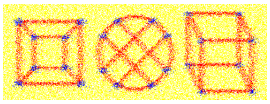
Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 27, no. 9, pp. 783–802 (2023)
DOI: 10.7155/jgaa.00645

Drawing Order Diagrams Through Two-Dimension Extension

Dominik Dürrschnabel[1] *Tom Hanika*[2] *Gerd Stumme*[1]

[1]Knowledge & Data Engineering Group, Research Center for Information System Design & Department of Electrical Engineering and Computer Science, University of Kassel, Germany
[2]Institute of Computer Science, University of Hildesheim, Germany
Berlin School of Library and Information Science, Humboldt-Universität zu Berlin, Germany

Submitted: December 2022	Reviewed: April 2023	Revised: June 2023
Accepted: November 2023	Final: November 2023	Published: December 2023
Article type: Regular Paper	Communicated by: C. D. Tóth	

Abstract. Ordinal real-world data such as concept hierarchies, ontologies, genealogies, or task dependencies in scheduling often has the property to not only contain pairwise comparable, but also incomparable elements. Order diagrams provide an important tool for understanding and thus discovering knowledge in such data. Easily readable drawings of such order diagrams are hard to come by, even for small ordered sets. Many attempts were made to transfer classical graph drawing approaches to order diagrams. Although these methods produce satisfying results for some ordered sets, they unfortunately perform poorly in general. In this work, we present the novel algorithm `DimDraw` to decompose an ordered set (e.g., a concept hierarchy) in linear orders and to produce a corresponding order diagram. This algorithm is based on a relation between the dimension of an ordered set and the bipartiteness of its transitive incompatibility graph. To evaluate the quality of the algorithm, a user study was conducted where generated drawings were compared with ones from state-of-the-art drawing algorithms.

1 Introduction

Order is — together with proximity/distance — the predominant paradigm for humans to organize their physical and social environment. The exploration of order relations is thus an important knowledge discovery task. Ordered sets are used for modeling concept hierarchies in ontologies and track dependencies in scheduling. *Order diagrams*, also called *line diagrams* or *Hasse diagrams*,

E-mail addresses: duerrschnabel@cs.uni-kassel.de (Dominik Dürrschnabel) tom.hanika@uni-hildesheim.de (Tom Hanika) stumme@cs.uni-kassel.de (Gerd Stumme)

are a great tool for visualizing the underlying structure of ordered sets. In particular, they enable the reader to explore and interpret complex information through a compact representation.

In order diagrams every element is visualized by a point on the plane. Each edge of the covering relation is visualized as an ascending line connecting its points; and these lines are not allowed to touch other points. These strong requirements are often complemented with further soft conditions to improve the readability of diagrams. Examples of such conditions are the minimization of the number of crossing lines or of the number of different slopes. Another desirable condition is to draw as many chains as possible on straight lines to indicate structural dependencies. Lastly, the distance of points to (non-incident) lines should be maximized for improved readability.

Experience shows that in order to obtain (human) readable drawings one has to balance those criteria. Based on this, there are algorithms that produce drawings optimizing towards some of the criteria mentioned above. Drawings produced by such algorithms are sufficient to some extent. Still, they may not compete with those created manually by an experienced human. However, such experts are often not available, too expensive, or not efficient enough to create large numbers of order diagrams. Hence, finding efficient algorithms that visualize order diagrams at a suitable quality is still an open task.

An exemplary field of research in this domain is formal concept Analysis (FCA) [13], a theory that can be used to extract knowledge from data through clustering and ordering it. In this work we present a novel approach that does not employ the optimization techniques as described above. For this we make use of the structure and its properties that are already encapsulated in the ordered set. We base our idea on the observation that ordered sets of order dimension two can be embedded into the plane in a natural way. Building up on this, we show a procedure to embed the ordered sets of order dimension three and above by reducing them to the two-dimensional case. To this end we prove an essential fact about inclusion-maximal bipartite induced subgraphs in this realm. Based on this we link the naturally emerging NP-hard computation to an instance of the SAT problem. Our main contribution with respect to this is Theorem 6 (see Section 5).

We investigate our theoretical result on different real-world datasets using the just introduced algorithm `DimDraw`. The ordered sets used for this experiment arise from the theory of FCA and are evaluated through a user study of domain experts, as those are experienced in working with order diagrams. In this user study, the visualizations generated by the algorithm are compared with two classical approaches of automated order diagram drawing. Furthermore, we note how to incorporate heuristical approaches replacing the SAT solver for faster computations. Finally, we discuss surprising observations and formulate open questions.

2 Related Work

Order diagrams can be considered as acyclic (intransitive) digraphs that are drawn upward in the plane, i.e., every arc is a curve monotonically increasing in the y-direction. Much research has been conducted for such upward drawings. A frequently employed algorithm framework to draw such graphs is known as Sugiyama's framework [26]. This algorithm first divides the set of vertices of a graph into different layers, then embeds each layer on the same y-coordinate and minimizes crossings between consecutive layers. Crossing minimization can be a fundamental aesthetic for upward drawings. However, the underlying decision problem is known to be NP-hard even for the case of two-layered graphs [7]. A heuristic for crossing reduction can be found in [8]. The special case for drawing rooted trees can be solved using divide-and-conquer algorithms [23]. Such divide-and-conquer strategies can also be used for non-trees as shown in [22]. Several algorithms were developed to work directly on order diagrams. Relevant for our work is the dominance drawing

approach. There, comparable elements of the order relation are placed such that both Cartesian coordinates of one element are greater than the ones of the other [15]. This, combined with lattice-completions, is employed in [10] to draw ordered sets. Weak dominance drawings allow a certain number of elements that are placed as if they were comparable [17] even when they are not. Our approach is based on this idea. Previous attempts to develop heuristics are described in [18]. In [16] an algorithm is described that leverages weak dominance drawings to generate well-readable orthogonal drawings. Weak dominance drawings can also be used as an index structure [28] to provide reachability information in graphs. If an ordered set is a lattice there are algorithms that make use of the structure provided by this to compute drawings. The authors in [25] make use of geometrical representations for drawings of lattices. In [11] a force directed approach is employed, together with a rank function to guarantee that the "upward property" is preserved. A focus on additive diagrams is laid out in [12].

3 Notation and Definitions

We start by recollecting notation and notions from order theory [27]. In this work we call a pair (X, R) an *ordered set*, if $R \subseteq X \times X$ is an *order relation* on a set X, i.e., R is reflexive, antisymmetric, and transitive. In this setting X is called the *ground set* of (X, R). In some cases, we write $(X, \leq)$ instead of (X, R), to use the notation $(a, b) \in \leq$, $a \leq b$ and $b \geq a$ interchangeably. We write $a < b$ iff $a \leq b$ and $a \neq b$. Alike, if $b \geq a$ and $b \neq a$, we write $b > a$. We say that a pair $(a, b) \in X \times X$ is *comparable*, if $a \leq b$ or $b \leq a$, otherwise it is *incomparable*. An order relation L on X is called *linear* (or total) if all elements of X are pairwise comparable. For $(X, \leq)$ L called a *linear extension* of $\leq$ iff L is linear and $\leq \subseteq L$. If $\mathcal{R}$ is a family of linear extensions of $\leq$ and $\leq = \bigcap_{L \in \mathcal{R}} L$, we call $\mathcal{R}$ a *realizer* of $\leq$. The minimal d such that there is a realizer of cardinality d for $(X, \leq)$ is called its *order dimension*. We use the denotation of order dimension for ordered sets and order relations interchangeably. For a set X and $\mathcal{C} \subseteq X \times X$ we denote $\mathcal{C}^{-1} := \{(a, b) \mid (b, a) \in \mathcal{C}\}$. The *transitive closure* of $R \subseteq X \times X$ is denoted by R^+.

For our work, we consider simple graphs denoted by (V, E), where $E \subseteq \binom{V}{2}$. For an ordered set $(X, \leq)$, its *comparability graph* is defined as the undirected graph (X, E), such that $\{a, b\} \in E$, if and only if $a, b \in X$ are comparable. Similarly, the *co-comparability graph* (sometimes called incomparability graph) is the undirected graph on X where $\{a, b\}$ is an edge if and only if $a, b \in X$ are incomparable. The comparability graph and co-comparability graph are thus complementary to each other, i.e., they are defined on the same vertex set and one has exactly the edges that the other lacks. They depict the comparable or incomparable element pairs of the order respectively. Two order relations on the same ground set are called *conjugate* to each other, if the comparability graph of one is the co-comparability graph of the other one. Thus, if an order has a comparable pair (x, y), its conjugate order is incomparable on the same pair and vice versa. We refrain from a formal definition of the term *drawing* of $(X, \leq)$. However, we need to discuss the elements of such drawings used in our work. Each element of the ground set is drawn as a point on the plane. The *covering relation* is defined as $\mathrm{cr}(X, \leq) := \{(a, b) \in \leq \mid \nexists c \in X : a < c < b\}$ which is the transitive reduction of the order relation. It thus contains the pairs of the ordered set that do not follow from the transitivity of other pairs. In the drawing, each element of the covering relation is drawn as a monotonically increasing curve connecting its points.

