Building multi-adjoint concept lattices*

Juan Carlos Díaz, Bosco García, Jesús Medina, Rafael Rodríguez

Department of Mathematics. University of Cádiz
Email: {juancarlos.diaz,bosco.gacía,jesus.medina,rafael.rodriguez}@uca.es

Abstract

In Formal Concept Analysis it is very important to study fast algorithms to compute concept lattices. This paper introduces an algorithm on the multi-adjoint concept lattice framework, in order to compute the whole concept lattice. This fuzzy framework is very general and provides more flexibility in relational systems. This has been theoretically studied and, now, we need an algorithm to compute the concept lattice for each frame and context.

Keywords: Formal concept analysis; Galois connection; closure operators.

1. Introduction

Formal concept analysis (FCA) has been extended with ideas from fuzzy set theory [1, 2, 3], fuzzy logic reasoning [4, 5, 6], rough set theory [7, 8, 9], some integrated approaches such as fuzzy and rough [10], or rough and domain theory [11].

There exist several fuzzy extensions of formal concept analysis. To the best of our knowledge, the first one was given in [12], although it did not go much further beyond the basic definitions, probably due to the fact that it did not use residuated implications. Later, in [13, 1] the authors independently used complete residuated lattices as structures for the truth degrees.

Moreover, FCA has been related to rough set theory, which was originally proposed by Pawlak [14] as a formal tool for modelling and processing incomplete information in information systems, in a fuzzy environment in [15].

Multi-adjoint concept lattices were introduced [16, 17] as a new general approach to formal concept analysis, in which the philosophy of the multi-adjoint paradigm [18, 19] is applied to formal concept analysis. This environment provides a general and flexible framework in which the different approaches stated above could be conveniently accommodated. Moreover, richer information and better results from the initial databases can be obtained. For instance, this theory may be used in retrieval information as a tool to develop software which allows e.g. the efficient management of the energy in an infrastructure.

*Partially supported by the Spanish Science Ministry projects TIN2009-14562-C05-03 and TIN2012-39353-C04-04, and by Junta de Andalucía project P09-FQM-5233.

This paper presents an algorithm for calculating multi-adjoint concept lattices, which has been developed with a double goal: flexibility, from the point of view of the fast implementation of different algorithms, and the possibility of extension to the parallelization and high performance computing.

The current version uses a modification of the recently proposed algorithm given in [20], which computes the fixed points of closure operators and which is based on an estimate of the upper neighbors of each fixed point of the closure operator. Part of this algorithm has been implemented in *Python*, an interpreted programming language of general purpose that has dramatically increased in popularity in the last decade.

We have tried to exploit some of its advantages, including its versatility and speed of development as well as its orientation to objects. Although *Python*, as an interpreted language, is not remarkable for its great features of computing, we have evaluated especially the variety of options for adapting the algorithms to high performance computing, once they have been properly tested and validated. Among these possibilities there is a whole range that extends from the use of libraries oriented to matrix calculation, such as *scipy/numpy*, to the programming of specific modules *C* or *C++* (the integration of *Python* and *C/C++* is natural and there are many possibilities for it), that could potentially cover all the program, thus making it possible to achieve optimal computation times.

This concern in processing speed is especially important in the scope of formal concept analysis, where the number of operations grows dramatically when the size of the set of objects and attributes increases. Furthermore, in the current version, we have made a significant effort too in these operations, maximizing the use of entire arithmetic and eliminating the use of floating point operations, thus reducing the computing time.

The plan of this paper is the following: multi-adjoint concept lattice framework is recalled in Section 2, together with several results. Section 3 considers new results and introduces a new module to compute, given a closure operator, the upper neighbors of each closure set. Later, this module is completed in order to obtain an algorithm to compute the whole multi-adjoint concept lattice and an example is introduced in Section 4. Section 5 presents the comparison of the algorithm introduced here to the algorithm that properly uses the one given

in [20] to obtain the fixed points of a closure operator. Lastly, the paper concludes with several conclusions and prospects for future work.

2. Multi-adjoint concept lattices

This section recalls one of the most important fuzzy concept lattices frameworks: multi-adjoint concept lattices [16]. Before that, the common operators used in these concept lattices will be presented.

The basic operators in this environment are the adjoint triples [21], which are formed by three mappings: a non-commutativity conjunctor and two residuated implications [22], which satisfy the well-known adjoint property.

Definition 1 *Let* $(P_1, \leq_1)$, $(P_2, \leq_2)$, $(P_3, \leq_3)$ *be posets and* $\&\colon P_1 \times P_2 \to P_3$, $\swarrow\colon P_3 \times P_2 \to P_1$, $\nwarrow\colon P_3 \times P_1 \to P_2$ *be mappings, then* $(\&, \swarrow, \nwarrow)$ *is an* adjoint triple *with respect to* P_1, P_2, P_3 *if:*

$$x \leq_1 z \swarrow y \quad \textit{iff} \quad x \,\&\, y \leq_3 z \quad \textit{iff} \quad y \leq_2 z \nwarrow x$$

for all $x \in P_1$, $y \in P_2$ *and* $z \in P_3$.

