Generating Euler Diagrams Through Combinatorial Optimization

Peter Rottmann¹, Peter Rodgers², Xinyuan Yan³, Daniel Archambault⁴, Bei Wang³, Jan-Henrik Haunert¹
¹University of Bonn, Institute of Geodesy and Geoinformation, Germany
²University of Kent, UK
³University of Utah, USA
⁴Newcastle University, UK

Abstract
Can a given set system be drawn as an Euler diagram? We present the first method that correctly decides this question for arbitrary set systems if the Euler diagram is required to represent each set with a single connected region. If the answer is yes, our method constructs an Euler diagram. If the answer is no, our method yields an Euler diagram for a simplified version of the set system, where a minimum number of set elements have been removed. Further, we integrate known wellformedness criteria for Euler diagrams as additional optimization objectives into our method. Our focus lies on the computation of a planar graph that is embedded in the plane to serve as the dual graph of the Euler diagram. Since even a basic version of this problem is known to be NP-hard, we choose an approach based on integer linear programming (ILP), which allows us to compute optimal solutions with existing mathematical solvers. For this, we draw upon previous research on computing planar supports of hypergraphs and adopt existing ILP building blocks for contiguity-constrained spatial unit allocation and the maximum planar subgraph problem. To generate Euler diagrams for large set systems, for which the proposed simplification through element removal becomes indispensable, we also present an efficient heuristic. We report on experiments with data from MovieDB and Twitter. Over all examples, including 850 non-trivial instances, our exact optimization method failed only for one set system to find a solution without removing a set element. However, with the removal of only a few set elements, the Euler diagrams can be substantially improved with respect to our wellformedness criteria.

CCS Concepts
• Human-centered computing → Information visualization; • Theory of computation → Integer programming; • Mathematics of computing → Hypergraphs;

1. Introduction
Euler diagrams are frequently used to visualize set systems. They represent each set as a region in the plane that is bounded by a closed curve. An area in an Euler diagram that is occupied by one or multiple regions indicates the existence of set elements that are contained in the corresponding sets and in no other set. An advantage of Euler diagrams is that they are intuitive to understand. However, they can become cluttered even for medium-sized set systems. For a given set system, it is possible that no Euler diagram exists when requiring the regions to be connected. Previous approaches [RZF08, SA08] split sets into two or more separate regions. However, splitting makes it harder to identify regions belonging to the same set, and increases the overall number of regions in the visualization.

Our work addresses two research problems. First, none of the existing methods can decide for a given set system whether an Euler diagram with a single connected region for each set exists, let alone generate an Euler diagram in every case where it is possible. Second, Euler diagrams have typically been considered as a tool to visualize set systems without loss of information. We think Euler diagrams can be used as a tool to visually summarize large set systems, but for this summary, the generation of Euler diagrams has to be combined with a simplification of the given data.

Our contribution is a new combinatorial optimization approach. Given a set system as input, our goal is to compute a simplified but still similar version of the set system that can be drawn nicely as an Euler diagram. The simplification is achieved by removing some set elements from the set system. To quantify the loss of information caused by the removal of elements, we assume that every element has a weight expressing the cost of its removal. For example, the weight of each element may be one, or it can be equal to the number of sets containing the element. More generally, any measure expressing the importance of the element can be applied.

Our most important contribution is an exact optimization method that minimizes the total weight of the removed set elements while ensuring that the resulting simplified set system can be embedded in the plane as an Euler diagram with a single connected region for each set. Moreover, we contribute multiple extensions of our
method to integrate additional optimization criteria that improve the resulting Euler diagram with respect to known wellformedness conditions [RZP11]. For large set systems, we introduce an efficient heuristic method. We conduct experiments with both the exact method and the heuristic on realistic set systems and assess the quality of their solutions as well as their efficiency.

2. Preliminaries and basic problem

To discuss our contribution in the context of related work and develop our methods, we view the given set system as a hypergraph. A hypergraph \( H = (V, E) \) consists of two sets, a vertex set \( V \) and a hyperedge set \( E \), where each hyperedge \( X \in E \) is a subset of \( V \), i.e., \( X \subseteq V \). In the hypergraph representation of a set system, every vertex \( v \in V \) corresponds to a set element and every hyperedge in \( E \) to a set. A support of a hypergraph \( H = (V, E) \) is a graph \( G \) with vertex set \( V \) such that each vertex set \( X \in E \) (i.e., each hyperedge) induces a connected subgraph in \( G \), meaning that the subgraph of \( G \) whose node set is \( X \) and whose edge set contains an edge for every two nodes in \( X \) that are adjacent in \( G \) is connected. In the context of Euler diagrams, it is most interesting to find a planar support of a given hypergraph \( H \), i.e., a support of \( H \) that can be drawn in the plane without edge crossings; see Fig. 1. In essence, a planar support of a given hypergraph can serve as the dual graph of the Euler diagram that is to be constructed; see Figs. 2 and 3.

Figure 1: A planar support of the hypergraph with nodes \( 1,\ldots,5 \) and hyperedges \( \{1,2,3\}, \{1,4,5\}, \{2,4\} \), each inducing a connected subgraph.

When we say that we remove an element \( v \) from a hypergraph \( H = (V, E) \), we imply that \( v \) is removed from the vertex set \( V \) of \( H \) as well as from every hyperedge \( X \in E \) in which \( v \) occurs. We introduce for every node \( v \in V \) a weight \( w(v) \in \mathbb{R}_{\geq 0} \) measuring the cost of its removal. If the aim is to minimize the number of element removals, we set \( w(v) = 1 \) for each set element. Else, if the aim is to minimize the loss of set memberships, we set \( w(v) = |\ell(v)| \), where \( \ell(v) \) is the set of hyperedges containing \( v \), i.e., \( \ell(v) = \{ X \in E \mid v \in X \} \). With \( H - v \), we refer to the hypergraph resulting from a hypergraph \( H = (V, E) \) after removing a set \( S \subseteq V \) from it. With this, we are ready to state a basic problem that asks for an optimal simplification of a set system to generate an Euler diagram.

