Similarity Based Rough Sets and its Applications in Data Mining

Egyetemi doktori (PhD) értekezés

Nagy Dávid

Témavezető:
Dr. Mihálydeák Tamás

DEBRECENI EGYETEM
Természettudományi és Informatikai Doktori Tanács
Informatikai Tudományok Doktori Iskola
Debrecen, 2020

Ezen értekezést a Debreceni Egyetem Természettudományi és Informatikai Doktori Tanács Informatikai Tudományok Doktori Iskola Elméleti számítástudomány, adatvédelem és kriptográfia programja keretében készítettem a Debreceni Egyetem természettudományi doktori (PhD) fokozatának elnyerése céljából.

Nyilatkozom arról, hogy a tézisekben leírt eredmények nem képezik más PhD disszertáció részét.

Debrecen, 2020. . . . .

a jelölt aláírása

Tanúsítom, hogy Nagy Dávid doktorjelölt 2015-2018 között az Informatikai Tudományok Doktori Iskola Elméleti számítástudomány, adatvédelem és kriptográfia programjának keretében irányításommal végezte munkáját. Az értekezésben foglalt eredményekhez a jelölt önálló alkotó tevékenységével meghatározóan hozzájárult. Nyilatkozom továbbá arról, hogy a tézisekben leírt eredmények nem képezik más PhD disszertáció részét.

Az értekezés elfogadását javasolom.

Debrecen, 2020. . . . .

a témavezető aláírása

SIMILARITY BASED ROUGH SETS AND ITS APPLICATIONS IN DATA MINING

Értekezés a doktori (Ph.D.) fokozat megszerzése érdekében
az informatika tudományágban

Írta: Nagy Dávid okleveles programtervező informatikus

Készült a Debreceni Egyetem
Informatikai Tudományok Doktori Iskolája
(Elméleti számítástudomány, adatvédelem és kriptográfia programja)
keretében

Témavezető: Dr. Mihálydeák Tamás

A doktori szigorlati bizottság:

elnök:	Dr.	.....
tagok:	Dr.	.....
	Dr.	.....

A doktori szigorlat időpontja: 20... ....

Az értekezés bírálói:

Dr.	.....
Dr.	.....
Dr.	.....

A bírálóbizottság:

elnök:	Dr.	.....
tagok:	Dr.	.....
	Dr.	.....
	Dr.	.....
	Dr.	.....

Az értekezés védésének időpontja: 20... ....

Contents

Preface

Nowadays the amount of data is growing exponentially. However, data are often incomplete or inconsistent. There can be many reasons if a value is missing. For example, it can be unknown, unassigned or even inapplicable. Inconsistency occurs when the data are contradictory. These issues can cause some undesirable events (bad prediction, inappropriate decision making, etc). In computer science, there are numerous ways to handle these kinds of inaccuracies.

Rough set theory can be considered as a rather new field in computer science. Its fundamentals were proposed by professor Pawlak in the 80's [42, 43, 45]. The pawlakian spaces handle the uncertainty among the data with a relation that is based on the indiscernibility of objects. In many cases, based on the available knowledge, two objects cannot be distinguished from each other. Two arbitrary objects can be treated as indiscernible if all of their considered, relevant properties are the same. This indiscernibility can be modeled by an equivalence relation which represents our background knowledge or its limits. It can affect the membership relation by making the judgment on this relation uncertain. It makes a set vague because a decision about a certain object has an effect on the decisions about all the objects that are indiscernible from the given object. This uncertainty can be represented by set-approximation tools. If one wants to extract as much useful information as possible from large-scale information systems, then it is inevitable to handle the indiscernibility. Rough set theory tries to answer how certain sets can be characterized or if a given object belongs to a set generated by some property.

In the last 40 years, many generalizations of the original pawlakian spaces saw the light of day. In some cases, the equivalence relation, which can usually be too strict for practical applications, is replaced with a tolerance relation representing similarity [25, 47, 44]. Many rough set models exist that are based on the probability theory [27, 46, 48, 51, 52]. Last but not least, the hybridization with fuzzy set theory needs to be mentioned[54, 19, 20].

Data mining became a very important and growing field in computer science due to the incredible increase in data. Data mining is a technique by which useful information can be extracted automatically from a large amount of data. Its goal is to search for new and useful patterns, which could otherwise remain unknown, in data repositories. Data mining methods can be applied in many areas of life. With its help, one can answer questions like: is it true that if a customer buys diapers then they will also buy beer. Naturally, not every information retrieval task can be considered as data mining. For instance, searching for records by a database system or finding certain web pages by a web search engine are tasks of the information retrieval field.

There are 3 main steps of data mining: pre-processing, knowledge discovery (data min-

ing) and post-processing. The goal of pre-processing is to convert the raw data into an appropriate format. Its basic steps contain the following: uniting the data coming from different sources, cleaning the data from noise and redundancy and choosing the records and variables that are essentials in the given task. After the first main step, the data mining algorithm gets the pre-processed data as its input. The result is a pattern, a model or sometimes one can just say "knowledge". However, these patterns can be uninterpretable or useless in their format. That is why the so-called post-processing is needed which helps the decision-making with various visualization and evaluation techniques.

Rough set theory can be crucial in data sciences [10] because handling the uncertainty is necessary in case of a large amount of data. In the field of data pre-processing, there are many methods based on rough set theory.

Discretization is a process where a continuous variable is converted to a nominal one by applying a set of cuts to the domain of the original attribute and treating each interval as a discrete value. Important rough set based discretization techniques can be seen in [31, 41].

The so-called feature selection is a process where the irrelevant features (attributes) are discarded from the system. With an increasing number of attributes, the execution time and the resource requirements of an algorithm also increase. The rough set-based feature selection methods rest on the concept reduct. Essentially, a reduct is a minimal subset of features that generates the same granulation of the universe as that induced by all features [49, 50, 30].

Another very important pre-processing task is the so-called instance selection. In data mining, the supervised learning methods (e.q., classification) divide the input dataset into two parts: training and test data. The models can be taught by the training set, while with the help of the test dataset it can be evaluated. The aim of the instance selection is to reduce the number of examples in order to bring down the size of the training set. As a result, a new training set can be obtained by which the efficiency of the system can be improved. A rough set-based technique is described in [12].

Naturally, rough set theory can be applied not only in pre-processing. In many data mining techniques, it proved to be very useful. For example decision rule induction [26, 28], association rule mining [29, 17], clustering [18] etc.

