Karlsruhe Institute of Technology

Minimum-weight tree supports and tree-shaped area-proportional Euler diagrams

Studienarbeit von

Boris Klemz

An der Fakultät für Informatik
Institut für theoretische Informatik
Lehrstuhl Algorithmik I

Gutachter:	Prof. Dr. Dorothea Wagner
Betreuende Mitarbeiter:	Dr. Tamara Mchedlidze Dr. Martin Nöllenburg

Bearbeitungszeit: 1. Juli 2013 – 21. Oktober 2013

Acknowledgements

I want to thank my advisors Dr. Tamara Mchedlidze and Dr. Martin Nöllenburg for the many hours of interesting discussions and great support in the form of helpful advises and corrections during the writing process.

I declare that I have developed and written the enclosed thesis by myself, and have not used sources or means without declaration in the text.

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Karlsruhe, 21. Oktober 2013

. . . .

(**Boris Klemz**)

This is the revised version of the thesis, in which the runtime of Algorithm MTS-general was more closely examined.

Karlsruhe, 15. November 2013

Abstract

Euler diagrams allow the visual representation of set systems and their intersection relations. Every set is represented by a collection of closed curves such that elements of two or more collections intersect if and only if the intersection of the corresponding sets is non-empty. In the area-proportional case each intersection's area conveys numerical information. There exist several well-formedness conditions for Euler diagrams, which formalize desirable aesthetic properties. In this thesis we explore a close relationship between hypergraphs and Euler diagrams to obtain an area-proportional representation of the latter. A support for a hypergraph H is a graph G on the same set of vertices such that each hyperedge of H induces a connected subgraph of G. Supports that are trees serve as a basic tool for our purpose and are called tree-supports.

We present an efficient approach that generates an area-proportional Euler diagram if there exists a tree-support for the input's hypergraph. The generated Euler diagrams are guaranteed to satisfy a set of well-formedness conditions. Particularly, all used curves are simple, exactly one curve is assigned to each set, there exists exactly one connected region in the plane for every set intersection and finally, all the regions are convex. Additionally, our tree-support minimizes the total number of concurrent curves that separate the pairs of internal regions that are placed next to each other in the plane.

As a byproduct, we obtain an algorithm that, for a given hypergraph, computes a minimum-weight tree-support for an arbitrary edge-weight function (if one exists). The algorithm has running time $O(n^2(m+\log n))$, where n is the number of vertices and m is the number of hyperedges. This improves the best known previous algorithm for this problem [KS03], which has a time complexity of $O(n^4m^2)$.

Deutsche Zusammenfassung

Euler Diagramme erlauben die visuelle Darstellung von Mengenbeziehungen. Jede Menge wird durch eine Menge geschlossener Kurven repräsentiert, so dass sich zwei oder mehr Kurven aus verschiedenen Kurvenmengen genau dann schneiden, wenn der Schnitt der entsprechenden Mengen nicht leer ist. In der flächenproportionalen Variante liefert zudem der Flächeninhalt jedes Schnitts numerische Informationen. Sogenannte Wohlgeformtheitsbedingungen formalisieren wünschenswerte ästhetische Eigenschaften von Euler Diagrammen. In dieser Studienarbeit untersuchen wir einen engen Zusammenhang zwischen Hypergraphen und Euler Diagrammen, um flächenproportionale Euler Diagramme zu generieren. Ein Support eines Hypergraphen H ist ein Graph G, wobei G und H dieselbe Menge von Knoten haben und jede Hyperkante von H einen zusammenhängenden Subgraphen von G induziert. Supports, die Bäume sind, spielen in dem entwickelten Visualisierungsverfahren eine wichtige Rolle und werden Baumsupports genannt.

Das vorgestellte Verfahren erlaubt das effiziente Zeichnen eines flächenproportionalen Euler Diagramms, falls ein Baumsupport für den Hypergraphen der Eingabe existiert. Die generierten Euler Diagramme erfüllen diverse Wohlgeformtheitsbedingungen, so dass garantiert ist, dass nur einfache Kurven verwendet werden, dass jeder Menge genau eine Kurve zugewiesen wird, dass es für jede Schnittmenge genau ein repräsentierendes Gebiet in der Ebene gibt, sowie dass jedes dieser Gebiete konvex ist. Weiterhin wird die Gesamtzahl der überlappenden Kurven minimiert, die jedes Paar von inneren Gebieten, die in der Ebene nebeneinander liegen, separiert.

Für einen Teilschritt des Verfahrens entwickeln wir einen Algorithmus, der für einen gegebenen Hypergraphen einen Baumsupport mit minimalem Gewicht bezüglich einer beliebigen Kantengewichtsfunktion berechnet (falls ein solcher existiert). Die Laufzeit des Algorithmus ist $O(n^2(m + \log n))$, wobei n die Anzahl der Knoten und m die Anzahl der Hyperkanten ist. Damit wird eine Verbesserung gegenüber dem bisher besten bekannten Algorithmus für dieses Problem [KS03], dessen Zeitaufwand $O(n^4m^2)$ beträgt, erzielt.

Contents

1. Introduction

Euler diagrams were first introduced by Euler [EdCC43] and allow visual representation of relationships between sets. The Euler diagram depicted in Figure 1.1, for example, conveys the information that some animals can swim, some animals can fly, some animals can even swim and fly and that no animal is a tree and vice versa. Euler diagrams can also convey cardinal information about the represented set intersections and are in this case called area-proportional. If the Euler diagram in Figure 1.1 is interpreted as an area-proportional Euler diagram, it additionally conveys the information, for example, that there are more animals that can swim than animals that can fly. Euler diagrams are used to visualize data in many areas including computer file organization [DCES03], library environments [TVVb05] and medicine [SDC$^+$03].

Given an abstract Euler diagram description, which consists of a list of sets and a list of set intersections (and in the area-proportional case an area specification for each set intersection), the generally considered task is to generate an (area-proportional) Euler diagram that visualizes the given data. Since the purpose of Euler diagrams is to visually convey information to viewers, it is desirable that the generated Euler diagrams have nice aesthetic properties, which are formalized as so called well-formedness conditions. Depending on the input data, one might want to ensure that all regions of the resulting diagram are convex, that all used curves are simple, that at no point more than two curves intersect at once or that none of the sets is represented by more than one curve.

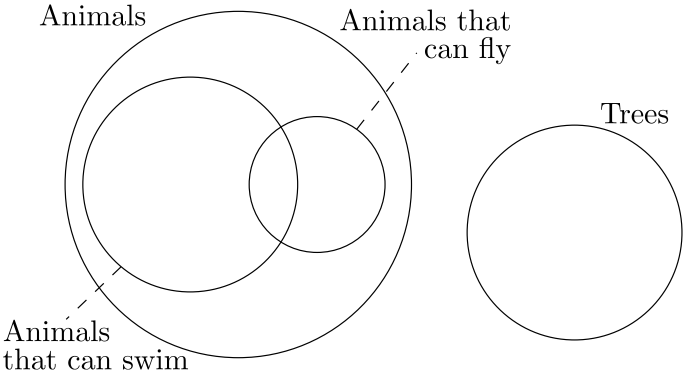


Figure 1.1: An Euler diagram.

A *hypergraph* H is an ordered pair (V, S), where V is a set of vertices and S is a set of *hyperedges*, which are subsets of V. Hypergraphs can be seen as a generalization of graphs, whose edges connect exactly 2 vertices. A *support* for hypergraph H is a graph G with vertex set V such that for every hyperedge $s \in S$ the subgraph of G induced by the vertex set s is connected. In addition, if G is a tree, G is called *tree-support* and if there exists a tree-support for H, then H is called *tree-hypergraph.* Application areas for hypergraphs and supports include modeling relational database schemes [BFMY83] and social networks [BWR07]. Korach and Stern [KS03] introduced and solved the problem of finding a tree-support T for a hypergraph H such that the weight of T regarding some weight function is minimal amongst all tree-supports for H (in [KS03] such tree-supports are called 'minimum cost clustering spanning trees'). They motivated this problem with the following scenario related to communication networks. Given is a graph $G = (V, E)$ and a collection S of non-disjoint subsets of V. Each vertex of V represents a customer, each edge $e \in E$ represents a potential connection between two customers that can be constructed for a cost of $c(e)$ and each set $s \in S$ represent groups of customers. The task is to construct a tree-shaped communication network $T = (V, E')$, $E' \subseteq E$ with minimal construction cost such that each group of customers of S is connected by a subtree of T, the motivation being that none of the groups of customers is sensitive to network faults occurring outside of their group.

Chow [Cho07] related the topics of Euler diagrams and hypergraphs by mapping a set system, which is collections of subsets of an item set, to a hypergraph. Analogously, we map an abstract Euler diagram description D to a hypergraph $\mathrm{H}(D)$, which we call labeled hypergraph for D (formally defined in Section 2.1). This hypergraph contains a vertex for each specified set intersection and a hyperedge for every set. The hyperedge corresponding to a set s contains precisely the vertices that correspond to set intersetions that are subsets of s.

In this thesis we present a framework for algorithms that generate area-proportional Euler diagrams for the class of abstract Euler diagram descriptions whose labeled hypergraphs are tree-hypergraphs. The generated Euler diagrams are guaranteed to satisfy several well-formedness conditions. We also present an algorithm that computes tree-supports with minimal weight. The algorithm has runtime complexity $O(n^2(m + \log n))$, where n is the number of vertices and m is the number of hyperedges of the input hypergraph. It therefore improves upon the algorithm presented in [KS03], whose runtime complexity is $O(n^4 m^2)$.

Now, in Section 1.1, we recall basic concepts of graph theory, drawings in the plane, Euler diagrams and hypergraphs and introduce related definitions and notation. We also present an overview of related work. Thereafter, in Section 1.2, we more precisely delineate our contribution and present an overview of the remainder of this thesis.

1.1 Basic concepts and related work

In this section we introduce and recall the basic concepts, definitions and notation related to the results of this thesis. We cover graph theory and drawings in the plane, Euler diagrams and their well-formedness conditions and hypergraphs and their supports. We also present an overview of work related to our results in the fields of Euler diagram generation and the computation of supports for hypergraphs.

1.1.1 Graph theory and drawings in the plane

In this subsection we recall basic graph-theoretic concepts and concepts related to drawings in the plane. Note that the corresponding definitions are adopted to the context of this thesis and may therefore slightly differ in the literature.

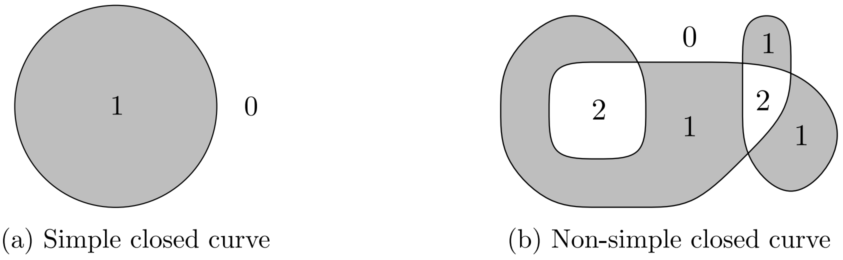
A *graph* G is an ordered pair (V, E), where V is a set of *vertices* and E is a set of *edges*, which are subsets of V with cardinality 2. We also use $\mathrm{V}(G)$ to refer to the vertex set and $\mathrm{E}(G)$ to refer to the edge set of G. If there exists an edge $\{u, v\} \in E$ for any two distinct vertices $v, u \in V$, then G is called *complete* graph. A vertex $v \in V$ and an edge $e \in E$ are said to be *incident* if $v \in e$. The *degree* $\mathrm{d}(v) = |\{e \in E \mid v \in e\}|$ of a vertex $v \in V$ is the number of edges incident to v. Two vertices $u, v \in V$ are *neighbours* in G if there exists an edge that is incident to both u and v. A *subgraph* of G is a graph $G' = (V', E')$ with $V' \subseteq V$ and $E' \subseteq E$. The graph G' is said to be *induced* by vertex set V' if $E' = \{\{v, u\} \in E \mid v, u \in V'\}$ and it is said to be *induced* by edge set E' if $V' = \{v \in V \mid \exists e' \in E' : v \in e'\}$.

Let $(v_1, \ldots, v_k) \in V^k$ be a sequence of vertices. We refer to the corresponding sequence of edges $p = (e_1, \ldots, e_{k-1}) \in E^{k-1}$ with $e_i = \{v_i, v_{i+1}\}$ for $1 \leq i \leq k-1$ as a *path* in G between v_1 and v_k and we say that the edges $e_1, \ldots, e_{k-1}$ *constitute* a path. A path $(e_j, \ldots, e_l), 1 \leq j < l \leq k-1$ is called *subpath* of p. A path is said to be *simple* if the vertices $v_1, \ldots, v_k$ except for maybe v_1 and v_k are pairwise distinct. If $v_1 = v_k$, we refer to p as a *cycle*. We say that the edge set E *contains* a cycle if there exists a sequence of edges of E which is a cycle. Since we will only consider simple paths and cycles, we will usually omit the term simple, unless this property is particularly important.

Two vertices of V are said to be *connected* in G if there exists a path with edges of E between them. A *connected component* C of the graph G is a subgraph of G such that the vertices in $\mathrm{V}(C)$ are pairwise connected and there exists no path between a vertex of $\mathrm{V}(C)$ and a vertex of $V \setminus \mathrm{V}(C)$. A graph is *connected* if it has a single connected component. A connected graph whose edge set does not contain a cycle is called a *tree*. If all connected components of a graph are trees, the graph is called a *forest*. Let T be a tree and $v, u \in \mathrm{V}(T)$. Since the edge set of T does not contain a cycle and T is connected, there exists exactly one path between v and u in T. We use $\mathrm{p}(v, u, T)$ to refer to this path. A subgraph of the graph $G = (V, E)$ that is a tree is called *subtree* of G. A subtree of G whose vertex set is V is called a *spanning-tree* of G. The graph G is called *edge-weighted* if a function $w : E \to \mathbb{R}$ is provided that assigns a real *weight* to every edge in E. The *weight* of a subgraph G' of G regarding w is the sum of the weights that w assigns to the edges of $\mathrm{E}(G')$. A *minimum* spanning-tree of G regarding w is a spanning-tree of G that has minimum weight regarding w amongst all spanning-trees of G.

A *breadth-first search* [CLRS09] is a strategy for searching a vertex $v \in V$ of graph G with $\mathrm{V}(G) = V$ by traversing the vertices of G starting at some vertex $v_1 \in V$. A first-come-first-served queue Q is initialized with v_1. In every step the first vertex u in Q is removed and all not previously visited neighbours of u are added to Q. This process is repeated until v is found or Q is empty. If all vertices of V are visited with this strategy, we call the sequence $(v_1, \ldots, v_{|V|})$ in which they are visited a *breadth-first traversal* of G.

A *curve* in the plane is a continuous function $c : [a, b] \to \mathbb{R}^2$ that maps elements of a real interval to points in the plane. The points $c(a)$ and $c(b)$ are called *endpoints* of c. If $c(a) = c(b)$, then c is called a *closed* curve. The *image* of c is the set of points $\mathrm{image}(c) = \{p \in \mathbb{R}^2 \mid \exists d \in [a, b] : c(d) = p\}$. Let $d, e \in (a, b)$ be real numbers. The curve c is called *simple* if $c(d) = c(e)$ implies that $d = e$. A simple and closed curve c partitions the plane into two regions, which are the connected components of $\mathbb{R}^2 \setminus \mathrm{image}(c)$. One of these regions is bounded, the other region is unbounded. We say a point is *interior* to c if it is located in the bounded region and it is *exterior* if it located in the unbounded region. For non-simple closed curves the concept of a point being interior or exterior can be more generally defined by the points *winding number* [SRHT07] which informally is the number of times the curve 'winds' around the point. If this number is odd, then the point is interior, if it is even, the point is exterior. We use $\mathrm{interior}(c)$ and $\mathrm{exterior}(c)$ to denote the



(a) Simple closed curve

(b) Non-simple closed curve

Figure 1.2: Images of closed curves (black), the sets of points interior (grey) and exterior (white) to them and the corresponding winding numbers.

set of points interior or exterior to a closed curve c respectively. Figure 1.2 illustrates this concept.

A *drawing* of graph $G = (V, E)$ can be obtained by mapping the vertices of V to distinct points in the plane and representing the edges of E with simple curves in the plane such that the endpoints of the curve associated with an edge $\{v, u\} \in E$ are the points associated with u and v. A drawing of G is called *planar* if the images of the curves associated with the edges of E only intersect at their respective endpoints and therefore only at points associated with vertices. The graph G is called *planar* if there exists a planar drawing of G. The regions bounded by the images of the curves in a planar drawing of a graph are called *faces*. The outer, infinitely large region is called *outer face*. Let f be a face of the planar drawing D of G, let E_f be the set of edges that correspond to the curves whose images bound f in D and let $G_f = (V_f, E_f)$ be the subgraph of G induced by E_f. The vertices in V_f are said to be *adjacent* to f in D. If there exists a planar drawing of the graph $G = (V, E)$ such that all vertices in V are adjacent to the outer face of the drawing, then G is called *outerplanar* or 1-*outerplanar*. The graph G is called k-*outerplanar*, with $k \geq 2$, if there exists a planar drawing of G such that if all vertices adjacent to the outer face of the drawing are removed from V all of the connected components of the subgraph of G induced by the remaining vertices are $(k-1)$-outerplanar.

1.1.2 Euler diagrams

In the literature there exists no uniform set of definitions for the concept of Euler diagrams. In fact the definitions for Euler diagrams and their well-formedness conditions tend to differ severely from author to author. In this subsection we therefore first present the set of definitions that is used in this thesis, then comment on some of the differences to the definitions used by other authors and conclude with an overview of related work.

