Concept Lattices as a Search Space for Graph Compression

Lucas Bourneuf, Jacques Nicolas

Concept Lattices as a Search Space for Graph Compression

Lucas Bourneuf✉ and Jacques Nicolas✉

Univ Rennes 1
Campus de Beaulieu, 35042 Rennes cedex, France.
{lucas.bourneuf,jacques.nicolas}@inria.fr

Abstract. Because of the increasing size and complexity of available graph structures in experimental sciences like molecular biology, techniques of graph visualization tend to reach their limit. To assist experimental scientists into the understanding of the underlying phenomena, most visualization methods are based on the organization of edges and nodes in clusters. Among recent ones, Power Graph Analysis is a lossless compression of the graph based on the search of cliques and bicliques, improving the readability of the overall structure. Royer et al. introduced a heuristic approach providing approximate solutions to this NP-complete problem. Later, Bourneuf et al. formalized the heuristic using Formal Concept Analysis. This paper proposes to extend this work by a formalization of the graph compression search space. It shows that (1) the heuristic cannot always achieve an optimal compression, and (2) the concept lattice associated to a graph enables a more complete exploration of the search space. Our conclusion is that the search for graph compression can be usefully associated with the search for patterns in the concept lattice and that, conversely, confusing sets of objects and attributes brings new interesting problems for FCA.

Keywords: concept lattice, search space formalization, graph, visualization

1 Introduction: graph compression for graph visualization

Visual representation of graphs remains a difficult problem either because of their size or because of the complexity of their topological analysis. Colors and spatial organization of nodes and edges are usually used to simplify the visualization [9], but such approaches remain limited, especially in experimental domains where the aim is to discover relevant graph structures.

Before displaying smartly or mining the interesting subgraphs in the graph, one can simplify the overall structure by replacing them by easier-to-read objects, for instance replacing multiple edges by a single one, which is known to increase readability [4]. This is the core idea of Power Graph Analysis: identifying bicliques and cliques in the graph in order to abstract them into simpler objects. The choice of bicliques and cliques is motivated by their prevalence in organized

data, such as biological [1, 10] or social networks [15]. The resulting compressed graph is equivalent to the input graph, but provides high compression rate over the edges, hence increasing readability and simplifying the interpretation work for application such as protein-protein interactions [13], graph comparison [11] or community detection in social networks [15]. Figure 3 features an example of Power Graph Analysis applied to the graph in Figure 2.

The best known effort to find a high level canonical structure in graphs that helps its analysis and visualization is Modular Decomposition [7]. However, it has been designed with different objectives in mind than Power Graph Analysis.

Modular Decomposition is a canonical representation enabling a complexity reduction of standard problems, such as MCE [8], graph drawing [12] or clustering [14]. It defines modules as sets of nodes having exactly the same neighbors outside the module. This restrictive relation enables a fast computation (linear with respect to the graph size) of a unique graph representation.

In contrast, Power Graph Analysis introduces powernodes, sets of nodes sharing common neighbors (thus a superset of modules), created together with poweredges, set of edges linking two powernodes. Thus edges and nodes are clustered at the same time. Because of the less restrictive definition of powernodes compared to modules, Power Graph Analysis is resistant to noise, allowing to extract motifs in the graph even in presence of interfering edges. This contrasts with the behaviour of Modular Decomposition, where a biclique is not guaranteed to appear simply as two parallel modules in a series module if the biclique partially interacts with other elements. The richness of structures in Power Graph Analysis leads to a NP-complete problem, but allows a more suitable formalism, to capture some relevant knowledge from the graph, and to obtain and visualize a more abstract representation of it. For instance, Power Graph Analysis provided, compared to Modular Decomposition, a more complete overview of protein-protein interaction networks, graphically extracting not only protein complexes but also hub proteins and domain induced interactions [13, 6].

Because of the high complexity of Power Graph Analysis, Modular Decomposition efficiency could probably help to draw powergraphs, or as a heuristic to search for interesting clusters of nodes that could be used as seeds for more complex structures in Power Graph Analysis. To our knowledge, combining Modular Decomposition and Power Graph Analysis has never been tried.

Power Graph: Given a graph $G = (V, E)$, a *Power Graph* is defined as a special graph $PG = (PV, PE)$, with nodes $PV \subseteq 2^V$ and edges $PE \subseteq 2^E$, which fulfill the three following conditions:

subgraph condition Any pair of powernodes connected by a poweredge represents a biclique in G. As a special case, a clique in G is represented by a single powernode and a poweredge looping on this powernode.

powernode hierarchy condition Any two powernodes are either disjoint, or one is included in the other. From the point of view of classification, the sets of vertices clustered in powernodes form a hierarchy.

poweredge decomposition condition poweredges form a partition of the set of edges.

It has been shown that finding an optimally compressed Power Graph (having the minimal number of poweredges) is an NP-complete problem [5]. A greedy strategy for graph compression consists to look for maximal cliques and bicliques iteratively. A previous article [2] dealt with the use of FCA to formalize this heuristic iterative search for concepts of maximal surface. This method however provides no guarantee to find the optimal compression This new study also show that this approach is incomplete (section 6) because some particular graphs cannot be optimally covered by maximal bicliques.

