Department of Computer Science
Faculty of Science
Palacký University Olomouc

MASTER THESIS

Factor analysis of ordinal data

Algorithms and experiments

2020

Tomáš Chlup

Supervisor: prof. RNDr. Radim Bělohlávek, Ph.D., DSc

Study field: Computer Science, full-time form

Bibliografické údaje

Autor:	Tomáš Chlup
Název práce:	Faktorová analýza relačních dat (Algoritmy a experimenty)
Typ práce:	diplomová práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2020
Studijní obor:	Informatika, prezenční forma
Vedoucí práce:	prof. RNDr. Radim Bělohlávek, Ph.D., DSc
Počet stran:	66
Přílohy:	1 CD/DVD
Jazyk práce:	anglický

Bibliograhic info

Author:	Tomáš Chlup
Title:	Factor analysis of ordinal data (Algorithms and experiments)
Thesis type:	Master thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2020
Study field:	Computer Science, full-time form
Supervisor:	prof. RNDr. Radim Bělohlávek, Ph.D., DSc
Page count:	66
Supplements:	1 CD/DVD
Thesis language:	English

Anotace

Formální konceptuální analýza (FCA) je metoda analýzy relačních dat. Tato práce se věnuje využití FCA k analýze objekto-atributových relací, ve kterých relační vztah může mít více než dva stavy (0,1). Práce obsahuje shrnutí potřebné teorie, a také experimenty s reálnými daty.

Synopsis

Formal concept analysis is a method of relation data analysis. This thesis deals with the usage of FCA for analysis of object-attribute relations, where the relationship could have more than just two states (0,1). This thesis summarizes the underlying theory and also experiments with real datasets.

Klíčová slova: formální konceptuální analýza, faktorová analýza, fuzzy logika, relační data

Keywords: formal concept analysis, FCA, factor analysis, fuzzy logic, ordinal data

I am grateful to everyone who supported me during the work on this thesis.

I hereby declare that I have completed this thesis including its appendices on my own and used solely the sources cited in the text and included in the bibliography list.

date of thesis submission author's signature

Contents

List of Figures

List of Tables

1 The basic motivation of the thesis

Formal concept analysis (FCA) is a method of data analysis, which is nowadays taking place across various domains. It can be described as a data mining technique, which can obtain or describe the relationships between a particular set of objects and their attributes. FCA has growing popularity and usage. Examples of uses are data mining, informational retrieval, and further data analysis. Input of FCA is set of several objects, set of attributes and the relation between these objects and attributes. One of the essential terms in FCA is a formal concept, which is one of the main outputs from FCA. Formal concepts are clusters, which should represent the human-like concept of real life notions like "winter sports" or "car with rare wheel drive." The primary purpose of the FCA is to find various kinds of patterns in the given data. Some of these patters could be obvious, but on the other side, we can find interesting hidden patterns that can not be easily found by any human. If we have, for example, a dataset with different kinds of sports, there could occur a natural pattern typical for all winter sports, and the FCA method can find it. The second important output from the FCA method are attribute implications. Attribute implication describes valid dependency in data, such as "every number divisible by 2 and 3 is also divisible by 6". Attribute implications are well known from another data mining method, so-called Market Basket Analysis.

Research in this area is widespread and has several difficulties and challenges which have to be solved. Classical FCA is designed for boolean data analysis. In case of boolean data as input, we have just two states for object-attributes relation. In simple words, the object has a particular attribute, or it does not have it. If we want to analyze the ordinal data, which are supposed to have more than two values for object-attribute relation, we have to deal with data transformation or extend the classical FCA algorithm. In this thesis, we will aim at the algorithms, which can be used for explanation of object-attribute relation datasets with a small set of essential factors. We will introduce and test methods of FCA, which will be extended for ordinal data analysis with the help of fuzzy logic (many-valued logic). Let us also denote, that wee will work only with the main FCA output, formal concepts.

In the next chapters, we will start with basics about the classical FCA, then the usage of FCA for boolean matrix decomposition, and expand this knowledge to factor analysis of ordinal data in the later chapters. In the very last chapter, we will show the usage of the FCA method on real datasets and also measure the performance of individual variants of our method.

2 Introduction to Formal Concept Analysis

This chapter is a summary of the basic theory of FCA. It consists of definitions and examples which are essential for the rest of the thesis. Content of this chapter was written with the help of [1].

2.1 Basic setting

The input data for FCA is the relation I between a particular set of objects X and a set of their attributes Y. This relation is most of the time represented as a matrix that has $|X|$ rows and $|Y|$ columns. For clarification, the $|X|$ denotes the number of elements in the set X. Since the matrix represents boolean object-attribute relation, it contains just element 0 or 1. A table entry $I_{x,y}$ containing 1 indicates that the corresponding object x has attribute y.

Example 1 (Example of matrix I)

$$I = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \end{array} \begin{array}{c} \begin{array}{ccc} y_1 & y_2 & y_3 \end{array} \\ \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix} \end{array}$$

where $X = \{x_1, x_2, x_3\}$, $Y = \{y_1, y_2, y_3\}$ and $I = \{\langle x_1, y_1\rangle$,$\langle x_1, y_3\rangle$,$\langle x_2, y_1\rangle$,$\langle x_2, y_2\rangle$,$\langle x_2, y_3\rangle$,$\langle x_3, y_3\rangle\}$.

Formally we will call matrix I the formal context.

Definition 2 (Formal context)
A formal context is a triplet $\langle X, Y, I\rangle$ where $X \neq \emptyset$, $Y \neq \emptyset$ and I is a binary relation between X a Y ($I \subseteq X \times Y$).

To define the formal concept and concept lattice, which is an output of FCA, we have to specify two special operators. Every formal context induces pair of operators $\uparrow, \downarrow$. We call them concept-forming operators. As these operations are induced by the triplet $\langle X, Y, I\rangle$, we should use $\uparrow_I, \downarrow_I$, but in this thesis simplified label will be written so we will omit the bottom label if the context is clear. Operator $\uparrow$ assign set of shared attributes for a set of objects. Analogically the $\downarrow$ for the set of attributes assigns a set of objects which share them.

Definition 3 (Concept-forming operators)
For $A \subseteq X$, $B \subseteq Y$ and $I \subseteq X \times Y$ let operators $\uparrow$,$\downarrow$ be defined by

$$A^{\uparrow} = \{y \in Y | \forall x \in A : \langle x, y\rangle \in I\},$$
$$B^{\downarrow} = \{x \in X | \forall y \in B : \langle x, y\rangle \in I\}.$$

Example 4 (Matrix I and operations $\uparrow, \downarrow$)

$$I = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{array} \begin{array}{c} \begin{array}{cccc} y_1 & y_2 & y_3 & y_4 \end{array} \\ \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

For example:
$\{y_1, y_3\}^{\downarrow} = \{x_1, x_2, x_4\}$,
$\{y_1, y_2, y_3\}^{\downarrow} = \{x_4\}$,
$\{x_2, x_5\}^{\uparrow} = \{y_4\}$,
$\{x_1, x_2, x_4\}^{\uparrow} = \{y_1, y_3\}$,
and elementary $X^{\uparrow} = \emptyset, \emptyset^{\uparrow} = Y$.

Now we will define the formal concept, which is a fundamental notion in FCA, and it is also one of the most important notions for the rest of the chapters in this thesis. Formal concepts is a pair $\langle A, B \rangle$, where A is a set of objects, and B is a set of attributes. These sets have to fulfill conditions associated with our concept-forming operators.

Definition 5 (Formal concept)
A pair $\langle A, B \rangle$ in formal context $\langle X, Y, I \rangle$ where $A \subseteq X$, $B \subseteq Y$ such that $A^{\uparrow} = B$ and $B^{\downarrow} = A$ is formal concept in I.

Formally for formal concept $\langle A, B \rangle$ A is the intent and B is extent. Formal concepts can be seen as a particular cluster in our matrix I.

Example 6 (Formal concepts)

$$I = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{array} \begin{array}{c} \begin{array}{cccc} y_1 & y_2 & y_3 & y_4 \end{array} \\ \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

Example of formal concepts in matrix I:
$\langle A_1, B_1 \rangle = \langle \{x_1, x_2, x_3, x_4\}, \{y_1\} \rangle$, $\langle A_2, B_2 \rangle = \langle \{x_3, x_4\}, \{y_1, y_2\} \rangle$, $\langle A_3, B_3 \rangle = \langle \{x_1, x_2, x_4\}, \{y_1, y_3\} \rangle$, $\langle A_4, B_4 \rangle = \langle \{x_2, x_5\}, \{y_4\} \rangle$.

Formal concepts can be as well defined as maximal rectangles in the matrix. At first, we have to define rectangle in a formal context.

Definition 7 (Rectangle in formal context)
A rectangle in formal context $\langle X, Y, I\rangle$ is pair $\langle A, B\rangle$ where $A \in X$ and $B \in Y$ such that $A \times B \subseteq I$ (alternatively for $x \in A$ and $y \in B$ there is $\langle x, y\rangle \in I$).

Also we put $\langle A_1, B_1\rangle \sqsubseteq \langle A_2, B_2\rangle$ if and only if $A_1 \subseteq A_2$ and $B_1 \subseteq B_2$.

Rectangle $\langle A_1, B_1\rangle$ is maximal, if there is no rectangle $\langle A_2, B_2\rangle$ such that $\langle A_1, B_1\rangle \sqsubseteq \langle A_2, B_2\rangle$.

Theorem 8 (Maximal rectangles as formal concepts)
$\mathcal{I} = \langle X, Y, I\rangle$ is a formal context. Pair $\langle A, B\rangle$ is a formal concept of $\mathcal{I}$ if and only if $\langle A, B\rangle$ is a maximal rectangle in $\mathcal{I}$.

Proof
Let $\langle A, B\rangle$ be the maximal rectangle. From the definition of $\uparrow$ we know that $\langle A, A^{\uparrow}\rangle$ is also a rectangle. From the definition of the maximal rectangle, we know that we cannot add anything to set A. So we have $A = A^{\uparrow\downarrow}$ and also $A^{\uparrow} = B$. □

2.2 Closure operators and Galois connections

In the previous section, we presented basic settings of FCA, but for purposes of our algorithms, we also have to define the closure operators and mathematical structures behind FCA.

Definition 9 (Closure operator)
The closure operator in a set X is mapping $C : 2^X \to 2^X$ which for each $A, A_1, A_2 \subseteq X$ holds

$$A \subseteq C(A) \tag{1}$$
$$A_1 \subseteq A_2 \Rightarrow C(A_1) \subseteq C(A_2) \tag{2}$$
$$C(A) = C(C(A)) \tag{3}$$

We will see that the concept-forming operators can be together used as closure operators, and they also form a representative case of Galois connections and their fixpoints.

Definition 10 (Galois connection)

A Galois connection between sets X and Y is a pair $\langle f, g\rangle$ where $f : 2^X \times 2^Y$ and $g : 2^Y \times 2^X$ holds for $A, A_1, A_1 \subseteq X$ and $B, B_1, B_2 \subseteq Y$:

$$A_1 \subseteq A_2 \Rightarrow f(A_2) \subseteq f(A_1) \tag{4}$$
$$B_1 \subseteq B_2 \Rightarrow g(B_2) \subseteq g(B_1) \tag{5}$$
$$A \subseteq g(f(A)) \tag{6}$$
$$B \subseteq f(g(B)) \tag{7}$$

Theorem 11 (Concept-forming operators as Galois connection)

For a formal context $\langle X, Y, I\rangle$, the pair of induced concept-forming operators $\langle\uparrow,\downarrow\rangle$ is a Galois connection between X and Y.

Proof

Let $\mathcal{I} = \langle X, Y, I\rangle$ be formal context and $\langle\uparrow,\downarrow\rangle$ its induced concept-forming operators. We will verify only that (4) and (6) holds, as the verification of (5),(7) can be obtained analogically.
Lets start with verification of (4) as

$$A_2^{\uparrow} = \{y \in Y | \forall x \in A_2 : \langle x, y\rangle \in I\} =$$

$$\{y \in Y | \forall x \in A_1 : \langle x, y\rangle \in I \bigwedge \forall x \in A_2 \setminus A_1 : \langle x, y\rangle \in I\} =$$

$$\{y \in Y | \forall x \in A_1 : \langle x, y\rangle \in I\} \bigcap \{y \in Y | \forall x \in A_2 \setminus A_1 : \langle x, y\rangle \in I\} =$$

$$A_1^{\uparrow} \bigcap \{y \in Y | \forall x \in A_2 \setminus A_1 : \langle x, y\rangle \in I\} \Rightarrow A_2^{\uparrow} \subseteq A_1^{\uparrow}.$$

Lets prove also (6). Let $A^{\uparrow} = B$ and thus

$$B^{\downarrow} = \{x \in X | \forall y \in B : \langle x, y\rangle \in I\} =$$

$$\{x \in A \vee x \in X \setminus A | \forall y \in B : \langle x, y\rangle \in I\ \} =$$

$$\{x \in A | \forall y \in B : \langle x, y\rangle \in I\ \} \bigcup \{x \in X \setminus A | \forall y \in B : \langle x, y\rangle \in I\ \} \Rightarrow A \subseteq A^{\uparrow\downarrow}.$$

□

Definition 12

Let $\langle f, g\rangle$ be Galois connection between sets X and Y. Then the set of pairs

$$fix(\langle f, g\rangle) = \{\langle A, B\rangle \in 2^X \times 2Y | f(A) = B, g(B) = A\}$$

is called a set of fixpoints of $\langle f, g\rangle$.

As we can see, the fixpoints of Galois connection between sets X and Y are also useful regarding formal concepts. The formal concepts are $fix(\langle\uparrow,\downarrow\rangle)$. Another interesting and for us important consequence of the previous definitions and lemma is the calculation of the fixpoints itself. The behavior of Galois

connections and their chaining allows us to effectively calculate the minimal formal concept for a given set of objects from X or a given set of attributes from Y. The chaining of the Galois connection will later help our algorithms to calculate the formal concepts.

Lemma 13 (Chaining of Galois connections)
Let X and Y be the sets and $\langle f, g\rangle$ the Galois connection between them. For any $A \subseteq X$ and $B \subseteq Y$ we have $f(A) = f(g(f(A)))$ and $g(B) = g(f(g(B)))$ respectively.

Proof
First we will prove $f(A) = f(g(f(A)))$. Lets start with the

$$f(A) \subseteq f(g(f(A))).$$

This equation follows directly from (7) by putting $B = f(A)$. Now we have to verify the second direction

$$f(A) \supseteq f(g(f(A))).$$

This follows from $A \subseteq g(f(A))$ by (6) and if we also apply the (4) we have $A \subseteq g(f(A)) \Rightarrow f(g(f(a))) \subseteq f(A)$ where $f(g(f(a))) \subseteq f(A)$ is the same as $f(A) \supseteq f(g(f(a)))$ which we wanted to prove.
We proved both directions so we have $f(A) = f(g(f(A)))$. We will omit the $g(B) = g(f(g(B)))$ as it is a dual problem and the proof would just need to use another equation from Galois connections definition. □

3 Boolean matrix decomposition

3.1 Matrix decomposition

In previous section we described the theory needed for boolean matrix decomposition. Now we will define a problem of matrix decomposition and after that, we will introduce the algorithms which are able to solve the problem. Content of this chapter was written with the help of [2].
For the formal definition of the matrix decomposition problem, we need to define matrix multiplication. Matrix multiplication is an operation performed on two matrixes A, B and we will mark it as $A \circ B$.

Definition 14 (Matrix multiplication ($A \circ B$))
Matrix product of matrix A and B is defined as

$$(A \circ B)_{i,j} = \bigvee_{l=1}^{k} A_{i,l} \wedge B_{l,j}$$

where $\bigvee$ is the maximum and $\wedge$ is a multiplication.
In the boolean case, the maximum is a logical disjunction, and multiplication is conjunction.

Example 15 ($A \circ B$)

$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \circ \begin{pmatrix} 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Now we can precisely define the problem of matrix decomposition.

Definition 16 (Matrix decomposition problem)
The boolean matrix decomposition for matrix I with n columns and m rows $(n \times m)$ is the problem of finding matrix A and B such that $A \circ B = I$. The size of matrix A is $n \times k$ and size of B is $k \times m$, and the dimension k is as small as possible.

To more specify the problem, it is not hard to find the matrix A and B, if we do not care about the dimension k. The crucial part is to find the smallest k possible. Currently, there is not an efficient algorithm that would be able to solve this problem. Every effective algorithm, which tries to solve this problem is not always able to find the best solution. If the input is complex, that means

that matrix I is large, we are not able to find the best solution or decide if there is no better solution.
To find the best solution for the input, we would need to start with $k = 1$, generate all possible matrixes A, B with corresponding size, and verify if any pair meets the requirement $A \circ B = I$. If none held the requirement, we would need to increment the parameter k and repeat until we find some A, B, which would hold the requirement. The problem is that for example for input $I_{100,100}$ and $k = 15$ there exist 5062500000000 possible solutions $(((100 \cdot 15)^2 \cdot (15 \cdot 100)^2))$. It is obvious that going through all the possible solutions is not effective, and it is not even possible for bigger instances with current computation power.
Most of the currently useful algorithms use the greedy approach. When the algorithm uses a greedy approach, it means that the algorithm builds the solution from scratch, and it decides what the best move from current information in the current state is. This approach could lead the algorithm to the "bad branch" of computation of the solution and end up with obtaining a sub-optimal solution.
The input for decomposition is matrix $M_{n,m}$ which we can also see as corresponding formal context $\langle X, Y, I \rangle$, where $|X| = n, |Y| = m$ and $M_{i,j} = 1$ if $\langle X_i, Y_j \rangle \in I$. For clarification, let us assume that sets X, Y are ordered, so every element in the set has an index. That means that our algorithm can use the theory of FCA to deal with decomposition. Now we will introduce the term coverage of the matrix. The matrix coverage is important to construct the algorithm and also for the performance measurement.

Definition 17 (Coverage)
Let $\mathcal{F}_1$ be the formal concept and $\mathcal{I} = \langle X, Y, I \rangle$ the formal context. Formal concept $\mathcal{F}_1 = \langle \{x_n\}, \{y_n\} \rangle$ where $x_n \in X$ a $y_n \in Y$ cover one element in formal context $\mathcal{I}$, if $\langle x_n, y_n \rangle \in I$ $(I_{x_n,y_n} = 1)$.

Theorem 18 (Coverage of matrix)
For every formal context $\mathcal{I} = \langle X, Y, I \rangle$ there exists the set of formal concepts $\mathcal{F}$ which cover all the ones in I.

Proof (Coverage of matrix)
For every $\langle x, y \rangle \in I$ we can construct the formal concept F_i, so $F_i = \langle \{x\}, \{y\} \rangle$. The set $\mathcal{F}$ of F_i will naturally cover the formal context $\mathcal{I}$ □

Let $\mathcal{F}$ be the set of the k formal concepts which covers the formal context $\mathcal{I}$. Corresponding matrix for $\mathcal{I}$ is I. Let us show that with formal concepts from $\mathcal{F}$ we are able to construct a $n \times k$ object–factor matrix A and $k \times m$ factor–attribute matrix B, which together hold $A \circ B = I$.

Definition 19 (Construction of matrix decomposition from a set $\mathcal{F}$)
Let $\mathcal{I} = \langle X, Y, I \rangle$ be the formal context and $\mathcal{F}$ set of formal concepts, which covers all entries in formal context $\mathcal{I}$. We can build matrix A and B from formal concept $F_i = \langle C_i, D_i \rangle \in \mathcal{F}$ so that $A_{ji} = 1$ if $j \in C_i$ else $A_{ji} = 0$ and $B_{ij} = 1$ if $j \in D_i$ else $B_{ij} = 0$. Then $(A \circ B) = I$.

In other words, we construct the matrix A from $\mathcal{F}$ by columns. Column i is represented by extent from $F_i \in \mathcal{F}$ and the element j of the column is 1 if object x_j is present in the extent of F_i.
Analogically it is for matrix B, but it is constructed from rows. Row i is represented by intent from F_i, and the element j is 1 if attribute y_j is present in the intent.

Example 20 (Construction of matrix decomposition from a set $\mathcal{F}$)

$$I_1 = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{array} \begin{array}{c} \begin{array}{ccccc} y_1 & y_2 & y_3 & y_4 & y_5 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

Let formal concepts be $\langle A_1, B_1 \rangle = \langle \{x_3, x_4\}, \{y_1, y_2, y_5\} \rangle$, $\langle A_2, B_2 \rangle = \langle \{x_2, x_6\}, \{y_1, y_4\} \rangle$, $\langle A_3, B_3 \rangle = \langle \{x_2, x_5\}, \{y_3, y_4\} \rangle$, $\langle A_4, B_4 \rangle = \langle \{x_1, x_6\}, \{y_2\} \rangle$,
So the set is $\mathcal{F} = \{\langle A_1, B_1 \rangle, \langle A_2, B_2 \rangle, \langle A_3, B_3 \rangle, \langle A_4, B_4 \rangle\}$.
The final solution is $I = A_{\mathcal{F}} \circ B_{\mathcal{F}}$

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \circ \begin{pmatrix} 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

We can also see that one place in the matrix could be covered by more than one concept.
Another approach is that each formal concept from I ($n \times m$) could be interpreted as $n \times m$ matrix with only the values which particular formal concept covers and then I is a union of these rectangles. By union, it is meant the $\vee$-superposition of the matrices made from formal concepts. Let us show this idea in the previous example.