The following set of definitions adepts and combines the definitions given in [FFH08] and [SRH11]. An *Euler diagram* is an ordered pair $\mathcal{D} = (\mathcal{C}, \mathcal{L})$, where $\mathcal{C}$ is a finite collection of closed curves in the plane and $\mathcal{L} : \mathcal{C} \to L$ is a function that returns a curve's *label*, with L being some set of labels. The set of curves with some specific label $l \in L$ is called *contour* with label l. We use interior(c) and exterior(c) to denote the set of points interior to at least one curve of contour c or exterior to all curves of contour c respectively and say that the points in interior(c) are *interior* to c and that the points in exterior(c) are *exterior* to c. A *minimal region* of the Euler diagram $\mathcal{D} = (\mathcal{C}, \mathcal{L})$ is a connected component of $\mathbb{R}^2 \setminus \bigcup_{c \in \mathcal{C}} \text{image}(c)$, i.e., a set of points in the plane that is bounded by the union of the images of all curves of $\mathcal{D}$. Each minimal region can be described as being the interior of the simple closed curve that describes its boundary. Let m_1 and m_2 be minimal regions of $\mathcal{D}$ and c_1 and c_2 the simple closed curves that describe their respective boundaries. We say that m_1 and m_2 are *neighbours* if there exists a connected component of image(c_1)$\cap$image(c_2) that contains more than one point. Let $\mathcal{X}$ be the set of all contours

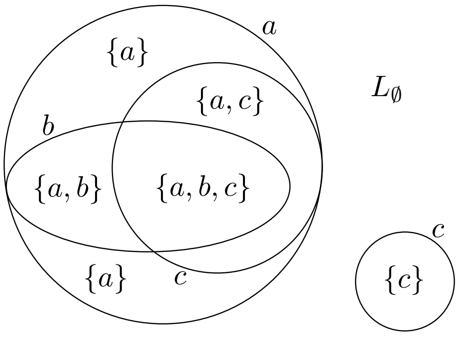


Figure 1.3: Euler diagram realizing $(\{a,b,c\},\{L_\emptyset,\{a\},\{c\},\{a,b\},\{a,c\},\{a,b,c\}\})$

of $\mathcal{D}$, let $X \subseteq \mathcal{X}$ be a set of contours of $\mathcal{D}$, let $L_X \subseteq L$ be the set of labels of the contours in X and let P_X be the set of points that is interior to the contours in X and exterior to the remaining contours of $\mathcal{D}$, more precisely $P_X = (\bigcap_{x\in X} \text{interior}(x)) \cap (\bigcap_{x\in\mathcal{X}\setminus X} \text{exterior}(x))$. If $P_X \neq \emptyset$, then P_X is called *concrete zone* of $\mathcal{D}$ with label set L_X. Note that every concrete zone of $\mathcal{D}$ is a union of minimal regions of $\mathcal{D}$ and that the concrete zone $P_\emptyset$ of $\mathcal{D}$ with the empty label set $L_\emptyset = \emptyset \subseteq L$ is the set of points $\bigcap_{x\in\mathcal{X}} \text{exterior}(x)$ which is the set of minimal regions whose points are exterior to all contours of $\mathcal{D}$.

An *abstract Euler diagram description* is an ordered pair $D = (L, Z)$, where L is a set of labels and Z is a set of *zones*, which are subsets of L with $L_\emptyset = \emptyset \in Z$. Let $\mathcal{D} = (\mathcal{C}, \mathcal{L})$ be an Euler diagram and $\mathcal{Z}$ be the set of all concrete zones of $\mathcal{D}$. We say that $\mathcal{D}$ *realizes* the abstract Euler diagram description $D = (L, Z)$ if $|\mathcal{Z}| = |Z|$ and for any zone $z \in Z$ there is a concrete zone in $\mathcal{Z}$ with label set z in $\mathcal{D}$. If in addition to the abstract Euler diagram description $D = (L, Z)$ a function $area : Z \setminus \{L_\emptyset\} \to \mathbb{R}^+$ is provided, then an Euler diagram $\mathcal{D}$ that realizes D is called *area-proportional* with respect to *area* if the area covered in the plane by the concrete zone of $\mathcal{D}$ with label set z is $area(z)$ for any zone $z \in Z \setminus \{L_\emptyset\}$.

The Euler diagram depicted in Figure 1.3 realizes the abstract Euler diagram description $(\{a,b,c\},\{L_\emptyset,\{a\},\{c\},\{a,b\},\{a,c\},\{a,b,c\}\})$. The letter next to each curve is its label and denoted inside each minimal region is the label set of the concrete zone that the minimal region belongs to. Note that the concrete zone with label set $\{a\}$ consists of two minimal regions. Also note that contour with label c consists of two curves. The minimal region belonging to the concrete zone with label set $\{a, b\}$ and the minimal region belonging to the concrete zone with label set $\{a, b, c\}$ are neighbours, however, the minimal region belonging to the concrete zone with label set $\{a, b\}$ and the minimal region belonging to the concrete zone with the empty label set $L_\emptyset$ are not neighbours since the curves that describe their minimal regions intersect at exactly one point.

A general task that is considered in the context of Euler diagrams is, given an abstract Euler diagram description D, to generate an (area-proportional) Euler diagram that realizes D. Figure 1.4 shows five Euler diagrams. The first four Euler diagrams all realize the abstract Euler diagram description $(\{a,b\},\{L_\emptyset,\{a\},\{b\},\{a,b\}\})$, however, some of them do this in an arguably more aesthetically pleasing way than others. In order to describe the aesthetic properties of Euler diagrams in a more precise manner we now define several *well-formedness conditions* for Euler diagrams. Let therefore $\mathcal{D} = (\mathcal{C}, \mathcal{L})$ be an Euler diagram, $\mathcal{M}$ be the set of all minimal regions of $\mathcal{D}$ and $\mathcal{Z}$ be the set of all concrete zones of $\mathcal{D}$. If each contour of $\mathcal{D}$ consists of exactly one curve, then $\mathcal{D}$ possesses the *unique curve labels* property. This property is not satisfied by the Euler diagram in Figure 1.4b since there are two curve labeled b. If all curves in $\mathcal{C}$ are simple, then $\mathcal{D}$ possesses the *simple curves* property. This property is violated by the Euler diagram in Figure 1.4b,

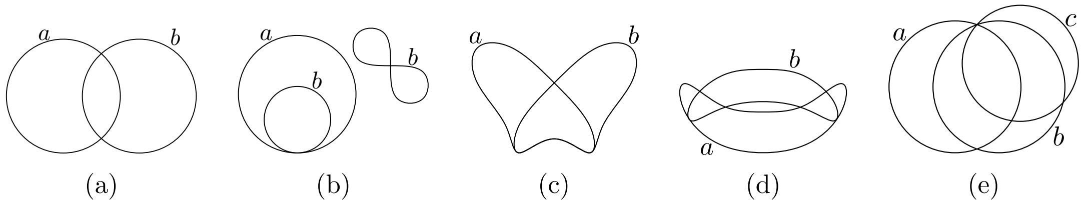


Figure 1.4: Differently well-formed Euler diagrams

because the right curve labeled b is self-intersecting. We stated above that every concrete zone of $\mathcal{D}$ is a union of minimal regions of $\mathcal{D}$. If for any concrete zone $P_z \in \mathcal{Z}$ there exists a minimal region $P_m \in \mathcal{M}$ such that the point sets P_z and P_m are equal, then $\mathcal{D}$ possesses the *connected concrete zones* property. In the Euler diagram in Figure 1.4d each of the concrete zones with the label sets $\{a\}$, $\{a, b\}$ and $L_\emptyset$ consists of multiple minimal regions, therefore the 'connected concrete zones' property is not satisfied. If for any two distinct curves $c, c' \in \mathcal{C}$ all connected components of $\text{image}(c) \cap \text{image}(c')$ consist of exactly one point, then $\mathcal{D}$ possesses the *no concurrency* property. If, however, for two distinct curves $c, c' \in \mathcal{C}$ there exists a connected component of $\text{image}(c) \cap \text{image}(c')$ consisting of more than one point, then c and c' are said to be *concurrent*, which is the case for the two curves of the Euler diagram in Figure 1.4c, which therefore does not satisfy the 'no concurrency' property. Euler diagram $\mathcal{D}$ possesses the *crossing* property if it possesses the 'no concurrency' property and if whenever two distinct curves in $c, c' \in \mathcal{C}$ intersect they *cross*, which means that for any connected component $p \in \text{image}(c) \cap \text{image}(c')$ and for any $\varepsilon > 0$ there exist distinct points $p_1 \in \text{interior}(c) \cap \text{image}(c')$, $p_2 \in \text{interior}(c') \cap \text{image}(c)$, $p_3 \in \text{exterior}(c) \cap \text{image}(c')$ and $p_4 \in \text{exterior}(c') \cap \text{image}(c)$ such that $||p_i - p||_2 \leq \varepsilon$ for $1 \leq i \leq 4$. The Euler diagram in Figure 1.4c does not possess the 'no concurrency' property and therefore does also not possess the 'crossing' property. In both the Euler diagram in Figure 1.4b and the Euler diagram in Figure 1.4d a curve labeled b 'touches' a curve labeled a, these Euler diagram therefore also do not satisfy the crossing property. If for any three distinct curves $c_1, c_2, c_3 \in \mathcal{C}$ the set of points $\text{image}(c_1) \cap \text{image}(c_2) \cap \text{image}(c_3)$ is empty, then $\mathcal{D}$ possesses the *no triple points* property, which is not the case for the Euler diagram in Figure 1.4e. Finally, let $m_\infty \in \mathcal{M}$ be the infinitely large minimal region that is exterior to all contours of $\mathcal{D}$ and therefore subset of the concrete zone with label set $L_\emptyset$. If every minimal region in $\mathcal{M} \setminus \{m_\infty\}$ is convex, then $\mathcal{D}$ possesses the *convex minimal regions* property. This property is violated by the Euler diagrams in Figure 1.4c and Figure 1.4d. The Euler diagram in Figure 1.4a satisfies all of our well-formedness conditions and could therefore by described as being well-formed.

Rodgers et al. [RZP12] provide reports on empirical studies to determine how well-formedness conditions affect user comprehension of Euler diagrams. The studies indicate that users perform particularly less well if the 'connected concrete zones' and the 'no concurrency' property are not satisfied. It is also implied that users deem the 'simple curves' property to be more important than the 'crossing' or the 'no triple points' property. Note, however, that the requirement of the 'no concurrency' property is a very restricting constraint for Euler diagram generating algorithms, as there exist even very simple abstract Euler diagram descriptions for which there exists no Euler diagram that satisfies the 'no concurrency' property, for example, the abstract Euler diagram description $D = (\{a, b\}, \{\{a, b\}\})$. Also note, that, on the other hand, if the 'unique curve labels' property is not required, it is easy to generate trivial Euler diagrams for any abstract Euler diagram description by drawing disjoint circles such that the intererior of each circle is a concrete zone, Figure 1.5 depicts such a trivial Euler Diagram. These Euler diagrams even satisfy the 'connected concrete zones', the 'convex minimal regions' and the 'simple curves' properties, however, they do not convey more information than a list of all intersections.

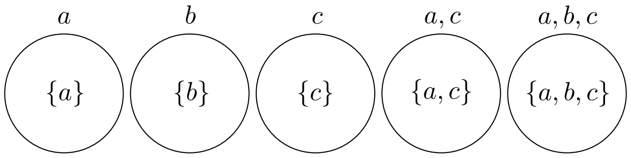


Figure 1.5: A trivial Euler diagram realizing $(\{a, b, c\}, \{\{a\}, \{b\}, \{c\}, \{a, c\}, \{a, b, c\}\})$

As mentioned in the beginning of this subsection, there exist many different definitions for Euler diagrams in the literature. This is the case mostly due to the fact that authors incorporate several of the above defined well-formedness properties into their respective definitions of Euler diagrams. In this thesis Euler diagrams are basically sets of labeled closed curves which, as stated above, is similarly handled in [FFH08] and [SRH11]. However, Chow [Cho07], for example, defined Euler diagrams as sets of simple and uniquely labeled curves such that there exists exactly one minimal region for every zone. Therefore one of Chow's Euler diagrams corresponds to one of our Euler diagrams that satisfies the 'simple curves', the 'unique curve labels' and the 'connected concrete zones' property. Similarly, there are many other definitions for Euler diagrams that differ more or less from both the definitions given in this thesis and the definitions of Chow.

Related work

We now present a small overview of work related to the generation of Euler diagrams. Observe that whether the area-proportional case is considered or not is a crucial distinction to make as having to respect an area specification for every zone is a severe restriction when generating Euler diagrams. The first three results consider non-area-proportional Euler diagrams wheras the latter two consider the area-proportional case. We have adapted each respective result to match our set of definitions.

Rodgers et al. [RZF08] present an algorithm together with an implementation that generates Euler diagrams satisfying the 'connected concrete zones' and the 'simple curves' property for any abstract Euler diagram description.

Flower et al. [FFH08] identify properties which classify an abstract Euler diagram description as *drawable*, which in our definition context means that there exists an Euler diagram that realizes the abstract Euler diagram description and satisfies the 'simple curves', the 'unique curve labels', the 'crossing', the 'no triple points' and the 'connected concrete zones' property. They present a high level algorithms that checks these properties and draws a diagram if the description is drawable. One step of this algorithm is related to solving a $\mathcal{NP}$-hard problem, the authors therefore provide an implementation of the algorithm for abstract Euler diagram descriptions limited to having up to four different labels.

Chow [Cho07] shows that deciding if there exists an Euler diagram that realizes some abstract Euler diagram description while satisfying the 'simple curves', the 'unique curve labels' and the 'connected concrete zones' property is an $\mathcal{NP}$-complete problem. However, deciding if such an Euler diagram exists while also satisfying the 'no concurrency' property or the 'no concurrency' and the 'no triple points' property are considered open problems.

Chow and Ruskey [CR04] present an algorithm together with an implementation that generates area-proportional Euler diagrams that satisfy the 'simple curves', the 'unique curve labels' and the 'connected concrete zones' property while realizing abstract Euler diagram descriptions with up to 3 different labels. In the case with 1 or 2 labels all curves describe circles. If the abstract Euler diagram has 3 different labels, then the curves describe rectangles. In this case the zone that contains all labels must be part of the descriptions zone set.

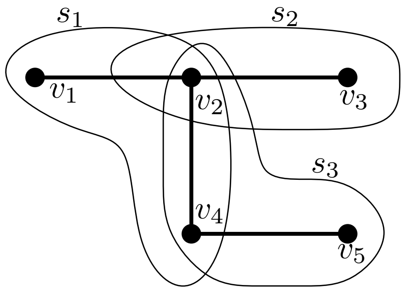


Figure 1.6: Illustration of a hypergraph together with a tree-support.

Stapleton et al. [SRH11] present a high level algorithm that allows the generation of an area-proportional Euler diagram for any abstract Euler diagram description. They represent the Euler diagram description as a graph and compute a set of cycles in this graph. The cycles are used to obtain the actual Euler diagram. For the 'unique curve labels', the 'simple curves', the 'no concurrency', the 'no triple points', the 'crossing' and the 'connected concrete zones' property they present constraints for the computed cycles. If the cycles comply with a specified set of constraints, then the generated diagram satisfies the corresponding well-formedness properties. However, depending on the chosen set of constraints and the abstract Euler diagram description, there may not exists a set of cycles that complies with the set of constraints so that no according Euler diagram can be generated. The algorithm's runtime complexity is not considered.

1.1.3 Hypergraphs

In this subsection we introduce definitions and notation related to hypergraphs and provide an overview of some results related to our work. First, we recall the definitions from the introduction (Chapter 1) in order to keep this section self-contained. A hypergraph H is an ordered pair (V, S), where V is a set of vertices and S is a set of hyperedges, which are subsets of V. A support for hypergraph $H = (V, S)$ is a graph G with $\mathrm{V}(G) = V$ such that for every hyperedge $s \in S$ the subgraph of G induced by the vertex set s is connected. The support G is called tree-support if G is a tree and hypergraph H is called tree-hypergraph if there exists a tree-support for H. Every support for H is a subgraph of the complete graph K with vertex set $\mathrm{V}(K) = V$. If a weight function $w : \mathrm{E}(K) \to \mathbb{R}$ is provided that assigns a real weight to every edge of K, then a tree-support for H is called *minimum* tree-support regarding w if its weight regarding w is minimal amongst the weight of all tree-supports for H. Hypergraphs are often illustrated by assigning a point of the plane to each vertex and a simple closed curve to each hyperedge such that the interior of each hyperedge curve contains the points assigned to the vertices contained in the hyperedge and the exterior of each hyperedges curve contains the points assigned to the hypergraphs remaining vertices. Figure 1.6 illustrates hypergraph $H = (\{v_1, \ldots, v_5\}, \{s_1, s_2, s_3\})$ with $s_1 = \{v_1, v_2, v_4\}$, $s_2 = \{v_2, v_3\}$ and $s_3 = \{v_2, v_4, v_5\}$ and also depicts the drawing of a tree-support for H.

Related work

It is possible to decide in linear time if a given hypergraph has a cycle-support [BKM$^+$10] or a path-support [BKM$^+$10], where a cycle-support is defined as a connected support such that the degree of all its vertices is 2 and a path-support is a connected support such that there exist 2 vertices with degree 1 and the degree of the remaining vertices is 2. Deciding if a hypergraph has a tree-support [JP87] or a cactus-support [BCPS11], where a cactus-support is described as a connected support such that each of its edges is contained in at most one cycle, is possible in polynomial time. It can also be decided in polynomial time if for a hypergraph $H = (V, S)$ and a value d_v for every vertex $v \in V$ there exists a tree-support T for H such that the degree of v in T is at most d_v [BKM+10]. On the other hand, it is $\mathcal{NP}$-complete to decide if a given hypergraph has a planar [JP87] or 2-outerplanar support [BKM+10]. Deciding if a hypergraph $H = (V, S)$ has a planar or outerplanar support is possible in polynomial time if H is closed under intersection and differences, which means that for any two hyperedges $s_1, s_2 \in S$ with $s_1 \cap s_2 \neq \emptyset$, $s_1 \setminus s_2 \neq \emptyset$ and $s_2 \setminus s_1 \neq \emptyset$ both the containments $s_1 \cap s_2 \in S \cup \{\{v\} \subseteq V \mid v \in V\}$ and $s_1 \setminus s_2 \in S \cup \{\{v\} \subseteq V \mid v \in V\}$ are fulfilled [BCPS11]. However, whether there exists a polynomial time algorithm to decide if a hypergraph that is not closed under intersection and differences has an outerplanar support or not is an open problem.

