Computing Graph-Based Lattices from Smallest Projections

Sergei O. Kuznetsov

State University Higher School of Economics (SU HSE),
Pokrovskii bd. 11, Moscow 109028, Russia
skuznetsov@hse.ru

Abstract. From the mathematical perspective, lattices of closed descriptions, which arise often in practical applications can be reduced to concept lattices by means of the Basic Theorem of Formal Concept Analysis (FCA). From the computational perspective, in many cases it is more advantageous to process closed descriptions and their lattices directly, without reducing them to concept lattices. Here a method for computing lattices with descriptions given by sets of graphs, starting with rough approximations is considered and compared to previous approaches.

1 Introduction

Recently, the problem of analyzing data given by labeled (hyper)graphs attracted much attention in various computer science communities due to its importance in many applications, from chemistry to text analysis [2, 9, 14, 15, 17, 20, 28–30]. Like in Data mining of 1990s the researchers came to the idea of a closed graph which can be very useful for defining association rules on graphs: As reported in [30], CloseGraph algorithm computes frequent graphs much faster than its forerunner gSpan [29], and WARMR [15], an ILP program.

As for FCA, a lattice on (closed) sets of labeled graphs, representing molecules, was proposed much earlier [18–21]. The lattice on graph sets is induced by an operation which takes a pair of graphs to the set of its maximal common subgraphs. This operation induces a meet (infimum) operation on sets of labeled (hyper)graphs: it is idempotent ($X \sqcap X = X$), commutative ($X \sqcap Y = Y \sqcap X$), and associative ($X \sqcap (Y \sqcap Z) = (X \sqcap Y) \sqcap Z$). These properties allow one to compute similarity of graph sets by means of algorithms for computing closed sets (see review [16]) well-known in Formal Concept Analysis [12].

In [11] we described a general framework, called pattern structures, which allows one to define similar lattices for sets of arbitrary partially-ordered descriptions and relate them to concept lattices using the Basic Theorem of FCA.

The main problem in practical implementation of these lattices is that of computational complexity, e.g., in case of graph sets even testing subgraph isomorphism is an NP-complete problem, the problem of computing common maximal subgraphs of two graphs being NP-hard.

K.E. Wolff et al. (Eds.): KONT/KPP 2007, LNAI 6581, pp. 35–47, 2011.

A theoretical means for approximate computation in semilattices, called projections of pattern structures, was proposed in [11] and the first computer implementation was described in [7].

The algorithm computing the lattice of graph sets described in [20] constructs the lattice (seen in the same perspective as a concept lattice) in a bottom-up way, starting from object "intents" (given by graphs). This algorithm is a simple modification of a standard FCA algorithm, however it does not realize the idea of constructing a sequence of projections, starting with the roughest one, and refining it until an appropriate level [11]. In standard FCA, this would be an algorithm which proceeds from attribute extents, however, one does not have attributes when working with graph sets.

In this paper we propose an algorithm which constructs the lattice of graph sets in a top-down way, starting from smallest subgraphs of the graphs in a dataset. We show the advantages of this algorithm over the previous, bottom-up algorithm.

The paper is organized as follows. In the second section we describe the general theoretical framework for computing similarity (meet) of graph sets together with a means for its approximate computations. In the third section we discuss a method for analyzing graph datasets based on the framework and discuss its drawbacks. In the fourth section we propose a new algorithm, discuss its complexity and advantages over the previous approach.

2 Pattern Structures on Sets of Graphs

In [18–20] a semilattice on sets of graphs with labeled vertices and edges was proposed and in [11] this semilattice was generalized to arbitrary pattern structures. As in the general case, the lattice on graph sets is based on a natural "containment" (i.e., in case of graphs, subgraph isomorphism) relation between graphs with labeled vertices and edges. Consider an ordered set P of connected graphs[1] with vertex and edge labels from the set $\mathcal{L}$ with partial order $\preceq$. Each labeled graph Γ from P is a quadruple of the form $((V, l), (E, b))$, where V is a set of vertices, E is a set of edges, $l\colon V \to \mathcal{L}$ is a function assigning labels to vertices, and $b\colon E \to \mathcal{L}$ is a function assigning labels to edges.

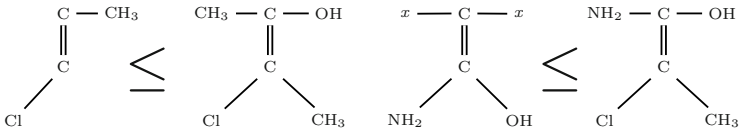
For two graphs $\Gamma_1 := ((V_1, l_1), (E_1, b_1))$ and $\Gamma_2 := ((V_2, l_2), (E_2, b_2))$ from P we say that Γ_1 **dominates** Γ_2 or $\Gamma_2 \leq \Gamma_1$ (or Γ_2 is a **subgraph** of Γ_1) if there exists an injection $\varphi\colon V_2 \to V_1$ such that it

- respects edges: $(v, w) \in E_2 \Rightarrow (\varphi(v), \varphi(w)) \in E_1$,
- fits under labels: $l_2(v) \preceq l_1(\varphi(v))$, $(v, w) \in E_2 \Rightarrow b_2(v, w) \preceq b_1(\varphi(v), \varphi(w))$.