For formal concepts F_1, F_2, F_3, F_4 and formal context I:

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \vee \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \end{pmatrix} \vee$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \vee \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

We can see that the union of these four matrices forms the initial formal context I.
This approach is also an important assumption that allows us to reduce the problem of matrix decomposition to the problem of finding the set of factors that together covers the initial matrix.

3.2 Grecond algorithm

Grecond is the algorithm that solves the matrix decomposition problem by greedy approach, but generally with high-quality output (close to optimum solution). It covers the input matrix with factors, and then, it outputs the factors, from which we can construct the matrix A and B with ease. The dimension k of the $A \in \{0,1\}^{n \times k}$ and $B \in \{0,1\}^{k \times m}$ will be the same as the number of factors which the algorithm finds. Let us note that the algorithm uses the fact that operators $\uparrow, \downarrow$ together form Galois connection, and it uses chaining of Galois connections for computing of formal concepts (their extents and intents). The algorithm tries to find as few factors as possible, but as we mentioned before, it does not always return the optimal solution.
The principle of the algorithms is to cover each table entry with 1 in the input matrix I with at least one factor. The algorithm starts with an empty set of factors $\mathcal{F}$. It stops when the $\mathcal{F}$ covers every 1 in the input table (element).
It starts with big factors that cover many entries in the I, and then it continues with smaller factors to the factors, which covers, for example, just one element. As it may seem like Grecond calculates all the factors and selects the most valuable one at the moment, it is not true. Grecond builds each formal concept from $\mathcal{F}$ incrementally from scratch. It starts with an empty extent and adds the most promising attribute to it, while it also enriches the current formal concept by using the concept-forming operators. Then it selects the next attribute, which will be added to the extent of a factor concerning the maximization of the current factor cover of uncovered elements of I. If there does not exist the attribute which would increase the cover of the current factor, it adds the current factor to the $\mathcal{F}$ and starts again with an empty extent.

Algorithm 1 Grecond

procedure GRECOND(I) ▷ (Boolean matrix)
  set $U = \{\langle i, j \rangle | I_{i,j} = 1\}$
  set $\mathcal{F} = \emptyset$
  while $U \neq \emptyset$ **do**
    set $D = \emptyset$
    set $V = 0$
    while exist $j \notin D$, for which $*|D \odot j| > V$ **do**
      Select $j \notin D$, which maximizes $D \odot j$:
        set $D = (D \cup \{j\})^{\downarrow\uparrow}$
        set $V = |(D^{\downarrow} \times D) \cap U|$
    set $C = D^{\downarrow}$
    add $\langle C, D \rangle$ **to** $\mathcal{F}$
    for each $\langle i, j \rangle \in C \times D$ **do**
      delete $\langle i, j \rangle$ **from** U
  return $\mathcal{F}$
▷ $*D \odot j = ((D \cup \{j\})^{\downarrow} \times (D \cup \{j\})^{\downarrow\uparrow}) \cap U$

Example 21 (Grecond algorithm output example)
Let I be the input matrix.

$$I = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

Let us briefly explain the operation of the algorithm. Outer while loop is searching for attribute which maximizes actual formal concept $\langle D, D^{\downarrow} \rangle$ so it covers as much uncovered elements in U. U is auxiliary set which tracks the uncovered elements. These are the elements, which are not covered by any yet calculated formal concepts from $\mathcal{F}$. The first output factor is $F_1 = \langle \{1, 2\}, \{2, 3\} \rangle$ which covers 4 uncovered elements. These elements are removed from set U. Next factor is $F_2 = \langle \{0, 1\}, \{0\} \rangle$. It covers 2 uncovered elements, again we need to delete them from U. The last factor is $F_3 = \langle \{0, 2\}, \{1\} \rangle$. The auxiliary set U is empty after the update, that means that we have covered all elements in input matrix I and algorithm returns $\mathcal{F} = F_1, F_2, F_3$.
From $\mathcal{F}$ we can easily construct the matrix A and B which holds $A \circ B = I$. The final output is as below.

$$\begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \circ \begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

The algorithm described in this section is limited to the boolean data input. That means that if we have data with more than one value for the relationship between objects and attributes, we can not use Grecond. An example of the data with such many-valued relations could be data with students and their grades from particular subjects. If we wanted to use this algorithm on this dataset, we would need to transform the data. Let us show the simple principle which could be used for data transformation.

Example 22 (Ordinal data matrix transformation)
Let I_1 be the matrix with data about three students and their grades from physics and math.

$$I_x = \begin{array}{c} \\ S_1 \\ S_2 \\ S_3 \end{array} \begin{array}{c} \begin{array}{cc} M & P \end{array} \\ \begin{pmatrix} 2 & 3 \\ 3 & 4 \\ 1 & 1 \end{pmatrix} \end{array}$$

Where S_i is a student, M means math, and P means physics. Grade 1 is the best; 5 is the worst.

To transform I_x to such a boolean matrix, which will represent the same data, we need to replace a set of attributes with a new set of attributes. For every attribute with more than one possible value for object-attribute relation,

we will create new attributes. The number of new attributes will be the number of possible values. Then, for a particular object and attribute, we will create the new relation only with the newly created attribute, which corresponds with the value which the object had. The boolean matrix I created from I_x is as follows.

$$I = \begin{array}{c} \\ S_1 \\ S_2 \\ S_3 \end{array} \begin{array}{c} \begin{array}{cccccccccc} M_1 & M_2 & M_3 & M_4 & M_5 & P_1 & P_2 & P_3 & P_4 & P_5 \end{array} \\ \left(\begin{array}{cccccccccc} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{array} \right) \end{array}$$

When a student S_x has the attribute M_i it means, that the student has the grade i from math. Analogically for physics.

As we can observe, the number of objects grows very fast if we want to transform the input into the boolean case. The transformation can also cause the data to be far less readable. In the next chapters, we will introduce fuzzy logic fundaments and use these approaches to extend our algorithm so it will be able to handle the ordinal data as input.

4 Preliminaries from fuzzy logic

This chapter is a summary of the underlying theory of fuzzy logic. It consists of definitions and examples which are essential for the extension of the Grecond algorithm to be able to handle ordinal data. Content of this chapter was written with the help of [3].

4.1 Introduction to fuzzy logic

Fuzzy logic is a type of many-valued logic, which could handle the case where some statement is not fully true, but not false. It is inspired by the observation from the real world as there is a gradual transition between true and false statements, which is not sharp. For example, we can have a set of "long books", and one of them has 400 pages. The question is that if we have another book with 390 or even 399 pages, will it also be a member of the "long books"? As we can see in the real world, there is often not clearly defined the border between being a member of the collection and not being a member of a collection. In classical mathematics, there is always defined the edge point or restrictions for the membership in the set. As in the real world many terms and states are not strict; fuzzy logic can be a better choice for the modeling of the real world situation. That is the reason why fuzzy logic found its way to many areas in the real world, for example, engineering. It is widely used for temperature regulation where it can define the state "being cold" or "being hot" in a more natural way than classical mathematics.
In fuzzy logic, the truth value could be any real number from interval $[0, 1]$. Boolean logic is the special case of fuzzy logic, where statements could have the truth value only 1 or 0. In this thesis, we will not use the whole continuous interval $[0, 1]$, but we will focus on finite sets of truth values from it. Most of the time, we will label the set of truth values as L. For example we will use a few elements scale as $L = \{0, 0.25, 0.5, 0.75, 1\}$. In particular, people could attach linguistic labels to these value such as "not true", "maybe", "middle", "probably" and "fully true".

Let us start with the formal setting of a fuzzy structure. In the next chapter, we will then define the problem of matrix decomposition for the matrix, which entries will be more complex than just two values.

4.2 Fuzzy structures

Graded truth approach directly leads to the assumption that the set L of truth values is partially ordered. We denote this by $\langle L, \leq \rangle$. The complete definition of the fuzzy set is as follows.

Definition 23 (Fuzzy set)
Let $\langle L, \leq \rangle$ be a partially ordered set with the least element 0 and the greatest element 1. A fuzzy set (L-set) in a universe X is a mapping

$$A : X \to L$$

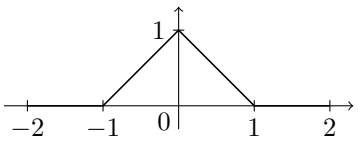
A is the mapping of X to A.

Elements from$L(a \in L)$ are called truth degrees. Truth degree $L(x)$ expresses the degree of membership of element $x \in X$ in A. Mapping A is called L-set in X and we can read this it as "fuzzy set" in X. We can denote the L-set in X also by $\{{}^{A(x)}/_{x} | x \in A\}$. For elements x, y, $A(x) \leq A(y)$ means, that y belongs to A at least with the same degree as x.

Example 24 (Fuzzy set)
Let L-set be defined as

$$A(x) = \begin{cases} -x - 1 & \text{if } -1 \leq x \leq 0 \\ 1 - x & \text{if } 0 < x \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

Then A is a fuzzy set that represents the concept “approximately 0”. The following graph visualizes A.



From the graph, we can see that the number x will have a higher membership value $A(x)$ the nearer it will be to 0, and it will be the full truth if the $x = 0$.

We have defined the L-set, and now we can define the residuated lattice, which plays a fundamental role in fuzzy logic. Residuated lattice will equip the L-set with operations and also restrict the behavior of these operations.

Definition 25 (Residuated lattice)
A residuated lattice is an algebra $\mathbf{L} = \langle L, \vee, \wedge, \otimes, \rightarrow, 0, 1\rangle$ such that L is partially ordered set of truth degrees, $\langle L, \vee, \wedge, 0, 1\rangle$ is a lattice with the least and greatest element, $\langle L, \otimes\rangle$ is a commutative monoid and $\otimes, \rightarrow$ satisfy adjointness:

$$a \otimes b \leq c \text{ iff } a \leq b \rightarrow c$$

Let us clarify the definition of residuated lattice. $\otimes$ is the truth function of many-valued conjunction, we can also call it multiplication. $\rightarrow$ is the truth function of many-valued implication and it is called residuum. The $\vee$ and $\wedge$ are infimum and supremum. 0 and 1 is the least and greatest element of L. Commutative monoid means that $\otimes$ is commutative, associative and $a \otimes 1 = 1 \otimes a = a$ is satisfied for each $a \in L$. Adjointness is satisfied, if for all elements $a, b, c \in L$ $a \otimes b \leq c$ hold if and only if $a \leq b \rightarrow c$. When two operations satisfy the adnjointness, we can call them an adjoint couple.
Two elements boolean algebra $\langle \{0, 1\}, \vee, \wedge, \otimes, \rightarrow, 0, 1\rangle$ is therefore a special case of residuated lattice.

Adjoint couples $\otimes, \rightarrow$ on $[0, 1]$ have many forms. For $a, b \in L$ we call the operation $a \rightarrow$ b the residuum of b by a. For each operation $\otimes$ there exist at most one $\rightarrow$ which forms the adjoint couple with it. That means that for each $\otimes$ the residuum $\rightarrow$ is uniquely defined. Let us mention a few most common pairs of operation which forms a complete residuated lattice $\mathbf{L} = \langle L, \vee, \wedge, \otimes, \rightarrow, 0, 1\rangle$.

Definition 26 (Łukasiewicz structure)

$$a \otimes b = \max(a + b - 1, 0)$$
$$a \rightarrow b = \min(1 - a + b, 1)$$

Definition 27 (Gödel structure)

$$a \otimes b = \min(a, b)$$
$$a \rightarrow b = \begin{cases} 1 \text{ if } a \leq b \\ 0 \text{ otherwise} \end{cases}$$

Definition 28 (Product structure)

$$a \otimes b = a \cdot b$$
$$a \rightarrow b = \begin{cases} 1 \text{ if } a \leq b \\ b/a \text{ otherwise} \end{cases}$$

The corresponding algebras are called standard product algebra, standard Łukasiewicz algebra, and standard Gödel algebra. We will use only the Łukasiewicz

algebra for our purposes. Let us just remind that the pair of operation $\langle +, - \rangle$ is not associative, so when we are calculating the values of the $\otimes$ or $\rightarrow$, the exact order of operations needs to be preserved.
Use of another pair of operations such as Gödel in the matrices decomposition could be the area of further research.

From now we will assume that the residuated lattice used in all examples will be $\mathbf{L} = \langle L, \otimes, \rightarrow, \wedge, \vee, 0, 1 \rangle$, where $\rightarrow$ and $\otimes$ are the Łukasiewicz operations. The set of truth values L will be different within the various examples so that it will be specified for each example separately.

4.3 Binary fuzzy relations

For a lot of various problems, binary relations turn out to be a useful and straightforward tool. However, binary relation with just boolean elements is often not descriptive enough. For example, if we have a relation between students and school subjects or a particular test, it is natural that grades themselves represent the relation. As an example, let $\{1, 2, 3\}$ be the set of grades. The students can have one grade for each subject or test. We can transform the rich relation between grades and students to just boolean relation by some kind of transformation, but it could end up into information loss or making the relation less readable. If we want to avoid these problems, we can manipulate with the grades themselves by assigning each of them the value from the fuzzy set. Let us define the binary fuzzy relation.

Definition 29 (Binary fuzzy relation)
Fuzzy relation (**L**-relation) between nonempty sets X and Y is any mapping $R : X \times Y \rightarrow L$, where L is the support set of the complete residuated lattice $\mathbf{L}$.

For $x \in X$ and $y \in Y$, the $R(x, y) \in L$ is the truth degree to which x and y are in the relation R. For the students and subjects, we can work with the grades as with corresponding truth degrees from L. Let $L = \{0.0, 0.25, 0.5, 0.75, 1.0\}$ be the set of truth degrees. The grade 1 could be represented by truth degree 1.0, while the degree 2 would be represented by 0.75 et cetera.

We defined everything which will be needed to step deeper into the ordinal data decomposition problem. In the next chapter, we will continue with the FCA enriched with fuzzy logic.

5 Formal concept analysis of ordinal data

This chapter is an introduction to the basic theory of FCA extended with fuzzy logic, which will later allow us to analyze ordinal data. Content of this chapter was written with the help of [4] and [5].

5.1 Fuzzy contexts

The form of elementary knowledge about a given domain of interest is as in the boolean case, the triple consisting of a collection of objects, collection of attributes, and a relation between objects and attributes. The only difference is that in the boolean case, the relation between objects and attributes is strictly sharp, which means that the particular object has the attribute or does not have it. In fuzzy logic, the relation between objects and attributes can have more degrees of relationship between full truth and false. This corresponds with real world concepts, where the relation is not that strict most of the time. When we describe the real world, we often use vague terms to define the attributes of the objects. For example, the empirical attribute for cities, the size of the city, is not strictly defined. The cities with around one million citizens or more are usually recognized as big cities, and the closer the number of citizens is to one million, the more is the city perceived as big. The vague definition of various attributes of objects leads to the idea that the attribute applies to a given object to a certain degree (truth degree). As we can easily see, fuzzy logic seems to be helpful for the formalization of real world relations.
We can continue with the formulation of formal context enriched with the fuzzy relation. The data from the triplet are naturally depicted in the matrix, where rows correspond to objects, columns to attributes, and the matrix entries specify the degree to which the objects have the attributes. Let us formalize this in more mathematical terms as follows: We suppose that there is a nonempty set X (elements of the X are called objects) and the nonempty set Y (elements of Y are called attributes). We choose a truth degree structure L, which provides an appropriate scale of truth values with its structure, and lastly, there is a binary fuzzy relation (L-relation) between objects and attributes. The formal definition of a fuzzy context is as follows.

Definition 30 (Fuzzy context)
The triplet $\langle X, Y, I\rangle$, where $X \neq \emptyset$, $Y \neq \emptyset$ and $I : X \times Y \to L$ is the binary fuzzy relation (L-relation).

For clarification of fuzzy context definition, the L is the set of partially ordered truth values. For each element $a \in L$ applies that $a \in \langle 0, 1\rangle$. We will call the elements from L the truth degrees; hence the entry in the table which represents the fuzzy context is the truth degree, in which attribute $y \in Y$ applies to object $x \in X$. Let us show an easy interpretable fuzzy context as an example.

Example 31 (Fuzzy context)

Let $X = \{s_1, s_2, s_3, s_4, s_5, s_6\}$ be the set of students and set $Y = \{math, english, german, physics, biology\}$ the set of subjects, in which were the students evaluated. The set of truth degrees is $L = \{0, 0.25, 0.5, 0.75, 1\}$. The binary fuzzy relation I is represented by following matrix.

$$I = \begin{array}{c} \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \end{array} \begin{array}{c} \begin{array}{ccccc} math & english & german & physics & biology \end{array} \\ \begin{pmatrix} 1 & 0.75 & 0.75 & 1 & 1 \\ 0.25 & 0.5 & 0.5 & 0.25 & 0.5 \\ 0.75 & 0.55 & 0.5 & 0.75 & 0.5 \\ 1 & 1 & 1 & 1 & 1 \\ 0.75 & 0.5 & 0.5 & 0.5 & 0.75 \\ 0.5 & 1 & 1 & 0.5 & 0.5 \end{pmatrix} \end{array}$$

The possible matrix values are $\{0, 0.25, 0.5, 0.75, 1\}$ and these values correspond to the grades $\{5, 4, 3, 2, 1\}$ which are commonly used to evaluate students' efforts. Matrix entry represents the grade which a student obtained in a particular subject. For example s_3 has the grade 4 from math, 3 from english, 3 from german, 4 from physics and 3 from biology.

Let us continue with the definition of concept forming operators, which we defined in the second chapter for formal context with boolean values yet, but now we will extend the definition for fuzzy formal context.

5.2 Concept forming operators

Each fuzzy context $I = \langle X, Y, I \rangle$ induces a pair of operators $\uparrow, \downarrow$, we call them concept-forming operators. As these operations are induced by the I, we should use $\uparrow_I, \downarrow_I$, but will omit the bottom label if the parent context is clear. Operator $\uparrow$ assigns a fuzzy set of common attributes to a given fuzzy set of objects. Analogically the $\downarrow$ for a fuzzy set of attributes assigns a fuzzy set of objects which share them. Operators $\uparrow, \downarrow$ may be thought of as mappings $\uparrow$: $L^X \to L^Y$, $\downarrow$: $L^Y \to L^X$. The value $A^\uparrow(y)$ for attribute $y \in Y$ and $A \subseteq X$ is the truth degree in which y belongs to $A^\uparrow$ and the value follows from the proposition "for each $x \in A$, x has y in at degree at least $A^\uparrow(y)$". Conversely the $B^\downarrow(x)$ for $B \subseteq Y$ and $x \in X$ is the truth degree to which each attribute of B is shared by the object x. Let us show the precise definition of these operators.

Definition 32 (Concept forming operators)
Let $I = \langle X, Y, I \rangle$ be the fuzzy context. The induced operator $\uparrow_I$ is defined as

$$A^{\uparrow_I}(y) = \bigwedge A(x) \to I(x, y)$$

where $A \subseteq X$, $y \in Y$ and $\bigwedge$ is the infimum.

Conversely the induced operator $\downarrow_I$ is defined as

$$B^{\downarrow_I}(x) = \bigwedge B(x) \to I(x, y)$$

where $B \subseteq Y$, $x \in X$ and $\bigwedge$ is the infimum.

As the calculation of sets $A^{\uparrow}$ and $B^{\downarrow}$ could not be entirely clear from the definitions of operators, let us continue with a short example of its usage.