Korach and Stern [KS03] presented an algorithm for computing minimum tree-supports for hypergraphs in $O(n^4m^2)$ time, n being the number of vertices and m being the number of hyperedges of the input hypergraphs. They define the *intersection-closure* of a hypergraph's hyperedge set S to be the set of vertex sets $\text{cl}(S) = \{s \subseteq V \mid \exists S' \subseteq S : s = \bigcap_{s' \in S'} s'\}$ and *necessary* edges to be the elements in a hyperedge set's intersection-closure that have cardinality 2. They observe that if the hypergraph's hyperedges are restricted to have cardinality at most 3 the necessary edges are element of the the edge set of every minimum tree-support. They devise an algorithm for this restricted case that computes all necessary edges. If the obtained edge set is not yet the edge set of a tree, they add edges according to their weight. The algorithm for general hypergraphs utilizes a contraction idea. The input hypergraph is successively contracted until the hypergraph's vertex set contains at most 3 vertices. After applying the algorithm for the restricted case the contraction is reversed in a backwards process to obtain a minimum tree-support.

1.2 Our contribution and thesis overview

In this thesis we present a framework for efficient algorithms that for the class of abstract Euler diagram descriptions whose labeled hypergraphs are tree-hypergraphs generate area-proportional Euler diagrams that satisfy the 'simple curves', the 'connected concrete zones', the 'unique curve labels' and the 'convex minimal regions' property and also have a minimal total number of concurrent curves between the neighbour pairs of minimal regions that belong to concrete zones with non-empty label sets. Labeled hypergraphs are introduced in Section 2.1 and act as an interface between abstract Euler diagram descriptions and hypergraphs. In Section 2.2 we present an algorithm that computes minimum tree-supports for labeled hypergraphs and in Section 3.1 we present the aforementioned framework for generating Euler diagrams. To follow up, in Section 3.2 we then present several ideas to improve the aesthetic aspects of the generated Euler diagrams.

We also provide an algorithm that computes minimum tree-supports for hypergraphs. This algorithm is presented in Section 2.3. It converts a given hypergraph into a labeled hypergraph, applies the algorithm presented in Section 2.2 and then transforms the result into a minimum tree-support for the original hypergraph. The algorithm's runtime complexity is $O(n^2(m + \log n))$, where n is the number of vertices and m is the number of hyperedges. It therefore improves upon the algorithm presented in [KS03], which has runtime $O(n^4m^2)$.

We conclude our work in Chapter 4 and also point at some future directions and open problems.

2. Computing minimum tree-supports for (labeled) hypergraphs

In this chapter we first define labeled hypergraphs, which serve as an interface between abstract Euler diagram descriptions and hypergraphs. We then motivate why it is advantageous to be able to compute minimum tree-supports for labeled hypergraphs and provide an algorithm that accomplishes this task. Finally we provide an extended version of the algorithm that allows us to also compute minimum tree-supports for general hypergraphs and show that this approach improves upon the one presented in [KS03] in terms of time complexity.

2.1 Labeled hypergraphs and a motivation for minimum tree-supports

In this section we first introduce labeled hypergraphs, which are hypergraphs that relate to specific abstract Euler diagram descriptions and then motivate why one should be concerned with computing minimum tree-supports for these kind of hypergraphs.

Hypergraphs have been related to the concept of Euler diagrams in the past. Chow [Cho07] defined a function $\varphi(S) = (X(S), S \setminus \{\emptyset\})$, which maps a set system to a hypergraph. A set system S is defined as a set of subsets of $X(S)$, which is a set of items. The function φ is used to compare hyperedge-based and vertex-based Venn diagrams, which were introduced by Johnson and Pollak [JP87], to Chow's version of Euler diagrams. In a similar manner, we define $\mathrm{H}(D) = (Z \setminus \{L_\emptyset\}, \mathrm{S}(L))$ to be the *labeled* hypergraph for the abstract Euler diagram description $D = (L, Z)$ with $\mathrm{S}(L) = \{s \subseteq Z \setminus \{L_\emptyset\} \mid \exists l \in L : s = \mathrm{s}(l)\}$ and $\mathrm{s}(l) = \{z \in Z \setminus \{L_\emptyset\} \mid l \in z\}$. For example, Figure 2.1 illustrates the labeled hypergraph for the abstract Euler diagram description $(\{a, b, c\}, \{\{a\}, \{a, b\}, \{a, b, c\}\})$. For the remainder of this section, let $D = (L, Z)$ be an abstract Euler diagram description and let $\mathrm{H}(D) = (Z \setminus \{L_\emptyset\}, \mathrm{S}(L))$ be the labeled hypergraph for D.

In Chapter 3 we provide a framework for algorithms that given a tree T with vertex set $\mathrm{V}(T) = Z \setminus \{L_\emptyset\}$ produce an Euler diagram $\mathcal{D}$ that realizes D such that for any zone $z \in Z$ there exists a concrete zone of $\mathcal{D}$ with label set z that consists of exactly one minimal region. Furthermore, two minimal regions m_1 and m_2 which belong to the concrete zones with label set z_1 and z_2 respectively are neighbours in $\mathcal{D}$ if and only if there exists an edge $\{z_1, z_2\} \in \mathrm{E}(T)$ or if either $z_1 = L_\emptyset$ or $z_2 = L_\emptyset$.

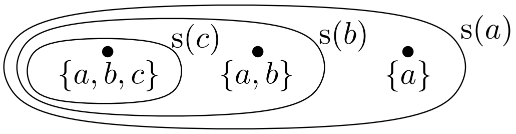


Figure 2.1: Illustration of a labeled hypergraph.

Let $l \in L$ be a label. If the input tree T is a tree-support for $\mathrm{H}(D)$, then the subgraph of T induced by $\mathrm{s}(l)$ is connected and therefore the minimal regions that correspond to concrete zones whose label set contains l form a succession of neighbours in $\mathcal{D}$ implying that one closed curve labeled l is sufficient in the curve set of $\mathcal{D}$. Therefore $\mathcal{D}$ can easily possess the 'unique curve labels' property if the algorithm used to compute $\mathcal{D}$ is implemented accordingly. This motivates computing tree-supports for the labeled hypergraphs.

Chow [Cho07] defined Euler dual graphs for Euler diagrams, in which the vertex set contains a vertex for every minimal region and the edge set contains an edge for each pair of vertices whose corresponding minimal regions are neighbours. Assigned to each edge $\{v_1, v_2\}$ is a label set containing the label of each curve that separates the minimal regions corresponding to v_1 and v_2 and therefore causes them to be neighbours. For example, let v_1 be the vertex corresponding to a minimal region of the concrete zone with label set $\{a, b\}$, let v_2 be the vertex corresponding to a minimal region of the concrete zone with label set $\{a, b, c, d\}$ and let v_1 and v_2 be neighbours. The edge $\{v_1, v_2\}$ then has the label set $\{c, d\}$ attached to it. Chow stated that an Euler diagram possesses the 'no concurrency' property if and only if the label set attached to each edge contains exactly one element (Proposition 4.4.1 in [Cho07]).

In order to formalize this idea, we define the *concurrency* weight function $\mathrm{w}(D) : \mathrm{E}(\mathrm{K}(D)) \to \mathbb{N}_0$ for $D = (L, Z)$ as $\mathrm{w}(D)(\{z, z'\}) = |(z \cup z') \setminus (z \cap z')|$, where $\mathrm{K}(D)$ is the complete graph with vertex set $Z \setminus \{L_\emptyset\}$. As seen in Section 1.1.2, there exist even very simple abstract Euler diagram descriptions for which there exists no Euler diagram that realizes them while possessing the 'no concurrency' property, which makes the requirement of this property a very restricting constraint for Euler diagram generating algorithms. Observe, however, that $\mathrm{w}(D)$, in addition to enabling us to verify whether $\mathcal{D}$ possesses the 'no concurrency' property or not, provides more information. Let $z, z' \in Z$ be the label sets of two concrete zones of $\mathcal{D}$ whose corresponding minimal regions are neighbours in $\mathcal{D}$. If $\mathrm{w}(D)(\{z, z'\}) > 1$, the value $\mathrm{w}(D)(\{z, z'\})$ is the number of concurrent curves whose images separate the two minimal regions in $\mathcal{D}$. An implication of this is that minimizing the weight of T regarding $\mathrm{w}(D)$ minimizes the number of concurrent curves between each neighbour pair of minimal regions in $\mathcal{D}$ that belong to concrete zones with non-empty label sets, thereby motivating the computation of minimum tree-supports for labeled hypergraphs.

2.2 Computing minimum tree-supports for labeled hypergraphs

In this section we present Algorithm MTS-labeled, which computes minimum tree-supports for labeled hypergraphs. The input of Algorithm MTS-labeled is a labeled hypergraph $\mathrm{H}(D) = (Z, \mathrm{S}(L))$ for some abstract Euler diagram description $D = (L, Z \cup \{L_\emptyset\})$. Recall that this implies that every vertex $z \in Z$ is a subset of the set of labels L and that there is a hyperedge $\mathrm{s}(l) \in \mathrm{S}(L)$ for every $l \in L$ that contains all vertices in Z containing l. Note that we explicitly chose to denote the set of zones of D as $Z \cup \{L_\emptyset\}$ so that we can assume that $L_\emptyset \notin Z$ and so that the vertex set of its labeled hypergraph $\mathrm{H}(D)$ can be denoted as Z, resulting in a more convenient notation for this section.

```
Algorithm: MTS-labeled
  input  : labeled hypergraph $\mathrm{H}(D) = (Z, \mathrm{S}(L))$ for abstract Euler diagram
           description $D = (L, Z \cup \{L_\emptyset\})$,
           weight function $w : E \to \mathbb{R}$ where $E$ is the edge set of a complete graph with
           vertex set $Z$
  output: minimum tree-support for $\mathrm{H}(D)$ regarding $w$ or infeasibility notification

  Feasibility Check
if not FeasibilityTreeSupport($\mathrm{H}(D)$) then
    return 'not feasible'

  Computing edge cardinality sets
for $i = 0$ to $|\mathrm{S}(L)| - 1$ do
    $\mathcal{E}_i = \emptyset$
for each edge $\{z, z'\} \in E$ do
    $C(D)(\{z, z'\}) = |z \cap z'|$
    $\mathcal{E}_{C(D)(\{z,z'\})} = \mathcal{E}_{C(D)(\{z,z'\})} \cup \{z, z'\}$

  Computing minimum tree-support
$F = \emptyset$
for each vertex $z \in Z$ do
    MakeSet($z$)
for $i = |\mathrm{S}(L)| - 1$ to $0$ do
    sort edges in $\mathcal{E}_i$ into nondecreasing order by weight $w$
    for each edge $\{z, z'\} \in \mathcal{E}_i$ taken in nondecreasing order by weight $w$ do
        if FindSet($z$) $\neq$ FindSet($z'$) then
            $F = F \cup \{\{z, z'\}\}$
            Union($z, z'$)
return $T = (Z, F)$
```

```
Algorithm: MST-Kruskal [CLRS09]
  input  : graph $G = (V, E)$,
           weight $w(e)$ for every edge $e \in E$
  output: minimum spanning-tree of $G$
$F = \emptyset$
for each vertex $v \in V$ do
    MakeSet($v$)
sort edges of $E$ into nondecreasing order by weight $w$
for each edge $\{u, v\} \in E$, taken in nondecreasing order by weight do
    if FindSet($u$) $\neq$ FindSet($v$) then
        $F = F \cup \{\{u, v\}\}$
        Union($u, v$)
return $T = (V, F)$
```

First Algorithm MTS-labeled performs a feasibility check to verify whether there actually exists a tree-support for $\mathrm{H}(D)$ or not. Johnson and Pollak [JP87] suggest that this can be done efficiently for any hypergraph $H = (V, S)$ by performing an acyclicity check on the *dual* hypergraph $H^* = (V^*, S^*)$ of H, which is a hypergraph with vertex set $V^* = S$ and edge set $S^* = \{S' \subseteq S \mid \exists v \in V : v \in \bigcap_{s \in S'} s,\ \forall s \in S \setminus S' : v \notin s\}$. Acyclicity is a property of hypergraphs for which Beeri, Fagin, Maier and Yannakakis [BFMY83] present a definition as well as a total of twelve equivalent characterizations. Johnson and Pollak derive from one of these characterizations that the dual of a tree-hypergraph is an acyclic hypergraph. For a linear time acyclicity check for hypergraphs they refer to the work of Tarjan and Yannakakis [TY84].

Recall that every tree-support for $\mathrm{H}(D)$ is a subgraph of the complete graph with vertex set Z. We now define a special kind of complete graph. The *zone skeleton* of an abstract Euler diagram description $D' = (L', Z')$ is an edge-weighted complete graph G' with vertex set $\mathrm{V}(G') = Z' \setminus \{L_\emptyset\}$ and the weight function for the edge weights of G' is the *cardinality* weight function $\mathrm{C}(D') : \mathrm{E}(G') \to \mathbb{N}_0$ for D' defined as $\mathrm{C}(D')(\{z, z'\}) = |z \cap z'|$. The value $\mathrm{C}(D')(\{z, z'\})$ is called *cardinality* of edge $\{z, z'\}$ for any two distinct zones $z, z' \in Z' \setminus \{L_\emptyset\}$. Let $G = (Z, E)$ be the zone skeleton for D. After performing the acyclicity check Algorithm MTS-labeled computes the cardinality $\mathrm{C}(D)(e)$ for each edge of E and then partitions the edges of E into $|\mathrm{S}(L)|$ subsets of E, one for each possible cardinality.

The remainder of Algorithm MTS-labeled is very similar to the popular algorithm of Kruskal for computing minimum spanning-trees of graphs. The pseudocode version Algorithm MST-Kruskal of this algorithm is taken directly from [CLRS09]. In the following, we first provide a brief description of Kruskals algorithm and then describe how we adopted its approach in our algorithm. The version of Kruskals algorithm presented in [CLRS09] uses the disjoint-set forests data-structure, which provides the means to efficiently maintain disjoint sets of elements. $\mathrm{MakeSet}(x)$ creates a new set containing element x, $\mathrm{FindSet}(x)$ returns a handle to the set which contains x and $\mathrm{Union}(x, y)$ merges the sets containing element x and element y. Given an edge-weighted graph $G' = (V', E')$ Algorithm MST-Kruskal creates a minimum spanning-tree of G'. The algorithm grows the edge set of a forest with vertex set V' by successively adding edges from E'. Every edge in E' is considered in increasing order by its weight and added to the edge set of the forest if the vertices incident to the edge are in distinct connected components of the forest. This can be efficiently verified by using the disjoint-set forest data-structure to maintain one set per connected component of the forest. The total runtime complexity is $O(|E'|\, log\, |V'|)$. For a more detailed explanation of both Kruskals algorithm and the disjoint-set forest data structure we refer to [CLRS09].

In Algorithm MTS-labeled we essentially apply Algorithm MST-Kruskal to G but we alter the order in which the edges in E are considered: We perform $|\mathrm{S}(L)|$ *hierarchy steps*, in each of which only edges of a certain cardinality are considered. In the first hierarchy step only the edges with the potentially highest cardinality $|\mathrm{S}(L)| - 1$ are considered, in the second hierarchy step the edges with cardinality $|\mathrm{S}(L)| - 2$ are considered and so forth. Figure 2.2 shows the three hierarchy steps of the application of Algorithm MTS-labeled to the hypergraph $H(D) = (Z, \mathrm{S}(L))$ with $Z = \{\{a\}, \{a, b, c\}, \{a, b, c, d\}, \{c, d, e\}, \{c, d, e, f\}\}$ and $L = \{a, b, c, d, e, f\}$ for the abstract Euler diagram description $D = (L, Z)$. The continuous edges are the edges already added to F and the numbers next to each edge are the edge's cardinality/weight. The idea for the approach used in Algorithm MTS-labeled is based on Korach and Stern's [KS03] observation that every element of a hypergraph's hyperedge set's intersection-closure induces a connected subgraph in every tree-support for this hypergraph (Theorem 5.2.1 in [KS03]). This observation was also utilized by Buchin et al. [BKM+10], who compute tree-supports with bounded degrees (see Section 1.1.3).