Problem 1 (SetSystemSimplification)

Given a hypergraph \( H = (V, E) \) and a weighting \( w: V \rightarrow \mathbb{R}_{\geq 0} \), find a minimum-weight set \( S \subseteq V \) such that \( H - S \) has a planar support and return such a planar support as output.

We would like to point out that SetSystemSimplification has a solution with \( S = \emptyset \) if and only if the given hypergraph \( H \) has a planar support. Therefore, any exact optimization algorithm for SetSystemSimplification can be used as a tool to decide whether a hypergraph has a planar support. Since deciding whether a hypergraph has a planar support is NP-hard [JP87], we conclude that SetSystemSimplification is NP-hard, too.

As our problem is NP-hard, there is no reasonable hope for an efficient exact algorithm. Therefore, we focus on an approach based on integer linear programming, which is a general method for solving combinatorial optimization problems and has the advantage that we can employ existing mathematical solvers. With this, we can solve both the new problem SetSystemSimplification and the known problem of finding a planar support for a given hypergraph.

Solving SetSystemSimplification is insufficient since it neglects important criteria. Therefore, we will extend it to a multi-criteria problem, MCSetSystemSimplification, after reviewing wellformedness conditions for Euler diagrams. Both our exact method and our heuristic can deal with this extended problem.

3. Workflow

While we consider the exact optimization method and the heuristic for MCSetSystemSimplification as our main contribution, we also provide a complete workflow that yields an Euler diagram visualizing a simplified version of a given hypergraph. This workflow is illustrated in Fig. 4 for an artificial example and outlined below.

In Step 1 of our workflow, we replace every set of nodes that are contained in exactly the same hyperedges with a single node, yielding the condensed hypergraph. The motivation for this step is that nodes contained in exactly the same hyperedges should not be separated in the final visualization. Moreover, working with the condensed graph greatly reduces the running time of our methods. However, that this step may affect the existence of a planar support [vBKK+22]. Therefore, we may optionally skip this step, for example, if the aim is to compute a planar support of the original hypergraph.

In Step 2 of the workflow, the condensed hypergraph is used to compute the superdual graph, whose node set contains every node of the condensed hypergraph as well as a special node \( v_0 \) representing the outer face, and whose edge set contains every edge that may be useful for the dual graph of the Euler diagram. Next, in Step 3, a solution to MCSetSystemSimplification is computed in the form of a selection of nodes and edges of the superdual graph, which determines both the simplified hypergraph and the dual graph of the Euler diagram. This is achieved either with our exact optimization approach via integer linear programming or our heuristic method.

We use existing methods for the last three steps of our workflow. First, in Step 4, we compute a planar embedding of the dual graph generated in Step 3 [CP95]. In Step 5, the planar embedding is used to construct an initial Euler diagram [RZF08] that is guaranteed to contain only simple curves. Finally, in Step 6, the curves of the initial Euler diagram are smoothed [SAS16].

4. Related work

The visualization of set systems has been an active research topic for several years. A comprehensive study and classifica-
tion of the literature into six categories was done by Alsallakh et al. [AMA∗16]. Here, we focus on Euler diagrams, which have been investigated by two overlapping communities: information visualization and graph drawing.

4.1. Euler diagrams

Much research on Euler diagrams has focused on the definition of wellformedness conditions and how to ensure them. These conditions describe properties of Euler diagrams that target an improved comprehension of the diagrams. Sets are represented by labeled closed curves and set intersections are represented by zones, which are regions enclosed by the required closed curves. Typical wellformedness conditions include:

- **Connected zones**: each zone in the diagram is connected, broken in Fig. 5a.
- **No concurrency**: no pair of curves run concurrently, broken in Fig. 5b.
- **Transversality**: intersecting curves always intersect transversally (that is they cross, rather than just 'touch'), broken in Fig. 5c.
- **Simplicity**: all curves are simple curves. A curve is simple if it does not cross itself, broken in Fig. 5d.
- **No triple points**: there are no triple points of intersection among the curves, broken in Fig. 5e.
- **Unique label**: no curve label is used more than once, broken in Fig. 5f.

For a more formal definition of wellformedness conditions, see [SRHT07]. To verify the importance of each property, Rodgers et al. [RZP11] conducted a user study measuring time and errors during the completion of given tasks. Disconnected zones and concurrent curves caused significant errors and an increase in completion time. Moreover, representing a set with more than a single curve had a significantly adverse impact on task completion time.

Early approaches to Euler diagram embedding aimed to create wellformed Euler diagrams with rather strict conditions. Flower and Howe [FH02] defined concrete Euler diagrams, which met all of the wellformedness conditions above. However, strictly requiring all conditions means many set systems cannot be drawn.