Example 33 (Concept forming operators)
Let us use our fuzzy context from the previous example. So $X = \{s_1, s_2, s_3, s_4, s_5, s_6\}$, $Y = \{math, english, german, physics, biology\}$ and

$$I = \begin{array}{c} \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \end{array} \begin{array}{c} \begin{array}{ccccc} math & english & german & physics & biology \end{array} \\ \left(\begin{array}{ccccc} 1 & 0.75 & 0.75 & 1 & 1 \\ 0.25 & 0.5 & 0.5 & 0.25 & 0.5 \\ 0.75 & 0.75 & 0.5 & 0.75 & 0.5 \\ 1 & 1 & 1 & 1 & 1 \\ 0.75 & 0.5 & 0.5 & 0.5 & 0.75 \\ 0.5 & 1 & 1 & 0.5 & 0.5 \end{array} \right) \end{array}$$

Just to remind, the set of truth degrees is $L = \{0, 0.25, 0.5, 0.75, 1\}$ and we use Łukasiewicz operations $\to, \otimes$.
$A = \langle 1, 0, 0, 0.75, 1, 0 \rangle$ is the fuzzy set of students. The students s_1 and s_5 fully belong to the set A, the student s_4 belongs to A in degree 0.75 and the other students do not belong to the set A at all. Let us calculate the fuzzy set $A^{\uparrow}$.

$$
\begin{aligned}
A^{\uparrow}(math) = & \bigwedge_{s \in A} A(s) \to I(s, math) \\
= & (A(s_1) \to I(s_1, math)) \wedge (A(s_2) \to I(s_2, math)) \\
& \wedge (A(s_3) \to I(s_3, math)) \wedge (A(s_4) \to I(s_4, math)) \\
& \wedge (A(s_5) \to I(s_5, math)) \wedge (A(s_6) \to I(s_6, math)) \\
= & (1 \to 1) \wedge (0 \to 0.25) \wedge (0 \to 0.75) \wedge (0.75 \to 1) \wedge (1 \to 0.75) \\
& \wedge (0 \to 0.5) = 1 \wedge 1 \wedge 1 \wedge 1 \wedge 0.75 \wedge 1 = 0.75 \\
A^{\uparrow}(english) = & \bigwedge_{s \in A} A(s) \to I(s, english) \\
= & (1 \to 0.75) \wedge (0 \to 0.5) \wedge (0 \to 0.75) \wedge (0.75 \to 1) \wedge (1 \to 0.5) \\
& \wedge (0 \to 1) = 0.75 \wedge 1 \wedge 1 \wedge 1 \wedge 0.5 \wedge 1 = 0.5 \\
A^{\uparrow}(german) = & (1 \to 0.75) \wedge (0 \to 0.5) \wedge (0 \to 0.5) \wedge (0.75 \to 1) \wedge (1 \to 0.5) \\
& \wedge (0 \to 1) = 0.75 \wedge 1 \wedge 1 \wedge 1 \wedge 0.5 \wedge 1 = 0.5 \\
A^{\uparrow}(physics) = & (1 \to 1) \wedge (0 \to 0.25) \wedge (0 \to 0.75) \wedge (0.75 \to 1) \wedge (1 \to 0.5) \\
& \wedge (0 \to 0.5) = 1 \wedge 1 \wedge 1 \wedge 1 \wedge 0.5 \wedge 1 = 0.5 \\
A^{\uparrow}(biology) = & (1 \to 1) \wedge (0 \to 0.5) \wedge (0 \to 0.5) \wedge (0.75 \to 1) \wedge (1 \to 0.75) \\
& \wedge (0 \to 0.5) = 1 \wedge 1 \wedge 1 \wedge 1 \wedge 0.75 \wedge 1 = 0.75
\end{aligned}
$$

So our final fuzzy set $A^{\uparrow}$ is $A^{\uparrow} = \langle 0.75, 0.5, 0.5, 0.5, 0.75 \rangle$. If we have considered to not include the student s_5 to our set A, the truth degree in which most attributes belong to $A^{\uparrow}$ would be higher, as the set A would consist of only "great students".
Now we use the operator $\downarrow$ to calculate the fuzzy set of objects for a fuzzy set of attributes. As the set of attributes B we will use our set $A^{\uparrow} = \langle 0.75, 0.5, 0.5, 0.5, 0.75 \rangle$.

$$
\begin{aligned}
B^{\downarrow}(s1) = & \bigwedge_{a \in B} B(a) \to I(a, s1) \\
= & \ (B(math) \to I(math, s_1)) \wedge (B(english) \to I(english, s_1)) \\
& \wedge (B(german) \to I(german, s_1)) \wedge (B(physics) \to I(physics, s_1)) \\
& \wedge (B(biology) \to I(biology, s_1)) \\
= & \ (0.75 \to 1) \wedge (0.5 \to 0.75) \wedge (0.5 \to 0.75) \wedge (0.5 \to 1) \\
& \wedge (0.75 \to 1) = 1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 = 1 \\
B^{\downarrow}(s2) = & \bigwedge_{a \in B} B(a) \to I(a, s2) \\
= & \ (0.75 \to 0.25) \wedge (0.5 \to 0.5) \wedge (0.5 \to 0.5) \wedge (0.5 \to 0.25) \\
& \wedge (0.75 \to 0.5) = 0.5 \wedge 1 \wedge 1 \wedge 0.75 \wedge 0.75 = 0.5 \\
B^{\downarrow}(s3) = & \ (0.75 \to 0.75) \wedge (0.5 \to 0.75) \wedge (0.5 \to 0.5) \wedge (0.5 \to 0.75) \\
& \wedge (0.75 \to 0.5) = 1 \wedge 1 \wedge 1 \wedge 1 \wedge 0.75 = 0.75 \\
B^{\downarrow}(s4) = & \ (0.75 \to 1) \wedge (0.5 \to 1) \wedge (0.5 \to 1) \wedge (0.5 \to 1) \\
& \wedge (0.75 \to 1) = 1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 = 1 \\
B^{\downarrow}(s5) = & \ (0.75 \to 0.75) \wedge (0.5 \to 0.5) \wedge (0.5 \to 0.5) \wedge (0.5 \to 0.5) \\
& \wedge (0.75 \to 0.75) = 1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 = 1 \\
B^{\downarrow}(s6) = & \ (0.75 \to 0.5) \wedge (0.5 \to 1) \wedge (0.5 \to 1) \wedge (0.5 \to 0.5) \\
& \wedge (0.75 \to 0.5) = 0.75 \wedge 1 \wedge 1 \wedge 1 \wedge 0.75 = 0.75
\end{aligned}
$$

The fuzzy set $B^{\downarrow} = \langle 1, 0.5, 0.75, 1, 1, 0.75 \rangle$.
Furthermore, the $A^{\uparrow}(x)$ is the truth degree to which each student of A is successful in a subject (course) x and the $B^{\downarrow}(s)$ is the truth degree to which each subject (course) of B is handled by student s.

As we can see, concept forming operators for fuzzy context need a bit more computation than the calculation for boolean formal context, which we described in the second chapter.

5.3 Fuzzy concepts

Now when we have defined the fuzzy context and concept forming operators, we can step forward and move closer to our main aim, the formal concepts in fuzzy context. The main idea is the same as for formal context, which we defined for the boolean context, but the intent and extent are a little more complicated. Previously, the extents and intents were just classic sets, which have or have not the particular element. In this case, the extents and intents are fuzzy sets, which means that every element has to belong to the set in a particular truth degree. It also means that every object from the initial fuzzy context will belong to each intent in some truth degree, and also every attribute will belong to each extent

in some truth degree. For clarification, when the truth degree of some element in the fuzzy set is 0, it means that the element does not belong to the set, it is the analogy to boolean set when the element is not present in the set. We will call the formal concept in a fuzzy context the fuzzy concepts.

Definition 34 (Fuzzy concept)
A pair $\langle A, B\rangle$ in fuzzy context $\langle X, Y, I\rangle$ where $A \in L^X$, $B \in L^Y$ such that

$$A^{\uparrow} = B \text{ and } B^{\downarrow} = A$$

is fuzzy concept in I.

The members of pair $\langle A, B\rangle$ are called extent and intent.

Example 35 (Fuzzy concept)
Let us use the same fuzzy context as in the example of concept forming operators and let us also use the calculated fuzzy sets B and $B^{\downarrow}$.
Let $\langle E, F\rangle$ be the potential L-concept, where the $E = \langle 1, 0.5, 0.75, 1, 1, 0.75\rangle$ and $F = \langle 0.75, 0.5, 0.5, 0.5, 0.75\rangle$. Our fuzzy set E is the previously calculated fuzzy set $B^{\downarrow}$ and the set F is the previously calculated B which was calculated as $A^{\uparrow}$. We need to verify $E^{\uparrow} = F$ and $F^{\downarrow} = E$.
We can easily see that the second equation directly follows from the definition of our initial sets F and E, because we put $F = B$ and $E = B^{\downarrow}$. So the equation $F^{\downarrow} = E$ holds.
The equation $E^{\downarrow} = F$ also holds, but it is not so obvious at the first sight. We can manually verify it, but we can also use the theory from the first chapter, mainly the observation that the usage of both context-forming operators is closure operator. We know that the $E = B^{\downarrow}$ and we also know that we calculated the set B as $A^{\uparrow}$ so $E = B^{\downarrow} = A^{\uparrow\downarrow}$ and thus $E^{\uparrow} = B^{\downarrow\uparrow} = A^{\uparrow\downarrow\uparrow}$. From the fact that operators $\uparrow, \downarrow$ form Galois connections and from the chaining of Galois connections we have $A^{\uparrow\downarrow\uparrow} = A^{\uparrow}$ so we have $E^{\uparrow} = A^{\uparrow} = B = F$. Therefore we verified that pair $\langle E, F\rangle$ is a fuzzy concept.

The role of the fuzzy concepts for our purposes is that they could be helpful when we are describing the data. They are considered as new variables of the context itself, which are hidden but can be more fundamental than the original object and attributes variables. So now, when we have defined the fuzzy context and also the idea of a fuzzy concept, we can extend our main problem of optimal boolean matrix decomposition to optimal Lmatrix decomposition.

5.4 Optimal decomposition

The problem we considered may be formulated in terms of matrices or equivalently in terms of relation. We proceed for matrices, which framework is commonly used for this problem. We talked about the optimal boolean matrix decomposition in the 3rd chapter, so let us talk about crucial differences. Let L denote a partially ordered set of grades bounded by 0 and 1 and matrix $I \in L^{n \times m}$, which we would like to decompose into matrix A and B. The main difference between the optimal decomposition we defined before will be that now we will consider the decomposition of matrice I, which has more complex ordinal data entries instead of just boolean entries. Namely, each entry $I_{i,j}$ of I will represent the grade in which the particular object i has the attribute j. Thus matrix I could represent fuzzy relation of some fuzzy context $\langle X, Y, I \rangle$. To precisely define the problem of fuzzy matrix decomposition, we first need to extend the matrix multiplication by the fuzzy operations. This matrix multiplication operation is also known as a sup-t-norm-product.

Definition 36 (Sup-t-norm-product ($A \circ B$))
Let the $\langle L, \vee, \wedge, \otimes, \rightarrow, 0, 1 \rangle$ be the complete residuated lattice. Let A and B be the matrices with entries from L. The sup-t-norm-product $\circ$ is defined by

$$(A \circ B)_{i,j} = \bigvee_{l=1}^{k} A_{i,l} \otimes B_{l,j}$$

The sup-t-norm-product could have many forms, depending on the operations, which we choose as $\otimes$ and $\rightarrow$. The general example could be the definitions of the boolean matrix multiplication, which is a special case of sup-t-norm-product and it is defined as $(A \circ B)_{i,j} = max_l^k(min(A_{i,l}, B_{l,j}))$. Another version of the $\circ$ could also be obtained by using another fuzzy structure for the operations $\otimes$ and $\rightarrow$. We will use just the Łukasiewicz structure for our calculations, algorithms, and examples. The usage of another structure in decomposition algorithms could be the area of further research. Let us recall, that in Łukasiewicz structure we have $a \otimes b = max(a + b - 1, 0)$ and $a \rightarrow b = min(2 - a + b, 1)$. The precise define the fuzzy context decomposition problem is as follows.

Definition 37 (Fuzzy context decomposition problem)
The fuzzy context decomposition for context $\mathcal{I} = \langle X, Y, I \rangle$ represented by matrix I with $|X|$ columns and $|Y|$ rows $(n \times m)$ is the problem of finding matrix A and B such that $A \circ B = I$. The size of matrix A is $|X| \times k$ and size of B is $k \times |Y|$. Our aim is to have the dimension k as small as possible.

Let us denote that as the I is the matrix which represents the binary fuzzy relations, its entries are from some $L = \langle 0, 1 \rangle$. This also applies to matrix A and B. As we can see, the problem of binary matrix decomposition was the special case where $L = \{0, 1\}$. We can also transform the matrix I which represents the

fuzzy relation to the boolean matrix $I^{\{0,1\}}$ which will represent the same data, but in general the matrix $I^{\{0,1\}}$ will be much bigger.
The transformation for I from $\mathcal{I} = \langle X, Y, I \rangle$ where $I_{x,y} \in \{0, 0.25, 0.5, 0.75, 1\}$ could be defined as follows. For each $y \in Y$ we will create attributes $\{y_0, y_{0.25}, y_{0.5}, y_{0.75}, y_1\}$ and we will construct $I^{0,1}$ as

$$I^{0,1}_{x,y_i} = \begin{cases} 1 \text{ if } I_{x,y} <= i \\ 0 \text{ otherwise} \end{cases}$$

In other words, for each attribute, we will create attributes that will represent all the degrees in which the object could have a particular attribute.
This transformation was mentioned just to extend the viewpoint on the decomposition problem. A comparison of the outputs from fuzzy decomposition and boolean decomposition of the transformed matrix could be an area of further research.

6 Algorithm for the decomposition of a matrix with ordinal data

6.1 General setting

In this chapter, we will introduce the extended version of the Grecond algorithm, which we talked about in the chapter about the boolean matrix decomposition. Content of this chapter was written with the help of [6]. Let us briefly remind how Grecond algorithm works.
Grecond was designed to take the boolean matrix $I^{0,1}$ as input. The matrix $I^{0,1}$ could represent the relation from formal context $\langle X, Y, I^{0,1}\rangle$. Grecond incrementally covers the input matrix $I^{0,1}$ with the formal concepts. It starts with a big formal concept and continues with the smaller formal concepts which appear in input data. Grecond uses the operators $\uparrow, \downarrow$ for the calculation of formal concepts and it also uses the fact that $\uparrow, \downarrow$ together form Galois connection, so it also uses the properties of Galois connections, namely the Chaining of Galois connections for the extent and intent computation.
With the help of fuzzy logic, we will extend the Grecond algorithm so it will be able to handle the real-valued input. To be specific, the input will be real-valued matrix I^L, which stands for the fuzzy relation from fuzzy context $\langle X, Y, I\rangle$ and L is a set of truth degrees.

6.2 Basic algorithm

The input for the algorithm will be the fuzzy context $\langle X, Y, I\rangle$. The X is the set of objects, Y is a set of attributes, and I is the matrix which represents the Lrelation between objects and attributes where $L \in \langle 0, 1\rangle$ is the partially ordered set of truth values.
The main idea of the algorithm is still the same as with the boolean version. Algorithms start with the empty set $\mathcal{F}$ of formal concepts, and then it systematically builds the $\mathcal{F}$ until the set of uncovered matrix entries I_{ij} is empty. There are a few obstructions that we need to solve to obtain the extended version of Grecond, which we will call FuzzyGrecond or L-Grecond.
The first issue is about the formal concept itself. In the classic version, the intents and extents are just sets that contain particular elements. In the L-Grecond, we can not use the framework of the standard set to model the formal concepts. We need to use fuzzy concept, which use fuzzy sets as intents and extents. This, in general, means that intent and extents contain every element in some truth degree, at least in degree 0.
The second difference which needs to be solved is the calculation of promising factors. In the classic version, we choose the next factor regrading to some set of promising attributes from Y. Now we have to deal also with the degree of membership of the attributes in the promising set. The issue is that the formal concept we calculate from some extent, which is a fuzzy set of attributes where

every attribute has the membership degree equal to 1, could have the final coverage value far worse than the same fuzzy set in which some of the attributes will have the membership degree lowered to 0.9 or 0.75. This is caused by the fact that objects $x \in X$, which have not the attribute in at least the same degree as the initial extent, are not included in the fuzzy concept at all. To solve this, we have to include also the membership degree to the calculation of new concepts. This is solved by going through all elements $a \in L$ when calculating the next fuzzy concept.

The last issue and also the biggest one is the selection of the next fuzzy concept, which has to be added to the final solution. In the boolean version, the algorithm selects the next formal concept, which fully covers the most significant amount of uncovered 1 entries in the input matrix. For simplification, let us name the number of covered entries by the concept, the coverage score. In the fuzzy version, the exact and sounds calculation of coverage itself is hard to obtain. We describe the problematic situation as follows. Let us have some fuzzy concept which almost covers all matrix entries in I, but none of them entirely. Namely, the particular entry $I_{ij} = 0.9$, but our concept covers just 0.8. This concept will have a coverage score of 0, but it describes the input data very well, and it is a valuable factor. This leads us to the problem of matrix similarity, which will be described more further later in this chapter. Now we can stick with the coverage as the exact coverage, which counts just the fully covered entries to the coverage score of the potential concept.

We can proceed with the pseudocode of the L-Grecond algorithm. Pseudocode of L-Grecond uses auxiliary procedure $selectBestTriplet$, which returns the most promising triplet, which consists of attribute, truth degree, and the corresponding coverage score.

Algorithm 2 L-Grecond

procedure L-GRECOND($I \in L^{n\times m}$) ▷ I is a real valued matrix
  set $U = \{\langle i, j\rangle | I_{i,j} > 0\}$
  set $\mathcal{F} = \emptyset$
  set $J = [n, m]$ ▷ Empty matrix, the same size as I
  while $U \neq \emptyset$ **do**
    set $D = \emptyset$ ▷ Fuzzy set - intent of factor
    set $V = 0$
    set $\langle j, a, v\rangle = selectBestTriplet(D, \mathcal{F}, U, I, J)$
    while $v > V$ **do**
      set $V = v$
      set $D = (D[j] = a)^{\downarrow\uparrow}$
      set $\langle j, a, v\rangle = selectBestTriplet(D, \mathcal{F}, U, I, J)$
    set $C = D^{\downarrow}$
    add $\langle D^{\downarrow}, D\rangle$ **to** $\mathcal{F}$
    update J with $\langle C, D\rangle$
    for each $\langle i, j\rangle \in C \times D$ **do**
      if $J_{i,j} = I_{i,j}$ **then**
        delete $\langle i, j\rangle$ **from** U
  return $\mathcal{F}$ ▷ $\mathcal{F} = \{\langle A, B\rangle \in B(I) | A \in L^N, B \in L^M\}$

Algorithm 3 *selectBestTriplet*

procedure $selectBestTriplet(D, \mathcal{F}, U, I, J)$ ▷ $I, J \in L^{n,m}$
  set $\langle j, a, v\rangle = \langle 0, 0.0, 0\rangle$
  for $j_1 \in 0, ..., m-1$ **do**
    for $a_1 \in L$ **do** ▷ L is a set of distinct values $\in \{0, .., 1\}$
      set $D_1 = D$
      set $D_1 = (D_1[j_1] = a_1)^{\downarrow\uparrow}$
      set $v_1 = s(I, J, \langle D_1^{\downarrow}, D_1\rangle, U)$
      if $v_1 > v$ **then**
        set $\langle j, a, v\rangle = \langle j_1, a_1, v_1\rangle$
  return $\langle j, a, v\rangle$

▷ s is one of the similarity functions. It compares the matrix I with the auxiliary matrix J and updated with the potential factor $\langle D_1^{\downarrow}, D_1\rangle$. It also compares only indexes which are in U.

For simplification, we separate the selection of the next promising fuzzy concept from the root procedure. As we can see, the root procedure is almost the same as in the Grecond algorithm. Let us describe the pseudocode meaning more further.
The algorithm starts with the initialization of the auxiliary structures. The set U is the set that contains all of the matrix entries which need to be covered. U is important to track the progress of the main loop of the algorithm, and when it is empty, the algorithm will stop. The second set, which we initialize, is the set $\mathcal{F}$ of concepts, which starts as an empty set, and during the algorithm progress, it is filled and then returned as the output. The last auxiliary structure is a matrix J, which begins as an empty matrix with the same size as the input matrix I. J helps the algorithm to track the coverage of the input matrix by the concepts from $\mathcal{F}$, and it is updated every time we obtain a new concept which we are adding to the solution. After the initialization of the auxiliary structures algorithm continues with the main while loop, which is executed until it covers all the uncovered matrix entries from U. The while loop starts with the initialization of D, which represents the fuzzy set of attributes (extent) and V, which is used to store the best-found coverage score. After initialization, it calls the auxiliary procedure $selectBestTriplet$, which returns the triplet of the most promising attribute j, the truth degree a in which is the attribute j the most valuable regarding the coverage and the actual coverage score v. Main while loop then extends fuzzy set D by adding the attribute j in degree a to it, and then it updates D by using the concept forming operators together, to obtain new attributes which also fit into the actual concept. Then it tries to find new attributes that could magnify the current concept score, and if there is not any possibility, it forms the final fuzzy concept, which will be added to the solution and also updates the matrix J and set U regarding the new concept. After the last iteration of main while loop algorithm returns $\mathcal{F}$ from which we can construct the matrix A and B for which holds $A \circ B = I$. The construction of the matrix A and B remains the same as we described in the chapter about boolean matrix decomposition.
Let us also describe a procedure $selectBestTriplet$. The procedure input is the fuzzy set of attributes D, which represents the intent of the fuzzy concept, which we want to extend, set of previously obtained $\mathcal{F}$, set of uncovered entries U and also input matrix I and auxiliary matrix J. The algorithm's main calculation process consists of two nested for loops. The first for loop is going through all attributes j (columns of I), and the second is going through all truth degrees $a \in L$. In each nested iteration algorithm tries to extend the intent D as D_1 with the current attribute and truth degree, and then it calculates coverage score of the D_1. The coverage score is calculated depending on the selected matrix similarity function s, which in general compares the difference between the input matrix I and auxiliary matrix J also updated with D_1. We will talk more about suitable similarity functions later as it is an important topic. Function s also does not have to compare all elements of I and J; it could just compare the

elements which are currently uncovered and are tracked in set U. Lastly, the algorithm compares the score of the D_1, and if it is bigger than the score of the previously best concept, it updates the currently best triplet $\langle j, a, v \rangle$. After all iterations of nested loops are done, the best triplet $\langle j, a, v \rangle$ is returned.
Let us also denote that algorithm will never overcover any I_{ij} which means $\forall_{i,j} (A_{\mathcal{F}} \circ B_{\mathcal{F}})_{ij} \leq I_{ij}$ because it approximates I from below.

The output of the algorithm is a set of formal concepts that describe entirely the input data I. This set can be used to form matrices A, B, which will hold $A \circ B = I$, but formal concepts can also be used for data analysis purposes.

6.3 Matrix similarity

In this section, we will describe the matrix similarity problem, and we will also introduce some of the matrix similarity functions, which will be later used in the variants of L-Grecond algorithm. In the last chapters, we will also compare the output of the algorithm with the usage of individual matrix similarity function and also measure their efficiency. The definition and substitution of matrix similarity is an essential area of our fuzzy context decomposition algorithm. When we are talking about matrix similarity, we denote the measure $s : L^{n \times m} \times L^{n \times m} \to [0, 1]$ of similarity, or approximate equality where L is the partially ordered set of truth degrees bounded by 0 and 1. In the boolean matrix case, the matrix similarity is very intuitive, as we can just check how many matrix entries are the same. In the case of matrices with grades, it is a bit more complicated as the individual matrix entries could be close to each other, but not equal. The closeness of the individual matrix entries will be our main topic in this section. We will also need to verify if the functions s, which will be introduced in this section, are complements of metrics, and in other words, if the functions $1 - s$ are metrics. This is important for the usage in the decomposition algorithm because we use the matrix similarity functions as the matrix distance measurement, and we need to verify if the functions are suitable for this purpose.

