Expert Systems with Applications 39 (2012) 11172–11182

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Learning from graph data by putting graphs on the lattice

Viet Anh Nguyen *, Akihiro Yamamoto

Graduate School of Informatics, Kyoto University, Kyoto, Japan

ARTICLE INFO

Keywords:
Learning from graph data
Formal Concept Analysis
Graph similarity measure
Graph classification

ABSTRACT

Graph data have been of common practice in many application domains. However, it is very difficult to deal with graphs due to their intrinsic complex structure. In this paper, we propose to apply Formal Concept Analysis (FCA) to learning from graph data. We use subgraphs appearing in each of graph data as its attributes and construct a lattice based on FCA to organize subgraph attributes which are too numerous. For statistical learning purpose, we propose a similarity measure based on the concept lattice, taking into account the lattice structure explicitly. We prove that, the upper part of the lattice can provide a reliable and feasible way to compute the similarity between graphs. We also show that the similarity measure is rich enough to include some other measures as subparts. We apply the measure to a transductive learning algorithm for graph classification to prove its efficiency and effectiveness in practice. The high accuracy and low running time results confirm empirically the merit of the similarity measure based on the lattice.

1. Introduction

In many practical application domains such as drug discovery, image analysis, and network analysis, data objects usually have structures and therefore, are naturally represented as graphs (Aggarwal & Wang, 2000; Felzenszwalb & Huttenlocher, 1998; Nicolaou & Pattichis, 2006; Nowozin, Tsuda, Uno, Kudo, & Bakir, 2007; Wasserman & Faust, 1994). The statistical learning problem on these domains faces a fundamental difficulty of making sense out of structures compared to the usual domains of categorical or numerical spaces. To bridge the gap between structured domain and common learning frameworks, graph data are usually represented using local patterns such as subgraphs, subtrees, walks, paths (Borgwardt & Kriegel, 2005; Gärtner, Flach, & Wrobel, 2003; Ramon & Gärtner, 2003) or represented in a pairwise manner such as similarities or distances (Bunke & Shearer, 1998; Wallis, Shoubridge, Kraetz, & Ray, 2001) directly.

The central problem of all methods for learning in graphs, both the ones that generate patterns explicitly or the ones that measure pairwise relations, boils down to the issue of using local patterns to represent graphs for learning purposes. The problem is that there are exponentially many such patterns (Amini, Fomin, & Saurabh, 2009; Charikar, 2000). This results in two problems: finding meaningful patterns from data directly, and finding them in a computationally feasible way. The good news is, however, these patterns, by themselves, follow a rich relationship. Once these relationships are constructed from the given data, it is up to the learning systems to exploit them for the purpose of finding meaningful and non-redundant patterns while avoiding the exponential time complexity on large data.

In this work, we formalize the organization of local patterns of subgraphs with Formal Concept Analysis (Davey & Priestley, 2002; Ganter & Wille, 1998) and prove that subgraph patterns can also be put into a lattice structure in the same manner as itemset patterns (Hereth, Stumme, Wille, & Wille, 2000; Poelmans, Elzinga, Viaene, & Dedene, 2010). The construction of the lattice is based on the definition of a pair of mappings, called a Galois connection, between a set of graph objects and a set of graph attributes. We show that the construction of the lattice can be facilitated by making use of the results of efficient graph mining algorithms. The lattice structure, once built, could facilitate pattern understanding, knowledge discovery, and interactive exploration of complex graph data.

For the purpose of statistical learning, we provide a similarity measure that utilizes the hierarchical structure of concepts in the lattice. Notions of similarity appear to be particularly suitable for structured data as they do not require direct access to the features of data, and thus the data can be of any type, as long as the similarity is well defined. If there is a reliable graph similarity measure that can be computed efficiently, this would be an important contribution to the task of learning from graph data.

The central problem of computational cost for generating all possible subgraphs is often solved by limiting to only statistically significant patterns such as frequent ones. This is also the case of our similarity measure as it has to be computed on the iceberg lattice of frequent concepts. We show formally that on average, the

* Corresponding author.
E-mail addresses: vietanh@iip.ist.i.kyoto-u.ac.jp (V.A. Nguyen), akihiro@i.kyoto-u.ac.jp (A. Yamamoto).

0957-4174/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2012.03.035

similarity measure defined on the iceberg lattice is lower-bounded linearly with the minimum supports. This means that we can efficiently approximate the true similarity between graphs by setting an appropriate value of the minimum support.

We also show that the well-known graph edit distance measure (Bruno, Messmer, & Bunke, 1998; Sanfeliu & Fu, 1983) and the maximum common subgraph based distance measures (Bunke & Shearer, 1998; Wallis et al., 2001) could be computed by using a component of the lattice. However, by using the properties of the lattice, we show that these measures, while being computationally expensive, might not contain all necessary information for the task of learning from graph data.

To empirically verify the usefulness of the proposed similarity measure, we apply the measure to the problem of graph classification using the k nearest neighbor classifier. For large datasets, we propose to use the transductive setting to avoid the out-of-sample similarity computations that require an exponential time complexity for a subgraph isomorphism. In experiments, our method outperforms other state-of-the-art algorithms with large margin on various large real-world datasets. The empirical advantages confirm that the proposed similarity measure captures well the domain knowledge from graph data. We also observe that the prediction performance does not increase after minimum supports are lowered to certain extents, justifying the usefulness of using only the iceberg lattice for similarity measurement.

We give an outline of the paper as follows. In Section 2, we introduce formal concept lattices using subgraph patterns as attributes based on Formal Concept Analysis. In Section 3, we propose a similarity measure which utilize the structure of the lattice and give an application of using this measure in graph classification. In Section 4, we describe our experiments on various real-world datasets to verify the usefulness of the proposed measure. Finally, we summarize the paper and provide directions for future work in Section 5.

2. An FCA-based lattice for graph data

In the past few years, the problem of learning from complex graph data has been studied extensively under the name of subgraph mining. The main strategy in existing work is first finding frequent subgraphs that occur in a sufficient number of input objects and then constructing structural rules or making implications based on discovered frequent subsgraphs. Soon a new problem is how to reduce the huge number of possible structural rules extracted by the learning algorithms. This problem requires thorough investigation of relationship and correlation between extracted rules.

Naturally, the subgraph–supergraph containment relation between subgraphs provides a good tool for doing so. However, this containment relation does not represent all the particularities hidden in the graph data. Generally, there can exist two subgraphs such that they occur in the same set of input objects, but there is no subgraph–supergraph relation between them. None of the two such subgraphs can be considered better than the other, and therefore we need to consider both of them. A promising solution to deal with this situation can be found from the field of Formal Concept Analysis (FCA) as FCA offers a compact mode of knowledge representation called attribute implications which are closely associated with functional dependencies, and thus, provides ways to reason about implications implicit in the graph data.

The theory of FCA is based on a *Galois connection* that is a pair of mappings between a set of objects and a set of attributes. The Galois connection enables a closure system, i.e., all objects within a *formal concept* share the same maximal set of attributes and vice versa. Formal concepts with a partial order among them form a lattice structure called *formal concept lattice*. FCA theory has been studied and applied successfully in many diverse fields including knowledge formalization and acquisition, ontology design, information retrieval, social networking, software engineering, and data mining (Hereth et al., 2000; Priss, 2007; Snásel, Horák, & Abraham, 2008; Valtchev, Missaoui, & Godin, 2004; Wille, 2002; Zaki & Ogihara, 1998). In this paper we show that FCA theory can be applied to graph data as well. We start by introducing the basics of FCA. Readers who want more detail about FCA are referred to, for example, (Davey & Priestley, 2002; Ganter & Wille, 1998).

2.1. Preliminary from Formal Concept Analysis

Given two sets $\mathcal{O}$ and $\mathcal{A}$, and let $\mathcal{I}$ be a relation between them, i.e., $\mathcal{I} \subseteq \mathcal{O} \times \mathcal{A}$. Two operators (mappings) $\alpha : 2^{\mathcal{O}} \to 2^{\mathcal{A}}$ and $\omega : 2^{\mathcal{A}} \to 2^{\mathcal{O}}$ are defined as

$$\alpha(O) = \{a \in \mathcal{A} | (o,a) \in \mathcal{I} \text{ for all } o \in O\}, \quad \text{for } O \subseteq \mathcal{O},$$
$$\omega(A) = \{o \in \mathcal{O} | (o,a) \in \mathcal{I} \text{ for all } a \in A\}, \quad \text{for } A \subseteq \mathcal{A}.$$

Then the pair (O,A) that satisfies $\alpha(O) = A$ and $O = \omega(A)$ is called the *formal concept* of the *formal context* $(\mathcal{O}, \mathcal{A}, \mathcal{I})$. All formal concepts form a *concept lattice* of $(\mathcal{O}, \mathcal{A}, \mathcal{I})$ when they are ordered by

$$(O,A) \leqslant (O',A') \iff O \subseteq O' (\text{equivalent to } A' \subseteq A).$$

In FCA theory, elements of $\mathcal{O}$ are viewed as "objects", those of $\mathcal{A}$ as "attributes", and $(o,a) \in \mathcal{I}$ means that the object o verifies the attribute a. For a concept (O,A), O is called the *extent* and A is called the *intent* of the concept. If any pair of concepts have one upper bound, the *Join* element, and one lower bound, the *Meet* element, then the concept lattice is said complete. The Join ($\vee$) and the Meet ($\wedge$) operators are defined as

$$(O,A) \vee (O',A') = (\omega \circ \alpha(O \cup O'), A \cap A'),$$
$$(O,A) \wedge (O',A') = (O \cap O', \alpha \circ \omega(A \cup A')).$$

In the next subsections, we deal with the formalization of a lattice for graph data, which is not a trivial task.