Example of adjoint triples are the Gödel, product and Łukasiewicz t-norms together with their residuated implications.

Multi-adjoint formal concept lattice framework generalizes the classic formal concept analysis and several fuzzy ones [17, 16]. The following definition presents the basic structure which allows the existence of several adjoint triples with respect to L_1, L_2, P, where $(L_1, \preceq_1)$ and $(L_2, \preceq_2)$ are complete lattices.

Definition 2 *A* multi-adjoint frame *is a tuple*

$$(L_1, L_2, P, \preceq_1, \preceq_2, \leq, \&_1, \swarrow^1, \nwarrow_1, \ldots, \&_n, \swarrow^n, \nwarrow_n)$$

where $(L_1, \preceq_1)$ *and* $(L_2, \preceq_2)$ *are complete lattices,* $(P, \leq)$ *is a poset and, for all* $i = 1, \ldots, n$, $(\&_i, \swarrow^i, \nwarrow_i)$ *is an adjoint triple with respect to* L_1, L_2, P.

Given a frame, a *multi-adjoint formal context* is a tuple consisting of sets of objects, attributes and a fuzzy relation among them; in addition, the multi-adjoint approach also includes a function which assigns an adjoint triple to each object (or attribute).

Definition 3 *Let* $(L_1, L_2, P, \&_1, \ldots, \&_n)$ *be a multi-adjoint frame, a* context *is a tuple* (A, B, R, σ) *such that* A *and* B *are non-empty sets (usually interpreted as attributes and objects, respectively),* R *is a* P*-fuzzy relation* $R\colon A \times B \to P$ *and* $\sigma\colon A \times B \to \{1, \ldots, n\}$ *is a mapping which associates any pair* (a, b) *in* $A \times B$ *with some particular adjoint triple in the frame.*

The set of mappings $g\colon B \to L_2$, $f\colon A \to L_1$ will be denoted as usual L_2^B and L_1^A, respectively. On these sets a pointwise partial order can be assumed from the partial orders in $(L_1, \preceq_1)$ and $(L_2, \preceq_2)$, which provides L_2^B and L_1^A with the structure of a complete lattice, that is, abusing notation, $(L_2^B, \preceq_2)$ and $(L_1^A, \preceq_1)$ are complete lattices, where $\preceq_2$ is defined pointwise, given $g_1, g_2 \in L_2^B$, $f_1, f_2 \in L_1^A$, $g_1 \preceq_2 g_2$ if and only if $g_1(b) \preceq_2 g_2(b)$, for all $b \in B$; and $f_1 \preceq_1 f_2$ if and only if $f_1(a) \preceq_1 f_2(a)$, for all $a \in A$.

Given a multi-adjoint frame and a context for that frame, the mappings $^{\uparrow_\sigma}\colon L_2^B \longrightarrow L_1^A$ and $^{\downarrow^\sigma}\colon L_1^A \longrightarrow L_2^B$, which generalize the classic definitions given in [23], and that can be seen as generalizations of those fuzzy mappings given in [4, 24], are defined, for all $g \in L_2^B$ and $f \in L_1^A$ as:

$$g^{\uparrow_\sigma}(a) = \inf\{R(a,b) \swarrow^{\sigma(b)} g(b) \mid b \in B\} \quad (1)$$

$$f^{\downarrow^\sigma}(b) = \inf\{R(a,b) \nwarrow_{\sigma(b)} f(a) \mid a \in A\} \quad (2)$$

It is not difficult to show that these two arrows generate a Galois connection [16].

The notion of concept is defined as usual: a *multi-adjoint concept* is a pair $\langle g, f\rangle$ satisfying that $g \in L_2^B$, $f \in L_1^A$ and that $g^{\uparrow_\sigma} = f$ and $f^{\downarrow^\sigma} = g$; with $(^{\uparrow_\sigma}, ^{\downarrow^\sigma})$ being an antitone Galois connection.

Therefore, one important feature of these mappings is that their compositions form closure operators. This property is very important to allow to calculate fix points, that is, the concepts in this framework.

Definition 4 *Let* $(P, \leq)$ *be a poset,* $c\colon P \to P$ *is a* closure operator *if it is increasing,* $x \leq c(x)$ *and* $c(c(x)) = c(x)$*, for all* $x \in P$.

Finally, the definition of concept lattice in this framework is defined [16].

