Automated Layout of Concept Lattices Using Force Directed Placement and Genetic Algorithms

Richard Cole
School of Information Technology
Griffith University
r.cole@gu.edu.au

Abstract

Concept lattices represent a conceptual hierarchy inherent in a data set. A labelled line diagram for such a lattice represents this information diagrammatically. A diagram for a concept lattice may be algebraicly generated by a set of vectors assigned to elements of the lattice. Such a diagram is called an additive line diagram, and is considered useful because it makes apparent the inherent structure of the lattice.

This paper reports on experience with two approaches to automated concept lattice layout: (i) using genetic algorithms optimising over a finite discrete space of diagrams, and (ii) force directed placement optimising over a continuous rational space.

The layout of concept lattices is of relevance to the layout of lattices in general since any lattice can be represented simply by a concept lattice.

1 Formal Concept Analysis

Formal Concept Analysis (FCA) is a method for analysing data. The data to be analysed is first mathematised by a *formal context*. A *formal context* is defined as a triple (G, M, I) where G is a set of objects, M is a set of attributes, and $I \subseteq G \times M$ is a relation between G and M describing when an object g has an attribute m. If $(g, m) \in I$ then we write gIm and say object g has attribute m.

FCA defines the notion of a *formal concept* of the *formal* context (G, M, I) as a pair (A, B) where $A \subseteq G$ is called the *extent*, $B \subseteq M$ is called the *intent*, A is exactly the set of all the objects in G that possess all the attributes in the intent B, and B is exactly the set of all attribute common to all the objects in A. Two operators are defined that help to determine formal concepts. If for a *formal* context (G, M, I), $A \subseteq G$ and $B \subseteq M$ then we define

$$A' = \{m \in M \mid \forall g \in A : gIm\}$$

$$B' = \{g \in G \mid \forall m \in B : gIm\}$$

Thus, if for a pair (A, B) with $A \subseteq G$, and $B \subseteq M$, it is the case that $A = B'$ and $B = A'$, then (A, B) is a formal concept.

Formal concepts have a specialisation ordering defined on them. A concept (A_1, B_1), is more specific than another concept, (A_2, B_2), written $(A_1, B_1) \leq$

(A_2, B_2), if $A_1 \subseteq A_2$ (or equivalently $B_2 \subseteq B_1$). The set of concepts, together with this $\leq$ relation forms a lattice, denoted $\underline{\mathfrak{B}}(G, M, I)$, and called the *concept lattice* of the *formal* context (G, M, I).

A line diagram represents the concepts as small circles, and the covering relation[3] (derived from the ordering relation) defined on the concepts as lines between the circles. If concept c_1 covers concept c_2 then there must be a straight line from the circle representing c_1 to the circle representing c_2 and c_2 will appear below c_1 in the diagram. This arrangement places the circle for the most specific concept at the bottom of the diagram.

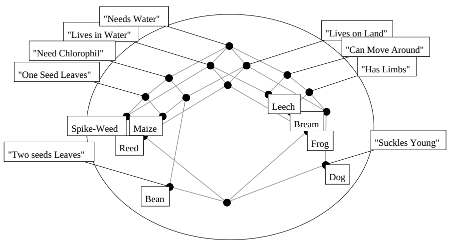

Figure 1. Example Concept Lattice

An example of a concept lattice is shown in Figure 1. This lattice is about the animals and plants mentioned in a wild life documentary for children. The objects in the concept lattice are animals and plants, and the attributes are properties mentioned in the film. The intent for a concept may be found by collecting attributes going upwards in the diagram along lines, while the extent is is found by collecting objects following lines downwards in the diagram. So for example the concept labelled Maize in the diagram collects *Reed* in its extent by going down, and *One Seed Leaves*, *Needs Chlorophyll*, and *Needs Water* in its intent by going up. This concept indicates that both *Maize* and *Reed* possess all the collected attributes, i.e *One Seed Leaves*, *Needs Chlorophyll*, and *Needs Water*.

Because the concept for *Reed* is below, and connected to, the concept for *Maize*, the object *Reed* must possess all the attributes of the object *Maize*, plus some more. The additional attribute in this case is *Lives in Water*. A further discussion of how to read concept lattices may be obtained from Ganter and Wille [5].

