Finding the best visualization of an ontology

Christina Valentin Fabritius Nadia Lyngaa Madsen
Jens Clausen Jesper Larsen

Informatics and Mathematical Modelling
Technical University of Denmark
2800 Kgs. Lyngby, Denmark

19th May 2004

Abstract

An ontology is a classification model for a given domain. In information retrieval ontologies are used to perform broad searches. An ontology can be visualized as nodes and edges. Each node represents an element and each edge a relation between a parent and a child element. Working with an ontology becomes easier with a visual representation. An idea is to use the expressive power that a 3D representation to provide visualization for the user.

In this paper we propose a new method for positioning the elements of the visualized concept lattice in the 3D world based on Operations Research (OR) methods. One method uses a discrete location model to create an initial solution and we propose heuristic methods to further improve the visual result. We evaluate the visual results according to our success criteria and the feedback from users. Running times of the heuristic indicate that an improved version should be feasible for on-line processing and what-if analysis of ontologies.

1 Introduction

Ontologies can be represented as hierarchical structures which, if visualized, can make it easier to work with ontologies. Visualization, which is best done in 3D, includes positioning elements of the ontology in parallel 2D planes in the 3D world, so the visual representation of the ontology gives a good overview.

1.1 Ontologies, lattices and concept lattices

An ontology is a classification model for a given domain, e.g. a biological taxonomy is an ontology. In information retrieval ontologies are used to perform broad searches, e.g. for information about cars and thus all specifications of the concept cars. Because some concepts belong to several categories, searches with ontologies can benefit from lattice theory where mathematical properties make it possible to perform algebraic operations for queries.

A lattice is a special type of partial order. It can be viewed as a hierarchical structure where each element can have more than one parent. Thus a parent element may be seen as the union of all its children. Correspondingly, a child element may be seen as the intersection of all of its parents.

An ontology can be viewed as a lattice which contains interrelated concepts in a hierarchical structure. A concept parent element may be seen as the generalization of all its children. Correspondingly, a concept child element may be seen as the specialization of all of its parents. We call lattices that represent ontologies concept lattices.

1.2 Visualization

A lattice can be visualized as nodes and edges. Each node represents an element and each edge a relation between a parent and a child element. Visualization of a concept lattice is interesting as it makes the information contained in the lattice more accessible and easier to work with for humans. A visual representation of a concept lattice can be used to search for information, to create or expand the represented ontology and to find errors in the information represented such as doublets of concepts or invalid relations.

Working with a concept lattice becomes easier with visual representation. However, using the concept lattice to represent and contain all the valuable information make the lattice grow very rapidly. Consequently, visualizing the lattice becomes a complex task, because a user should be able to get an overview of the entire lattice while still being able to work with individual elements. The overview is needed to identify possible clusters in the information while individual interaction with single elements provides detailed information. Currently no visualization tool can handle this dual need for visualization of large lattices.

Visualizing lattices in 2D have performed well for small lattices but in 2D visualization it is impossible to avoid intersecting edges. For large lattices the number of intersecting edges in a 2D visualization becomes too large, thereby spoiling the overview of a visualized concept lattice. An obvious idea is therefore to use the extra expressive power that a 3D representation provides. Of course we will need to display the lattice on the screen by downgrading the final representation to 2D. How-

ever, working in a 3D universe enables us to provide perspective, lighting, shadows, 3D geometry, focusing etc. which the human eye will experience as 3D.

A 3D visualization tool for visualizing concept lattices needs some common viewing and moving functionalities. It is easier to get an overview of the entire concept lattice and the individual elements if the user can move around in the virtual world. Note that we look at the virtual 3D world through a virtual camera, so moving around in the virtual world actually means moving the camera around. Furthermore, the vision of the viewing camera needs to adjust its view according to the elements in focus. Being able to search for elements by their name or selecting them with the mouse on the screen is useful for navigating in the visualized concept lattice.

In order to measure the visual quality of the visualized lattice we set up several criteria for successful visualization. For instance, we require that each element we visualize should be clearly separated from other visualized elements in the visualized concept lattice, with a minimum distance between the neighboring elements. Viewing the relations between all visualized elements should be as easy as possible. However, whether the criteria are fullfilled is a subjective estimate for each user of the visualization tool. We therefore test the final visual results on several users who work with lattices in different ways. We also compare a visualized lattice against what we call *an optimal lattice*. We construct an optimal lattice by positioning each element manually to get a good overview of the lattice as well as a clear separation of each element and its relations.

1.3 Organization of the paper

In section 2 we describe ontologies and concept lattices in more detail and give an example of an ontology. We describe visualization, navigation and geometry in a 3D world and we introduce some existing lattice visualization tools. In section 3 we propose a new method for positioning all elements of the visualized concept lattice in the 3D world using Operations Research (OR) methods. One method uses a discrete location model to create an initial solution and we propose heuristic methods to further improve the visual result. As an alternative approach we consider a continuous location model. In section 4 we evaluate the visual results according to our success criteria and the feedback from several users. We measure the running time of the positioning algorithm and comment on the result. Finally, we draw conclusions and make suggestions for further research.

2 Ontologies and concept lattices

2.1 Ontology

According to the Danish research project OntoQuery [1] "*an ontology is a, possibly simplified, description of an application domain comprising the essential concepts in the domain and the important semantic relationships between these concepts ...*". The core of an ontology is a taxonomy describing the domain concepts and their conceptual inclusion relationships. Besides the taxonomy the ontology comprises a number of semantic relations pertaining to the understanding of the target domain. An exact definition of an ontology is still a subject of research.