Definition 5 *The* multi-adjoint concept lattice *associated with a multi-adjoint frame* $(L_1, L_2, P, \&_1, \ldots, \&_n)$ *and a context* (A, B, R, σ) *is the set*

$$\mathcal{M} = \{\langle g, f\rangle \mid g \in L_2^B, f \in L_1^A \textit{ and } g^{\uparrow_\sigma} = f, f^{\downarrow^\sigma} = g\}$$

in which the ordering is defined by $\langle g_1, f_1\rangle \preceq \langle g_2, f_2\rangle$ *if and only if* $g_1 \preceq_2 g_2$ *(equivalently* $f_2 \preceq_1 f_1$*).*

In [16], the authors proved that the ordering just defined above provides $\mathcal{M}$ with the structure of a complete lattice. Moreover, a representation theorem to multi-adjoint concept lattices was proven, which generalizes the classic one and some other fuzzy generalizations.

3. Computing the fixed points of a closure operator

The concepts of a multi-adjoint concept lattice are obtained by the composition of the concept-forming operators. This composition is a closure operator, because these operators form a Galois connection. Therefore, computing the fixed points of a closure

operator is a very important step in any algorithm developed in order to obtain these concepts.

Among the algorithms introduced in the literature to obtain the fixed points of a closure operator, the most notable is the one presented in [20]. Specifically, it defines three modules: NEIGHBORS($B, \mathcal{C}$) (Algorithm 1), which gets the upper neighbors of a fuzzy set B from a closure operator $\mathcal{C}$. GENERATEFROM(B) (Algorithm 2, where B^* and B_* denote the upper and lower set of a subset B) is a recursive function that, from a set B, gets all the fixed points of the closure operator $\mathcal{C}$ greater than B, their upper and lower neighbors. Finally, LATTICE($\mathcal{C}, Y$) (Algorithm 3) applies the previous module to the lowest fixed point of the closure operator $\mathcal{C}$, obtaining the total set of fixed points of $\mathcal{C}$ along with their upper and lower neighbors, i.e. all the information needed to build the complete lattice of fixed points. For more details see [20].

Algorithm 1: NEIGHBORS(B, $\mathcal{C}$)

input : B, $\mathcal{C}$
output: Upper neighbors of B by $\mathcal{C}$

$\mathcal{U} := \varnothing$;
Min := $\{y \in Y \mid B(y) < 1\}$;
foreach $y \in Y$ **such that** $B(y) < 1$ **do**
  $D := [y]^{\mathcal{C}}_B$;
  Incr := $\{z \in Y \mid z \neq y,\ B(z) < D(z)\}$;
  if *Min* $\cap$ *Incr* $= \varnothing$ **then**
    add D **to** $\mathcal{U}$
  else
    remove y **from** Min
return $\mathcal{U}$

Algorithm 2: GENERATEFROM(B)

while $B \neq Y$ **do**
  $B^* :=$ NEIGHBORS($B, \mathcal{C}$);
  $\mathcal{N} := B^* - \mathcal{F}$;
  foreach $D \in B^*$ **do**
    add B **to** D_*;
    if $D \in \mathcal{N}$ **then**
      add D **to** $\mathcal{F}$;
  foreach $D \in \mathcal{N}$ **do**
    GENERATEFROM(D)
return

Now, a new module NEIGHBORS($B, \mathcal{C}$) (Algorithm 4) will be introduced. For that, different definitions and new results are introduced.

From now on, a set Y, a finite lattice $(L, \preceq)$ and a closure operator $\mathcal{C}\colon L^Y \to L^Y$, will be fixed. Although L does not need to be linear, in order to clarify the algorithm this will be assumed. Using linear extensions of the possibly non-linear order on

Algorithm 3: LATTICE($\mathcal{C}, Y$)

$\mathcal{F} := \varnothing$;
$B := \mathcal{C}(\varnothing)$;
add B **to** $\mathcal{F}$;
GENERATEFROM(B)
return
  $\langle \mathcal{F}, \{B^* \mid B \in \mathcal{F}\}, \{B_* \mid B \in \mathcal{F}\}\rangle$

L a new algorithm can be obtained.

The elements of L are written as $L = \{l_1, l_2, \ldots, l_k\}$, where $0 = l_1 < l_2 < \cdots < l_k = 1$, and we will write l_i^+ instead of l_{i+1}, for each $i \in \{1, \ldots, k-1\}$. Moreover, given $y \in Y$ and a fuzzy subset $B \in L^A$, such that $B(y) < 1$, the fuzzy subset $B \cup \{B(y)^+/y\}$ is defined as B except for the element y, in which it is defined as the next element, $B(y)^+$. The closure of this set, $\mathcal{C}\left(B \cup \{B(y)^+/y\}\right)$ is denoted as $[y]^{\mathcal{C}}_B$.

Note that, in general, given $B \in \mathrm{fix}(\mathcal{C})$, the element $B \cup \{B(y)^+/y\}$ may not be a fixed point of $\mathcal{C}$, but $[y]^{\mathcal{C}}_B = \mathcal{C}\left(B \cup \{B(y)^+/y\}\right) \in \mathrm{fix}(\mathcal{C})$. However, we cannot ensure that $[y]^{\mathcal{C}}_B$ is, in general, an upper neighbor of B.