Obviously, $(P, \leq)$ is a partially ordered set.

Example 1. Let $\mathcal{L} = \{\mathrm{C}, \mathrm{NH_2}, \mathrm{CH_3}, \mathrm{OH}, x\}$ then we have the following relations:

[1] Omitting the condition of connectedness, one obtains a (computationally harder) model that accounts for multiple occurrences of subgraphs.



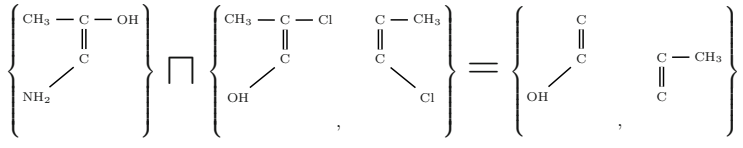
Now a *similarity operation* $\sqcap$ on graph sets can be defined as follows: For two graphs X and Y from P

$$\{X\} \sqcap \{Y\} := \{Z \mid Z \leq X, Y,\ \forall Z_* \leq X, Y\ \ Z_* \not\geq\ \ Z\},$$

i.e., $\{X\} \sqcap \{Y\}$ is the set of all maximal common subgraphs of graphs X and Y. Similarity of non-singleton sets of graphs $\{X_1, \ldots, X_k\}$ and $\{Y_1, \ldots, Y_m\}$ is defined as

$$\{X_1, \ldots, X_k\} \sqcap \{Y_1, \ldots, Y_m\} := \mathrm{MAX}_{\leq}(\cup_{i,j}(\{X_i\} \sqcap \{Y_j\})),$$

where $\mathrm{MAX}_{\leq}(X)$ returns maximal (w.r.t. $\leq$) elements of X. Here is an example of applying $\sqcap$:



The similarity operation $\sqcap$ on graph sets is commutative: $X \sqcap Y = Y \sqcap X$ and associative: $(X \sqcap Y) \sqcap Z = X \sqcap (Y \sqcap Z)$.

A set X of labeled graphs from P for which $\sqcap$ is idempotent, i.e., $X \sqcap X = X$ holds, is called a *pattern* of P. For patterns we have $\mathrm{MAX}_{\leq}(X) = X$. For example, for each graph $g \in P$ the set $\{g\}$ is a pattern. On the contrary, for $\Gamma_1, \Gamma_2 \in P$ such that $\Gamma_1 \leq \Gamma_2$ the set $\{\Gamma_1, \Gamma_2\}$ is not a pattern. Denote by D the set of all patterns of P, then $(D, \sqcap)$ is a semilattice with infimum (meet) operator $\sqcap$. The natural subsumption order on patterns is given by

$$c \sqsubseteq d :\Longleftrightarrow\ c \sqcap d = c.$$

Let E be a set of example names, and let $\delta : E \to D$ be a mapping, taking each example name to $\{g\}$ for some labeled graph $g \in P$ (thus, g is the "graph description" of example e). The triple $(E, (D, \sqcap), \delta)$ is a particular case of a *pattern structure* [11]. Another example of an operation $\sqcap$ may be the following semilattice on closed intervals from [19]: for $a, b, c, d \in R$, $[a, b] \sqcap [c, d] = [\max\{a, c\}, \min\{b, d\}]$ if $[a, b]$ and $[c, d]$ overlap, otherwise $[a, b] \sqcap [c, d] = \emptyset$. This semilattice, where numbers are values of the activation energy (computed for molecules by a standard procedure, e.g. see [32]) was used in predicting toxicity of alcohols and halogen-substituted hydrocarbons (see Section 4). The resulting similarity semilattice in this application is that on pairs, where the first element is a graph set and the second element is a numerical interval.

Derivation operators are defined as

$$A^\diamond := \sqcap_{e \in A}\ \delta(e) \qquad \text{for } A \subseteq E$$

and

$$d^\diamond := \{e \in E \mid d \sqsubseteq \delta(e)\} \qquad \text{for } d \in D.$$

For $a, b \in D$ the *pattern implication* $a \to b$ holds if $a^\diamond \subseteq b^\diamond$, and the *pattern association rule* $a \longrightarrow_{c,s} b$ with *confidence* c and *support* s holds if $\frac{|a^\diamond \cap b^\diamond|}{|G|} \geq s$ and $\frac{|a^\diamond \cap b^\diamond|}{|a^\diamond|} \geq c$. Implications are the exact association rules, i.e., association rules with confidence = 1. Operator $(\cdot)^{\diamond\diamond}$ is a closure operator on patterns, since it is

idempotent: $d^{\diamond\diamond\diamond\diamond} = d^{\diamond\diamond}$,
extensive: $d \sqsubseteq d^{\diamond\diamond}$,
monotone: $d^{\diamond\diamond} \sqsubseteq (d \sqcup c)^{\diamond\diamond}$.