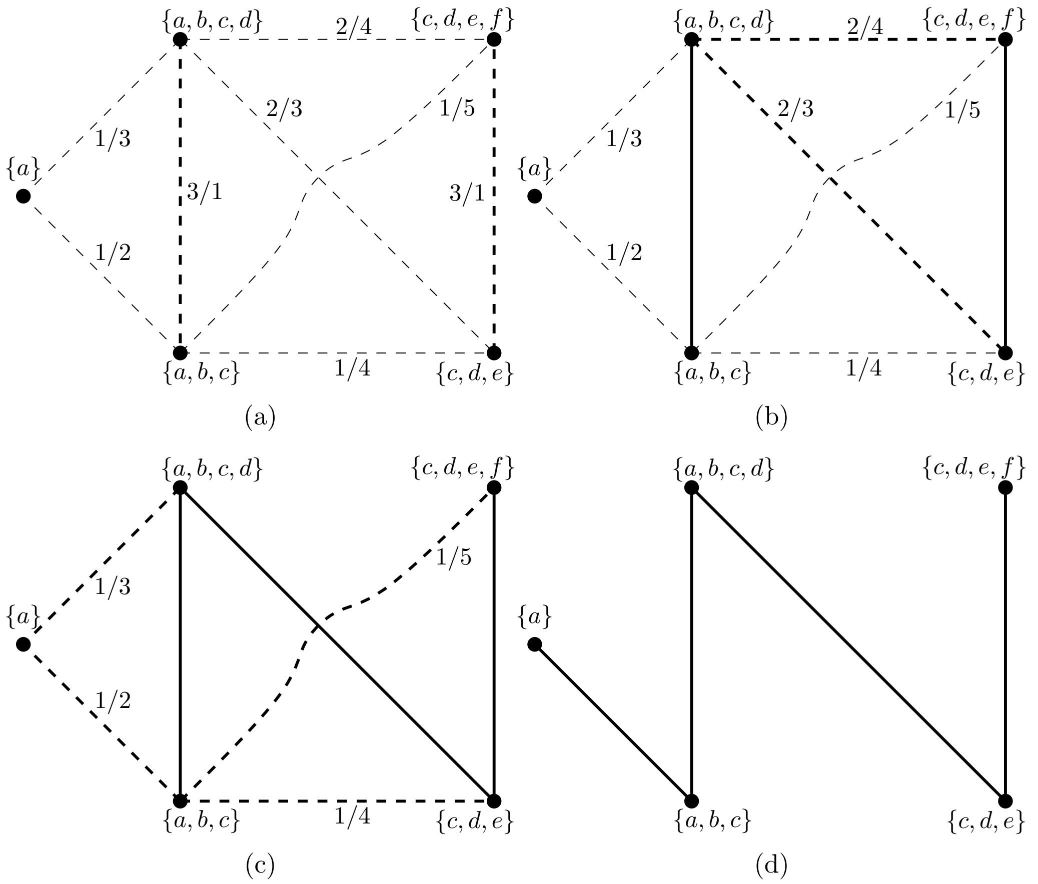
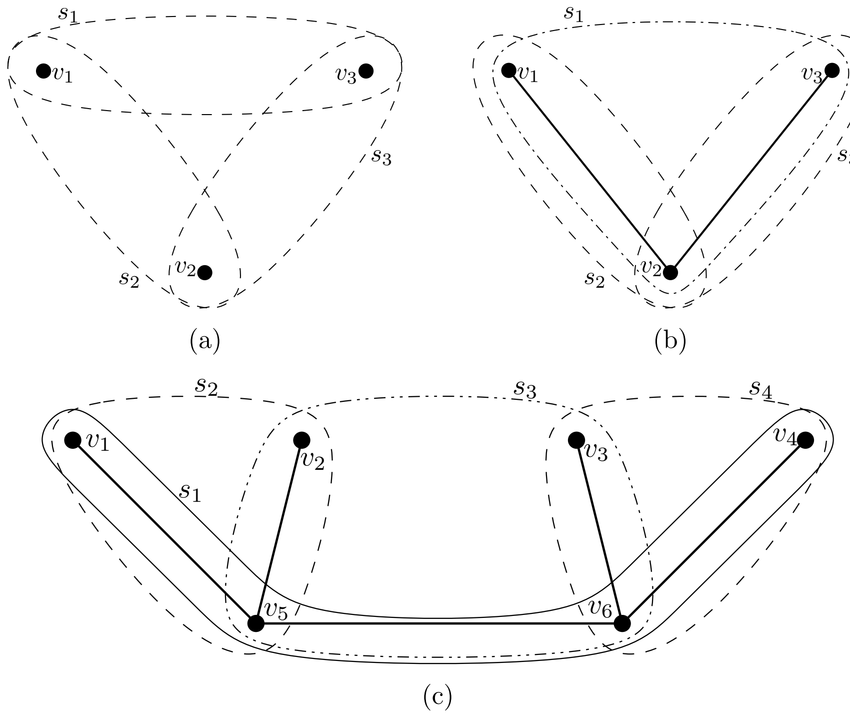


Figure 2.2: Exemplary application of Algorithm MTS-labeled.

We now present a few auxiliary results that allow us to later prove the correctness of Algorithm MTS-labeled.

First we introduce a few definitions. Let $H = (V, S)$ be a hypergraph, let $S' \subseteq S$ be a set of hyperedges, let G_H be a graph with $\mathrm{V}(G_H) = V$ and let be $e = \{v, u\} \in \mathrm{E}(G_H)$. We say that e *actively supports* S' if $\{v, u\} \subseteq s'$ for any hyperedge $s' \in S'$. Note that for our labeled hypergraph $\mathrm{H}(D)$, its zone skeleton G and some support G' for $\mathrm{H}(D)$ this definition conveys information about an edge's cardinality. If edge $e' \in \mathrm{E}(G')$ actively supports hyperedge set $\mathrm{S}(L')$ for some label set $L' \subseteq L$ the cardinality $\mathrm{C}(D)(e')$ of e' is at least $|L'|$. We say that the pairwise distinct vertices $v_1, \ldots, v_t \in V, t \geq 3$ have a *circular relationship* in H if there exist pairwise distinct hyperedge sets $S_1, \ldots, S_t \subseteq S$ with $v_1 \in \bigcap_{s \in S_1 \cup S_2} s, v_2 \in \bigcap_{s \in S_2 \cup S_3} s, \ldots, v_{t-1} \in \bigcap_{s \in S_{t-1} \cup S_t} s$ and $v_t \in \bigcap_{s \in S_t \cup S_1} s$. In this case the hyperedge sets $S_1, \ldots, S_t$ are said to *cause* the circular relationship of $v_1, \ldots, v_t$. Note that for a specific set of vertices that have a circular relationship in a hypergraph there may exist multiple sets of hyperedge sets that cause this vertex set to have a circular relationship. Figure 2.3 illustrates some hypergraphs whose vertices have circular relationships. In the hypergraph displayed in Figure 2.3a the vertices v_1, v_2, v_3 have a circular relationship caused by the hyperedge sets $S_i = \{s_i\}$, $1 \leq i \leq 3$ because $v_1 \in \bigcap_{s \in S_1 \cup S_2} s, v_2 \in \bigcap_{s \in S_2 \cup S_3} s$ and $v_3 \in \bigcap_{s \in S_3 \cup S_1} s$. There is no tree-support for this hypergraph because every tree with vertex-set $\{v_1, v_2, v_3\}$ fails to support one of the hyperedges s_1, s_2, s_3. The hypergraph displayed in Figure 2.3b is identical to the one in Figure 2.3a except for hyperedge s_1, which now has vertex-set $\{v_1, v_2, v_3\}$. As before vertices v_1, v_2, v_3 have a circular relationship due to their inclusion in intersections of the hyperedge sets $S_i = \{s_i\}$, $1 \leq i \leq 3$, but this time the depicted tree with edge-set $\{\{v_1, v_2\}, \{v_2, v_3\}\}$ is a tree-support. Figure 2.3c depicts a more complex hypergraph.



(a)

(b)

(c)

Figure 2.3: Hypergraphs with circular relationships.

The vertices of each the the four vertex sets $\{v_1, v_2, v_3, v_4\}$, $\{v_1, v_2, v_5\}$, $\{v_2, v_3, v_5, v_6\}$ and $\{v_3, v_4, v_6\}$ have a circular relationship. Observe how the displayed tree-support does not connect any of the vertices in $\{v_1, v_2, v_3, v_4\}$ directly.

The following Lemma characterizes the edges of paths in tree-supports between vertices that are part of a circular relationship and allows us to derive information about the cardinality of these edges.

Lemma 1. *Let $H = (V, S)$ be a tree-hypergraph, let $T = (V, E)$ be a tree-support for H and let $v_1, \ldots, v_t \in V$ be vertices that have a circular relationship in H caused by $S_1, \ldots, S_t$ with $v_1 \in \bigcap_{s \in S_1 \cup S_2} s, v_2 \in \bigcap_{s \in S_2 \cup S_3} s, \ldots, v_{t-1} \in \bigcap_{s \in S_{t-1} \cup S_t} s$ and $v_t \in \bigcap_{s \in S_t \cup S_1} s$. For any two distinct vertices v_i, v_j with $1 \leq i, j \leq t$ every edge of path $\mathrm{p}(v_i, v_j, T)$ actively supports at least two of the hyperedge sets $S_1, \ldots, S_t$.*

Proof. Let without loss of generality $1 \leq i, j \leq t$ be two indices with $i < j$ and $e = \{u, v\}$ be an edge of path $\mathrm{p}(v_i, v_j, T)$. Since T is a tree, the graph $G = (V, E \backslash \{e\})$ has two connected components $C_1 = (V_1, E_1)$ and $C_2 = (V_2, E_2)$ with $V = V_1 \cup V_2$ and $E \setminus \{e\} = E_1 \cup E_2$ and we can find an index $i \leq p < j$ with $v_p \in V_1$ and $v_{p+1} \in V_2$. The vertices v_p and v_{p+1} are part of the circular relationship, it therefore follows that $\{v_p, v_{p+1}\} \subseteq s$ for every hyperedge $s \in S_{p+1}$. Since T is a tree support, every edge of path $\mathrm{p}(v_p, v_{p+1}, T)$ must actively support S_{p+1} and since any path in T between a vertex of V_1 and V_2 contains e, it follows that e actively supports S_{p+1}.

Since $v_i, v_p \in V_1$ and $v_{p+1}, v_j \in V_2$, there exist indices $1 \leq a, b \leq t$ such that $v_a \in V_2$ and $v_b \in V_1$ with either $j \leq a < t$ and $b = a + 1$, $a = t$ and $b = 1$ or $1 \leq a < i$ and $b = a + 1$. In any case, the circular relation in H implies that $\{v_a, v_b\} \subseteq s$ for every hyperedge $s \in S_b$. Therefore every edge of the path $\mathrm{p}(v_a, v_b, T)$ actively supports S_b. Edge e is one of the edges of path $\mathrm{p}(v_a, v_b, T)$ because $v_a \in V_2$ and $v_b \in V_1$, which concludes the proof since $b \notin \{i + 1, i + 2, \ldots, j\}$ and therefore $b \neq p + 1$. □

The next results allows us to exchange a subtree of a spanning-tree of some graph G with a specific kind of subtree of G to obtain another spanning-tree of G.

Lemma 2. *Let $G = (V, E)$ be a graph, let $T = (V, E_0 \cup E_1)$ be a spanning-tree of G with $E_0 \cap E_1 = \emptyset$ and let $T_1 = (V_1, E_1)$ be the subtree of T induced by E_1. For any subtree $T_1' = (V_1, E_1')$ of G the graph $T' = (V, E_0 \cup E_1')$ is a spanning-tree of G.*

Proof. Assume that there exists a cycle in T'. Both E_0 and E_1' are edge sets of forests, it therefore follows that every cycle in T' must be constituted by edges from both E_0 and E_1'. Since $T_1' = (V_1, E_1')$ is a subtree of G with vertex set V_1, there must exists a cycle in T' constituted by the edge set of a path $\mathrm{p}(v, v', T_1')$ containing edges exclusively from E_1' and the edge set of a path $\mathrm{p}(v', v, T)$ containing edges exclusively from E_0, with $v, v' \in V_1$. But since T_1 is also a subtree of G with vertex set V_1, the edge sets of the paths $\mathrm{p}(v, v', T_1)$ and $\mathrm{p}(v', v, T)$ also constitute a cycle, contradicting to T being a tree, since T_1 is a subtree of T. Therefore, there exists no cycle in T' and since $\mathrm{V}(T_1) = \mathrm{V}(T_1') = V_1$, it follows that $|E_1| = |\mathrm{E}(T_1)| = |\mathrm{E}(T_1')| = |E_1'|$ and therefore that T' is also connected and thus, T' is a tree. □

We now extends the result above to also be able to exchange multiple subtrees of a spanning-tree of a graph at once.

Corollary 1. *Let $G = (V, E)$ be a graph, let $T = (V, E_0 \cup E_1 \cup \ldots \cup E_t)$ be a spanning-tree of G with $E_i \cap E_j = \emptyset$ for every $0 \leq i \leq t$ and $0 \leq j \leq t$ with $i \neq j$ and let $T_k = (V_k, E_k)$ be the subtree of T induced by E_k for every $1 \leq k \leq t$. For any forest consisting of trees $T_1' = (V_1, E_1'), \ldots, T_t' = (V_t, E_t')$ the graph $T' = (V, E_0 \cup E_1' \cup \ldots \cup E_t')$ is a spanning-tree of G.*

Proof. The graphs G, T, T_1 and T_1' meet the preconditions of Lemma 2, therefore, the graph $\tilde{T_1} = (V, E_0 \cup E_1' \cup E_2 \cup \ldots \cup E_t)$ is a spanning-tree of G. The tree T_2 is a subtree of $\tilde{T_1}$ induced by E_2, therefore, the graphs G, $\tilde{T_1}$, T_2 and T_2' now meet the preconditions of Lemma 2 and it follows that the graph $\tilde{T_2} = (V, E_0 \cup E_1' \cup E_2' \cup E_3 \cup \ldots \cup E_t)$ is also a spanning-tree of G.

This process can be repeated to, analogously, exchange the edge sets $E_3, \ldots, E_t$ of the remaining trees $T_3, \ldots, T_t$ with the edge sets $E_3', \ldots, E_t'$ of the trees $T_3', \ldots, T_t'$ to finally obtain the spanning-tree $T' = \tilde{T_t} = (V, E_0 \cup E_1' \cup \ldots \cup E_t')$ of G. □

We are now prepared to prove the correctness and analyze the time complexity of Algorithm MTS-labeled over the course of the next three Lemmata and the concluding Theorem. First, we introduce some notation that is used for the remainder of this section. Let $\mathrm{H}(D) = (Z, \mathrm{S}(L))$ be the labeled hypergraph for abstract Euler diagram description $D = (L, Z \cup \{L_\emptyset\})$ and let $G = (Z, E)$ be the zone skeleton of D. The labeled hypergraph $\mathrm{H}(D)$ and a weight function $w : E \to \mathbb{R}$ are the input for Algorithm MTS-labeled. Let T be the output of Algorithm MTS-labeled for the case that $\mathrm{H}(D)$ is a tree-hypergraph. We define $\lambda = |\mathrm{S}(L)|$. Let $\mathcal{E}_i \subseteq E$ be set of edges with cardinality i in E, let $E_i \subseteq E$ be the set of edges with cardinality at least i in E and let G_i be the subgraph of G induced by E_i for $0 \leq i \leq \lambda - 1$. Let finally $F_{\lambda - j}$ for $1 \leq j \leq \lambda$ be the edge set that is obtained by Algorithm MTS-labeled after hierarchy step j, therefore, after considering all edges with cardinality at least $\lambda - j$.

Lemma 3. *The set of edges $\mathcal{E}_{\lambda-1}$ is a subset of the edge set of any tree-support for* $\mathrm{H}(D)$.

Proof. If $\mathrm{H}(D)$ is not a tree-hypergraph or if $\mathcal{E}_{\lambda-1} = \emptyset$ then our proposition holds trivially. Let therefore $\mathrm{H}(D)$ be a tree-hypergraph, $\mathcal{E}_{\lambda-1} \neq \emptyset$ and $e = \{z, z'\} \in \mathcal{E}_{\lambda-1}$. The cardinality of e is $\mathrm{C}(D)(e) = |z \cap z'| = |L| - 1 = |\mathrm{S}(L)| - 1 = \lambda - 1$. Since the vertices in Z are pairwise distinct sets of labels from L, this can only be the case if either z or z' is the complete set of labels L. Let without loss of generality be $z = L$ and $z' = L \setminus \{l'\}$ with $l' \in L$ and let $\tilde{T}$ be a tree-support for $\mathrm{H}(D)$. Since $z, z' \in \bigcap_{l \in L \setminus \{l'\}} \mathrm{s}(l)$, every edge of $\mathrm{p}(z, z', \tilde{T})$ has to actively support $\mathrm{S}(L \setminus \{l'\})$ and therefore has cardinality $|L \setminus \{l'\}| = |L| - 1 = |\mathrm{S}(L)| - 1 = \lambda - 1$. As stated above, this is only true for edges incident to z, thus e is the only edge that actively supports $\mathrm{S}(L \setminus \{l'\})$ and therefore $e \in \mathrm{E}(\tilde{T})$. □

Lemma 4. *If* $\mathrm{H}(D)$ *is a tree-hypergraph, then Algorithm MTS-labeled computes a tree-support for* $\mathrm{H}(D)$.

Proof. Let $\mathrm{H}(D)$ be a tree-hypergraph. As described above, every tree-support for $\mathrm{H}(D)$ is a subgraph of its zone skeleton $G = (Z, E)$. In Algorithm MTS-labeled we grow the edge set F of a forest with vertex set Z by successively adding edges from $E = \bigcup_{0 \leq j \leq \lambda-1} \mathcal{E}_j$ and finally return the subgraph $T = (Z, F)$ of G. We now show by induction that after each hierarchy step of Algorithm MTS-labeled, if F is the current set of edges computed by the algorithm there exists a tree-support for $\mathrm{H}(D)$ whose edge set contains all edges of F.

For the induction base case we need to show that our hypothesis is true after the first hierarchy step, therefore, that there exists a tree-support for $\mathrm{H}(D)$ whose edge set contains all edges of $F_{\lambda-1}$, the edge set obtained by Algorithm MTS-labeled after considering $\mathcal{E}_{\lambda-1}$, which is the set of edges with the potentially highest cardinality. Since Lemma 3 states that the set of edges $\mathcal{E}_{\lambda-1}$ is a subset of the edge set of any tree-support for $\mathrm{H}(D)$, our proposition holds, thus concluding the base case. Observe that since $\mathcal{E}_{\lambda-1}$ is a subset of the edge set of every tree-support, the edges in $\mathcal{E}_{\lambda-1}$ can not constitute a cycle in G, therefore Algorithm MTS-labeled will in fact add all edges of $\mathcal{E}_{\lambda-1}$ to F, hence $F_{\lambda-1} = \mathcal{E}_{\lambda-1}$.