The goal of the following results is to obtain the upper neighbors of a fixed point B. Hence, given $B, D \in \mathrm{fix}(\mathcal{C})$, such that $B \subset D$, we will consider the sets $S_B(D) = \{y \in Y \mid B(y) < D(y)\}$ and $G_B(D) = \{y \in Y \mid D = [y]^{\mathcal{C}}_B\}$.

Proposition 6 *Given $B \in \mathrm{fix}(\mathcal{C})$, $y \in Y$ and $D \in \mathrm{fix}(\mathcal{C})$, such that $B \subset D$. If $y \in S_B(D)$, then $[y]^{\mathcal{C}}_B \subseteq D$.*

Proof: Since $B(y)^+ \leq D(y)$, the inclusions

$$B \subset B \cup \{B(y)^+/y\} \subseteq D$$

hold. Hence, by the monotonicity of $\mathcal{C}$, we obtain the result:

$$B \subset [y]^{\mathcal{C}}_B \subseteq D$$

□

Proposition 7 *Let $B, D \in \mathrm{fix}(\mathcal{C})$, such that $B \subset D$, then D is an upper neighbor of B if and only if $S_B(D) = G_B(D)$.*

Proof: First of all, we assume that D is an upper neighbor of B. If $y \in G_B(D)$, then $D = [y]^{\mathcal{C}}_B$ and so $B(y) < [y]^{\mathcal{C}}_B(y) = D(y)$, that is $y \in S_B(D)$. If $y \in S_B(D)$, then, by Proposition 6, the inclusions $B \subset [y]^{\mathcal{C}}_B \subseteq D$ hold, and, as D is an upper neighbor of B, we obtain that $[y]^{\mathcal{C}}_B = D$, which leads us to $y \in G_B(D)$.

On the other hand, we assume that D is not an upper neighbor of B, then there exists $E \in \mathrm{fix}(\mathcal{C})$, such that $B \subset E \subset D$. Since $S_B(E) \neq \varnothing$, there exists an element y in $S_B(E)$ and, by Proposition 6, $[y]^{\mathcal{C}}_B \subseteq E$ holds.

Hence, we obtain that $[y]^{\mathcal{C}}_B \subseteq E \subset D$, and, consequently, $[y]^{\mathcal{C}}_B \neq D$. Therefore, $y \notin G_B(D)$, but $y \in S_B(D)$, because $B(y) < E(y) \leq D(y)$. Thus, $S_B(D) \neq G_B(D)$, which leads us to a contradiction and proves the result. □

Proposition 8 *Given $B, D \in \text{fix}(\mathcal{C})$, such that $B \subset D$. The equality $S_B(D) = G_B(D)$ holds if and only if the number of elements in $S_B(D)$ is equal to the number of elements in $G_B(D)$, that is $|S_B(D)| = |G_B(D)|$.*

Proof: The first implication is clear. For the other one, we have that if $y \in G_B(D)$ then $B(y) \leq [y]^{\mathcal{C}}_B(y) = D(y)$, and so $y \in S_B(D)$. Therefore, $G_B(D) \subseteq S_B(D)$ and, moreover, as they are finite sets, the assumption that $|S_B(D)| = |G_B(D)|$ leads to $S_B(D) = G_B(D)$. □

As a consequence of Propositions 7 and 8, the following result holds.

Corollary 9 *Given $B, D \in \text{fix}(\mathcal{C})$ satisfying $B \subset D$, the fuzzy subset D is an upper neighbor of B if and only if $|S_B(D)| = |G_B(D)|$.*

Hence, using the previous results, the module given in Algorithm 4 computes the upper neighbors of each fixed point B of $\mathcal{C}$.

This module computes, first of all, the sets $[y]^{\mathcal{C}}_B$, for each $y \in B$, such that $B(y) < 1$, and they are kept in P. However, it may be possible that there exist $y, z \in Y$, with $y \neq z$, such that $[y]^{\mathcal{C}}_B = [z]^{\mathcal{C}}_B$, hence, only one of these elements is considered and it is called *generator*.[1] Since, Y is ordered by an index set, we will consider the least of the generators of a set $[y]^{\mathcal{C}}_B$ and they will be kept in *Gen*.

Now, for each $y \in Gen$, the number of elements in $S([y]^{\mathcal{C}}_B)$ will be computed and added to $S\_counter(y)$. Moreover, the numbers of generators of each set $G([y]^{\mathcal{C}}_B)$ will be kept in $G\_counter(y)$ (that is, the elements in each equivalence class will be computed).

Finally, using Corollary 9, for each $y \in Gen$, if $S\_counter(y) = G\_counter(y)$, then $P(y) = [y]^{\mathcal{C}}_B$ is an upper neighbor and will be kept in U, otherwise, we continue with the next element.

This proves the correctness and termination of the algorithm. Moreover, it is clear the difference between Algorithm 1 and Algorithm 4.