Pawlakian spaces rely on an equivalence relation which represent indiscernibility. As a generalization of these spaces, some approximation spaces have appeared that are not based on an equivalence relation but on a tolerance relation that represents similarity. These spaces preserve the property of the Pawlakian space that the union of the base sets gives out the universe. However, they give up the requirement that the base sets are pairwise disjoint. The base sets are generated in a way where for each object, the objects that are similar to the given object, are taken. This means that the similarity to a given object is considered. In the worst case, it can happen that the number of base sets equals those of objects in the universe. This significantly increases the computational resource need of the set approximation process and limits the efficient use of them in large databases. To overcome this problem, a possible solution is presented in this dissertation. The space is called similarity based rough sets [36] where the system of base sets is generated by the correlation clutstering. Therefore, the real similarity is taken into consideration not the similarity to a distinguished object. The space generated this way, on the one hand, represents the interpreted similarity properly and on the other hand, reduces the number of base sets to a manageable size. This work deals with the properties and applicability of this space, presenting all the advantages that can be gained from the correlation clustering. The structure of the dissertation is the following. In

chapter 1, the fundamentals of rough set theory and some of the main types of approximation spaces are presented. In the second chapter, the similarity based rough sets approximation space is introduced. In this chapter, some of its tools and improvements are also presented [37, 35, 6, 5, 7]. Similarity based rough sets was applied to graphs so with its help graphs can be approximated as well. This work is presented in chapter 4. This method can be used in the field of feature selection.

Chapter 1

Theoretical Background

In this chapter, the basic notations and techniques are introduced which are the basis of this dissertation. In the first subsection, the fundamentals of rough set theory is presented and then some of its possible generalizations. The collected definitions and methods are not part of the work of the author of this dissertation. They are merely required to understand the ideas presented in this dissertation.

In practice, a set is a collection of objects and it is uniquely identified by its members. It means that if one would like to decide, whether an object belongs to this set, then a precise answer can be given which is yes or no. A good example is the set of numbers which are divisible by 3 because it can be decided if an arbitrary number is divisible by 3 or not. Of course, it is required that one knows how to use the modulo operation. This fact can be considered as a background knowledge and it allows someone to decide if a number belongs to the given set. Naturally, not everybody knows how to use the modulo operation for each number. Some second graders may not be able to divide numbers greater than 100. They would not be able to decide if 142 is divisible by 3. For them, 142 is neither divisible nor indivisible by 3. So there is some uncertainty (vagueness) based on their background knowledge. Rough set theory was proposed by professor Pawlak in 1982 [42]. The theory offers a way to handle vagueness determined by some background knowledge. Each object of a universe can be described by a set of attribute values. If two objects have the same known attribute values, then these objects cannot be distinguished. The indiscernibility generated in this way is the mathematical basis of rough set theory.

Definition 1.1. *A general approximation space is an ordered 5-tuple* $\langle U, \mathfrak{B}, \mathfrak{D}, \mathsf{l}, \mathsf{u} \rangle$ *where:*

- $U \neq \varnothing$ *is the universe of objects*
- $\mathfrak{B}$ *is the set of base sets for which the following properties hold:*
  - $\mathfrak{B} \neq \varnothing$
  - *if* $B \in \mathfrak{B}$ *then* $B \subseteq U$ *and* $B \neq \varnothing$
- $\mathfrak{D}$ *is the set of definable sets which can be given by an inductive definition whose base is* $\{\varnothing\} \cup \mathfrak{B}$
- $\mathsf{l}, \mathsf{u} : 2^U \to \mathfrak{D}$ *form an approximation pair*

Definition 1.2. *The set of definable sets* $\mathfrak{D}$ *can be given by the following inductive definition.*

1. $\mathfrak{B} \subseteq \mathfrak{D}$*;*
2. $\varnothing \in \mathfrak{D}$*;*
3. *if* $D_1, D_2 \in \mathfrak{D}$*, then* $D_1 \cup D_2 \in \mathfrak{D}$*.*

The members of the Boole algebra generated by $\mathfrak{B}$ *appear as definable sets.*

The set $\mathfrak{D}$ defines how the background knowledge represented by the base sets can be used.

Many interesting properties can be checked on approximation pairs. Here, the following properties are examined (full description can be seen in [14, 21]):

Definition 1.3.

- *Monotonicity:* l *and* u *are said to be monotone if* $S \subseteq S'$ *then* $\mathsf{l}(S) \subseteq \mathsf{l}(S')$ *and* $\mathsf{u}(S) \subseteq \mathsf{u}(S')$
- *Weak approximation property:* $\forall S \in 2^U : \mathsf{l}(S) \subseteq \mathsf{u}(S)$
- *Strong approximation property:* $\forall S \in 2^U : \mathsf{l}(S) \subseteq S \subseteq \mathsf{u}(S)$
- *Normality of* l*:* $\mathsf{l}(\varnothing) = \varnothing$
- *Normality of* u*:* $\mathsf{u}(\varnothing) = \varnothing$

Definition 1.4. *The functions* l, u *form a Pawlakian approximation pair* $\langle \mathsf{l}, \mathsf{u} \rangle$ *if the followings are true for an arbitrary set* S*:*

1. $\mathsf{l}(S) = \bigcup \{B \mid B \in \mathfrak{B} \text{ and } B \subseteq S\}$*;*
2. $\mathsf{u}(S) = \bigcup \{B \mid B \in \mathfrak{B} \text{ and } B \cap S \neq \varnothing\}$*.*

In this dissertation, only Pawlakian approximation pairs are used.

Theorem 1.5. *All of the properties described in definition 1.3. are true for The Pawlakian approximation pair*

Definition 1.6.
A general approximation space is a Pawlakian approximation space if the set of base sets is the following: $\mathfrak{B} = \{B \mid B \subseteq U,$ *and* $x, y \in B$ *if* $x\mathcal{R}y\}$*, where* $\mathcal{R}$ *is an equivalence relation on* U*. The approximation pair is a Pawlakian approximation pair.*

A Pawlakian approximation space [42, 43, 45] can be characterized as an ordered pair $\langle U, \mathcal{R}\rangle$ where U is the same as in the case of a general approximation space. $\mathcal{R}$ is an equivalence relation on U. $\mathcal{R}$ is an equivalence relation based on the indiscernibility of objects. The system of base sets represents the background knowledge or its limit. The functions l and u give the lower and upper approximation of a set. The lower approximation contains objects that surely belong to the set, and the upper approximation contains objects that possibly belong to the set.

A rough set can be defined in several ways. In a general sense, it can be treated as an orthopair [13] which means it is a pair of sets such that the two sets are disjoint. The other possible way is when it is considered as a pair of sets such that one of them is a subset of the other. In the case of approximation spaces, both definitions are commonly used in the literature. If the rough set corresponding to the set S is considered as an orthopair, then it is defined as $\left\langle \mathsf{l}(S), \mathsf{l}(\overline{S})\right\rangle$, where $\overline{S}$ is the complement of S. In the other case, it can be given as $\langle \mathsf{l}(S), \mathsf{u}(S)\rangle$.

Definition 1.7. *The set* $NEG(S) = \mathsf{l}(\overline{S})$ *is called the negative region of the set* S*.*

A rough set can also be characterized numerically by the accuracy of the approximation.