The main paper contribution is to use the FCA framework and the concept lattice, to give a formalization of the Power Graph search space. Section 3 exposes an introductory example providing the necessary prerequisites for our framework, and describes how to compress a graph from a concept lattice. Section 4 proposes a simplification of the search space in order to improve its exploration. Sections 5 and 6 explain the search for two specific motifs, cliques and concept cycles. Section 7 concludes with a discussion on the potential applications of such a formalization and the open questions that this general approach raises.

2 Graph Contexts

A *graph context* is a formal context (X, Y, I) where the set of objects X and attributes Y are subsets of vertices in an undirected graph $G = \{V, E\}$ and the binary relation $I \in X \times Y$ represents its edges E. Usually, formal contexts are defined on disjointed sets X and Y. In graph contexts objects and attributes represent the same kind of elements and can intersect. For an undirected graph, the graph context is symmetric.

G. Chiaselotti and T. Gentile [3] have proposed to take $X = Y = V$ for a graph context. This corresponds to a standard representation of a graph by its (symmetrical) adjacency matrix. Note however that for a same graph, there may exist several interesting smaller representations trying to minimize the intersection of X and Y. For instance, a bipartite graph can be defined by a bipartition of disjointed sets X and Y. This can be important from a practical perspective since a number of treatments depend on the size of the matrix.

The authors shown that for simple undirected graphs (no loops, no multiple edges), the standard application of FCA leads to a coincident derivation operator for object and attributes: the derivative of a subset of vertices X in terms of graph is the neighborhood intersection of all its elements. It is equivalently the set of vertices whose neighborhood includes X. Since graphs are simple, the derivative of a subset of vertices X has no intersection with X. Thus concepts are pairs of disjointed sets. The concept lattice is by construction self-dual.

For graphs with loops or even for simple graphs if we not do not worry about reflexive edges, it is interesting to accept concepts that consider pairs of possibly intersecting sets. We study in this paper the case of simple graphs. The idea is to look at maximal rectangles in the adjacency matrix, as for standard concepts analysis, with the small but important difference that the elements on the main diagonal are don't care positions. Due to the symmetry of a formal concept,

several representations are possible for a same concept and the difficulty is to properly handle the inclusion relation between concepts and retain the maximal rectangles.

	1	2	3	4
1		×	×	×
2	×		×	×
3	×	×		
4	×	×		

Fig. 1: Formal context of a small graph

Take for instance the small graph context in Figure 1. With the classical definition proposed by Chiaselotti et al. and apart from the top and bottom concepts, concepts are $C_1 = \{1\} \times \{2,3,4\}$, $C_2 = \{2\} \times \{1,3,4\}$, and $C_3 = \{1,2\} \times \{3,4\}$, duplicated with their dual concepts $C'_1 = \{2,3,4\} \times \{1\}$, $C'_2 = \{1,3,4\} \times \{2\}$, and $C'_3 = \{3,4\} \times \{1,2\}$. C_1 and C_2 are linked to C_3 in the lattice an dually C'_3 to C'_1 and C'_2.

With the new definition allowing don't care values in the main diagonal, a tempting approach by its simplicity would be to put the diagonal at 1, then to apply standard FCA on the resulting graph context. Symmetry aside, this leads to three concepts:
$C_4 = \{1,2,3\} \times \{1,2,3\}$, $C_5 = \{1,2,4\} \times \{1,2,4\}$ and $C_6 = \{1,2\} \times \{1,2,3,4\}$. However, since the reflexive edges are don't care links these concepts can also be represented by the following equations that avoid duplicated edges: $C_4 = \{1,2\} \times \{2,3\}$, $C_5 = \{1,2\} \times \{2,4\}$ and $C_6 = \{1,2\} \times \{2,3,4\}$. Clearly, C_4 and C_5 are not maximal and C_6 is the sole admissible concept. Therefore, applying FCA on such contexts produce undesirable results. Power Graph Analysis requires in fact a more conservative approach, introducing "1" values only on demand on the diagonal. In the next section, we introduce first this graph compression issue as a search in the concept lattice of graph contexts.

3 Compression From the Lattice

A simple graph compression can be achieved by picking one concept and compressing the edges it describes in poweredges. It is possible to proceed incrementally until complete compression (remaining edges cannot be clustered in larger bicliques).

Let us consider the graph defined by the formal context in Figure 2. It can be compressed by choosing first concept number 11 $\{d,i,j,k\} \times \{e,f,g,h\}$ in the concept lattice, then concept number 5 $\{d,e\} \times \{a,b,c,g\}$ (see Figure 3). This compression order is noted $(11,5)$. Since these concepts overlap (they share nodes g, e and d, and edge $d \times g$), meeting the decomposition and hierarchy

conditions require to split the second concept. It will be split in this case into three poweredges: $\{a, b, c\} \times \{d\}$, $\{a, b, c\} \times \{e\}$ and $\{g\} \times \{e\}$.

Choosing these concepts leads us to an optimal compression of the graph, but other concept lists may lead to different compressions with different number of poweredges: for instance, order $(1, 2, 3, 4)$ will give a 5 poweredges compression. The chosen concepts and their ordering determine the final compressed graph. The standard heuristic is to always compress first the concept, or in case of overlap the poweredge, covering the highest number of edges. This will however not necessarily yield the best compression.