4. Algorithm to compute the multi-adjoint concept lattice

In the classic formal concept analysis there are different algorithms to get the whole set of concepts and the relationship between them, i.e. the complete lattice of concepts, fastly and efficiently, such as those given in [25, 26]. However, there are not many efficient algorithms for the fuzzy case.

[1] Note that the equality $[y]^{\mathcal{C}}_B = [z]^{\mathcal{C}}_B$ provides an equivalence relation in Y and we only choose one element of each equivalence class.

Algorithm 4: Neighbors(B)

```
input : B, C
output: Upper neighbors of B by C
1  U := ∅; P := ∅;
   generators := ∅;
   S_counter := ∅;
   G_counter := ∅
2  foreach y ∈ Y such that B(y) < 1 do
3      P(y) := [y]^C_B
4      S_counter(y) := 0
5      foreach z ∈ Y such that z < y do
6          if B(z) < P(y)(z) then
7              S_counter(y) + +
8              if B(y) < P(z)(y) then
9                  G_counter(z) + +
10                 go to no_gen_min
11     add y to Gen
12     G_counter(y) := 0
13     foreach z ∈ Y such that z > y do
14         if B(z) < P(y)(z) then
15             S_counter(y) + +
16     label no_gen_min
17 foreach y ∈ Gen do
18     if S_counter(y) = G_counter(y) then
19         add P(y) to U
20 return U
```

We have introduced an algorithm to obtain multi-adjoint concept lattices considering the one given in [20] and the module explained in the previous section. We must emphasize that the algorithm requires finite carrriers. For instance, the lattices L_1, L_2 and P can be regular partitions of $[0, 1]$, and these can be different for each L_1, L_2 and P, i.e. we can consider $L_1 = [0, 1]_{20}$, $L_2 = [0, 1]_8$ and $P = [0, 1]_{100}$, as in the example introduced in [16].

To its credit is the fact that you can use different adjoint triples, because the composition of the mappings of a Galois connection in the multi-adjoint approach is a closure operator, even though different adjoint triples are considered.

Now, we present some examples where performed implementation has been used. The first of these is taken from [20], and we have asserted that the same results are given.

Example 10 *The considered multi-adjoint frame*

is $\mathcal{L} = (L, \preceq, \&_L)$*, where* $L = \{0, 0.5, 1\}$ *and* $\&_L$ *is the Łukasiewicz conjunctor defined on* L*. In this framework, the context is* (A, B, R, σ)*, where* $A = \{y_1, y_2, y_3, y_4, y_5\}$*,* $B = \{x_1, x_2, x_3\}$*,* σ *is constantly* $\&_L$ *and* $R\colon A \times B \to L$ *is given by Table 1. In this example, as there are* $3^3 = 27$ *fuzzy subsets*

Table 1: Relation R of Example 10.

R	x_1	x_2	x_3
y_1	1	1	0
y_2	0.5	1	0
y_3	0.5	1	0.5
y_4	1	1	0.5
y_5	1	0.5	1

of objects and $3^5 = 243$ *fuzzy subsets of attributes, at most there may be* 27 *concepts, if all subsets of objects were intents of concepts, but the number of actual concepts is less. In this case, the concept lattice* $(\mathcal{M}, \preceq)$*, associated with the framework and context previously set, has 10 concepts listed below.*

$$
\begin{aligned}
C_0 &= \langle \{0.5/x_1, 0.5/x_2\}, \\
&\quad \{1.0/y_1, 1.0/y_2, 1.0/y_3, 1.0/y_4, 1.0/y_5\}\rangle \\
C_1 &= \langle \{0.5/x_1, 1.0/x_2\}, \\
&\quad \{1.0/y_1, 1.0/y_2, 1.0/y_3, 1.0/y_4, 0.5/y_5\}\rangle \\
C_2 &= \langle \{1.0/x_1, 0.5/x_2\}, \\
&\quad \{1.0/y_1, 0.5/y_2, 0.5/y_3, 1.0/y_4, 1.0/y_5\}\rangle \\
C_3 &= \langle \{0.5/x_1, 0.5/x_2, 0.5/x_3\}, \\
&\quad \{0.5/y_1, 0.5/y_2, 1.0/y_3, 1.0/y_4, 1.0/y_5\}\rangle \\
C_4 &= \langle \{1.0/x_1, 1.0/x_2\}, \\
&\quad \{1.0/y_1, 0.5/y_2, 0.5/y_3, 1.0/y_4, 0.5/y_5\}\rangle \\
C_5 &= \langle \{0.5/x_1, 1.0/x_2, 0.5/x_3\}, \\
&\quad \{0.5/y_1, 0.5/y_2, 1.0/y_3, 1.0/y_4, 0.5/y_5\}\rangle \\
C_6 &= \langle \{1.0/x_1, 0.5/x_2, 0.5/x_3\}, \\
&\quad \{0.5/y_1, 0.5/y_2, 0.5/y_3, 1.0/y_4, 1.0/y_5\}\rangle \\
C_7 &= \langle \{1.0/x_1, 1.0/x_2, 0.5/x_3\}, \\
&\quad \{0.5/y_1, 0.5/y_2, 0.5/y_3, 1.0/y_4, 0.5/y_5\}\rangle \\
C_8 &= \langle \{1.0/x_1, 0.5/x_2, 1.0/x_3\}, \\
&\quad \{0.5/y_3, 0.5/y_4, 1.0/y_5\}\rangle \\
C_9 &= \langle \{1.0/x_1, 1.0/x_2, 1.0/x_3\}, \\
&\quad \{0.5/y_3, 0.5/y_4, 0.5/y_5\}\rangle
\end{aligned}
$$