Methods for generating Euler diagrams with relaxed wellformed conditions have been developed by Rodgers et al. [RZF08] and Simonetto and Auber [SA08]. However, these methods allow concurrency and duplicated curve labels. Moreover, they are heuristic and may introduce duplicated curve labels even when it is unnecessary. Since some of the existing methods (e.g., [SAA09]) yield Euler diagrams that can look distorted and stretched, EulerSmooth has been developed to smooth the curves of an existing Euler diagram [SAS16]. For achieving this goal, the algorithm applies a curve shortening flow approach. During this approach, the algorithm ensures that every set element stays within the correct zone using a force-directed edge-aware algorithm, ImpPrEd [SAAB11]. SPEULER constructs Euler diagrams where set elements are arranged using circular layouts [KGWD22]. The resulting Euler diagrams are wellformed, which is partly a result by allowing only neighboring zones with no concurrency. RectEuler represents set-like data as Euler diagrams using rectangles as enclosing curves [PKS+23]. The resulting Euler diagrams are wellformed. However, they split the diagram into multiple diagrams if necessary. Additionally, there exist approaches for drawing the set elements of set systems as glyphs within the corresponding region of the Euler diagram [Bra12].

![Figure 4: Results after different steps of our hypergraph visualization workflow.](image)

![Figure 5: Examples of wellformedness conditions.](image)
Recently, Zhou et al. [ZRPW23] presented a method for the simplification of hypergraphs and used Euler-like diagrams to visualize the results, but the simplification method does not consider criteria regarding the resulting visualization. Oliver et al. [OZZZ23] presented a method for hypergraph simplification and visualization which, however, does not use Euler diagrams. Instead, the hyperedges are represented as polygons, whose vertices are the nodes.

### 4.2. Hypergraph drawings and planar supports

In the graph drawing community, a set system is usually viewed as a hypergraph, and different types of hypergraph drawings are considered. Mäkinen [Mäk90] introduced two types of drawings, edge standard and subset standard. The latter type includes vertex-based Venn diagrams, which were defined by Pollack [JP87] and which correspond to the type of Euler diagrams that we address with our work, i.e., Euler diagrams that are based on planar supports. Later, vertex-based Venn diagrams were generalized into subdivision drawings [KvKS09]. Johnson and Pollack [JP87] also showed that it is NP-hard to decide whether a given hypergraph has a planar support. Moreover, they presented an efficient algorithm that yields a tree support if it exists, i.e., a planar support that is a tree. Following up on this study, efficient algorithms were developed that can deal with edge weights to express preferences for including certain edges in a tree support [KS03, KMN14]. Also, it is known that one can efficiently decide whether a hypergraph has a planar support that is a path [KS03] or a path-based tree support [BKM*11], i.e., a planar support that is a tree such that every hyperedge induces a connected path in it.

Another line of research has dealt with finding planar supports for hypergraphs with special properties. Brandes et al. [BCPS11] presented an efficient algorithm for hypergraphs that are closed under intersections and differences, i.e., every intersection and difference of two hyperedges is also a hyperedge. Moreover, a hypergraph has a planar support if there is a non-empty intersection of all hyperedges [CR05], there are at most eight hyperedges [VV04], or the hypergraph results from the intersection of two families of regions that have a special property, called non-piercing [RR18]. None of the existing methods, however, can decide for an arbitrary hypergraph whether it has a planar support.

Recently, van Bevern et al. [vBKK*22] studied the role of twins in computing planar supports, where two vertices are twins if they occur in exactly the same hyperedges. They showed that when replacing a set of twins by a single vertex, the hypergraph may cease to have a planar support. However, the proof by Johnson and Pollak [JP87] shows that even for hypergraphs without twins, it is NP-hard to decide whether a planar support exists. Our method for computing a planar support can deal with hypergraphs that contain twins, but we propose an optional preprocessing step to replace each set of twins by a single node to ensure that it will be represented as a single connected zone in the output Euler diagram.

### 4.3. Set visualizations

Several methods for set visualization have been developed that are related to hypergraph drawings or Euler diagrams.

**LineSets** aim at the reduction of clutter in representations of set systems [ARRC11]. They represent each set of the system as a single continuous curve connecting all set elements. Similar to LineSets, Kelp Diagrams connect every zone corresponding to the same set with a line [DvKSW12]. If two sets share the same line, they are stacked and the lower one has an increased width. Additionally, the lines contain only straight segments. KelpFusion is an extension to Kelp diagrams [MRS*13]. It fills regions that are enclosed by lines corresponding to the same set.

A different approach of set visualization is UpSet [LGS*14]. UpSet is based on a matrix visualization where sets are represented in columns and set interactions are displayed in rows. MapSets draw a region containing the elements of a set [EHKP15]. However, these elements belong to only a single set, e.g., political parties. This results in no shared regions between sets. Additionally, the geometric position of set elements is fixed. As a result, the visualization is a partition of the plane containing set elements. ClusterSets are a generalization to MapSets that also allows single edges between points of the same set [GCH*21] and sets to be split into multiple regions. Similarly, GMAP draws general graphs as geographic-like maps [GHK09]. The positions of the individual set elements are determined by the relation to adjacent set elements.

### 4.4. Integer linear programming

Integer linear programming is a general method for solving combinatorial optimization problems [NW88]. The approach is to encode a problem in the form of a set of variables, an objective function, and a set of constraints that have the form of an integer linear program (ILP), and to let a solver compute a variable assignment that is optimal under the constraints. The variable vector \( \mathbf{x} \) of an ILP contains integer variables. The objective function and the constraints of an ILP in canonical form are \( f(\mathbf{x}) = c^T \mathbf{x} \) and \( A \mathbf{x} \leq b, \mathbf{x} \geq 0 \), respectively, where \( c, A, \) and \( b \) are given as constants. An equality constraint can be encoded in this form using two inequality constraints, and a binary variable can be expressed as an integer variable with upper bound 1. Although solving ILPs requires exponential time in the worst case, there exist ILP solvers that often perform well in practice. Especially for NP-hard problems, such as SetSystemSimplification, this approach is justifiable and often successful.