For a set X the set $X^{\diamond\diamond}$ is called *closure* of X. A set of labeled graphs X is called *closed* if $X^{\diamond\diamond} = X$. This definition is related to the notion of a closed graph [30], which is important for computing association rules between graphs. Closed graphs are defined in [30] in terms of "counting inference" as follows.

Given a labeled graph dataset D, the support of a graph g or $support(g)$ is the set (or the number) of graphs in D, in which g is a subgraph. A graph g is called *closed* if no supergraph f of g (i.e., a graph such that g is isomorphic to a subgraph of f) has the same support.

Note that the definitions distinguish between a closed graph g and the closed set $\{g\}$ consisting of one graph g. Closed sets of graphs form a *meet semilattice* w.r.t. the infimum or meet operator. A finite meet semilattice is completed to a lattice by introducing a unit (maximal) element. Closed graphs do not have this property, since in general, there can be no supremum and/or no infimum of two given closed graphs. Let a dataset described by a pattern structure $(E, (D, \sqcap), \delta)$ be given.

Proposition. The following two properties hold for a pattern structure $(E, (D, \sqcap), \delta)$:

1. For a closed graph g there is a closed set of graphs G such that $g \in G$.
2. For a closed set of graphs G and an arbitrary $g \in G$, graph g is closed.

Proof. 1. Consider the closed set of graphs $G = \{g\}^{\diamond\diamond}$. Since G consists of all maximal common subgraphs of graphs that have g as a subgraph, G contains as an element either g or a supergraph f of g. In the first case, property 1 holds. In the second case, we have that each graph in G that has g as a subgraph also has f as a subgraph, so f has the same support as g, which contradicts the fact that g is closed. Thus, $G = \{g\}^{\diamond\diamond}$ is a closed set of graphs satisfying property 1. 2. Consider a closed set of graphs G and $g \in G$. If g is not a closed graph, then there is a supergraph f of it with the same support as g has and hence, with the same support as G has. Since G is the set of all maximal common subgraphs of the graphs describing examples from the set $G^\diamond$ (i.e, its support), $f \in G$ should

hold. This contradicts the fact that $g \in G$, since a closed set of graphs cannot contain as elements a graph and a supergraph of it (otherwise, its closure does not coincide with itself). □

Therefore, one can use algorithms for computing closed sets of graphs, e.g., the algorithm in [20], to compute closed graphs. With this algorithm one can also compute all *frequent* closed sets of graphs, i.e., closed sets of graphs with support above a fixed *minsup* threshold (by introducing a minor variation of the condition that terminates computation branches).

Computing $\sqcap$ may require considerable computation resources: even testing $\sqsubseteq$ is NP-complete. To approximate graph sets we consider projection (kernel) operators [11], i.e. mappings of the form $\psi\colon D \to D$ that are

monotone: if $x \sqsubseteq y$, then $\psi(x) \sqsubseteq \psi(y)$,
contractive: $\psi(x) \sqsubseteq x$, and
idempotent: $\psi(\psi(x)) = \psi(x)$.

Any projection of the semilattice $(D, \sqcap)$ is $\sqcap$-preserving, i.e., for any $X, Y \in D$

$$\psi(X \sqcap Y) = \psi(X) \sqcap \psi(Y),$$

which helps us to relate learning results in projections to those with initial representation (see Section 3).

In our computer experiments in [22] we used several types of projections of sets of labeled graphs that are natural in chemical applications:

- *k-chain* projection: a set of graphs X is taken to the set of all chains with k vertices that are subgraphs of at least one graph of the set X;
- *k-vertex* projection: a set of graphs X is taken to the set of all subgraphs with k vertices that are subgraphs of at least one graph of the set X;
- *k-cycles* projection: a set of graphs X is taken to the set of all subgraphs consisting of k adjacent cycles of a minimal cyclic basis of at least one graph of the set X.

3 Analyzing Graph Datasets Using Lattice-Based Approaches

JSM-hypotheses were defined in [6] by means of a special logical language for standard object-attribute representation. These hypotheses were redefined as *JSM-* or *concept-based hypotheses* in [10, 11, 19, 20] in terms of Formal Concept Analysis (FCA). For graph sets hypotheses can be defined as follows. Suppose we have a set of positive examples E_+ and a set of negative examples E_- w.r.t. a target attribute.