2.2. The concept lattice for graph patterns

In this paper, we deal with labeled connected graphs. An unlabeled graph can be viewed as a labeled graph if we consider each vertex and edge of the graph has the same label.

Definition 1. A labeled graph is represented as a 4-tuple $G = (V,E,L,l)$, where V is a set of vertices (nodes), $E \subseteq V \times V$ is a set of edges, L is a set of labels, and l is a labeling function that assigns labels to the vertices and edges. A labeled graph is connected if there is a path between any pair of vertices.

A graph g is subgraph of another graph g' if there exists a subgraph isomorphism from g to g'.

Definition 2. A subgraph isomorphism is an injective function $f : V(g) \to V(g')$ such that (1) $\forall v \in V(g)$, $l(v) = l'(f(v))$, and (2) $\forall (u,v) \in E(g)$, $(f(u),f(v)) \in E(g')$ and $l(u,v) = l'(f(u),f(v))$ where l and l' are the labeling function of g and g', respectively.

We denote by $g \preceq g'$ if the graph g is a subgraph of the graph g'. If $g \preceq g'$, we also say that g' contains g or g' is a supergraph of g. If g is a proper subgraph of g', we write $g \prec g'$.

We regard a graph instance as an object, and let $\mathcal{O}$ be the set of all graph objects. Since graph data have no well defined attributes, we treat subgraphs contained in graphs as attributes. Let $\mathcal{A}$ be the set of all subgraphs treated as attributes. To formalize the lattice of subgraph patterns, we have to re-define the operators α and ω, the subsumption order between two sets of subgraphs, and the

intersection (required for the Join operator) of sets of subgraphs. Note that the Meet operator does not require any further modification.

The operator $\alpha(O)$ maps a set of graph objects O into a set of attributes which are common subgraphs of all objects in O. Here we restrict to the *maximal common subgraphs* only, by this way, sub-subgraphs are tacitly included. The operator ω, defined on a set of subgraph attributes A, returns the set of graph objects that contain all the subgraph attributes in A.

Definition 3. Let S be a set of graphs. A common subgraph g of graphs in S is maximal if $g \not\prec g'$ for any common subgraph g' of graphs in S.

In Fig. 1, a set of two graphs g_1 and g_2 and their common subgraphs of are presented. Although there are eleven common subgraphs of g_1 and g_2, only two of them are maximal.

Definition 4. The operators α and ω for graph data are defined as

$$\alpha(O) = \{a \in \mathcal{A} | a \text{ is a maximal common subgraph of all objects in } O\},$$
$$\omega(A) = \{o \in \mathcal{O} | a \preceq o, \text{for all subgraph attribute } a \in A\}.$$

The new definitions of operators α and ω in Definition 4 will be used in the remains of the paper. A pair (O,A) satisfying $\alpha(O) = A$ and $O = \omega(A)$ is a formal concept of graph patterns. In Fig. 2, a set of three graphs g_1, g_2, and g_3 are given. The first pair of graph objects and subgraph attributes is a formal concept while the second pair is not a formal concept according to the definitions of operators α and ω.

Definition 5. Let S and S' be two sets of subgraphs. S is *subsumed* by S', denoted by $S \models S'$, iff for any $s \in S$, there exists $s' \in S'$ such that $s \preceq s'$.

It follows that if S is subsumed by S' and S' is subsumed by S then $S = S'$.

Proposition 1. *Let O, O' be sets of graph objects, and let A, A' be sets of graph attributes, then*

1. $O \subseteq O' \Rightarrow \alpha(O') \models \alpha(O)$,
2. $A \models A' \Rightarrow \omega(A') \subseteq \omega(A)$,
3. $O \subseteq \omega \circ \alpha(O)$,
4. $A \models \alpha \circ \omega(A)$.

Proof

(1) If $a \in \alpha(O')$, then a is a maximal common subgraph of all objects in O'. Because $O \subseteq O'$, we have a is a maximal common subgraph of all objects in O and thus $a \in \alpha(O)$.

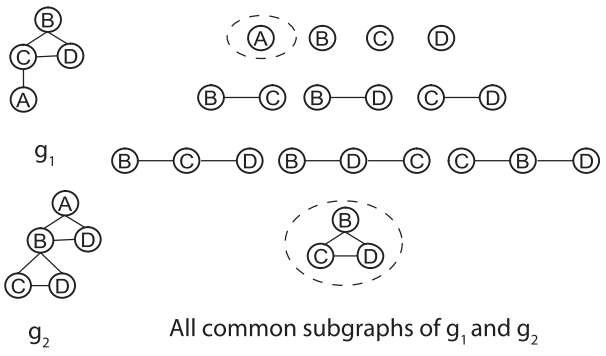

Fig. 1. An example of common subgraphs and maximal common subgraphs. Common subgraphs in dashed circles are maximal.

(2) If $o \in \omega(A')$, then for all $a \in A'$, $a \preceq o$. Because $A \models A'$, we have $a \preceq o$ for all $a \in A$, and thus $o \in \omega(A)$.
(3) If $o \in O$, then for all $a \in \alpha(O)$, $a \preceq o$, which implies $o \in \omega \circ \alpha(O)$.
(4) If $a \in A$, then $a \preceq o$ for all $o \in \omega(A)$, which implies $a \in \alpha \circ \omega(A)$. □

The pair (α,ω) of operators, $\alpha : 2^{\mathcal{O}} \to 2^{\mathcal{A}}$ and $\omega : 2^{\mathcal{A}} \to 2^{\mathcal{O}}$, that fulfils conditions 1, 2, 3, and 4 of Proposition 1 forms a Galois connection between $2^{\mathcal{O}}$ and $2^{\mathcal{A}}$.

Proposition 2. *It follows that $\alpha = \alpha \circ \omega \circ \alpha$ and $\omega = \omega \circ \alpha \circ \omega$.*

Proof. With $A = \alpha(O)$ we have $\alpha(O) \models \alpha \circ \omega \circ \alpha(O)$ by Proposition 1, property 4. We also have $O \subseteq \omega \circ \alpha(O)$ (Proposition 1, property 3), and by Proposition 1, property 1, $\alpha \circ \omega \circ \alpha(O) \models \alpha(O)$. □

Proposition 3. *The composite operator $\alpha \circ \omega$ is a closure operator on $2^{\mathcal{A}}$ and the composite operator $\omega \circ \alpha$ is a closure operator on $2^{\mathcal{O}}$.*

Proof. An operator ϕ on a set G is a closure operator iff the following conditions satisfied for $X, Y \subseteq G$

1. $X \subseteq Y \Rightarrow \phi(X) \subseteq \phi(Y)$ (*monotony*).
2. $X \subseteq \phi(X)$ (*extensity*).
3. $\phi(\phi(X)) = \phi(X)$ (*idempotency*).

Monotony and extensity of the composite operators follow immediately from the definition of a Galois connection, and idempotency follows from Proposition 2. □

We are now ready to define the order between two concepts, the intersection of intents of concepts, and the Join operator. The order between two concepts is given by

$$(O,A) \leqslant (O',A') \Longleftrightarrow O \subseteq O' \quad (\text{equivalent to } A' \models A).$$

This order allows to construct the concept lattice of the (partially) ordered concepts. If $(O,A) \leqslant (O',A')$ are two concepts of the concept lattice, (O,A) is called a *sub-concept* of (O',A'), and (O',A') is called a *super-concept* of (O,A).

In order to define the join of two concepts, we give below a definition of the intersection of two sets of subgraphs.

Definition 6. Let s_1 and s_2 be two subgraphs. The intersection of s_1 and s_2, denoted as $s_1 \sqcap s_2$, is the set of all maximal common subgraphs of s_1 and s_2. Let S_1 and S_2 be two sets of subgraphs, and let S denote the set $S = \bigcup_{s_1 \in S_1, s_2 \in S_2} s_1 \sqcap s_2$. The intersection of S_1 and S_2, denoted as $S_1 \sqcap S_2$, is defined as

$$S_1 \sqcap S_2 = \{s | s \in S, \text{ and there does not exist } s' \in S \text{ such that } s \prec s'\}.$$

Given a formal concept (O,A), because $\alpha(O)$ contains all common maximal subgraphs of all objects in O, $\alpha(O)$ is actually the intersection of all graph objects in O.

