, \ldots, n\}$, $\mathcal{M} \models \varphi_i$.

The corresponding decision problem $\mathcal{L}$-SAT takes the same input and outputs yes, iff there exists a model $\mathcal{M}$ such that for all $i \in \{1, \ldots, n\}$, $\mathcal{M} \models \varphi_i$.

Proposition 1. *$\mathcal{L}$-SAT is NP-hard and in PSPACE.*

3.2 Soft Semantics

φ	Objective functions $obj(\varphi)$
$DC(a,b)$	$max(0, 2^{(a.r+b.r)-d(a.c,b.c)} - 0.0001)$
$EC(a,b)$	$\|d(a.c, b.c) - (a.r + b.r)\|$
$PO(a,b)$	$\|d(a.c, b.c) - max(a.r, b.r)\|$
$TPP(a,b)$	$\|d(a.c, b.c) - (b.r - a.r)\|$
$TPP^{-1}(a,b)$	constraint of $TPP(b,a)$
$NTPP(a,b)$	$\|d(a.c, b.c) - \frac{(b.r-a.r)}{2}\| + max\left(0, 0001 + \frac{a.r-b.r}{b.r}\right)$
$NTPP^{-1}(a,b)$	constraint of $NTPP(b,a)$
$EQ(a,b)$	$d(a.c, b.c) + \|a.r - b.r\|$
$radius(a) = r$	$\|a.r - r\|$
$\varphi \vee \psi$	$min(obj(\varphi), obj(\psi))$

Fig. 3. Objective functions

Now, a formula is evaluated according to an objective function $obj : \mathcal{L} \rightarrow \mathbb{R}$, defined by induction on φ as given in figure 3. Now we interpret obj over models $\mathcal{M}$. We note $obj(\varphi)_{\mathcal{M}}$ the value obtained in $\mathcal{M}$.

Proposition 2. *If $obj(\varphi)_{\mathcal{M}} = 0$, then $\mathcal{M} \models \varphi$.*

Note that one could have chosen other objective functions than proposition 2. Those objective functions have been chosen experimentally so that the local search algorithm described in the next section works.

4 Local Search

Given a problem instance $I = \langle \varphi_1, \ldots, \varphi_n \rangle$, we use a local search approach to determine an Euler diagram respecting the constraints of I. Generally speaking, local search constitutes a simple optimization approach which improves iteratively the current solution based on a neighborhood relation [10]. In our case, the local search algorithm explores the search space Ω of possible drawings $\mathcal{M}$ of a set of circles with the purpose of finding a feasible drawing satisfying the predicates (constraints) of the given formula. The pseudo-code is defined as follows:

```
$\mathcal{M}$ := generate randomly a drawing
while true do
    $\mathcal{M}_{new} := getSolutionInNeighborhood(\mathcal{M})$
    if $\mathcal{M}_{new}$ is better than $\mathcal{M}$ then
        $\mathcal{M} := \mathcal{M}_{new}$
```

The algorithm never stops and keeps improving the current solution $\mathcal{M}$. To represent a model $\mathcal{M}$ (i.e. a drawing), $\mathcal{M}$ is considered as a vector, where indices are constant symbols a and each element $\mathcal{M}[a]$ is a circle represented by its center $(\mathcal{M}[a].x, \mathcal{M}[a].y)$ and its radius $\mathcal{M}[i].r$.

The function $getSolutionInNeighborhood(\mathcal{M})$ returns a new solution $\mathcal{M}_{new}$, where for all constant symbols a, $\mathcal{M}_{new}[a].x$, $\mathcal{M}_{new}[a].y$, $\mathcal{M}_{new}[a].r$ are respectively obtained by adding randomly chosen numbers in an interval $[-\epsilon, \epsilon]$ to respectively $\mathcal{M}[a].x$, $\mathcal{M}[a].y$, $\mathcal{M}[a].r$. That is, a new drawing is obtained by moving every circle center from its current position to a new position and modifying slightly each radius (this move operator defines thus the neighborhood relation of our local search algorithm).

Solutions are compared with the following total order.

Definition 2. *Given two candidate solutions (drawings) $\mathcal{M}, \mathcal{M}_{new} \in \Omega$, $\mathcal{M}_{new}$ is better than $\mathcal{M}$ if $\sum_{i=1}^{n} obj(\varphi_i)_{\mathcal{M}_{new}} \leq \sum_{i=1}^{n} obj(\varphi_i)_{\mathcal{M}}$, where $obj(\varphi_i)_{\mathcal{M}}$ and $obj(\varphi_i)_{\mathcal{M}_{new}}$ are the values of the objective function $obj(\varphi_i)$ that corresponds to the i^{th} constraint φ_i for respectively $\mathcal{M}_{new}$ and $\mathcal{M}$.*

5 Implementation

Our local search algorithm available as a web application written in Javascript can be found here: `http://www.irisa.fr/prive/fschwarz/constrainteddrawing`.