Now consider $j \in \mathbb{N}$ with $1 \leq j \leq \lambda - 1$ and assume that our induction hypothesis holds for the first j hierarchy steps, implying that there exists a tree-support whose edge set contains all edges in $F_{\lambda-j}$, which is the set of edges obtained by Algorithm MTS-labeled in the first j hierarchy steps. For the induction step we show there exists a tree-support for $\mathrm{H}(D)$ whose edge set contains all edges in $F_{\lambda-j-1}$, which is the edge set obtained by Algorithm MTS-labeled after hierarchy step $j + 1$, in which the edges of $\mathcal{E}_{\lambda-j-1}$ are considered. Let therefore $T_{\lambda-j}$ be a tree-support for $\mathrm{H}(D)$ with $F_{\lambda-j} \subseteq \mathrm{E}(T_{\lambda-j})$, which exists by our induction hypothesis. Recall that $G_{\lambda-j-1}$ is the subgraph of G induced by the edges of E with cardinality at least $\lambda - j - 1$. Let C be a connected component of $G_{\lambda-j-1}$ and let $T^C_{\lambda-j-1} = (\mathrm{V}(C), \mathrm{E}(C) \cap F_{\lambda-j-1})$ be the tree of the forest with edge set $F_{\lambda-j-1}$ that is a spanning-tree of C.

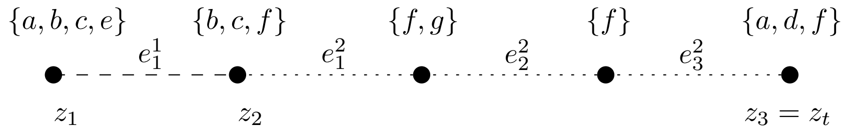
We will now show that the vertices of $\mathrm{V}(C)$ are connected by edges with cardinality at least $\lambda - j - 1$ in $T_{\lambda-j}$, which implies the existence of a subtree of $T_{\lambda-j}$ with vertex set exactly $\mathrm{V}(C)$, since C is a connected component of $G_{\lambda-j-1}$. Let therefore be $e = \{z, z'\} \in \mathrm{E}(T^C_{\lambda-j-1})$ be an edge of $T^C_{\lambda-j-1}$ and observe the path $\mathrm{p}(z, z', T_{\lambda-j})$. The cardinality of e is $\mathrm{C}(D)(e) \geq \lambda - j - 1$. Assume that there exists an edge $\tilde{e}$ in $\mathrm{p}(z, z', T_{\lambda-j})$ with $\mathrm{C}(D)(\tilde{e}) < \lambda - j - 1$. Then there exists a label $\tilde{l} \in z \cap z'$ for which $\tilde{e}$ does not actively support $\{s(\tilde{l})\}$, which contradicts to $T_{\lambda-j}$ being a tree-support for $\mathrm{H}(D)$.

Since the statement above is true for any connected component of $G_{\lambda-j-1}$, it follows that for every connected component of $G_{\lambda-j-1}$ and its corresponding tree in the forest with edge set $F_{\lambda-j-1}$ there exists a subtree of $T_{\lambda-j}$ whose vertex set is equal to the vertex set of the connected component of $G_{\lambda-j-1}$. The graphs G, $T_{\lambda-j}$ and its subtrees and the forest with edge set $F_{\lambda-j-1}$ therefore meet the preconditions of Corollary 1 and it follows that we can obtain spanning-tree $T_{\lambda-j-1}$ of G by substituting the edge sets of the subtrees of $T_{\lambda-j}$ by $F_{\lambda-j-1}$. Since the subtrees of $T_{\lambda-j}$ are spanning-trees of the connected components of $G_{\lambda-j-1}$, which is the subgraph of G induced by the edges of E with cardinality at least $\lambda-j-1$, it follows that $|\mathrm{E}(T_{\lambda-j}) \cap E_{\lambda-j-1}| = |F_{\lambda-j-1}|$ and we can therefore conclude that the edge set of $T_{\lambda-j-1}$ is $\mathrm{E}(T_{\lambda-j-1}) = (\mathrm{E}(T_{\lambda-j}) \setminus E_{\lambda-j-1}) \cup F_{\lambda-j-1}$. Note that $T^C_{\lambda-j-1}$ is a subtree of $T_{\lambda-j-1}$.

We know by the induction hypothesis that $\mathrm{E}(T_{\lambda-j}) \cap E_{\lambda-j} \supseteq F_{\lambda-j}$ and we know by construction that $\mathrm{E}(T_{\lambda-j-1}) \cap E_{\lambda-j} = F_{\lambda-j}$. Before we proceed to show that the tree $T_{\lambda-j-1}$ is a support for $\mathrm{H}(D)$, we show that $\mathrm{E}(T_{\lambda-j}) \cap E_{\lambda-j} = F_{\lambda-j}$, a result which we will use in the final part of the induction step. Let therefore $T^C_{\lambda-j}$ be the subtree of $T_{\lambda-j}$ with $\mathrm{V}(T^C_{\lambda-j}) = \mathrm{V}(C)$ and let $c \in \mathbb{N}$ be a natural number with $\lambda - j \leq c \leq \lambda - 1$. It suffices to show that there exists no edge with cardinality c in $\mathrm{E}(T^C_{\lambda-j})$ that is not an element of $\mathrm{E}(T^C_{\lambda-j-1})$ because this result is then true not only for C but for every connected component of $G_{\lambda-j-1}$. Assume that $e = \{z, z'\} \in \mathrm{E}(T^C_{\lambda-j})$ is an edge with this property. The subset of edges with cardinality at least c in $\mathrm{E}(T^C_{\lambda-j-1})$ together with e does not constitute a cycle because $T^C_{\lambda-j}$ is a tree whose edge set contains all these edges. This is a contradiction to $F_{\lambda-j}$ being the set of edges obtained in the first j hierarchy steps because Algorithm MTS-labeled would in this case have added e to F when considering the edge set $\mathcal{E}_{\mathrm{C}(D)(e)} = \mathcal{E}_c$.

We are now prepared to show that $T_{\lambda-j-1}$ is a support for $\mathrm{H}(D)$. Since C is a connected component of $G_{\lambda-j-1}$, it suffices to show that $T^C_{\lambda-j-1}$ is a support for hypergraph $H_C(D) = (\mathrm{V}(C), S_C(L))$ with $S_C(L) = \{s_c \subseteq \mathrm{V}(C) \mid \exists s \in \mathrm{S}(L) : s_c \neq \emptyset, s_c = s \cap \mathrm{V}(C)\}$ because then for any edge $e = \{z_1, z_2\} \in \mathrm{E}(T^C_{\lambda-j})$ the edges of the path $\mathrm{p}(z_1, z_2, T^C_{\lambda-j-1})$ actively support the hyperedge set $\mathrm{S}(z_1 \cap z_2)$. Since this then is true for the edge set of every tree of the forest with edge set $F_{\lambda-j-1}$ and since $T_{\lambda-j}$ is a support for $\mathrm{H}(D)$, it follows that for any two vertices $z'_1, z'_2 \in Z$ the edges of the path $\mathrm{p}(z'_1, z'_2, T_{\lambda-j-1})$ actively support $S(z'_1 \cap z'_2)$ and hence $T_{\lambda-j-1}$ is a support for $\mathrm{H}(D)$.

Assume that $T^C_{\lambda-j-1}$ is not a support for $H_C(D)$. Then there exist vertices $z_1, z_t \in \mathrm{V}(C)$ and not all edges of the path $\mathrm{p}(z_1, z_t, T^C_{\lambda-j-1})$ actively support hyperedge set $\mathrm{S}(L')$ with label set $L' = z_1 \cap z_t$. We now examine the path $\mathrm{p}(z_1, z_t, T^C_{\lambda-j-1})$ more carefully. Let therefore without loss of generality be $\mathrm{p}(z_1, z_t, T^C_{\lambda-j-1}) = (e^1_1, \ldots, e^1_{h_1}, e^2_1, \ldots, e^2_{h_2}, \ldots, e^{t-1}_1, \ldots, e^{t-1}_{h_{t-1}})$, where $L_1, \ldots, L_{t-1} \subseteq L$ are pairwise distinct label sets, in which for $1 \leq i \leq t-1$ the edge sequence $(e^i_1, \ldots, e^i_{h_i})$ is a path between $z_i \in \mathrm{V}(C)$ and $z_{i+1} \in \mathrm{V}(C)$ whose edges actively support the hyperedge set $\mathrm{S}(L_i)$ and $z_1 \in \bigcap_{s \in \mathrm{S}(L') \cup \mathrm{S}(L_1)} s$, $z_2 \in \bigcap_{s \in \mathrm{S}(L_1) \cup \mathrm{S}(L_2)} s$, $\ldots$, $z_{t-1} \in \bigcap_{s \in \mathrm{S}(L_{t-2}) \cup \mathrm{S}(L_{t-1})} s$ and $z_t \in \bigcap_{s \in \mathrm{S}(L_{t-1}) \cup \mathrm{S}(L')} s$. We may also assume without loss of generality that there exists no index $i \in \{1, \ldots, t-1\}$ with $\mathrm{S}(L') \subseteq \mathrm{S}(L_i)$. See Figure 2.4 for an example of such a path that illustrates the notation. The edges of path $\mathrm{p}(z_1, z_t, T^C_{\lambda-j})$ actively support $\mathrm{S}(L')$. Let e' be any edge of $\mathrm{p}(z_1, z_t, T^C_{\lambda-j})$ and let e be an edge of $\mathrm{p}(z_1, z_t, T^C_{\lambda-j-1})$ that has the lowest cardinality amongst all edges of $\mathrm{p}(z_1, z_t, T^C_{\lambda-j-1})$. The vertices $z_1, \ldots, z_t$ have a circular relationship in $\mathrm{H}(D)$ caused by $\mathrm{S}(L'), \mathrm{S}(L_1), \ldots, \mathrm{S}(L_{t-1})$, therefore, Lemma 1 states that e' actively supports at least two of the hyperedge sets $\mathrm{S}(L'), \mathrm{S}(L_1), \ldots, \mathrm{S}(L_{t-1})$, one of which has to be $\mathrm{S}(L')$. Let $i' \in \{1, \ldots, t-1\}$ be the index of the label set corresponding to another hyperedge set actively supported by e'. Since there exists no index $i \in \{1, \ldots, t-1\}$ with $\mathrm{S}(L') \subseteq \mathrm{S}(L_i)$, there exists at least one label $l \in L'$ for which $\mathrm{s}(l) \notin \mathrm{S}(L_{i'})$. Therefore the cardinality of e' is $\mathrm{C}(D)(e') \geq |\mathrm{S}(L') \cup \mathrm{S}(L_{i'})| > |\mathrm{S}(L_{i'})| \geq \mathrm{C}(D)(e) \geq |\mathrm{S}(L)| - j - 1 = \lambda - j - 1$, thus the



$L' = \{a\}$ $\quad$ $\mathrm{S}(L') = \{\mathrm{s}(a)\} = \{\{\{a,b,c,e\},\{a,d,f\}\}\}$

$L_1 = \{b,c\}$ $\quad$ $\mathrm{S}(L_1) = \{\mathrm{s}(b),\mathrm{s}(c)\} = \{\mathrm{s}(b)\} = \{\{\{a,b,c,e\},\{b,c,f\}\}\}$

$L_2 = \{f\}$ $\quad$ $\mathrm{S}(L_2) = \{\{\{b,c,f\},\{f,g\},\{f\},\{a,d,f\}\}\}$

Figure 2.4: Example for the in the proof of Lemma 4 supposedly existing vertices z_1, z_t and their path $\mathrm{p}(z_1, z_t, T^C_{\lambda-j-1})$ with $t = 3$.

cardinality of all edges of $\mathrm{p}(z_1, z_t, T^C_{\lambda-j})$ is at least $\lambda - j$. Since $\mathrm{E}(T_{\lambda-j}) \cap E_{\lambda-j} = F_{\lambda-j}$, this implies that $\mathrm{p}(z_1, z_t, T^C_{\lambda-j}) = \mathrm{p}(z_1, z_t, T^C_{\lambda-j-1})$, contradicting to $T_{\lambda-j}$ being a support for $\mathrm{H}(D)$ and it follows that $T^C_{\lambda-j-1}$ is indeed a support for $H_C(D)$ and therefore, that $T_{\lambda-j-1}$ is a tree-support for $\mathrm{H}(D)$, which concludes the induction step.

By the principle of induction, our hypothesis holds and there exists a tree-support whose edge set contains the final forest F_0, which is obtained after considering the edges of all cardinalities. Since Algorithm MTS-labeled considers every edge in E and adds it to F if the two vertices of the edges are in distinct connected components of the graph with the current edge set, F_0 is the edge set of a tree with vertex set Z, implying that the output $T = (Z, F_0)$ is a tree-support for $\mathrm{H}(D)$. □

Lemma 5. *If* $\mathrm{H}(D)$ *is a tree-hypergraph, then Algorithm MTS-labeled computes a minimum tree-support for* $\mathrm{H}(D)$ *regarding* w.

Proof. Let $\mathrm{H}(D)$ be a tree-hypergraph. We show by induction that after each hierarchy step of Algorithm MTS-labeled, if F is the current set of edges computed by the algorithm, there exists a minimum tree-support for $\mathrm{H}(D)$ regarding w whose edge set contains all edges of F. We will refer to this induction hypothesis as the *outer* induction hypothesis, as we will use a second induction proof as part of the induction step of the outer induction proof.

For the outer induction base case we need to show that our outer induction hypothesis is true after the first hierarchy step, therefore, that there exists a minimum tree-support for $\mathrm{H}(D)$ regarding w whose edge set contains all edges of $F_{\lambda-1}$, the edge set obtained by Algorithm MTS-labeled after considering $\mathcal{E}_{\lambda-1}$, which is the set of edges with the potentially highest cardinality. Like in the proof of Lemma 4, we utilize Lemma 3, which states that the set of edges $\mathcal{E}_{\lambda-1}$ is a subset of the edge set of any tree-support for $\mathrm{H}(D)$. This directly concludes the outer induction base case.

Now let be $1 \leq j \leq \lambda - 1$ and let our induction hypothesis be true for the first j hierarchy steps, therefore, there exists a minimum tree-support for $\mathrm{H}(D)$ regarding w whose edge set contains all edges in $F_{\lambda-j}$, which is the set of edges obtained by Algorithm MTS-labeled in the first j hierarchy steps. For the induction step we show there exists a minimum tree-support for $\mathrm{H}(D)$ regarding w whose edge set contains all edges in $F_{\lambda-j-1}$, which is the edge set obtained by Algorithm MTS-labeled after hierarchy step $j+1$, in which the edges of $\mathcal{E}_{\lambda-j-1}$ are considered. Let therefore $T_{\lambda-j}$ be a minimum tree-support for $\mathrm{H}(D)$ regarding w whose edge set contains $F_{\lambda-j}$. If $\mathcal{E}_{\lambda-j-1} = \emptyset$, it follows that $F_{\lambda-j-1} = F_{\lambda-j}$, therefore, $T_{\lambda-j}$ concludes the outer induction step. Let now be $\mathcal{E}_{\lambda-j-1} \neq \emptyset$. We show by induction that at any point during hierachy step $j+1$ if F is the current set of edges computed by Algorithm MTS-labeled there exists a minimum tree-support for $\mathrm{H}(D)$ regarding w whose edge set contains all edges in F. We will refer to this induction hypothesis as the *inner* induction hypothesis.

In the beginning of hierarchy step $j+1$ the current edge set is $F = F_{\lambda-j}$, therefore, $T_{\lambda-j}$ fulfills all required properties to conclude the inner induction base case. Now let the inner induction hypothesis be true for some non-final step of hierarchy step $j+1$, let $\hat{T}$ be a minimum tree-support for $\mathrm{H}(D)$ regarding w with $F \subseteq \mathrm{E}(\hat{T})$ and let $e = \{z, z'\} \in \mathcal{E}_{\lambda-j-1}$ be the next edge that is considered by Algorithm MTS-labeled. If z and z' are in the same connected component of the graph with edge set F and vertex set Z, F remains unchanged and $\hat{T}$ concludes the inner induction step. Let therefore now z and z' be in distinct connected components of the graph with edge set F and vertex set Z, so that e is the next edge that is added by Algorithm MTS-labeled. If $e \in \mathrm{E}(\hat{T})$, then again $\hat{T}$ concludes the inner induction step. Let therefore be $e \notin \mathrm{E}(\hat{T})$.

The edge set $\{e\} \cup \mathrm{E}(\hat{T})$ contains a cycle. Let $\hat{R}$ be the edge set of that cycle. Lemma 4 states that T, the output of Algorithm MTS-labeled, is a tree-support for $\mathrm{H}(D)$. The edge set $\mathrm{E}(T) = F_0$ contains all edges of $F \cup e$. We will now show that there exists an edge $\hat{e} = \{\hat{z}, \hat{z}'\} \in \hat{R} \setminus F_0$ such that the edge set $\{\hat{e}\} \cup \mathrm{E}(T)$ contains a cycle whose edge set contains e. The edge set $\hat{R} \setminus F_0$ can not be empty because this would imply that F_0 contains all edges of $\hat{R}$ and therefore a cycle. Assume that there is no edge in $\hat{R} \setminus F_0$ with the properties of $\hat{e}$ and let be $\hat{R} = (e, e_1, \dots, e_w) = (\{z, z'\}, \{z_1, z_1'\}, \dots, \{z_w, z_w'\})$ and let $1 \leq f_1 < \dots < f_v \leq w$ be the indices of the edges of $\hat{R}$ that are elements of $\hat{R} \setminus F_0$. A subset of the edges of the paths $\mathrm{p}(z', z_{f_1}, T)$, $\mathrm{p}(z_{f_1}, z_{f_1}', T)$, $\mathrm{p}(z_{f_1}', z_{f_2}, T)$, $\mathrm{p}(z_{f_2}, z_{f_2}', T), \dots, \mathrm{p}(z_{f_v}, z_{f_v}', T)$, $\mathrm{p}(z_{f_v}', z, T)$ is the edge set of a simple cycle and since the paths only contain edges of $\mathrm{E}(T)$, this implies that there is a cycle in T, contradicting to T being a tree.