A graph set $h \in D$ is a *positive hypothesis* iff

$$h^\diamond \cap E_- = \emptyset \text{ and } \exists A \subseteq E_+ : A^\diamond = h.$$

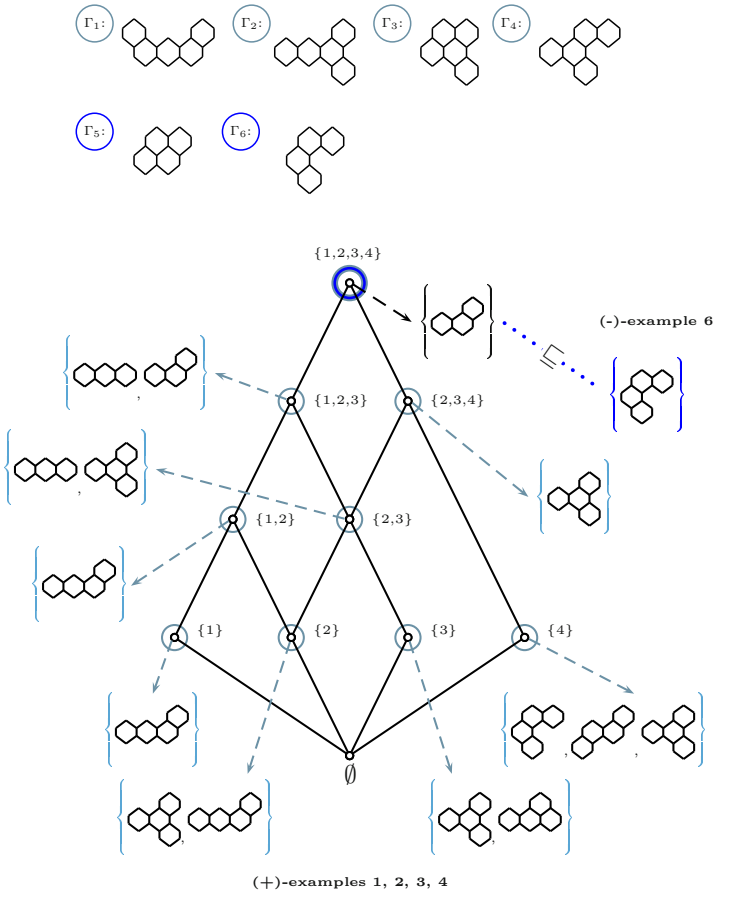


Fig. 1.

Informally, a positive hypothesis is a similarity of positive examples, which does not cover any negative example. A *negative hypothesis* is defined analogously, by interchanging + and −.

The meet-preserving property of projections implies that a hypothesis H_p in data under projection ψ corresponds to a hypothesis H in the initial representation for which the image under projection is equal to H_p, i.e., $\psi(H) = H_p$.

Hypotheses are used for classification of undetermined examples along the lines of [6] in the following way. If e is an undetermined example (example with the unknown target value), then a hypothesis h with $h \sqsubseteq \delta(e)$ is *for the positive classification* of g if h is a positive hypothesis and h is *for the negative classification* of e if h is a negative hypothesis.

An undetermined example e is *classified positively* if there is a hypothesis for its positive classification and no hypothesis for its negative classification. Example e is *classified negatively* in the opposite case. If there are hypotheses for

both positive and negative classification, then some other methods (e.g., based on standard statistical techniques) may be applied. Obviously, for classification purposes it suffices to use only hypotheses minimal w.r.t. subsumption $\sqsubseteq$.

Notwithstanding its simplicity, the model of learning and classification with concept-based hypotheses proved to be efficient in numerous computer experiments, including PTC competition [4, 25].

An algorithm for computing hypotheses on closed graph sets was described in [20]. In [22] the algorithm was realized by simulating $\sqcap$ operation with usual set-theoretic intersection $\cap$ in the following way. For each example e described by a labeled graph $\delta(e)$ first a set of all subgraphs of $\delta(e)$ is computed up to the projection level $k = N$. Each such subgraph is declared to be a binary attribute and example e is represented by the set $S(e)$ of binary attributes that correspond to subgraphs of $\delta(e)$. For two examples e_1 and e_2 the intersection $S(e_1) \cap S(e_2)$ is equivalent to finding the similarity $\psi(\delta(e_1)) \sqcap \psi(\delta(e_2))$.

Example 2. In Figure 1 consider JSM-hypotheses for the dataset with positive examples described by graphs Γ_1,..,Γ_4 and negative examples described by graphs Γ_5 and Γ_6. Here $\Gamma_1 \sqcap \Gamma_2 \sqcap \Gamma_3$ and $\Gamma_2 \sqcap \Gamma_3 \sqcap \Gamma_4$ are minimal positive hypotheses, whereas $\Gamma_1 \sqcap \Gamma_2 \sqcap \Gamma_3 \sqcap \Gamma_4$ is not a positive hypothesis.