2 Layout Approaches

2.1 Hierarchical Layout

Partially ordered sets, for the purposes of layout, are seen as hierarchies. An annotated bibliography by Battista et. al[1] describes the approach as

> A hierarchical drawing of an acyclic diagram is an upwards polyline drawing where the vertices and bends are constrained to lie on a set of equally spaced horizontal lines.

Sugiyama's algorithm[8] is typical of a variety of algorithms proposed for the layout of such diagrams. It may be summarised as a three step process:

1. Partition the vertices into layers.
2. Reduce the number of edge crossings by swapping the horizontal ordering of vertices.
3. Minimize the number of bends, where bends are dummy nodes inserted to ensure that lines travel only between adjacent layers.

This formulation of the problem is designed clearly to place great emphasis on reducing the number of edge crossings. It is not clear however that the number of edge crossings is a significant factor in either the aesthetic quality of concept lattice line diagram, or the amount of structure evident in a concept lattice line diagram.

2.2 Additive Line Diagrams

Wille noted in his 1993 paper [10]:

> "Lattices in data analysis are more than just mathematical structures: They carry meaning. Therefore, drawings of such lattices should not only reflect the mathematical structure but also give a meaningful presentation for the data."

For the purposes of describing line diagrams of concept lattices, Ganter and Wille [5] introduce the notion of an additive line diagrams which we repeat here:

A *set representation* of an ordered set $(P, \leq)$ is an order embedding of $(P, \leq)$ into the power set of a set X, i.e a map

$$\text{rep} : P \to \mathcal{P}(X)$$

with the property

$$x \leq y \Leftrightarrow \text{rep}(x) \subseteq \text{rep}(y).$$

For an additive line diagram of an ordered set $(P, \leq)$ we need a set representation rep : $P \to \mathcal{P}(X)$ as well as a *grid projection*

$$\text{vec} : X \to \mathbb{R}^2$$

assigning a real vector with a positive y-coordinate to each element of X. By

$$\text{pos}(p) := n + \sum_{x \in \text{rep}(p)} \text{vec}(x)$$

we obtain a positioning of the element $p \in P$ in the plane. Here n is a vector which can be chosen arbitrarily in order to shift the entire diagram. By allowing only positive y-coordinates for the grid projection we make sure that no element p is positioned below an element q with $q < p$.

Every finite line diagram can be interpreted as an additive line diagram with respect to an appropriate line diagram set representation. For concept lattices we usually use the representation by means of the irreducible objects and/or attributes.

In this paper we shall experiment with the production of additive line diagrams based on the representation by means of irreducible objects and/or attributes.

In a *formal* context (G, M, I) certain objects and attributes may be removed without altering the structure of the concept lattice produced. The objects and attributes remaining are called *irreducible* attributes and irreducible objects, and a context containing just irreducible objects and attributes is called a *purified* context. For a full discussion see [5]. Since the construction of a purified context is straight forward we shall for the rest of this paper assume all formal contexts are purified contexts.

An additive line diagram based on attributes is produced by a set representation using the objects and attributes, i.e. $rep : \underline{\mathfrak{B}}(G, M, I) \to M$,

$$\text{rep}((A, B)) := B$$

and a grid projection from the attributes to vectors with positive y-components:

These attribute based additive line diagrams are useful in reading concept lattices because the difference in the intent between two concepts may be determined by decomposing the vector between the two concepts into its constituent parts.

Consider two concepts c_1 and c_2 of the concept lattice $\underline{\mathfrak{B}}(G, M, I)$. Let B_1 be the attributes in the intent of c_1 and B_2 be the attributes in the intent of c_2 then:

$$\text{pos}(c_1) - \text{pos}(c_2) =$$

$$\sum_{x \in S} \text{vec}(x) - \sum_{x \in T} \text{vec}(x)$$

where (recalling that rep(c_1) is a set and taking A/B to denote set difference)

$$S = \text{rep}(c_1)/\text{rep}(c_2)$$

and

$$T = \text{rep}(c_2)/\text{rep}(c_1).$$

In other words the vector between two concepts gives a clue as to the symmetric difference between the intents of those two concepts.

In comparing two concepts that are in a covering relation to one another there is only one meet attribute in the change, and therefore the vector is easily recognised.