In the field of information visualization, two ILP-based methods are most related to our own. Castermans et al. [TvGW*19] considered the problem of finding a support of a hypergraph with fixed vertex positions. The support is required to be a crossing-free straight-line graph; hence, a stricter requirement than planarity is enforced. This can easily be achieved by forbidding the selection of crossing pairs of candidate edges. Our situation, however, is much more involved, since we need to ensure the planarity of the support without knowing the positions of its nodes. Another ILP-based method, MosaicSets, embeds a set system into a prescribed host graph, e.g., a regular grid [RWB*23]. There, the planarity is automatically ensured by choosing a planar host.

When modeling SetSystemSimplification as an ILP, the challenge lies in expressing constraints to ensure the connectivity requirement for hyperedges and the planarity requirement. We adapt a formulation by Shirabe [Shi05] that enforces the connectivity of a region that results from the allocation of a set of faces of a planar subdivision. Our constraint formulation for planarity is adapted from
Chimani et al. [CHW19], who developed an ILP for the maximum planar subgraph problem, which for a given graph asks for a planar subgraph with the maximum number of edges.

5. Extended problem

In this section, we extend the SetSystemSimplification to model the most important wellformedness conditions, drawing upon the user study by Rodgers et al. [RZP11]. We require the connected zones wellformedness condition and consider the no-concurrency wellformedness condition in the objective function. We consider the unique curves labels’ wellformedness condition in the sense that we require that each set is represented as a connected region. However, we allow regions with holes. During the rendering process, we ensure that the Euler diagram satisfies the simplicity wellformedness condition by constructing the initial Euler diagram using a method that guarantees simple curves [RZF08]. The method that we apply for smoothing the Euler diagram [SAS16] considers the transversality wellformedness condition, but it does not strictly guarantee it. Before giving a formal definition of the extended problem, MCM-SetSystemSimplification, we provide technical details on how the input for the problem evolves from Steps 1 and 2 of our workflow.

In Step 1, which is optional, we compute for a given node-weighted hypergraph $H^* = (V^*, E^*)$ the condensed hypergraph $H = (V, E)$ and its node weighting $w$. We partition $V^*$ into clusters, such that two nodes $u, v \in V^*$ are in the same cluster if and only if $\ell(u) = \ell(v)$, meaning that $u$ and $v$ are contained in exactly the same set of hyperedges. Then, for each cluster, we introduce a corresponding node in $V$, whose weight $w(v)$ we set to the total weight of the cluster. For every hyperedge $e^* \in E^*$, we add a corresponding hyperedge $e$ to $E$, which for each node in $e^*$ contains the node representing its cluster. If we wish to skip Step 1, we simply set $H = H^*$. Only with Step 1, however, we ensure that our results fulfill the connected zones wellformedness condition.

In Step 2, based on $H = (V, E)$, we compute the superdual graph $G = (Z, F)$, from which we will select a subgraph that will serve as the dual graph of the Euler diagram. The node set of $G$ is $Z = V \cup \{v_0\}$, where $v_0$ is a special node representing the outer face of the Euler diagram. The edge set $F$ of $G$ contains two types of edges, i.e., $F = F_1 \cup F_2$, where $F_1$ and $F_2$ are defined as follows:

- The set $F_1$ contains all edges that may be useful for a planar support of $H$, that is,
  \[
  F_1 = \{ (u, v) : u, v \in V, u \neq v, \ell(u) \cap \ell(v) \neq \emptyset \}. \tag{1}
  \]
  Eq. 1 means that $F_1$ contains an edge for every two distinct nodes of $H$ that are contained together in at least one hyperedge.

- The set $F_2$ contains edges that represent possible adjacency relationships between the outer face $v_0$ and the nodes in $V$. In principle, one could include one edge $(v_0, v)$ for each node $v \in V$. It is sufficient, however, to include only those edges whose selection has an influence on the objective function. We will introduce an objective function that rewards adjacency relationships between $v_0$ and nodes that are contained in few hyperedges (ideally, in only one hyperedge). Hence, we define
  \[
  F_2 = \{ (v_0, u) : u \in V, |\ell(u)| = \min_{w \in V} |\ell(w)| \}. \tag{2}
  \]

Selecting an edge $(u, v) \in F_1$ for the dual graph causes a violation of the no-concurrency wellformedness condition if the labels $\ell(u)$ and $\ell(v)$ differ by more than one. We generally tolerate concurrency, but aim to keep their number low. Therefore, to count the number of concurrency implied with the selected dual graph, we introduce an edge weight $\omega(e)$ for every edge $e \in F_1$. This weight of an edge $e = \{u, v\}$ is set to $\omega(e) = |\ell(u) \Delta \ell(v)| - 1$. Hereby, $\Delta$ is the symmetric difference.

Using the node weighting $w$, the newly introduced edge set $F_1$ and $F_2$, and the edge weighting $\omega$, we define three objective functions used in our extended problem definition. For a resulting planar graph $G'$, the kept set elements and set memberships are quantified by $f_{\text{weight}}$. We compute the total number of concurrency of $G'$ with $f_{\text{concur}}$ and count the total number of favorable adjacency relationships between nodes and the outer face by $f_{\text{outer}}$.

Using the additional definitions, we extend the problem SetSystemSimplification to MCM-SetSystemSimplification.