The Hasse diagram of this lattice is shown in Fig. 1.

Considering the Gödel conjunctor in the frame, $(L, \preceq, \&_G)$*, the complete lattice obtained is isomorphic to the one above, although the concepts are different.*

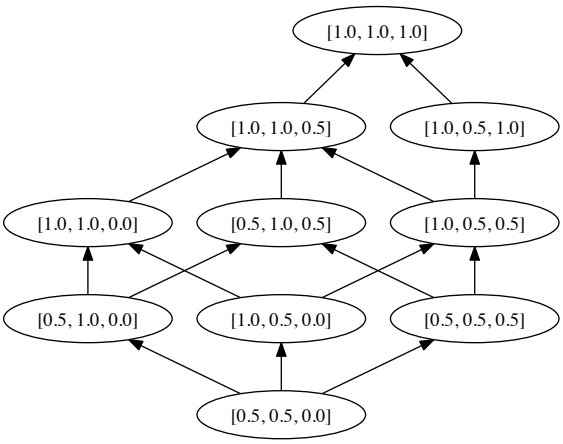

Figure 1: The Hasse diagram of $(\mathcal{M}, \preceq)$

$$
\begin{aligned}
C'_0 &= \langle \{0.5/x_1, 0.5/x_2\}, \\
&\quad \{1.0/y_1, 1.0/y_2, 1.0/y_3, 1.0/y_4, 1.0/y_5\}\rangle \\
C'_1 &= \langle \{0.5/x_1, 1.0/x_2\}, \\
&\quad \{1.0/y_1, 1.0/y_2, 1.0/y_3, 1.0/y_4, 0.5/y_5\}\rangle \\
C'_2 &= \langle \{1.0/x_1, 0.5/x_2\}, \\
&\quad \{1.0/y_1, 0.5/y_2, 0.5/y_3, 1.0/y_4, 1.0/y_5\}\rangle \\
C'_3 &= \langle \{0.5/x_1, 0.5/x_2, 0.5/x_3\}, \\
&\quad \{1.0/y_3, 1.0/y_4, 1.0/y_5\}\rangle \\
C'_4 &= \langle \{1.0/x_1, 1.0/x_2\}, \\
&\quad \{1.0/y_1, 0.5/y_2, 0.5/y_3, 1.0/y_4, 0.5/y_5\}\rangle \\
C'_5 &= \langle \{0.5/x_1, 1.0/x_2, 0.5/x_3\}, \\
&\quad \{1.0/y_3, 1.0/y_4, 0.5/y_5\}\rangle \\
C'_6 &= \langle \{1.0/x_1, 0.5/x_2, 0.5/x_3\}, \\
&\quad \{0.5/y_3, 1.0/y_4, 1.0/y_5\}\rangle \\
C'_7 &= \langle \{1.0/x_1, 1.0/x_2, 0.5/x_3\}, \\
&\quad \{0.5/y_3, 1.0/y_4, 0.5/y_5\}\rangle \\
C'_8 &= \langle \{1.0/x_1, 0.5/x_2, 1.0/x_3\}, \\
&\quad \{0.5/y_3, 0.5/y_4, 1.0/y_5\}\rangle \\
C'_9 &= \langle \{1.0/x_1, 1.0/x_2, 1.0/x_3\}, \\
&\quad \{0.5/y_3, 0.5/y_4, 0.5/y_5\}\rangle
\end{aligned}
$$

Now, if the granularity of the carriers is changed to $[0, 1]_4 = \{0, 0.25, 0.5, 0.75, 1\}$*, then there are* $5^3 = 125$ *fuzzy subsets of objects and* $5^5 = 3\,125$ *fuzzy subsets of attributes. Considering the frame* $([0, 1]_4, \preceq, \&_G)$*, we obtain the same number of concepts in this case, that is,* 8*. However, for* $([0, 1]_4, \preceq, \&_L)$ *the number of concepts is* 35*, see Fig. 2.*

Applying the flexibility of the multi-adjoint framework, if we assume that the object x_3 *is preferred to the rest, then we can consider the context* (A, B, R, σ)*, where* $\sigma(a, x_3) = \&_L$ *and* $\sigma(a, x_3) = \sigma(a, x_3) = \&_G$*, for all* $a \in A$*. In this case,* 17 *concepts are obtained and they are represented in Fig. 3.*