By having the definition of the intersection of two sets of subgraphs, the Join operator is re-defined as

$$(O,A) \vee (O',A') = (\omega \circ \alpha(O \cup O'), A \sqcap A').$$

We give an example of the Join operator in Fig. 3. As can be seen in the figure, the intent of the concept as the result of the Join operator is subsumed by the intent of both (O,A) and (O',A').

Proposition 4. *The join of two formal concepts by the Join operator $\vee$ is again a formal concept. Similarly, the meet of two formal concepts by the Meet operator $\wedge$ is again a formal concept.*

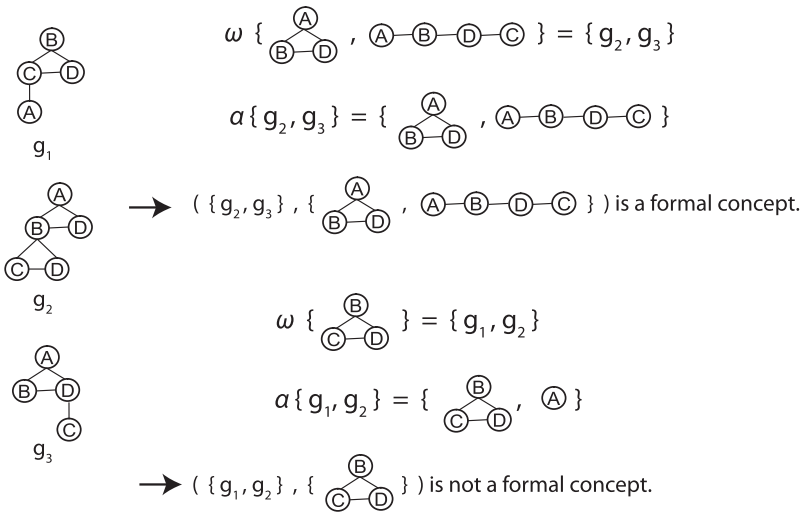

Fig. 2. An example of operators α and ω.

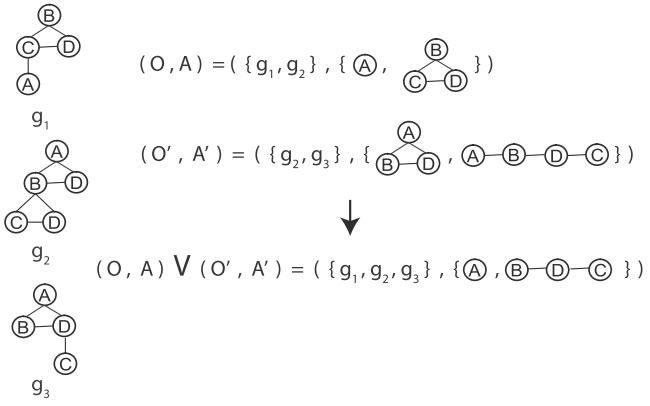

Fig. 3. An example of the Join operator.

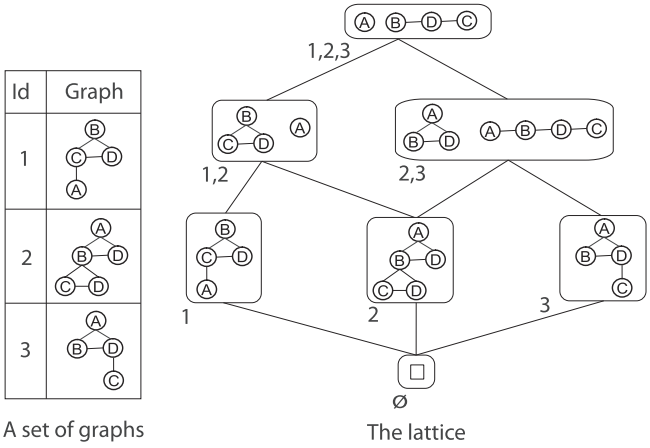

Fig. 4. A set of graphs and the corresponding lattice.

Proof. We give the proof for the join case. We need to prove that $\omega(A \sqcap A') = \omega \circ \alpha(O \cup O')$ and $\alpha \circ \omega \circ \alpha(O \cup O') = A \sqcap A'$.

If $a_1 \in A$ and $a_2 \in A'$ then a_1 is a common subgraph of all objects in O and a_2 is a common subgraph of all objects in O'. If $a \in a_1 \sqcap a_2$ then a is a common subgraph of all objects in $O \cup O'$. Therefore, $A \sqcap A'$ consists of all maximal common subgraphs of $O \cup O'$ according to Definition 6. By the definition of the α operator, we have $A \sqcap A' = \alpha(O \cup O')$. Applying the ω operator, we have $\omega(A \sqcap A') = \omega \circ \alpha(O \cup O')$.

Because $\alpha \circ \omega \circ \alpha = \alpha$ (Proposition 2), we have $\alpha \circ \omega \circ \alpha(O \cup O') = \alpha(O \cup O') = A \sqcap A'$. □

Let L be the set of all formal concepts of graph patterns. The 4-tuple $(L, \leqslant, \vee, \wedge)$ forms a complete formal concept lattice. If $x \leqslant y$ for x and y are two concepts then y is called an *ancestor* of x in the lattice. If z is a common ancestor of concepts x and y, and z is not an ancestor of another common ancestor of x and y, then z is called the *lowest common ancestor* of x and y. The top concept of the lattice (the *root*) is the common ancestor of all concepts of the lattice. The extent of the root concept contains all input graphs. The bottom concept of the lattice has the intent that subsumes the intent of every concept of the lattice. In case this intent is not found from any graph object, it is denoted by the □ symbol. The extent of the bottom concept in this case is the empty set.

Fig. 4 shows a set of three graph objects and the corresponding complete lattice. Subgraphs in the rounded rectangles are intents of the concepts. The extents are represented as sets of graph ids.

2.3. Correspondence between formal concepts and closed subgraphs

The construction of the complete lattice of graph patterns is infeasible in general since even testing the subgraph isomorphism and computing the intersection of graphs are NP-hard for labeled graphs. In this subsection we provide some useful connections between formal concepts and *closed subgraphs*. These connections provide a feasible way to build the upper part, called the *iceberg lattice*, of the lattice.

The problem of subgraph mining have been studied intensively in the data mining community, take for examples (Horvath, Ramon, & Wrobel, 2006; Inokuchi, Washio, & Motoda, 2000; Kuramochi & Karypis, 2001; Yan & Han, 2002). Closed subgraphs are useful lossless condensed representations of all subgraphs and have interesting properties for pruning, thus they can help to reduce the number of patterns generated and increase the mining efficiency. A closed subgraph is defined as follows.

Definition 7. Given a dataset of graphs G, a subgraph s is closed if none of proper supergraphs of s has the same support that s has, where the support of s is the number of graph objects in G that contain s.

According to this definition, closed subgraphs are maximal ones in the set of graph objects they are contained in.

In a formal concept lattice of itemsets, each closed itemset corresponds with a formal concept, that is, it is the intent of the corresponding concept. The intent of a formal concept of graph data is, however, not a single subgraph, but a set of subgraphs. To avoid ambiguity, we want to note here that a closed subgraph is closed under the addition of edges while the extent and the intent of a formal concept of graph data are closed sets which are closed under intersections.

Theorem 1. *Let $(\mathcal{O}, \mathcal{A}, \mathcal{I})$ be a context of graph objects and subgraph attributes, let L be the set of all formal concepts of the context, and let C be the set of all closed subgraphs mined from $\mathcal{O}$. The following statements hold:*

1. For a formal concept $(O,A) \in L$, a subgraph $a \in A$ implies $a \in C$.
2. If c is a closed subgraph, that is $c \in C$, then there exists $(O,A) \in L$ such that $c \in A$.

Proof

1. Let $G(a)$ be the set of graph objects that contain the subgraph a. Either $G(a) = O$ or $G(a) \supset O$ must hold. If $G(a) = O$ then obviously a is closed by the definition of the operator α. We prove that a is closed by contradiction in the case $G(a) \supset O$. Suppose that a is not closed then there must exist a subgraph a' s.t. $a \prec a'$ and $G(a) = G(a')$. This means that a is not maximal in $G(a)$ and thus not maximal in O. By the definition of the operator α we have $a \notin A$ which contradicts with $a \in A$.
2. Again, let $G(c)$ be the set of graph objects that contain the closed subgraph c. Because c is closed, we have $c \in \alpha(G(c))$. It follows that $(G(c), \alpha(G(c)))$ is a formal concept whose the intent containing c. □

Theorem 1 has an important application in the construction of the concept lattice because it implies that we can make use of the results of closed subgraph mining for the construction task.