Definition 1.8. *Let* U *be finite, then* α_S *is called the accuracy of the approximation (*$|\ |$ *denotes the cardinality),*

$$\alpha_S = \frac{|\mathsf{l}(S)|}{|\mathsf{u}(S)|}$$

Naturally $0 \leq \alpha \leq 1$.

Definition 1.9. *A set* S *is crisp if* $\alpha_S = 1$*.*

In many real-world applications, information on objects are stored in datasets or databases. Datasets can be given by an information system, which is a pair $IS = (U, A)$, where U is a set of objects called the universe and A is a set of attributes. Let $a : U \to V_a$ be a function, where V_a denotes the domain of attribute a. An information system can be represented by a table, where each row contains data on an entity of the universe, and the columns represent the attributes. Any pair (x, a), where $x \in U$ and $a \in A$ in the table is a cell whose value is $a(x)$.

Table 1.1 shows a very simple information system containing 8 rows. Each of them represents a patient and each has 3 attributes: headache, body temperature, and muscle pain. The base sets contain patients with the same symptoms (which means they are indiscernible from each other) and it is the following:

$$\mathfrak{B} = \{\{x_1\}, \{x_2\}, \{x_3\}, \{x_4\}, \{x_5, x_7\}, \{x_6, x_8\}\}$$

Based on some background knowledge, let the patients x_1, x_2 and x_5 have the flu. Let S be the following set of these patients: $\{x_1, x_2, x_5\}$. The approximation of the set S is the following:

Table 1.1. Information System

Object	Headache	Body Temp.	Muscle Pain
x_1	YES	Normal	NO
x_2	YES	High	YES
x_3	YES	Very high	YES
x_4	NO	Normal	NO
x_5	NO	High	YES
x_6	NO	Very high	YES
x_7	NO	High	YES
x_8	NO	Very High	YES

- $\mathsf{l}(S) = \{\{x_1\}, \{x_2\}\}$
- $\mathsf{u}(S) = \{\{x_1\}, \{x_2\}, \{x_5, x_7\}\}$

Here, $\mathsf{l}(S)$ relates to patients that surely have the flu. Patient x_5 is not in the lower approximation because there is one other patient, x_7, who is indiscernible from x_5, and there is no information about, whether x_7 has the flu or not. So the base set $\{x_5, x_7\}$ can only be in the upper approximation.

The indiscernibility modeled by an equivalence relation represents a sort of limit of our knowledge embedded in an information system (or background knowledge). Indiscernibility has an effect on the membership relation. In some situation, it makes our judgment of the membership relation uncertain – making the set vague – because a decision about a given object affects the decision about all the other objects which are indiscernible from the given object.

In practical applications not only the indiscernible objects must be handled in the same way but also those that are similar to each other based on some property. (Irrelevant differences for the purpose of the given applications should not be taken into account.) Over the years, many new approximation spaces have been created as the generalization of the original Pawlakian space [33]. The main difference between these kinds of approximation spaces (with a Pawlakian approximation pair) lies in the base sets (members of $\mathfrak{B}$). Only four main kinds of approximation spaces are mentioned in this dissertation: the original Pawlakian; covering generated by a tolerance relation; general covering; general (partial).

Pawlakian approximation spaces have been generalized using tolerance relations that are based on similarity. These relations are symmetric and reflexive but not necessarily transitive.

Covering-based approximation spaces generated by tolerance relations [47] generalize Pawlakian approximation spaces in the following points.

Definition 1.10. *A general approximation space is a covering-based approximation space generated by a tolerance relation if there is a tolerance relation $\mathcal{R}$ such that $\mathfrak{B} = \{[x] \mid x \in U\}$, where $[x] = \{y \mid y \in U, x\mathcal{R}y\}$.*

In these covering spaces, a base set contains objects that are similar to a distinguished member. This means that the similarity to a given element generates the system of base sets.

General covering approximation spaces [56, 53] are not necessarily based a tolerance.

Definition 1.11. *A general approximation space is a general covering approximation space if* $\bigcup \mathfrak{B} = U$.

In these covering spaces a property generates the system of base sets.

In the case of general (partial) approximation spaces [15] the last requirement is also given up: any family $\mathfrak{B}$ of nonempty subsets of U can be a set of base sets. In these spaces also some property generates the system of base sets, but there is also a lack of information due to partiality.

Why is the system of base sets important from the theoretical point of view? Because it represents a sort of limit of our knowledge embedded in an information system. Sometimes it makes the judgment of the membership relation uncertain – thus making the set vague – because a decision about a given object affects the decision about all the other objects that are in the same base set.

The main source of uncertainty is in our background knowledge. Let S be a subset of U, and $x, y \in U$. What can be said about y with respect to x?

1. In an original Pawlakian space:

   - if $x \in \mathsf{l}(S)$ (i.e. x is surely a member of S), then $y \in S$ for all $y, x\mathcal{R}y$;
   - if $x \in \mathsf{u}(S) \setminus \mathsf{l}(S)$ (i.e. x is possibly a member of S), then y may be a member of S for all $y, x\mathcal{R}y$ (it means that there are y_1, y_2 such that $x\mathcal{R}y_1, y_1 \in S$, and $x\mathcal{R}y_2, y_2 \notin S$);
   - if $x \in \mathsf{l}(\overline{S})(= U \setminus \mathsf{u}(S))$ (i.e. x is surely not a member of S), then $y \notin S$ for all $y, x\mathcal{R}y$.

   This means that if an object x is in the lower approximation of some set S, then all the objects that are indiscernible from x are in S as well. If an object x is in the boundary region of S, then there is at least one object which is indiscernible from x and a member of S and also there is at least one object which is indiscernible from x and not a member of S. If an object x is in the negative region of S, then none the objects that are indiscernible from x are in S.

2. In a covering space generated by a tolerance relation $\mathcal{R}$:

   - if $x \in \mathsf{l}(S)$ (i.e. x is surely a member of S), then $y \in S$ for all $y, y \in [x']$ where $x' \in [x]$ and $[x'] \in \{B \mid B \in \mathfrak{B} \text{ and } B \subseteq S\}$;
   - if $x \in \bigcup(\{B \mid B \in \mathfrak{B} \text{ and } B \cap S \neq \varnothing\} \setminus \{B \mid B \in \mathfrak{B} \text{ and } B \subseteq S\})$ (i.e. x is possibly a member of S), then there is an x' and a base set $[x']$ such that $x \in [x']$, $[x'] \cap S \neq \varnothing$, $[x'] \not\subseteq S$ and y may be a member of S for all $y \in [x']$;
   - if $x \in \mathsf{l}(\overline{S})(= U \setminus \mathsf{u}(S))$ (i.e. x is surely not a member of S), then $y \notin S$ for all $y, x\mathcal{R}y$.