5.1 Syntax Used in The Software

The user can add circles and constraints by clicking on the appropriate buttons in the palette. Let us describe the syntax we use in the software to define circles and constraints. In the left part of the screen, the user adds a circle by writing `circle(name);` where `name` is a string for the name of the circle. Constraints are created with functions. For instance `TPP(name1, name2)` creates a TPP constraints between the circle named `name1` and the circle named `name2`. The construction `or(constraint1, constraint2)` returns a constraint that represents the disjunction of `constraint1` and `constraint2`. The construction `addConstraint(constraint)` adds the constraint `constraint` in the set of constraints.

5.2 Interaction

The user may *assist* the local search. During the local search, the user can move the circles by drag and drop and modify the radius of each circle. When the user makes a modification in the drawing, she directly modifies the current model $\mathcal{M}$. Those modifications are directly taken in account in real-time by the local search algorithm.

6 Related Work

6.1 Region Connection Calculus

RCC-8 [11] is a first order logic for spatial reasoning. Contrary to the version we adopt in this article, variables are interpreted by regions of an abstract topological space. The satisfiability problem of a first order formula given in **RCC-8** is undecidable, more precisely not recursively enumerable [7].

Nevertheless, the satisfiability problem, called RSAT, of a formula of the form $\exists x_1, \ldots \exists x_n, \bigwedge_{i,j \in \{1,\ldots n\}} \bigvee_{R \in C(i,j)} R(x_i, x_j)$ where n is a positive integer, $C(i, j)$ a subset of $\mathbb{REL}_{\mathbf{RCC\text{-}8}}$, is NP-complete [12].

The satisfiability problem for **RCC-8** formulas over *disc-homeomorphs* is NP-complete [14,13]. We should have emphasized that the problem addressed in our paper is not about disc-homeomorphs but about discs. We here tackle the satisfiability problem for **RCC-8** formulas over *discs* its exact complexity is still an open problem (see proposition 1). An extension of **RCC-8** with Boolean operations over sets has been studied in [6]. Soft semantics for **RCC-8** are also given in [15,18].

6.2 Constrained Graph Drawing

Drawing with Constraint. Constraints have long been used for graph drawing. Generally, the positions of constrained objects to draw can be computed in polynomial time. For instance, in drawing software like Geogebra, one may state, for example, that Δ_1 contains point A and is orthogonal to line Δ_2 [5]. Similarly, in a graphical user interface library, the layout is computed from easily solvable constraints as 'the window is horizontally separated in two parts. The first part is a textbox. The second contains three buttons displayed vertically'. For these systems, various layout algorithms have been studied [2,8]. Finally, there exist tools to compute nice graphical representations of graphs [1,3]. Displaying graphs consists in solving constraints such as two connected nodes are close and two different edges do not cross.

Bottom-Up Approach for Drawing Euler Diagrams. The visualization tool Tulip integrates a functionality for Euler diagrams [17]. The input of this system is given by an extensive description of the elements of sets. For instance, the following can be a possible input:

$$P := \{path, linearprog\}$$
$$NP := \{path, linearprog, intlinearprog, sat\}$$
$$coNP := \{path, linearprog, intlinearprog, valid\}$$

Tulip is a 'bottom-up' approach. It considers the elements (in the example, elements are *path*, *linearprog*, etc.) as nodes in a graph constrained by the set-theoretical relations (in the example, $P \subseteq NP$, etc.). Tulip displays the graph and extracts an Euler diagram from it. The shape of a region corresponding to a set (for instance P) is delimited by the positions of the elements in that set (for instance, *path* and *linearprog*). Thus, the shape can be arbitrary and the diagram may be difficult to read. A similar approach can be found in [21].

On the contrary, our approach is 'top-down'. The shape of the region are circles. We do not specify elements that are in sets. Furthermore, contrary to Tulip, our framework can be extended to capture constraints as 'the radius of the disc representing NP is $10cm$'.

Other top-down Approaches for Euler Diagrams. The authors of [4] describe a software tool for Euler diagrams which are made up of circles (see the site: `http://www.eulerdiagrams.com/software.htm`). Their algorithm is based on the theory of piercings [19] and is able to draw nice diagrams. Yet, their approach does not capture topological constraints as TPP (circle a is a tangential proper part of b) and the size of circles are not easily adjustable. Very recently, another interesting tool is presented in [9] which is able to draw not only circles, but also ellipses.

Compared to these tools, our approach distinguishes itself by some interesting features. First, our tool is based on the **RCC-8** language which enables both precise and rich constraint specifications. For instance, our tool allows the specification of topological constraints. Second, one can specify the radius of circles, and our system can then adjust dynamically these radius for a better visualization. Last but not least, in our approach, the user can always modify the drawing by moving and resizing circles and the system will adjust the drawing accordingly and adaptively.

7 Conclusion

This study makes the bridge between logical framework **RCC-8**, generation of Euler diagrams (and more generally drawings under constraints), as well as heuristic search.

A first extension is to add a large collection of elements in addition of circles (rectangles, splines, etc.). Then an interesting perspective is to combine constraints that do not require search (for instance constraints of Geogebra, or tractable fragments of **RCC-8** [12]) and constraints that require search. That is, the tool should be able to choose how to solve the constraints by detecting which method to apply and on which part of the drawing.