Proposition 5. *Let L be the set of all formal concepts of the context $(\mathcal{O}, \mathcal{A}, \mathcal{I})$. Any graph object $g \in \mathcal{O}$ is the intent of a formal concept of L.*

Proof. Apply the operator ω on g and let $G(g) = \omega(\{g\})$. We have that $G(g)$ is the set of all graph objects, including g, that take g as a subgraph. By the definition of the operator α, we have $\alpha(G(g)) = \{g\}$. Thus, $(G(g),\{g\})$ is a formal concept which takes g as its intent. □

Proposition 5 suggests a way to estimate the similarity between two graphs by measuring the similarity between two formal concepts in the concept lattice. A formal concept whose the intent is a graph object often lies near the bottom of the lattice. Proposition 5 also implies that every graph object is also a closed subgraph.

2.4. Lattice construction

The construction of a lattice has two steps: finding formal concepts and making links among them. The construction of the whole lattice is often infeasible since it requires the complete set of closed graphs. Therefore, it is only feasible to build the upper part of the lattice from the set of *closed frequent subgraphs*. A closed subgraph is frequent if its support is not less than a given threshold *minsup*. Similarly, we say a concept (O,A) is frequent if all closed subgraphs in A are frequent, or $|O| \geqslant minsup$. All frequent concepts form the iceberg lattice. Next, we discuss how to find formal concepts from the set of mined closed subgraphs.

We denote by $G(s)$ the set of graphs containing a subgraph s. For a formal concept (O,A), eventhough $\omega(A) = O$, it does not ensure that $G(a) = O$ for all $a \in A$. In general, there could exist a subgraph $a \in A$ such that $G(a) \supset O$. Such a member of A, called a *joint-member*, can also be member of the intent of other concepts in the lattice. Given a concept (O,A) with a is a joint-member of A. Let (O',A') be the formal concept where $O' = G(a)$ and $A' = \alpha(G(a))$. Since $O' \supseteq O$ so (O',A') is an ancestor of (O,A). According to the $\models$ relationship defined in Definition 5, a must appear in all concepts on the path from (O,A) to (O',A'). This means that a can be found in an immediate ancestor of (O,A). The set of all such joint-members of A, thus, can be computed as

$$A_{joint} = \bigcup \{a | a \in A^i, \{a\} \not\models A\} \quad (1)$$

where A^i is the intent of an immediate ancestor of (O,A).

There may exist a concept where all members of the intent are joint-members (see Fig. 5). We call such a concept an *agent concept*. Finding all agent concepts could be expensive because in the worst case, it may require to check an exponential number of combinations of closed subgraphs. For this reason, we do not include agent concepts in building the lattice. We believe that the number of agent concepts is insignificant. Although this assumption has not been verified theoretically, in all of our experiments on real-life datasets, we achieve very good classification results with the lattice without agent concepts.

By the discussions above, we can proceed to build formal concepts from the mined closed subgraphs as follows. First, closed subgraphs of same idlist (the list of input graphs that contain these subgraphs) are grouped together to find concepts with the intent may not be complete. After the structure of the lattice is found (step 2), we will update the intent of these concepts using Eq. 1. Because joint-members of the intent of a concept can be found from the immediate ancestors of the concept, to update the intents we can start from the top-most concept and traverse down the lattice in the breadth-first manner.

The second step of lattice construction is to make the links (subconcept-superconcept relations) of the formal concepts explicit. The link between two concepts (O,A) and (O',A') can be determined either by checking the subset relation $O \subseteq O'$ of extents of concepts, or by checking the subsumption relation $A' \models A$ of intents of concepts. A naive algorithm that considers all pair of concepts to find the parent–child relations has time complexity proportional to $|FC|^2$, where FC is the set of all formal frequent concepts. We provide below an efficient method for this task by using subset checking of extents.

Procedure *MakeLinks*(*FC*, *minsup*)
Input: a set of frequent concepts *FC* sorted in ascending order of extent size, a *minsup*.
Output: the lattice with links among concepts are made explicit.
1: *Candidate* $\leftarrow \emptyset$;
2: **for** each concept $c \in FC$ **do**
3: **Compute** the intersections of the extent of c with the members of *Candidate*, and let *Extents* be the set of maximal intersections whose size is not less than *minsup*;
4: **for** each extent $e \in Extents$ **do**
5: **Find** the concept c_e in *FC* such that $extent(c_e) = e$;
6: **Set** c_e as the child node of c;
7: **Remove** e from *Candidate* if $e \in Candidate$;
8: **Add** the extent of c into *Candidate*;

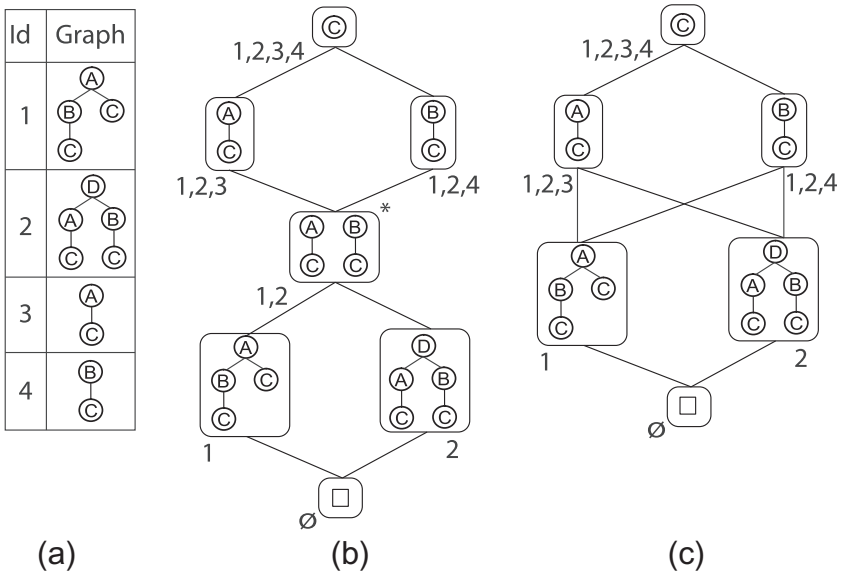

Fig. 5. (a) A graph dataset. (b) The complete lattice. All members of the intent of the agent concept marked by * are joint-members. (c) The reduced lattice after removing the agent concept.

We sort frequent concepts in ascending order of extent size and make links among concepts in a bottom-up manner. Given a concept c, a concept c' is a sub-concept (an immediate predecessor) of c, and thus there is a link between c and c' if $extent(c') \subset extent(c)$ and there does not exist a concept c'' s.t. $extent(c') \subset extent(c'') \subset extent(c)$. By this reason, we do not need to check c against all concepts in FC but only those that are sorted before c in FC. Next, suppose that a link has been made between two concepts c' and c, $c' \leqslant c$. If we are now considering a concept cc that is sorted right after c in FC, then we do not need to check cc against c' because the intersection of cc and c' is included in the intersection of cc and c. Given a concept c, we call the set of concepts that will be checked against c the *Candidate* set of c. We give by the procedure MakeLinks the pseudo-code for the task of making links among formal concepts explicit. By using the *Candidate* set, the time complexity becomes proportional to $y \times |FC|$ where y is the mean size of the *Candidate* set.

Finally, we give the pseudo-code for constructing the iceberg lattice from the set of closed frequent subgraphs in Algorithm 1.

The built formal concept lattice can be used for multiple learning tasks. It can be used in dimensionality reduction as in Bartl, Rezankova, and Sobisek (2011). It can be used in detecting functional dependency as in Ishida, Honda, and Kawahara (2008). It can also be used directly in classification as in Coustaty, Bertet, Visani, and Ogier (2011) and in clustering as in Gao and Lai (2010). In this paper, we would like to exploit the lattice in a different way, that is, to use the structure of lattice to define a reliable similarity measure between graphs.

Algorithm 1: Iceberg Lattice Construction

Input: C – the set of all closed frequent subgraphs, *minsup* - a minimum threshold
Output: the concept lattice
1: Group all closed frequent subgraphs of same idlist to make concepts
2: Make subconcept-superconcept relations using the MakeLinks procedure
3: Traverse the lattice, breadth first, starting from the top concept, and update the intent of concepts using Eq. 1
4: Output the lattice

3. A lattice-based similarity measure for graphs

In this section we propose a similarity measure that takes advantage of the lattice structure of subgraph patterns for learning purpose. We show formally that the similarity measure based on the iceberg lattice is lower-bounded linearly with the minimum supports. We also show that the measures based on the maximum common subgraph can be regarded as subpart of our similarity measure.

3.1. The similarity measure

Given two graphs g_1 and g_2, and let c_1 and c_2 be two formal concepts whose the intents are respectively $\{g_1\}$ and $\{g_2\}$. Using the lattice, it is natural to represent the commonalities between g_1 and g_2 as the set of lowest common ancestors of c_1 and c_2, denoted as $lca(g_1,g_2)$. Two graphs are considered more similar if they share more commonalities and are less similar if they have more differences. Thus, based on that, we define the similarity between two graphs as the ratio of their common information divided by the total amount of information of the two graphs

$$S(g_1,g_2) = \frac{I(lca(g_1,g_2))}{I(description(g_1,g_2))} \quad (2)$$

where $description(g_1,g_2)$ is a proposition that describes g_1 and g_2, $I(s)$ is the amount of information contained in s.