From here on some definitions are less common. For $(X, \leq)$ we denote the set of incomparable pairs of elements by $\mathrm{inc}(X, \leq)$. Note that the set of incomparable pairs contains for each pair (a, b) also the pair (b, a). We call two incomparable pairs $(a, b), (c, d) \in \mathrm{inc}(X, \leq)$ *incompatible*, if their addition to $\leq$ creates a cycle in the emerging relation, i.e., if there is some sequence of

elements $c_1, \ldots, c_n \in X$, such that each pair $(c_i, c_{i+1}) \in \leq \cup \{(a,b),(c,d)\}$ with $i \in \{1,\ldots,n\}$ and $c_{n+1} = c_1$. The *transitive incompatibility graph* is defined as the undirected graph $((\mathrm{inc}(X,\leq), E)$ with $\{(a,b),(c,d)\} \in E$ iff (a,b) and (c,d) are incompatible. We denote this graph by $\mathrm{tig}(X,\leq)$. An example for an ordered set with its transitive incompatibility graph is given in Figure 1. Here, for example, the relation $\leq \cup \{(4,3),(6,4)\}$ contains the cycle $3,6,4$, and thus the pairs $(4,3)$ and $(6,4)$ are incompatible. Therefore, they are connected by an edge in the transitive incompatibility graph. We say a pair $(a,b) \in \mathrm{inc}(X,\leq)$ *enforces* another pair $(c,d) \in \mathrm{inc}(X,\leq)$ iff $(c,d) \in (\leq \cup \{(a,b)\})^+$. If (a,b) enforces (c,d) in $\leq$, we use the notation $(a,b) \to (c,d)$. In the ordered set of Figure 1, it holds that $(4,6) \to (1,6)$, as $(1,6) \in (\leq \cup \{(4,6)\})^+$.

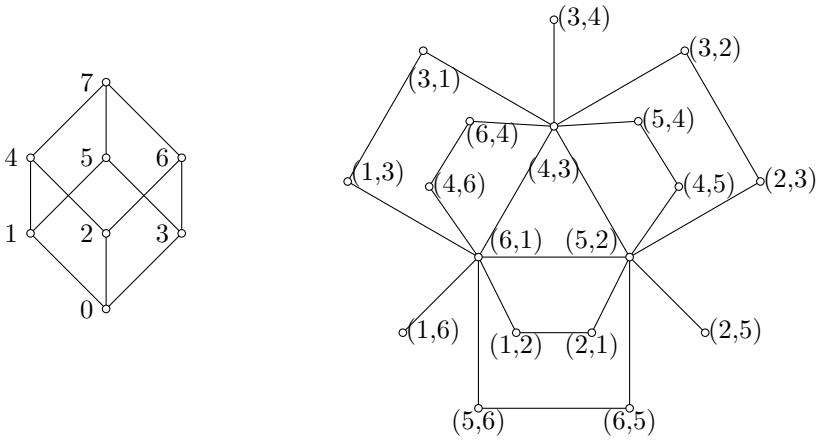

Figure 1: *Left:* the ordered set B_3. *Right:* its transitive incompatibility graph (tig). This graph has the incomparable pairs of B_3 as its vertices. Incompatible pairs are connected by an edge.

4 Drawing Ordered Sets of Dimension Two

Ordered sets of order dimension 2 have a natural way to be visualized using a realizer by their dominance drawings [15]. Let $(X,\leq)$ be an ordered set of dimension two. First define the *position* for each $x \in X$ in a linear extension L as the number of vertices that are smaller, i.e., $\mathrm{pos}_L(x) \coloneqq |\{y \in X \mid y < x\}|$. Thus, it holds for two elements x and y with $x \neq y$ that $\mathrm{pos}_L(x) < \mathrm{pos}_L(y)$ if and only if $(x,y) \in L$.

Now let $\mathcal{R} = \{\leq_1, \leq_2\}$ be a realizer consisting of two linear extensions of the order relation $\leq$. Each element is embedded into a two-dimensional grid at the pair of coordinates $(\mathrm{pos}_{\leq_1}(x), \mathrm{pos}_{\leq_2}(x))$. Embedding this grid into the plane is done using the generating vector $(-1,1)$ for x_1 and $(1,1)$ for x_2. The coordinate system is depicted in Figure 2, the ticks correspond to the positions of the elements.

Each embedded element point then divides the plane into four quadrants using the two lines that are parallel to x_1 and x_2. It holds that $a < b$, if and only if the point b is in the quadrant above the point a. For the generated drawing, the lines of the covering relation are drawn as straight lines. This guarantees that they are monotonically increasing curves.

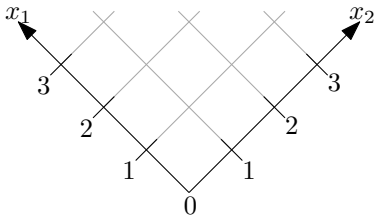

Figure 2: The grid an order diagram is embedded into.

In order to compute such drawings we first have to check whether a dominance drawing for a given ordered set exists. The following theorem gives a characterization for orders with this property.

Theorem 1 (Dushnik and Miller, 1941 [3])**.** *The dimension of an ordered set $(X, \leq)$ is at most 2, if and only if there is a conjugate order $\leq_C$ on X. A realizer of $(X, \leq)$ is given by $\mathcal{R} = \{\leq \cup \leq_C, \leq \cup \geq_C\}$ where $\geq_C$ is defined as the relation $\{(a, b) \subseteq X \times X \mid (b, a) \in \leq_C\}$.*

This theorem does not only provide a connection between the existence of a conjugate order and its two-dimensionality, it also gives a way to compute the realizer if the conjugate order is known. The computation of the conjugate order is possible in polynomial time because of an algorithm proposed by Golumbic in 1977 [14]. This algorithm checks whether a graph has a transitive orientation, i.e., whether there is an order on its vertices, such that the graph is exactly the comparability graph of this order. It computes such an order, if it exists. This algorithm runs in $\mathcal{O}(n^3)$, with n being the number of vertices of the graph. When this algorithm is executed on the co-comparability graph of an ordered set, it computes its conjugate order. This, together with the theorem by Dushnik and Miller from above, thus composes an algorithm for an ordered set to check if it is two-dimensional and compute the realizer in this case. For the sake of completeness, note that there are faster algorithms (as fast as linear [21]) for computing transitive orientations. However, those only work if the graph is actually transitive orientable and return erroneous results otherwise. On a final note, deciding dimension three or larger is known to be NP-complete as shown by Yannakis in 1982 [29]. This is in contrast to the just stated fact about two-dimensional orders.

5 Drawing Ordered Sets of Higher Dimensions

The idea of embedding two-dimensional orders does generalize in a natural way to higher dimensions, i.e., gives us a nice way to embed n-dimensional ordered sets into n-dimensional space (Euclidean space) by using an n-dimensional realizer. However, this approach turns out to result in two problems. Deciding the order dimension of an ordered set is NP-complete for ordered sets of dimensions 3 or more. This results in the fact that computing realizers of such ordered sets is computationally hard. Furthermore, this results in an embedding of the ordered set in a higher-dimensional space while we are interested in embeddings into only two dimensions. One could try to project the so-generated higher-dimensional drawings into two dimensions. However, our experiments in this direction were not successful. See, for example, the parallel-projections of an ordered set in example Figure 3, which are all not readable to a sufficient degree. For this reason our algorithm makes use of a different method to compute higher dimensional drawings that is still based on the concept of using the realizer.

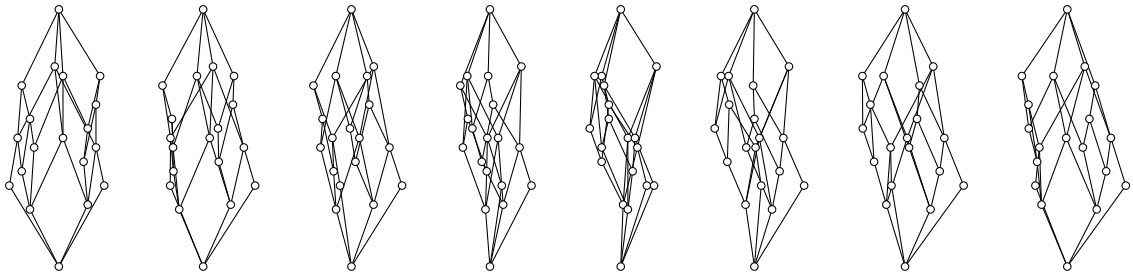

Figure 3: A three-dimensional ordered set embedded — based on its realizer — into three-dimensional Euclidean space and then projected into the plane using a parallel projection from multiple angles. Even though the structure of the ordered set is recognizable, the drawings are all not satisfactory. For better drawings of the same ordered set refer to Figure 7.