5. Comparing both algorithms

This section considers the algorithm introduced previously and the other one that uses the algorithm given in [20] to obtain the fixed points of a closure

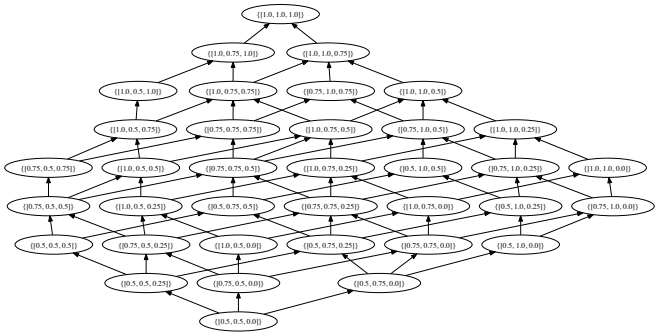

Figure 2: Concept lattice w.r.t. the frame $([0,1]_4, \preceq, \&_L)$

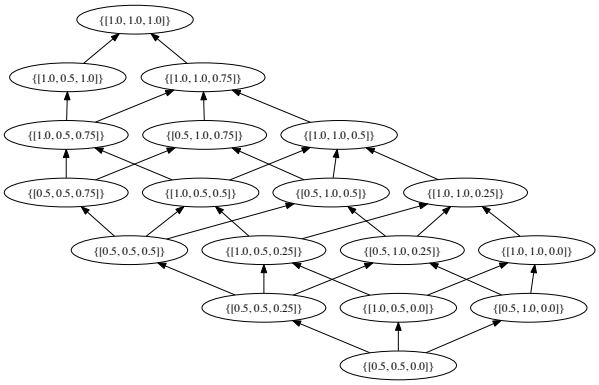

Figure 3: x_3 is preferred to the rest of objects

operator. With the purpose of comparing in practice the performance of both algorithms, a battery of experimental tests has been performed.

Specifically, we have chosen the frame

$$([0,1]_{20}, [0,1]_8, [0,1]_{100}, \leq, \leq, \leq, \&_P^*)$$

where $[0,1]_m$ denotes a regular partition of $[0,1]$ into m pieces and $\&_P^*$ the discretization of the product conjunctor. Moreover, we consider the contexts (A, B, R_{k_i}, σ), where $R_{k_1}, \ldots, R_{k_N}$ is a family of squared matrices, representing relations with an increasing number of rows (elements in A) and columns (elements in B). Random values have been assigned to the elements of each matrix R_{k_i}, with $i \in \{1, \ldots, N\}$.

For each relation R_{k_i}, a program is designed which builds the multi-adjoint concept lattice using two algorithms, one of them properly uses the algorithm given in [20] and the other one considers the new module presented in Section 3. The program saves the time employed by the computer using each one of these algorithms. In order to avoid any possible interference due to differences in the loading time of the computer when executing the test, the process has been repeated a large number of times (500 times for each algorithm). The process is repeated for $k_i = 20 * i$, with $i \in \{1, \ldots, 15\}$. The time employed by both algorithms is increasing super-linearly and the results are almost indistinguishable, although both algorithms have different structures. From this result, one can conjecture that the times employed by both algorithms are asymptotically equivalent when the number of objects (and attributes) diverge to $+\infty$.

However, new variables can be considered in the comparison, such as the granularity of the assumed carriers, that is, given a number of objects and attributes, we can increase m in the regular partitions $[0,1]_m$, in which the objects, attributes and relations are evaluated.

The study of the complexity with respect to the number of attributes n provides that both modules, Algorithms 1 and 4, have a quadratic complexity, in the worse case: $\mathcal{O}(n^2)$. Hence, the complexity is not improved but an alternative mechanism is considered with the same complexity. This can be interesting if efficient computational procedure can be applied to Algorithm 4 instead of Algorithm 1, such as using neural networks, etc. Moreover, the new module has been introduced based on new interesting results, which are useful to know the general building of the multi-adjoint concept lattices.

6. Conclusions and future work

An algorithm to build multi-adjoint concept lattices has been introduced. This has been implemented using *Python* programming language and *C++*. The main properties of *Python* that have been used are its versatility and speed of development, as well as its orientation to objects. These have been complemented with the properties of *C++*.

In addition, we have developed our own library of fuzzy sets (instead of opting for the various existing implementations in *Python*) that is directly oriented to sets of evenly spaced truth values, and so we can represent them by finite ranges of integer numbers. As a consequence, we have eliminated the use of floating point operations, which should lead to a reduction in computation time.

Furthermore, new results and a new module have been developed to obtain the upper neighbors of each closure set. This module has been completed in order to obtain a new algorithm, which computes the whole multi-adjoint concept lattice. This algorithm is compared to the one given in [20]. At the moment, we are finding out that the new module does not introduce new benefits in time, however it provides a different way to obtain the upper neighbors of each closure set, which opens a new way to future research. Moreover, this modification shows that the algorithm given in [20], in order to obtain the fixed points of a closure operator, is not easy to be improved.