Definition 38
A metric space is an ordered pair (M, d) where M is a set and d is a metric on M. Thus d is a function $d : M \times M \to R$ such that for any $x, y, z \in M$:

1. $d(x, y) \geq 0$
2. $d(x, y) = 0 \Leftrightarrow x = y$
3. $d(x, y) = d(y, x)$
4. $d(x, y) \leq d(x, y) + d(y, z)$

In our case the set M will be the set of all matrices $I \in L^{n \times m}$ thus the function $d : L^{n \times m} \times L^{n \times m} \to [0, 1]$. Let us show some suitable similarity functions which could be helpful in the decomposition topic. For each function which we mention,

we will also verify if they are metric spaces and thus suitable for the use in the decomposition algorithm.
The first similarity function, which is also the easiest to handle, will be the similarity-equality.

Definition 39 (Equality similarity $s_=$)

$$s_=(I,J) = \frac{|i,j; 1 \leq i \leq n, 1 \leq j \leq m, I_{ij} = J_{ij}|}{n \cdot m}$$

where $I, J \in L^{n \times m}$.

The idea which is behind this function is implicitly used in the boolean matrix decomposition algorithm. $s_=$ just calculates the percentage of the similar elements by counting the number of matrix entries, which are the same in the input matrices, and dividing the number by the size of the matrix. Now we need to prove that function s is a metric space in the universe of matrices $I \in L^{n \times m}$. We will prove it by direct verification of $1. - 4.$ from the metric space definition.

Proof

For matrices $I, J, X, Y, Z \in L^{n \times m}$ and function

$$s_=(I,J) = \frac{|i,j; 1 \leq i \leq n, 1 \leq j \leq m, I_{ij} = J_{ij}|}{n \cdot m}$$

we have:

1. $1 - s_=(I,J) \geq 0$ holds, because maximum for $s_=(I,J)$ is $\frac{n \cdot m}{n \cdot m} = 1$

2. $s_=(I,J) = 1 \Leftrightarrow \forall_{i,j} I_{i,j} = J_{i,j}$, thus $1 - s_=(I,J) = 0 \Leftrightarrow I = J$

3. $s_=(I,J) = \frac{|i,j; 1 \leq i \leq n, 1 \leq j \leq m, I_{ij} = J_{ij}|}{n \cdot m} = \frac{|i,j; 1 \leq i \leq n, 1 \leq j \leq m, J_{ij} = I_{ij}|}{n \cdot m} = s_=(J,I)$

4. $1 - s_=(X,Y) \leq (1 - s_=(X,Z)) + (1 - s_=(Z,Y))$
   thus $s_=(X,Z) + s_=(Z,Y) \leq 1 + s_=(X,Y)$.
   Let $n(I,J) = |i,j; 1 \leq i \leq n, 1 \leq j \leq m, I_{ij} = J_{ij}|$
   and then $n(X,Z) + n(Z,Y) \leq n \cdot m + n(X,Y)$.
   We need to prove if $\forall i,j \; n(X_{i,j}, Z_{i,j}) + n(Z_{i,j}, Y_{i,j}) \leq 1 + n(X_{i,j}, Y_{i,j})$,
   which holds, because if $n(X_{i,j}, Z_{i,j}) + n(Z_{i,j}, Y_{i,j}) = 2$
   then $X_{i,j} = Z_{i,j} \wedge \mathrm{Z}_{i,j} = Y_{i,j} \Rightarrow \mathrm{X}_{i,j} = Y_{i,j}$

□

$s_=$ seems like a great choice of the similarity function for our purposes, but it could be problematic in some cases. We want our algorithm to discover hidden properties of input data, but with $s_=$, these properties could stay hidden for our algorithm. These hidden properties are fuzzy concepts, and the decomposition algorithm builds the set of fuzzy concepts from the essential concepts to less

important concepts. Usage of $s_=$ in the selection of next fuzzy concept will rate the fuzzy concepts by the number of fully covered uncovered entries; thus, the algorithm will never prefer some of the important fuzzy concepts which cover most of the entries but just partially. This leads us to consider another operation than $=$ for the comparison of individual matrix elements.
The next similarity function will use the biresiduum operation from fuzzy logic for the comparison of individual matrix entries.

Definition 40 (Biresiduum similarity $s_\leftrightarrow$)

$$s_\leftrightarrow(I,J) = \frac{\sum_{i,j=1}^{n,m} I_{ij} \leftrightarrow J_{ij}}{n \cdot m}$$

where $I, J \in L^{n\times m}$.

The biresiduum $\leftrightarrow$ is defined in terms of the residuum $a \to b = \bigvee\{c \mid a \otimes c \leq b\}$ via $a \leftrightarrow b = \min(a \to b, b \to a)$. The biresiduum naturally measures closeness of truth degrees. For instance, $a \leftrightarrow b = 1-|a-b|$ for the Łukasiewicz conjunction $\otimes$. We can observe that $s_\leftrightarrow(I,J) = 1$ iff $I = J$. Let us prove that $1 - s_\leftrightarrow$ is also metric space.

Proof
For matrices $I, J, X, Y, Z \in L^{n\times m}$ and function

$$s_\leftrightarrow(I,J) = \frac{\sum_{i,j=1}^{n,m} I_{ij} \leftrightarrow J_{ij}}{n \cdot m}$$

we have
1. $s_\leftrightarrow(I,J) = \frac{\sum_{i,j=1}^{n,m} I_{ij}\leftrightarrow J_{ij}}{n\cdot m} = \frac{\sum_{i,j=1}^{n,m} 1-|I_{ij}-J_{ij}|}{n\cdot m}$, so maximum is $\frac{n\cdot m}{n\cdot m} = 1$, thus $1 - s_\leftrightarrow(I,J) \geq 0$

2. " $\Rightarrow$ " $\quad 1 - s_\leftrightarrow(I,J) = 0 \Leftrightarrow \forall_{i,j} I_{i,j} = J_{i,j} \Rightarrow I = J$
" $\Leftarrow$ " $\quad I = J \Leftarrow s_\leftrightarrow(I,J) = 1 \Leftarrow 1 - s_\leftrightarrow(I,J) = 0$

3. Because $|I_{i,j} - J_{i,j}| = |J_{i,j} - I_{i,j}|$ we have
$s_\leftrightarrow(I,J) = \frac{\sum_{i,j=1}^{n,m} 1-|I_{ij}-J_{ij}|}{n\cdot m} = \frac{\sum_{i,j=1}^{n,m} 1-|J_{ij}-I_{ij}|}{n\cdot m} = s_\leftrightarrow(J,I)$

4. $1 - s_\leftrightarrow(X,Y) \leq (1 - s_\leftrightarrow(X,Z)) + (1 - (Z,Y))$ thus
$s_\leftrightarrow(X,Z) + s_\leftrightarrow(Z,Y) \leq 1 + s_\leftrightarrow(X,Y)$ and moreover
$\forall_{i,j}\ (1 - |X_{i,j} - Z_{i,j}|) + (1 - |Z_{i,j} - Y_{i,j}|) \leq 1 + (1 - |X_{i,j} - Y_{i,j}|)$.
From triangle inequality we have $|X_{i,j} - Y_{i,j}| \leq |X_{i,j} - Z_{i,j}| + |Z_{i,j} - Y_{i,j}|$ so the previous statement holds.

□

Usage of the $s_\leftrightarrow$ will solve the previously mentioned issue as it will also give a score for partially covered entries and not just the fully covered. We can

generalize the $s_{\leftrightarrow}$ as

$$s_L(I,J) = \frac{\sum_{i,j=1}^{n,m} s_l(I_{ij}, J_{ij})}{n \cdot m}, \tag{8}$$

where $s_l : L \times L \to L$ is function which meets some requirements. The function s_l needs to satisfy three basic requirements.
Namely $\forall x, y, z$ function s_l needs to hold

$$\begin{aligned} &1. s_l(x,y) = 1 \Leftrightarrow x = y \\ &2. s_l(x,y) = s_l(y,x) \\ &3. s_l(x,z) + s_l(z,y) \leq s_l(x,y). \end{aligned}$$

If the function s_l meets these requirements, it could be used inside the s_L, and s_L will meet the requirements to be the complement of metric space.
Example of function s_l is $s_l(x,y) = x \leftrightarrow y$ or $s_l(x,y) = \begin{cases} 1 \text{ if } x = y \\ 0 \text{ otherwise} \end{cases}$.

Usage of great function s in the decomposition algorithm is crucial to obtain valuable fuzzy concepts in the output. We showed the function of $s_=$, which has some disadvantages and the function $s_{\leftrightarrow}$, which solves them. However, $s_{\leftrightarrow}$ also has some issues. The decomposition algorithm, which uses the score obtained by $s_{\leftrightarrow}$ for the selection of the next fuzzy concept, could tend to prefer "flat" concepts, which are wide but usually not valuable. As example we can imagine matrix I in which every matrix entry is $I_{i,j} = 0.9$ or $I_{i,j} = 0.5$. In this case, the algorithm will select the most valuable factor, the fuzzy concept, which explains the whole matrix I but only in degree 0.5. The penalization of these flat factors leads us to the extension of $s_{\leftrightarrow}$ as

$$s_f(I,J) = \frac{\sum_{i,j=1}^{n,m} f(I_{ij} \leftrightarrow J_{ij})}{n \cdot m} \tag{9}$$

where $f : L \to L$. The function f needs to be not decreasing and also it needs to hold $f(x) = 1 \Leftrightarrow x = 1$. If we put $f(a) = a$ we will obtain the $s_{\leftrightarrow}$. The setting of function f gives us the possibility to penalize some kind of factors. In our case we will use $f(a) = a^{q\sqrt{mn}}$ where parameter q can be adjusted for the bigger or lesser penalization of "flat" factors. Tuning of parameter q allows us to adjust the preference of factors between $s_=$ and $s_{\leftrightarrow}$. Parameter q should also be changed among datasets as it works together with $\sqrt{mn}$, which depends on the input size. Let us name this similarity function as a Scaled similarity of matrices.

Definition 41 (Scaled biresiduum similarity $s_{f\leftrightarrow}$)

$$s_{f\leftrightarrow}(I,J) = \frac{\sum_{i,j=1}^{n,m}((I_{ij} \leftrightarrow J_{ij})^{q\sqrt{mn}})}{n \cdot m}$$

where $I, J \in L^{n \times m}$.

6.4 Variants of basic algorithm

We can obtain different variants of the decomposition algorithm by using different similarity functions inside the auxiliary procedure *selectBestTriplet*. The substitution of s could change the preference of factors in the algorithm, and therefore algorithm can provide us another set of formal concepts as output. We will compare the differences in the output when using the similarity functions we mentioned in the previous section. We will equip our decomposition algorithm with $s_{=}$, $s_{\leftrightarrow}$ or $s_{f\leftrightarrow}$ and then check the difference in the sets of fuzzy concepts returned.

6.5 Complexity

As mentioned about boolean matrix decomposition, the Grecond algorithm does not guarantee to find the optimal solution. The optimal solution for input I is the one which has the smallest possible dimension k for the matrices $A \in \{0,1\}^{n\times k}$ and $B \in \{0,1\}^{k\times m}$ for which holds $A \circ B = I$. The k is also the number of concepts returned as a solution. If we want to find the optimal solution in the boolean case, we have to go through all possible boolean matrices A and B while increasing the dimension k from 1 until we find such $A \circ B = I$. The universe of boolean matrices with dimension $k > 15$ is almost impossible to go through, and the situation is getting much worse when we want to browse the universe of matrices $A \in L^{n\times k}$ and $B \in L^{k\times m}$. That is why the L-Grecond has to use the greedy approach to build a set of fuzzy concepts and does not guarantee to find the optimal solution. Let us talk about the real input complexity of the L-Grecond algorithm and also the variables which affect the complexity.
Our algorithm starts with the empty set of fuzzy concepts and uses the procedure *selectBestTriplet* for the greedy steps of choosing the next fuzzy concept. If we simplify the whole process, we can say the algorithm calls the *selectBestTriplet* until the set U is not empty. So the first variable which affects the complexity is the number of not null entries in the input matrix I. Now let us talk about the procedure *selectBestTriplet*. *selectBestTriplet* is looping through all attributes which are not present in the input set D, and for each such attribute, it also loops through all $a \in L$. Therefore the main variables which affect the complexity of *selectBestTriplet* are the number of attributes and number of truth degrees in L. The number of calls of procedure *selectBestTriplet* is hard to estimate, as we do not know how many concepts will cover the input matrix. The number of formal concepts present in the context could be exponential regarding the size of the context, and the algorithm may need to go through all of them. However, it is not likely to happen.
The main variables which affect the complexity of the algorithm are the size of U, the number of attributes, and also the number of truth degrees. The number of objects also affects the complexity, but not directly as the algorithm is not looping through them. Objects affect the complexity during the usage of concept forming operators $\downarrow, \uparrow$. The selection of different function s has a minor impact

on complexity. For clarification, each similarity function s, if implemented well, compares just the differences in matrix entries in U and while using the $s_{f\leftrightarrow}$ the $\sqrt{nm}$ should be calculated just once.

7 Experiments with real data

In this chapter, we will examine in detail various data as decathlon results, students' performance, and election results. We will compare and evaluate the output of the decomposition algorithm equipped with different similarity function s. Our main aim will be to evaluate the reasonability of dominant factors and also compare the basic properties as the size of the output set of factors. The truth degree scale L for object-attribute relation will be different within the examples, but we will not use more comprehensive L than $L = \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$. As the datasets are not usually in the form we would like them to be, we will also talk about the transformation and preparation of the data within the examples.
The framework of the experiments will be as follows. First, we will describe the transformation of the data to the desired form. Then we will run L-Grecond algorithm equipped with each of the similarity function $s_{=}, s_{\leftrightarrow}$ and $s_{f\leftrightarrow}$. We will check the reasonability of dominant factors from each output $\mathcal{F}$, and we will also compare the number of factors, coverage score of each factor included in different solutions, and the associated coverage of the first $F_i \in \mathcal{F}$ factors in the solution. Let us talk more about the coverage itself.
We will use two different methods to calculate the coverage. These methods will be giving us the percentage of cover based on matrix similarity functions $s_{=}$ and $s_{f\leftrightarrow}$. So for each variant of the algorithm, we will have two different coverage measurements for individual factors and two different coverage measurements of merged factors. The first coverage will be called *Exact coverage*, which will tell us the percentage of the fully covered matrix entries; therefore, it tells us how many percents of the constructed matrix is the same as the input matrix. The second coverage will be called *Biresiduum coverage*, which will determine the percentage of entries that are covered concerning $s_{\leftrightarrow}$; thus, it will also count the not fully covered entries. We also considered the score calculated regarding $s_{f\leftrightarrow}$, but we will omit it in this chapter as it does not give much information. In the first factors performance comparison section, we will present the information obtained more thoroughly, and in the rest, it will be simplified.

7.1 Decathlon

The first dataset which we will examine is decathlon results from the summer Olympic Games 2004. The dataset consists of 5 best athletes and their scores in corresponding decathlon disciplines. The raw data obtained using the IAAF Scoring Tables are shown in Table 1.

Table 1: Decathlon score table

	100m	*LJ*	*SP*	*HJ*	*400m*	*110mH*	*DT*	*PV*	*JT*	*1500m*
Sebrle	894	1020	873	915	892	968	844	910	897	680
Clay	989	1050	804	859	852	958	873	880	885	668
Karpov	975	1012	847	887	968	978	905	790	671	692
Macey	885	927	835	944	863	903	836	731	715	775
Warners	947	995	758	776	911	973	741	880	669	693

The rows from the decathlon score table are particular athletes, and the columns are disciplines such as 100-meter sprint race, long jump, shot put, high jump, 400 meters sprint race, 110 meters hurdles, discus throw, pole vault, javelin throw, and 1500 meter run. Therefore if we want to clarify our table as context, the rows are objects, and columns are the attributes of objects. Next, we will talk about the transformation of the data to a suitable form. Let us also denote here that for experiments with decathlon dataset the parameter q which is used in $s_{f\leftrightarrow}$ was set to $q = 0.5$.

7.1.1 Data transformation

We have specified the objects and attributes of our dataset, but we also need to specify the fuzzy relation to obtain the full fuzzy context, which we need to run our algorithm. As the matrix entries in the decathlon score table are not truth degrees, we will need to do data transformation. Data transformation could usually be handled in many different ways, but for this dataset, there is just one reasonable transformation that we will use. We need to normalize the matrix entries to the elements from interval $[0, 1]$. To be more concrete, we will transform the data from Table 1 to eleven-element scale

$$L = \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$$

by linear transformation and rounding. For every discipline, we will take the lowest and highest score achieved among all present athletes, and then we will perform a linear transformation from interval $[lowest, highest]$ to $[0, 1]$ and round to the closest element of L. For the 100-meter sprint race, we will obtain

$$f_{100m}(x) = \frac{x - 782}{989 - 782} = \frac{x - 782}{207} \in [0, 1].$$

As an example we can show the calculation of scaled value for Sebrle, thus

$$f_{100m}(894) = \frac{894 - 782}{207} = \frac{112}{207} = 0.541$$

which we can round to the $0.5 \in L$.
The function f is different for each column as the highest and lowest score is different in every discipline. After we obtain all functions f, we need to do the transformation of each matrix entry to obtain the fuzzy relation. The final fuzzy context, in this case, is depicted in 2.

Table 2: Decathlon as fuzzy context

	100m	*LJ*	*SP*	*HJ*	*400m*	*110mH*	*DT*	*PV*	*JT*	*1500m*
Sebrle	0.6	0.9	1.0	0.9	0.7	1.0	0.8	0.7	1.0	0.7
Clay	1.0	1.0	0.7	0.7	0.6	0.9	0.9	0.6	1.0	0.6
Karpov	0.9	0.9	0.9	0.8	1.0	1.0	1.0	0.3	0.2	0.7
Macey	0.6	0.6	0.8	1.0	0.6	0.7	0.8	0.2	0.4	1.0
Warners	0.8	0.8	0.5	0.4	0.8	1.0	0.4	0.6	0.2	0.7

The matrix entries are now members of a fuzzy relation. The meaning of individual entry is the degree of skill that the athlete has for the discipline.