In short, the main idea of this section is the following: for a given order relation we want to insert some number of additional pairs to make it two-dimensional. This allows the resulting order to be drawn using the algorithm for the two-dimensional case described in the previous section. Afterwards we remove all the inserted pairs. By construction, the property that if $a < b$ in the original order, the point a is inserted in the quadrant below b is still preserved. The quadrant below a hereby refers to area that is spanned by the lines parallel to the axes of the coordinate system. Such drawings are sometimes called *weak dominance drawings* [17]. However, for each inserted pair (a, b), we obtain two points in the drawing that are drawn as if a and b were comparable. This poses the question of minimizing the number of inserted pairs.

Definition 2. Let $(X, \leq)$ be an ordered set. A set $\mathcal{C} \subseteq \mathrm{inc}(X)$ is called a *two-dimension-extension* of $(X, \leq)$ iff $\leq \cup\, \mathcal{C}$ is an order on X and the ordered set $(X, \leq \cup\, \mathcal{C})$ is two-dimensional.

A two-dimension extension exists for every ordered set $(X, \leq)$ of higher dimension. To see this, consider a linear extension $\leq_L$ of the order $\leq$. For some incomparable pair (a, b) with respect to $\leq$, the relation $\leq_L \setminus \{(a, b)\}$ is a superset of $\leq$ and has order dimension two. Thus, it is a two-dimension extension of $\leq$.

It is known to be NP-complete to decide whether an ordered set can be altered to be two-dimensional by inserting k pairs [1]. Hence, we propose an algorithm that tackles the problem for approximating the corresponding optimization problem using an idea that is based on the following theorem.

Theorem 3 (Doignon et al. 1984 [2])**.** *The ordered set* $(X, \leq)$ *has order-dimension at most two if and only if* $\mathrm{tig}(X, \leq)$ *is bipartite.*

Thus, for $(X, \leq)$ of dimension greater than two, $\mathrm{tig}(X, \leq)$ is non-bipartite. Consider for example Figure 1. Because of the cycle of length 3 given by the pairs $(4, 3)$, $(6, 1)$ and $(5, 2)$ in its transitive incompatibility graph, it cannot have dimension two. Our approach is therefore to find a maximal induced bipartite subgraph in $\mathrm{tig}(X, \leq)$. The idea is that this subset will correspond to a part of the order relation that is already two-dimensional. From here on, we use the dual formulation of this problem, i.e., we look for an inclusion minimal set of vertices, which, when deleted, make the transitive incompatibility graph bipartite.

Lemma 4. *Let $(X, \leq)$ be an ordered set and $(a, b), (c, d) \in \mathrm{inc}(X, \leq)$. Then the following statements are equivalent:*

i) $d \leq a$ *and* $b \leq c$.

ii) $(a, b) \to (d, c)$.

iii) $(c, d) \to (b, a)$.

iv) (a, b) *and* (c, d) *are incompatible.*

v) (b, a) *and* (d, c) *are incompatible.*

Proof. $(i) \Rightarrow (iv)$. Consider the relation $\prec := \leq \cup (a, b) \cup (c, d)$. This yields $d \prec a \prec b \prec c \prec d$, i.e, $\prec$ contains a cycle. Analogously $(i) \Rightarrow (v)$.
$(iv) \Rightarrow (ii)$. The assumption directly implies that $(d, c) \in (\leq \cup (a, b))^+$ due to the cycle generated by (a, b) and (c, d). By the same argument $(v) \Rightarrow (iii)$.
$(ii) \Rightarrow (i)$. Assume $d \not\leq a$ or $b \not\leq c$. Since $(d, c) \in (\leq \cup (a, b))^+$ it follows $(d, c) \in \leq$ which contradicts $(c, d) \in \mathrm{inc}(X, \leq)$. Similarly, follows $(iii) \Rightarrow (i)$. □

Recall the definition of $\mathrm{tig}(X, \leq)$ defined on $\mathrm{inc}(X, \leq)$ with incompatible pairs being connected. We call a cycle in $\mathrm{tig}(X, \leq)$ *strict* if and only if for each two adjacent pairs (a, b) and (c, d) it holds that $d < a$ and $b < c$. A strict path is defined analogously.

Lemma 5 (Doignon et al. [2])**.** *Let $(X, \leq)$ be an ordered set and let the pair $v \in \mathrm{inc}(X, \leq)$. Then the following statements are equivalent:*

i) v is contained in an odd cycle in $\mathrm{tig}(X, \leq)$.

ii) v is contained in an odd strict cycle in $\mathrm{tig}(X, \leq)$.

Remark. This is stated implicitly in their proof of Proposition 2.

Theorem 6. *Let $(X, \leq)$ be an ordered set. Let $\mathcal{C} \subseteq \mathrm{inc}(X, \leq)$ be minimal with respect to set inclusion, such that $\mathrm{tig}(X, \leq) \backslash \mathcal{C}^{-1}$ is bipartite. Then $(X, \leq \cup \mathcal{C})$ is an ordered set.*

Proof. Refer to the bipartition elements of $\mathrm{tig}(X, \leq) \backslash \mathcal{C}^{-1}$ with P_1 and P_2.

Claim. The arrow relation is transitive, i.e., if $(a, b) \to (c, d)$ and $(c, d) \to (e, f)$ then $(a, b) \to (e, f)$. If $(a, b) \to (c, d)$ then $c \leq a$ and $b \leq d$ by definition. Similarly, $(c, d) \to (e, f)$ implies $e \leq c$ and $d \leq f$. By transitivity of $\leq$, this yields that $e \leq a$ and $b \leq f$, which in turn implies that $(a, b) \to (d, f)$.

Claim $(\star)$. Let $(a, b), (c, d) \in \mathrm{inc}(X, \leq)$ with $(a, b) \notin \mathcal{C}^{-1}$ and $(a, b) \to (c, d)$, then $(c, d) \notin \mathcal{C}^{-1}$. Assume the opposite, i.e., $(c, d) \in \mathcal{C}^{-1}$. Without loss of generality let $(a, b) \in P_1$. As $(c, d) \in \mathcal{C}^{-1}$ there has to be a pair $(e, f) \in P_1$ that is incompatible to (c, d), i.e., $(e, f) \to (d, c)$, otherwise (c, d) can be added to P_1 without destroying the independent set. However, $(a, b) \to (c, d)$ is equivalent to $(d, c) \to (b, a)$ and yields together with the transitivity of the arrow relation $(e, f) \to (b, a)$. But than (e, f) and (a, b) are incompatible, a contradiction since both are in the independent set P_1.

Claim $(\star\star)$. If $(x, y) \in \mathcal{C}$, then $(y, x) \notin \mathcal{C}$. As $(y, x) \in \mathcal{C}^{-1}$, there is a pair (a, b) in P_1, such that (a, b) and (y, x) are incompatible, i.e., $(a, b) \to (x, y)$, otherwise (y, x) can be added to P_1. However, since $(a, b) \notin \mathcal{C}^{-1}$ follows $(x, y) \notin \mathcal{C}^{-1}$ directly from Claim $(\star)$.

Reflexivity: $\forall x \in X$ we have $(x,x) \in \leq \subseteq \leq \cup \mathcal{C}$.

Antisymmetry: assume $(x,y) \in \leq \cup \mathcal{C}$ and $(y,x) \in \leq \cup \mathcal{C}$. We have to consider three cases. First, $(x,y) \in \leq$ and $(y,x) \in \leq$. Then $x = y$, as $\leq$ is an order relation. Secondly, $(x,y) \in \leq$ and $(y,x) \in \mathcal{C}$. If $(x,y) \in \leq$, then x and y are comparable, i.e., the pair (y,x) can't be in $\mathrm{inc}(P,\leq)$. Then $(y,x) \notin \mathcal{C}$, a contradiction. Thirdly, $(x,y) \in \mathcal{C}$ and $(y,x) \in \mathcal{C}$. This may not occur because of Claim $(\star\star)$.