An additive line diagram based on both the attributes and the objects is produced by a set representation[1], rep : $\underline{\mathfrak{B}}(G, M, I) \rightarrow M \cup G$,

$$\text{rep}((A, B)) := B \cup (G/A)$$

and a grid projection vec : $M \cup G \rightarrow \mathbb{R}^2$:

Figure 2 shows an additive line diagram based on both objects and attributes for the smallest non-distributive lattice. The left most concept has for intent $\{A\}$ and for extent a. Its position is given by vec(A) + vec(b) + vec(c). These vectors are shown in the diagram by a dotted line and all have positive y-components.

For the purposes of formulating optimisation algorithms it is useful to translate the construction for an additive line diagram given by Ganter and Wille into a matrix equation.

[1]We assume for simplicity that M and G are disjoint, although we are not restricted to this case.

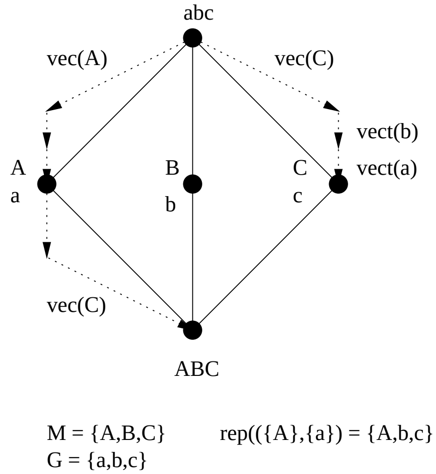

Figure 2. Example of an additive line diagram for the smallest non-distributive lattice.

Let $(P, \leq)$ be a partially ordered set with $P = \{p_1, \ldots, p_m\}$, rep : $P \rightarrow X$ be a set representation for P with $X = \{x_1, \ldots, x_n\}$, and vec : $X \rightarrow \mathbb{R}^2$ a grid projection. We define the matrix $\mathbf{A} = (a_{i,j})_{n,m}$ with $n = |X|$, $m = |P|$ ($|P|$ denotes the cardinality of the set $|P|$), and $a_{ij} = 1$ if $x_j \in$ rep(p_i) and $a_{ij} = 0$ otherwise. We also define the vector $\mathbf{V} = (v_i)_n^T$, where T is the transpose, with $v_i =$ vec(x_i) and the vector $\mathbf{D} = (d_i)_m^T$ with $d_i =$ pos(p_i).

Then we have:

$$\mathbf{AV} = \mathbf{D}$$

We shall assume for our optimisation problem that the set representation is fixed and that $\mathbf{V}$ is the vector to be optimised with respect to some objective function given on $\mathbf{D}$.

Lattice	$\|M\|$	$\|G\|$	$\|L\|$	Source
FD3	8	8	18	[5]
Film	9	8	19	[5]
Fig8	10	10	15	[10]
Knives	8	18	17	[7]
ID4	6	4	11	[5]
Triangles	7	7	18	[10]

Table 1. Test concept lattices.

3 Choice of Test Lattices

In choosing test lattices we had two objectives. The first objective was to choose lattices that are representative of the range of concept lattices encountered in practice. The concept lattice shown were taken from the literature and are summarised in Table 1.

Two important metrics on the concept lattices for predicting performance are: (i) the number of attributes and objects (responsible for determining the length of $\mathbf{V}$), and (ii) the number of concepts (responsible for determining the number of rows in $\mathbf{A}$).

Another important feature of the lattices is whether or not they are distributive (i.e $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$ and $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$. Distributive lattices can be effectively drawn by performing a chain decomposition of the lattice [10]. Of the examples only Knives is non-distributive.

Our reason for experimenting with both distributive and non distributive lattices is that we are searching for a layout technique that will work for most practically encountered concept lattices.

4 Genetic Algorithms

A genetic algorithm (GA) package called GALIB [9] was employed to experiment with standard genetic algorithms. A GA maintains a population of genomes. Each genome represents a position in the search space. The GA then generates new genomes by breading two parent genomes together, or altering an individual via an operation called mutation. The new genomes are then added to the population displacing some old genomes. Both the replacement and selection for breeding of the genomes is based on a calculation of the fitness of the individual genomes within the population of genomes. The fitness is calculated via a comparison of objective function calculated on each genome. We experimented with the classic GA described by Goldberg [6], and the DeJong GA [4] and found that, for our layout experiments, the DeJong GA produced better results.