In the following sections we study three features of this search space. First, due to the symmetry of the formal context, the lattice has itself a symmetry exchanging the extent and intent. In our example, 5 and 9 are symmetric as well as 3 and 11. Therefore, $(11, 9)$, $(3, 9)$, $(3, 5)$ and $(11, 5)$ will give the same compressions. The treatment of concept lattice symmetry and other properties is detailed in section 4.

Second, bicliques are not sufficient. Our example does provides some cliques of 3 elements, but they cannot be compressed as such, since concepts encode only bicliques. Section 5 show how to introduce both cliques and bicliques motifs into the search space.

Finally, there is no guarantee that an optimal compression can always be represented by an optimal ordering of concepts. Section 6 gives an example of such case, and discusses how it may be handled.

	a	b	c	d	e	f	g	h	i	j	k
a				×	×						
b				×	×						
c				×	×						
d	×	×	×		×	×	×	×			
e	×	×	×	×			×		×	×	×
f				×					×	×	×
g				×	×				×	×	×
h				×					×	×	×
i					×	×	×	×			
j					×	×	×	×			
k					×	×	×	×			

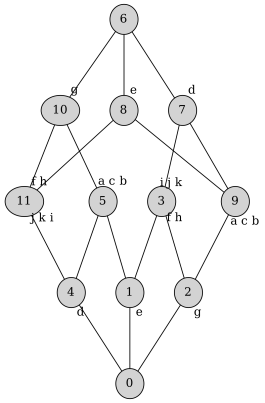

Fig. 2: Formal context describing the graph shown in Figure 3 and its associated concept lattice.

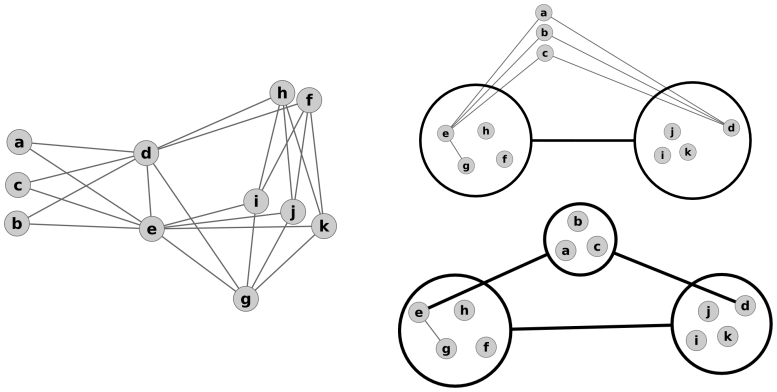

Fig. 3: The graph associated to the formal context in Figure 2 (left) and its Power Graph compression (lower-right), where the concept/biclique $\{d, i, j, k\} \times \{e, f, g, h\}$ (number 3 or 11 in Figure 2) was compressed first (leading to the partially compressed graph in upper-right), followed by concept/biclique $\{a, b, c, g\} \times \{d, e\}$ (number 5 or 9 in Figure 2), that is split in 3 poweredges.

4 Reducing the Search Space

4.1 Reducing the Concept Lattice

Given a set of n concepts, the set of all possible orderings is the number of permutations of size $1..n$, suggesting a factorial complexity for a complete search. It is therefore important to try to minimize the number of concepts needed to contain exponential growth.

The formal context is symmetric by construction : all nodes are present both in objects and in attributes. This property is visible in the concept lattice, where a concept (A, B) has a symmetric (B, A). One of the two can be discarded from the set of concepts to consider during the search for an order, because they cover exactly the same edges, i.e. they are redundant with respect to compression.

A naive solution to handle symmetry would be to fix an arbitrary ordering $>$ on the object/attribute sets and remove duplicates (A, B) if $A > B$. This is correct but many inclusion links between concepts may be lost in the remaining structure since the choices are independent from the lattice structure.

A structure-preserving procedure has therefore been designed, described in algorithm 1. It uses an arbitrary fixed total ordering on concepts (numbering) to choose a direct child of the supremum to keep, and consequently a direct parent of the infimum to discard. Each direct child of the supremum is called a *root*. Their symmetrical are the direct parents of the infimum. By marking a root as *kept* (i.e. deciding to keep it in the reduced lattice), we also set its symmetrical to be *discarded* (absent in the reduced lattice). All childs of the root are also kept, and therefore all parents of the root symmetrical (which includes at least

one root) are discarded. This is repeated until no more root is marked either as kept or discarded.

Because a node cannot be linked to itself in the formal context, it is not possible to have a concept marked both as kept and discarded. The resulting structure is not itself a concept lattice, it is just used to choose the list of concepts for the compression.

A reduced concept lattice of the example graph in Figure 3 can be found in Figure 4.

Algorithm 1 Compute a reduction of a given symmetric lattice by deciding nodes/concepts to keep. There are multiple reductions possible (depending of the root order), but all are equivalent in term of edge cover.