Transitivity: let $(x,y) \in \leq \cup \mathcal{C}$ and $(y,z) \in \leq \cup \mathcal{C}$ we show $(y,z) \in \leq \cup \mathcal{C}$. We have to consider four cases. First, $(x,y) \in \leq$ and $(y,z) \in \leq$ implies $(x,z) \in \leq$. Secondly, $(x,y) \in \leq$ and $(y,z) \in \mathcal{C}$ and assume that $(x,z) \notin (\leq \cup \mathcal{C})$. Then $(z,x) \notin \mathcal{C}^{-1}$, but $(z,x) \to (z,y)$, as $(x,y) \in \leq$. From Claim $(\star)$ follows that $(z,y) \notin \mathcal{C}^{-1}$, a contradiction to $(y,z) \in \mathcal{C}$. The case $(x,y) \in \mathcal{C}$ and $(y,z) \in \leq$ is treated analogously. Lastly, $(x,y) \in \mathcal{C}$ and $(y,z) \in \mathcal{C}$ and assume $(x,z) \notin \mathcal{C}$, i.e., $(z,x) \in \mathcal{C}^{-1}$. There has to be an odd cycle in $P_1 \cup P_2$ together with (y,z), otherwise (y,z) can be added to $P_1 \cup P_2$ to create a larger bipartite graph. By Lemma 5, there also has to be a strict odd cycle. Let the neighbors of (y,z) in $P_1 \cup P_2$ be (a,b) and (c,d). Then $a < z$, $c < z$, $y < b$ and $y < d$, and the pairs (a,b) and (c,d) are connected by a strict path on an even number of vertices through the strict odd cycle. By the same argument there are pairs (e,f) and (g,h) with $e < y$, $g < y$, $x < f$ and $x < h$ and (e,f) and (g,h) connected by a strict odd path. We now show that (z,x) is in an odd cycle with $P_1 \cup P_2$ to yield a contradiction. For this consider the following paths $A = (c,f)(z,x)(h,a)$, $B = (d,h)(h,g)$ and $C = (f,e)(e,b)$. Each of those is a path in $\mathrm{tig}(P,\leq)$ by definition.

Claim. Between (h,a) and (d,h) there is a path on an even number of vertices in $P_1 \cup P_2$. To show this, let $(a_1,b_1),\ldots,(a_{2k},b_{2k})$ be the strict path on an even number of vertices connecting (a,b) and (d,c) such that $(a_1,b_1) = (a,b)$ and $(a_{2k},b_{2k}) = (d,c)$. This implies $a_{2i+1} < b_{2i+2}$, $a_{2i} < b_{2i+1}$, $b_{2i+1} > a_{2i+2}$ and $a_{2i} > b_{2i+1}$ and for each $i \in \{0,\ldots,k-1\}$. However, this yields the path $(h,a) = (h,a_1)(b_2,h)(h,a_3)\cdots(b_{2k},h) = (d,h)$ which is even and connecting (h,a) and (d,h) in $P_1 \cup P_2$, as required.

Analogously we obtain a path between (c,f) and (b,f) on an even number of vertices. Moreover, (h,g) and (f,e) are also connected by a path on an even number of vertices in $P_1 \cup P_2$, since (g,h) and (e,f) are connected by an even path. Reversing all pairs of this path yields the required path. Combining the segments A, B and C with the paths connecting them yields an odd cycle in $P_1 \cup P_2 \cup \{(z,x)\}$, a contradiction. □

5.1 The Importance of Inclusion-Minimality

Consider once again the example from Figure 1. The ordered set corresponding to the order diagram on the left is well-known to be three-dimensional. However, it becomes an order relation of dimension two by the insertion of a single one of the pairs $(2,5)$, $(1,6)$, or $(3,4)$. Thus, if one of the pairs $(5,2)$, $(6,1)$, or $(4,3)$ is removed from the transitive incompatibility graph, this graph becomes bipartite. The sets $\{(5,2)\}$, $\{(6,1)\}$, or $\{(4,3)\}$ are inclusion minimal sets as required by Theorem 6. To see that inclusion minimality is a necessary condition, consider for example the set $\{(5,2),(2,5)\}$. Its removal makes the transitive incompatibility graph bipartite, however it is not inclusion minimal with respect to this property, as it contains the previously discussed set $\{(5,2)\}$ as a subset. The important observation is that the relation $\leq \cup (5,2),(2,5)$ is not an order relation, otherwise it would follow from the antisymmetry of order relations that $5 = 2$. Thus, inclusion-minimality is a necessary condition in Theorem 6.

Figure 4: An example how new incompatibilities can arise. $\leq$ is the continuous line, $\mathcal{C}$ is the dashed line. (a, b) and (c, d) are not incompatible in $\leq$ and incompatible in $\leq \cup \mathcal{C}$.

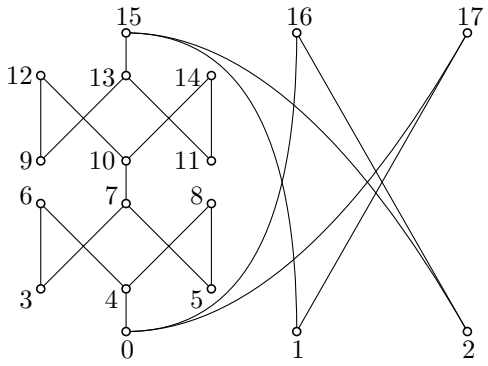

Figure 5: The transitive incompatibility graph of this ordered set has an inclusion-minimal bipartite subgraph that does not give rise to a two-dimension extension.

5.2 Finding Bipartite Subgraphs is not Sufficient

From Theorem 6 one might conjecture that finding an inclusion-minimal bipartite subgraph of $\mathrm{tig}(X, \leq)$ is sufficient to find a two-dimensional extension of $(X, \leq)$. However, it may occur that two pairs are not incompatible in $\leq$ but are incompatible in $\leq \cup \mathcal{C}$ with $\mathcal{C}$ being an inclusion-minimal set such that $\mathrm{tig}(X, \leq) \backslash \mathcal{C}^{-1}$ is bipartite. This can arise in particular, if the following pattern occurs: the ordered set contains the elements a, b, c and d, such that all elements are pairwise incomparable, except $b < d$ and $(c, a) \in \mathcal{C}$; see Figure 4. Then (a, b) and (c, d) are not incompatible in $\mathrm{tig}(X, \leq)$, but they become incompatible with the relation $\leq \cup \mathcal{C}$. An example for this is provided by the ordered set in Figure 5. The transitive incompatibility graph of this ordered set has 206 vertices. An inclusion minimal set that makes the tig graph bipartite if it is deleted is given by

`(5,9), (3,13), (6,15), (17,15), (5,15), (5,13), (11,10), (9,6), (5,6),`
`(4,15), (1,8), (3,15), (11,12), (2,8), (11,7), (0,15), (1,16).`

However, if we insert the complement of this set into the order relation and compute the transitive incompatibility graph of the emerging order, we receive once again a non-bipartite graph. Thus, the newly generated ordered set is also not two-dimensional. We have to add the additional pair `(8,17)` to obtain an ordered set of dimension two. It may be remarked at this point that it is in fact possible to make the transitive incompatibility graph bipartite by adding only eleven pairs (in contrast to the seventeen added in this particular example), see Figure 11. Those pairs give rise to a two-dimension extension.

6 Algorithm

Building up on the ideas and notions from the previous sections we propose the algorithm `DimDraw` as depicted in Algorithm 1. Given an ordered set $(X, \leq)$, one calls `Compute_Coordinates`. Until this procedure identifies a conjugate order using the `Compute_Conjugate_Order` (and in turn the algorithm of Golumbic [14]), it computes the transitive incompatibility graphs using Lemma 4.

Algorithm 1 `DimDraw`

Execute `Compute_Coordinates` on the ordered set that is to be drawn.

```
Input: Ordered set (P,≤)
Output: Conjugate order of (P,≤)

def Compute_Conjugate_Order(P,≤):
    C := Co_comparability_Graph(P,≤)
    if Has_Transitive_Orientation(C):
        (P,≤_C) := Transitive_Orientation(C)
        return ≤_C
    else:
        return ⊥
```

```
Input: Ordered set (P,≤)
Output: Coordinates of the drawing of (P,≤)

def Compute_Coordinates(P,≤):
    ≤_C := Compute_Conjugate_Order(P,≤)
    C := ∅
    while ≤_C = ⊥:
        I := Transitive_Incompatibility_Graph(P,≤∪C)
        B := Maximal_Bipartite_Subgraph(I)
        C = C ∪ V(I\B)^-1
        ≤_C := Compute_Conjugate_Order(P,≤∪C)
    ≤_1 := ≤ ∪ ≤_C
    ≤_2 := ≤ ∪ ≥_C
    for x in P:
        Coord(x,1) := |{k | k ≤_1 x}| - 1
        Coord(x,2) := |{k | k ≤_2 x}| - 1
    return Coord
```

It then identifies which vertices should be removed from the transitive incompatibility graph and adds the reverse pairs to the order $\leq$. Once the `Compute_Coordinates` identifies a conjugate order, it uses this order to compute the realizer. The positions in the realizer are then used to compute the coordinates of all elements which are returned by the algorithm. As, for finite ground sets, the number of incomparable pairs decreases in every repetition step of the algorithm, the number of vertices of the transitive incompatibility graph also decreases and the algorithm thus terminates. However, the computation of the bipartite subgraph, which the algorithm has to solve in each step, entails an NP-complete problem.