4.1 Encoding

The formulation at the end of Section 2.2 required the optimisation of $\mathbf{V}$ with respect to an objective function defined on $\mathbf{D}$. We therefore use a vector encoding in which each genome is an instance of $\mathbf{V}$. The separate elements of a genome are called genes.

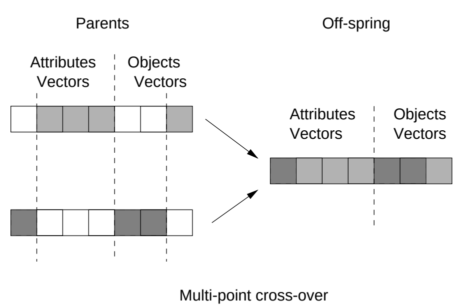

Figure 3. Genome encoding of a diagram.

Figure 3 shows two *parent* genomes being used to create a new *child* genome containing portions from both *parent* genomes. Each element in these genomes encodes a separate value of the grid projection. That is each element is a 2-dimentional vector in the real plane. The elements corresponding to objects are considered seperate to the elements corresponding to attributes. To this end two cross-over sites are selected so that no vectors are copied from attribute genes to object genes and vice versa.

We experimented with vector elements chosen from the sequences $(1, k)$, and (k, k^2) for $k = 1 \ldots n$, since these vectors appear commonly in concept lattices drawn by hand.

4.2 Optimisation Criteria

The objective function for evaluating the quality of a diagram D was derived from two types of constraint: hard and soft constraints. There was a single hard constraint: that the diagram did not contain any two concepts with the same position. In the case that the hard constraint was violated the objective function was:

$$\phi(\mathbf{D}) = \frac{10}{m - |\{d_i \mid i = 1 \ldots m\}|}$$

This hard constraint has an order complexity of $O(m)$. Recall that m is the number of concepts.

In the case that there are no two concepts with the same position a weighted sum of soft objective functions was taken, with one soft objective function for each soft constraint. Some of the soft constraints in the diagram are based on a notion of the set of edges in the diagram:

Let P be a partially ordered set, then the set of edges in the diagram of P is given by:

$$E := \{(i, j) \mid p_i \prec p_j\}$$

where $p_i \prec p_j$ if $p_i < p_j$ and for every $x < p_j$ it is not the case that $x > p_i$.

The soft constraints are listed below. Each constraint gives rise to a soft objective function. We denote these soft objective functions $\phi_1 \ldots \phi_5$.

1. *Minimize the total length of diagram edges.* The sum is taken over the length of all edges in the diagram.

$$\phi_1(\mathbf{D}) = \sum_{(i,j) \in E} |d_i - d_j|$$

2. *Minimize the number of edge vectors.* The number of distinct edge vectors.

$$\phi_2(\mathbf{D}) = |\{d_i - d_j \mid (i, j) \in E\}|$$

3. *Maximize the symmetry in the diagram.*

$$\begin{aligned} \phi_3(\mathbf{D}) = |\{(i, j, k) \mid \\ & (i, j) \in E, (k, j) \in E, \\ & (d_i + d_k) \bullet u_x = 2(d_j \bullet u_x)\}| \end{aligned}$$

where $(d_i + d_k) \bullet u_x$ is taken to mean the x-component of the sum of d_i and d_k. Here we count the number of sibling pairs in the diagram whose x-components sum to zero.

4. *Maximize the number of chain elements.* Count the number of ways three elements can be chosen (grandparent, parent, child) such that the vector from the grandparent to the parent is the same as the vector from the parent to the child.

$$\begin{aligned} \phi_5(\mathbf{D}) = |\{(i, j, k) \mid \\ & (i, j) \in E, (j, k) \in E, \\ & (d_i - d_j) = (d_j - d_k)\}| \end{aligned}$$

5. *Maximize the number of parallel edges.*

$$\phi_4(\mathbf{D}) = |\{d_i - d_j \mid (i,j) \in E\}|$$

The soft objective functions are weighted by taking either the ratio, or the inverse of the ratio, of their value for the current diagram with their value for the first diagram encountered satisfying the hard constraint, denoted $\mathbf{D}_0$. This first diagram is effectively chosen at random.