Require: Symmetric Concept Lattice L, symmetries between concepts
Ensure: Compute the set of concepts to keep as *taken*

$taken \leftarrow \emptyset$
$discarded \leftarrow \emptyset$
$roots \leftarrow direct\_childs(supremum(L))$
for all $root \in roots$ **do**
  if $root \notin taken \cup discarded$ **then**
    $taken \leftarrow taken \cup subconcepts(root)$
    $discarded \leftarrow discarded \cup supconcepts(symmetric(root))$
  end if
end for

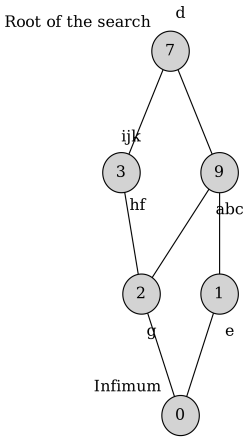

Fig. 4: A reduced concept lattice representation of the lattice in Figure 2. Concept 7 is the only root since it is the only child of the lattice supremum.

4.2 Reducing the number of concept permutations

Listing all permutations of concept sets ranging in size from 1 to n is only feasible for toy examples where there are no more than half a dozen concepts. Reducing the space of concept orderings is therefore necessary to solve real instances where lattices would typically contain thousands of concepts.

The problem can be formulated as follow : let C be a set of n concepts $\{c_1, c_2, ..., c_n\}$. Each permutation of k concepts $\in C$, $1 \leq k \leq n$, is associated with a score, so that:

- adding a concept to a permutation cannot increase its score (maximal score is given by the uncompressed graph)
- adding a concept (A, B) to a permutation will decrease its score only if there are at least two edges in $A \times B$ that are not covered, nor put in a different block by the treatment of a previous concept in the ordering.

The goal is to find a permutation associated with the smallest score without exploring the complete set of permutations.

Compression Order Cleaning: Because adding a concept covering no new edges cannot decrease the score, it is possible to add a concept to a permutation of concepts if it is not fully covered by these concepts.

Necessary Concepts (Kernel): A concept that appear in all the optimal permutations (regardless of symmetry) is necessary. An obvious optimization would be to discard any permutation that does not contains all the necessary concepts. Determining the necessary concepts is out of the scope of this article, but the following may give an insight of their properties.

Let a concept C be both an object-concept and an attribute-concept. C is the only concept (with its symmetric, removed by reduction) to cover edges $(x, y) \quad \forall x \in X, y \in Y$. Therefore the only way to compress them is to use C. If X and Y are singletons, only a single edge is concerned, thus there are no compressible edges specific to that concept. If $|X \cup Y| > 2$, there are at least two edges to compress together.

In the example of section 3, concept 11 and its symmetric 3 are the only ones to cover edges $\{f, h\} \times \{i, j, k\}$. They must be, one or the other, compressed at some point.

Note that these concept are not guaranteed to be necessary for concept cycles that will be introduced in section 6.

5 Handling cliques: Graph Contexts and Graph Concepts

Cliques are with bicliques the main structures managed by Power Graph Analysis. In particular, cliques are frequent in dense graphs. In formal contexts of simple graphs, they generate many concepts, none of them being associated directly with a clique. As a consequence, a clique will not be compressed simply

(i.e. as a powernode with a reflexive poweredge), but as many bicliques. Therefore, one cannot rely only on standard concepts for the management of cliques. We are thus proposing an extension of concepts, called triplet concepts, that is tailored for the treatment of simple graphs.

Let us consider the new extended concepts formed by possibly overlapping extent and intent. These concepts can be written as pairs $(A \cup C, B \cup C)$, where A, B and C are disjoint and C is the set of vertices common to the extent and the intent of the concept. Reflexive edges on C are considered as don't care positions. To be concepts, we have further to check that it is maximal with the standard meaning in FCA: it is not possible to add an element to A, B or C that adds a non-reflexive edge, which leads to a subgraph of the context graph. We will denote such concepts by a triplet (A, B, C). Note that from a graph perspective, C is a clique and (A, B) is a biclique. A triplet concept is thus a biclique between a clique and a biclique.

Definition 1. *(**Triplet concept**) Given a graph concept, a triplet concept is a triplet of disjointed sets (A, B, C) such that $A \times B$ is a biclique, C is a clique, and $(A \cup B) \times C$ is a biclique. Moreover (A, B, C) is maximal with respect to inclusion, i.e., it is not possible to add an element to either A, B or C adding a non-reflexive edge, which leads to a subgraph of the context graph.*

Note that triplet concepts (A, B, C) and (B, A, C) are two representations of the same concept and for practical computation one can impose A to be a set of size larger than or equal to B and use a lexicographic order on sets of same cardinality to compute one of the two forms.

	1	2	3	4	5	6	7
1		×	×	×	×	×	×
2	×		×	×	×	×	
3	×	×			×	×	×
4	×	×			×	×	
5	×	×	×	×			
6	×	×	×	×			×
7	×		×			×	

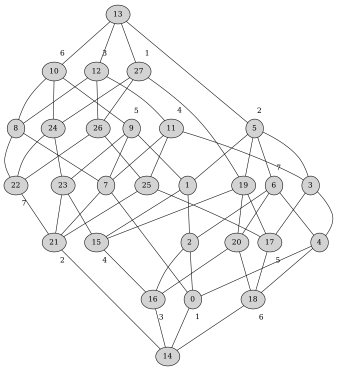

Fig. 5: A formal context and its concept lattice with reduced labelling.