Problem 2 (MCM-SetSystemSimplification) Given

- a hypergraph $H = (V, E)$,
- the superdual graph $G = (Z, F)$ for $H$, with $F = F_1 \cup F_2$,
- a node weighting $w : V \rightarrow \mathbb{R}_{\geq 0}$,
- an edge weighting $\omega : F_1 \rightarrow \mathbb{R}_{\geq 0}$, and
- parameters $\alpha, \beta \in \mathbb{R}_{\geq 0}$,

find

- a set $S \subseteq V$ and
- a planar subgraph $G' = (Z', F')$ of the superdual graph $G$ for $H$, such that $G' = (Z \setminus \{v_0\}, F_1 \cap F')$ is a planar support of the hypergraph $H - S$ and

\[
  f(G') = f_{\text{weight}}(G') - \alpha \cdot f_{\text{concur}}(G') + \beta \cdot f_{\text{outer}}(G') \tag{3}
\]

is maximized, where

\[
  f_{\text{weight}}(G') = \sum_{v \in V \setminus S} w(v), \tag{4}
\]

\[
  f_{\text{concur}}(G') = \sum_{e \in F_1 \cap F'} \omega(e), \tag{5}
\]

\[
  f_{\text{outer}}(G') = |F_2 \cap F'|. \tag{6}
\]

Here, $G'$ is the dual graph of the Euler diagram excluding the node $v_0$ representing the outer face and its incident edges. The hyperedge that corresponds to a set forms a connected component in $G'$. This ensures that for each set the corresponding region is connected, which avoids certain violations of the unique curve labels wellformedness condition, such as the one in Fig. 5f. However, we allow regions with holes. Therefore, a solution may contain multiple curves with the same label, i.e., the exterior ring and the multiple interior rings of a region. The function $f_{\text{concur}}$ evaluates only the set of edges belonging to $G'$, i.e., the set $F_1 \cap F'$. In contrast, the function $f_{\text{outer}}$ counts the number of selected edges incident to $v_0$, which equals the size of $F_2 \cap F'$. 

© 2024 The Authors.
Computer Graphics Forum published by Eurographics and John Wiley & Sons Ltd.
6. Methodology

In this section, we present our exact method and our heuristic for MCGSetSystemSimplification; see Sect. 6.1 and Sect. 6.2, respectively. Recall that both methods take the superdual graph $G = (Z, F)$ as input and return a planar subgraph $G' = (Z', F')$ of it, which will serve as the Euler diagram’s dual graph.

6.1. An integer linear program

Variables encoding the selected subgraph For each node $u$ of the superdual graph $G = (Z, F)$, we define a binary variable $x_u$ indicating whether the node is part of the selected subgraph $G'$ $(x_u = 1)$. 
\[ x_u \in \{0, 1\} \quad \forall u \in Z \tag{7} \]

Moreover, we introduce one binary variable $z_{u,v}$ for each edge $e = \{u, v\} \in F$. This variable indicates whether $e$ is selected for $G'$. 
\[ z_{u,v} \in \{0, 1\} \quad \forall \{u, v\} \in F \tag{8} \]

These $z$-variables are coupled with the $x$-variables to ensure that an edge can be selected only if its two incident nodes are selected. 
\[ z_{u,v} \leq x_u \quad \text{and} \quad z_{u,v} \leq x_v \quad \forall \{u, v\} \in F \tag{9} \]

Objective The $x$- and $z$-variables introduced in the previous section are sufficient to express all three objectives of MCGSetSystemSimplification.

Maximize $\sum_{p \in E} w(p) \cdot x_p - \alpha \sum_{\{u,v\} \in F_1} \omega(u,v) \cdot z_{u,v} + \beta \sum_{\{u,v\} \in F_2} z_{u,v} \tag{10}$

Enforcing connectivity of hyperedges To ensure the connectivity of hyperedges in the selected subgraph $G'$ of $G$, we adopt a flow model by Shirabe [Shi05] for spatial unit allocation tasks occurring in spatial planning. In our application, we apply the model separately to each hyperedge $X$ in $E$. For this, we introduce a directed graph $\tilde{G}_X = (Z_X, A_X)$; see Fig. 6. The node set of $\tilde{G}_X$ contains every node of $G$ that is labeled with $X$, i.e., 
\[ Z_X = \{u \in Z \mid X \in \ell(u)\}. \tag{11} \]

The arc set of $\tilde{G}_X$ contains two opposite arcs for each edge of $G$ that connects two nodes labeled with $X$, i.e., 
\[ A_X = \{(u,v), (v,u) \mid \{u,v\} \in F, X \in \ell(u), X \in \ell(v)\}. \tag{12} \]

Due to the definition of $G$, $\tilde{G}_X$ is a complete, directed graph.

Figure 7 illustrates the flow model that ensures the connectivity of the subset $\{u \in X \mid x_u = 1\}$ selected from $X$. The flow in $\tilde{G}_X$ is represented with the following variables:
\[ y_{u,v} \in \{0, \ldots, |X| - 1\} \quad \forall X \in E, \forall (u,v) \in A_X. \tag{13} \]

These variables are constrained such that only arcs corresponding to edges of $G$ that are selected for $G'$ can carry flow. 
\[ y_{u,v} \leq (|X| - 1) \cdot z_{u,v} \quad \forall (u,v) \in A_X, \forall X \in E \tag{14} \]

Another set of variables models which node acts as a sink of the flow network for $X$. 
\[ c^X_u \in \{0, 1\} \quad \forall X \in E, \forall u \in Z_X \tag{15} \]

These variables are constrained such that the network for $X$ can contain at most one sink (node $e$ in Fig. 7, with $c^X_e = 1$). 
\[ \sum_{u \in Z_X} c^X_u \leq 1 \quad \forall X \in E \tag{16} \]

Finally, the next two constraints ensure that
- every selected node except the sink contributes at least one unit of flow to the network (i.e., nodes $b$, $c$, $d$, in Fig. 7),
- the sink (i.e., node $e$ in Fig. 7) is allowed to receive as much flow as there are nodes in the network, and
- every non-selected node receives no flow (i.e., node $a$ in Fig. 7). 
\[ \sum_{(v,w) \in A_X} y_{w,v}^X - \sum_{(u,v) \in A_X} y_{u,v}^X \geq x_v - |X| \cdot c^X_e \quad \forall X \in E, \forall v \in Z_X \tag{17} \]
\[ \sum_{(u,v) \in A_X} y_{u,v}^X \leq (|X| - 1) \cdot x_v \quad \forall X \in E, \forall v \in Z_X \tag{18} \]

To summarize, every flow that a node contributes to the network for a hyperedge $X$ has to reach the sink and can pass through nodes selected for $X$. The presented model up to now is sufficient to ensure that the nodes selected for a hyperedge $X$ induce a connected subgraph in the selected subgraph $G'$ of $G$.