In the case that the hard constraint is met the objective function was:

$$\phi(\mathbf{D}) = 10 + \frac{\phi_1(\mathbf{D}_0)}{\phi_1(\mathbf{D})} + \frac{\phi_2(\mathbf{D}_0)}{\phi_2(\mathbf{D})} + \frac{\phi_3(\mathbf{D})}{\phi_3(\mathbf{D}_0)} + \frac{\phi_4(\mathbf{D})}{\phi_4(\mathbf{D}_0)} + \frac{\phi_5(\mathbf{D})}{\phi_5(\mathbf{D}_0)}$$

5 Force Directed Placement

Force directed placement (FDP) is a popular technique for the layout of graphs. The idea of FDP is to characterize the problem of finding a good layout via forces applied to the nodes in the graph. One may then understand the graph as a mesh of springs that is given a shake and allowed to settle into a hopefully stable configuration.

Traditionally the forces are expressed as the rate of change of some objective function, $\phi : (\mathbb{R}^2)^m \to \mathbb{R}$, with respect to the diagram $\mathbf{D}$. This objective function is made up of a weighted sum of component objective functions that each encodes a single aesthetic objective.

$$\phi(\mathbf{D}) = \sum_{i=1}^{n} w_i \phi_i(\mathbf{D})$$

In attempting to minimise the objective function ϕ we move in the direction of maximum change. One system for limiting the distance moved was reported with its application to graph layout by Coleman and Parker [2].

$$\mathbf{D}_{i+1} = \mathbf{D}_i + \Delta_i \mathbf{D}_i$$

where

$$\Delta_i \mathbf{D} = \frac{\frac{d\phi}{d\mathbf{D}}}{\left|\frac{d\phi}{d\mathbf{D}}\right|} \min\left(\left|\frac{d\phi}{d\mathbf{D}}\right|, t_i\right)$$

where $|\frac{d\phi}{d\mathbf{D}}|$ indicates the determinant of $\frac{d\phi}{d\mathbf{D}}$, and t_i is a temperature coefficient modified according to the series.

$$t_i = \left(\frac{t_{\text{start}}}{t_{\text{end}}}\right)^{1/i}$$

Our aim is to produce an element based additive line diagram. There are two ways this can be achieved. Firstly, we could project the diagram produced by FDP onto the manifold of element based additive line diagrams. We found however that this projection can cause a large degradation in the aesthetic quality of the diagram.

A second approach is to translate the process of FDP from the space of all diagrams into the space of element based additive line diagrams. Recall the formulation from the end of Section 2.2.

$$\mathbf{AV} = \mathbf{D}$$

where

$$\mathbf{D} = \begin{bmatrix} d_{1,1} & d_{1,2} \\ \vdots & \vdots \\ d_{m,1} & d_{m,2} \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & \dots & a_{1,n} \\ \vdots & & \vdots \\ a_{m,1} & \dots & a_{m,n} \end{bmatrix}$$

and

$$\mathbf{V} = \begin{bmatrix} v_{1,1} & v_{1,2} \\ \vdots & \vdots \\ v_{n,1} & v_{n,2} \end{bmatrix}$$

It is then possible to determine the rate of change of the vectors $\mathbf{V}$ with respect to the fitness of $\mathbf{D}$.

$$\frac{d\phi}{d\mathbf{D}} = \begin{bmatrix} \frac{\delta\phi}{\delta d_{1,1}} & \frac{\delta\phi}{\delta d_{1,2}} \\ \vdots & \vdots \\ \frac{\delta\phi}{\delta d_{m,1}} & \frac{\delta\phi}{\delta d_{m,2}} \end{bmatrix}$$

We note that

$$\frac{\delta\phi}{\delta v_{i,j}} = \sum_{k=1}^{m} \frac{\delta\phi}{\delta d_{k,j}} \frac{\delta d_{k,j}}{\delta d_{i,j}}$$

and

$$\frac{\delta d_{k,j}}{\delta v_{i,j}} = a_{k,i}$$

which with some re-arrangement leads to

$$\frac{\delta\phi}{\delta v_{i,j}} = \sum_{k=1}^{m} a_{k,i} \frac{\delta\phi}{\delta d_{k,j}}$$

Thus we may use the same schedule as in Equation 5 for traversing the space of additive line diagrams for which $\mathbf{V}$ encodes a solution. We used the same objective functions as were reported by Coleman and Parker[2].