Another perspective is to improve the graphical interface. From a better graphical interface, we can start to make the tool tested by users and do experimental validations (can users write constraints they need? do users feel as if the tool understands their constraints?).

Finally, it would be interesting to integrate default reasoning in the tool. For instance the sole constraint P `TPP` NP (tangential proper part) should avoid the radius of P to be too small. This may be solved by using default reasoning: *by default*, P `TPP` NP implies that the radius of P is approximately the half of the radius of NP.

Another interesting research problem concerns the axiomatization. Is there an axiomation of **RCC-8** where objects are circles? Having an axiomatization may help us to improve the software so that it could give explanations for the generated drawings.

References

1. Auber, D.: Tulip a huge graph visualization framework. In: Graph Drawing Software, pp. 105–126. Springer (2004)
2. Borning, A., Marriott, K., Stuckey, P.J., Xiao, Y.: Solving linear arithmetic constraints for user interface applications. In: ACM Symposium on User Interface Software and Technology, pp. 87–96 (1997)
3. Ellson, J., Gansner, E.R., Koutsofios, E., North, S.C., Woodhull, G.: Graphviz and dynagraph static and dynamic graph drawing tools. In: Graph Drawing Software, pp. 127–148. Springer (2004)
4. Flower, J., Howse, J.: Generating euler diagrams. In: Hegarty, M., Meyer, B., Narayanan, N.H. (eds.) Diagrams 2002. LNCS (LNAI), vol. 2317, pp. 61–75. Springer, Heidelberg (2002)
5. Hohenwarter, M., Preiner, J.: Dynamic mathematics with geogebra. Journal of Online Mathematics and its Applications, 7 (2007)
6. Kontchakov, R., Nenov, Y., Pratt-Hartmann, I., Zakharyaschev, M.: On the decidability of connectedness constraints in 2d and 3d euclidean spaces. In: IJCAI Proceedings-International Joint Conference on Artificial Intelligence, vol. 22, p. 957 (2011)
7. Lutz, C., Wolter, F.: Modal logics of topological relations. Logical Methods in Computer Science 2, 1–14 (2006)
8. Marriott, K., Moulder, P., Stuckey, P.J.: Solving disjunctive constraints for interactive graphical applications. In: Walsh, T. (ed.) CP 2001. LNCS, vol. 2239, pp. 361–376. Springer, Heidelberg (2001)
9. Micallef, L., Rodgers, P.: Drawing area-proportional venn-3 diagrams using ellipses. In: 2012 Grace Hopper Celebration of Women in Computing, ACM Student Research Competition and Poster Session. ACM Press (2012)
10. Papadimitriou, C.H.: Computational complexity. John Wiley and Sons Ltd. (2003)
11. Randell, D.A., Cui, Z., Cohn, A.G.: A spatial logic based on regions and connection. KR 92, 165–176 (1992)
12. Renz, J., Nebel, B.: On the complexity of qualitative spatial reasoning: A maximal tractable fragment of the region connection calculus. Artificial Intelligence 108(1), 69–123 (1999)
13. Schaefer, M., Sedgwick, E., Štefankovič, D.: Recognizing string graphs in np. Journal of Computer and System Sciences 67(2), 365–380 (2003)
14. Schaefer, M., Stefankovic, D.: Decidability of string graphs. In: Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing, pp. 241–246. ACM (2001)
15. Schockaert, S., De Cock, M., Kerre, E.E.: Spatial reasoning in a fuzzy region connection calculus. Artificial Intelligence 173(2), 258–298 (2009)
16. Schwarzentruber, F., Hao, J.-K.: Drawing euler diagrams from region connection calculus specifications. Technical report
17. Simonetto, P., Auber, D., Archambault, D.: Fully automatic visualisation of overlapping sets. In: Computer Graphics Forum, vol. 28, pp. 967–974. Wiley Online Library (2009)

18. Sridhar, M., Cohn, A.G., Hogg, D.C.: From video to rcc8: exploiting a distance based semantics to stabilise the interpretation of mereotopological relations. In: Egenhofer, M., Giudice, N., Moratz, R., Worboys, M. (eds.) COSIT 2011. LNCS, vol. 6899, pp. 110–125. Springer, Heidelberg (2011)
19. Stapleton, G., Zhang, L., Howse, J., Rodgers, P.: Drawing euler diagrams with circles: The theory of piercings. IEEE Trans. Vis. Comput. Graph. 17(7), 1020–1032 (2011)
20. Van Harmelen, F., Lifschitz, V., Porter, B.: Handbook of knowledge representation, vol. 1. Elsevier (2008)
21. Verroust, A., Viaud, M.-L.: Ensuring the drawability of extended euler diagrams for up to 8 sets. In: Blackwell, A.F., Marriott, K., Shimojima, A. (eds.) Diagrams 2004. LNCS (LNAI), vol. 2980, pp. 128–141. Springer, Heidelberg (2004)