Enforcing planarity A graph is planar if and only if it does not contain a Kuratowski subdivision as a subgraph (i.e., a subdivision of $K_5$ or $K_3$) [Kur30]. Hence, we can ensure the planarity of the output graph $G'$ by requiring that, for each Kuratowski subdivision $K$ contained in $G$, at least one edge is unselected. For this, we add the constraint
\[ \sum_{\{u,v\} \in F(K)} z_{u,v} \leq |F(K)| - 1 \quad \forall K \in K(G), \tag{19} \]

where $K(G)$ is the set of all Kuratowski subdivisions contained in $G$, and $F(K)$ is the set of edges of a graph $K$.

A challenge with this formulation is that there can be exponentially many Kuratowski subdivisions in $G$. Therefore, setting up the ILP with all the $|K(G)|$ instances of Constraint 19 is prohibitive in practice. To tackle this challenge, we initially set up the ILP without Constraint 19, but we make sure that the solver detects relevant instances of Constraint 19 during the optimization process and adds them to the model. We implement this idea with a callback, which is a customized method that the solver automatically invokes at certain states of the optimization process. The callback that we introduce is invoked whenever the solver finds a new incumbent solution, i.e., a solution that satisfies all constraints of the current

Figure 6: The directed graph $\tilde{G}_X$ for a hyperedge $X$.

Figure 7: A feasible assignment of the variables modeling the connectivity of the selected subgraph for $X$. © 2024 The Authors. Computer Graphics Forum published by Eurographics and John Wiley & Sons Ltd.
model and that is better than any solution found before. For the incumbent solution that led to the invocation of the callback, let \( G' \) be the subgraph of \( G \) as determined with the \( z \)-variables. We apply the Boyer-Myrvold’s planarity testing algorithm [BM04] to \( G' \), which either reports that \( G' \) is planar or returns a Kuratowski subdivision \( K \) of \( G' \), thus proving that \( G' \) is non-planar. If \( G' \) is planar, we simply resume solving the current model, since there may be better solutions than \( G' \) that the solver has not found yet. Else, we instantiate Constraint 19 with the found Kuratowski subdivision \( K \) and add this instance of the constraint to the model, before resuming the solution procedure. This strategy is supported by state-of-the-art ILP solvers in a manner that guarantees an optimal solution as output.

To strengthen our initial model without Constraint 19, we add the following constraint.

\[
\sum_{\{u,v\} \in F} c_{uv} \leq 3 \cdot \sum_{p \in \mathcal{Z}} x_p - 6 \tag{20}
\]

This inequality does not ensure planarity but, due to Euler’s formula, is valid for all solutions satisfying the planarity requirement.

6.2. Heuristic approach

We now present our heuristic for selecting a planar subgraph \( G' = (Z', F') \) of the superdual graph \( G = (Z, F) \). The heuristic initializes \( G' \) with the node representing the outer face as the only node, i.e., \( Z' = \{v_0\} \) and \( F' = \emptyset \). Then, it lets \( G' \) grow in an iterative and greedy manner, ensuring that after every iteration the connectivity requirement for hyperedges and the planarity requirement are satisfied. Thus, the heuristic guarantees a feasible solution to MGSetSystemSimplification, but not an optimal solution.

Algorithm 1 describes the heuristic in more detail. The iterative growth is implemented with a while loop, where in each iteration the method ADDNEXTNODE is called. The method returns true or false, depending on whether it succeeded to grow \( G' \). If the method was unsuccessful, the algorithm terminates and returns \( G' \).

The method ADDNEXTNODE is presented in Algorithm 2. It first computes a list \( C \) of candidates, where a candidate \( c \) is an extension of \( G' \) by one node \( c.v \in Z \setminus Z' \) and a set of edges \( c\text{.edges} \) between \( c.v \) and nodes of \( G' \). The candidate list \( C \) is obtained as the union of multiple lists, each of which is computed based on one edge \( \{p, v\} \in F \) with \( p \in Z' \) and \( v \in Z \setminus Z' \), using a method CREATEEDGECANDIDATES. This method returns only candidates whose selection satisfies the connectivity requirement (as we will later explain). After \( C \) has been set up, we compute for each candidate \( c \in C \) the increase in the objective value that its selection would cause and store it as \( c.w \). Finally, we go through the candidates in decreasing order of \( c.w \) and choose the first candidate that does not violate the planarity requirement. For planarity testing, we use the Boyer-Myrvold algorithm [BM04], as in our ILP-based method.

The method CREATEEDGECANDIDATES yields for a given edge \( e = \{p, v\} \) a set of candidates; see Algorithm 3. For every candidate \( c \) in this set, \( c\text{.v} = v \) and \( c\text{.edges} \) contains \( e \). However, the algorithm returns multiple candidates, which differ with respect to the edges in \( c\text{.edges} \) in addition to \( e \). To compute this set, let \( X_1, \ldots, X_k \) be the hyperedges that contain \( v \) and whose connectivity would not be achieved when selecting \( v \) with edge \( e = \{p, v\} \) alone. For example, in Fig. 8, this holds for the hyperedge \( a \). For each such hyperedge, the connectivity can be repaired with a single edge among the edges incident to \( v \) other than \( e \), whose number is \( \deg(v) - 1 \).