Example

Consider the graph context in Figure 5. It admits 25 concepts when applying the approach proposed by Chiaselotti et al.

If one looks for triplet concepts instead, 5 are present in the graph:

- $TC_1 = (\{3,4\},\{5,6\},\{1,2\})$, corresponding to the biclique $\{1,2,3,4\} \times \{1,2,5,6\}$.
- $TC_2 = (\{1,2,3,4\},\{5,6\},\emptyset)$, corresponding to the biclique $\{1,2,3,4,6,7\} \times \{1,6\}$.
- $TC_3 = (\{2,7\},\emptyset,\{1,3,6\})$, corresponding to the biclique $\{1,2,3,6,7\} \times \{1,3,6\}$.
- $TC_4 = (\{2,3,4,7\},\emptyset,\{1,6\})$, corresponding to the biclique $\{1,2,3,4,6,7\} \times \{1,6\}$.
- $TC_5 = (\{2,5,6,7\},\emptyset,\{1,3\})$, corresponding to the biclique $\{1,2,3,5,6,7\} \times \{1,3\}$.

Note that as a special case, one of the set C or B may be empty. If C is empty, one gets the standard concepts. This is the case of TC_2. Three of the triplet concepts have a B set empty: TC_3, TC_4, and TC_5.

Triplet concepts may greatly reduce the needed number of concepts in case of dense context matrices: in this example the five triplets are equivalent to the 25 concepts, they completely cover the graph's edges. From the point of view of graph visualization, compression based on triplet concepts will be able to render the standard representation of cliques in Power Graph Analysis: a powernode with a reflexive poweredge.

Proposition 1. *Let A, B, C be three disjoint set of vertices from a context graph G. Using the standard derivation operation of FCA on this context graph, the following property holds:*

The triplet of disjointed non empty sets (A, B, C) is a triplet concept if and only if C is a maximal clique in the induced subgraph $G(A \cup B \cup C)$ completed by edges $(c, c), c \in C$, and there exist three sets D, E and F disjoint of sets A, B and C and three concepts (A, A'), (B, B'), and (C, C') such that:

$$A' = B \cup C \cup D \tag{1}$$
$$B' = A \cup C \cup E \tag{2}$$
$$C' = A \cup B \cup F \tag{3}$$
$$D \cap F = E \cap F = \emptyset \tag{4}$$

Proof. Assume first (A, B, C) is a triplet concept, then $A \times (B \cup C)$, $B \times A \cup C$, and $C \times B \cup C$ are bicliques of the graph. This implies that $B \cup C \subseteq A'$, $A \cup C \subseteq B'$, and $A \cup B \subseteq C'$ and the first three equations are valid. Assume that there exists an element $x \in D \cap F$. In such a case $x \in A'$ and $x \in C'$. This implies that $(A \cup C \cup \{x\}, B \cup C)$ is a valid biclique. Since the biclique $(A \cup C, B \cup C)$ is maximal, x should belong to A or C, a contradiction.

For the reciprocal, assume concepts (A, A'), (B, B'), and (C, C') exist with the properties in equations. Then $(A \cup C, B \cup C)$ is a biclique of the graph. One has to check that this biclique is maximal. If $(A \cup C \cup x, B \cup C)$ is a biclique, $x \notin A \cup C$, then $x \in E$ from the equation of B' in concept (B, B'). Now if $x \in B$, then x appears on both sides of $(A \cup C \cup \{x\}, B \cup C)$ and C is not a maximal clique on $G(A \cup B \cup C)$. Thus $x \notin B$. One can deduce $x \in F$ from the equation

of C' in concept (C, C'). The conclusion is that $x \in E \cap F$, a contradiction. The same reasoning applies if one adds an element to the right of the biclique. □

	a	b	c	d	e	f	g	h
a			×	×	×	×	×	×
b			×	×	×	×	×	×
c	×	×			×	×	×	×
d	×	×			×	×	×	×
e	×	×	×	×		×		
f	×	×	×	×	×			
g	×	×	×	×				×
h	×	×	×	×			×	

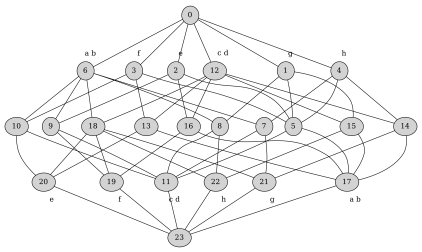

Fig. 6: A formal context and its concept lattice with reduced labelling, exhibiting the non-unicity of cliques in triplet concepts.

Note that for a given pair (A, B), the associated clique is not necessarily unique. Consider for instance the graph context in Figure 6. Two triplet concepts exist: $(\{a, b, c, d\}, \emptyset, \{e, f\})$ and $(\{a, b, c, d\}, \emptyset, \{g, h\})$.

Computing triplet concepts: The generation of triplet concepts can be achieved by a direct implementation of the previous proposition, shown in Algorithm 2: first all standard concepts are produced, then triplets are formed by a combination of three concepts that have all the desired properties. Finally, concepts that are covered by a triplet concept are removed.