Concepts in an ontology can belong to several categories (have several parent elements) as illustrated in Figure 1, which shows a small ontology for animals. Each connecting line represents an *is-a* relationship where, e.g., a black cat *is-a* (specialization of a) cat. The arrow on the line indicates the direction of the relation from specialization to generalization. The structure of an ontology can be represented as an acyclic digraph. Because of the hierachical structure an ontology

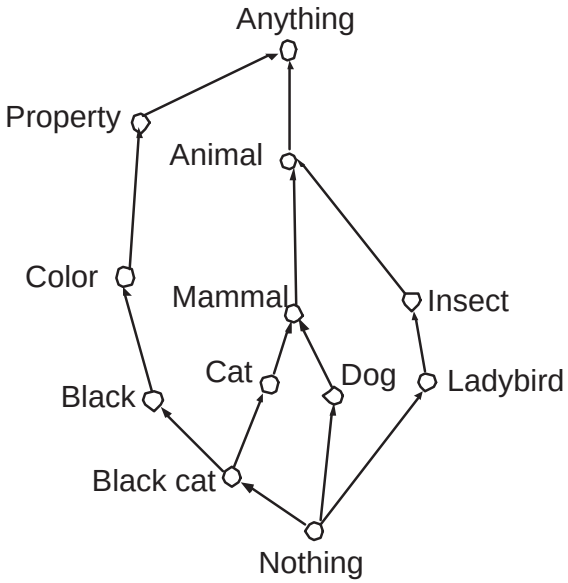

Figure 1: A small ontology for animals with hierachical structure.

can be considered as a concept lattice. Concept lattices are desribed below. Lattices enable an algebraic mode of representation that makes it possible to use the concepts for calculation purposes, but it is also defined as a partially ordered set, which more aptly reflects the *is-a* relations.

2.2 Lattices and concept lattices

The properties of lattices make them well-suited as methodological tools when structuring ontologies. Below, we give the mathematical definition of a lattice and a concept lattice and point out the differences. We assume that the reader is already familiar with the definition of a partially ordered set.

2.2.1 Mathematical definition of lattices.

A lattice is a partially ordered set with additional properties as described by [3, 6].

Definition 1 Let P be a partially ordered set and let $x, y \in P$. An element $d \in P$ is an **upper bound** of the pair (x, y) if $x \leq d$ and $y \leq d$. An element $c \in P$ is the **least upper bound** of (x, y), if c is an upper bound of (x, y) and for any other upper bound d of (x, y) we have that $c \leq d$.

Definition 2 Let P be a partially ordered set and let $x, y \in P$. An element $d \in P$ is a **lower bound** of the pair (x, y) if $d \leq x$ and $d \leq y$. An element $c \in P$ is the **greatest lower bound** of (x, y), if c is a lower bound of (x, y), and for any other lower bound d of (x, y) we have that $d \leq c$.

Definition 3 A **lattice** is a partially ordered set, in which each pair (x, y) of elements have a least upper bound and a greatest lower bound.

Definition 4 A **bounded lattice** is a lattice with two special elements **top** $\top$ and **bottom** $\bot$. $\top$ resp. $\bot$ is an upper resp. a lower bound of any element in the lattice. Every finite lattice is a bounded lattice.

A lattice can be represented by an acyclic digraph $G = (V, E)$ with a vertex v_x for each concept x and an edge for each relation between two concepts. The edge corresponding to the relation $x \leq y$ between two concepts x and y is (v_x, v_y) and is directed from v_x to v_y. With this convention, the direction of an edge is always "from bottom to top" in the case of a bounded lattice, cf. Figure 1.

2.2.2 Concept lattice

An ontology may have a set of relations, which gives rise to a conflict with the definition of a lattice. In an ontology, upper and lower bounds for a pair of elements may be unrelated, because concepts are 'fuzzy', see Figure 2.

A **concept lattice** is a partially ordered set with both a top and a bottom element. This ensures that each pair of elements have a common upper bound and a common lower bound. A bounded lattice is thus a special case of a concept lattice. In general, however, a lattice is not a concept lattice, nor is a concept lattice necessarily a lattice.

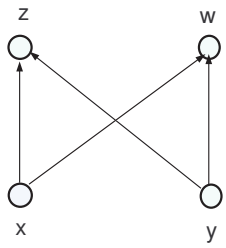

Figure 2: No least parent for (x, y) and and no greatest child element for (z, w) exist. This is valid for a concept lattice, but not for a lattice.

In an ontology, it may happen that no concept is the union of all other concepts. Likewise, there may not be a concept, which is the specialization of all other concepts. In this case we add the elements $\top$ and $\bot$ to the ontology. These can be viewed as the artificial concepts 'anything' and 'nothing', and are introduced to guarantee that any pair of concepts in a concept lattice have at least $\top$ as a common parent and $\bot$ as a common child. In Figure 1 the $\top$ is the concept 'anything'.

2.3 Visualization

2.3.1 Visualizing concept lattices

To enable error finding using a visualization tool, we allow that a given concept lattice visualized may have flaws which contradicts the definition of concept lattices. In the context of visualization, we require that the concept lattice is finite, that a $\top$ and $\bot$ exist, and that it can be represented as an acyclic digraph.

The user can explicitly specify transitive and reflexive edges in the lattice. We choose to visualize the transitive edges, because they may be useful for error finding while the reflexive edges are ignored when visualizing the lattice.

Visualizing lattices and concept lattices means to make a visual representation of their mathematical properties. As a lattice is represented by an acyclic digraph we can illustrate these properties by 1) placing the node corresponding to a parent element above the nodes corresponding to children elements, and 2) to place child elements "clustered" close to their parents. We partition the nodes of the concept lattice in question into subsets, each of which is to be placed in a separate layer of the 3D space. Every parent node is placed at least one layer above all nodes of its children. Thus the top element is placed in its own layer, and likewise with the bottom element. The exact positioning of the elements in their respective layers is a more complex operation. In section 3 we describe how we use OR-methods to solve this problem.