6.1 Postprocessing

The algorithm does not prevent a point from being placed on top of lines connecting two different points. This however is not allowed in order diagrams. Possible strategies to deal with this problem are the following. One strategy is to modify the coordinate system, such that the marks of different

integers are not equidistant. Another one is to perturb the points on lines slightly. A third way is to use splines for drawing the line in order to avoid such crossings.

6.2 Runtime Discussion

The problem that we investigate in this work entails the NP-complete problem related to the two-dimension extension. The algorithm that we propose tries to find a maximal induced bipartite subgraphs in each step of the algorithm. As deciding on the vertex cardinality of the maximal bipartite subgraph is also NP-complete, the algorithm proposed has exponential runtime. Nonetheless, we give a short analysis of the different computations in the algorithm. Let for this purpose n be the number of elements of the ordered set. The `Compute_Conjugate_Order` routine makes use of the `Transitive_Orientation`-Algorithm proposed by Golumbic [14] and runs therefore in $O(n^3)$ time. The size of the transitive incompatibility graph can be quadratic in n, if every pair of elements is incompatible. Computing this graph can be done in $O(n^2)$ time using Lemma 4. So, additionally to the exponential task of computing bipartite subgraphs, each repetition has a runtime complexity of at least $O(n^2)$. By our experience, which stems mostly from ordered sets generated using the theory of formal concept analysis, the algorithm terminates after a small number of repetitions, i.e., at most two or three.

7 Finding Large Induced Bipartite Subgraphs

Our algorithm has to compute an inclusion-minimal set of vertices $\mathcal{C}$ such that removing those vertices from the transitive incompatibility graph results in a bipartite graph. For each vertex in $\mathcal{C}$ we will have two incomparable elements in the drawing that are drawn as if they were comparable. It is therefore desirable to remove a truly minimal set of vertices to reduce the amount of such inaccurately drawn pairs. Deciding for a graph whether it is possible to make it bipartite by removing a set of cardinality k is known to be NP-complete [19]. Even approximations are known to be in this complexity class [20]. Therefore, we propose different approaches to tackle this problem.

7.1 Exact Solution using a Reduction to SAT

Even for small examples, i.e., orders on fewer then 30 elements, a naive approach is infeasible. As we will see in Section 8 the question of computing $\mathcal{C}$ results in $\binom{182}{5}$ tests for an example on 19 elements (Figure 7) and $\binom{294}{29}$ tests for example (Figure 8) on 24 elements. Therefore, we need a more sophisticated solution for the problem. We reduce the problem for finding bipartite subgraphs to a SAT problem and then solve this problem with a SAT-Solver, in our case MiniSat [9] in version 2.2. In other words we want to know for some graph $G = (V, E)$ on n vertices and m edges whether by deleting k vertices we can make the graph bipartite. Solving is done by finding a partition of V into the three sets $P_1, P_2, \mathcal{C}^{-1}$, such that P_1 and P_2 are independent sets and $|\mathcal{C}^{-1}| \leq k$. For this we construct a conjunctive normal form as follows: for each vertex v_i we have three variables, call them $V_{i,1}, V_{i,2}, V_{i,3}$. The first two variables indicate, whether the vertex is placed in P_1 or P_2, respectively and the third variable indicates, whether the vertex is placed in $\mathcal{C}^{-1}$. For each vertex we have to guarantee, that it is placed in at least one of P_1, P_2 or $\mathcal{C}^{-1}$, i.e., at least one of $V_{i,1}, V_{i,2}, V_{i,3}$ is true for each i. We achieve this with the clause $V_{i,1} \vee V_{i,2} \vee V_{i,3}$ for all $i \in \{1, \ldots, n\}$. Also, we want to guarantee that P_1 and P_2 are independent sets, i.e., no two vertices in P_1 or P_2 are connected by an edge. We achieve this by adding the two clauses $\neg V_{i,1} \vee \neg V_{j,1}$ and $\neg V_{i,2} \vee \neg V_{j,2}$

Clauses to guarantee that every vertex is assigned to at least one of the classes P_1, P_2 or $\mathcal{C}$:
$((3,4)_1 \vee (3,4)_2 \vee (3,4)_3) \wedge ((3,1)_1 \vee (3,1)_2 \vee (3,1)_3) \wedge ((5,4)_1 \vee (5,4)_2 \vee (5,4)_3) \wedge ((6,4)_1 \vee (6,4)_2 \vee (6,4)_3) \wedge ((1,3)_1 \vee (1,3)_2 \vee (1,3)_3) \wedge ((4,3)_1 \vee (4,3)_2 \vee (4,3)_3) \wedge ((4,6)_1 \vee (4,6)_2 \vee (4,6)_3) \wedge ((1,6)_1 \vee (1,6)_2 \vee (1,6)_3) \wedge ((4,5)_1 \vee (4,5)_2 \vee (4,5)_3) \wedge ((6,1)_1 \vee (6,1)_2 \vee (6,1)_3) \wedge ((5,2)_1 \vee (5,2)_2 \vee (5,2)_3) \wedge ((1,2)_1 \vee (1,2)_2 \vee (1,2)_3) \wedge ((2,1)_1 \vee (2,1)_2 \vee (2,1)_3) \wedge ((5,6)_1 \vee (5,6)_2 \vee (5,6)_3) \wedge ((2,3)_1 \vee (2,3)_2 \vee (2,3)_3) \wedge ((2,5)_1 \vee (2,5)_2 \vee (2,5)_3) \wedge ((6,5)_1 \vee (6,5)_2 \vee (6,5)_3)$
Clauses to ensure that no two adjacent vertices are placed in the same class P_1 or P_2:
$(\neg(3,4)_1 \vee \neg(4,3)_1) \wedge (\neg(3,4)_2 \vee \neg(4,3)_2) \wedge (\neg(4,3)_1 \vee \neg(3,1)_1) \wedge (\neg(4,3)_2 \vee \neg(3,1)_2) \wedge (\neg(4,3)_1 \vee \neg(6,4)_1) \wedge (\neg(4,3)_2 \vee \neg(6,4)_2) \wedge (\neg(3,1)_1 \vee \neg(1,3)_1) \wedge (\neg(3,1)_2 \vee \neg(1,3)_2) \wedge (\neg(6,4)_1 \vee \neg(4,6)_1) \wedge (\neg(6,4)_2 \vee \neg(4,6)_2) \wedge (\neg(1,3)_1 \vee \neg(6,1)_1) \wedge (\neg(1,3)_2 \vee \neg(6,1)_2) \wedge (\neg(4,6)_1 \vee \neg(6,1)_1) \wedge (\neg(4,6)_2 \vee \neg(6,1)_2) \wedge (\neg(4,3)_1 \vee \neg(6,1)_1) \wedge (\neg(4,3)_2 \vee \neg(6,1)_2) \wedge (\neg(4,3)_1 \vee \neg(5,2)_1) \wedge (\neg(4,3)_2 \vee \neg(5,2)_2) \wedge (\neg(4,3)_1 \vee \neg(3,2)_1) \wedge (\neg(4,3)_2 \vee \neg(3,2)_2) \wedge (\neg(4,3)_1 \vee \neg(5,4)_1) \wedge (\neg(4,3)_2 \vee \neg(5,4)_2) \wedge (\neg(3,2)_1 \vee \neg(2,3)_1) \wedge (\neg(3,2)_2 \vee \neg(2,3)_2) \wedge (\neg(5,4)_1 \vee \neg(4,5)_1) \wedge (\neg(5,4)_2 \vee \neg(4,5)_2) \wedge (\neg(4,5)_1 \vee \neg(5,2)_1) \wedge (\neg(4,5)_2 \vee \neg(5,2)_2) \wedge (\neg(2,3)_1 \vee \neg(5,2)_1) \wedge (\neg(2,3)_2 \vee \neg(5,2)_2) \wedge (\neg(6,1)_1 \vee \neg(1,6)_1) \wedge (\neg(6,1)_2 \vee \neg(1,6)_2) \wedge (\neg(5,2)_1 \vee \neg(2,5)_1) \wedge (\neg(5,2)_2 \vee \neg(2,5)_2) \wedge (\neg(6,1)_1 \vee \neg(1,2)_1) \wedge (\neg(6,1)_2 \vee \neg(1,2)_2) \wedge (\neg(1,2)_1 \vee \neg(2,1)_1) \wedge (\neg(1,2)_2 \vee \neg(2,1)_2) \wedge (\neg(2,1)_1 \vee \neg(5,2)_1) \wedge (\neg(2,1)_2 \vee \neg(5,2)_2) \wedge (\neg(6,1)_1 \vee \neg(5,6)_1) \wedge (\neg(6,1)_2 \vee \neg(5,6)_2) \wedge (\neg(5,6)_1 \vee \neg(6,5)_1) \wedge (\neg(5,6)_2 \vee \neg(6,5)_2) \wedge (\neg(6,5)_1 \vee \neg(5,2)_1) \wedge (\neg(6,5)_2 \vee \neg(5,2)_2) \wedge (\neg(6,1)_1 \vee \neg(5,2)_1) \wedge (\neg(6,1)_2 \vee \neg(5,2)_2)$

Figure 6: The SAT clauses for our running example from Figure 1. Additionally, clauses are needed to ensure that at most k of the variables $\{(a,b)_3 \mid (a,b) \in inc(X,\leq)\}$ are assigned true.

for each edge $\{v_i, v_j\} \in E$. This ensures that no two vertices connected by an edge are placed in set P_1 or P_2. Finally, we have to guarantee that there are at most k vertices in P_3, i.e. that at most k of the variables $\{V_{1,3}, V_{2,3}, \ldots, V_{n,3}\}$ are true. There are multiple ways to achieve this. We employ the method from [24]. This results in $(n-1) \cdot k$ auxiliary variables and $2nk + n - 3k - 1$ clauses. Altogether our SAT instance has $(n-1)(k+3)+3$ variables and $2m + 2nk + 2n - 3k - 1$ clauses. For this CNF the following holds by construction.