6 Results

Figures **??**, **??** and **??** show the results of layout using FDP, FDP with additive diagrams, and DeJong's genetic algorithm respectively. All diagram were produced in a single run using the same parameters.

The genetic algorithm was run with a population size of 50 genomes, 750 generations, a mutation probability of 0.01, and a replacement rate of 50%. On average the genetic algorithm performed 17109 diagram evaluations.

The FDP algorithms we allowed 1000 iterations, but took as similar amount of time to the GA because the objective function is more expensive to compute $O(m^2)$ as compared with $O(m)$.

Figure **??** shows the layout produced by performing FDP in the space of all diagram. The resulting layouts can not be described as attribute and object based additive line diagrams.

The layouts for the three lattices on the left side, FD3, Film, and ID4 are generally quite good exhibiting some and a significant number of parallel edges. It is impossible to directly see the repeated sublattice in the Fig8 lattice at the top right, and the regular structure of Triangles at the bottom right is not evident.

Performing FDR in the space of attribute and object based additive line diagrams, shown in Figure **??**, produced very similar results. The knives lattice still doesn't exhibit much structure, the *triangles* lattice however is much more structured.

Figure **??** shows the results of performing layout using the genetic algorithm. While all the lattices exhibit a very regular structure — a product of the space over which the algorithm searches — several of the lattices, namely Film (bottom middle), ID4 (bottom left), and Triangles (bottom right) suffer from a distended base.

A distended base occurs when the bottom element is a long way away from the rest of the nodes in the diagram. This occurs in an additive diagram based only on attributes. Since the bottom element has all attributes if its upper cover contains concepts with say half the attributes, then the vectors for the remaining attributes, whose sum is the vector to the bottom element from the upper cover, will be lengthy. Figure 2 showed a solution to this problem, the GA however was unable to converge on such a solution.

The repeated sub-lattice in Fig8 (top right in Figure **??**) is clearly evident.

7 Conclusions

This paper has demonstrated how two common layout approaches may be adapted to the layout of concept lattices via the descriptive mechanism of an additive line diagram. Although the FDP algorithm is restricted to expressing objective functions via forces on lattice elements it was in several test lattices able to produce a diagram showing clearly the structure present in the lattice. The GA produced much more structured results and was successful in finding a good layout for the only non-distributive lattice example. Several diagrams generated by the GA however suffered from the problem of a distended base.

Further experimentation could be conducted with the GA algorithm with additional objective functions, and having the vectors for attributes and objects chosen from different sets.

References

[1] G. Battista, P. Eades, R. Tamassia, and I. Tollis. Algorithms for drawing graphs, an annotated bibiliography. *Computational Geometry, Theory and Applications*, 4:235–282, 1994.

[2] M. Coleman and D. Stott Parker. Aesthetics-based graph layout for human consumption. *Software—Practice and Experience*, 26(1), January 1996.

[3] B.A. Davey and H.A. Priestly. *Introduction to Lattices and Order Theory*. Cambridge University Press, 1990.

[4] K. A. De Jong. *An analysis of the behavior of a class of genetic adaptive systems*. PhD thesis, University of Michigan, 1975.

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

[6] David Goldberg. *Genetic Algorithms in Search Optimisation and Machine Learning*. Addison-Wesley, 1989.

[7] Rolf Langsdorf, Martin Skorsky, Rudolf Wille, and Albrecht Wolf. An approach to automated drawing of concept lattices. Technical Report 1874, Technical University of Darmstadt, Schlossgatenstasse, Darmstadt, Germany, 1996.

[8] K. Sugiyama, S. Tagawa, and M. Toda. Structureing information for understanding complex systems: A basis for decision making". *IEEE Transactions on Systems, Man and Cybernetics*, SMC-11(2):109–125, 1981.