3. In a general covering space:

   - if $x \in \mathsf{l}(S)$ (i.e. x is surely a member of S), then there is a base set B, such that $x \in B$ and $B \in \{B \mid B \in \mathfrak{B} \text{ and } B \subseteq S\}$)therefore $y \in S$ for all $y \in B$;

- if $x \in \bigcup(\{B \mid B \in \mathfrak{B} \text{ and } B \cap S \neq \varnothing\} \setminus \{B \mid B \in \mathfrak{B} \text{ and } B \subseteq S\})$ (i.e. x is possibly a member of S), then there is a base set B such that $x \in B$, $B \cap S \neq \varnothing$ and $B \not\subseteq S$ therefore y may be a member of the set S for all $y \in B$;
- if $x \in \mathsf{l}(\overline{S})(= U \setminus \mathsf{u}(S))$ (i.e. x is surely not a member of S), then there is a base set B such that $B \cap S = \varnothing$ therefore $y \notin S$ for all $y \in B$.

4. In a general partial space:

   - if $x \in \mathsf{l}(S)$ (i.e. x is surely a member of S), then there is a base set B, such that $x \in B$ and $B \in \{B \mid B \in \mathfrak{B} \text{ and } B \subseteq S\}$)therefore $y \in S$ for all $y \in B$;
   - if $x \in \bigcup(\{B \mid B \in \mathfrak{B} \text{ and } B \cap S \neq \varnothing\} \setminus \{B \mid B \in \mathfrak{B} \text{ and } B \subseteq S\})$ (i.e. x is possibly a member of S), then there is a base set B such that $x \in B$, $B \cap S \neq \varnothing$ and $B \not\subseteq S$ therefore y may be a member of the set S for all $y \in B$;
   - if $x \in \mathsf{l}(\overline{S})$ (i.e. x is surely not a member of S), then there is a base set B such that $B \cap S = \varnothing$ therefore $y \notin S$ for all $y \in B$;
   - otherwise nothing is known about x (i.e. there is no base set B such that $x \in B$), therefore nothing can be said about y with respect to x.

In the dissertation, there is a strict distinction between the general covering spaces and the covering spaces generated by a tolerance relation. This distinction is based on the observation that, although spaces generated by tolerance relations are covering ones, not all covering space can be generated by some tolerance relation. The main reason for this is the symmetric property of the tolerance relation and that in spaces generated by the tolerance relation each base set has at least one generator object. If $\mathcal{R}$ is a tolerance relation on U, then the system of base sets is $\mathfrak{B} = \{[x] \mid x \in U\}$, where $[x] = \{y \mid y \in U, x\mathcal{R}y\}$ (x is the generator object of $[x]$). In support of this statement, the followings highlight the difference between the general and the tolerance-based covering spaces.

1. At first, it is shown that in some cases, a tolerance relation generated by a covering does not necessarily generate the same base sets. Let $\mathfrak{B}$ a general covering. The tolerance relation $R^\star$ generated by the $\mathfrak{B}$ is as follows: $xR^\star y$ if there is $B \in \mathfrak{B}$ such that $x, y \in B$.

   (a) *Base set-loss*: Suppose that $\mathfrak{B}$ has B_1, B_2 members ($B_1, B_2 \subseteq U$) such that $B_2 \subseteq B_1$. In this case, the base set B_2 is not created by the tolerance relation $R^\star$, since every member of B_2 generates at least B_1. This makes the approximation of certain sets less efficient. Generally speaking, a tolerance relation $R^\star$ (generated by a covering) cannot generate base sets, that are part of the general covering, where $B_2 \subseteq B_1$. On the left side of Fig. 1.1 some base sets of a general covering space can be seen. On the right, some members of the base sets generated by the covering based on $R^\star$ can be seen. (For the sake of simplicity, only a few base sets are shown in both figures, so the union of the base set does not give out U). In the case of the Pawlakian definition of the lower approximation of the set S, the space generated by the tolerance relation $R^\star$ contains members of B_3 but does not contain members of B_2, and the general covering space contains both the members of B_2 and B_3. This happens because B_2 does not appear at all

among the base sets generated by the tolerance relation $R^\star$. So in this case, the general covering space gave a finer approximation.

(b) *Base set-gain*: Let B_1 and B_2 be some members of the general covering $\mathfrak{B}$ such that their intersection is not empty. Such a pair does not exist only if the base sets are pairwise disjoint. In this case, however, the $R^\star$ relation (generated by the general covering space) becomes an equivalence relation and it generates the classical Pawlakian space. Let us suppose that $B_1 \cup B_2 \notin \mathfrak{B}$ and that $B_1 \cup B_2$ does not have a common member with the other members of $\mathfrak{B}$. Let $x \in B_1 \cap B_2$. Based on the $R^\star$ tolerance relation, the set of objects similar to x is $B_1 \cup B_2$. Therefore, $B_1 \cup B_2$ is a member of the system of base set generated by $R^\star$. The appearance of this base set (in the case of the Pawlakian approximation pair) significantly modifies the upper approximation, consequently, it degrades the efficiency of the approximation. On the left side of Fig. 1.2 some base sets of a general covering space can be seen. On the right, some members of the base sets generated by the covering based on $R^\star$ can be seen. (For the sake of simplicity, only a few base sets are shown in both figures, so the union of the base set does not give out U). If one wants to define the upper approximation of the set S (using a Pawlakian type approximation), then B_2 will not be a subset of the upper approximation in the case of the general covering. However, it will be a subset of the system of base sets generated by the $R^\star$ tolerance relation because $B_1 \cup B_2$ appears as a base set. So, the upper approximation became larger in the case of the space generated by $R^\star$ which increases the uncertainty relative to the set S.

2. Secondly, it is shown that there is a general covering space for which there is no tolerance relation that generates the same base sets that are the members of the general covering. Indirectly, suppose that $\mathfrak{B}$ is a general covering space and R is a tolerance relation that generates exactly the member of $\mathfrak{B}$. Let U be the universe of the general covering, $x \in U$, $\{x\} \in \mathfrak{B}$, $B_1, B_2 \in \mathfrak{B}$ be two base sets which are different from the base set $\{x\}$ such that $\{x\} = B_1 \cap B_2$. If B_1, B_2 are generated by the tolerance relation R, then there must be some $y, z \in U$ such that $B_1 = \{u \mid u \in U, yRu\}$, and $B_2 = \{u \mid u \in U, zRu\}$ (y, z are the generator objects of B_1 and B_2 respectively). As B_1 and B_2 are different from the base set $\{x\}$, $x \neq y$ and $x \neq z$. The base set $\{x\}$ is a singleton, therefore its generator object must be x, so $[x] = \{u \mid u \in U, xRu\} = \{x\}$. yRx and zRx hold because $x \in B_1$ and $x \in B_2$. Due to the similarity of the tolerance relation R, xRy and xRz also hold. This means that $y, z \in [x](= \{x\})$ which is a contradiction.