7.1.2 L-Grecond equipped with $s_=$

Let the fuzzy context $\mathcal{I} = \langle X, Y, I \rangle$, where X are our athletes, Y are disciplines and I is the fuzzy relation between the athletes and disciplines depicted in 2. The algorithm equipped with the similarity function $s_=$ found for the input context $\mathcal{I}$ a set

$$
\begin{aligned}
\mathrm{F}_{s_=} = \{ & \\
& \langle \{1.0, 0.9, 1.0, 0.7, 0.6\} , \{0.6, 0.9, 0.8, 0.8, 0.7, 1.0, 0.8, 0.3, 0.2, 0.7\} \rangle, \\
& \langle \{0.9, 0.7, 0.8, 1.0, 0.4\} , \{0.6, 0.6, 0.8, 1.0, 0.6, 0.7, 0.8, 0.2, 0.4, 0.8\} \rangle, \\
& \langle \{0.6, 1.0, 0.7, 0.6, 0.8\} , \{1.0, 1.0, 0.7, 0.6, 0.6, 0.9, 0.6, 0.6, 0.4, 0.6\} \rangle, \\
& \langle \{0.7, 0.6, 1.0, 0.6, 0.4\} , \{0.9, 0.9, 0.9, 0.8, 1.0, 1.0, 1.0, 0.3, 0.2, 0.7\} \rangle, \\
& \langle \{0.9, 0.8, 0.7, 0.6, 1.0\} , \{0.7, 0.8, 0.5, 0.4, 0.8, 1.0, 0.4, 0.6, 0.2, 0.7\} \rangle, \\
& \langle \{1.0, 0.7, 0.2, 0.4, 0.2\} , \{0.6, 0.9, 1.0, 0.9, 0.7, 1.0, 0.8, 0.7, 1.0, 0.7\} \rangle, \\
& \langle \{0.9, 1.0, 0.2, 0.4, 0.2\} , \{0.7, 1.0, 0.7, 0.7, 0.6, 0.9, 0.9, 0.6, 1.0, 0.6\} \rangle, \\
& \langle \{0.7, 0.6, 0.7, 1.0, 0.7\} , \{0.6, 0.6, 0.8, 0.7, 0.6, 0.7, 0.7, 0.2, 0.4, 1.0\} \rangle \}
\end{aligned}
$$

of eight fuzzy concepts.
The factor concepts are shown in order in which the algorithm obtained them. If we construct corresponding matrices $A_\mathcal{F}, B_\mathcal{F}$ we can obtain $I = A_\mathcal{F} \circ B_\mathcal{F}$. For our $\mathcal{F}_=$ the corresponding matrices are

$$
A_\mathcal{F} = \begin{pmatrix}
1.0 & 0.9 & 0.6 & 0.7 & 0.9 & 1.0 & 0.9 & 0.70 \\
0.9 & 0.7 & 1.0 & 0.6 & 0.8 & 0.7 & 1.0 & 0.6 \\
1.0 & 0.8 & 0.7 & 1.0 & 0.7 & 0.2 & 0.2 & 0.7 \\
0.7 & 1.0 & 0.6 & 0.6 & 0.6 & 0.4 & 0.4 & 1.0 \\
0.6 & 0.4 & 0.8 & 0.4 & 1.0 & 0.2 & 0.2 & 0.7
\end{pmatrix}
$$

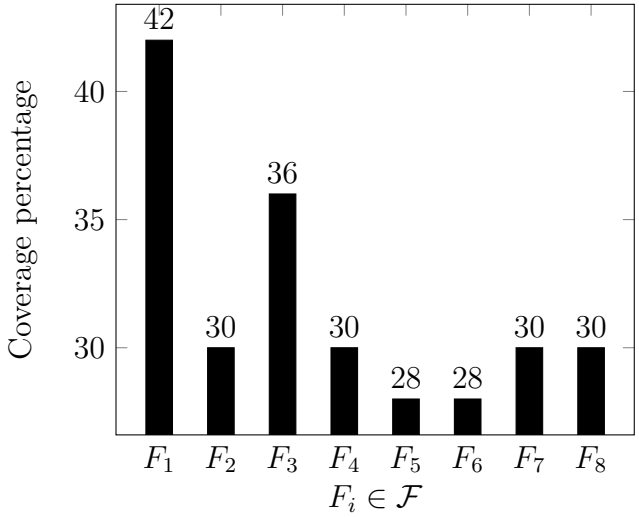
$$B_{\mathcal{F}} = \begin{pmatrix} 0.6 & 0.9 & 0.8 & 0.8 & 0.7 & 1.0 & 0.8 & 0.3 & 0.2 & 0.7 \\ 0.6 & 0.6 & 0.8 & 1.0 & 0.6 & 0.7 & 0.8 & 0.2 & 0.4 & 0.8 \\ 1.0 & 1.0 & 0.7 & 0.6 & 0.6 & 0.9 & 0.6 & 0.6 & 0.4 & 0.6 \\ 0.9 & 0.9 & 0.9 & 0.8 & 1.0 & 1.0 & 1.0 & 0.3 & 0.2 & 0.7 \\ 0.7 & 0.8 & 0.5 & 0.4 & 0.8 & 1.0 & 0.4 & 0.6 & 0.2 & 0.7 \\ 0.6 & 0.9 & 1.0 & 0.9 & 0.7 & 1.0 & 0.8 & 0.7 & 1.0 & 0.7 \\ 0.7 & 1.0 & 0.7 & 0.7 & 0.6 & 0.9 & 0.9 & 0.6 & 1.0 & 0.6 \\ 0.6 & 0.6 & 0.8 & 0.7 & 0.6 & 0.7 & 0.7 & 0.2 & 0.4 & 1.0 \end{pmatrix}.$$

In the rest of the examples, we will omit the set of fuzzy concepts $\mathcal{F}$ as the matrices $A_{\mathcal{F}}$ and $B_{\mathcal{F}}$ are more readable. The individual fuzzy concepts of $\mathcal{F}$ can be obtained from $A_{\mathcal{F}}, B_{\mathcal{F}}$ as columns of $A_{\mathcal{F}}$ are intents and rows from $B_{\mathcal{F}}$ are extents.

7.1.2.1 Performance of individual factors

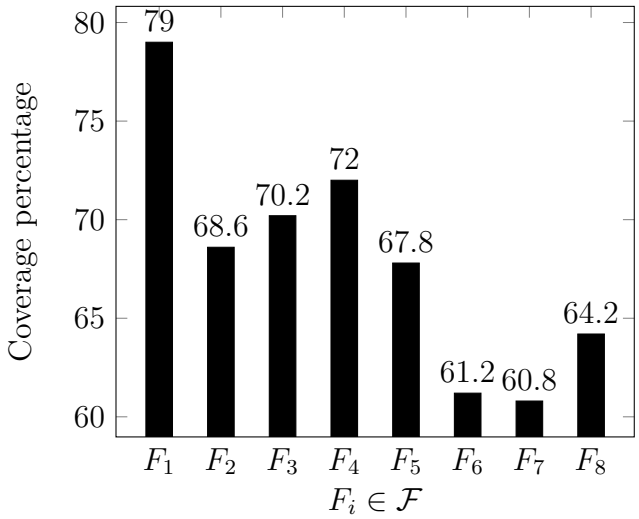
Let us talk about the portion of data explained by individual factors from $\mathcal{F}$. The coverage percentage of an individual factor can be calculated by comparing the input matrix I with the matrix obtained by $A_{F_i} \circ B_{F_i}$. We will visualize the percentage of data explained by the bar graph in which the x axis will represent the fuzzy concept from $\mathcal{F}$ and y axis will stand for the percentage of input covered. Let us start with a graph which shows *Exact coverage* of the factors.

Figure 1: Exact coverage for $F_{s_=}$ (Decathlon)



We can see that F_1 and F_3 are more dominant than the others, but all the factors have reached a great coverage score. Let us show the coverage against the *Biresiduum coverage*.

Figure 2: Biresiduum coverage for $F_{s_=}$ (Decathlon)



We can observe that the F_1 is still a dominant factor, but the other factors are more equivalent. The main reason the percentages are far higher against the *Exact coverage* measurement is because the *Biresiduum coverage* also counts the entries which are covered just partially. We can say that the F_1 is the most dominant factor, which reasonably explains almost the whole decathlon dataset. F_1 is the factor in which long jump and 110 meters hurdles are significant, and pole vault and javelin throw are almost not necessary at all. Therefore the best five athletes are excellent in the long jump and 110 meters hurdles, and they are not the best among all athletes in the javelin throw and pole vault.

7.1.2.2 Merged performance of factors

In this section, we will compare the performance of a grouped coverage of first F_i factors instead of comparing just their individual coverage. A merged coverage can be calculated by comparing the input matrix I with the output of $A \circ B$, where the matrices A, B are constructed just with the first i concepts $F_i \in \mathcal{F}$. With the growing i, the percentage of input coverage will converge to 100%. For the clarification of the following graph, the y-axis represents the percentage of input covered, and the x-axis indicates how many of the first factors are included in the coverage calculation. Let us start with the *Exact merged coverage*, which is shown in Figure 3.

We can observe that the first five factors exactly explain almost 90% of the input data; therefore, the factors F_6, F_7, F_8 are probably not as essential and universal. We will continue with the biresiduum coverage. The corresponding graph is shown in Figure 4.

As we can see, we can obtain an almost similar matrix as the input matrix I just by using the F_1, F_2, F_3 for its construction. Also, we can observe that F_8 is an almost unnecessary factor which covers just the least 0.4% of I, so if we omit F_8 from the solution, the output from $A_{\mathcal{F}} \circ B_{\mathcal{F}}$ will be probably indistinguishable

Figure 3: Exact merged coverage for $F_{s_=}$ (Decathlon)

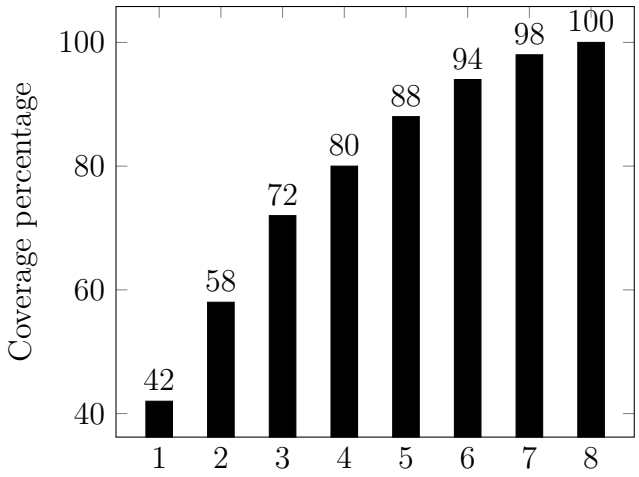
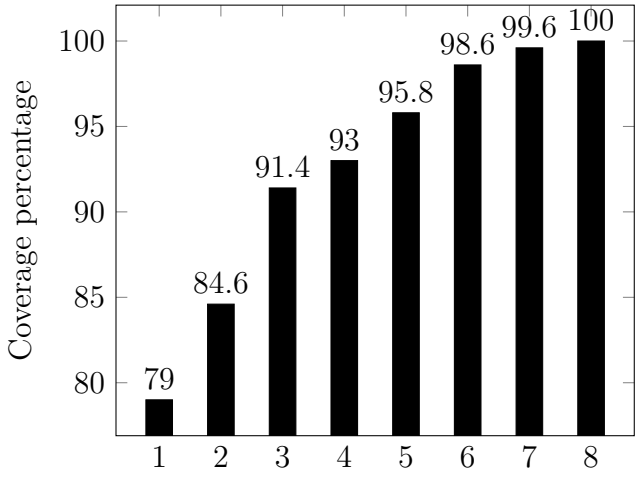


Figure 4: Biresiduum merged coverage for $F_{s_=}$ (Decathlon)



from the I.
The graphs presented in this section were presented distinctly just for clarification of the concept of measurement and transparency. In the following examples, we will merge these graphs to make them more useful for comparison purposes.

7.1.3 L-Grecond equipped with $s_{\leftrightarrow}$

The fuzzy context $\mathcal{I} = \langle X, Y, I \rangle$ remains the same. For reference, fuzzy relation I is depicted in Table 2. Algorithm equipped with the similarity function $s_{\leftrightarrow}$ found $\mathcal{F}_{\leftrightarrow} = \{F_i = \langle C_i, D_i \rangle | 1, ..., 10\}$. Accordingly, the algorithm needed two more concepts to explain the data, which is caused by another preference of factors during the calculation. The matrices $A_{\mathcal{F}_{\leftrightarrow}}, B_{\mathcal{F}_{\leftrightarrow}}$ constructed from $\mathcal{F}_{\leftrightarrow}$ are as follows.

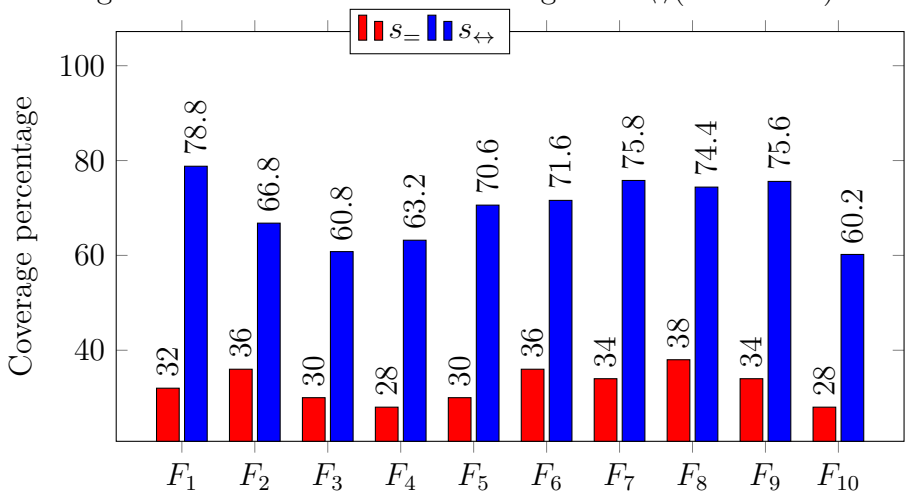
$$A_{\mathcal{F}\leftrightarrow} = \begin{pmatrix} 0.9 & 1.0 & 0.9 & 0.7 & 0.7 & 0.6 & 1.0 & 1.0 & 0.8 & 1.0 \\ 0.8 & 0.9 & 1.0 & 0.6 & 0.6 & 1.0 & 0.7 & 0.9 & 0.9 & 1.0 \\ 0.9 & 0.6 & 0.2 & 0.7 & 1.0 & 0.9 & 0.9 & 1.0 & 1.0 & 0.2 \\ 0.7 & 0.5 & 0.4 & 1.0 & 0.6 & 0.6 & 0.8 & 0.7 & 0.8 & 0.4 \\ 0.5 & 0.9 & 0.2 & 0.4 & 0.8 & 0.8 & 0.5 & 1.0 & 0.4 & 0.2 \end{pmatrix}$$

$$B_{\mathcal{F}\leftrightarrow} = \begin{pmatrix} 0.7 & 0.9 & 0.9 & 0.9 & 0.8 & 1.0 & 0.9 & 0.4 & 0.3 & 0.8 \\ 0.6 & 0.9 & 0.6 & 0.5 & 0.7 & 1.0 & 0.5 & 0.7 & 0.3 & 0.7 \\ 0.7 & 1.0 & 0.7 & 0.7 & 0.6 & 0.9 & 0.9 & 0.6 & 1.0 & 0.6 \\ 0.6 & 0.6 & 0.8 & 1.0 & 0.6 & 0.7 & 0.8 & 0.2 & 0.4 & 1.0 \\ 0.9 & 0.9 & 0.7 & 0.6 & 1.0 & 1.0 & 0.6 & 0.3 & 0.2 & 0.7 \\ 1.0 & 1.0 & 0.7 & 0.6 & 0.6 & 0.9 & 0.6 & 0.4 & 0.3 & 0.6 \\ 0.6 & 0.8 & 1.0 & 0.9 & 0.7 & 0.9 & 0.8 & 0.4 & 0.3 & 0.7 \\ 0.6 & 0.8 & 0.5 & 0.4 & 0.7 & 1.0 & 0.4 & 0.3 & 0.2 & 0.7 \\ 0.8 & 0.8 & 0.8 & 0.8 & 0.7 & 0.9 & 1.0 & 0.3 & 0.2 & 0.7 \\ 0.6 & 0.9 & 0.7 & 0.7 & 0.6 & 0.9 & 0.8 & 0.6 & 1.0 & 0.6 \end{pmatrix}.$$

7.1.3.1 Performance of individual factors

Figure 5 shows all of the coverage scores for individual factors. For clarification, the y-axis is the percentage covered, the x-axis indicates the concept, and the bar color distinguishes the method of coverage measurement. The methods used for measurement are the same as in the previous examples, but now we aggregated both methods to one graph. The coverage percentage concerning $s_{=}$ is marked by red color, while the coverage concerning $s_{\leftrightarrow}$ is marked by blue.
We can observe that the coverage percentage is almost the same as for the factors found by the algorithm equipped with $s_{=}$. The first factor found is the same as in set $\mathcal{F}_{=}$, and the rest has just a little difference regarding the coverage. The main difference is only that the current variant of the algorithm needed one more concept to explain the input completely, but the individual coverage of found factors is quite similar to these in $\mathcal{F}_{=}$.

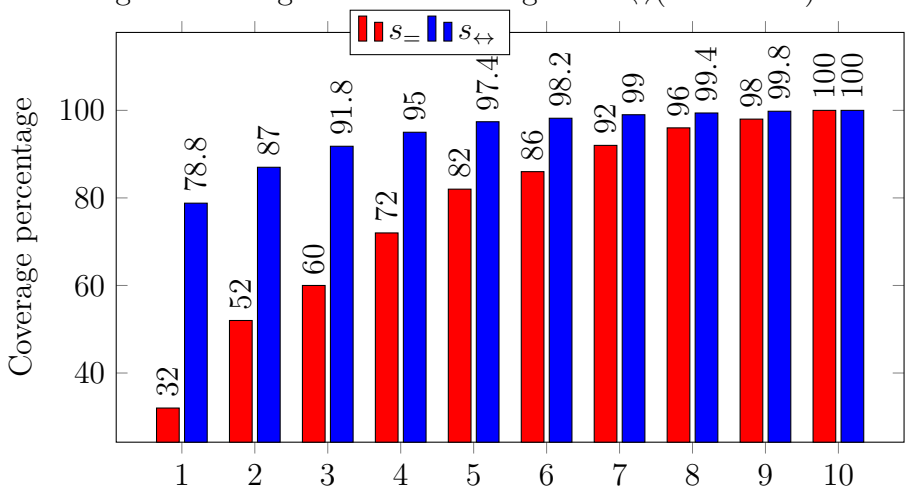
Figure 5: Individual factors coverage for $F_{\leftrightarrow}$(Decathlon)



7.1.3.2 Merged performance of factors

Now we will talk about the merged coverage of factors in $\mathcal{F}_{\leftrightarrow}$. The percentages are shown in Figure 6, and all of the coverage measurements are again aggregated in one bar graph. The bar colors and meaning of the y-axis remain the same. The x-axis indicates how many of $F_i \in \mathcal{F}_{\leftrightarrow}$ are included in the calculation of the particular comparison matrix.
The performance is again pretty similar to the performance of factors from $\mathcal{F}_=$, but if we look more thoroughly, we can observe that F_1, F_2 together have a slightly better performance concerning the biresiduum. Although, in general, the obtained factors have a lesser performance concerning the exact coverage, in some cases, the coverage against the biresiduum cover could have more value when obtaining reasonable factors from the dataset.

Figure 6: Merged factors coverage for $F_{\leftrightarrow}$(Decathlon)



7.1.4 L-Grecond equipped with $s_{f\leftrightarrow}$

The fuzzy context $\mathcal{I} = \langle X, Y, I\rangle$ remains the same. Corresponding fuzzy relation I used as input is depicted in Table 2. Algorithm equipped with the similarity function $s_{f\leftrightarrow}$ and $q = 0.5$ found $\mathcal{F}_{f\leftrightarrow} = \{F_i = \langle C_i, D_i\rangle | 1, ..., 8\}$. Therefore algorithm was able to explain the input I by the same number of concepts as algorithm equipped with $s_{=}$. The matrices $A_{\mathcal{F}_{f\leftrightarrow}}, B_{\mathcal{F}_{f\leftrightarrow}}$ constructed from $\mathcal{F}_{f\leftrightarrow}$ are as follows.

$$A_{\mathcal{F}_{f\leftrightarrow}} = \begin{pmatrix} 1.0 & 0.8 & 0.9 & 0.7 & 0.7 & 0.6 & 1.0 & 0.9 \\ 0.8 & 0.9 & 1.0 & 0.6 & 0.6 & 1.0 & 0.7 & 0.8 \\ 1.0 & 0.7 & 0.2 & 0.7 & 1.0 & 0.9 & 0.2 & 1.0 \\ 0.7 & 0.6 & 0.4 & 1.0 & 0.6 & 0.6 & 0.4 & 0.8 \\ 0.6 & 1.0 & 0.2 & 0.4 & 0.4 & 0.8 & 0.2 & 1.0 \end{pmatrix}$$

$$B_{\mathcal{F}_{f\leftrightarrow}} = \begin{pmatrix} 0.6 & 0.9 & 0.9 & 0.8 & 0.7 & 1.0 & 0.8 & 0.3 & 0.2 & 0.7 \\ 0.8 & 0.8 & 0.5 & 0.4 & 0.7 & 1.0 & 0.4 & 0.6 & 0.2 & 0.7 \\ 0.7 & 1.0 & 0.7 & 0.7 & 0.6 & 0.9 & 0.9 & 0.6 & 1.0 & 0.6 \\ 0.6 & 0.6 & 0.8 & 1.0 & 0.6 & 0.7 & 0.8 & 0.2 & 0.4 & 1.0 \\ 0.9 & 0.9 & 0.9 & 0.8 & 1.0 & 1.0 & 1.0 & 0.3 & 0.2 & 0.7 \\ 1.0 & 1.0 & 0.7 & 0.6 & 0.6 & 0.9 & 0.6 & 0.4 & 0.3 & 0.6 \\ 0.6 & 0.9 & 1.0 & 0.9 & 0.7 & 1.0 & 0.8 & 0.7 & 1.0 & 0.7 \\ 0.7 & 0.8 & 0.5 & 0.4 & 0.8 & 0.9 & 0.4 & 0.3 & 0.2 & 0.7 \end{pmatrix}.$$

We will not go into the details about the coverage data obtained as the coverage percentage data are almost the same.
In the Figure 7 an 8 we can check the percentage coverage of the factors.
Let us denote that the usage of the particular similarity function does not guarantee to obtain smaller output (fewer factors). The reason why the algorithms equipped with $s_{=}$ and $s_{f\leftrightarrow}$ were able to find a smaller set of factors that fully explained the output is more about the luck of greedy choice. Tuning of parameter q can lead us to obtain another set of factors, as it can tune algorithm preference of factors between two extremes - the exact similarity and biresiduum.