2.3.2 Visualizing in 3D - Geometry

We need to define how the concept lattice itself should be visualized before we move on to which functionalities should exist in the virtual 3D world. As the lattice is visualized in 3D the elements and edges must be 3D as well. We visualize the lattice elements (concepts of the ontology) as spheres and their relations as bars (slim cylinders). Lighting and navigation in the virtual world will ensure a true 3D experience. However, we need to determine the necessary number of 2D planes, slicing up the 3D world that is required in the visualization.

For each element, the lattice data only provides information about its relations and the given concept. The size of the sphere that represents the element, and the length, radius and angle of the bars to its parents and children are the parameters to be chosen as part of the design of the visualization tool.

Once the geometry of the visualized lattice is in place we need to implement navigation tools so we can move around in the virtual world in order to fully utilize the 3D space and get an overview of the lattice.

2.3.3 Navigating in 3D

To further enhance an overview of a visualized lattice at least the following navigation tools should be implemented:

- **Rotation**: Instead of rotating the lattice, wasting valuable calculation resources we orbit the camera around the lattice. To make the illusion of the rotating lattice complete we orbit the light source along with the camera.
- **Movement**: We implement movement of the camera (back, forth, right, and left), manipulated by the arrow keys of the keyboard.
- **Zoom**: We implement a zoom functionality which adjust the angle of the camera lens. Narrowing the angle gives the experience of zooming in and widening the angle zooms out the picture.
- **Searching**: It is possible to search for specific elements by typing the name or the first letters of a concept. Only elements that represent a concept matching the query will be highlighted.
- **Selecting**: Using the mouse, the user can select an element by clicking the mouse button. This causes the current element and its parents to be highlighted. Hands on, this is done by shooting a ray from the camera through the 3D coordinates of the mouse clicking point. Any object which intersects this ray will be highlighted along with its parents.

2.3.4 Methods for Positioning in 3D

As earlier described we must position each element in the virtual world in the 2D plane of its appropriate layer. There are several approaches to this problem, and below we briefly review some of the existing lattice visualization tools.

2.3.5 Other tools

Currently, there are no tools for visualizing large lattices. However, we wish to briefly present two tools for visualizing small lattices. Ralph Freese, University of Hawaii [4], has created a tool for visualizing small lattices, in which he positions the elements of the lattice by using an analogue positioning method. The method uses tractive and repulsive forces until the lattice has reached a stable condition. The method ensures that the elements are placed neither too close nor too far away from each other. The latter is important in order to keep the lattice dense and easy to overview. Freese's tool visualizes lattices in 2D.

A more interesting tool is the Galicia, developed at the University of Montreal [9]. Galicia offers an opportunity of visualizing lattices in 3D by using, similarly, an analogue method. Galicia actually relies on two methods for positioning the elements in 3D. One is using analogue methods to position the elements in 2D before placing the elements on the edge of a sphere. The second is calculating the positions of the elements in 3D, using an analogue method.

Unfortunately, neither of these tools can handle large lattices, large being defined as 300+ elements. The tools are therefore not suitable for concept lattices that often consist of a far higher number of elements.

Visualization tools also exist for trees and graphs. The tools for trees cannot handle elements with more than one parent, and the tools for graphs are not designed to support the view of the hierarchical structures of concept lattices. Therefore, there is a need for creating a new tool for visualizing concept lattices and lattices in general. We have used an Operation Research approach for positioning each element. This is described in the next section.

3 Location of lattice points in 3D

As described in the previous sections, ontologies can be described by concept lattices, and these in turn give rise to acyclic digraphs. The task is to place the vertices of such a digraph in 3D space in such a way that the image, when presented in 2D on a computer screen, is as good as possible. In the following we denote the graph to be considered $G = (V, E)$. Constraints are that the vertices are located in planes orthogonal to the third coordinate axis (the z-axis) so that any vertex is located in a plane with a z-coordinate that is smaller than the z-coordinates of all its immediate

predecessors. The first task is therefore to determine the number of z-planes to be used.

The minimal number of planes necessary corresponds to the longest path in G. Using standard techniques from graph theory [2], this can be calculated by first performing a topological sort of V and then calculating the longest path using a dynamic programming approach. Due to the acyclicity, the resulting algorithm is linear [2]. The number of planes necessary is then equal to the number of vertices in the longest path.

Since the longest path emanates between the top and the bottom vertex of the ontology, these two vertices must be located in the plane with a maximum or a minimum z coordinate, respectively. For each of the other vertices, the number of edges in a longest path from the top vertex determines the plane in which the vertex is located. The remaining task is thus to find the exact location for the vertices in each of the planes.

Two models have been considered: A discrete model, where only grid points may be used as locations for vertices, and a continuous model, where the coordinates chosen for a vertex are continuous. Clearly the continuous model has the advantage that the location of the vertices are not confined to the grid points. On the other hand the discrete model, though hard to solve to optimality, admits a simple heuristic approach, whereas we cannot solve the continuous model because of the complexity of real world problems. The models are described below. In the final visualization tool we have implemented a heuristic to solve the discrete location model.