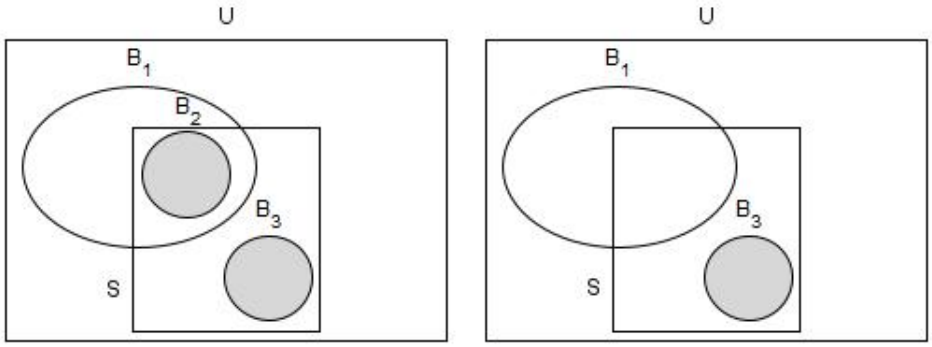


Figure 1.1. Base set-loss

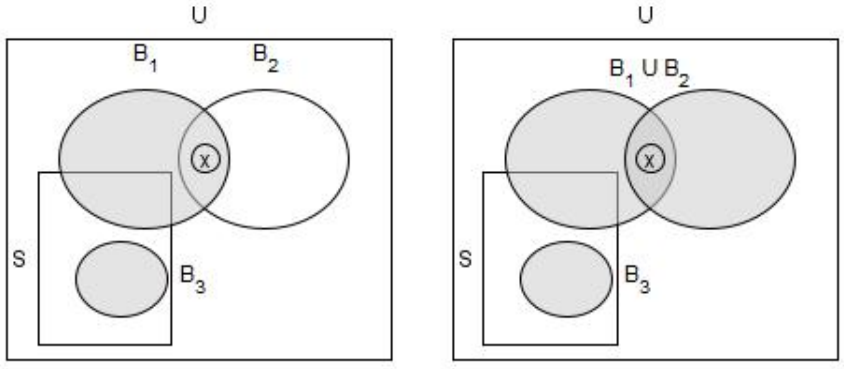


Figure 1.2. Base set-gain

Boundary regions are also essential in the representation of uncertainty. In [13] the authors showed that theoretically different boundary regions can be introduced into a general partial approximation space $\langle U, \mathfrak{B}, \mathfrak{D}_{\mathfrak{B}}, \mathsf{l}, \mathsf{u}\rangle$:

1. $\mathsf{b}_1(S) = \mathsf{u}(S) \setminus \mathsf{l}(S)$;
2. $\mathsf{b}_2(S) = \bigcup(\{B \mid B \in \mathfrak{B} \text{ and } B \cap S \neq \varnothing\} \setminus \{B \mid B \in \mathfrak{B} \text{ and } B \subseteq S\})$;
3. $\mathsf{b}_3(S) = \bigcup \mathcal{C}^{\mathsf{b}}(S)$, where $\mathcal{C}^{\mathsf{b}}(S) = \{B \mid B \in \mathfrak{B},\ B \cap S \neq \varnothing, \text{ and } B \not\subseteq S\}$.

In original Pawlakian spaces there is no difference between the aforementioned boundary regions, i.e. if $\langle U, \mathfrak{B}, \mathfrak{D}_{\mathfrak{B}}, \mathsf{l}, \mathsf{u}\rangle$ is an original Pawlakian space characterized by an ordered pair $\langle U, \mathcal{R}\rangle$, then $\mathsf{b}_1(S) = \mathsf{b}_2(S) = \mathsf{b}_3(S)$ for all $S \subseteq U$. In general case, the boundary regions defined according to the first point are not definable sets necessarily, therefore this definition cannot be used in general approximations spaces where one wants to rely on only

definable sets. If there are only finite number of base sets (i.e. $\mathfrak{B}$ is finite), then the sets $\mathsf{b}_2(S), \mathsf{b}_3(S)$ are definable for all $S \subseteq U$. Some important connections between different types of boundary regions were showed in [16, 13]:

- $\mathsf{b}_1(S) \subseteq \mathsf{b}_2(S) \subseteq \mathsf{u}(S)$;
- $\mathsf{b}_1(S) = \mathsf{b}_2(S)$ if and only if $\mathsf{b}_2(S) \cap \mathsf{l}(S) = \varnothing$;
- if $\mathfrak{B}$ is one-layered (i.e. the base sets are pairwise disjoint), then there is no difference between different types of boundary regions, i.e.
  - $\mathsf{b}_1(S) = \mathsf{b}_2(S) = \mathsf{b}_3(S)$;
  - $\mathsf{b}_1(S)$ is definable;
  - $\mathsf{b}_\mathsf{i}(S) \cap \mathsf{l}(S) = \varnothing$, where $i = 1, 2, 3$;
  - $\mathsf{u}(S) = \mathsf{l}(S) \cup \mathsf{b}_\mathsf{i}(S)$, where $i = 1, 2, 3$.

Notice that only lower and upper approximations (and so only background and embedded knowledge represented by base sets) are used, and in a finite one-layered case there is no real difference between different types of boundary regions.

The next step is to make clear the 'nature', the usage and the influences of background (and embedded) knowledge.

1. In the original Pawlakian case, the limit of our knowledge appears explicitly: base sets consist of indiscernible objects, there is no way to distinguish them from each other.
2. In covering structures generated by tolerance relations, a base set contains objects which are similar to a given object, and therefore they are treated in the same way. Being similar to a given object is a property, but it is a very special (not a general) one, it is generated by the tolerance relation.
3. In general covering spaces, base sets can be considered as the representations of real properties, and it is supposed that all objects have at least one (known, represented) property. Objects with the same property (members of a base set) are handled in the same way. (The system of base sets cannot be generated by tolerance relations in some cases.)
4. General partial spaces are similar to general covering ones, but it is not supposed that all objects have at least one property represented by a base set. In practical cases information systems are not total, there is no relevant information about an object: it may be in our database but some information is missing, and so it does not have any property represented by a base set.

Some problems appear in different cases. In practical applications indiscernibility (as an equivalence relation) may be too strong. In the case of a huge number of objects if there is a reflexive and symmetric relation, then it may be difficult to decide whether it is transitive. Covering spaces generated by tolerance relations give possibilities to use only reflexive and symmetric relations, but to many base sets appear, (each object generate a base set). These base sets are not about similarity (in general), but only about the similarity to given objects

(to their generators). In general covering and partial spaces, there is no room for similarity, these spaces rely on only common properties of objects. A pairwise disjoint system of base sets generated from a covering space (relying on a tolerance relation or a family of properties) or a general partial space is not a real solution: it is difficult to give any meaning represented by received base sets and too many small base sets appear, therefore the system may become very close to classical set theory. If every base set is a singleton (contains only one member), then the lower and upper approximation of any set will coincide. Therefore, every set will be crisp in this case.