Hence, we generate \( O(\deg(v) - 1)^K \) candidates, where \( K < |\ell(v)| \). In Fig. 8, this yields two candidates, visualized with different colors. Explicitly enumerating the candidates can be done reason-

\begin{algorithm}
\caption{Heuristic set system simplification}
\begin{algorithmic}[1]
\Require Undirected super dual graph \( G = (Z, F) \) with node weights \( w \) and edge weights \( \omega \)
\Procedure{Simplify}{$G$}
\State \( Z' \leftarrow \{v_0\}, F' \leftarrow \emptyset, G' \leftarrow (Z', F') \)
\State \( s \leftarrow \text{True} \)
\While{\( s \)}
\State \( C \leftarrow \text{empty list of candidates} \)
\For{\( \{p, v\} \in F \) such that \( p \in Z' \) and \( v \not\in Z' \)}
\State \( C_{pv} \leftarrow \text{CREATEEDGECANDIDATES}(G', G, p, v) \)
\EndFor
\State \( C \leftarrow C \cup C_{pv} \)
\State \( s \leftarrow \text{ADDNEXTNODE}(G', G) \)
\State return \( G' = (Z', F') \)
\EndWhile
\EndProcedure
\end{algorithmic}
\end{algorithm}

\begin{algorithm}
\caption{Greedy addition of the next node together with edges connecting it to the current graph}
\begin{algorithmic}[1]
\Require Current selected dual graph \( G' = (Z', F') \), super dual graph \( G = (Z, F) \)
\Procedure{AddNextNode}{$G', G$}
\State \( C \leftarrow \text{empty list of candidates} \)
\For{\( \{p, v\} \in F \) such that \( p \in Z' \) and \( v \not\in Z' \)}
\State \( C_{pv} \leftarrow \text{CREATEEDGECANDIDATES}(G', G, p, v) \)
\EndFor
\State \( C \leftarrow C \cup C_{pv} \)
\State \( s \leftarrow \text{True} \)
\While{\( C \text{.size} > 0 \)}
\State \( c \leftarrow C\text{.pollFirst()} \)
\State Add node \( c\text{.v} \) to \( Z' \) and edges \( c\text{.edges} \) to \( F' \)
\If{isPlanar($G'$)}
\State return True
\Else
\State Remove \( c\text{.v} \) from \( Z' \) and \( c\text{.edges} \) from \( F' \)
\EndIf
\EndWhile
\State return False
\EndProcedure
\end{algorithmic}
\end{algorithm}

Figure 8: Graph \( G' \) (black) after adding three nodes and three edges with our heuristic. Node \( a \cup b \) has not been added yet (left). Calling CREATEEDGECANDIDATES for edge \( \{a, ab\} \) yields two candidates: The red candidate with edges \( \{a, ac\}, \{b, ab\} \) and the blue candidate \( \{ac, a\}, \{b, ab\} \) (right).
The code and datasets are available under a GPL open source license from https://gitlab.igg.uni-bonn.de/geoinfo/generating-euler-diagrams.

<table>
<thead>
<tr>
<th>dataset</th>
<th>#set systems</th>
<th>#sets</th>
<th>#elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>MovieDB</td>
<td>569</td>
<td>Avg</td>
<td>4.39</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>44.30</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Max</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>288</td>
</tr>
<tr>
<td>TwitterCircles</td>
<td>281</td>
<td>Avg</td>
<td>6.19</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>73.83</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Max</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>197</td>
</tr>
</tbody>
</table>

Table 1: Statistics of MovieDB and TwitterCircles.

8. Experiments

Datasets We evaluate our methods with two datasets.

- MovieDB from the 2007 InfoVis contest [KJKC]: Each set system is based on a single director. The sets are the movies of the director. The set elements are actors of the corresponding movie.
- TwitterCircles [LK14]: The dataset consists of users and their interest groups. Each set system is based on a user’s ego network. Each set is based on the user’s interest circles while the set elements are followers belonging to at least one interest circle.

In total, there are 930 set systems in TwitterCircles and 16884 in MovieDB. However, we omit all set systems that have less than five nodes in addition to $v_0$ in the superdual graph since those always have a planar embedding, simply because they cannot have a $K_5$ or a $K_{3,3}$ as a minor. Thus, we select 281 and 569 set systems, respectively. Table 1 provides further details. For all experiments, we defined the weight of an element as the number of sets containing it, and we applied the optional Step 1, i.e., the condensation of the hypergraph. All experiments were executed on an AMD Ryzen 9 7950X with 64GB of RAM using Java 11 and Gurobi 9.5.1 for solving the LPs.

Before analyzing the results, we show the full workflow on the set system for the director Keith Hooker from MovieDB in Fig. 9. The set systems consists of 5 sets with 19 set elements. Our ILP approach produces an Euler diagram where each single-labeled face is adjacent to the outer face. Additionally, the minimization of concurrencies results in a nested structure. Nodes with only a few labels are placed near the outer face. In contrast, regions with a higher label count are moved toward the center of the Euler diagram.

Parameter influence To study the influence of the parameters $\alpha$ and $\beta$, we computed multiple solutions for all set systems from MovieDB using our ILP approach. For $\beta = 0.1$ and $\alpha = 0.01$ the average number of concurrencies was 1.43 and all set elements were selected. Keeping $\beta$ fixed and increasing $\alpha$ to 1.25 reduces the average number of concurrencies per Euler diagram by 54.98%, whereas the weight of the selected set elements decreases by 0.42%.