3.1 A discrete location model for the vertex location problem

We index the planes considered by their z-coordinates. These constitute the set Z, and w.l.o.g. we assume that $Z = \{0, \ldots, m\}$. Within each plane, the feasible grid points are indexed by i and j. The feasible sets for x- and y-coordinates are denoted I and J respectively. We assume that $V = \{1, \ldots, n\}$, i.e. that the vertices of G has been numbered $1, \ldots, n$. V is partitioned into subsets $V_0, \ldots, V_m$ corresponding to the vertices belonging to each of the z-planes. The function $z(\cdot)$ maps the elements of V into their corresponding subsets/planes, with V_m being equal to the top vertex and V_0 being equal to the bottom vertex. Furthermore, we define a_{kp} to be 1 exactly if vertex k belongs to plane p, that is $z(v_k) = p$, and 0 otherwise. Note that a_{kp} can be determined once the plane for each vertex has been decided.

We now introduce the binary variables $x_{kij}, k \in V, i \in I, j \in J$. The variable x_{kij} equals 1 if k is located in the grid point (i, j) of the plane $z(k)$. Each vertex to be located in plane p must now be assigned to some grid point in such a way that no two vertices are located in the same point:

$$\begin{array}{lcll} \sum_{k\in V} a_{kp}x_{kij} & \leq & 1 & (i,j) \in I \times J \\ \sum_{(i,j)\in I\times J} a_{kp}x_{kij} & = & 1 & k \in V \end{array}$$

These are the assignment constraints regarding the location of the vertices in V_p in the grid points in plane p.

We quantify the quality of a feasible location of the vertices by the sum of the lengths of the edges of the resulting 3D representation of the graph G. Denote by $d[(i,j),(v,w),p,r]$ the distance between the grid points (i,j) in plane p and (v,w) in plane r. The objective function to be minimized is:

$$\sum_r \sum_p \sum_{(k_1,k_2)\in E} \sum_{i\in I} \sum_{j\in J} \sum_{v\in I} \sum_{w\in J} a_{k_1p}a_{k_2r}d[(i,j),(v,w),p,r]x_{k_1ij}x_{k_2vw}$$

The problem is a quadratic 0-1 integer programming problem. The general quadratic integer programming problem is NP-hard, and furthermore, the number of binary variables in the formulation is excessive, even for small ontologies. We have therefore chosen to implement a heuristic based on an iterative assignment of vertices to locations.

3.2 A heuristic for iterative location of vertices in planes

Since the top vertex is always the only vertex located in the m-plane, we initially assign it to the point $(0,0)$ in the m-plane. Suppose now that we have located all vertices in the planes $\{m,\ldots,p+1\}$ for some p. Each vertex k to be located in plane p is an immediate successor of one or more vertices positioned in the planes $\{m,\ldots,p+1\}$. Let pred(k) denote the set of predecessors of k. For $q \in \text{pred}(k)$ we denote the actual location of q by $(i(q),j(q))$. Suppose q is located in plane $z(q)$. If k is located in (i,j) in plane p, the cost of this location is $d[(i(q),j(q)),(i,j),z(q),p]$. Therefore, the best location of the vertices V_p to be located in plane p relative to the vertices already located in the planes $\{m,\ldots,p+1\}$, can be found by solving the assignment problem:

$$\begin{array}{lll} \min & \sum_{k\in V_p,i,j\in I,J} \sum_{q\in \text{pred}(k)} d[(i(q),j(q)),(i,j),z(q),p]x_{kij} & \\ \text{st.} & \sum_{k\in V} a_{kp}x_{kij} \leq 1 & (i,j) \in I \times J \\ & \sum_{(i,j)\in I\times J} a_{kp}x_{kij} = 1 & k \in V \end{array}$$

Note that since we do not take into account the costs incurred relative to vertices located in planes $\{p-1,\ldots,0\}$, the objective function is an approximation to the cost defined in Section 3.1.

In our heuristic, we also approximate the distance from a predecessor q to the vertex k to be located. Instead of using the true 3D distance $d[(i(q),j(q)),(i,j),z(q),p]$,

we use $d[(i(q), j(q)), (i, j), p, p]$, that is, the distance from (i, j) to the projection of $z(q)$ on plane p:

Our heuristic contains m iterations each consisting of first calculating the cost of locating $k \in V_p$ at point (i, j) with respect to the immediate predecessors of k as described above, and then solving a linear assignment problem with these costs (for a further description of the linear assignment problem see [5]). The assignment problem is solved using CPLEX version 7.0. Note that we choose to locate $\perp$ in plane 0 even though we do not always include it in the visualization.

The resulting complete location of points in planes is a feasible solution to the problem, but since only approximated costs to predecessors are taken into account it is most likely not optimal. Nevertheless, judging from the visualization provided by the implemented software, the solution quality is quite acceptable from a visualization point of view.

3.3 Improvements of the location method

An obvious way to improve the feasible solution generated by our heuristic is to use the above described iterative heuristic as input to one of the wealth of improvement heuristics based on neighborhood (see eg. [7]). A number of neighborhoods suggest themselves, the most simple one being that a neighboring solution to a given location is obtained by either switching locations of two vertices in the same plane or by moving one vertex in a plane to an unoccupied point in the same plane. The design and implementation of such a heuristic in the next version of the software is currently in progress.

3.4 A continuous location model for the location problem

Instead of using the assignment problem for locating the vertices in each of the planes in the iterative heuristic a continuous location model can be used. In the continuous location model there is not a fixed set of points in each plane, in which all vertices must be located. The objective function is again to minimize the sum of the distances from each vertex to its parents:

$$\min \quad \sum_{p\in\{0,\cdots,m\}} \sum_{k\in V_p} \sum_{q\in \mathrm{pred}(k)} d[(i(q), j(q)), (i(k), j(k)), z(q), p]$$

Note that coordinates for the point, in which k is located, are allowed to vary continuously in the plane p. We now need to exclude that vertices are located too close to each other. In the discrete model this was achieved by the grid imposed in each of the planes. In the continuous model we must add constraints that ensure a predefined distance between each pair of vertices. We get a set of quadratic

constraints each corresponding to a pair of vertices (k, h) to be located in the same plane p:

$$d[(i(k), j(k)), (i(h), j(h)), p, p] \geq Kx_{\text{radius}} \qquad k, h \in V_p,\ k \neq h$$

As noted previously this model has a non-linear objective function and non-linear constraints, and cannot be solved to optimality when applied to real world problems because they are too complex. Therefore we have not considered this model further.