As mentioned earlier, in practical applications indiscernibility can be too strong. Sometimes the similarity of objects is enough. From a mathematical point of view, it can be modeled by a tolerance relation. In computer science, correlation clustering a typical method that uses a tolerance relation. So a natural question arises: can the clusters (gained by the correlation clustering) represent the system of base sets? If so, then how do they modify the approximation space? In the next chapter, a new approximation space is presented that tries to answer the previous question.

Chapter 2

Similarity Based Rough Sets [1]

Some covering approximation spaces use tolerance relations, which represent similarity, (described in Section 1) instead of equivalence relations, but the usage of these relations is very special. It emphasizes the similarity to a given object and not the similarity of objects 'in general'. One can recognize this feature when they try to understand the precise meaning of the answer coming from an approximation relying on a covering approximation space (based on a tolerance relation). If one is interested in whether $x \in S$ (where S is the set to be approximated), then there are three possible answers [2] (see Fig. 2.1):

- if $x \in \mathsf{l}(S)$ (i.e. x is surely a member of S), then $y \in S$ for all $y, y \in [x']$ where $x' \in [x]$ and $[x'] \in \{B \mid B \in \mathfrak{B} \text{ and } B \subseteq S\}$;
- if $x \in \bigcup(\{B \mid B \in \mathfrak{B} \text{ and } B \cap S \neq \varnothing\} \setminus \{B \mid B \in \mathfrak{B} \text{ and } B \subseteq S\})$ (i.e. x is possibly a member of S), then there is an x' and a base set $[x']$ such that $x \in [x']$, $[x'] \cap S \neq \varnothing$, $[x'] \not\subseteq S$ and y may be a member of S for all $y \in [x']$;
- if $x \in \mathsf{l}(\overline{S})(= U \setminus \mathsf{u}(S))$ (i.e. x is surely not a member of S), then $y \notin S$ for all $y, x\mathcal{R}y$.

Some practical problems of covering approximation spaces based on a tolerance relation:

1. The former answers show, that generally the lower and upper approximations are not close in the following sense (see Fig. 2.2):
   (a) If $x \in \mathsf{l}(S)$, then it cannot be said that $[x] \subseteq S$.
   (b) If $x \in \mathsf{u}(S)$, then it cannot be said that $[y] \cap S \neq \varnothing$ for all $y \in [x]$.
2. The number of base sets is not more than the number of members of U, so there are too many base sets for practical applications.

If one wants to avoid these problems, then a Pawlakian approximation space can be generated by constructing a system of disjoint base sets [13] (see Fig. 2.3). If there are two

[1] The work described in this chapter was based on [36, 37, 35, 6, 22, 5, 7, 38]

[2] The formal definition of these answers can also be seen in the first chapter but for better readability, they are also listed here again focusing on covering approximation spaces based on a tolerance relation.

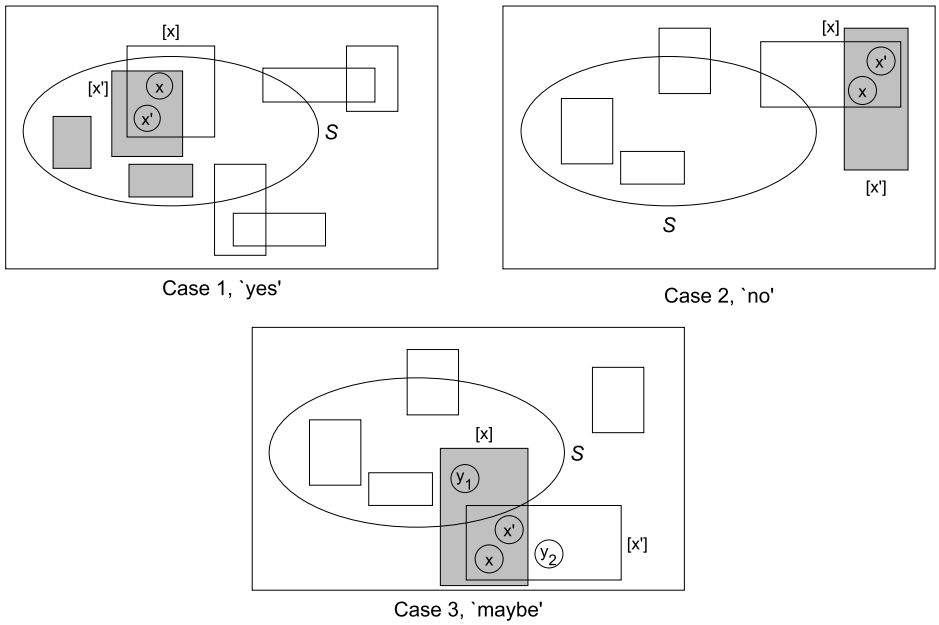


Figure 2.1. Some base sets in covering (based on a tolerance relation) cases

base sets B_1, B_2, such that $B_1 \cap B_2 \neq \varnothing$, then they can be substituted with the following three sets: $B_1 \setminus B_2$, $B_2 \setminus B_1$, $B_1 \cap B_2$. Applying this iteratively the reduction can be got. Although, it is not a real solution from the practical point of view. The base sets can become too small for practical applications. The smaller the base sets are, the closer the system gets to the classical set theory. (If all base sets are singleton, then there is no difference between the approximation space and classical set theory)

In rough set theory, the members of a given base set share some common properties.

- In Pawlak's original space all members of a given base set have the same attributes (i.e. they have the same properties with respect to the represented knowledge).
- In covering approximation spaces based on a tolerance relation, all members of a given base set are similar to a distinguished object (which is used to generate the given base set).

A further generalization is possible: general (partial) Pawlakian approximation spaces can be obtained by the generalization of the set of base sets:

- let $\mathfrak{B}$ be an arbitrary nonempty set of nonempty subsets of U.

These spaces are Pawlakian in the sense that they use Pawlakian definition of definable sets and approximation pairs. This generalization is very useful because a base set can be taken as a collection of objects with a given property, and one can use very different properties to define different base sets. The members of the base set can be handled in the same way relying