One way of measuring the amount of information contained in a concept is to compute the total size of subgraphs contained in the intent. The approach based on the size of subgraphs is actually adopted by methods in Bunke and Shearer (1998) and Wallis et al. (2001). However, this approach, while requiring to know the structure of subgraphs explicitly, does not take into account the structure of the lattice which is potentially useful in characterizing the similarity between graphs.

Fig. 6 shows a part of a lattice containing the ancestors of three graphs g_1, g_2 and g_3. Assume that g_1, g_2 and g_3 are of the same size. Assume further that the total size of $lca(g_1,g_2)$ equals that of $lca(g_2,g_3)$. The measure of information content based on the size of subgraphs will state that $S(g_1,g_2) = S(g_2,g_3)$. However, if we consider each ancestor of a concept as a feature or a function that

characterizes the concept, then we can say that $lca(g_2,g_3)$ contains more information than $lca(g_1,g_2)$ does, or g_2 is more similar to g_3 than it is to g_1. In connection with graph size, we could say that a concept characterized by many features has a high possibility to have large intent size. On the contrary, a large concept but having too few features may contain lots of noise. Based on this observation, we propose to measure the information contained in a concept by considering its ancestors in the lattice.

Definition 8. Let s be a set of formal concepts. The set of all ancestors of concepts in s, together with the links between them form a substructure of the lattice. We call this substructure the *trace* of s in the lattice.

The trace of any set of formal concepts always shares a part of the hierarchical structure of the lattice starting from the root concept. Every concept in the lattice is assigned with a level, and the level of the root concept is 0 (Fig. 7). Using the notion of traces, the similarity measure between two graphs g_1, g_2 is now given by

$$S(g_1,g_2)=\frac{I(t_{12})}{I(T_{12})} \tag{3}$$

where t_{12} is the trace of the $lca(g_1,g_2)$, and T_{12} is the trace of $\{c_1,c_2\}$ where c_1, c_2 are concepts whose intents are $\{g_1\}$, $\{g_2\}$, respectively.

Let the lattice be assigned with a function $p(c):L\to[0,1]$, where p is the probability of encountering an instance of the concept c in the lattice. According to the Shannon's information theory, the information content of a concept c is given by $-\log p(c)$. We assume that all concepts on the same level of the lattice have same information content. We assign to a concept on the level i of the lattice a probability $p_i=\frac{w_i}{N}$, where N is the number of concepts of the lattice, and w_i is a weight satisfying $w_i\geqslant w_j$ for $i\geqslant j$, and $\sum w_iN_i=N$ where N_i is the number of concepts on the level i of the lattice. The definition of concept probability implies that the information content is monotonically non-decreasing as one moves down the lattice. The total amount of information of a trace t is given by the sum

$$I(t)=\sum_{c\in t}I(c)=-\sum_{i}^{l(t)}n_i(t)\log\frac{w_i}{N} \tag{4}$$

where $n_i(t)$ is the number of concepts on the level i of the trace t and $l(t)$ is the number of levels of t. Plugging Eq. 4 into Eq. 3, we obtain

$$S(g_1,g_2)=\frac{\sum_i^{l(t_{12})}n_i(t_{12})\log\frac{w_i}{N}}{\sum_i^{l(T_{12})}n_i(T_{12})\log\frac{w_i}{N}} \tag{5}$$

If we consider that all concepts of the lattice have the same information content, then $w_i=1$ for all levels of the lattice. The similarity measure in this case is given by

$$S(g_1,g_2)=\frac{\sum_i^{l(t_{12})}n_i(t_{12})}{\sum_i^{l(T_{12})}n_i(T_{12})}=\frac{|t_{12}|}{|T_{12}|} \tag{6}$$

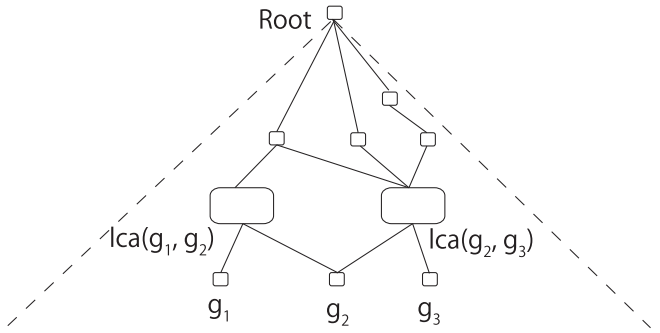

Fig. 6. An example of comparing graphs by using their lowest common ancestors.

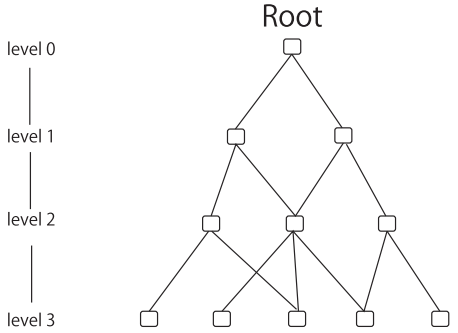

Fig. 7. The hierarchical structure of the lattice.

where $|t|$ is number of concepts in a trace t. Because t_{12} comprises of ancestors of concepts in $lca(g_1,g_2)$, $|t_{12}|$ can be computed by counting the concepts whose extent containing both g_1 and g_2. Similarly, $|T_{12}|$ is computed by counting concepts whose extent containing at least one of the two graphs g_1 and g_2.

It is noteworthy that, the similarity measure given in Eq. 5 requires the explicit structure of the lattice, that is, we must know the subconcept–superconcept relations. The measure given in Eq. 6 is based on the assumption that all concepts are equal, thus, we do not need to know the lattice structure explicitly. Moreover, both equations do not require to know the intent of concepts.

3.2. Approximating the similarity measure using the iceberg lattice

In graph mining, it is often infeasible to generate all subgraphs from a graph database of even modest size. To deal with this computation problem, a minimum support threshold is used to restrict the subgraphs generated to only frequent ones. This setting, however, results in the lost of low frequency closed subgraphs which constitute the lower part of the graph concept lattice. Since the lowest common ancestors of two graphs often locate near the bottom of the lattice, they are very likely to be lost. The similarity measures that are based on the computation of the size of these lowest common ancestors are, therefore, become impossible in this setting. Fortunately, the upper parts of the traces of these lowest common ancestors are preserved in the upper part of the lattice (the *iceberg lattice*), this means that our similarity measure based on traces can be estimated on the lattice of frequent concepts.

In this subsection, we show formally that, on average, the lower bound of the similarity measure given in Eq. 6 when computed over the iceberg lattice is linear to the minimum supports.

Given a minimum support *minsup*, and let g be any graph. The average similarity between g and the set of graphs $\mathcal{O}$ is given by $\frac{1}{|\mathcal{O}|}\sum_{h\in\mathcal{O}}S(g,h)$. The average similarity between g and $\mathcal{O}$ computed on the complete lattice is denoted as S_A, and the average similarity computed on the iceberg lattice is denoted as $\widehat{S}_A$.

Theorem 2. *It holds that* $\widehat{S}_A\geqslant S_A-\frac{minsup-1}{|\mathcal{O}|}$.

Proof. Let t_g be the trace of g in the lattice. We denote by t_g^* the trace of g in the iceberg lattice and by $t_g\setminus t_g^*$ the part of t_g that belongs to the lower part of the lattice. This means that $t_g\setminus t_g^*$ contains concepts of support less than *minsup*. Denote by $\mathcal{T}$ the set of the traces of all graphs in the lattice, we have

$$\begin{aligned}S_A&=\frac{1}{|\mathcal{O}|}\sum_{h\in\mathcal{O}}\frac{|t_{gh}|}{|T_{gh}|}=\frac{1}{|\mathcal{O}|}\sum_{c\in t_g}\sum_{t_h\in\mathcal{T}|c\in t_h}\frac{1}{|T_{gh}|}\\&=\frac{1}{|\mathcal{O}|}\sum_{c\in t_g^*}\sum_{t_h\in\mathcal{T}|c\in t_h}\frac{1}{|T_{gh}|}+\frac{1}{|\mathcal{O}|}\sum_{c\in t_g\setminus t_g^*}\sum_{t_h\in\mathcal{T}|c\in t_h}\frac{1}{|T_{gh}|}.\end{aligned}$$

Denote by $\mathcal{T}^*$ the set of the traces after removing all infrequent concepts from $\mathcal{T}$. The first term is bounded as

$$\frac{1}{|\mathcal{O}|}\sum_{c\in t_g^* t_h\in\mathcal{T}|c\in t_h}\frac{1}{|T_{gh}|}\leqslant\frac{1}{|\mathcal{O}|}\sum_{c\in t_g^* t_h\in\mathcal{T}|c\in t_h}\frac{1}{|T_{gh}^*|}=\frac{1}{|\mathcal{O}|}\sum_{c\in t_g^* t_h\in\mathcal{T}^*|c\in t_h}\frac{1}{|T_{gh}^*|}=\widehat{S}_A.$$