4 Experimental results

In the following we evaluate the result of the visualization of a concept lattice and collect feedback from three users.

1. User A - who evaluates the program as a general visualization tool for lattices (information retrieval).
2. User B - who evaluates the program as an ontology visualizer, which enables searches, error finding, ontology expansion and building (information retrieval, ontology expansion, error finding).
3. User C - a computer graphics engineer, who evaluates the quality of the overview, with suggestions for improvements such as semi-transparency, fogging effect etc.

Furthermore, we measure the time spent using the discrete localization method with various norms.

4.1 Evaluating different measures

We first give an evaluation of the visual result according to our own opinion based on 4 different measures:

- **Euclidean norm**: (see figure 3) Using the Euclidean norm as distance measure results in a fairly good overview of the lattice in spite of the large data set, as well as a good view of the individual elements and their relations.
- **Manhatten norm** and **Max norm**: (see figure 4 and 5) The Manhatten norm and Max norm both provide a good overview of the lattice as well as a good overview of the individual elements and their relations. However, the overview of the entire lattice is somewhat less satisfactory than the ones resulting from using the Euclidean norm. The Manhatten norm and the Max norm provide slightly different results, but none of these methods provide better visual result than the other.

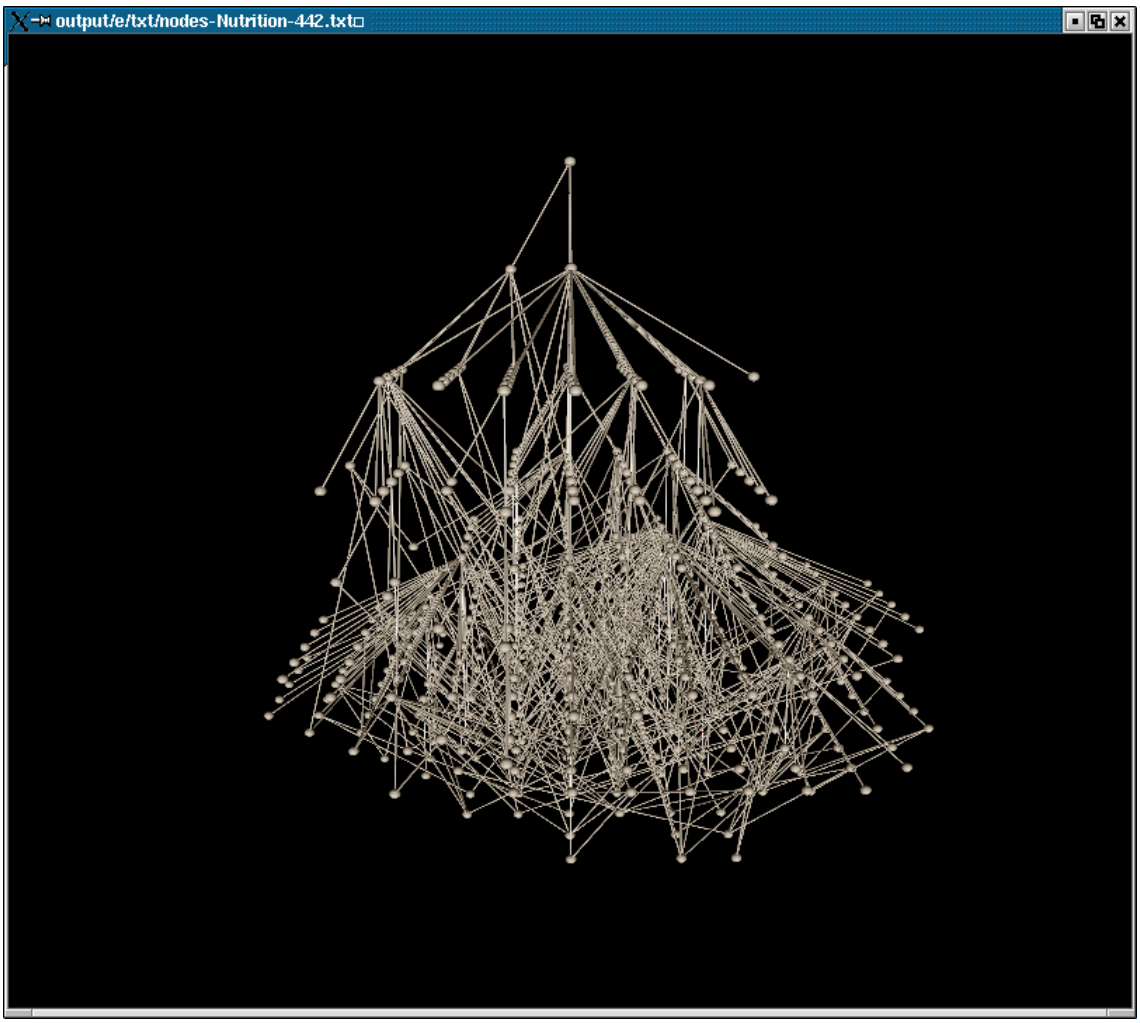

Figure 3: Screenshot of visualizing a concept lattice with 442 elements by means of the Euclidean norm. The Euclidean norm delivers a good visualization of the entire concept lattice.