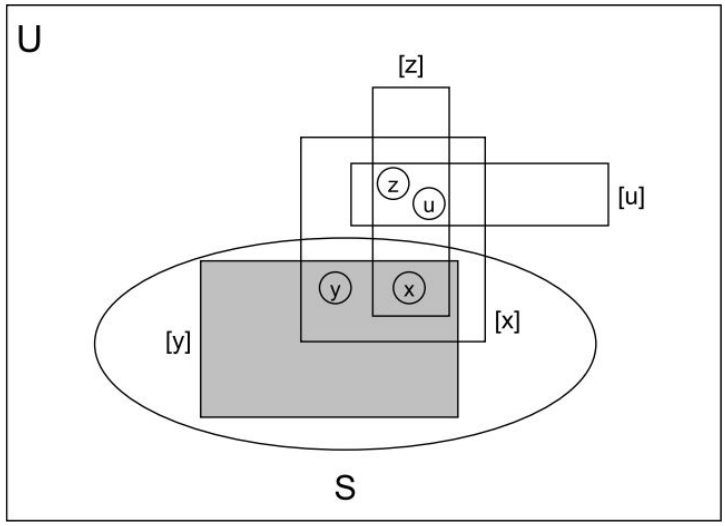
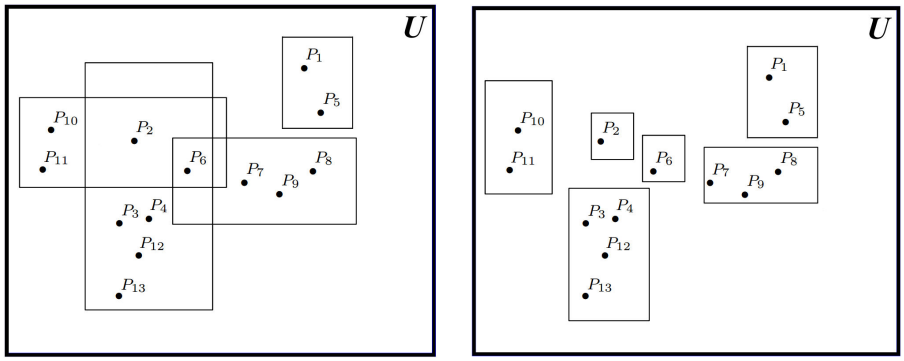


Figure 2.2. In covering (based on a tolerance relation) the lower and upper approximations are not closed



(a) Covering (b) Partition

Figure 2.3. Covering (based on a tolerance relation) and its reduction to a partition

on their common property. In this case there is no way to give a corresponding relation which can generate base sets (similarly to covering approximation spaces), so a general (partial) Pawlakian approximation space can be characterized only by the pair $\langle U, \mathfrak{B} \rangle$, since the lower and upper approximations of a subset of U are determined by the members of $\mathfrak{B}$. However, any system of base sets induces a tolerance relation $\mathcal{R}$ on U: $x\mathcal{R}y$ if there is a base set $B \in \mathfrak{B}$ such that $x, y \in B$. If this relation is used to get the system of base sets, the result can be totally different from our original base system (see Fig. 2.4).

In Fig. 2.4 x is in the intersection of B_1 and B_2 (B_1 and B_2 are defined by some properties). It means that it has common properties with all y_i and z_i, where $i = 1, 2, 3$. So if some

$x \in B_1 \cap B_2$ it means that:

- $x\mathcal{R}y$ for all $y \in B_1$
- $x\mathcal{R}z$ for all $z \in B_2$

Therefore the base set generated by x is the following: $[x] = B_1 \cup B_2$ (In this example only two base sets were used, but it is the same when there are more.)

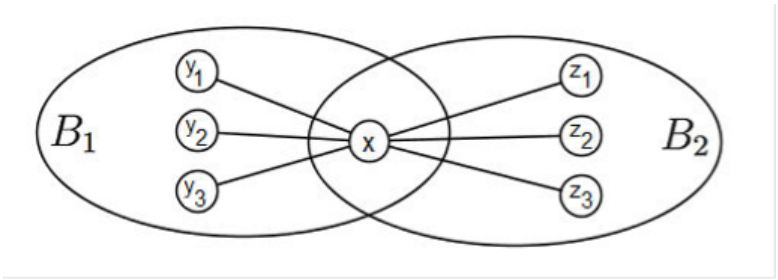


Figure 2.4. Base sets by properties of objects

When one would like to define the base sets, then the background knowledge embedded in a given information system is used. In the case of a Pawlakian space, two objects are called indiscernible if all of their known attribute values are the same. In many cases covering spaces rely on a tolerance relation based similarity. As it was mentioned earlier, some problems can come up using these covering spaces. A base set contains members that are similar to a distinguished member. This means that these spaces do not consider the similarity itself but the similarity with respect to a distinguished object. If correlation clustering (described in section 2.1) is used, based on the tolerance relation, a partition of the universe is obtained. The clusters contain objects which are mostly similar to each other (not just to a distinguished member). So the partition can be understood as a system of base sets. As a result, a new approximation space appears which uses the same tolerance relation as the aforementioned covering spaces and has the following features:

- the similarity of objects relying on their properties (and not the similarity to a distinguished object) plays a crucial role in the definition of base sets;
- the system of base sets consists of disjoint sets, so the lower and upper approximation are closed;
- only the necessary number of base sets appears (in applications, an acceptable number of base sets must be used);
- the size of base sets is not too small, or too big.

The formal definition can be seen here:

Definition 2.1.

$$\mathfrak{B} = \{B \mid B \subseteq U, \textit{ and } x, y \in B \textit{ if } p(x) = p(y)\},$$

where p is the partition gained from the correlation clustering.

Singleton clusters sets represent very little information (its member is only similar to itself). Without increasing the number of conflicts, its member cannot be considered similar to any object. So, they always require an individual decision. By deleting the singletons, a partial system of base sets can be defined. In section 2.4 a method is proposed

2.1 Correlation clustering

Cluster analysis is a widely used technique in data mining. Our goal is to create groups in which objects are more similar to each other than to those in other groups. Usually, the similarity and dissimilarity are based on the attribute values describing the objects. Although there are some cases when the objects cannot be described by numbers, but something about their similarity or dissimilarity can still be stated. Think of humans for example. It is hard to detail someone's looks by a number, but one still makes statements whether two persons are similar to each other or not. Of course, these opinions are dependent on the persons. Some can treat two random persons as similar, while others treat them dissimilar. If one wants to formulate the similarity and dissimilarity by using mathematics, then a tolerance relation is needed. If this relation holds for two objects, then they are similar. If this relation does not hold, then they are dissimilar. Of course, each object is similar to itself, so the relation needs to be reflexive, and it is easy to show, that it also needs to be symmetric. But one cannot go any further, e.g. the transitivity does not hold necessarily.

If a human and a mouse are taken, then due to their inner structure they are similar. This is the reason why mice are used in drug experiments. Moreover, a human and a Paris doll are similar due to their shape. This is the reason why these dolls are used in show-windows. But there is no similarity between a mouse and a doll except that both are similar to the same object. Correlation clustering is a clustering technique based on a tolerance relation [8, 9, 57].

Our task is to find an $\mathsf{R} \subseteq U \times U$ equivalence relation *closest* to the tolerance relation. A (partial) tolerance relation $\mathcal{R}$ [47, 32] can be represented by a matrix M. Let matrix $M = (m_{ij})$ be the matrix of the partial relation $\mathcal{R}$ of similarity: $m_{ij} = 1$ whenever objects i and j are similar, $m_{ij} = -1$ whenever objects i and j are dissimilar, and $m_{ij} = 0$ otherwise.