Our goal is to demonstrate the complexity of the datasets and the advantages of our simplification through a single example. Therefore, the director Joel Schoenbach is shown in Fig. 10. This example shows that our approach can simplify the set system to reduce concurrency. However, this comes at the cost of the removal of elements. We show additional examples in the supplemental material.

Similarly, we investigate the influence of the parameter $\beta$. We compute two solutions for the same set system; see Fig. 11. Both solutions were obtained with $\alpha = 0.01$. On the left, we set $\beta = 0.1$. On the right, we set $\beta = 0$, which results in only a single node being adjacent to the outer face. The latter leads to a hardly readable visualization. The sets are nested and form holes within the set that is connected to the outer face. Comparing both Euler diagrams, it is harder to follow individual set outlines for $\beta = 0$. As a result, it would be difficult to determine labels within the interior of the diagram when only a single label for each curve is given.

Comparison of solutions of exact method and heuristic We now compare the results of the ILP and of the heuristic for the 850 set systems given with the two datasets. For this purpose, we evaluate the value of the overall objective function $f$ and the functions $f_{\text{weight}}$, $f_{\text{conc}}$, and $f_{\text{outer}}$ expressing the three criteria of MCSetsystem-Simplification. Based on our experiences with the different parameter settings discussed in the previous section, we fixed $\alpha = 0.01$ and $\beta = 0.1$ for all the experiments that we report here.

We use a time limit of one hour for the ILP solver and report the best solution. For all 850 set systems, the solver found a solution within the time limit, but in four examples from TwitterCircles the solver was not able to prove the optimality of the solution. In three cases, the solution contained 100% of the total weight of the set system. In one example (id = 779715), a solution with 50.82% of the total weight was returned and a gap of 96.64% was reported, meaning that there might be a solution whose objective value is roughly twice as high as that of the found solution.

Across all instances, we observe that the heuristic performs worse than the ILP, in the sense that it achieves a lower value for the objective function $f$ (see the column $f$ in Table 2 for MovieDB and Table 3 for TwitterCircles). This is mainly due to two factors: the heuristic includes less node weight in the dual graph (column $f_{\text{weight}}$) and selects edges with more concurrency (column $f_{\text{conc}}$).

For all instances in MovieDB, our ILP approach found a planar
support with all set elements. In contrast, the heuristic did not include all set elements in every solution. The average value of $f_{\text{weight}}$ is 0.05% less with the heuristic than with the ILP. For the instances in TwitterCircles, the ILP in a single case yielded a solution without all set elements, and the average value of $f_{\text{weight}}$ is 0.16% less with the heuristic than with the ILP.

To evaluate the results with respect to concurrent curves, we compare the average values of $f_{\text{concur}}$ of our approaches. For TwitterCircles, $f_{\text{concur}}$ increased from 2.02 to 4.12. For MovieDB, $f_{\text{concur}}$ increased from 1.43 to 2.78. The increased number of concurrencies is also reflected by the maximum values of $f_{\text{concur}}$, which increased from 89 to 111 and from 55 to 90, respectively.

We investigate the distribution of the increased number of concurrencies by the factor of two in Fig. 12. None of the instances drastically exceeds this factor. The additional concurrencies are...
8. Conclusion

We have proposed a novel ILP approach for finding a planar support of a given hypergraph that represents a set system. We have integrated this approach into a complete workflow for generating Euler diagrams that represent each set as a single connected region. The ILP maximizes the preserved number of set elements or set memberships. Additionally, it minimizes the total number of concurrent curves and places a maximum number of faces contained in only few curves adjacent to the outer face. Moreover, we have developed a heuristic that tries to optimize the same objective by greedily adding nodes to an empty graph. Our experiments show that our ILP can be used to produce optimal general Euler diagrams that comply with our well-formedness criteria. Moreover, our approach can simplify a set system in order to reduce concurrency.

Comparisons of the heuristic with the ILP show that the heuristic produces results with similar objective values while increasing the concurrency by a factor of two. A benefit of the heuristic is that it needs only a fraction of the processing time.

For future work, producing Euler diagrams without concurrency by either limiting the edges within the supergraph or merging sets might be possible. When concurrency cannot be avoided, equally distributing concurrency across edges instead of accumulating the total concurrency in a single edge is favorable. To improve readability, we could consider additional well-formedness conditions, such as the avoidance of triple points. With regards to the heuristic, the computation of candidates can be improved to avoid the brute force approach. As new nodes are added to the dual graph, the algorithm must update existing candidates. Additionally, excluding certain edges from candidate creation might be beneficial. Our ILP could be accelerated, e.g., by generating multiple Kuratowski subdivisions at once, instead of one by one. It may also be possible to explore ways of displaying removed set elements and indicating the lost information using visual mechanisms, such as texture, shading or icons. Furthermore, we would like to encode the degree of simplification and completeness of the Euler diagram using visualizations for quantitative information, such as area proportional layouts or applying a color scale.

Acknowledgements. For the purpose of open access, the authors have applied a Creative Commons Attribution (CC-BY) license to any Author Accepted Manuscript version arising from this submission. We thank Dagstuhl Seminar 22462 “Set Visualization and Uncertainty” (November 13-18, 2022) for initializing this research. BW was partially supported by grants DOE DE-SC0021015 and DOE DE-SC0021016.
References


© 2024 The Authors.
Computer Graphics Forum published by Eurographics and John Wiley & Sons Ltd.


© 2024 The Authors.
Computer Graphics Forum published by Eurographics and John Wiley & Sons Ltd.