Figure 7: Individual factors coverage for $\mathcal{F}_{f\leftrightarrow}$ (Decathlon)

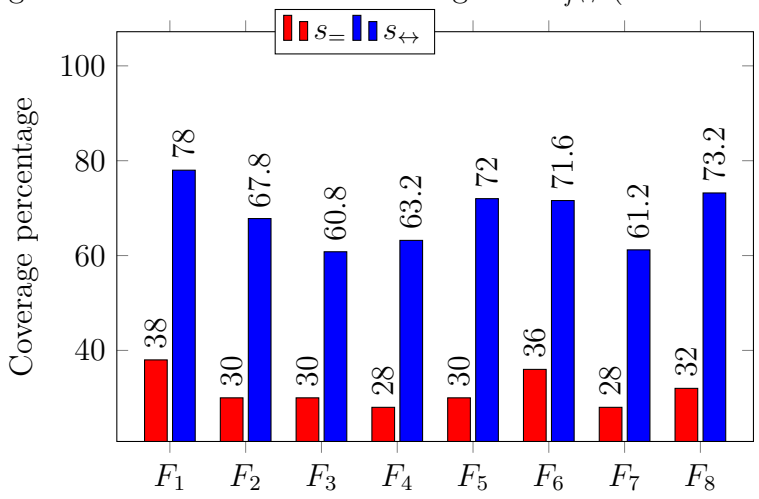
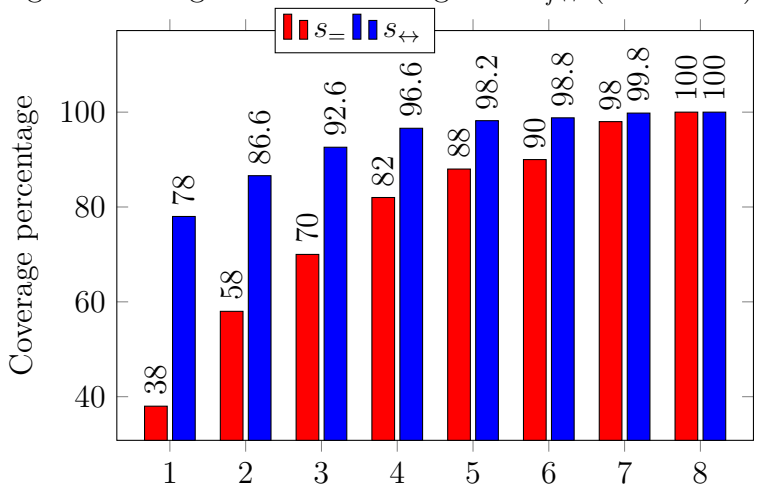


Figure 8: Merged factors coverage for $\mathcal{F}_{f\leftrightarrow}$ (Decathlon)



7.2 Education data

In this chapter, we will analyze data that consist of anonymized data coming from examination tests that are used by UK universities to measure and select appropriate students. Initial work on this dataset is described in paper [7], where the authors tried to find factors that may explain the students' performance with the help of the FCA method. This dataset is exciting for our purposes as it exhibits the property of so-called "flat" factors, and we will try to analyze how to deal with them.

The dataset contains results of 2774 individual students, who were performing on a given examination in the subject "Government and Politics". The whole examination consists of four modules, but we will deal just with the second module, which covers questions about the current British governance. In the British governance module, students have to choose two out of four topics. The dataset contains results just from the topics "parliament" and "executive", as these two are the most popular combination among students. Each of these topics consists of three questions. Examiners assess the first question with regards to "knowledge and understanding", and the second and third questions are assessed additionally to two more objectives, namely "analysis and evaluation" and "communication". Therefore for each student, we have seven examination results for each topic, and thus 14 results in total. Each examination result is marked from 4 to 0, where 4 represents the best performance and 0 the worst. The sum of all marks gives the total mark of maximum value 80. A students' final result is based on the summed value by thresholding, and possible grades are A, B, C, D, E, and N, where A is the best result. In the whole dataset, 607 students obtained grade A as the final result, and we will try to analyze their factors. An example of a few raw data rows is shown in Table 3. In the data transformation section, we will talk about how data are transformed into a suitable form to be analyzed by our methods. Then we will discuss and measure the performance of the factor found by each of the variants of our algorithm.

Table 3: Education data

ID	*Total*	*Grade*	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	56	5	1	3	3	3	3.5	3	3	3.5	3	3	3	3	3	3
2	60	5	2	2	4	4	3.5	3	3	3.5	3	3	3	3	3	3
3	50	4	2	2	3	2	1.5	3	3	3.5	3	3	3	3	3	3
4	61	5	2	2	4	4	3.5	3	3	3.5	3	3	3	3	3	3
5	44	3	3	2	2	2	3.5	2	2	3.5	3	3	3	2	2	2
6	40	2	2	3	3	2	3.5	3	2	3.5	2	2	2	2	2	2
7	47	3	2	2	3	2	3.5	3	3	3.5	3	3	3	2	2	2
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

7.2.1 Data transformation

As we analyze the performance of the best students who obtained the final grade A, the dataset which we will analyze is represented by 607×14 matrix I. Each of 607 student examination is represented by 14 fuzzy attributes over a five-element scale $L = \{0, 0.25, 0.5, 0.75, 1\}$ whose degrees correspond the marks $0, ..., 4$. In the raw data (Table 3), we can observe that in the two columns, there are some not crisp values. We will round these values down to solve inaccuracies. The value 1.5 will be represented by $0.25 \in L$ and 3.5 by $0.75 \in L$. A part of the final transformed data which is suitable for our analysis is shown in Table 4. For clarification, Table 4 is yet filtered on A-grade students; therefore the first three lines correspond to the students with IDs 1,2,4, and the other lines do not have their corresponding student shown in Table 3.

Table 4: Education data as fuzzy context (A students)

1	2	3	4	5	6	7	8	9	10	11	12	13	14
0.25	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75
0.5	0.5	1	1	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75
0.5	0.5	1	1	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75
0.75	1	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75
0.5	1	0.75	0.5	0.75	1	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75
0.75	0.5	0.75	0.75	0.75	0.75	0.75	0.75	0.75	1	1	0.5	0.5	0.75
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

In the next sections we will analyze a following fuzzy context $\mathcal{I} = \langle X, Y, I \rangle$, where X are the best performing students, Y are the examination results and $I \in L^{607 \times 14}$. First, we will show the performance of individual methods, and then we will discuss the difference between the output from each of them. Every variant of the algorithm needed more than 30 factors to explain the dataset fully. We will focus on the first 14 of them as in all variants, the first 14 factors explain around 95% of input data, so the rest is not reasonably important for us.

7.2.2 Performance of L-Grecond equipped with $s_=$

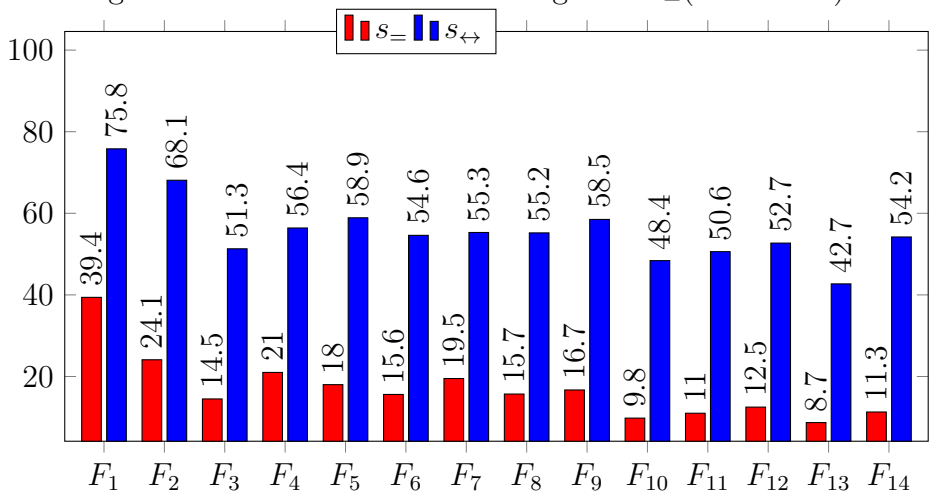
The algorithm equipped with the similarity function $s_=$ found for the input context $\mathcal{I}$ a set $\mathcal{F}_= = \{F_i = \langle C_i, D_i \rangle | 1, ..., 31\}$ of 31 concepts (factors). The factors are generated one by one, from the most significant ones in terms of data coverage to the least significant. As the number of factors is high and the higher is the number of factors, the lesser is the data coverage growth, we will deal with just the first 14 factors as they together explain more than 95% of the data. We will also omit the intents as the corresponding matrix A has 607 rows, so we will

just show the first 14 extents which are depicted in matrix $B_{\mathcal{F}_=}$.

$$B_{\mathcal{F}_=} = \begin{pmatrix}
0.75 & 0.75 & 1.0 & 0.75 & 1.0 & 0.75 & 0.75 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\
0.75 & 0.75 & 1.0 & 1.0 & 0.5 & 1.0 & 1.0 & 0.5 & 1.0 & 0.75 & 1.0 & 0.75 & 0.75 & 1.0 \\
1.0 & 1.0 & 0.5 & 0.5 & 0.25 & 0.0 & 0.0 & 0.0 & 0.75 & 0.75 & 0.75 & 0.75 & 0.5 & 0.75 \\
0.5 & 0.25 & 0.5 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & 1.0 & 1.0 & 1.0 & 0.75 & 0.75 & 0.75 \\
0.25 & 0.5 & 0.5 & 0.25 & 0.25 & 1.0 & 0.75 & 0.75 & 0.75 & 0.5 & 0.75 & 0.75 & 0.5 & 0.75 \\
0.5 & 0.5 & 1.0 & 0.75 & 0.75 & 0.0 & 0.0 & 0.0 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.75 \\
0.5 & 0.5 & 0.5 & 0.5 & 0.25 & 0.25 & 0.25 & 0.25 & 0.75 & 0.75 & 0.5 & 1.0 & 1.0 & 1.0 \\
0.5 & 1.0 & 0.5 & 0.5 & 0.75 & 0.5 & 0.5 & 0.75 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\
0.5 & 0.5 & 0.5 & 0.5 & 0.75 & 0.0 & 0.0 & 0.0 & 0.5 & 0.75 & 1.0 & 0.5 & 0.5 & 0.75 \\
0.5 & 0.5 & 0.75 & 1.0 & 0.25 & 0.25 & 0.5 & 0.25 & 0.75 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\
0.5 & 0.5 & 0.5 & 0.25 & 0.25 & 0.75 & 1.0 & 0.25 & 0.75 & 0.5 & 0.75 & 0.5 & 0.5 & 0.75 \\
0.25 & 0.5 & 0.5 & 0.25 & 0.75 & 0.25 & 0.25 & 0.25 & 0.5 & 0.5 & 0.5 & 0.5 & 0.75 & 1.0 \\
1.0 & 0.5 & 0.25 & 0.25 & 0.25 & 0.0 & 0.0 & 0.0 & 0.5 & 0.5 & 0.75 & 0.5 & 0.5 & 0.5 \\
0.5 & 0.5 & 0.5 & 0.25 & 0.25 & 0.5 & 0.5 & 0.75 & 0.5 & 1.0 & 0.75 & 0.5 & 0.5 & 0.5
\end{pmatrix}$$

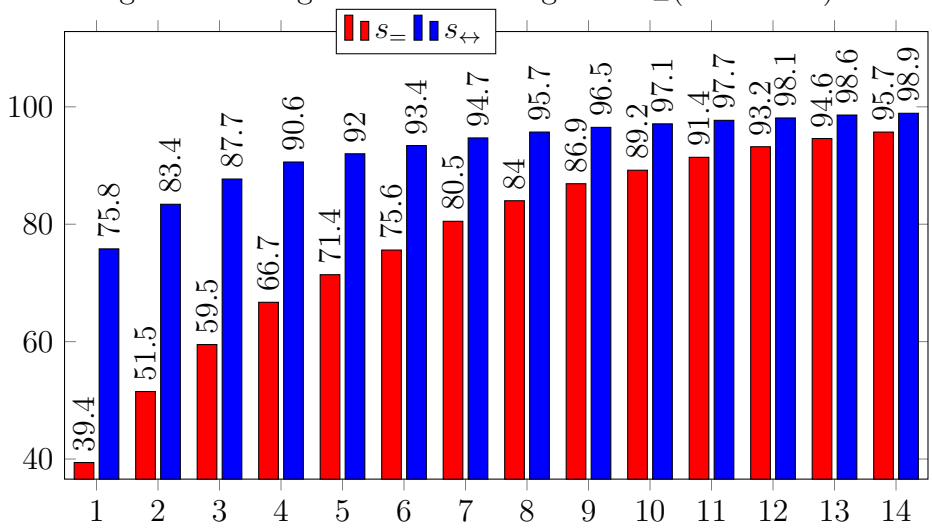
The coverage scores of individual factors is as follows.

Figure 9: Individual factors coverage for $\mathcal{F}_=$(Education)



From the individual factors coverage score graph showed in Figure 9, we can observe that the first factor is very dominant as it precisely explains nearly 40% of the input. The second and fourth factors also accurately explain more than 20% of the data, but the others are less significant when standing alone. Let us also show the merged coverage percentage and then discuss the whole output.

Figure 10: Merged factors coverage for $\mathcal{F}_{=}$(Education)



In the merged coverage, which is depicted in Figure 10, we can observe that biresiduum-wise, the first four factors together explain more than 90% of the input. This means that 90% of the input entries are nearly covered, but we can also observe that exactly covered is just 66.7%. After this point, the increase in biresiduum coverage is less significant, and the 14 factors together explain more than 95% of the input measured by biresiduum and also exact similarity.

7.2.3 Performance of L-Grecond equipped $s_{\leftrightarrow}$

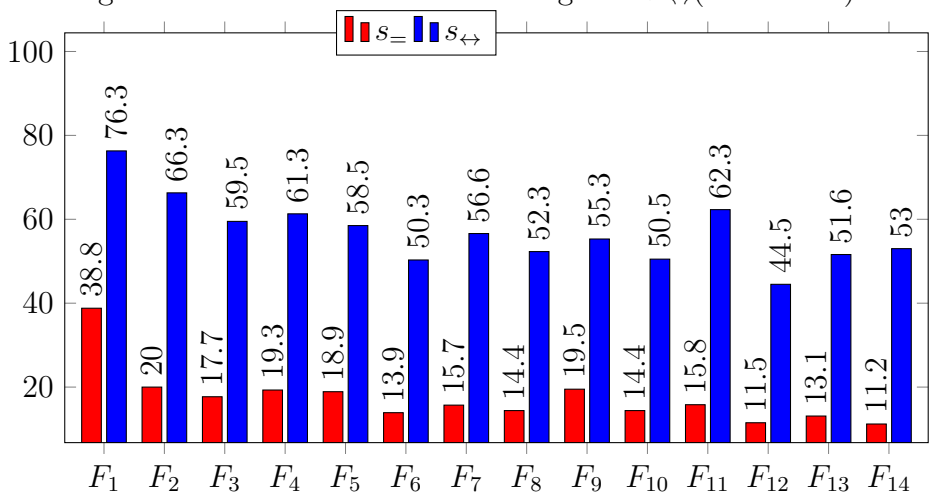
The algorithm equipped with the similarity function $s_{\leftrightarrow}$ found for the input context $\mathcal{I}$ a set $\mathcal{F}_{\leftrightarrow} = \{F_i = \langle C_i, D_i \rangle | 1, ..., 31\}$ of 31 L-concepts. Let us show the matrix $B_{\mathcal{F}_{\leftrightarrow}}$ which corresponds to the 14 intents of the $\mathcal{F}_{\leftrightarrow}$.

$$B_{\mathcal{F}_{\leftrightarrow}} = \begin{pmatrix}
0.5 & 0.5 & 0.5 & 0.5 & 0.75 & 0.5 & 0.5 & 0.25 & 0.75 & 0.75 & 0.75 & 0.75 & 0.75 & 0.75 \\
0.75 & 0.75 & 1.0 & 0.75 & 0.5 & 1.0 & 0.75 & 0.75 & 0.75 & 0.75 & 1.0 & 0.5 & 0.5 & 0.75 \\
0.75 & 1.0 & 0.5 & 0.5 & 0.25 & 0.5 & 0.5 & 0.75 & 0.75 & 0.75 & 0.75 & 0.5 & 0.5 & 0.75 \\
0.5 & 0.5 & 0.75 & 1.0 & 0.5 & 0.75 & 1.0 & 1.0 & 1.0 & 0.75 & 0.75 & 0.75 & 0.75 & 1.0 \\
1.0 & 0.5 & 0.75 & 0.75 & 0.25 & 0.25 & 0.25 & 0.25 & 1.0 & 1.0 & 1.0 & 0.75 & 0.75 & 0.75 \\
0.5 & 0.5 & 1.0 & 1.0 & 0.75 & 0.25 & 0.25 & 0.25 & 0.75 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\
0.5 & 0.5 & 0.5 & 0.5 & 0.25 & 0.25 & 0.25 & 0.25 & 0.5 & 0.75 & 1.0 & 0.5 & 0.75 & 1.0 \\
0.5 & 0.5 & 0.5 & 0.25 & 0.25 & 1.0 & 1.0 & 0.25 & 0.75 & 0.5 & 0.75 & 0.75 & 0.5 & 0.75 \\
0.5 & 0.5 & 0.5 & 0.5 & 0.25 & 0.25 & 0.25 & 0.25 & 0.75 & 0.75 & 0.5 & 1.0 & 1.0 & 1.0 \\
0.5 & 0.25 & 0.5 & 0.25 & 0.25 & 0.0 & 0.0 & 0.0 & 0.75 & 1.0 & 1.0 & 0.5 & 0.5 & 0.5 \\
0.25 & 0.25 & 0.5 & 0.25 & 0.75 & 0.5 & 0.25 & 0.75 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\
1.0 & 1.0 & 0.5 & 0.5 & 0.25 & 0.0 & 0.0 & 0.0 & 0.5 & 0.5 & 0.75 & 0.5 & 0.5 & 0.5 \\
0.5 & 0.5 & 1.0 & 0.75 & 0.25 & 0.0 & 0.0 & 0.0 & 0.5 & 0.5 & 0.5 & 0.75 & 0.5 & 0.5 \\
0.5 & 0.25 & 0.5 & 0.25 & 0.25 & 0.25 & 0.5 & 0.25 & 1.0 & 0.75 & 0.75 & 0.75 & 0.5 & 0.5
\end{pmatrix}$$

If we compare the $B_{\mathcal{F}_{\leftrightarrow}}$ with $B_{\mathcal{F}_{=}}$, we can find out that the extents are quite different. Let us check how these factors are performing in the coverage field.

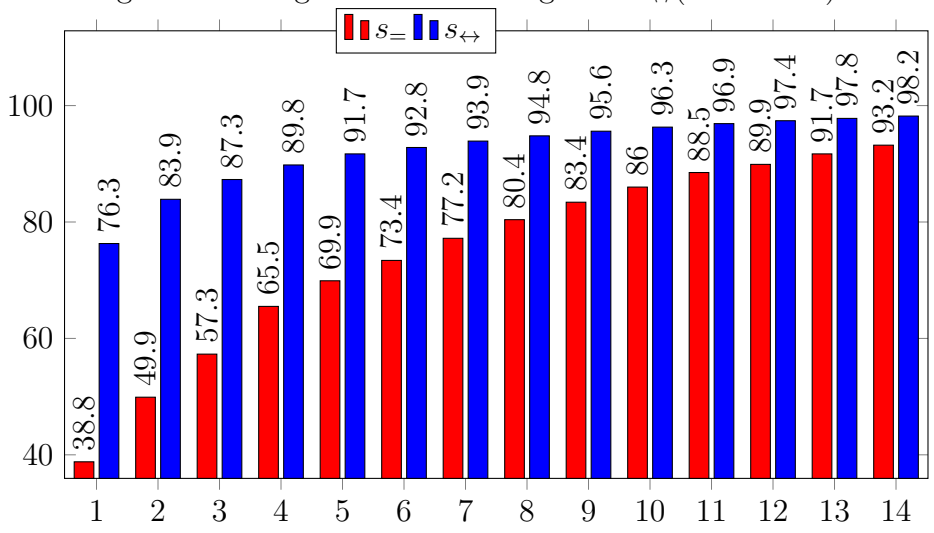
From the individual factors coverage graph, we can assume that even though the

Figure 11: Individual factors coverage for $\mathcal{F}_{\leftrightarrow}$(Education)



factors found are different, in the coverage field, they perform similarly. We will check the difference between the most dominant factors more thoroughly later in this chapter. Figure 12 showed us that the factors generated by our method

Figure 12: Merged factors coverage for $\mathcal{F}_{\leftrightarrow}$(Education)



equipped with $s_{\leftrightarrow}$, perform a little bit worse than factors generated with the help of $s_=$. The fact that we obtained different and a little bit worse set of factors may not mean that these factors are less valuable, and we will try to compare the value of the most dominant ones at the end of this chapter.