Let now R be the cycle in the edge set $\{\hat{e}\} \cup \mathrm{E}(T)$ with $e \in R$. Since T is a tree-support for $\mathrm{H}(D)$, the edges of path $\mathrm{p}(\hat{z}, \hat{z}', T)$, which are the edges of $R \setminus \{\hat{e}\}$, actively support $\mathrm{S}(\hat{z} \cap \hat{z}')$ and therefore e also actively supports $\mathrm{S}(\hat{z} \cap \hat{z}')$. On the other hand, since $\hat{T}$ is a tree-support for $\mathrm{H}(D)$, the edges of path $\mathrm{p}(z, z', \hat{T})$, which are exactly the edges of $\hat{R} \setminus \{e\}$, actively support $\mathrm{S}(z \cap z')$ and therefore, $\hat{e}$ also supports $\mathrm{S}(z \cap z')$ since $\hat{e} \in \hat{R}$. It follows that $z \cap z' = \hat{z} \cap \hat{z}'$ and therefore, that all edges in $\hat{R}$ actively support $\mathrm{S}(\hat{z} \cap \hat{z}')$. This implies that the tree $\hat{T}_e = (Z, (\mathrm{E}(\hat{T}) \setminus \{\hat{e}\}) \cup \{e\})$ is also a tree-support for $\mathrm{H}(D)$ because for any vertices $z_1, z_2 \in Z$ with $\hat{e} \in \mathrm{p}(z_1, z_2, \hat{T})$ the hyperedge set which has to be actively supported by any edge of $\mathrm{p}(z_1, z_2, \hat{T}_e)$ is $\mathrm{S}(z_1 \cap z_2) \subseteq \mathrm{S}(\hat{z} \cap \hat{z}')$ and the edge sets of the paths $\mathrm{p}(z_1, z_2, \hat{T})$ and $\mathrm{p}(z_1, z_2, \hat{T}_e)$ are equal except for the edges of the subpaths of $\mathrm{p}(z_1, z_2, \hat{T})$ and $\mathrm{p}(z_1, z_2, \hat{T}_e)$ whose edge sets are subsets of $\hat{R}$.

Since $\hat{T}$ is a minimum tree-support for $\mathrm{H}(D)$ regarding w, $\mathrm{C}(D)(e) = \mathrm{C}(D)(\hat{e})$ and $w(e) \leq w(\hat{e})$ because Algorithm MTS-labeled considered e before $\hat{e}$, it follows that $\hat{T}_e$ is a also minimum tree-support for $\mathrm{H}(D)$ regarding w. The edge set of $\hat{T}_e$ contains $F \cup \{e\}$, which concludes the induction step of the inner induction proof and by the principle of induction, it follows that there exists a minimum tree-support for $\mathrm{H}(D)$ regarding w whose edge set contains the in hierarchy step $j+1$ finally obtained edge set $F_{\lambda-j-1}$. This on the other hand concludes the induction step of the outer induction proof.

Therefore, by the principle of induction, our outer induction hypothesis is true and there exists a minimum tree-support for $\mathrm{H}(D)$ regarding w whose edge set contains the final edge set F_0, which is obtained after the last hierarchy step. Since Algorithm MTS-labeled considers every edge in E and adds it to F if the two vertices of the edges are in distinct connected components of the graph with the current edge set, F_0 is the edge set of a tree with vertex set Z, implying that the output $T = (Z, F_0)$ is a minimum tree-support for $\mathrm{H}(D)$ regarding w. □

Theorem 1. *Given a labeled hypergraph* $\mathrm{H}(D) = (Z, \mathrm{S}(L))$ *for an abstract Euler diagram description* $D = (L, Z \cup \{L_\emptyset\})$ *and a weight function* $w : E \to \mathbb{R}$*, where* E *is the edge set of a complete graph with vertex set* Z*, Algorithm MTS-labeled computes a minimum tree-support for* $\mathrm{H}(D)$ *regarding* w *or produces an infeasibility notification in* $O(n^2 m)$ *time with* $n = |Z|$ *and* $m = |L| = |\mathrm{S}(L)|$.

Proof. We utilize the feasibility check presented by Johnson and Pollak [JP87], which is described in the beginning of this section, to test whether H is a tree-hypergraph. If this is the case, Lemma 5 states that the output tree T is a minimum tree-support for $\mathrm{H}(D)$ regarding w. However, if $\mathrm{H}(D)$ is not a tree-hypergraph, we produce an infeasibility notification and thus, Algorithm MTS-labeled is correct.

For the feasibility check we have to compute the dual hypergraph $\mathrm{H}(D)^* = (Z^*, S^*) = (\mathrm{S}(L), \{S' \subseteq \mathrm{S}(L) | \exists z \in Z : z \in \bigcap_{s \in S'} s, \forall s \in \mathrm{S}(L) \setminus S' : z \notin s)\})$ of $\mathrm{H}(D)$ and test whether $\mathrm{H}(D)^*$ is acyclic. Computing Z^* takes $O(m)$ time and the hyperedge set of $\mathrm{H}(D)^*$ can be computed in $O(n^2 m)$ by testing for every vertex $z \in Z$ which hyperedges of $\mathrm{S}(L)$ contain z. The acyclicity test by Tarjan and Yannakakis [TY84] is described as being a linear time algorithm in the sense that its runtime is in $O(n' + m')$, where n' is the number of vertices of the input hypergraph and m' is the total size of the of the input hypergraphs hyperedge set, which is the sum of the number of elements of all hyperedges. Dual hypergraph $\mathrm{H}(D)^*$ has $|Z^*| = m$ vertices and $|S^*| = n$ hyperedges. Since every hyperedge of $\mathrm{H}(D)^*$ contains up to $|Z^*| = m$ elements, the time complexity for the acyclicity check is $O(n\, m)$ and the total time complexity for the feasibility check is therefore $O(n^2 m)$.

To calculate the cardinality values we have to compute the intersection $z \cap z'$ for all $O(n^2)$ pairs of vertices $\{z, z'\} \subseteq Z, z \neq z'$. Since every vertex is a set of at most m labels, this can be done in $O(n^2 m)$ time.

Recall that the zone skeleton $G = (Z, E)$ of D is a complete graph. The time complexity of Algorithm MST-Kruskal to compute a minimum spanning-tree for a graph with n' vertices and m' edges is $O(m' \log n')$ [CLRS09]. We now show that the time complexity for the operations performed in Line 8 to Line 16 of Algorithm MTS-labeled for the input $\mathrm{H}(D) = (Z, \mathrm{S}(L))$ is asymptotically equivalent to the time complexity of an application of Algorithm MST-Kruskal to $G = (Z, E)$ and some weight function w'.

We sort the edges of each cardinality individually, but since the sets $\mathcal{E}_i$ with $0 \leq i \leq |\mathrm{S}(L)| - 1$ are disjoint and $\bigcup_{0 \leq i \leq |\mathrm{S}(L)|-1} \mathcal{E}_i = E$, the time complexity for doing so is asymptotically equivalent to the time complexity for sorting all edges of E at once, which is done by Algorithm MST-Kruskal. After the sorting step Algorithm MST-Kruskal considers each edge in E exactly once and per edge performs two FindSet operations and up to one Union operation, totaling to $O(m)$ FindSet and $O(m)$ Union operations. Algorithm MTS-labeled considers the edges of E in a different order than Algorithm MST-Kruskal, but since the sets $\mathcal{E}_i$ with $0 \leq i \leq |\mathrm{S}(L)| - 1$ are disjoint and $\bigcup_{0 \leq i \leq |\mathrm{S}(L)|-1} \mathcal{E}_i = E$, each edge of E is also considered exactly once by Algorithm MTS-labeled resulting in an asymptotically equivalent number of $O(m)$ FindSet and $O(m)$ Union operations. Therefore the total time complexity for the operations performed in Line 8 to Line 16 of Algorithm MTS-labeled is $O(|E| \log |Z|)$. Zone skeleton G is a complete graph implying that $|E| = O(|Z|^2)$. The vertices in Z are pairwise distinct subsets of label set L, therefore, $\log |Z| = O(|L|)$. Since $|L| = |\mathrm{S}(L)| = m$ and $|Z| = n$, the complexity of Line 8 to Line 16 can be restated to $O(n^2 m)$, which then also is the total time complexity of Algorithm MTS-labeled. □

2.3 Computing minimum tree-supports for hypergraphs

In this section we present a modification for Algorithm MTS-labeled that allows us to also compute minimum tree-supports for the general version of hypergraphs instead of being restricted to the labeled case.

The optimization problem of computing a minimum tree-support for a hypergraph was introduced by Korach and Stern [KS03]. They present an algorithm, which is briefly described in Section 1.1.3, that solves this problem in $O(n^4m^2)$ time, where n is the number of vertices and m is the number of hyperedges of the input hypergraph. The algorithm presented in this section improves upon the approach of Korach and Stern by solving the problem in $O(n^2(m + \log n))$ time.

In a labeled hypergraph for an abstract Euler diagram description each vertex is a set of labels which encodes in which hyperedges the vertex is contained. The general idea to solve the problem of finding a minimum tree-support regarding some weight function w for a hypergraph $H = (V, S)$ is to compute the labeled hypergraph $\mathrm{H}(D) = (Z, \mathrm{S}(L))$ for the abstract Euler diagram description $D = (L, Z \cup \{L_\emptyset\})$, where L contains a label for each hyperedge of S and for each vertex $v \in V$ there exists a label set $z \in Z$ such that the labels of z correspond to the labels associated with the hyperedges of S that contain v. We can then apply Algorithm MTS-labeled to obtain a minimum tree-support for $\mathrm{H}(D)$ and finally retranslate the result to obtain a minimum tree-support for H regarding w.

In order to realize this idea, however, we must address the following problem. Since the zones of an abstract Euler diagram description are pairwise distinct label sets, the vertices of a labeled hypergraph for an abstract Euler diagram description are pairwise distinct with regard to their containment in the hyperedges. In the hypergraph H, however, any number of vertices can be contained in exactly the same set of hyperedges. As a result, D might be what we call a *relaxed* abstract Euler diagram description, which is an abstract Euler diagram description in which the zones are not necessarily pairwise distinct label sets. The labeled hypergraph $\mathrm{H}(D)$ for D can be created exactly like in the not relaxed case, however, in order to use Algorithm MTS-labeled we need a to make an adjustment. Since the vertex set of $\mathrm{H}(D)$ can contain multiple vertices with label set L, the potentially highest edge cardinality is now $\lambda = |L| = |\mathrm{S}(L)|$ instead of $\lambda - 1$. We therefore define the new Algorithm MTS-relaxed, which works exactly like Algorithm MTS-labeled, except that the the input is a labeled hypergraph for a relaxed abstract Euler diagram description and that the first hierarchy step considers the edges with cardinality λ instead of $\lambda - 1$. In the following Lemma we show that MTS-relaxed is correct and discuss it's runtime complexity.

Lemma 6. *Given a labeled hypergraph $\mathrm{H}(D) = (Z, \mathrm{S}(L))$ for a relaxed abstract Euler diagram description $D = (L, Z \cup \{L_\emptyset\})$ and a weight function $w : E \to \mathbb{R}$, where E is the edge set of a complete graph with vertex set Z, Algorithm MTS-relaxed computes a minimum tree-support for $\mathrm{H}(D)$ regarding w or produces an infeasibility notification in $O(n^2(m + \log n))$ time with $n = |Z|$ and $m = |\mathrm{S}(L)| = |L| = \lambda$.*

Proof. Observe that since Algorithm MTS-relaxed works almost exactly like Algorithm MTS-labeled, most arguments used in the correctness and complexity proofs for Algorithm MTS-labeled are valid for Algorithm MTS-relaxed as well. In these proofs, we use the facts that the considered abstract Euler diagram description is not relaxed and that the first hierachy step considers the edges with cardinality $\lambda - 1$ at exactly two occasions: (1) In the induction base cases of the proofs of Lemma 4 and Lemma 5 and (2) in the runtime analysis in the proof of Theorem 1. It is therefore sufficient to discuss how to adapt (1) and (2) to conclude this Lemma's proof.

(1) In the induction base cases of the proofs of Lemma 4 and Lemma 5 we argue that the edge set $\mathcal{E}_{\lambda-1}$ is a subset of the edge set of any tree-support for the given labeled hypergraph for an abstract Euler diagram description. We thereby show that there exists a minimum tree-support whose edge set contains the edge set $F_{\lambda-1}$, which is the set of edges obtained after the first hierachy step of Algorithm MTS-labeled. We now consider Algorithm MTS-relaxed, in which the first hierachy step considers $\mathcal{E}_{\lambda}$, which is the set of edges with cardinality λ. We therefore need to show that there exists a minimum tree-support for $\mathrm{H}(D)$ regarding w that contains the edge set that we obtain after the first hierachy step of Algorithm MTS-relaxed, which we denote with F_{λ}. If $\mathcal{E}_{\lambda} = \emptyset$, then our propositions holds trivially. Let therefore be $\mathcal{E}_{\lambda} \neq \emptyset$ and consider an edge $e = \{z, z'\} \in \mathcal{E}_{\lambda}$. The cardinality of e is $\mathrm{C}(D)(e) = \lambda$, therefore, both z and z' have label set L. Let $Z_L \subseteq Z$ be the set of vertices with label set L, let $\tilde{T} = (Z, \tilde{E})$ be a minimum tree-support for $\mathrm{H}(D)$ regarding w and consider the path $\mathrm{p}(z, z', \tilde{T})$. Since $z, z' \in \bigcap_{l \in L} \mathrm{s}(l)$, every edge of $\mathrm{p}(z, z', \tilde{T})$ has to actively support $\mathrm{S}(L)$ and therefore has cardinality λ, implying that $\mathcal{E}_{\lambda}$ induces a subtree $\tilde{T}_L = (Z_L, E_L)$ of $\tilde{T}$ since there exists an edge in $\mathcal{E}_{\lambda}$ for each pair of vertices in Z_L.

In the first hierachy step of MTS-relaxed, we obtain F_{λ} by essentially applying Algorithm MST-Kruskal to the subgraph G_{λ} of the zone skeleton G of D induced by $\mathcal{E}_{\lambda}$, where the zone skeleton of a relaxed abstract Euler diagram description is defined analogously to the not relaxed case. The trees $\tilde{T}$, $\tilde{T}_L$ and $T_L = (Z_L, F_{\lambda})$ meet the preconditions of Lemma 2 and we can therefore substitute $\tilde{T}_L$ with T_L to obtain $T_{\lambda} = (Z, (\tilde{E} \setminus E_L) \cup F_{\lambda})$, which is a spanning-tree of G. Since every edge of T_L actively supports the hyperedge set $\mathrm{S}(L)$ and since T_L is a minimum spanning-tree for G_{λ}, it follows that T_{λ} is a minimum tree-support for $\mathrm{H}(D)$ regarding w

(2) In the proof of Theorem 1, we showed that the time complexity of both the feasibility check and the computation of the edge cardinalities in Algorithm MTS-labeled (as well as in Algorithm MTS-relaxed) is $O(n^2 m)$ and that the total time complexity for the operations performed in Line 8 to Line 16 of is $O(n^2 \log n)$. We used the fact that the input of Algorithm MTS-labeled is labeled hypergraph for an (not relaxed) abstract Euler diagram description to argue that $\log n = O(m)$ and therefore, that the total runtime of Algorithm MTS-labeled is $O(n^2 m)$. However, we can not use this argument for MTS-relaxed, therefore, the total runtime of MTS-relaxed is $O(n^2(m + \log n))$. □

We now prove the correctness and analyze the runtime of Algorithm MTS-general, which utilizes Algorithm MTS-relaxed to realize the idea for computing minimum tree-supports for hypergraphs which was presented in the beginning of this section.

Theorem 2. *Given a hypergraph $H = (V, S)$ and a weight function $w : E \to \mathbb{R}$, where E is the edge set of a complete graph with vertex set V , Algorithm MTS-general computes a minimum tree-support for H regarding w or produces an infeasibility notification in $O(n^2(m + \log n))$ time with $n = |V|$ and $m = |S|$.*

Proof. Again we utilize Johnson and Pollaks [JP87] feasibility check to test if H is a tree-hypergraph and print an infeasibility notification if this is not the case.

Let now H be a tree-hypergraph. The hypergraph $H(D)$ constructed by Algorithm MTS-general is *equivalent* to H in the sense that for any set of indices $I \subseteq \{1, 2, \ldots, n\}$ the hyperedge $\{v \in V \mid \exists i \in I : v = v_i\}$ is element of the hyperedge set S of H if and only if the hyperedge $\{z \in Z \mid \exists i \in I : z = z(v_i)\}$ is element of the hyperedge set $S(L)$ of $H(D)$. An implication of this equivalence relationship is that for any tree-support T_g of H the edge set $\{\{z, z'\} \subseteq Z \mid \exists i, j \in \{1, 2, \ldots, n\} : z = z(v_i), z' = z(v_j), \{v_i, v_j\} \in \mathrm{E}(T_g)\}$ is the edge set of a tree-support for $H(D)$ and that for any tree-support T_l of $H(D)$ the edge

Algorithm: MTS-general

```
input : hypergraph $H = (V, S) = (\{v_1, \dots, v_n\}, \{s_1, \dots, s_m\})$,
        weight function $w : E \to \mathbb{R}$ where $E$ is the edge set of a complete graph with
        vertex set $V$
output: minimum tree-support for $H$ regarding $w$ or infeasibility notification

Feasibility Check
if FeasibilityTreeSupport($H$) then
    return ‘not feasible‘

Computing labeled hypergraph $H(D)$
for $j = 1$ to $m$ do
    $l(s_j) =$ ‘$s_j$‘
for $i = 1$ to $n$ do
    $z(v_i) = \emptyset$
    for $j = 1$ to $m$ do
        if $v_i \in s_j$ then
            $z(v_i) = z(v_i) \cup \{l(s_j)\}$
for $j = 1$ to $m$ do
    $s(l(s_j)) = \emptyset$
    for $i = 1$ to $n$ do
        if $v_i \in s_j$ then
            $s(l(s_j)) = s(l(s_j)) \cup \{z(v_i)\}$
$Z = \{z(v_1), \dots, z(v_n)\}$
$S(L) = \{s(l(s_1)), \dots, s(l(s_m))\}$
$H(D) = (Z, S(L))$

Translating weight function
for $1 \leq i < l \leq n$ do
    $w'(\{z(v_i), z(v_l)\}) = w(\{v_i, v_l\})$

Computing minimum tree-support
$T'$ =MTS-relaxed($H(D), w'$)
$T = (V, \{\{v, v'\} \subseteq V \mid \exists i, j \in \{1, 2, \dots, n\} : v = v_i, v' = v_j, \{z(v_i), z(v_j)\} \in \mathrm{E}(T')\})$
return $T$
```

set $\{\{v, v'\} \subseteq V \mid \exists i, j \in \{1, 2, \ldots, n\} : v = v_i, v' = v_j, \{z(v_i), z(v_j)\} \in \mathrm{E}(T_l)\}$ is the edge set of a tree-support for H.