For example we consider the graph context in Figure 7, containing 93 formal concepts from which 7 triplet concepts can be inferred:

$(\{a, b, c, d, e, f, g, j, k\}, \emptyset, \{h, i\})$
$(\{d, e, f, g, j, k\}, \emptyset, \{c, h, i\})$
$(\{a, b, c, g, j, k\}, \{d, e\}, \{h, i\})$
$(\{g, j, k\}, \{d, e\}, \{c, h, i\})$
$(\{d, e, f\}, \emptyset, \{c, h, i, j, k\})$
$(\{c, h, i, j, l\}, \{d, e, f\}, \{k\})$
$(\{a, b, c\}, \{d, e\}, \{g, h, i\})$

92 of the standard concepts are covered by the 7 triplet concepts. The only concept that is not covered is $(\{a, b, c, g, h, i, j, k, l\}, \{d, e\})$. As a consequence, the compression search space for this graph is constituted of 8 triplet concepts, a neat reduction with respect to the initial concept lattice. These 8 elements E can be partially ordered, so that $(A_1, B_1, C_1) \leq (A_2, B_2, C_2)$ if $A_1 \subseteq A_2$, $B_1 \supseteq B_2$ and $C_1 \supseteq C_2$. For the symmetric triplets, $\overline{E}$, the partial order reverse the direction of inclusions for the cliques: $(A_1, B_1, C_1) \leq (A_2, B_2, C_2)$ if $A_1 \subseteq A_2$, $B_1 \supseteq B_2$ and $C_1 \subseteq C_2$. For edges linking an element in E and an element in $\overline{E}$ (bold edges in the Figure), the relation of inclusion between cliques can appear in both directions. The complete line diagram is given in Figure 8.

Algorithm 2 A brute-force algorithm to compute triplet concepts.

Require: Nodes $\mathcal{N}$, concepts $\mathcal{C}$
Ensure: Computation of triplet concepts
for all $A \subset \mathcal{N}$ **do**
  for all $B \subset \mathcal{N} \mid A \subseteq B'$ **do**
    for all $C \subset \mathcal{N} \mid A \cup B \subseteq C'$ *and clique*(C) **do**
      $D \leftarrow A' \setminus (B \cup C)$
      $E \leftarrow B' \setminus (A \cup C)$
      $F \leftarrow C' \setminus (A \cup B)$
      if $D \cap E = E \cap F = \emptyset$ *and* $B \cup C \subseteq A'$ *and* $A \cup C \subseteq B'$ **then**
        yield (A, B, C)
      end if
    end for
  end for
end for

	a	b	c	d	e	f	g	h	i	j	k	l
a				×	×		×	×	×			
b				×	×		×	×	×			
c				×	×	×	×	×	×	×	×	
d	×	×	×				×	×	×	×	×	×
e	×	×	×				×	×	×	×	×	×
f			×					×	×	×	×	×
g	×	×	×	×	×			×	×			
h	×	×	×	×	×	×	×		×	×	×	
i	×	×	×	×	×	×	×	×		×	×	
j			×	×	×	×		×	×		×	
k			×	×	×	×		×	×	×		×
l				×	×	×					×	

Fig. 7: Formal context of a dense graph, yielding a total of 93 formal concepts. Two of the maximal cliques are intersecting : $\{d, g, h, i\}$ and $\{c, h, i, j, k\}$.

6 Triplet concepts are not sufficient: Concept-Cycles

One specific class of concepts layout is resisting to the previous concept approach. We call them cycles of concepts, an example being displayed in Figure 9 for a 4-cycle. A cycle of concepts is a series of concepts that form a circular chain by inclusion of the intent of one concept in the extent of the next one. Any cycle from 3 to any number of concepts will in fact never be optimally compressed by the procedure we have used so far iterating on the choice of concepts.

The previous section has shown the interest of clique motifs in addition to bicliques to compress graphs and we highlight here a new cycle pattern that underlines the richness of this pattern recognition approach.

As already sketched, the peculiar status of this cycling motif is due to a special organization of concepts, that the concept lattice helps to unravel. In a concept cycle motif, all involved formal concepts of the form $\{(A_1, B_1), ..., (A_n, B_n)\}$ are ordered so that $A_k \subset B_{k+1} \forall k \in 1..n-1$ and $A_n \subset B_1$. In fact, the basic building block for a cycle is a pair of overlapping concepts $A_1 \times (A_2 \cup B_1)$ and