If c is an infrequent concept then the number of traces in $\mathcal{T}$ that contain c is at most $minsup-1$. The second term can be bounded as

$$\frac{1}{|\mathcal{O}|}\sum_{c\in t_g\setminus t_g^* t_h\in\mathcal{T}|c\in t_h}\frac{1}{|T_{gh}|}\leqslant\frac{1}{|\mathcal{O}|}\sum_{c\in t_g\setminus t_g^* t_h\in\mathcal{T}|c\in t_h}\frac{1}{|t_g|}\leqslant\frac{1}{|\mathcal{O}|}\sum_{c\in t_g\setminus t_g^*}\frac{minsup-1}{|t_g|}\leqslant\frac{minsup-1}{|\mathcal{O}|}.$$

Thus, $S_A\leqslant\widehat{S}_A+\frac{minsup-1}{|\mathcal{O}|}$, or $\widehat{S}_A\geqslant S_A-\frac{minsup-1}{|\mathcal{O}|}$. □

Theorem 2 shows that, by using the iceberg lattice, average similarity may decrease at most $\frac{minsup-1}{|\mathcal{O}|}$. We want to remark here that when *minsup* is small, just a little decrease in *minsup* can incur considerable computation and storage overhead for a graph mining algorithm, while the similarity measure is just very slightly different.

The similarity measure computed on the iceberg lattice is, however, not upper bounded by the similarity measure computed on the complete lattice, that is, it is possible that $\widehat{S}_A > S_A$. The reason is that, the measure based on the complete lattice uses infrequent subgraphs where a lot of these can be considered as noises. The advantages of using the iceberg lattice are therefore twofold. First, it provides a computationally feasible way of defining similarity. Second, by its capability of filtering out noise, the measure can capture well the true similarity between graphs.

3.3. Relation to other similarity measures for graph data

Graph edit distance (Bruno et al., 1998; Sanfeliu & Fu, 1983) and the *maximum common subgraph* based distance measures (MCS-based measures) (Bunke & Shearer, 1998; Wallis et al., 2001) are well-known methods for comparing graphs. The edit distance between two graphs is defined as the minimum cost of edit operations that transforms one graph into the other. It is obvious that edit distance measure critically depends on the costs of the underlying edit operations. Bunke and Allermann (1997) proves that, under particular cost functions, computing the graph edit distance is equivalent to finding maximum common subgraphs. In this subsection, we investigate the relation between MCS-based measures and our measure based on the graph lattice.

Definition 9 Bunke and Allermann (1997). Let g_1 and g_2 be two graphs. A common subgraph g of g_1 and g_2 is called maximum if there exists no other common subgraph of g_1 and g_2 that has more nodes than g.

Notice that the maximum common subgraph is not uniqely defined by Definition 9. The maximum common subgraph of two graphs g_1 and g_2 can be any of common subgraphs of g_1 and g_2 which has the largest number of nodes. In Bunke and Allermann (1997), the insertion or deletion of an edge incident to a node that is also inserted or deleted, therefore, are defined of having zero cost. Given two graphs g_1 and g_2, it is safe to pick up a subgraph having the largest number of edges among all common subgraphs of g_1 and g_2 which have the largest number of nodes as the maximum common subgraph.

Definition 10. Let g_1 and g_2 be two graphs, and let $MCS(g_1,g_2)$ be the set of common subgraphs of g_1 and g_2 which have the largest number of nodes. The maximum common subgraph of g_1 and g_2, denoted as $mcs(g_1,g_2)$, is a subgraph that has the largest number of edges among all subgraphs in $MCS(g_1,g_2)$.

Proposition 6. *Let g_1 and g_2 be two graphs, and let c_1 and c_2 be two formal concepts whose intents are respectively $\{g_1\}$ and $\{g_2\}$. The maximum common subgraph $mcs(g_1,g_2)$ of g_1 and g_2 is a member of the intent of a lowest common ancestor of c_1 and c_2.*

Proof. The maximum common subgraph $mcs(g_1,g_2)$, as defined by Definition 10, is also a closed subgraph. By Theorem 1, there exists a formal concept c such that $mcs(g_1,g_2)$ is a member of the intent of c. Therefore, c is a common ancestor of c_1 and c_2 according to the $\leqslant$ order. Suppose that c is not a lowest common ancestor of c_1 and c_2, then there is a common ancestor c' of c_1 and c_2 such that $c'\leqslant c$. In this case, $mcs(g_1,g_2)$ is subsumed in a member of the intent of c', or $mcs(g_1,g_2)$ is not maximal. Thus, c is a lowest common ancestor of c_1 and c_2. □

MCS-based methods (Bunke & Shearer, 1998; Wallis et al., 2001) measure the similarity based on the computation of the size of the maximum common subgraph, thus, according to Proposition 6, they can be considered as using a concept in the lattice for the computation. Because MCS-based measures do not use all the concepts but a concept whose the intent containing a subgraph with the largest number of nodes, they have the following unwanted properties:

1. Similarity between two graphs is defined by a concept that may not be frequent, and therefore, not statistically significant.
2. There are possibly many concepts with intents are common subgraphs and make sense, but are not used by MCS-based measures.
3. Computing the maximum common subgraph is in general NP-Complete.

We give an example to illustrate the second property mentioned above. In chemistry, many chemical compounds share some C_nH_n and are different at, for example, COOH or OH. However, it is the COOH and the OH that determine the properties of being acid or ethanol of the compounds, not the large root C_nH_n. The third property is equivalent to the fact that it is in general infeasible to generate concepts locating near the bottom of the lattice.

3.4. An application in graph classification

A similarity measure can be used in various data analysis tasks including classification, clustering, distance-based outlier detection and many more. In this paper, to verify the usefulness of the proposed similarity measure, we apply it in the graph classification problem which is an important challenge in knowledge discovery from complex data.

The task of graph classification is learning to assign a graph to one of several classes. Most of algorithms proposed for this task fall into the feature-based approach (Bandyopadhyay et al., 2006;

Table 1
Several statistics of the used datasets.

Dataset	# graphs	MV	AV	ME	AE
NCI1	4117	111	29	119	32
NCI33	3298	111	30	119	33
NCI41	3108	111	30	119	33
NCI47	4068	111	30	119	32
NCI81	4812	111	30	119	32
NCI109	4149	111	30	119	32
NCI145	3911	106	29	107	32
NCI330	4608	120	25	132	27
DD	1178	5748	284	14,267	715
AIDS	5621	222	28	234	30

MV: maximum vertice size, AV: average vertice size, ME: maximum edge size, AE: average edge size.

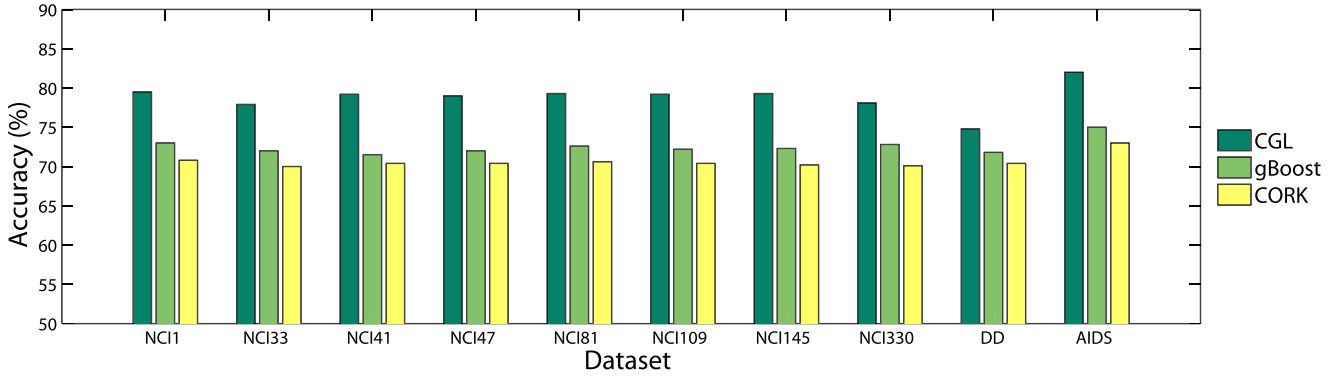

Fig. 8. Classification accuracies.