Since H is a tree-hypergraph, H and $H(D)$ are equivalent in the sense defined above and $H(D)$ is a labeled hypergraph for the relaxed abstract Euler diagram description $D = (Z, S(L))$, it follows by Lemma 6 that MTS-relaxed$(H(D), w')$ does indeed return a minimum tree-support T' for $H(D)$ regarding w'. Since H and $H(D)$ are equivalent and since $w(e) = w'(e')$ for any $e = \{v_i, v_j\} \subseteq V$ and $e' = \{z(v_i), z(v_j)\} \subseteq Z$ with $i, j \in \{1, 2, \ldots, n\}, i \neq j$, it follows that the constructed tree T is a minimum tree-support for H regarding w.

The time complexity of the feasibility check is $O(n^2 m)$ as shown in in the proof of Theorem 1. Keeping in mind that each hyperedge of S contains at most n vertices one can easily verify that the construction of $H(D)$ also takes $O(n^2 m)$ time. Computing w' can be done in $O(n^2)$ time. Since $|Z| = |V| = n$ and $|S(L)| = |S| = m$, Lemma 6 states that the computation of T' =MTS-relaxed$(H(D), w')$ takes $O(n^2(m + \log n))$ time. Finally, the tree T' has $n-1$ edges and to obtain T for each edge $\{z, z'\} \in \mathrm{E}(T')$ we have to look up the vertices $v, v' \in V$ with $v = v_i$, $v' = v_j$, $z = z(v_i)$ and $z' = z(v_j)$ for $i, j \in \{1, 2, \ldots, n\}, i \neq j$, which can be done in constant time per edge and therefore in $O(n)$ total time. □

3. A framework for generating area-proportional Euler diagrams

In this chapter we present a framework for algorithms that generate area-proportional Euler diagrams for abstract Euler diagram descriptions whose labeled hypergraphs are tree-hypergraphs. We first introduce the general drawing approach and then conclude the chapter by presenting several aesthetic improvements and variations.

3.1 A general drawing approach

In this section we present a general approach to efficiently draw area-proportional Euler diagrams that satisfy several well-formedness conditions. We therefore first introduce a way to represent a tree in the plane using curves. To obtain an area-proportional Euler diagram $\mathcal{D}$ that realizes an abstract Euler diagram description $D = (L, Z \cup \{L_\emptyset\})$ we use a tree T with $\mathrm{V}(T) = Z$ and modify its representation in the plane. The Euler diagram $\mathcal{D}$ is guaranteed to satisfy several well-formedness conditions. We show that if T is a tree-support for the labeled hypergraph $\mathrm{H}(D)$ of D the resulting Euler diagram $\mathcal{D}$ has even superior well-formedness properties. We conclude the section by summarizing the results of Section 2.2 and Section 3.1.

Chow explained a general concept for obtaining an Euler diagram, given a connectivity graph (Theorem 5.0.4 in [Cho07]). Connectivity graphs are graphs with properties similar to the properties of planar supports for labeled hypergraphs. The concept involves drawing the connectivity graph's planar dual. Our general approach for obtaining an Euler diagram via a representation of a graph is similar to the approach used by Chow, so that, on the one hand, the framework described in this section can be seen as a concrete instantiation of Chow's general concept. However, recall that our definition of Euler diagrams is more general than Chow's and so is our general drawing concept.

First we introduce some definitions and related notation. Let $G = (V, E)$ be a planar graph. A *contact representation* of G is a set of curves $\mathcal{R}$ that contains a simple closed curve for every vertex in V such that the images of any two distinct curves c, c' of $\mathcal{R}$ dot not cross and such that the images of c and c' intersect if and only if there exists an edge in E that is incident to both the vertex associated with c and the vertex associated with c'. Furthermore, if c and c' intersect then each of the connected components of $\mathrm{image}(c) \cap \mathrm{image}(c')$ contains more than one point. If the interior of each curve of $\mathcal{R}$ is convex, then $\mathcal{R}$ is called *convex*. If a function $area : V \to \mathbb{R}^+$ is provided, we call the contact representation $\mathcal{R}$ *area-proportional* with respect to *area* if for any $v \in V$ the region interior to the curve associated with v has area $area(v)$. We call half-lines originating at the point $0 = (0, 0) \in \mathbb{R}^2$ *rays*. Given a set of clockwise ordered rays $R = \{r_1, \dots, r_k\}$ we call r_i and r_{i+1} clockwise *consecutive* for any $1 \leq i \leq k-1$. We use $\overline{ab}$ to denote the line segment connecting points $a, b \in \mathbb{R}^2$. A line segment $\overline{ab}$ *connects* two consecutive rays r, r' of R if a is a point of r and b is a point of r' or vice versa.

We now present an approach to generate a convex and area-proportional contact representation of a tree.

Lemma 7. *Let $T = (V, E)$ be a tree, let $v_1 \in V$ be a vertex and let $area : V \to \mathbb{R}^+$ be a function. It is possible to construct a convex contact representation of T that is area-proportional with respect to area in $O(|V|)$ time.*

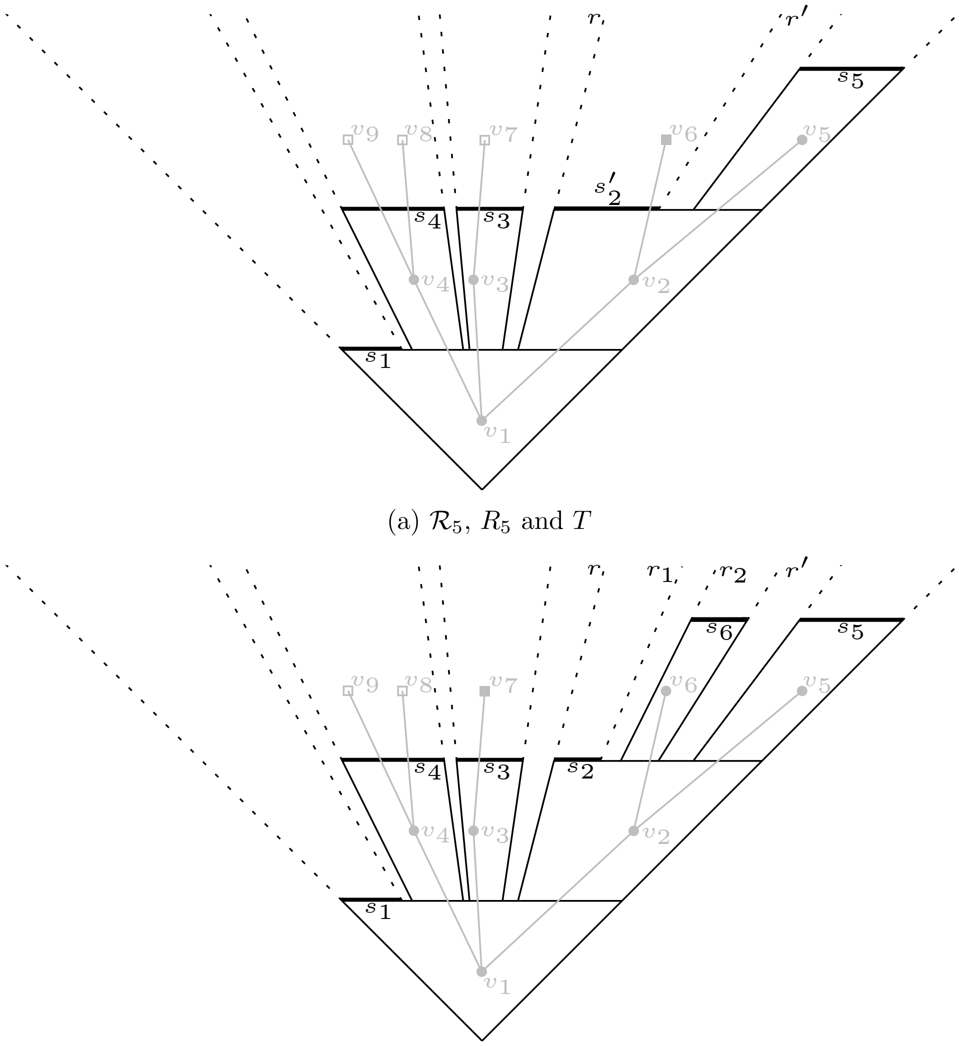
Proof. Let $(v_1, \dots, v_{|V|})$ be a breadth-first traversal of T and let T_i be the subtree of T induced by the vertex set $\{v_1, \dots, v_i\}$ for any $1 \leq i \leq |V|$. We show by induction that:

(1) For every $1 \leq i \leq |V|$ we can construct a convex contact representation $\mathcal{R}_i$ of T_i that is area-proportional with respect to *area*.
(2) We can obtain a set of rays R_i such that for each curve c of $\mathcal{R}_i$ there exists a line segment s in the image of c that connects two clockwise consecutive rays r, r' of R_i such that the closed unbounded region formed by r, r' and s does not contain any point of any curve of $\mathcal{R}_i$ except for the points of c in s.

For the induction base case we have to construct a contact representation $\mathcal{R}_1$ of $T_1 = (\{v_1\}, \emptyset)$ and a set of rays R_1 that satisfy the conditions (1) and (2). We choose any two rays r, r' and a point p of r and a point p' of r' such that $||p||_2 = ||p'||_2 > 0$ and such that the line segments $\overline{0p}, \overline{0p'}$ and $\overline{pp'}$ form an equilateral triangle with area $area(v_1)$. We associate the curve c_1 that describes this triangle with v_1 and define $\mathcal{R}_1 = \{c_1\}$, $R_1 = \{r, r'\}$ and $s_1 = \overline{pp'}$ which is the required line segment in the image of c_1. It is easy to see that $\mathcal{R}_1$ and R_1 together with s_1 fulfill the conditions (1) and (2) and therefore, conclude the induction base case.

For the induction step let our induction hypothesis be true for some $i \in \mathbb{N}$ with $1 \leq i \leq |V| - 1$. Let $\mathcal{R}_i$ be a contact representation of $T_i = (\{v_1, \dots, v_i\}, E_i)$ and let R_i be a set of rays so that $\mathcal{R}_i$ and R_i satisfy the conditions of the induction hypothesis. We have to construct a contact representation $\mathcal{R}_{i+1}$ for $T_{i+1} = (\{v_1, \dots, v_{i+1}\}, E_i \cup \{e\})$ and obtain a set of rays R_{i+1} such that $\mathcal{R}_{i+1}$ and R_{i+1} also satisfy these conditions. The edge set of T_{i+1} contains exactly one more edge than the edge set of T_i, therefore, v_{i+1} has exactly one neighbour $v_j, 1 \leq j \leq i$ in T_{i+1}. Let c_j be the curve of $\mathcal{R}_i$ that is associated with v_j. By induction hypothesis there exists a line segment s'_j in the image of c_j that connects two clockwise consecutive rays r, r' of R_i such that the closed unbounded region formed by r, r' and s'_j does not contain any point of any curve of $\mathcal{R}_i$ except for the points of c_j in s'_j. Let $p, p' \in \mathbb{R}^2$ be the endpoints of s'_j so that $s'_j = \overline{pp'}$ and p is a point of r and p' is a point of r' and let $p_1, p_2 \in \{x \in \mathbb{R}^2 \mid \exists \lambda \in (0,1) : x = \lambda p + (1-\lambda)p'\}$ be distinct points of s'_j such that the rays r_1, r_2 through p_1 or p_2 respectively are clockwise consecutive rays in $\{r, r_1, r_2, r'\}$. We compute a point $\bar{p}_2$ of r_2 and a point $\bar{p}'$ of r' such that $\overline{\bar{p}_2, \bar{p}'}$ is parallel to $\overline{p_2, p'}$ and such that the trapezoid formed by $\overline{p_2, p'}$, $\overline{\bar{p}_2, \bar{p}'}$, $\overline{p_2, \bar{p}_2}$ and $\overline{p', \bar{p}'}$ is located in the closed unbounded region formed by r, r' and s'_j and has area $area(v_{i+1})$. We define c_{i+1} to be the curve that describes this trapezoid, associate v_{i+1} with c_{i+1} and set $s_{i+1} = \overline{\bar{p}_2, \bar{p}'}$. We also define $s_j = \overline{p, p_1}$ and set $\mathcal{R}_{i+1} = \mathcal{R}_i \cup \{c_{i+1}\}$ and $R_{i+1} = R_i \cup \{r_1, r_2\}$.

The closed unbounded region formed by r, r' and s'_j does not contain any point of any curve of $\mathcal{R}_i$ except for the points of c_j in s'_j. Therefore, the image of c_{i+1} intersects



(a) $\mathcal{R}_5$, R_5 and T

(b) $\mathcal{R}_6$, R_6 and T

Figure 3.1: Illustration of the induction step in the proof for Lemma 7 with $i = 5$ and $j = 2$.

with c_j but none of the other curves of R_i. Furthermore, curve c_{i+1} describes a trapezoid and therefore a convex region which by construction has area $area(v_{i+1})$, thus $\mathcal{R}_{i+1}$ is a convex area-proportional contact representation of T_{i+1} with respect to *area* (1). Since the closed unbounded region formed by r, r' and s'_j does not contain any point of any curve of $\mathcal{R}_i$ except for the points of c_j in s'_j, we know that this is also true for the closed unbounded region formed by r, r_1 and s'_j and the closed unbounded region formed by r_2, r' and s'_j. Note that the closed unbounded region formed by r, r_1 and s'_j is in fact the closed unbounded region formed by r, r_1 and s_j. The line segment s_j is therefore the required line segment in the image of c_j. Since the closed unbounded region formed by r_2, r' and s_{i+1} is a subregion of the closed unbounded region formed by r_2, r' and s'_j the line segment s_{i+1} is the required line segment in the image of curve c_{i+1} (2), thus concluding the induction step.

By the principle of induction, it follows that we can construct a convex contact representation $\mathcal{R}_{|V|}$ of $T_{|V|} = T$ that is area-proportional with respect to *area*. Obtaining a breadth-first traversal of T requires $O(|V| + |E|)$ time [CLRS09], which can be restated to $O(|V|)$ since $|E| = |V| - 1$ because T is a tree. The construction of the initial triangle and the trapezoids require constant time each, therefore, the total time complexity is $O(|V|)$. □

The next result relates contact representation to Euler diagrams and allows us efficiently generate Euler diagrams if a contact representation for a special kind of tree is provided.

Lemma 8. *Let $D = (L, Z \cup \{L_\emptyset\})$ be an abstract Euler diagram description with $L_\emptyset \notin Z$, let T be a tree with vertex set $\mathrm{V}(T) = Z$ and let $\mathcal{R}$ a contact representation of T. It is possible to generate an Euler diagram that realizes D and satisfies the 'simple curves' and the 'connected concrete zones' property in $O(|Z|\,|L|)$ time.*

Proof. Recall that every vertex of Z is a set of labels from L. Let $l \in L$ be a label and C be a connected component of the subgraph T_l of T induced by the vertices of Z containing l. In the contact representation $\mathcal{R}$ the zones of $\mathrm{V}(C)$ are represented by a set of curves $\mathcal{C}$ such that the union of the images of the curves in $\mathcal{C}$ is a connected component in the plane and therefore a closed curve with image $\bigcup_{c \in \mathcal{C}} \mathrm{image}(c) \setminus \{x \in \mathbb{R}^2 \mid \exists c, c' \in \mathcal{C} : c \neq c', x \in \mathrm{image}(c), x \in \mathrm{image}(c')\}$ describes a region of the plane which by $\mathcal{R}$ is used to represent zones with label l. We assign the label l to this curve and repeat this process for the other connected components of T_l to obtain a set of curves with label l. We furthermore repeat this process for every label in L to obtain a set of labeled curves and therefore an Euler diagram $\mathcal{D}$ that by construction realizes D.

Let $c \in \mathcal{R}$ be the curve that is associated with zone $z \in Z$. In $\mathcal{D}$ the concrete zone with label set z consists of exactly one minimal region which is the interior of c, therefore, $\mathcal{D}$ satisfies the 'connected concrete zones' property. Let now c be a curve of $\mathcal{D}$. Curve c describes a boundary of a union of the interior of simple closed curves and is therefore also simple since if two curves of $\mathcal{R}$ intersect, the connected components of their intersection contain more than one point. Therefore, $\mathcal{D}$ satisfies the 'simple curves' property.