7.2.4 Performance of L-Grecond equipped with $s_{f\leftrightarrow}$

While using the L-Grecond equipped with $s_{f\leftrightarrow}$, it is important to set the parameter q of the function f correctly, to obtain a desired output. As the parameter can tune the algorithm factors preference between $s_=$ and $s_\leftrightarrow$, we have set q to be 0.075 for our tests. This setting causes the algorithm preference of factors to be somewhere in the "middle" between $s_=$ and $s_\leftrightarrow$, and therefore we obtain the different output. The algorithm equipped with the similarity function $s_f \leftrightarrow$ and $q = 0.075$ found for the input context $\mathcal{I}$ a set $\mathcal{F}_{f\leftrightarrow} = \{F_i = \langle C_i, D_i\rangle | 1, ..., 30\}$ of 30 concepts (factors). Matrix $B_{\mathcal{F}_f\leftrightarrow}$ shows us the first 14 most important intents of factors from $\mathcal{F}_{f\leftrightarrow}$.

$$B_{\mathcal{F}_{f\leftrightarrow}} = \begin{pmatrix}
0.75 & 0.75 & 0.75 & 0.75 & 1.0 & 1.0 & 0.75 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\
0.75 & 0.75 & 1.0 & 0.75 & 0.25 & 0.5 & 0.75 & 0.25 & 0.75 & 0.75 & 0.75 & 0.5 & 0.5 & 0.75 \\
0.5 & 1.0 & 0.5 & 0.5 & 0.25 & 0.0 & 0.0 & 0.0 & 0.75 & 0.75 & 0.75 & 0.75 & 0.75 & 0.75 \\
0.5 & 0.5 & 0.75 & 1.0 & 0.5 & 0.75 & 0.5 & 0.25 & 1.0 & 1.0 & 1.0 & 0.75 & 0.75 & 1.0 \\
0.5 & 0.5 & 0.5 & 0.25 & 0.25 & 1.0 & 1.0 & 0.75 & 0.75 & 0.5 & 0.75 & 0.75 & 0.5 & 0.75 \\
1.0 & 0.5 & 0.5 & 0.5 & 0.75 & 0.0 & 0.0 & 0.0 & 0.75 & 0.75 & 1.0 & 0.5 & 0.5 & 0.50 \\
0.5 & 0.5 & 0.5 & 0.5 & 0.25 & 0.5 & 0.25 & 0.25 & 0.75 & 0.75 & 0.5 & 1.0 & 1.0 & 1.0 \\
0.5 & 0.25 & 0.5 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & 1.0 & 1.0 & 1.0 & 0.5 & 0.5 & 0.5 \\
0.5 & 0.5 & 0.5 & 0.5 & 0.75 & 0.25 & 0.25 & 0.25 & 0.5 & 0.75 & 1.0 & 0.5 & 0.75 & 1.0 \\
0.25 & 0.5 & 0.5 & 0.25 & 0.75 & 1.0 & 0.5 & 0.25 & 0.75 & 0.5 & 0.75 & 0.5 & 0.5 & 0.5 \\
0.25 & 0.25 & 0.5 & 0.25 & 0.75 & 0.5 & 0.25 & 0.75 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\
0.5 & 0.5 & 1.0 & 1.0 & 0.75 & 0.25 & 0.25 & 0.25 & 0.75 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\
1.0 & 0.5 & 0.25 & 0.25 & 0.25 & 0.0 & 0.0 & 0.0 & 0.75 & 0.5 & 0.75 & 0.75 & 0.5 & 0.75 \\
0.5 & 0.5 & 0.5 & 0.25 & 0.75 & 0.0 & 0.0 & 0.0 & 0.5 & 1.0 & 0.75 & 0.5 & 0.5 & 0.5
\end{pmatrix}$$

The intents are different from the ones obtained previously, and also the total number of needed concepts is lesser. Let us show how they compete in the coverage field. In Figure 13, we can check the coverage performance of individual factors, which is quite similar to the performance of the previous one. We can also observe that the most prominent individual coverage score has the first and fourth factors as in the output calculated by L-Grecond equipped with $s_=$.
Figure 14 shows the merged coverage percentages, which are also quite similar to the previous outputs. Most of the coverage scores are somewhere between the scores obtained by L-Grecond equipped with $s_=$ and $s_\leftrightarrow$. This can be caused by the setting of the parameter q as we wanted the preference of the algorithm to be between the two extremes $s_=$ and $s_\leftrightarrow$.

Figure 13: Individual factors coverage for $\mathcal{F}_{f\leftrightarrow}$(Education)

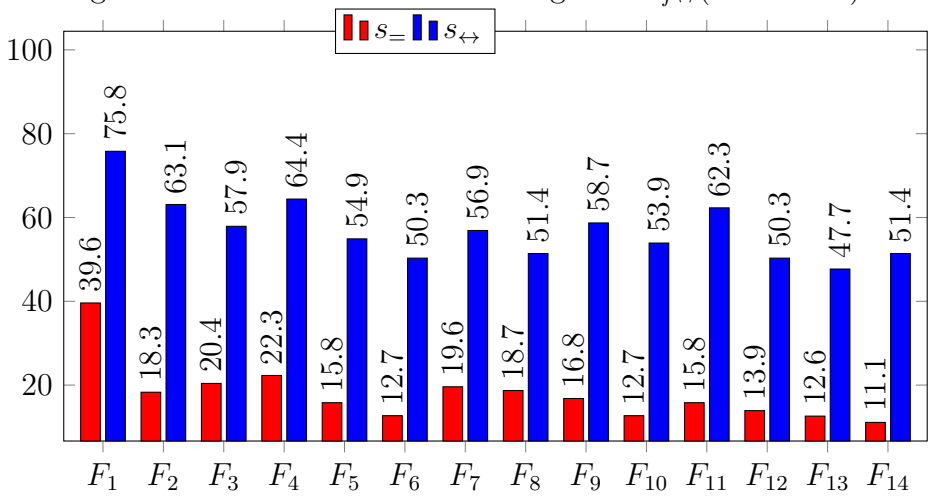
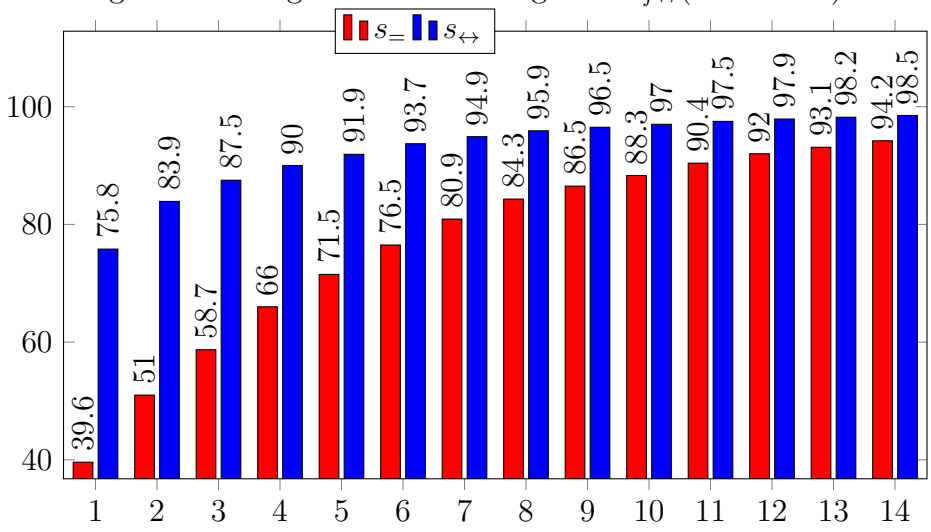


Figure 14: Merged factors coverage for $\mathcal{F}_{f\leftrightarrow}$(Education)



7.2.5 Comparison of dominant factors

In this section we will analyze and compare the first factor from each fuzzy concept sets $\mathcal{F}_{=}$, $\mathcal{F}_{\leftrightarrow}$ and $\mathcal{F}_{f\leftrightarrow}$ which were calculated by L-Grecond equipped with corresponding similarity function $s_{=}$, $s_{\leftrightarrow}$ and $s_{f\leftrightarrow}$. All of these factors explain nearly 40% of our data. We aim to check the differences of these factors and mainly to determine which one is the clearest and useful in the case of education dataset analysis. We will check the direct differences of the extents and to explore more about them; we will check how these factors apply to students with the help of intents.
To simplify marking of variables, we will label $D_1 \in \mathcal{F}_{=}$ as $D_{=}$, $D_1 \in \mathcal{F}_{\leftrightarrow}$ as $D_{\leftrightarrow}$ and $D_1 \in \mathcal{F}_{f\leftrightarrow}$ as $D_{f\leftrightarrow}$. In Table 5, we can check the difference between these intents.

Table 5: Intents comparison (Education)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$D_{=}$	0.75	0.75	1.0	0.75	1.0	0.75	0.75	1.0	1.0	1.0	1.0	1.0	1.0	1.0
$D_{\leftrightarrow}$	0.5	0.5	0.5	0.5	0.75	0.5	0.5	0.25	0.75	0.75	0.75	0.75	0.75	0.75
$D_{f\leftrightarrow}$	0.75	0.75	0.75	0.75	1.0	1.0	0.75	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Elements of intent $D_{=}$ are close to $D_{f\leftrightarrow}$, while the $D_{\leftrightarrow}$ is a far more different. $D_{=}$ and $D_{f\leftrightarrow}$ are different just on attributes $3, 6$, and they have most of the attributes present in a high degree. Conversely, the intent $D_{\leftrightarrow}$ does not have any attribute in degree 1.0, and it tends to have all of the attributes in the "middle" degree.
Let us talk about how these factors apply to our 607 students. $D_{=}$ applies to 395 students in degree 0.75, to 121 in degree 0.5, to 90 in 0.25, and one student does not have this factor at all. Therefore this factor applies to the students in degree 0.6248 on average.
$D_{\leftrightarrow}$ applies to 479 students in degree 1.0, and the rest has it in degree 0.75 or 0.5. An average degree is 0.9324 in this case.
$D_{f\leftrightarrow}$ applies to the students quite similarly as $D_{=}$, while the average degree is 0.6244.

This information together leads us to the conclusion that factor $D_{\leftrightarrow}$ can be considered as a "flat" factor. Although it describes a reasonable amount of input, it aims to cover the lesser degrees while just partially covering the higher degrees. In the education data, we wanted to find factors that would describe the interesting hidden properties of A grade students and not the general truth, which applies to all of them. Therefore for this kind of datasets, it should be essential to decide which kind of factors are valuable for us. By tuning the parameter q of the function f while using the L-Grecond equipped with $s_{f\leftrightarrow}$, we can obtain different sets of factors, while tuning between the exact similarity and biresiduum similarity extremes.

7.3 Elections data

A dataset which we will analyze in this section contains data about the elections in the Czech Republic in the year 2013. The raw dataset can be obtained on https://www.volby.cz/opendata/ps2013/ps2013_opendata.htm. This election dataset contains information about votes for political parties. The rows in the dataset represent the districts and the columns the information for the particular district. The raw dataset consists of 6336 districts, and for each district, it stores information about the number of people who can vote in the district, the number of people who participated in the elections, and mainly the number of votes for all of the major parties. The major parties are *CSSD*, *TOP09*, *ODS*, *KDUCSL*, *SPD*, *ANO*, *KSCM*, the rest of insignificant parties are aggregated as one column and we will omit them. Examples of a few rows from the dataset are shown in Table 6.

Table 6: Election data

	CSSD	*TOP09*	*ODS*	*KDUCSL*	*SPD*	*ANO*	*KSCM*
Praha 1	1410	4260	2011	1067	327	1671	692
Praha 2	2431	5295	2843	1290	524	2799	1188
Praha 3	4287	7536	3774	1637	945	4450	2584
Olomouc	10104	5714	4141	3598	3224	9370	6086
Bouzov	164	64	29	83	61	105	182
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

From the perspective of FCA we will talk about the context $\mathcal{I} = \langle X, Y, I \rangle$, where X are the districts, Y the political parties and I, which we will construct in the next section, will be the fuzzy relation between districts and parties based on the votes earned.
This dataset is complicated as it contains tiny areas with just a few votes and significant areas with thousands of votes. Therefore there are a lot of possible ways how to process this dataset. One possibility is to filter the areas by the number of people and analyze just a particular part of the dataset, for example, just big or small districts. We will deal with the whole dataset without any filtering. There are also a lot of ways of the raw data transformation to a suitable form for our algorithm. Let us continue with the data transformation section, where we will introduce a few possible ways of data transformation.

7.3.1 Data transformation

As the raw data of the dataset are not suitable to be used as input for our algorithm, we have to transform them so the matrix entries will be in $L \in \{0, 1\}$. In this section, we will present three different data transformation methods,

which will also have different scales L.
First, the most straightforward method will directly transform the percentage of votes eleven-element scale $L = \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$. For each row, it summarizes the number of votes and calculates the percentage of votes for each major political party. Then we will round the percentage to the nearest value of L. The transformation of data from Table 6 is shown in Table 7.

Table 7: Election data transformation 1

	CSSD	*TOP09*	*ODS*	*KDUCSL*	*SPD*	*ANO*	*KSCM*
Praha 1	0.1	0.3	0.1	0.1	0.0	0.1	0.0
Praha 2	0.1	0.3	0.1	0.1	0.0	0.1	0.1
Praha 3	0.1	0.2	0.1	0.1	0.0	0.1	0.1
Olomouc	0.2	0.1	0.1	0.1	0.1	0.2	0.1
Bouzov	0.2	0.1	0.0	0.1	0.1	0.1	0.2
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Although this transformation is justifiable and straightforward, we will not use it for experiments as it has just low grades across the dataset and in general the output matrix is sparse.
Let us continue with the next transformation approach, which has entirely different strategy and calculates the truth grades for each row individually. The transformation process starts with the selection of the worst and best placed political parties in the district, and it will use a five-element scale $L = \{0.0, 0.25, 0.5, 0.75, 1.0\}$. After the selection of the worst and best column, it will perform a linear transformation from interval $[worst, best]$ to $[0, 1]$ and round the particular values to the closest elements of scale L. As an example, we can use *Praha 1*. TOP09 achieved the best score through major parties with 4260 votes and the worst by SPD with 327 votes. The value for each of the columns of *Praha 1* can be obtained by

$$RowValue_{Praha1}(x) = \frac{x - 327}{4260 - 327} = \frac{x - 327}{3933} \in [0, 1].$$

Therefore we will obtain 0.0 for SPD, 1.0 for TOP09 and value for other parties like ODS can be calculated as

$$RowValue_{Praha1}(ODS) = \frac{2011 - 327}{3933} = 0.4281$$

which is rounded to $0.5 \in L$. The transformation of the data from Table 6 is shown in Table 8.
The third approach for election dataset transformation is the most complex one but has the most interesting outputs among our testings. This approach deals with the whole dataset, not the individual rows. First, it transforms all of the votes to percentages, which are calculated concerning the district. Then it finds

Table 8: Election data transformation 2

	CSSD	*TOP09*	*ODS*	*KDUCSL*	*SPD*	*ANO*	*KSCM*
Praha 1	0.25	1.0	0.5	0.25	0.0	0.25	0.0
Praha 2	0.5	1.0	0.5	0.25	0.0	0.5	0.25
Praha 3	0.5	1.0	0.5	0.0	0.0	0.5	0.25
Olomouc	1.0	0.25	0.25	0.0	0.0	1.0	0.5
Bouzov	1.0	0.25	0.0	0.25	0.25	0.5	1.0

the best percentage result among all districts and political parties. This particular value could be distorted by some small districts with the people voting for just one party, so we did not choose the best result directly but by aggregation of the top 100 results. Therefore the best value in our case is 37%, which is a reasonable value for the excellent performance of the political party. We will use five-element scale $L = \{0.0, 0.25, 0.5, 0.75, 1.0\}$. The process of assigning corresponding $a \in L$ for particular x is defined as assigning the a for which $a \cdot 0.37$ is the closest to x. For $L = \{0.0, 0.25, 0.5, 0.75, 1.0\}$ and $a \in L$ the values $a \cdot 0.37$ are $\{0, 0.09, 0.18, 0.28, 0.37\}$.
The transformation of the data from Table 6 is shown in Table 9.

Table 9: Election data transformation 3

	CSSD	*TOP09*	*ODS*	*KDUCSL*	*SPD*	*ANO*	*KSCM*
Praha 1	0.25	0.75	0.5	0.25	0.0	0.25	0.25
Praha 2	0.25	0.75	0.5	0.25	0.0	0.5	0.25
Praha 3	0.5	0.75	0.25	0.25	0.0	0.5	0.25
Olomouc	0.5	0.25	0.25	0.25	0.25	0.5	0.25
Bouzov	0.5	0.25	0.0	0.25	0.25	0.25	0.5

For testing of performance of our decomposition algorithm, we will use just the data transformed by the third approach as it has the clearest outputs and it is agreed to be the most reasonable transformation which preserves the dataset features and hidden factors.
In the next sections, we will test our algorithm on the election dataset with all of its 6366 rows, which are transformed the same way as the data, which are shown in Table 9.

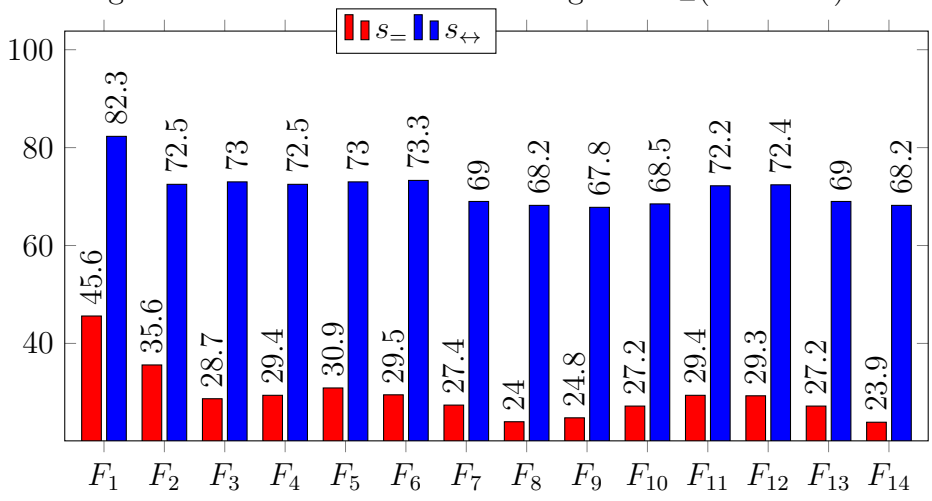
7.3.2 Performance of L-Grecond equipped with $s_=$

Input for the algorithm is $\mathcal{I} = \langle X, Y, I \rangle$ where X are the districts, Y are the rows, and fuzzy relation $I \in L^{6365 \times 7}$ is the dataset transformed by the third approach mentioned in the data transformation section. Part of fuzzy relation I is depicted in Table 9. Algorithm equipped with the similarity function $s_=$ found $\mathcal{F}_= = \{F_i = \langle C_i, D_i \rangle | 1, ..., 14\}$ of 14 concepts. As the extent $A_{\mathcal{F}_=}$ is big, we will omit it in the text, and we will just show the matrix $B_{\mathcal{F}_=}$.

$$B_{\mathcal{F}_=} = \begin{pmatrix} 1.00 & 0.75 & 0.50 & 0.50 & 0.75 & 1.00 & 1.00 \\ 1.00 & 1.00 & 1.00 & 0.50 & 1.00 & 0.50 & 0.50 \\ 1.00 & 0.25 & 0.25 & 0.75 & 0.50 & 0.25 & 0.25 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.25 & 1.00 & 0.00 \\ 0.50 & 0.00 & 0.00 & 0.00 & 0.50 & 0.00 & 1.00 \\ 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 1.00 & 0.50 & 0.25 & 0.00 & 0.25 & 0.00 \\ 0.00 & 0.00 & 0.00 & 1.00 & 0.25 & 0.00 & 0.00 \\ 0.00 & 0.25 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.50 & 0.25 & 0.00 & 0.00 & 1.00 & 0.25 & 0.25 \\ 0.25 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 \\ 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 \end{pmatrix}$$

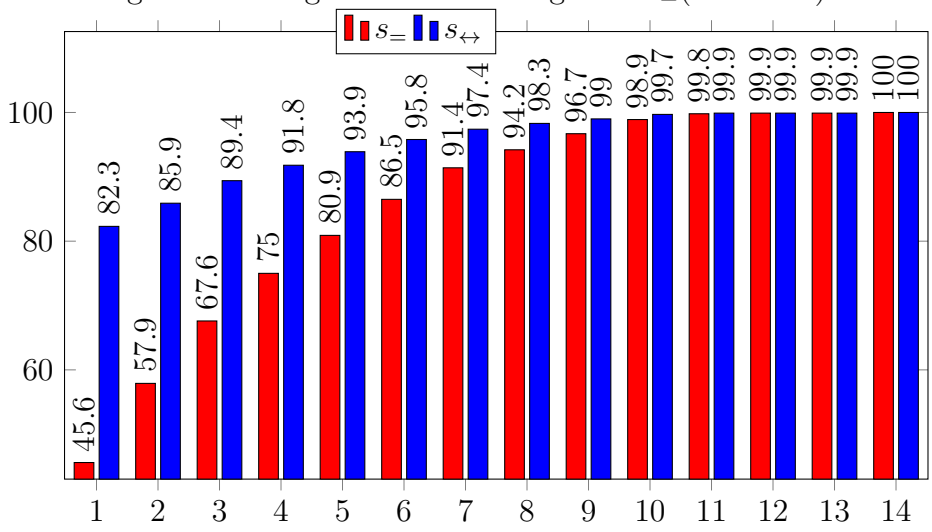
As we can see, the last few concepts are not useful as they are dealing just with one or two attributes; hence these factors are not genuinely hidden attributes of the dataset, and we can consider them as not so unusual. Let us continue with the coverage scores of individual factors.

Figure 15: Individual factors coverage for $\mathcal{F}_=$(Elections)



Let us also show the merged coverage percentage and then discuss the output.

Figure 16: Merged factors coverage for $\mathcal{F}_{=}$(Elections)



From the coverage graphs, we can observe that the first two concepts are dominant, and they exactly explain almost 60% of the input. The extent of the first factor consists of the strong votes for *CSSD* and strong votes for *ANO*, which also corresponds with the final result of the elections. In next section we will check how our decomposition algorithm processes the input with the usage of another similarity function.