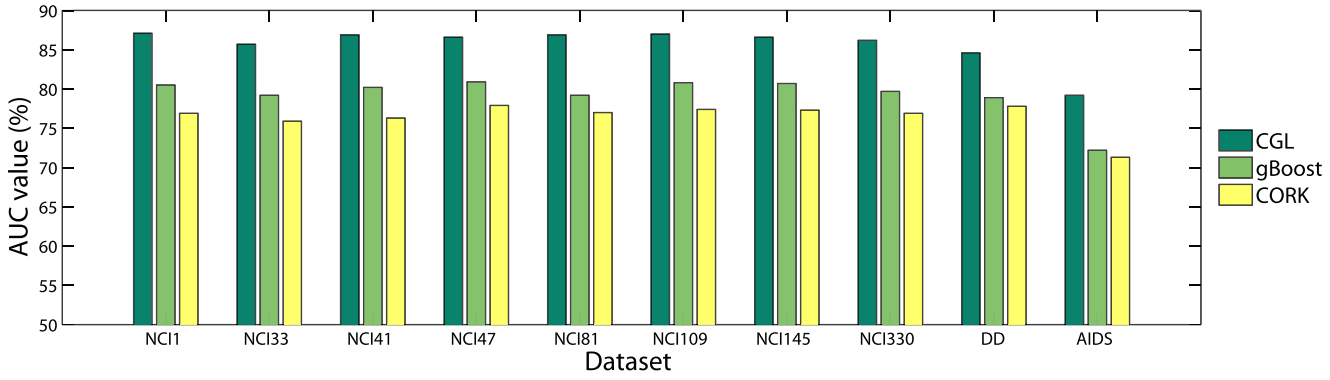

Fig. 9. Classification AUC values.

Table 2
Running time comparisons between CGL and gBoost.

Dataset	gBoost (secs)	CGL (secs)
NCI1	822	112
NCI33	840	72
NCI41	864	66
NCI47	896	116
NCI81	884	155
NCI109	860	111
NCI145	709	110
NCI330	912	98
DD	6473	386
AIDS	614	127

Deshpande, Kuramochi, & Karypis, 2003; Hasan & Zaki, 2009; Ranu & Singh, 2009; Rückert & Kramer, 2007; Saigo, Kraemer, & Tsuda, 2008; Saigo, Nowozin, Kadowaki, Kudo, & Tsuda, 2009; Schietgat, Costa, Ramon, & De Raedt, 2011; Thoma et al., 2009; Wale, Watson, & Karypis, 2008; Yan, Cheng, Han, & Yu, 2008; Zimmermann, Bringmann, & Rückert, 2010) as they all derive classification rules based on finding of subgraph patterns. Another approach, named similarity-based classification, estimates the class label of a test graph based on the similarities between the test graph and a set of labeled training graphs. The similarity-based classification, although is a popular method in usual domains of categorical data and Euclidean spaces, has not frequently been adopted for graph data. The main reason is that, for large graph datasets, the computation of similarity between graphs using existing methods becomes prohibitively expensive.

The k-Nearest Neighbor method (kNN), also called memory-based or case-based learning, predicts objects based on closest training examples according to some similarity or distance function. So it is natural to use kNN to verify the usefulness of the proposed similarity measure. To deal with the high computational intensity given by the out-of-sample similarity computations of the kNN method, we propose to apply the transductive setting. This means that the lattice is built from the set of all graphs. This solution is appropriate for applications where a large pool of unlabeled data is readily available, and the focus is the efficiency and accuracy of prediction such as the cases of chemical compounds and web documents.

The weighted k-nearest neighbor method is used to assign class to a test graph g as following

$$\hat{y} = \arg\max_{c \in C} \sum_{i=1}^{k} S(g, g_i)_{\{y_i = c\}} \tag{7}$$

where C is the set of all class labels.

4. Experimental evaluation

In this section, we evaluate the performance of the nearest neighbor algorithm using the similarity measure proposed in this paper and compare it with state-of-the-art algorithms for graph classification.

For the experiments, we employ the datasets shown in Table 1. This includes eight anti-cancer screen datasets (NCI), one Dobson and Doig (DD) molecular dataset, and one antiviral screen (AIDS) dataset.[1]

The classification algorithm for graph data based on the proposed similarity measure is called CGL (Classification of Graph

[1] These datasets are originally reported in Thoma et al. (2009) and available to download at http://www.dbs.ifi.lmu.de/~thoma/pub/sam2010/data.zip.

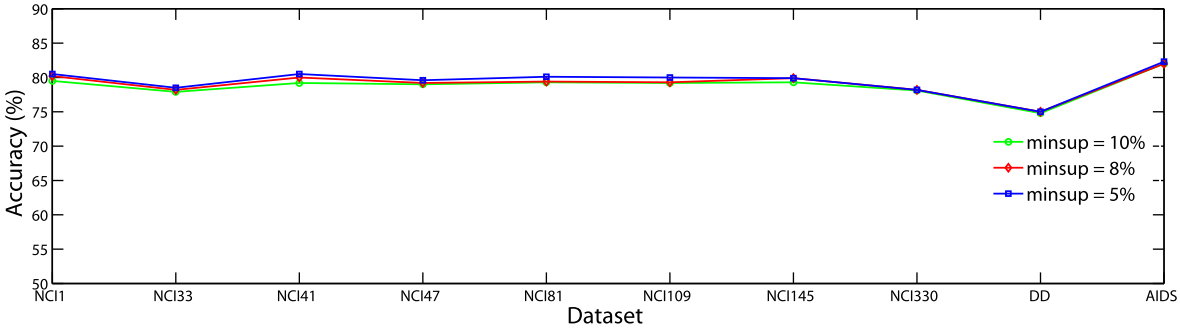

Fig. 10. Varying minimum support thresholds – accuracy values.

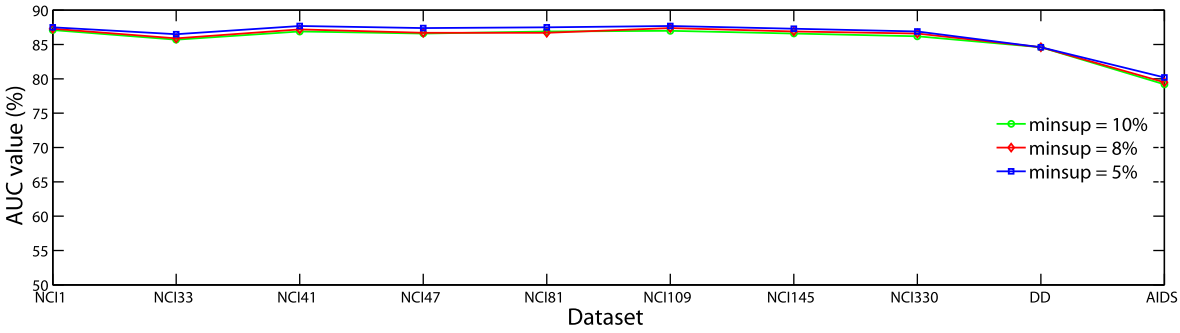

Fig. 11. Varying minimum support thresholds – AUC values.

Table 3
Running times for making links among concepts.

Dataset	# concepts	Time (secs)
NCI1	3156	13.6
NCI33	2891	9.7
NCI41	2930	9.5
NCI47	3180	13.5
NCI81	2826	12.7
NCI109	3055	12.8
NCI145	3268	14.0
NCI330	1527	6.8
DD	112,740	–
AIDS	499	0.6

Data Using Lattice-based Similarity). We compare CGL with gBoost (Saigo et al., 2009) and CORK (Thoma et al., 2009), two of the state-of-the-art algorithms that can run on the datasets mentioned above.

For CGL, the CloseGraph algorithm (Yan & Han, 2003) is used to generate closed subgraphs with *minsup* = 10%. The measure given in Eq. 6 is used. The number of nearest neighbors k is 6 for all datasets. For gBoost, we set *minsup* = 5% for all datasets and search the optimal parameter v within a range of four values {0.02, 0.04, 0.06, 0.08}. For CORK, we follow the setting in Thoma et al. (2009). Note that in CORK, *minsup* is adapted to the number of remaining graphs at each iteration so that subgraphs with low frequencies can also be mined. We use 10-fold cross-validation for classification. Experiments are measured on a 2.4 GHz Intel Core 2 Duo CPU with 2 GB of RAM, running Ubuntu 8.04.

We first measure the accuracy and the area under the ROC curve (AUC) for all the algorithms. The accuracy is shown in Fig. 8 and the AUC is shown in Fig. 9. We do not show the standard deviations since they are around 2%–4% for all these methods. As can be seen in the figures, CGL outperforms gBoost and CORK, both in accuracy and in AUC, for all the ten datasets. The consistent and significant improvement of classification results of our method suggests that, for non-vectorial data such as graphs, a local method based on the definition of a similarity measure between graphs could be a better choice over global inductive methods. The results also show that the graph similarity measure based on the iceberg lattice is accurate and reliable as we expected.

Next, we evaluate the runtime efficiency of the algorithms. Table 2 shows the runtimes of CGL and gBoost. We do not report the runtimes of CORK because the current release of CORK requires very long execution time. As observed in the table, CGL is much faster than gBoost, a fast global inductive method, on all the datasets. Since both CGL and gBoost use the set of subgraph patterns generated by a mining algorithm for the learning purpose, their runtime behaviors are mainly determined by the used mining algorithm. gBoost take longer time since it needs to call the mining algorithm multiple times with a low value of *minsup* in order to obtain a good predictive rule set.