- **Circle distance**: (see figure 6) The Circle distance measures to the edge of a circle, according to the Euclidean norm: $|\text{Circle}_{\text{Euclidean}}| = (i(k)-i(\text{pred}(k)))^2+(j(k)-j(\text{pred}(k)))^2-r^2$, i and j being coordinates and r the circle radius.

Though providing a good overview of individual nodes and their connections, the circle distance measure fails to give a good overview of the lattice, because elements are placed further away from their parents than for the other distance measures. This increase in distance to the parents results in a higher number of crossed relations, which makes the visual result of the lattice as a whole more confusing.

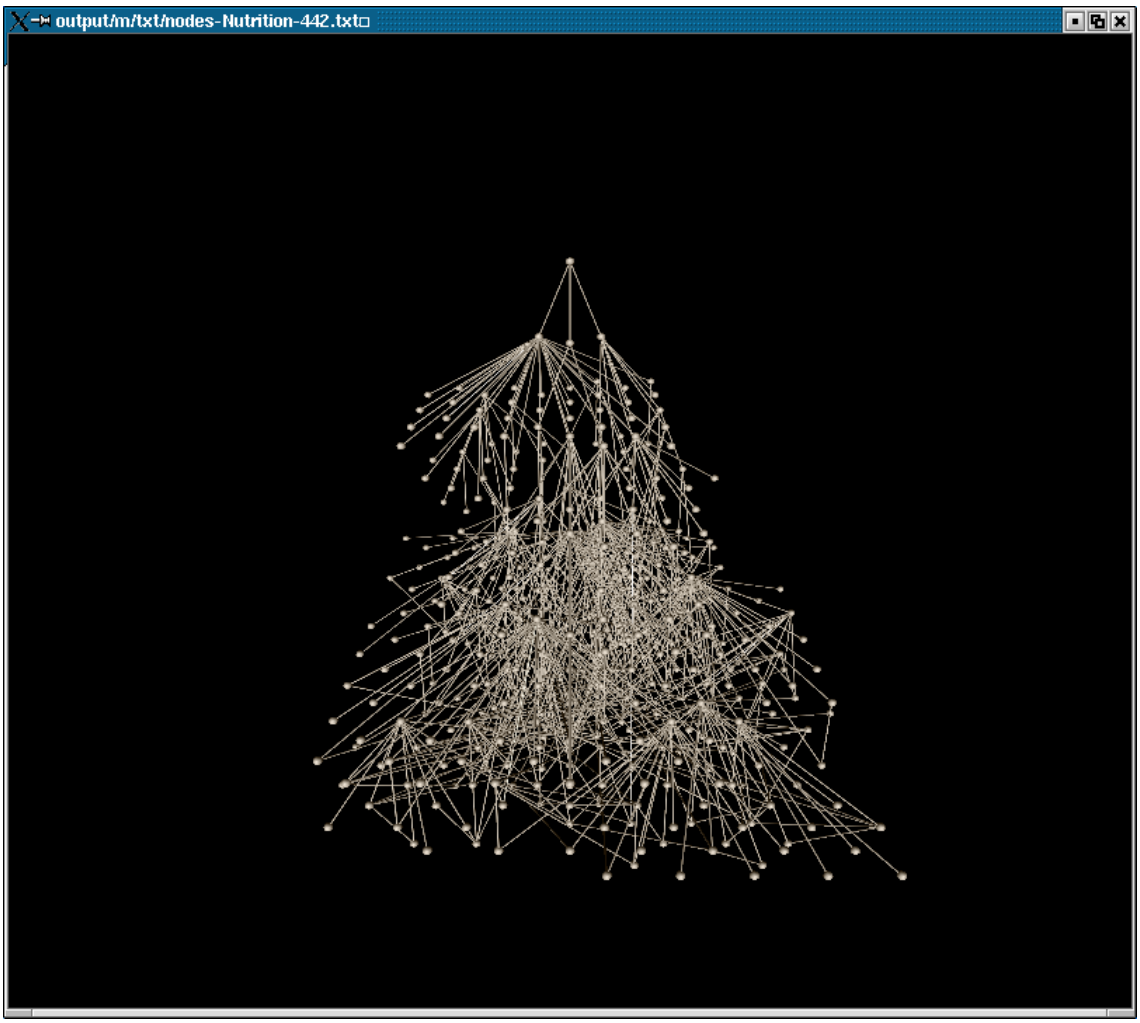

Figure 4: Screenshot of visualizing a concept lattice with 442 elements by means of the Manhatten norm.

4.2 User A

The user accepts the visual results provided by the Euclidean, Manhatten, and Max norms, as distance measures that provide a fine overview of concept lattices. The user does not distinguish so much between the three norms since the differences in the visual results crumble with any increase in the number of elements visualized. Since all three distance measures provide a good overview, the user focuses on the functionalities of the visualization program, which could further improve the use of the result.