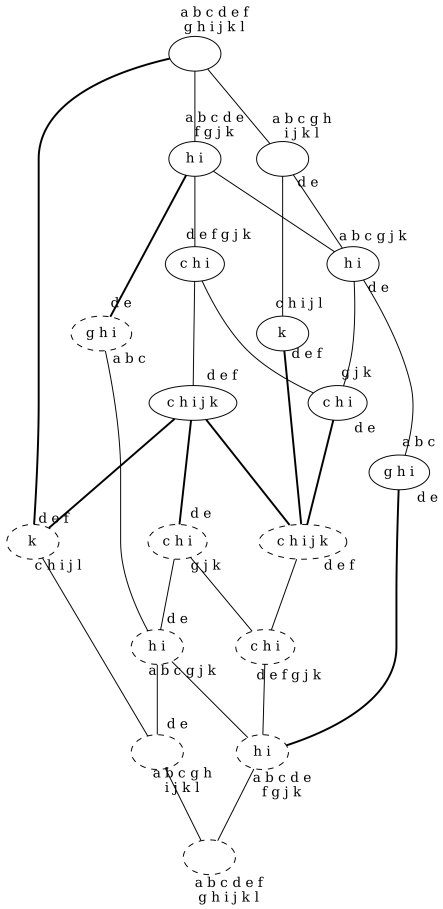

Fig. 8: A line diagram representation of the partial orders over computed triplet concepts (plain nodes) and their symmetric (dotted nodes) in graph context of Figure 7. Bold edges symbolize the "interface" between concepts and their symmetric. For a concept (A, B, C), A appears above the node, B below and C inside.

$A_2 \times (A_1 \cup B_2)$, where all sets are disjoint. The two concepts could be represented by a quadruplet (B_1, A_1, A_2, B_2), where all contiguous elements form a biclique. The biclique (A_1, A_2), also not maximal, is a consequence of the fact that a set may appear either as an extend or an intent.

It appears that cycle contexts often lead to a concept lattice that is made, apart from the top and bottom concept, of a graph cycle (see Figure 10 left) or two symmetric graph cycles (Figure 10 right). However, the 4-cycle (Figure 9) has a special shape where the cycle has been interrupted by intermediary concepts.

It is thus possible to find globally optimal bicliques organizations that are not based solely on the use of maximal bicliques but also use bicliques associated with overlapping concepts. An enumeration of the corresponding concepts can

lead to a systematic detection of the cyclic pattern, and the incorporation of better motif-concepts that would extend standard or triplet formal concepts.

Further work on the search space might also point to other specific motifs helping to better compress the graph through meaningful recurrent patterns.

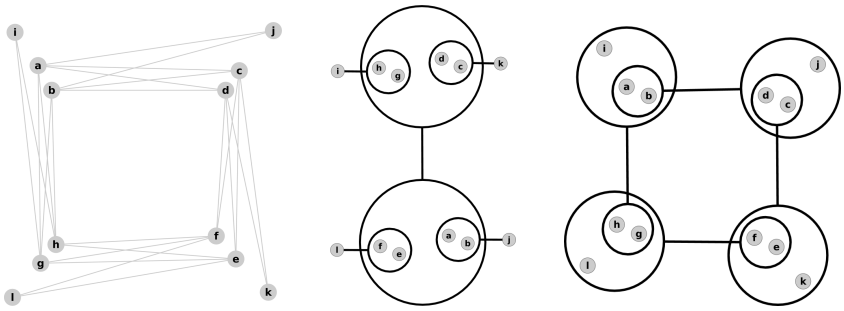

Fig. 9: A cycle of 4 concepts, non-compressed (left), compressed by a greedy method (middle), and optimally compressed (right). The greedy approach compresses first the largest concept (here $\{a, b, e, f\} \times \{c, d, g, h\}$), then 4 small bicliques, ending with 5 poweredges. The optimal compression is reached by using and splitting the four concepts $\{a, b\} \times \{c, d, e, f, j\}$, $\{c, d\} \times \{a, b, e, f, k\}$, $\{e, f\} \times \{c, d, g, h, l\}$, $\{g, h\} \times \{a, b, e, f, i\}$, leading to only 4 poweredges.

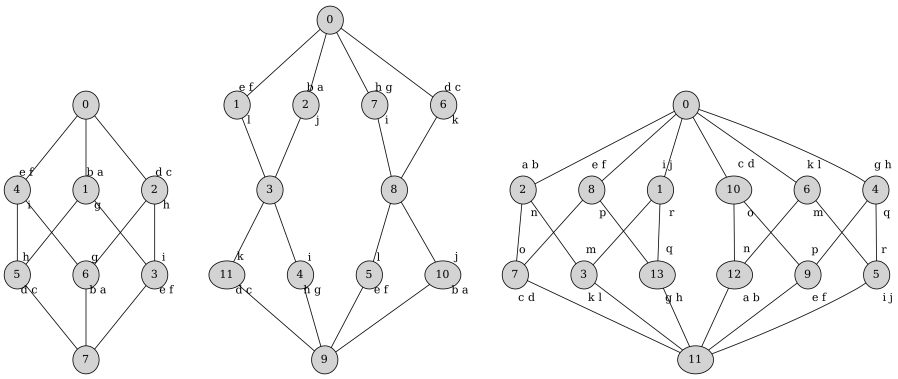

Fig. 10: The lattices of the 3-cycle (left), 4-cycle (middle) and 6-cycle (right) motifs. The latter presents two symmetric cycles, which is the consequence of a bipartite graph (an odd cycle). On the other hand, the 3-cycle is not bipartite, so the lattice symmetry is not perfectly separated.

7 Discussion and Conclusion

A general graph compression search space formalization in the framework of FCA was formulated, highlighting the main sources of difficulty of the problem. It shows that considering objects and attributes as a single set successfully renews FCA's issues.

Limits of the Concept-Only Methods A method seeking only for concept-based compression in the graph context representation cannot reach the optimal compression, because of at least two graph motifs: cliques and concept-cycles. This paper presents two propositions to handle these motifs.