Theorem 7. *The SAT instance as constructed above is satisfiable if and only if G has an induced bipartite subgraph on $n-k$ vertices.*

For our running example from Figure 1, the SAT clauses introduced in this work are depicted in Figure 6. Now we build this SAT instance for k increasing from 1 until it is satisfiable. Then the set $\{v_i \mid V_{i,3} = \text{true}\}$ is exactly the subset of vertices we have to remove to make the graph bipartite. Obviously methods like binary search may be applied here. For the instances we experimented with, it was always possible to compute the set $\mathcal{C}$ using the SAT-solver approach discussed above. We suspect that for the ordered sets of sizes, for which it is of interest to compute visualizations, this is always possible. Still, we could also plug heuristic procedures into our algorithm to find the inclusion-minimal set $\mathcal{C}$. To do so, we experimented with a greedy algorithm, a simulated annealing approach, and a genetic algorithm with promising results, especially for the genetic algorithm. However, it is preferred to use the SAT algorithm as long as there is enough computational power.

8 Evaluation

The `DimDraw` algorithm was originally designed with the idea in mind of drawing the order diagram of lattices. Those are employed in particular in formal concept analysis (FCA), a mathematical

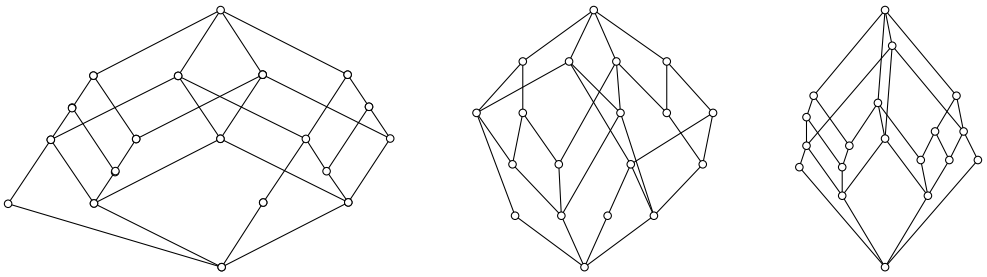

Figure 7: Three drawings of the "Living Beings and Water" lattice, by hand (left), with Sugiyama's framework (middle), and with our algorithm (right).

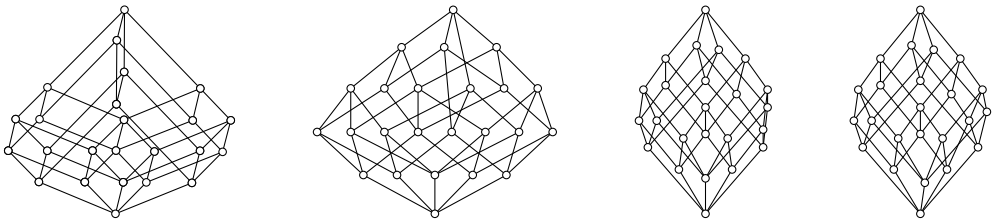

Figure 8: The "Drive concepts for motorcars" lattice. Drawn by an expert (far left), with Sugiyama's framework (middle left) and two drawings of our algorithm (middle and far right).

theory for analyzing data. Note that any complete lattice can be represented by a concept lattice in FCA. We tested our algorithm on all lattice examples from a standard literature book on FCA [13]. In all those cases the quality of the produced drawings seemed to reflect the underlying order structure, similarly to the hand drawn versions of experts. For example, consider the lattice that arises from "Living Beings and Water" [13, p.18]. In Figure 7 we compare the hand-drawn example (left) to the result drawn by our algorithm (right). For Figure 8 [13, p.40] there are two different solutions depicted, both having the minimal number of pairs inserted, note that the algorithm stops after it finds a single solution. Because of the importance of drawings in FCA we tested the algorithm on every lattice with eleven or fewer vertices. The reader may wish to have a look at the document containing all 44994 drawings on 7499 pages [4].

As it is however hardly possible to come up with a metric that quantifies a generated order diagram to be close to a hand-drawn drawing, we also conducted a user study. In this study, in each step the user was presented with three drawings of the same ordered set, one being generated by `DimDraw`, one by Sugiyama's framework, and one using Freese's algorithm. The user then had to decide which drawing was perceived as most readable. To ensure that no algorithm was at an advantage based on the positioning of the drawing, the order of the drawings was randomized in each step. Furthermore, the user was not told which drawing corresponds to which algorithm. Finally, the order in which the ordered set were presented to each user was also randomized. The dataset we compiled for this user study consists of 100 formal contexts from real world data as well as randomly generated contexts. For reproducibility we published the dataset at [5], together with a description of how every formal context in this set was collected. The ordered sets in this dataset have between 7 and 49 elements with an average of 18.34. The cardinality of the covering relation was between 7 and 100 and the mean was 31.96. On average, the ordered sets had 1.67

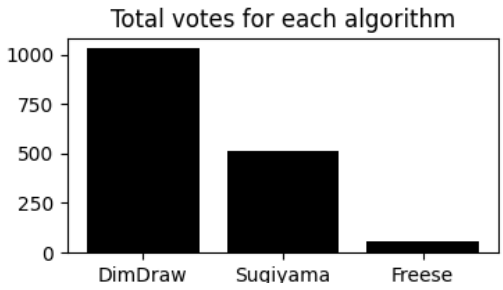

Figure 9: Total votes in the user study: A drawing generated by DimDraw was voted 1030 times as the best drawing, a drawing by Sugiyama's framework was voted 510 times and Freese's drawings were voted 54 times.

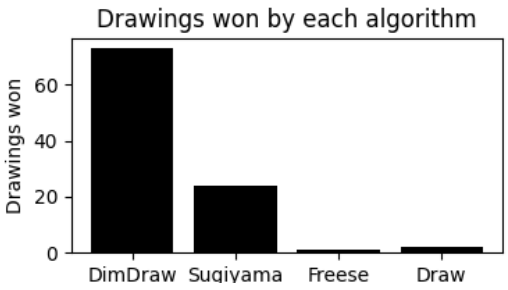

Figure 10: Most selected drawings by algorithm: Of all 100 ordered sets, DimDraw's was 73 times voted as the best drawing, Sugiyama's 24 times and Freese's 1 time. The remaining two times Sugiyama and DimDraw tied.

pairs in the order relation per element.

We restricted the user study to experts of the FCA community, as those are domain experts and experienced in working with order diagrams. A total of 42 such domain experts participated in the study, each rating 37.95 drawings on average, yielding a total of 1594 ratings. The results of this study broken down to the individual drawings is published at [6]. Of all the 1594 ratings, 1030 were in favor of DimDraw, 510 in favor of Sugiyama's framework and 54 in favor of Freese. The above results are visualized in Figure 9. We also compare for each ordered set of the study which drawing was perceived as best by the majority of users, i.e., which drawing got the most votes for each ordered set in Figure 10. Here the drawing generated by DimDraw was chosen 73 times, while Sugiyama's framework and the Freese layout was chosen 24 times and 1 time respectively. The remaining two drawings were tied between DimDraw and Sugiyama. Thus, it can be concluded that the quality of the drawings generated by `DimDraw` outperformed both Sugiyama and Freese in this user study.

Even though, we do not explicitly optimize on the number of crossings of the drawings we decided to test the number of crossings produced in the drawings. Surprisingly, `DimDraw` was able to outperform Sugiyama's framework on several instances, an algorithm that explicitly optimizes on the number of crossings in an upward graph, compare to Table 1. For the dataset we compiled for the user study, DimDraw is the only one that was able to draw each of the 27 included planar ordered sets without crossings. Sugiyama's framework drew 13 drawings without crossings and Freese produced 3 crossing-free drawings.