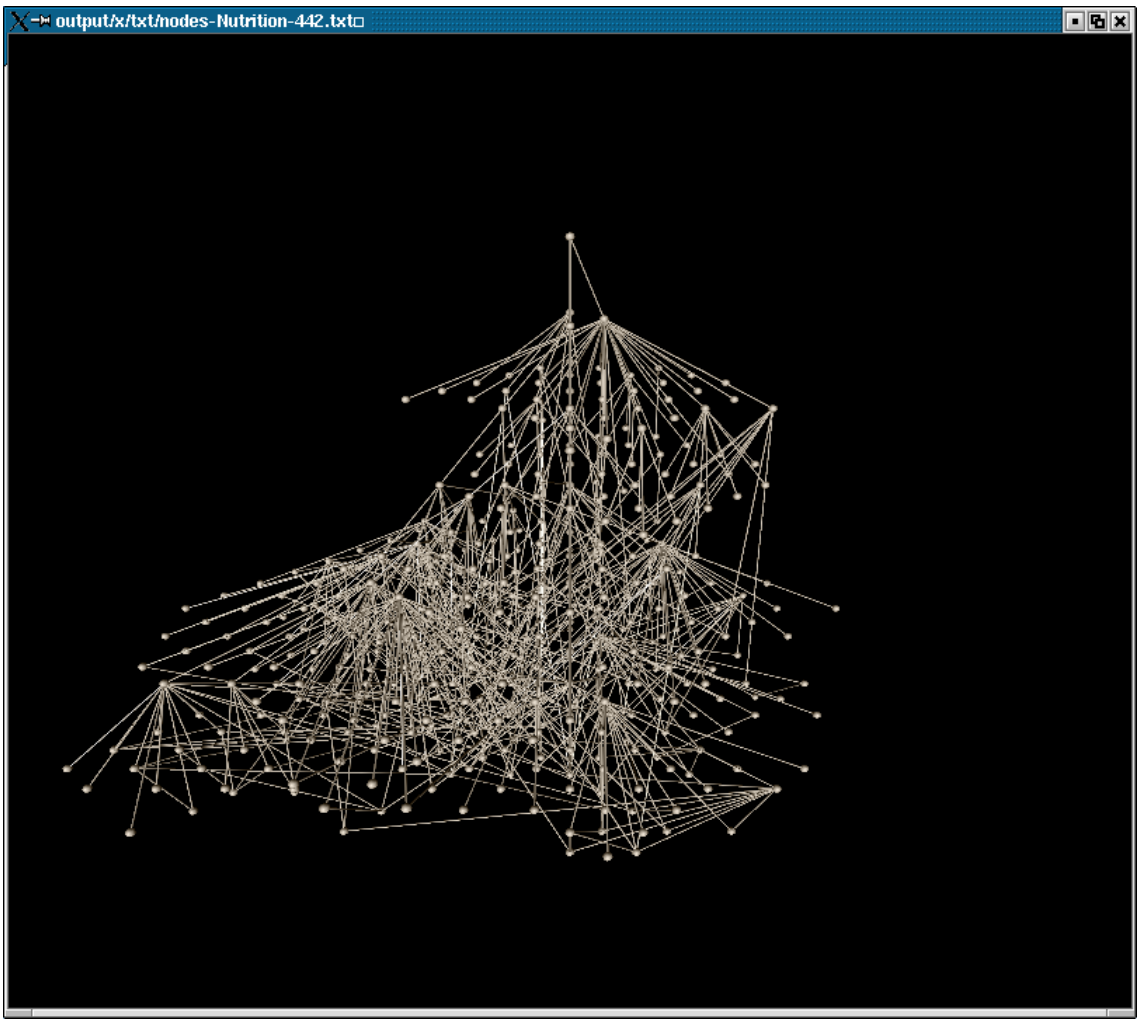

Figure 5: Screenshot of visualizing a concept lattice with 442 elements by means of the Max norm.

4.3 User B

The user succesfully used the program for error finding in his data set. The user notes that the Euclidean, Manhatten, and Max norm all provide a good overview of the concept lattice, e.g. by well visualiazing chunks of related data in the concept lattice. Using the circle distance measure does not offer the same visual advantages.