7.3.3 Performance of L-Grecond equipped with $s_{\leftrightarrow}$

Algorithm equipped with the similarity function $s_{\leftrightarrow}$ found $\mathcal{F}_{\leftrightarrow} = \{F_i = \langle C_i, D_i \rangle | 1, ..., 14\}$ also of 14 concepts. Let us also omit the extent $A_{\mathcal{F}_{=}}$ and just show the intent $B_{\mathcal{F}_{\leftrightarrow}}$.

$$B_{\mathcal{F}_{\leftrightarrow}} = \begin{pmatrix} 1.00 & 0.75 & 0.50 & 0.50 & 0.75 & 1.00 & 1.00 \\ 1.00 & 0.25 & 0.25 & 0.75 & 0.75 & 0.50 & 0.25 \\ 0.50 & 1.00 & 1.00 & 0.50 & 0.50 & 1.00 & 0.50 \\ 0.25 & 0.00 & 0.00 & 0.00 & 0.50 & 1.00 & 0.25 \\ 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.75 \\ 0.25 & 0.00 & 0.00 & 0.00 & 0.25 & 0.00 & 1.00 \\ 0.00 & 0.25 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.50 & 0.25 & 0.00 & 0.00 & 1.00 & 0.25 & 0.25 \\ 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.25 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 \\ 0.25 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 \\ 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 \end{pmatrix}$$

We can observe that the first factor is the same as for the $\mathcal{F}_{=}$. If we check the factors in more detail, we will find out that also $F_8, F_9, F_{11}, F_{12}, F_{13}, F_{14}$ are the same, but most of them were obtained in another order. Let us check the coverage score performance. The individual coverage is shown in Figure 17, and the merged coverage score is shown in Figure 18. In the case of $\mathcal{F}_{=}$, the first two

Figure 17: Individual factors coverage for $\mathcal{F}_{\leftrightarrow}$(Elections)

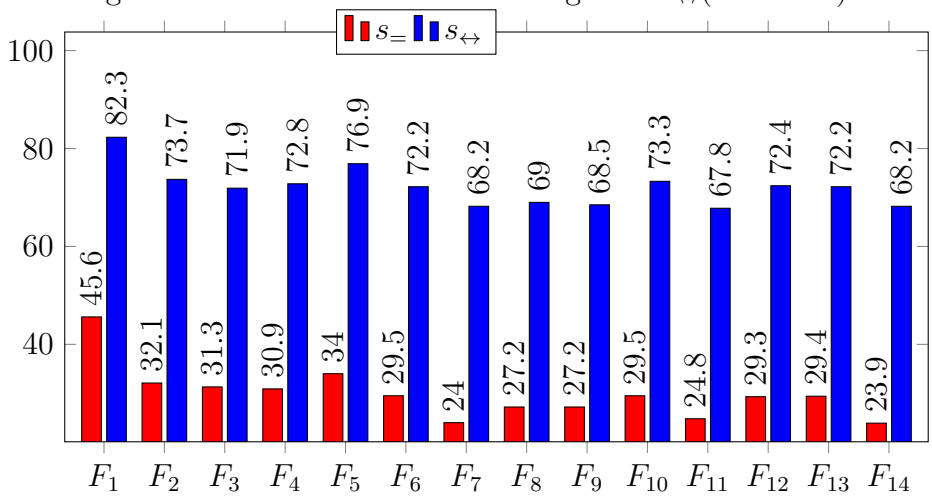
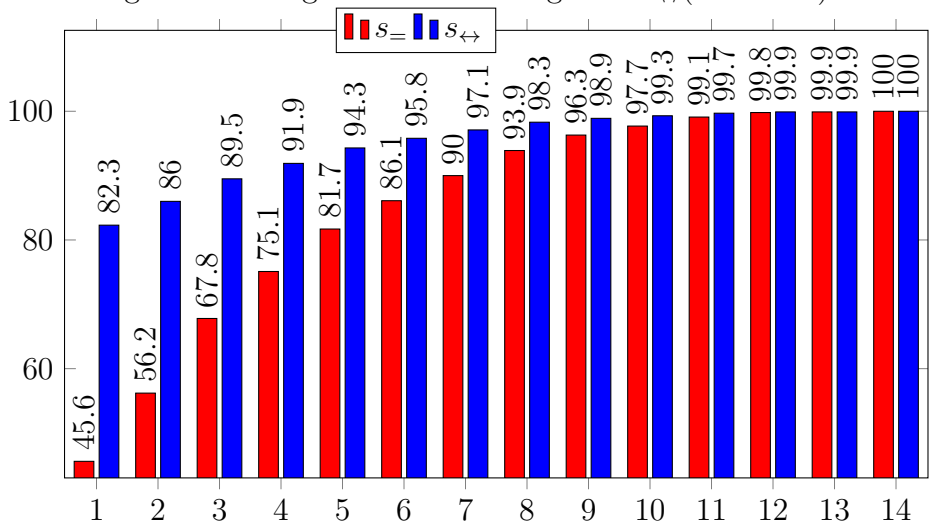


Figure 18: Merged factors coverage for $\mathcal{F}_{\leftrightarrow}$(Elections)



are more dominant than the others in the coverage performance. In this case, $F_2, ..., F_6$ are almost similar in their performance, and just F_1 is dominant. We will discuss the differences later at the end of this chapter.

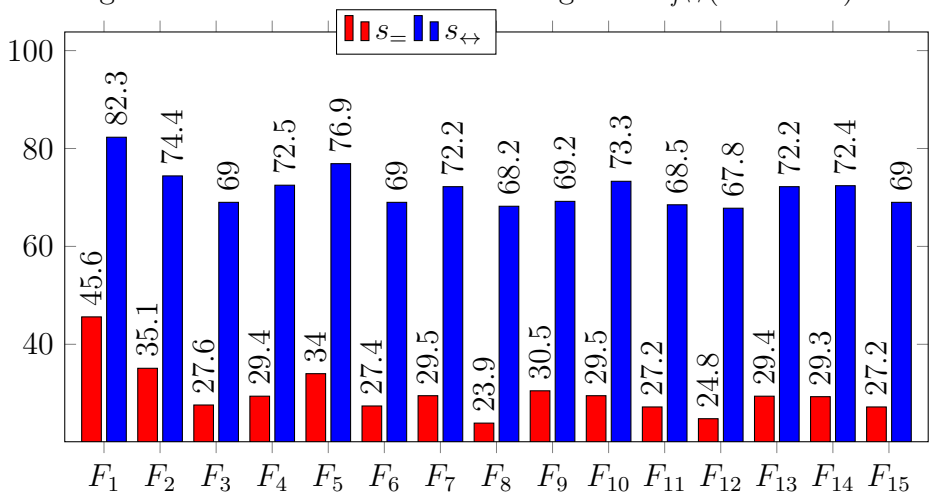
7.3.4 Performance of L-Grecond equipped with $s_{f\leftrightarrow}$

Algorithm equipped with the similarity function $s_{f\leftrightarrow}$ with the parameter $q = 0.02$ found $\mathcal{F}_{f\leftrightarrow} = \{F_i = \langle C_i, D_i\rangle | 1, ..., 15\}$ of 15 concepts. Let us also omit the extent $A_{\mathcal{F}_{f\leftrightarrow}}$ and just show the intent $B_{\mathcal{F}_{f\leftrightarrow}}$.

$$B_{\mathcal{F}_=} = \begin{pmatrix}
1.00 & 0.75 & 0.50 & 0.50 & 0.75 & 1.00 & 1.00 \\
1.00 & 0.75 & 0.75 & 0.25 & 0.25 & 0.50 & 0.25 \\
0.75 & 0.50 & 0.50 & 1.00 & 1.00 & 0.50 & 0.50 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.25 & 1.00 & 0.00 \\
1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.75 \\
0.00 & 1.00 & 0.50 & 0.25 & 0.00 & 0.25 & 0.00 \\
0.25 & 0.00 & 0.00 & 0.00 & 0.25 & 0.00 & 1.00 \\
0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 \\
0.75 & 0.50 & 1.00 & 0.50 & 1.00 & 0.50 & 0.50 \\
1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.50 & 0.25 & 0.00 & 0.00 & 1.00 & 0.25 & 0.25 \\
0.00 & 0.25 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.25 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 \\
0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00
\end{pmatrix}$$

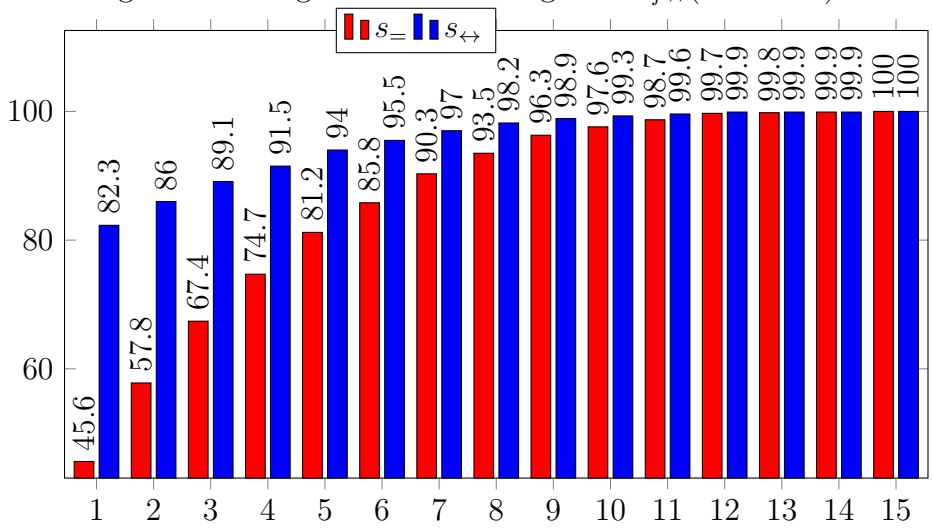
We can again observe that the first factor is the same as in previous sets $\mathcal{F}_=$ and $\mathcal{F}_\leftrightarrow$. Also, $F_4, ..F_8$ and $F_{10}, ..., F_{15}$ are present in one or both of previous sets, but most of them in a different order. Let us check the coverage score performance. The individual coverage is shown in Figure 19, and the merged coverage score is shown in Figure 20. We can observe that the coverage performance is

Figure 19: Individual factors coverage for $\mathcal{F}_{f\leftrightarrow}$(Elections)



similar to the two sets obtained before. The only noticeable difference is that

Figure 20: Merged factors coverage for $\mathcal{F}_{f\leftrightarrow}$(Elections)



the L-Grecond equipped with $s_{f\leftrightarrow}$ needed one more fuzzy concept to cover the input, but as the last seven concepts $F_9, ..., F_{15}$ together cover less than 4% of the input, it is not much important.

7.3.5 Comparison of dominant factors

In this section we will take a look at the output obtained by all variants of our algorithm and check the differences. The first factor F_1 is the same in every set $\mathcal{F}_=$, $\mathcal{F}_\leftrightarrow$ and $\mathcal{F}_{f\leftrightarrow}$. This is the sign that F_1 is a strong factor, as every variant of our algorithm gave this factor a high priority. F_1 consists of high degree for *CSSD*,*ANO* and *KSCM*, which corresponds with the final result of the elections. It is not surprising to see this kind of pattern as the most important factor in the dataset. This factor itself exactly covers more than 45% of the input, while the other factors obtained by various variants cover around 30% or less.
Let us compare the second factor, which is different across our sets of concepts. To simplify marking of variables, we will label $D_2 \in \mathcal{F}_=$ as $D_=$, $D_2 \in \mathcal{F}_\leftrightarrow$ as $D_\leftrightarrow$ and $D_2 \in \mathcal{F}_{f\leftrightarrow}$ as $D_{f\leftrightarrow}$. In Table 10 we can check the difference between their intents.

All of these intents have a strong degree in *CSSD*, where the degree of other

Table 10: Intents comparison (Elections)

	CSSD	*TOP09*	*ODS*	*KDUCSL*	*SPD*	*ANO*	*KSCM*
$D_=$	1.0	1.0	1.0	0.5	1.0	0.5	0.5
$D_\leftrightarrow$	1.0	0.25	0.25	0.75	0.75	0.5	0.25
$D_{f\leftrightarrow}$	1.0	0.75	0.75	0.25	0.25	0.5	0.25

attributes is various. Intents of $D_=$ and $D_{f\leftrightarrow}$ have a partly similar pattern, but intent of $D_\leftrightarrow$ is more different. The average degree in which the factors apply to

the districts is around 0.1269 for both $D_{=}$, $D_{f\leftrightarrow}$ and 0.3648 for $D_{\leftrightarrow}$ which is a very high average degree compared to $D_{=}$ and $D_{f\leftrightarrow}$. This can lead us to the verification if the $D_{\leftrightarrow}$ can be the so-called "flat" factor as the L-Grecond equipped with $s_{\leftrightarrow}$ tends to prefer them. When we check the actual individual coverage score of these three factors, we can observe that the biresiduum similarity coverage percentage is for the $D_{\leftrightarrow}$ lesser than for $D_{f\leftrightarrow}$, which is an interesting fact. This fact also tells us that this factor is probably not an occurrence of "flat" factor, as the biresiduum similarity score is lesser than for $D_{f\leftrightarrow}$. The greedy choices of algorithms caused this occurrence. The algorithm equipped with $s_{f\leftrightarrow}$ chose another direction during greedy steps based on $s_{f\leftrightarrow}$ scores and at the end, obtained a concept which has bigger biresiduum similarity score than the factor which was obtained by algorithm equipped with $s_{\leftrightarrow}$.
As a conclusion, we can say that the election dataset does not suffer from "flat" factors within the most important factors. Also, we can conclude that various variants of our algorithm or just tuning of parameter q used in f can lead us to obtain another set of factors that can be more interesting. In this case, the example is the second factor, where the algorithm equipped with $s_{\leftrightarrow}$ found factor with different kind of pattern than the two other variants. Further work of our election can include the usage of another data transformation and also filtering district by a certain restriction. Another interesting direction could be also to apply and measure the performance of obtained factors to filtered data, for example, small villages or big cities.

Conclusions

The aim of the thesis was to study and implement existing algorithms for formal concept analysis of ordinal data and measure their performance. The essential underlying theory was described, and we also analyzed three datasets with the different variants of the method. Regarding the method of data analysis, we described the L-Grecond algorithm, and we suggested suitable matrix similarity functions for which we also verified their correctness.

Provided dataset analysis could be a valuable resource for someone who will try to analyze ordinal data by FCA, as they give insight into the behavior of L-Grecond equipped by different matrix similarity functions and also the ideas of appropriate data transformation. Nevertheless, in general, we can not say which alternative of the L-Grecond is the most beneficial, and this decision is left on the future user. The best choice would probably always be to use a scaled similarity function and tune the parameter to obtain different points of view on a particular dataset until receiving an attractive output.

Future work within the formal concept analysis of the ordinal data could be, for example, the research of the impact while using a different adjoint couple instead of the Łukasiewicz. An interesting area could also be a comparison of the performance of the L-Grecond algorithm and the boolean Grecond algorithm, used for transformed input.

A Desktop application for L-Grecond analysis

In this chapter, we will briefly describe the application which was created for the testing of L-Grecond algorithm.
The application is written in C#, and therefore, it can be run only on computers running Windows with the .NET framework (version 4.6.1 or newer). The application can load data from a file and analyze them with the L-Grecond algorithm. Supported file formats for import of input matrix are *.mat and *.txt. As the application also supports reading *.mat files, which are Matlab files, it requires the Accord framework dynamic linked libraries to be present on the computer. Accord libraries (*.dll files) are present within the build of the application on the embedded CD.
The text files containing the input matrix have to be formatted as follows. Each row of the file represents one row of the matrix, while the matrix entries are real numbers separated by space. The decimal point of each number has to be represented by "." (dot).
After the import of the file, the user can select the similarity function, which will be used by the L-Greconds. If the input matrix is smaller than 15×15, it will be shown in the application window as a table with entries represented by squares filled with the grey color with different intensity depending on the real number of the entry. Users can also specify the parameter q, which will be used for the scaled biresiduum similarity. After the selection of parameters, users can run the computation. The progress of the calculation can not be estimated, so there is no remaining time shown. Usual calculation time for small datasets is less than a few seconds. Bigger datasets, like the mentioned education or election dataset, are usually done within 15 seconds. More extensive datasets were not analyzed with this application.
The output of the computation is a set of factors. The factors are represented by matrix A and B for which $A \circ B$ is equal to the input matrix. Smaller matrixes as 15×15 are shown in the application window. Bigger matrixes are not shown, but they can be optionally saved to a file. Let us mention that because we operate with real numbers, which does not have accurate representation in C#, the tolerance of 0.01 for the coverage of input entry was applied. The application calculates the coverage score of each factor and also the merged coverage score. The scores are calculated concerning all similarity functions within this thesis, and they are organized in the graph, which is present in the application window after the computation. The figure 21 shows the application window after the computation ended; the input was the decathlon dataset. Individual coverage scores are differentiated by the color and also the width of the line.

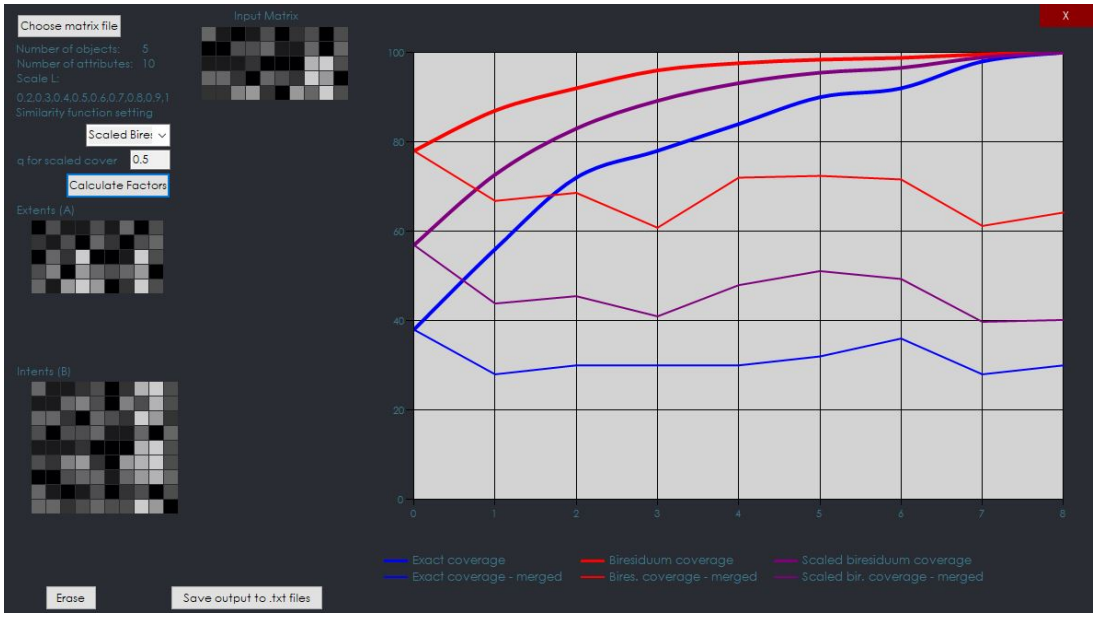


Figure 21: Application UI

The application can save the raw output to the file system. The raw output consists of three text files. The first and second files are matrixes A and B. The third output file is an info file, which contains all coverage scores for the factors, computation time, and also the location of the input and output matrixes. All output files are located in the "_FCAOutput" folder, which is created in the directory where the input file is.

B Contents of the embedded CD

Embedded CD has following directory structure:

bin/
: The bin folder contains the release build of the application for L-Grecond testing. It also contains dynamic-link libraries (*.dll), which are needed to read Matlab files. The "FuzzyFCAVisualization.exe" file runs the application.

datasets/
: Datasets folder contains Decathlon, Education, and Election dataset. It contains the raw data input, the transformed input, which was used for tests, and also the output files which were generated with the L-Grecond testing application.

src/
: Src folder contains the source codes for the L-Grecond testing application.

doc/
: Doc folder contains the pdf file with the full text of the thesis also with all Latex dependencies, which are needed to generate the pdf successfully.

readme.txt
: Instruction for the application start.

Bibliography

[1] Wille, Rudolf and Bernhard Ganter. Formal concept analysis: mathematical foundations. Translated from the German by Cornelia Franzke. Berlin: Springer-Verlag, c1999, x, 284 s. ISBN 3-540-62771-5.

[2] Belohlavek R., Vychodil V.: Discovery of optimal factors in binary data via a novel method of matrix decomposition. Journal of Computer and System Sciences 76(1)(2010), 3-20. [Elsevier Science, ISSN 0888-613X, DOI 10.1016/j.jcss.2009.05.002]Available at: http://belohlavek.inf.upol.cz/publications/BeVy_Dofbdnmmd.pdf

[3] Belohlavek, Radim. Fuzzy relational systems: foundations and principles. New York: Kluwer Academic, 2002c, xii, 369 s. ISBN 0306467771.

[4] Belohlavek, R., Krmelova, M.: Factor analysis of ordinal data via decomposition of matrices with grades. Annals of Mathematics and Artificial Intelligence 72(1)(2014), 23-44. [Springer, print ISSN 1012-2443, Online ISSN 1573-7470, doi:10.1007/s10472-014-9398-6]. Available at: http://belohlavek.inf.upol.cz/publications/BeKr-Faoddmg.pdf

[5] Belohlavek R.: Optimal decompositions of matrices with entries from residuated lattices. Journal of Logic and Computation 22(6)(2012), 1405-1425. [Oxford University Press, ISSN 0955-792X, doi: 10.1093/logcom/exr023, online: September 7, 2011]. Available at: http://belohlavek.inf.upol.cz/publications/Bel_Odmerl.pdf

[6] Belohlavek, R., Vychodil, V.: Factorization of matrices with grades. Fuzzy Sets and Systems 292(2016), 85-97. [Elsevier, ISSN 0165-0114, doi:10.1016/j.fss.2015.03.020]. Available at: http://belohlavek.inf.upol.cz/publications/BeVy_Fmg.pdf

[7] Bartl, E., Belohlavek, R., Scharaschkin, A.: Toward factor analysis of educational data. CLA 2018: Proceedings of the 14th International Conference on Concept Lattices and Their Applications, 2018, pp. 191-168. [ISBN 978–80–244–5328–6]