The verification of the proposed improvements with respect to other implementations will be developed as future work. Moreover, *C* or *C++* will

be used in other specific parts of the algorithm in order to decrease the computation time.

References

[1] R. Bělohlávek. Fuzzy concepts and conceptual structures: induced similarities. In *Joint Conference on Information Sciences*, pages 179–182, 1998.

[2] Xiaodong Liu, Wei Wang, Tianyou Chai, and Wanquan Liu. Approaches to the representations and logic operations of fuzzy concepts in the framework of axiomatic fuzzy set theory I. *Information Sciences*, 177(4):1007–1026, 2007.

[3] Xiaodong Liu, Wei Wang, Tianyou Chai, and Wanquan Liu. Approaches to the representations and logic operations of fuzzy concepts in the framework of axiomatic fuzzy set theory II. *Information Sciences*, 177(4):1027–1045, 2007.

[4] R. Bělohlávek. Concept lattices and order in fuzzy logic. *Annals of Pure and Applied Logic*, 128:277–298, 2004.

[5] Shi-Qing Fan, Wen-Xiu Zhang, and Wei Xu. Fuzzy inference based on fuzzy concept lattice. *Fuzzy Sets and Systems*, 157(24):3177–3187, 2006.

[6] C. Alcalde, A. Burusco, R. Fuentes-González, and I. Zubia. Treatment of L-fuzzy contexts with absent values. *Information Sciences*, 179:1–15, 2009.

[7] Ming-Wen Shao, Min Liu, and Wen-Xiu Zhang. Set approximations in fuzzy formal concept analysis. *Fuzzy Sets and Systems*, 158(23):2627–2640, 2007.

[8] Min Liu, Mingwen Shao, Wenxiu Zhang, and Cheng Wu. Reduction method for concept lattices based on rough set theory and its application. *Computers & Mathematics with Applications*, 53(9):1390–1410, 2007.

[9] Qiang Wu and Zongtian Liu. Real formal concept analysis based on grey-rough set theory. *Knowledge-Based Systems*, 22(1):38–45, 2009.

[10] Lidong Wang and Xiaodong Liu. Concept analysis via rough set and AFS algebra. *Information Sciences*, 178(21):4125–4137, 2008.

[11] Yinbin Lei and Maokang Luo. Rough concept lattices and domains. *Annals of Pure and Applied Logic*, 159(3):333–340, 2009.

[12] A. Burusco and R. Fuentes-González. The study of *L*-fuzzy concept lattice. *Mathware & Soft Computing*, 3:209–218, 1994.

[13] S. Pollandt. *Fuzzy Begriffe*. Springer, Berlin, 1997.

[14] Z. Pawlak. Rough sets. *International Journal of Computer and Information Science*, 11:341–356, 1982.

[15] J. Medina. Multi-adjoint property-oriented and object-oriented concept lattices. *Information Sciences*, 190:95–106, 2012.

[16] J. Medina, M. Ojeda-Aciego, and J. Ruiz-Calviño. Formal concept analysis via multi-adjoint concept lattices. *Fuzzy Sets and Systems*, 160(2):130–144, 2009.

[17] J. Medina and M. Ojeda-Aciego. Multi-adjoint t-concept lattices. *Information Sciences*, 180(5):712–725, 2010.

[18] J. Medina, M. Ojeda-Aciego, and P. Vojtáš. Similarity-based unification: a multi-adjoint approach. *Fuzzy Sets and Systems*, 146:43–62, 2004.

[19] P. Julian, G. Moreno, and J. Penabad. On fuzzy unfolding: A multi-adjoint approach. *Fuzzy Sets and Systems*, 154(1):16–33, 2005.

[20] R. Bělohlávek, B. De Baets, J. Outrata, and V. Vychodil. Computing the lattice of all fixpoints of a fuzzy closure operator. *IEEE Transactions on Fuzzy Systems*, 18(3):546–557, 2010.

[21] M.E. Cornejo, Jesús Medina, and Eloisa Ramírez. A comparative study of adjoint triples. *Fuzzy Sets and Systems*, 211:1–14, 2012.

[22] P. Hájek. *Metamathematics of Fuzzy Logic*. Trends in Logic. Kluwer Academic, 1998.

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

[24] S. Krajči. A generalized concept lattice. *Logic Journal of IGPL*, 13(5):543–550, 2005.

[25] R. Bělohlávek, B. De Baets, J. Outrata, and V. Vychodil. Lindig's algorithm for concept lattices over graded attributes. *Lecture Notes in Computer Science*, 4617:156–167, 2007.

[26] C. Lindig. Fast concept analysis. In G. Stumme, editor, *Working with Conceptual Structures-Contributions to ICCS 2000*, pages 152–161, 2000.