Concluding the experiments we want to present an example that our algorithm also works on non-lattices. Consider the ordered set from Figure 5. While the hand-drawn version of this order diagram makes use of splines, our algorithm-generated version Figure 11 uses exclusively straight lines.

An interesting observation during our experiments was the following. For all examples that we are aware of, one pass of the SAT solver approach is sufficient for reducing their order dimension to two. This is surprising, in particular in light of Figure 5 from Section 5.2.

Table 1: Comparing the number of crossings in the drawings of some ordered set as produced by Sugiyama's framework and DimDraw.

	Sugiy.	DimDraw
Figure 7	6	**5**
Figure 8	**39**	43
Figure 11	**6**	13
Figure 12	70	**64**
Figure 13	102	**98**
Figure 14	**26**	31

	Sugiy.	DimDraw
Figure 15	**4**	**4**
Figure 16	**3**	4
Figure 17.1	**0**	**0**
Figure 17.2	**2**	**2**
Figure 17.3	22	**20**
Figure 17.4	152	**130**

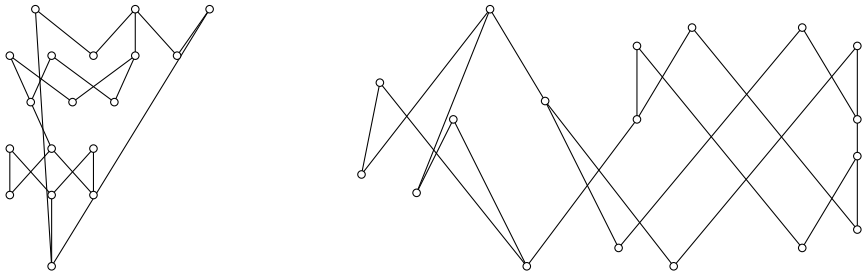

Figure 11: A non-lattice example of an ordered set. One drawing is produced by Sugiyama's framework (left) and one by our algorithm (right). See Figure 5 for a hand-drawn version.

9 Conclusion and Outlook

We presented in this work a novel approach for drawing diagrams of order relations. To this end we employed an idea by Doignon et al. [2] relating order dimension and bipartiteness of graphs and proved an extension. Furthermore, we linked the naturally emerging problem to SAT. Finally, we demonstrated various drawings in a user study. A user study with domain experts confirmed this observation. We would have liked to compare our algorithms to the heuristics developed in [18]. Unfortunately, we were not able to reproduce their results based on the provided description.

A notable observation is the fact that in all our experiments the SAT-Solver blend of `DimDraw` was able to produce a solution in the first pass, i.e., the algorithm found a truly minimal two-dimension extension. This raises the natural question whether the maximal induced bipartite subgraph approach does always result in a minimal two-dimension extension. Further open questions are concerned with employing heuristics and improving the postprocessing stage. The SAT-solver version of `DimDraw` is included in the software conexp-clj[1]. At a later time we also want to include heuristic versions.

[1] https://github.com/tomhanika/conexp-clj

Acknowledgements

The authors would like to thank Torsten Ueckerdt for pointing out the research about diametral pairs and Maximilian Stubbemann for helpful discussions.

References

[1] G. Brightwell and M. Massow. Diametral pairs of linear extensions. *SIAM Journal on Discrete Mathematics*, 27(2):634–649, Jan. 2013. doi:10.1137/080733140.

[2] J.-P. Doignon, A. Ducamp, and J.-C. Falmagne. On realizable biorders and the biorder dimension of a relation. *Journal of Mathematical Psychology*, 28(1):73–109, 1984.

[3] B. Dushnik and E. W. Miller. Partially ordered sets. *American journal of mathematics*, 63(3):600–610, 1941. doi:10.2307/2371374.

[4] D. Dürrschnabel, T. Hanika, and G. Stumme. Experimental evaluation of DimDraw, June 2019. doi:10.5281/zenodo.3242627.

[5] D. Dürrschnabel, T. Hanika, and G. Stumme. Dataset of user study for dimdraw, Oct. 2020. doi:10.5281/zenodo.4075207.

[6] D. Dürrschnabel, T. Hanika, and G. Stumme. User study of dimdraw, Oct. 2020. doi:10.5281/zenodo.4075215.

[7] P. Eades, B. D. McKay, and N. C. Wormald. On an edge crossing problem. In *Proc. 9th Australian Computer Science Conference*, volume 327, page 334, 1986.

[8] P. Eades and N. Wormald. *The Median Heuristic for Drawing 2-layered Networks*. Technical report. University of Queensland, Department of Computer Science, 1986.

[9] N. Eén and N. Sörensson. An extensible SAT-solver. In E. Giunchiglia and A. Tacchella, editors, *Theory and Applications of Satisfiability Testing*, pages 502–518, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. doi:10.1007/978-3-540-24605-3_37.

[10] D. Eppstein and J. A. Simons. Confluent Hasse diagrams. In M. J. van Kreveld and B. Speckmann, editors, *Graph Drawing*, volume 7034 of *Lecture Notes in Computer Science*, pages 2–13. Springer, 2011. doi:10.1007/978-3-642-25878-7_2.

[11] R. Freese. Automated lattice drawing. In P. Eklund, editor, *Concept Lattices*, volume 2961 of *LNCS*, pages 112–127. Springer, Berlin/Heidelberg, 2004.

[12] B. Ganter. Conflict avoidance in additive order diagrams. *Journal of Universal Computer Science,*, 10(8):955–966, 2004. doi:10.3217/jucs-010-08-0955.

[13] B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag, Berlin, 1999.

[14] M. C. Golumbic. The complexity of comparability graph recognition and coloring. *Computing*, 18(3):199–208, 1977. doi:10.1007/BF02253207.

[15] D. Kelly and I. Rival. Planar lattices. *Canadian Journal of Mathematics*, 27(3):636–665–, 1975.

[16] E. M. Kornaropoulos and I. G. Tollis. Overloaded orthogonal drawings. In M. J. van Kreveld and B. Speckmann, editors, *Graph Drawing - 19th International Symposium, GD 2011, Eindhoven, The Netherlands, September 21-23, 2011, Revised Selected Papers*, volume 7034 of *Lecture Notes in Computer Science*, pages 242–253. Springer, 2011. doi:10.1007/978-3-642-25878-7_24.

[17] E. M. Kornaropoulos and I. G. Tollis. Weak dominance drawings and linear extension diameter. *CoRR*, abs/1108.1439, 2011.

[18] E. M. Kornaropoulos and I. G. Tollis. Weak dominance drawings for directed acyclic graphs. In W. Didimo and M. Patrignani, editors, *Graph Drawing*, volume 7704 of *Lecture Notes in Computer Science*, pages 559–560. Springer, 2012. doi:10.1007/978-3-642-36763-2_52.

[19] J. M. Lewis and M. Yannakakis. The node-deletion problem for hereditary properties is NP-complete. *Journal of Computer and System Sciences*, 20(2):219 – 230, 1980. doi:10.1016/0022-0000(80)90060-4.

[20] C. Lund and M. Yannakakis. The approximation of maximum subgraph problems. In A. Lingas, R. Karlsson, and S. Carlsson, editors, *Automata, Languages and Programming*, pages 40–51, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

[21] R. M. Mcconnell. Linear-time transitive orientation. In *In Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1997.

[22] E. Messinger, L. A. Rowe, and R. R. Henry. A divide-and-conquer algorithm for the automatic layout of large directed graphs. *IEEE Trans. Systems, Man, and Cybernetics*, 21(1):1/2, 1991. doi:10.1109/21.101131.

[23] E. M. Reingold and J. S. Tilford. Tidier drawings of trees. *IEEE Trans. Software Eng.*, 7(2):223–228, 1981. doi:10.1109/TSE.1981.234519.

[24] C. Sinz. Towards an optimal CNF encoding of boolean cardinality constraints. In P. van Beek, editor, *CP*, volume 3709 of *Lecture Notes in Computer Science*, pages 827–831. Springer, 2005. doi:10.1007/11564751_73.

[25] G. Stumme and R. Wille. A geometrical heuristic for drawing concept lattices. In R. Tamassia and I. Tollis, editors, *Graph Drawing*, volume 894 of *LNCS*, pages 452–459, Heidelberg, 1995. Springer.

[26] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man & Cybernetics*, 11(2):109–125, 1981. doi:10.1109/TSMC.1981.4308636.

[27] W. T. Trotter. *Combinatorics and partially ordered sets: Dimension theory*, volume 6. JHU Press, 1992. doi:10.1137/103511.