[9] Matthew Wall. GAlib: A C++ library of genetic algorithm components. "http://lancet.mit.edu/ga/", August 1996.

[10] R. Wille. Lattices in data analysis: How to draw them with a computer. *Algorithms and Order*, pages 33–58, 1989.

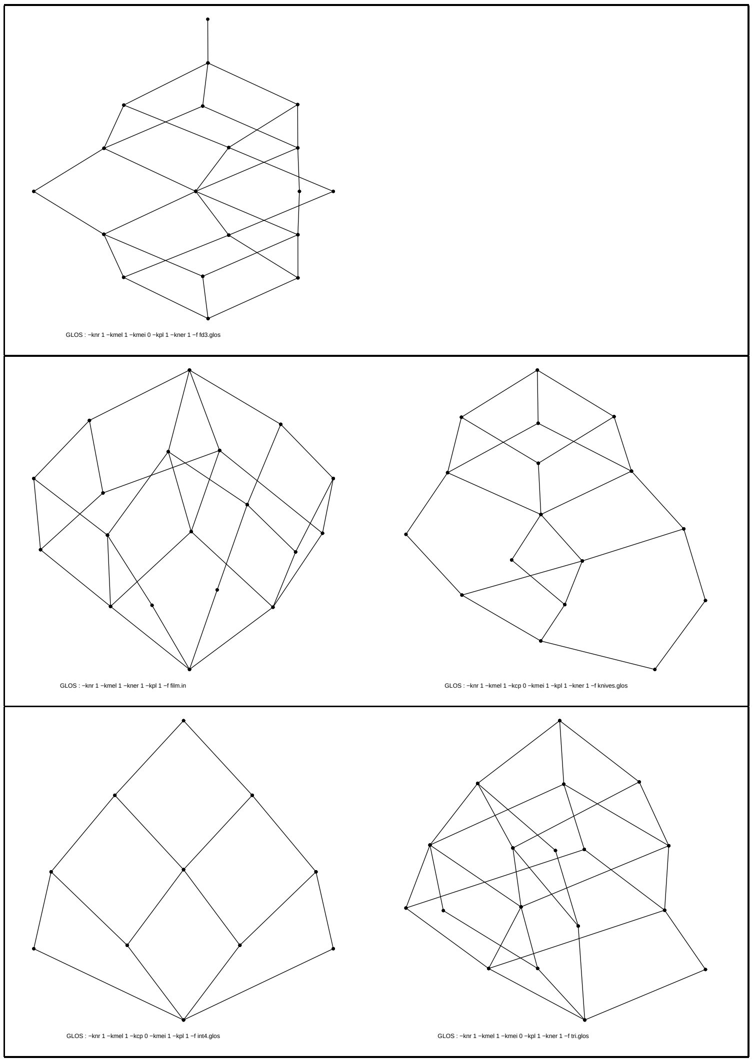

Figure 4. Force directed placement layout results for non-additive diagram. From top left, across and then down the lattices displayed are: FD3, Fig8, Film, Knives, ID4, and Triangles.