Standard Weka procedures for C4.5, Naive Bayes and JRip were run [22] in QuDA environment [13, 26]. Computing concept-based hypotheses in QuDA is realized by means of algorithms for computing lattices of closed sets (or concept lattices), see review [16].

After that *reduction of attributes* [12] was performed: each column of the example/attribute binary table that is equal to the (component-wise) product (conjunction) of some other columns, was removed. Reduction guarantees [12] that thus reduced set of columns gives rise to the isomorphic lattice of closed sets of attributes and thus, to the same set of concept-based hypotheses as defined above.

The whole **PBRL** (project-binarize-reduce-learn) procedure proposed in [22] looks as follows:

1. For each example e and for k compute i-projections of $\delta(e)$ for $1 \leq i \leq k$. The subgraphs from these projections are declared to be attributes;
2. Compose example/attribute context;
3. For each learning method LM run LM, classify examples from test sets, compute cross validation;
4. Clarify and reduce the binary (example/attribute) context;
5. For reduced context and for each learning method LM run LM, classify examples from test sets, compute cross-validation.

Using this approach we analyzed several chemical datasets[2] in [22]. For each dataset we computed graph projections: mostly, k-vertex projections and k-cycles projections for the dataset with polycyclic aromatic hydrocarbons. Every subgraph of each graph in the projection (up to isomorphism) was declared to

[2] These datasets can be downloaded from `http://ilp05-viniti.narod.ru`

be a binary attribute, so each graph dataset was turned into a binary object-attribute table, which was then reduced. For each dataset we ran several learning methods realized within QuDA environment (JSM or concept-based hypotheses, induction of decision trees by C4.5, Naive Bayes, JRip) comparing the results with those based on same learning methods and a chemical (descriptor) attribute language, called FCSS [1], with predefined descriptors.

Although the results of comparison were in favor of graph representation, the major problem of this approach that we encountered in [22] was that of space complexity: although usually the reduced set of attributes was feasible, the initial set of attributes before reduction was too large. For databases we studied, we usually could not produce projections of size larger than 9 for all graphs in the database. By the definition of a projection, one cannot do it object-incrementally, but one can do it attribute-incrementally. For graph projections this means that one should start from smallest subgraphs of graphs from a dataset and proceed by increasing the graph size. Another point is that instead of considering k-projections of graphs, one can consider closures of these projections, since they are subgraphs of the same graphs in the dataset as the initial k-projections. Thus, we come up to the necessity of designing an algorithm that would construct the pattern lattice on graph sets in the top-down way: starting with smallest possible graphs.

4 A Top-Down Algorithm

In this section we propose a top-down algorithm for computing the set of all pattern concepts together with the covering relation on the set of concepts (i.e., graph of the diagram).

Assume that graph edges are unlabeled. This does not result in the loss of generality, since the case with labeled edges can be reduced to the one with labeled vertices. Let the set of vertex labels be $L = \{l_1, \ldots, l_n\}$, the vertex labeling function is denoted by $l(\cdot)$ and $\mathcal{G} = \{\Gamma_1, \ldots, \Gamma_m\}$ denotes a set of graphs. For $i \in [1, m]$, let $\Gamma_i = (V_i, E_i)$ and for $j \in [1, |V_i|]$ let v_i^j denote the jth vertex of Γ_i. Let also $k \in [1, n]$, and Λ be a special symbol.

if $k < n$ then $\mathbf{next}(\mathcal{G}, \Gamma_i, v_i^j, a_k) = (\mathcal{G}, \Gamma_i, v_i^j, a_{k+1})$ else
if $j < |V_i|$ then $\mathbf{next}(\mathcal{G}, \Gamma_i, v_i^j, a_k) = (\mathcal{G}, \Gamma_i, v_i^{j+1}, a_1)$ else
if $i < m$ then $\mathbf{next}(\mathcal{G}, \Gamma_i, v_i^j, a_k) = (\mathcal{G}, \Gamma_{i+1}, v_{i+1}^1, a_1)$
else $\mathbf{next}(\mathcal{G}, \Gamma_i, v_i^j, a_k) = \Lambda$.

The function "+" below realizes the idea of a "minimal extension" of a graph set. If we consider only connected subgraphs, the function $+(\mathcal{G}, \Gamma_i, v_i^j, a_k)$ is defined as follows: $+\Lambda = \Lambda$,

$$+(\mathcal{G}, \Gamma_i, v_i^j, a_k) := (\mathcal{G} \setminus \Gamma_i) \cup \Gamma_i^*,$$

where

$$\Gamma_i^* = (V_i^*, E_i^*), V_i^* = V_i \cup \{v\}, E_i^* = E_i \cup (v_i^j, v), v \notin V_i, l(v) = a_k.$$

If one considers not necessarily connected subgraphs, then the definition of **next** and "+" should be made independent of the "connecting" vertex v_i^j, e.g., for "+" operation as

$$+(\mathcal{G}, \Gamma_i, a_k) := (\mathcal{G} \setminus \Gamma_i) \cup \Gamma_i^*,$$

where

$$\Gamma_i^* = (V_i^*, E_i), V_i^* = V_i \cup \{v\}, v \notin V_i, l(v) = a_k.$$

Further on we consider only the case of connected subgraphs, the case with general subgraphs is treated similarly.