We also test CGL with difference values of *minsup* to see how the prediction performance changes as *minsup* decreases. The test results agree with the result of Theorem 2. From Figs. 10 and 11, we can see that the accuracy values as well as the AUC values are just marginally increased or stayed the same as the *minsup* decreases from 10% to 5% in all the datasets. The results have significant meaning in dimensionality reduction and in reducing computation time since at higher minimum supports, much less subgraphs can be quickly generated.

As we have noted in Section 4, the computation of similarity measure given by Eq. 5 does not require to know the structure of the lattice explicitly. The time for making links among concepts, therefore, is not included in the previous tests. We show in Table 3 the runtime spent for this task of making links among concepts together with the number of concepts generated. As can be seen from Table 3, we achieve fast runtime for all dataset with only

one exception by the Dobson & Doig. Dobson & Doig dataset generates too many concepts, leading to long runtime and is forced to terminate after one hour of CPU time.

5. Conclusion

In this paper, we have shown that complex graph data can be modelized using Formal Concept Analysis. The exploitation of hidden, structural relationships among local subgraph patterns may help facilitate better understanding of graph data. For the purpose of statistical learning, we have devised a similarity measure for graph data taking into account the hierarchical structure of the lattice of subgraph patterns explicitly. We have shown that the upper part of the lattice, when reaching an appropriate size, provides a feasible and reliable estimation of true similarity among graphs. The lattice of subgraph patterns as well as the similarity measure devised on it might be well suited to unsupervised, semi-supervised and supervised learning. Kernel methods might also be applied in this framework. These will be topics of future work.

Acknowledgment

This work is partly supported by Kyoto University Global COE "Information Education and Research Center for Knowledge-Circulating Society".

References

Aggarwal, C., & Wang, H. (2000). *Managing and mining graph*. Springer.
Amini, O., Fomin, F., & Saurabh, S. (2009). Counting subgraphs via homomorphisms. In *ICALP-09* (pp. 71–82).
Bandyopadhyay, D., Huan, J., Liu, J., Prins, J., Snoeyink, J., Wang, W., et al. (2006). Structure-based function inference using protein family-specific fingerprints. *Protein Science, 15*, 1537–1543.
Bartl, E., Rezankova, H., & Sobisek, L. (2011). Comparison of classical dimensionality reduction methods with novel approach based on formal concept analysis. In *RSKT 11* (pp. 26–35).
Borgwardt, K. M., & Kriegel, H.P. (2005). Shortest-path kernels on graphs. In *DM-05* (pp. 74–81).
Bruno, B. T., Messmer, T., & Bunke, H. (1998). A new algorithm for error-tolerant subgraph isomorphism detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 20*, 493–504.
Bunke, H., & Allermann, G. (1997). On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters, 18*, 689–694.
Bunke, H., & Shearer, K. (1998). A graph distance metric based on the maximal common subgraph. *Pattern Recognition Letters, 19*, 255–259.
Charikar, M. (2000). Greedy approximation algorithms for finding dense components in a graph. In *APPROX-00* (pp. 84–95).
Coustaty, M., Bertet, K., Visani, M., & Ogier, J. M. (2011). A new adaptive structural signature for symbol recognition by using a galois lattice as a classifier. *IEEE Transactions on Systems, Man., and Cybernetics*, 1136–1148.
Davey, B., & Priestley, H. (2002). *Introduction to lattices and order* (2nd ed.). Cambridge University Press.
Deshpande, M., Kuramochi, M., & Karypis, G. (2003). Frequent sub-structure-based approaches for classifying chemical compounds. In *ICDM-03* (pp. 35–42).
Felzenszwalb, P. F., & Huttenlocher, D. P. (1998). Image segmentation using local variation. In *CVPR-98* (pp. 98–104).
Ganter, B., & Wille, R. (1998). *Formal concept analysis. Mathematical foundations*. Springer.
Gao, J., & Lai, W. (2010). Formal concept analysis based clustering for blog network visualization. In *ADMA 10* (pp. 394–404).
Gärtner, T., Flach, P., & Wrobel, S. (2003). On graph kernels: Hardness results and efficient alternatives. In *COLT-03* (pp. 129–143).
Hasan, M. A., & Zaki, M. J. (2009). Output space sampling for graph patterns. In *VLDB-09*.
Hereth, J., Stumme, G., Wille, U., & Wille, R. (2000). Conceptual knowledge discovery and data analysis. In *ICCS-00* (pp. 421–437).
Horvath, T., Ramon, J., & Wrobel, S. (2006). Frequent subgraph mining in outerplanar graphs. In *KDD-06* (pp. 197–206).
Inokuchi, A., Washio, T., & Motoda, H. (2000). An apriori-based algorithm for mining frequent substructures from graph data. In *PKDD-00* (pp. 13–23).
Ishida, T., Honda, K., & Kawahara, Y. (2008). Implication and functional dependency in intensional contexts. *Bulletin of Informatics and Cybernetics, 40*, 101–111.
Kuramochi, M., & Karypis, G. (2001). Frequent subgraph discovery. In *ICDM-01* (pp. 313–320).
Nicolaou, C. A., & Pattichis, C. S. (2006). Molecular substructure mining approaches for computer-aided drug discovery: A review. In *ITAB-06* (pp. 26–28).
Nowozin, S., Tsuda, K., Uno, T., Kudo, T., & Bakir, G. (2007). Weighted substructure mining for image analysis. In *CVPR-07* (pp. 1–8).
Poelmans, J., Elzinga, P., Viaene, S., & Dedene, G. (2010). Formal concept analysis in knowledge discovery: A survey. In *ICCS-10* (pp. 139–153).
Priss, U. (2007). Formal concept analysis in information science. *Annual Review of Information Science and Technology, 40*(1), 521–543.
Ramon, J., & Gärtner, T. (2003). Expressivity versus efficiency of graph kernels. In *Workshop on mining graphs, trees and sequences* (pp. 65–74).
Ranu, S., & Singh, A. K. (2009). Graphsig: A scalable approach to mining significant subgraphs in large graph databases. In *ICDE-09*.
Rückert, U., & Kramer, S. (2007). Optimizing feature sets for structured data. In *ECML-07* (pp. 716–723).
Saigo, H., Kraemer, N., & Tsuda, K. (2008). Partial least squares regression for graph mining. In *KDD-08* (pp. 578–586).
Saigo, H., Nowozin, S., Kadowaki, T., Kudo, T., & Tsuda, K. (2009). gBoost: A mathematical programming approach to graph classification and regression. *Machine Learning, 75*(1), 69–89.
Sanfeliu, A., & Fu, K. (1983). A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics, 13*, 353–362.
Schietgat, L., Costa, F., Ramon, J., & De Raedt, L. (2011). Effective feature construction by maximum common subgraph sampling. *Machine Learning, 83*(2), 137–161.
Snásel, V., Horák, Z., Abraham, A. (2008). Understanding social networks using formal concept analysis. In *IEEE/WIC/ACM 08* (pp. 390–393).
Thoma, M., Cheng, H., Gretton, A., Han, J., Kriegel, H., Smola, A., et al. (2009). Near-optimal supervised feature selection among frequent subgraphs. In *SDM-09*.
Valtchev, P., Missaoui, R., & Godin, R. (2004). Formal concept analysis for knowledge discovery and data mining: The new challenges. In *ICFCA-04* (pp. 352–371).
Wale, N., Watson, I. A., & Karypis, G. (2008). Comparison of descriptor spaces for chemical compound retrieval and classification. *Information System, 14*(3), 347–375.
Wallis, W. D., Shoubridge, P., Kraetz, M., & Ray, D. (2001). Graph distances using graph union. *Pattern Recognition Letters, 22*, 701–704.
Wasserman, S., & Faust, K. (1994). *Social network analysis: Methods and applications*. Cambridge University Press.
Wille, R. (2002). Why can concept lattices support knowledge discovery in databases? *Journal of Experimental and Theoretical Artificial Intelligence, 14*(2-3), 81–92.
Yan, X., & Han, J. (2002). gSpan: Graph-based substructure pattern mining. In *ICDM-02* (pp. 721–724).
Yan, X., & Han, J. (2003). Closegraph: Mining closed frequent graph patterns. In *KDD-03* (pp. 286–295).
Yan, X., Cheng, H., Han, J., & Yu, P. S. (2008). Mining significant graph patterns by leap search. In *ICMD-08* (pp. 433–444).
Zaki, M. J., & Ogihara, M. (1998). Theoretical foundations of association rules. In *ACM SIGMOD 98*.
Zimmermann, A., Bringmann, B., & Rückert, U. (2010). Fast, effective molecular feature mining by local optimization. In *PKDD-10* (pp. 563–578).