For the time complexity analysis observe that we can use the following procedure to obtain $\mathcal{D}$. Let $\{z, z'\}$ be an edge of T and let c, c' be the curves of $\mathcal{R}$ that are associated with z and z' respectively. We assign the label set $(z \cup z') \setminus (z \cap z')$ to the points $\mathrm{image}(c) \cap \mathrm{image}(c')$ and repeat this process for every edge of T, which takes $O(|Z|\,|L|)$ time in total since T is a tree and every vertex of Z is a set of up to $|L|$ labels. Let now $z \in Z$ be a vertex of T and let $c \in \mathcal{R}$ be the curve associated with z. We assign the label set z to the points of $\mathrm{image}(c)$ which have not yet been labeled in the previous step. These are the points that in

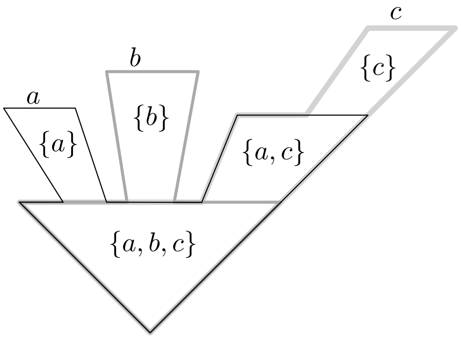


Figure 3.2: Euler diagram generated by using Lemma 7 and Corollary 2.

the contact representation $\mathcal{R}$ separate the interior of c and the outer infinitely large region. We repeat this process for every vertex of T which also requires total time $O(|Z|\,|L|)$. □

Observe that the procedure described in Lemma 8 can be used to generate an Euler diagram that realizes D and satisfies the 'simple curves' and the 'connected concrete zones' property even if T is a graph that is not necessarily a tree. In Section 2.1 we introduced labeled hypergraphs as a connection between hypergraphs and Euler diagrams. We now extend the results of Lemma 8 by showing that the generated Euler diagram also possesses the 'unique curve labels' property if T is a tree-support for the labeled hypergraph of D. Note, however, that if T is a support that is not necessarily a tree this additional property is not guaranteed, since there might be 'holes' in the concrete zones of the resulting diagram.

Corollary 2. *Let $D = (L, Z \cup \{L_\emptyset\})$ be an abstract Euler diagram description with $L_\emptyset \notin Z$, let T be a tree with vertex set $\mathrm{V}(T) = Z$ and let $\mathcal{R}$ be a contact representation of T. If T is a tree-support for the labeled hypergraph $\mathrm{H}(D) = (Z, \mathrm{S}(L))$ for D, it is possible to generate an Euler diagram that realizes D and satisfies the 'simple curves', the 'connected concrete zones' and the 'unique curve labels' property in $O(|Z|\,|L|)$ time.*

Proof. We apply the procedure described in Lemma 8 to obtain an Euler diagram $\mathcal{D}$ that realizes D and satisfies the 'simple curves' and the 'connected concrete zones' property in $O(|Z|\,|L|)$ time. Let $l \in L$ be a label. If T is a tree-support for $\mathrm{H}(D)$ the subgraph T_l of T induced by the vertices containing l is connected and therefore only one simple closed curve is required to describe the region of the plane which by $\mathcal{R}$ is used to represent the zones containing l, implying that $\mathcal{D}$ satisfies the 'unique curve labels' property. □

Figure 3.2 shows an Euler diagram that was generated by using Lemma 7 and Corollary 2. The realized abstract Euler diagram descriptions is $D = (L, Z \cup \{L_\emptyset\})$ with $L = \{a, b, c\}$ and $Z = \{\{a\}, \{b\}, \{c\}, \{a, c\}, \{a, b, c\}\}$. In Section 1.1.2 we have seen that the 'no concurrency' property is a strong constraint for Euler diagram generating algorithms in the sense that there exist very simple abstract Euler diagram descriptions for which it is not possible to construct Euler diagrams that realize them while satisfying the 'no concurrency' property. The set of zones Z contains exactly two zones containing label b namely $\{b\}$ and $\{a, b, c\}$. Assuming one does not want to violate the 'simple curves' or the 'unique curve labels' property, it is not possible to realize D while satisfying the 'no concurrency' property since in this case a minimal region of the concrete zone with label set $\{b\}$ and a minimal region of the concrete zone with label set $\{a, b, c\}$ have to be neighbours, which is only possible if the curves labeled a and c are concurrent.

In Section 2.1, we defined the concurrency weight function $\mathrm{w}(D) : \mathrm{E}(\mathrm{K}(D)) \to \mathbb{N}_0$ for abstract Euler diagram description $D = (L, Z \cup \{L_\emptyset\})$ as $\mathrm{w}(D)(\{z, z'\}) = |(z \cup z') \setminus (z \cap z')|$, where $\mathrm{K}(D)$ is the complete graph with vertex set Z. For an edge $\{z, z'\}$ this function describes the number of (concurrent) curves that is needed if a minimal region belonging to the concrete zone with label set z is a neighbour of a minimal region belonging to the concrete zone with label set z'. We say that an Euler diagram realizing D that was generated using Corollary 2 has *minimal internal concurrency* if the input tree-support's weight regarding $\mathrm{w}(D)$ is minimal amongst all tree-supports for $\mathrm{H}(D)$, which means that the input is a minimum tree-support for $\mathrm{w}(D)$. Note that this definition should not be seen as a general well-formedness property as it is tied to Euler diagrams generated with the procedure presented in Corollary 2. Also take into account that this conditions does not factor in the number of concurrent curves between the minimal regions of concrete zones with some label set of Z and the outer minimal region belonging to the concrete zone with label set $L_\emptyset$. However, for Euler diagrams generated with Corollary 2, minimizing the input tree-support's weight regarding $\mathrm{w}(D)$ does minimize the number of concurrent curves between each neighbour pair of minimal regions belonging to concrete zones with non-empty label sets in the resulting Euler diagram. Observe that the tree used to generate the Euler diagram in Figure 3.2 is not a minimum tree-support as replacing the edge $\{\{a\}, \{a, b, c\}\}$ with the edge $\{\{a\}, \{a, c\}\}$ results in a tree-support with lower weight.

We now combine the results of Section 2.2 and Section 3.1 in a final theorem.

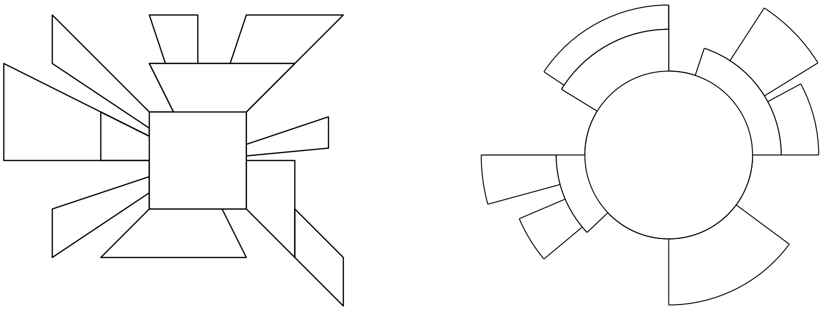
Theorem 3. *Let $D = (L, Z \cup \{L_\emptyset\})$ be an abstract Euler diagram description with $L_\emptyset \notin Z$ and let $area : Z \to \mathbb{R}^+$ be a function. If the labeled hypergraph $\mathrm{H}(D) = (Z, \mathrm{S}(L))$ for D is a tree-hypergraph, it is possible to generate an Euler diagram in $O(|Z|^2|L|)$ time that (1) realizes D, (2) is area-proportional with respect to area, satisfies (3) the 'simple curves', (4) the 'connected concrete zones', (5) the 'unique curve labels' and (6) the 'convex minimal regions' property and has (7) 'minimal internal concurrency'.*

Proof. First, we have to compute the labeled hypergraph $\mathrm{H}(D)$ for D. We create $|L|$ hyperedges and then add each zone to the hyperedges that correspond to the zones labels which can be done in $O(|Z|\,|L|)$ time. We use Theorem 1 to compute a minimum tree-support T for $\mathrm{H}(D)$ regarding the concurrency weight function $\mathrm{w}(D)$ in $O(|Z|^2|L|)$ time. Note that the edges' values respective to the concurrency weight function $\mathrm{w}(D)$ can be computed analogously to the edges' values respective to the cardinality weight function $\mathrm{C}(D)$ and therefore also in total time of $O(|Z|^2|L|)$. With Lemma 7 we construct a convex contact representation $\mathcal{R}$ of T that is area-proportional with respect to *area* in $O(|Z|)$ time and use Corollary 2 to generate an Euler diagram $\mathcal{D}$ that realizes D (1) and satisfies the 'simple curves' (3), the 'connected concrete zones' (4) and the 'unique curve labels' (5) property in $O(|Z|\,|L|)$ time. Since each concrete zone of $\mathcal{D}$ consists of exactly one minimal region that is the interior of the curve of $\mathcal{R}$ associated with the corresponding zone of Z and since $\mathcal{R}$ is convex and area-proportional with respect to *area*, it follows that $\mathcal{D}$ satisfies the 'convex minimal regions' property (6) and is area-proportional regarding *area* (2). Finally, since T is a minimum tree-support for $\mathrm{H}(D)$ regarding $\mathrm{w}(D)$, the resulting Euler diagram $\mathcal{D}$ has 'minimal internal concurrency' (7). □

3.2 Aesthetic improvements and variations

In this last section we want to present a brief list of variations and improvements related to the aesthetic aspects of the contact representations and Euler diagrams generated with Lemma 7 and Corollary 2.

During the construction of the contact representation it is not necessary to reserve a line segment s in a curve c that represents some vertex v if all neighbours of v are already represented. Therefore, when creating the curve for the last neighbour of v instead of



(a) (b)

Figure 3.3: Variations of the contact representations for trees from Lemma 7

choosing two new rays that are located clockwise consecutively between the rays r, r' that are connected by s it is possible to directly use the rays r and r'.

When constructing a contact representation with Lemma 7 the choice of the *root* vertex v_1 impacts the output. Depending on the given data, experimenting with different centrality measures for graphs, for example the 'center of a graph' [KLP$^+$05], might yield better results. In our basic contact representation construction we use a triangle to represent the root vertex and the trapezoids representing the remaining vertices are attached to one specific side of the triangle. In order to obtain a representation that could be considered more balanced we suggest representing the root vertex with a regular polygon and attaching the neighbours of the root vertex to all of the polygon's sides, see Figure 3.3a for an illustration. One might also consider using a polygon that is not necessarily regular but instead uses longer sides to attach neighbours that are supposed to cover a larger area or that have a higher degree compared to the remaining neighbours. To obtain a more smooth looking Euler diagram we suggest using a circle to represent the root vertex and using circular arcs instead of line segments to represent the remaining vertices, see Figure 3.3b for an example of such a representation. However, if this variation is used, the resulting Euler diagrams do not satisfy the 'convex minimal regions' property anymore. Observe that such contact representations have resemblance to radial, space-filling hierachy visualizations, see, for example, [SZ00].

Finally, the distance between the rays chosen when representing a vertex impacts how flat or thin the resulting minimal region will be. This can be used to regulate the overall diameter as well as other parameters of the final representation.

4. Conclusion

We have presented a framework for algorithms that for the class of abstract Euler diagram descriptions whose labeled hypergraphs are tree-hypergraphs generate area-proportional Euler diagrams that satisfy the 'simple curves', the 'connected concrete zones', the 'unique curve labels' and the 'convex minimal regions' properties and have 'minimal internal concurrency'. The algorithms in this framework are efficient and easy to implement. We have also provided an algorithm that computes minimum tree-supports for hypergraphs in $O(n^2(m + \log n))$ time, where n is the number of vertices and m is the number of hyperedges, and thereby substantially improves upon the previously known $O(n^4m^2)$ time algorithm by Korach and Stern [KS03].

For future directions, recall that the Euler diagram drawing approach used in Section 3.1 is not restricted to contact representations of abstract Euler diagram descriptions that are based on trees. In fact the approach works for contact representations of any planar graph, however, the resulting Euler diagrams are not guaranteed to satisfy the same number of well-formedness conditions. For example, even if a planar support is used, the resulting Euler diagrams are not guaranteed to satisfy the 'unique curve labels' property since there might be 'holes' in the concrete connected zones. Nevertheless, experimenting with other types of supports, for example outerplanar supports, might still be worthwhile. This motivates work related to the interesting open problem of deciding whether a hypergraph has an outerplanar support. Finally, it would be interesting to actually implement the Euler diagram generating algorithm as it is, to our knowledge, the first efficient algorithm capable of drawing accurate area-proportional Euler diagrams with any number of labels and without the constraint that the zone that contains all labels has to be represented and, additionally, the generated Euler diagrams are guaranteed to satisfy a set of nice, yet not too restricting well-formedness conditions, as explained in Section 1.1.2. However, our algorithm is of course still restricted to abstract Euler diagram descriptions whose labeled hypergraphs are tree-hypergraphs.

Bibliography

[BCPS11] U. Brandes, S. Cornelsen, B. Pampel, and A. Sallaberry, “Blocks of hypergraphs,” in *Combinatorial Algorithms*, ser. Lecture Notes in Computer Science, C. Iliopoulos and W. Smyth, Eds. Springer Berlin Heidelberg, 2011, vol. 6460, pp. 201–211. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-19222-7_21

[BFMY83] C. Beeri, R. Fagin, D. Maier, and M. Yannakakis, “On the desirability of acyclic database schemes,” *J. ACM*, vol. 30, no. 3, pp. 479–513, Jul. 1983. [Online]. Available: http://doi.acm.org/10.1145/2402.322389

[BKM$^+$10] K. Buchin, M. Kreveld, H. Meijer, B. Speckmann, and K. Verbeek, “On planar supports for hypergraphs,” in *Graph Drawing*, ser. Lecture Notes in Computer Science, D. Eppstein and E. Gansner, Eds. Springer Berlin Heidelberg, 2010, vol. 5849, pp. 345–356. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-11805-0_33

[BWR07] M. Brinkmeier, J. Werner, and S. Recknagel, “Communities in graphs and hypergraphs,” in *Proceedings of the sixteenth ACM Conference on Information and Knowledge management*, ser. CIKM ’07. New York, NY, USA: ACM, 2007, pp. 869–872. [Online]. Available: http://doi.acm.org/10.1145/1321440.1321563

[Cho07] S. Chow, “Generating and drawing area-proportional Euler and Venn diagrams,” Ph.D. dissertation, Victoria, B.C., Canada, Canada, 2007.

[CLRS09] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*, 3rd ed. The MIT Press, 2009.

[CR04] S. Chow and F. Ruskey, “Drawing area-proportional Venn and Euler diagrams,” in *Graph Drawing*, ser. Lecture Notes in Computer Science, G. Liotta, Ed. Springer Berlin Heidelberg, 2004, vol. 2912, pp. 466–477. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-24595-7_44

[DCES03] R. De Chiara, U. Erra, and V. Scarano, “VENNFS: a Venn-diagram file manager,” in *Information Visualization, 2003. IV 2003. Proceedings. Seventh International Conference on*, 2003, pp. 120–125.

[EdCC43] L. Euler and J. de Caritat Condorcet, *Lettres à une princesse d’Allemagne: sur divers sujets de physique et de philosophie.* Charpentier, 1843. [Online]. Available: http://books.google.fr/books?id=zKMAAAAAMAAJ

[FFH08] J. Flower, A. Fish, and J. Howse, “Euler diagram generation,” *J. Vis. Lang. Comput.*, vol. 19, no. 6, pp. 675–694, Dec. 2008. [Online]. Available: http://dx.doi.org/10.1016/j.jvlc.2008.01.004

[JP87] D. S. Johnson and H. O. Pollak, “Hypergraph planarity and the complexity of drawing Venn diagrams,” *Journal of Graph Theory*, vol. 11, no. 3, pp. 309–325, 1987. [Online]. Available: http://dx.doi.org/10.1002/jgt.3190110306

[KLP$^+$05] D. Koschützki, K. Lehmann, L. Peeters, S. Richter, D. Tenfelde-Podehl, and O. Zlotowski, "Centrality indices," in *Network Analysis*, ser. Lecture Notes in Computer Science, U. Brandes and T. Erlebach, Eds. Springer Berlin Heidelberg, 2005, vol. 3418, pp. 16–61. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-31955-9_3

[KS03] E. Korach and M. Stern, "The clustering matroid and the optimal clustering tree," *Mathematical Programming*, vol. 98, no. 1-3, pp. 385–414, 2003. [Online]. Available: http://dx.doi.org/10.1007/s10107-003-0410-x

[RZF08] P. Rodgers, L. Zhang, and A. Fish, "General Euler diagram generation," in *Diagrammatic Representation and Inference*, ser. Lecture Notes in Computer Science, G. Stapleton, J. Howse, and J. Lee, Eds. Springer Berlin Heidelberg, 2008, vol. 5223, pp. 13–27. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-87730-1_6

[RZP12] P. Rodgers, L. Zhang, and H. Purchase, "Wellformedness properties in Euler diagrams: Which should be used?" *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 7, pp. 1089–1100, Jul. 2012. [Online]. Available: http://dx.doi.org/10.1109/TVCG.2011.143

[SDC$^+$03] J. B. Soriano, K. J. Davis, B. Coleman, G. Visick, D. Mannino, and N. B. Pride, "The proportional Venn diagram of obstructive lung disease: Two approximations from the United States and the United Kingdom," *CHEST Journal*, vol. 124, no. 2, pp. 474–481, 2003. [Online]. Available: http://dx.doi.org/10.1378/chest.124.2.474

[SRH11] G. Stapleton, P. Rodgers, and J. Howse, "A general method for drawing area-proportional Euler diagrams," *J. Vis. Lang. Comput.*, vol. 22, no. 6, pp. 426–442, Dec. 2011. [Online]. Available: http://dx.doi.org/10.1016/j.jvlc.2011.07.001

[SRHT07] G. Stapleton, P. Rodgers, J. Howse, and J. Taylor, "Properties of Euler diagrams," *Electronic communications of the EASST: Proceedings of Layout of (Software) Engineering Diagrams*, vol. 7, 2007. [Online]. Available: http://journal.ub.tu-berlin.de/eceasst/article/view/92

[SZ00] J. Stasko and E. Zhang, "Focus+context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations," in *InfoVis'00: Proceedings of the IEEE Symposium on Information Visualization*, J. D. Mackinlay, S. F. Roth, and D. A. Keim, Eds. IEEE Computer Society, 2000, pp. 57–65.

[TVVb05] J. Thièvre, M. Viaud, and A. Verroust-blondet, "Using Euler diagrams in traditional library environment," in *Electronic Notes in Computer Science*, 2005, pp. 189–202.

[TY84] R. E. Tarjan and M. Yannakakis, "Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs," *SIAM J. Comput.*, vol. 13, no. 3, pp. 566–579, Jul. 1984. [Online]. Available: http://dx.doi.org/10.1137/0213035