By definitions, $(+\mathbf{next}(\mathcal{G}, \Gamma_i, v_i^j, a_k))^\diamond$ is the set of objects with common description $(+\mathbf{next}(\mathcal{G}, \Gamma_i, v_i^j, a_k))$, i.e., a pattern extent. The pattern

$$(+\mathbf{next}(\mathcal{G}, \Gamma_i, v_i^j, a_k))^{\diamond\diamond}$$

is the closure of the pattern

$$(+\mathbf{next}(\mathcal{G}, \Gamma_i, v_i^j, a_k)).$$

We set by definition $\Lambda^{\diamond\diamond} = \Lambda^\diamond = \emptyset$.

If X stays for a pattern concept $(\mathcal{G}^\diamond, \mathcal{G})$ the function $\mathbf{covers}(X)$ computes concepts which X covers in the lattice order, i.e., taking the set

$$\bigcup_{i,j,k} (+(\mathbf{next}(\mathcal{G}, \Gamma_i, v_i^j, a_k)))^\diamond$$

it returns concepts that correspond to maximal extents of this set. This function can be efficiently realized (in $O(|X|^2 \cdot |G|)$ time), since extents are given as tuples.

There can be various efficient implementations of storing concepts together with the covering relation on them. For the sake of simplicity we consider here the one, consisting of the (lexicographically ordered) extent list, where from each extent there is a pointer to the corresponding intent and a pointer to the set of concepts covered by the pattern concept with this extent. So, the function $\mathbf{save}(X,\text{covers}(X))$ takes a concept X, the corresponding set $\mathbf{covers}(X)$ and inserts extent of X in the list of extents, making pointers to intent of X and elements of $\mathbf{covers}(X)$ in the list of extents.

The top-down algorithm for graph lattices (TDAGL) traverses the diagram of the lattice in a standard depth-first way. The right margin shows the worst-case complexity of the algorithms steps

tdagl(X):
  mark X as **visited** $O(1)$
  process(X) $O((\alpha + \beta) \cdot \max_i |V_i| \cdot m \cdot n \cdot |G|)$
  for all $Y \in \mathbf{covers}(X)$ and Y not **visited** do
    tdagl(Y)

The procedure **process**(X) for $X = (\mathcal{G}^\diamond, \mathcal{G}^{\diamond\diamond})$ is described as follows:

process(X)
  for all i, j, k
    compute $(+\text{next}(\mathcal{G}, \Gamma_i, v_i^j, a_k))^{\diamond\diamond}$ $O(\alpha \cdot \max_i|V_i| \cdot m \cdot n \cdot |G|)$
    compute $(+\text{next}(\mathcal{G}, \Gamma_i, v_i^j, a_k))^{\diamond}$ $O(\beta \cdot \max_i|V_i| \cdot m \cdot n \cdot |G|)$
  save$(X,$ **covers**$(X))$ $O(|G|)$
  for all $Y \in$ **covers**(X) do
    if extent$(Y) \notin$ extentlist then **insert**(extent(Y), extentlist)
    $O(|G| \cdot |\textbf{covers}(X)|)$.

4.1 Top-Down and Bottom-Up Algorithms

The total worst-case complexity of TDAGL, which constructs the set of all pattern concepts together with the covering relation on them is

$$O((\alpha + \beta) \cdot |\prec| \cdot \max_i|V_i| \cdot m \cdot n \cdot |G|),$$

where $|\prec|$ is the size of the covering relation $\prec$ on pattern concepts (i.e., the number of edges in the lattice diagram). This complexity is similar to that of the bottom-up algorithm.

If we compare computing graph-based hypotheses with a bottom-up algorithm from [20] and TDAGL, we see that the latter one, supplied with an additional command line, can backtrack generating a graph set $\mathcal{G}$ such that $\mathcal{G}^\diamond \subseteq G_+$, thus outputting minimal hypotheses. With TDAGL one can efficiently compute "weaker" hypotheses, based on a relaxation of the definition of a positive hypothesis that allows for a restricted amount of counterexamples: $|\mathcal{G}^\diamond \cap G_-| \leq k$, where k is a parameter.