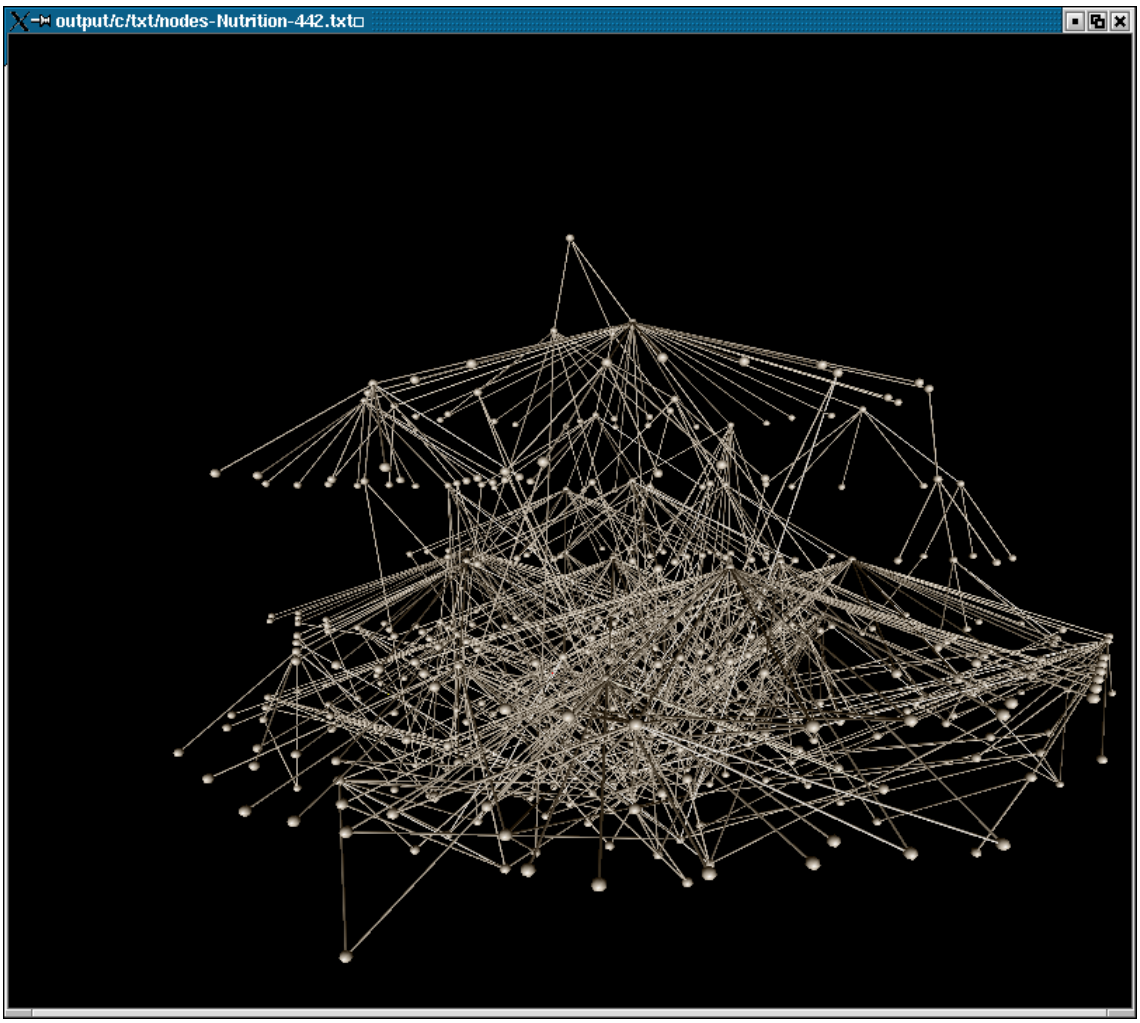

Figure 6: Screenshot of visualizing a concept lattice with 442 elements by means of the circle distance, using the Euclidean norm. The elements are placed at the disc around the parent element.

4.4 User C

The user is not so much interested in the data visualized as in *how* it is visualized. The user suggests improvements in the current 3D visualization for a better overview of the concept lattice, such as blurring elements and relations and make them semi-transparent dependent on their distance to the virtual world camera.

4.5 Running time

We use the discrete localization method described in section 3 and test the four distance measures: Euclidean norm, Manhatten norm, Max norm, and circle distance. The results are listed in table 1. The running times are of course dependent on the number of elements that are to be placed and the number of layers that the lattice spans over. The current version of the code has not been optimized with respect to running time.

From the table we see that processing a concept lattice with 1364 elements takes about 6 minutes, which is an acceptable time span. However, we expect these numbers to decrease through optimization of the program.

No. of elements	Time (sec)			
	Euclidean	Manhatten	Max	$\text{Circle}_{\text{Euclidean}}$
8	0.02	0.01	0.01	0.01
36	0.04	0.04	0.04	0.04
37	0.06	0.05	0.05	0.07
63	0.12	0.11	0.11	0.11
66	0.15	0.14	0.14	0.14
159	0.86	0.83	0.83	0.86
160	0.79	0.74	0.76	0.75
248	2.68	2.65	2.44	3.06
331	8.26	5.45	6.47	8.09
332	7.96	5.88	5.79	7.93
414	12.45	10.58	9.13	12.25
442	16.76	11.59	11.30	8.83
853	63.17	54.45	46.56	61.65
857	112.68	66.36	61.27	68.56
892	65.44	47.33	47.28	64.22
1364	321.91	175.55	172.54	133.61

Table 1: Running time using the discrete localization method for the Euclidean norm, Manhatten norm, Max norm, and circle distance using the Euclidean norm.

5 Conclusion and suggestions for further work

We have taken an untraditional approach to solve the positioning problem of elements in the context of visualizing ontologies using optimization-based methods from OR.

Two optimization models have been developed, one being discrete and one being continuous. Both models are quadratic, however, the discrete model has linear constraints and is, though the problem is NP-hard, tractable with heuristics, based on classical optimization methods.

The discrete model has been used to develop a heuristic, based on the iterative application of the classical assignment problem, to find a high quality visualization of a given ontology. The running times of the heuristic indicate that an improved version might be feasible for on-line processing and what-if analysis of ontologies. Regarding the quality of the visualization, we intend to enhance the current heuristic with a local search improvement heuristic, which uses the current solution as the starting solution.

From an ontology user's point of view, the method produces high quality results. Even the current initial version of the tool proves to be a good alternative to, e.g., analogue methods.

The current visualization tool can also be further improved in visualization and navigation. The virtual world may include fogging effects and use semi-transparent elements to further enhance the elements in focus. For navigation, users working with lattices may wish to extend query options, e.g. locating common parents of pairs or clusters of elements.

To make the tool compatible with other applications, it is necessary to standardize its input and output format for lattices, e.g. by using the SLF or XML standards. We intend to make the tool available on-line, so that a user feeds the tool with an input file and receives a small application with the resulting visualization.

References

[1] Hans Bruun, Jørgen Fischer Nilsson & Nikolaj Oldager: http://www.ontoquery.dk/description/index.php, 2004.

[2] W.J. Cook, W.H. Cunningham, W.R. Pulleyblank, A. Schrijver, *Combinatorial Optimization*, Wiley 1998.

[3] Jørgen Fischer Nilsson & Nikolaj Oldager, *Introduction to Orders and Lattices*, 2002.

[4] Ralph Freese, *Lattice Drawing.*. www.math.hawaii.edu/∼ralph/LatDraw.

[5] F. S. Hillier, G. J. Lieberman, *Introduction to Operations Research*, McGraw-Hill 2001.

[6] Mathworld. http://mathworld.wolfram.com/Lattice.html, 2004.

[7] C. R. Reeves, *Modern heuristic techniques for combinatorial problems*, McGraw-Hill 1995.

[8] Websters Online Dictionary. http://dictionary.reference.com.

[9] University of Montreal, *Galicia* – Galois lattice interactive constructor. http://www.iro.umontreal.ca/∼galicia/visualization.htm.