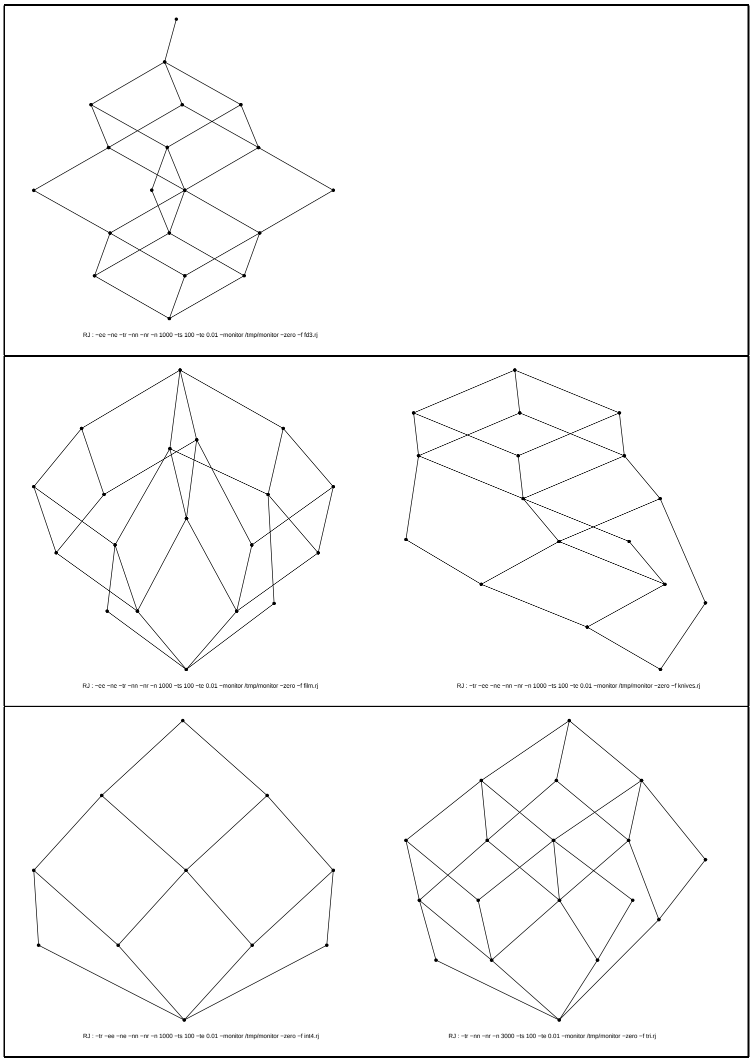

Figure 5. Force directed placement results for additive diagrams. From top left, across and then down the lattices displayed are: FD3, Fig8, Film, Knives, ID4, and Triangles.

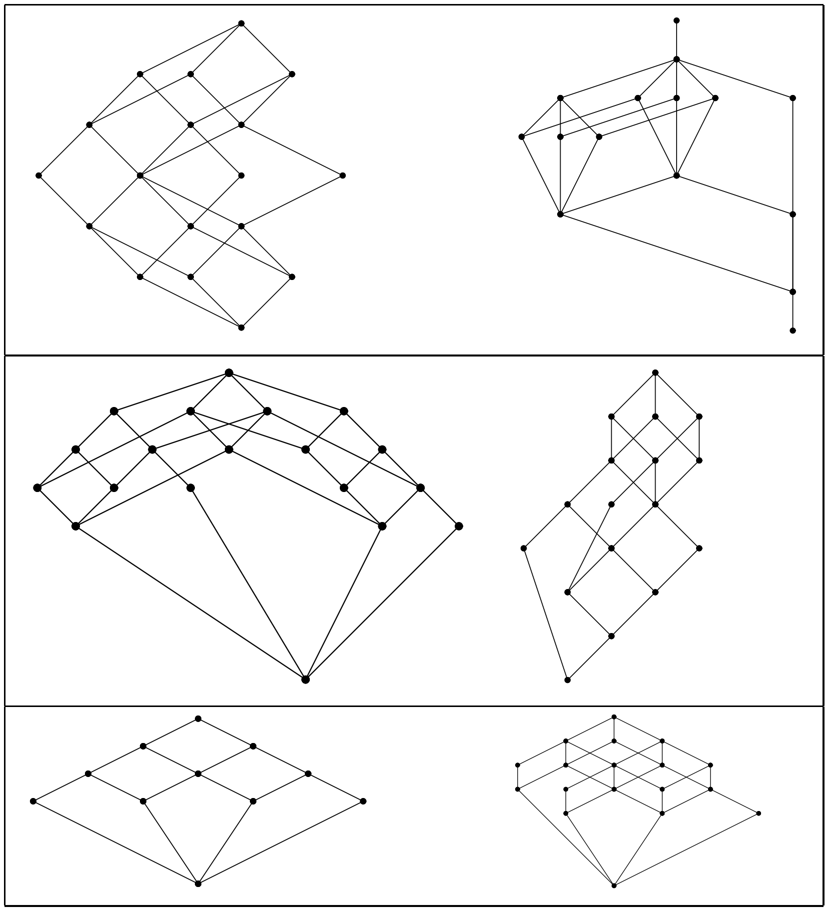

Figure 6. Additive layout using a genetic algorithm. From top left, across and then down the lattices displayed are: FD3, Fig8, Film, Knives, ID4, and Triangles.