Another advantage of TDAGL in comparison with the algorithm described in [20] is that the former can be considered as an algorithm for level-wise construction of projections, starting from smallest ones. To be more precise, TDAGL constructs closures (in the sense of $(\cdot)^{\diamond\diamond}$ operator) of projections, which is more useful. Indeed, consider two graph projections p_1 and p_2 that are subgraphs of same graphs from the graph dataset. In PBRL procedure (see Section 2) these two projections would first correspond to different attributes (they will be clarified only on step 4), thus resulting in the above mentioned drastic growth of the number of attributes. However, in TDAGL the two projections p_1 and p_2 will occur together in the same pattern intent. Actually, the kth level of TDAGL output gives closures of kth projections, the number of which is much smaller than that of kth projections. The graphs comprising these closures may be declared attributes of the representation context, thus making the number of attributes smaller than the number of initial attributes in PBRL. Thus, TDAGL allows one to change the first step of PBRL to

1*. For each example e and for k compute k-projections of $\delta(e)$. The graphs from closures of these projections are declared to be attributes.

5 Conclusions

Lattices arising from descriptions other than sets of attributes are well-known in FCA. Mathematically they are reduced to concept lattices by means of the Basic Theorem of FCA. From the computer science perspective these lattices present problems related to computational complexity.

A method for computing lattices with descriptions given by sets of graphs, starting with rough approximations (low-level projections) was proposed. The algorithm starts from minimal subgraphs and proceeds stepwise using lexicographical order on lists of extents. In contrast to the previous algorithm which constructs the lattice of graph sets bottom-up (starting from object descriptions), this approach allows one to stop at an appropriate level of approximation (projection), thus saving much time and space.

Acknowledgments. This work was supported by the joint COMO project of the Deutsche Forschungsgemeinschaft (DFG) and the Russian Foundation for Basic Research (RFBR).

References

1. Avidon, V.V., Pomerantsev, A.B.: Structure-Activity Relationship Oriented Languages for Chemical Structure Representation. J. Chem. Inf. Comput. Sci. 22(4), 207–214 (1982)
2. Borgelt, C., Berthold, M.R.: Mining Molecular Fragments: Finding Relevant Substructures of Molecules. In: Zhong, N., Yu, P.S. (eds.) Proc. 2nd IEEE International Conference on Data Mining, ICDM 2002, pp. 51–58 (2002)
3. Blinova, V.V., Dobrynin, D.A.: Languages for Representing Chemical Compounds for Intelligent Systems of Chemical Design. Automatic Documentation and Mathematical Linguistics 3 (2000)
4. Blinova, V.G., Dobrynin, D.A., Finn, V.K., Kuznetsov, S.O., Pankratova, E.S.: Toxicology analysis by means of the JSM-method. Bioinformatics 19, 1201–1207 (2003)
5. Cook, D.J., Holder, L.B.: Graph-Based Data Mining. IEEE Intelligent Systems 15(2), 32–41 (2000)
6. Finn, V.K.: Plausible Reasoning in Systems of JSM Type. Itogi Nauki i Tekhniki, Seriya Informatika 15, 54–101 (1991) (in Russian)
7. Ganter, B., Grigoriev, P.A., Kuznetsov, S.O., Samokhin, M.V.: Concept-Based Data Mining with Scaled Labeled Graphs. In: Wolff, K.E., Pfeiffer, H.D., Delugach, H.S. (eds.) ICCS 2004. LNCS (LNAI), vol. 3127, pp. 94–108. Springer, Heidelberg (2004)
8. Gonzalez, J.A., Holder, L.B., Cook, D.J.: Application of Graph-Based Concept Learning to the Predictive Toxicology Domain. In: Helma, C., King, R.D., Kramer, S., Srinivasan, A. (eds.) Proc. Workshop on Predictive Toxicology Challegnge at the 5th Conference on Data Mining and Knowledge Discovery, PKDD 2001, September 6 (2001)
9. Gonzalez, J.A., Holder, L.B., Cook, D.J.: Experimental Comparison of Graph-Based Relational Concept Learning with Inductive Logic Programming Systems. In: Matwin, S., Sammut, C. (eds.) ILP 2002. LNCS (LNAI), vol. 2583, pp. 84–100. Springer, Heidelberg (2003)