[28] R. R. Veloso, L. Cerf, W. M. Jr., and M. J. Zaki. Reachability queries in very large graphs: A fast refined online search approach. In *Proceedings of the 17th International Conference on Extending Database Technology, EDBT 2014, Athens, Greece, March 24-28, 2014*, pages 511–522. OpenProceedings.org, 2014. doi:10.5441/002/EDBT.2014.46.

[29] M. Yannakakis. The complexity of the partial order dimension problem. *SIAM Journal on Algebraic Discrete Methods*, 3(3):351–358, 1982. doi:10.1137/0603036.

A Availability of data and material

- dataset used for the user study: https://zenodo.org/record/4075207
- results of the user study: https://zenodo.org/record/4075215

B Code availability

The source code is available as a part of the research software conexp-clj https://github.com/tomhanika/conexp-clj.

C Additional Drawings

In this section we provide some additional drawings generated by `DimDraw`.

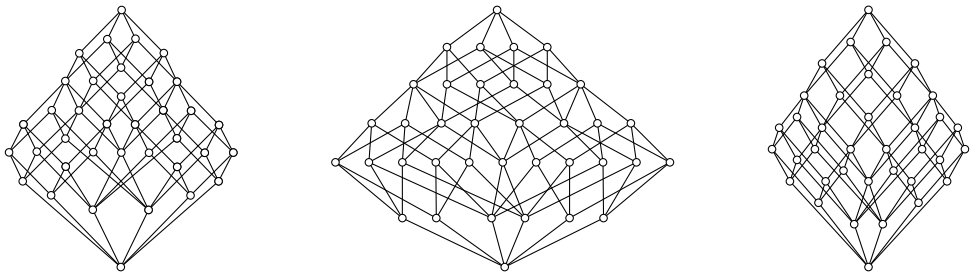

Figure 12: An example from [13, p.53]. The hand-drawn version is on the left, Sugiyama's framework in the middle, our version on the right.

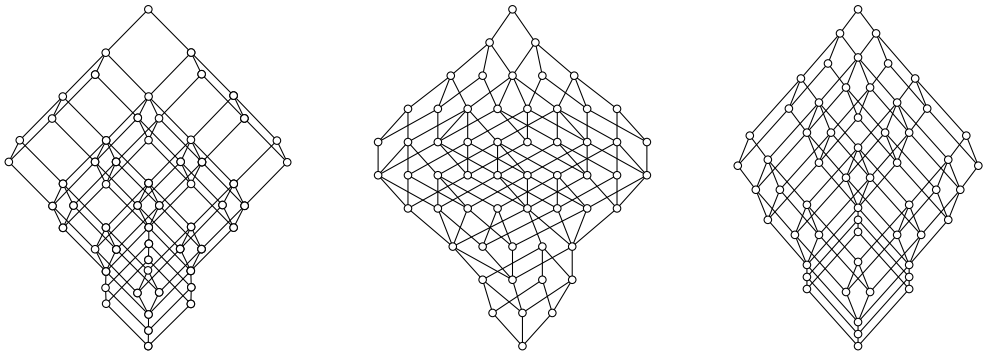

Figure 13: An example from [13, p.35]. The hand-drawn version is on the left, Sugiyama's framework in the middle, our version on the right.

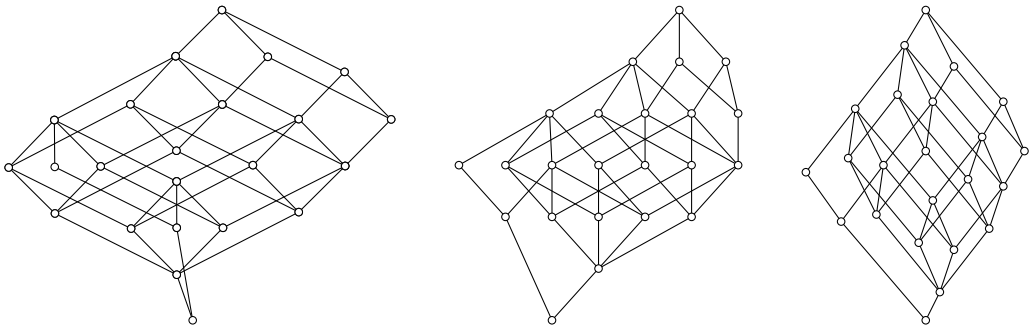

Figure 14: An example from [13, p.30]. The hand-drawn version is on the left, Sugiyama's framework in the middle, our version on the right.

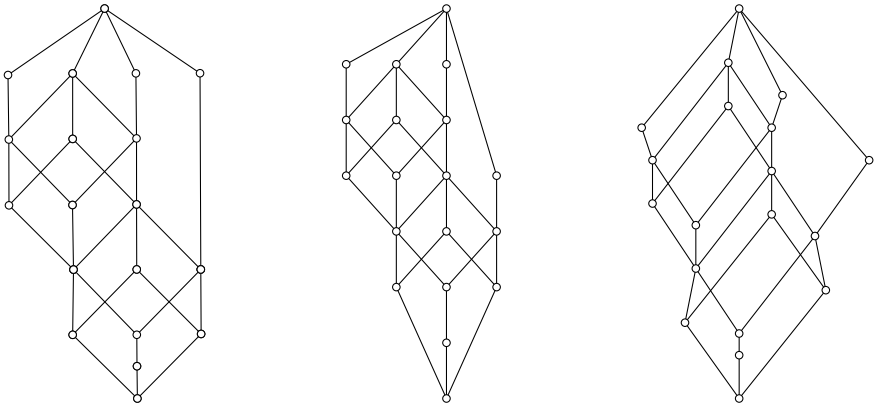

Figure 15: An example from [13, p.45]. The hand-drawn version is on the left, Sugiyama's framework in the middle, our version on the right.

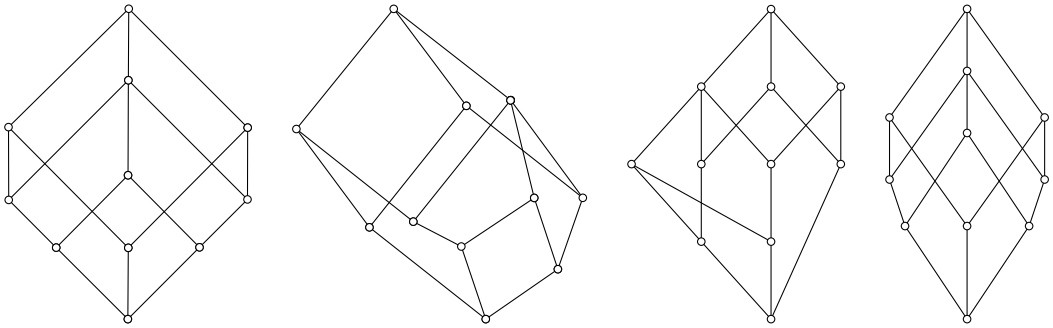

Figure 16: Order diagram of the lattice used in [12]. From left to right: Hand drawn, Ganter's algorithm, Sugiyama's framework, our algorithm.

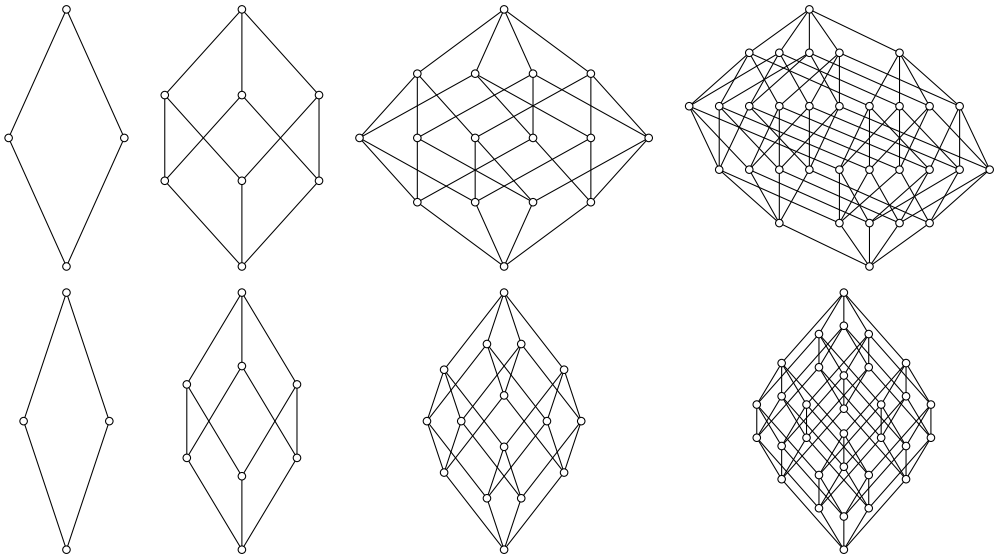

Figure 17: Order diagrams for Boolean lattices of dimension two, three, four, and five drawn by Sugiyama's framework (top) and our algorithm (bottom).