First, the notion of don't care position in the formal context allowed to handle cliques in an optimal way, i.e to understand them not as a intertwining of bicliques/concepts, but as the single coherent representation that is used by Power Graph Analysis: a single powernode with a reflexive poweredge.

Second, to handle the concept-cycle motif, which uses non maximal bicliques, some specific pattern detection on the lattice has to be designed.

For a more global point of view, applying this approach of pattern recognition in the concept lattice could also be the basis for any motif that would convey a specific meaning in the data.

A General Compression Method Once the final (triplet and motif) concepts covering the graph have been generated, the compression process can be expressed as the choice of an ordering of a subset of all concepts. We have shown that object and attribute-concepts are useful to focus on particular subsets and the selection of subsets remain an interesting track for further researches.

The standard heuristic for the generation of graph compression is fast but only computes an approximation of the minimal Power Graph [2, 13]. Once the (triplet) concepts have been generated, the results of the Power Graph Analysis can be reproduced by ordering the concepts by decreasing surface. This heuristic avoids to explore the space of all permutations, explaining its efficiency, despite that the approach based on a permutation over the concepts is not feasible for graphs having more than a dozen concepts.

Other approaches to graph compression ought to improve the speed or the optimality by allowing to reuse edges among multiple poweredges [5], or the overlapping of powernodes to handle simply non-disjoint sets [1]. Such approaches correspond to a relaxation of some of the constraints on Power Graphs. This can be encoded in the concept lattice formalization as a relaxation of the compression of concepts operating on the same nodes or edges. The search space for that matter is not different, despite the constraint relaxations.

Open problems We end with a series of open problems that our new framework has raised.

- From a (triplet) concept lattice find a minimal subset of concepts that are *sufficient* to cover the whole graph. Find a maximal set of concepts that are *necessary* in all optimal compression.

– Can one predict, without processing the compression itself, the number of poweredges that would result from compressing a given list of concepts in a given order ?
– What is the structure of the search space as defined by extended concepts ?

References

1. S. E. Ahnert. Generalised power graph compression reveals dominant relationship patterns in complex networks. *Scientific reports*, 4, 2014.
2. L. Bourneuf and J. Nicolas. Fca in a logical programming setting for visualization-oriented graph compression. In *Formal Concept Analysis: 14th International Conference, ICFCA 2017, Rennes, France, June 13-16, 2017, Proceedings*, pages 89–105. Springer International Publishing, 2017.
3. G. Chiaselotti, D. Ciucci, and T. Gentile. Simple undirected graphs as formal contexts. In *International Conference on Formal Concept Analysis*, volume 9113, pages 187–302, 06 2015.
4. T. Dwyer, N. Henry Riche, K. Marriott, and C. Mears. Edge compression techniques for visualization of dense directed graphs. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2596–2605, Dec. 2013.
5. T. Dwyer, C. Mears, K. Morgan, T. Niven, K. Marriott, and M. Wallace. Improved optimal and approximate power graph compression for clearer visualisation of dense graphs. *CoRR*, abs/1311.6996, 2013.
6. J. Gagneur, R. Krause, T. Bouwmeester, and G. Casari. Modular decomposition of protein-protein interaction networks. *Genome Biology*, 5(8):R57, July 2004.
7. M. Habib and C. Paul. A survey of the algorithmic aspects of modular decomposition. *Computer Science Review*, 4(1):41–59, 2010.
8. B. Jocelyn and H. Seba. Solving the Maximal Clique Problem on Compressed Graphs. In *24th International Symposium on Methodologies on Intelligent (ISMIS 2018)*, Foundations of Intelligent Systems, pages 45–55, LImassol, Cyprus, Oct. 2018.
9. A. D. King, N. Pržulj, and I. Jurisica. Protein complex prediction via cost-based clustering. *Bioinformatics*, 20(17):3013–3020, Nov. 2004.
10. S. Navlakha, M. C. Schatz, and C. Kingsford. Revealing biological modules via graph summarization. *Journal of Computational Biology*, 16(2):253–264, 2009.
11. H. Ogata, W. Fujibuchi, S. Goto, and M. Kanehisa. A heuristic graph comparison algorithm and its application to detect functionally related enzyme clusters. *Nucleic Acids Research*, 28(20):4021–4028, Oct. 2000.
12. C. Papadopoulos and C. Voglis. Drawing graphs using modular decomposition. In P. Healy and N. S. Nikolov, editors, *Graph Drawing*, pages 343–354, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.
13. L. Royer, M. Reimann, B. Andreopoulos, and M. Schroeder. Unraveling Protein Networks with Power Graph Analysis. *PLoS Comput Biol*, 4(7):e1000108, 2008.
14. P. Serafino. Speeding up graph clustering via modular decomposition based compression. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC '13, pages 156–163, New York, NY, USA, 2013. ACM.
15. G. Tsatsaronis, M. Reimann, I. Varlamis, O. Gkorgkas, and K. Nørvåg. Efficient community detection using power graph analysis. In *Proc. of the 9th Workshop on Large-scale and Distributed Informational Retrieval*, LSDS-IR '11, pages 21–26, New York, NY, USA, 2011. ACM.