10. Ganter, B., Kuznetsov, S.: Formalizing Hypotheses with Concepts. In: Ganter, B., Mineau, G.W. (eds.) ICCS 2000. LNCS (LNAI), vol. 1867, pp. 342–356. Springer, Heidelberg (2000)
11. Ganter, B., Kuznetsov, S.O.: Pattern structures and their projections. In: Delugach, H.S., Stumme, G. (eds.) ICCS 2001. LNCS (LNAI), vol. 2120, pp. 129–142. Springer, Heidelberg (2001)
12. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, Heidelberg (1999)
13. Grigoriev, P.A., Yevtushenko, S.A.: Elements of an Agile Discovery Environment. In: Grieser, G., Tanaka, Y., Yamamoto, A. (eds.) DS 2003. LNCS (LNAI), vol. 2843, pp. 311–319. Springer, Heidelberg (2003)
14. Inokuchi, A., Washio, T., Motoda, H.: Complete Mining of Frequent Patterns from Graphs: Mining Graph Data. Machine Learning 50(3), 321–354 (2003)
15. King, R.D., Srinivasan, A., Dehaspe, L.: WARMR: A Data Mining tool for chemical data. J. of Computer-Aided Molecular Design 15(2), 173–181 (2001)
16. Kuznetsov, S.O., Obiedkov, S.A.: Comparing performance of algorithms for generating concept lattices. J. Exp. Theor. Artif. Intell. 14(2-3), 189–216 (2002)
17. Kramer, S.: Structural Regression Trees. In: Proc. 13th National Conference on Artificial Intelligence, AAAI 1996, pp. 812–819. AAAI Press/MIT Press (1996)
18. Kuznetsov, S.O.: Similarity operation on hypergraphs as a basis of plausible inference. In: Proc. 1st Soviet Conference on Artificial Intelligence, vol. 1, pp. 442–448 (1988)
19. Kuznetsov, S.O.: JSM-method as a machine learning method. Itogi Nauki i Tekhniki, ser. Informatika 15, 17–50 (1991) (in Russian)
20. Kuznetsov, S.O.: Learning of Simple Conceptual Graphs from Positive and Negative Examples. In: Żytkow, J.M., Rauch, J. (eds.) PKDD 1999. LNCS (LNAI), vol. 1704, pp. 384–391. Springer, Heidelberg (1999)
21. Liquiere, M., Sallantin, J.: Structural Machine Learning with Galois Lattice and Graphs. In: Shavlik, J.W. (ed.) Proc. 15th International Conference on Machine Learning, ICML 1998, pp. 305–313 (1998)
22. Kuznetsov, S.O., Samokhin, M.V.: Learning Closed Sets of Labeled Graphs for Chemical Applications. In: Kramer, S., Pfahringer, B. (eds.) ILP 2005. LNCS (LNAI), vol. 3625, pp. 190–208. Springer, Heidelberg (2005)
23. Nguyen, P.C., Washio, T., Ohara, K., Motoda, H.: Using a Hash-Based Method for Apriori-Based Graph Mining. In: Boulicaut, J.-F., Esposito, F., Giannotti, F., et al. (eds.) PKDD 2004. LNCS (LNAI), vol. 3202, pp. 349–361. Springer, Heidelberg (2004)
24. Pfahringer, B.: The Futility of Trying to Predict Carcinogenicity of Chemical Compounds. In: Helma, C., King, R.D., Kramer, S., Srinivasan, A. (eds.) Proc. of the Workshop on Predictive Toxicology Challenge at the 5th Conference on Data Mining and Knowledge Discovery, PKDD 2001, September 7 (2001), http://www.predictive-toxicology.org/ptc/
25. Helma, C., King, R.D., Kramer, S., Srinivasan, A. (eds.): Proc. of the Workshop on Predictive Toxicology Challegnge at the 5th Conference on Data Mining and Knowledge Discovery, PKDD 2001, September 7 (2001), http://www.predictive-toxicology.org/ptc/
26. Grigoriev, P.A., Yevtushenko, S.A., Grieser, G.: QuDA, a data miner's discovery environment, FG Intellektik, FB Informatik, Technische Universität Darmstadt, Technical Report AIDA-03-06 (2003), http://www.intellektik.informatik.tu-darmstadt.de/~peter/QuDA.pdf

27. Sebag, M.: Delaying the Choice of Bias: A Disjunctive Version Space Approach. In: Saitta, L. (ed.) Proc. of the 13th International Conference on Machine Learning, pp. 444–452 (1996)
28. Washio, T., Motoda, H.: State of the art of graph-based data mining. SIGKDD Explorations Newsletter 5(1), 59–68 (2003)
29. Yan, X., Han, J.: gSpan: Graph-Based Substructure Pattern Mining. In: Proc. IEEE Int. Conf. on Data Mining, ICDM 2002, pp. 721–724 (2002)
30. Yan, X., Han, J.: CloseGraph: mining closed frequent graph patterns. In: Getoor, L., Senator, T.E., Domingos, P., Faloutsos, C. (eds.) Proc. of the 9th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, KDD 2003, pp. 286–295. ACM Press, New York (2003)
31. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools with Java Implementations. Morgan Kaufmann, San Francisco (2000)
32. Yan, L.-S.: Study of carcinogenic mechanism of polycyclic aromatic hydrocarbons-extended by region theory and its quantitative model. Carcinogenesis 6(1), 1–6 (1985)
33. Woo, Y.-T., et al.: Use of mechanism-based structure-activity relationships analysis in carcinogenic ranking for drinking water desinfection by-products. Environ. Health Perspect. (1), 75–87 (2002)