A relation is partial if there exist two elements (i, j) such that $m_{ij} = 0$. It means that if there is an arbitrary relation $\mathsf{R} \subseteq U \times U$, then there are two sets of pairs. Let $\mathsf{R}_{\mathsf{true}}$ be the set of those pairs of elements for which the R holds, and $\mathsf{R}_{\mathsf{false}}$ be the one for which R does not hold. If R is partial then $\mathsf{R}_{\mathsf{true}} \cup \mathsf{R}_{\mathsf{false}} \subseteq U \times U$. If R is total then $\mathsf{R}_{\mathsf{true}} \cup \mathsf{R}_{\mathsf{false}} = U \times U$.

A partition of a set S is a function $p : S \to \mathbb{N}$. Objects $x, y \in S$ are in the same cluster at partitioning p, if $p(x) = p(y)$.

The cost function counts the negative cases i.e. it gives the number of cases whenever two dissimilar objects are in the same cluster, or two similar objects are in different clusters. The cost function of a partition p and a relation R_M with matrix M is

$$f(p, M) = \frac{1}{2} \sum_{i<j} (m_{ij} + abs(m_{ij})) - \sum_{i<j} \delta_{p(i)p(j)} m_{ij},$$

where δ is the Knockecker delta symbol [39]. For a fixed relation, the partition with the minimal cost function value is called optimal. Solving a correlation clustering problem is equivalent to minimizing its cost function, for the fixed relation. If the value of this optimal cost function is 0, the partition is called perfect. Given the $\mathcal{R}$ and R the value f is called the

distance of the two relations. The partition given this way generates an equivalence relation. This relation can be considered as the closest to the tolerance relation.

It is easy to check that the solution cannot be generally perfect for a tolerance relation (based on similarity). Consider the simplest such case, given three objects x, y and z, and x is similar to both y and z, but y and z are dissimilar. In this situation, the following 5 partitions can be given:

$$\{\{x,y,z\},\{\{x,y\},\{z\}\},\{\{x,z\},\{y\}\},\{\{y,z\},\{x\}\},\{\{x\},\{y\},\{z\}\}\}.$$

In every of one them there is at least 1 conflict.

The number of partitions can be given by the Bell number [1], which grows exponentially. Hence, in general — even in the case of some dozens of objects — the optimal partition cannot be determined in a reasonable time, thus a search algorithm that produces a quasi-optimal partition would be more useful in practical cases. However, in practical examples, it gives us the right to handle objects, which are in the same class, the same way.

2.2 Comparison Between Covering Based on a Tolerance Relation and Similarity Based Rough Sets

As mentioned in the previous subsection, in these covering based spaces there can be too many base sets. Naturally, this can be very problematic if the dataset is huge because the approximation process can take a lot of time. Therefore, in practice, these spaces can be a little bit slow. So a natural question arises. How fast can the approximation be with similarity based rough sets? In [36] a comparison between covering (based on a tolerance relation) spaces (and its disjoint variant) and similarity based rough sets can be seen.

The execution time of the approximation process can be seen in Fig. 2.5. The axis x represents the number of points, and the axis y represents the execution time in milliseconds. The points here were chosen randomly.

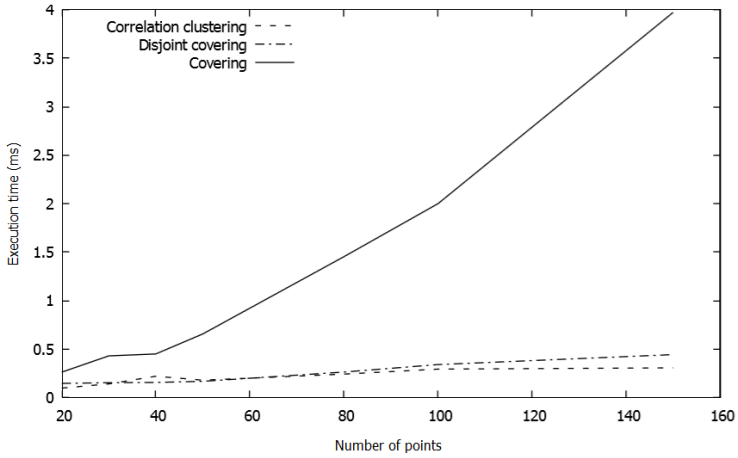


Figure 2.5. Execution time

If one takes a look at the figure, then it can be seen that the approximation by covering (based on a tolerance relation) is the slowest. This was expected because there are a lot of base sets to work with. Between the disjoint covering and the correlation clustering, there is no significant difference. Nevertheless, as the number of points increases, the correlation clustering gives the fastest way to approximate. It is an interesting fact that there is such a great difference between the covering (based on a tolerance relation) and its disjoint variant. Despite the fact that a disjoint covering has the largest number of base sets, their cardinality is much less (most of them are singleton) than that of covering based on a tolerance relation.

It is also interesting how the lower and upper approximation would look like in the case of similarity based rough sets. A test was run with 100 random points on a two-dimensional space.

The base of the tolerance relation (based on similarity) was the Euclidean distance of these objects (d). A similarity ($SIMM$) and a dissimilarity threshold ($DIFF$) were defined. $SIMM$ was set to 50 and $DIFF$ was set to 90. The tolerance relation $\mathcal{R}$ can be given this way for any objects x, y:

$$x\mathcal{R}y = \begin{cases} +1 & d(x,y) \leq SIMM \\ -1 & d(x,y) > DIFF \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

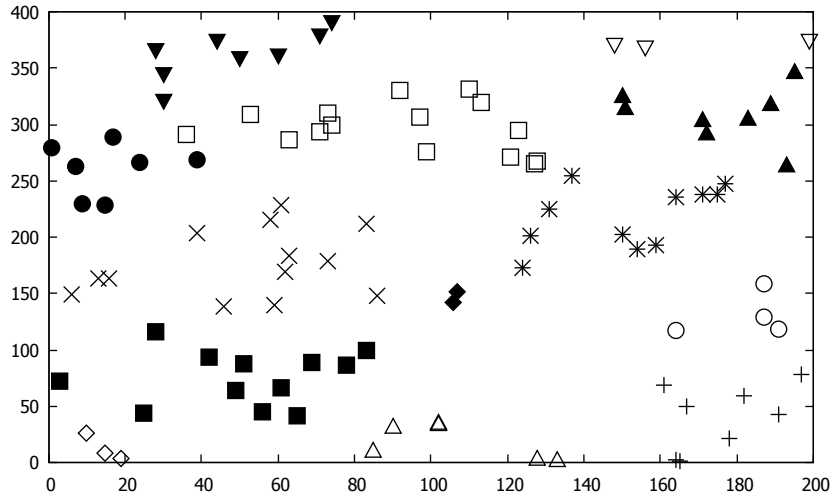


Figure 2.6. The clusters

Fig. 2.6 represents the clusters (base sets) created by the correlation clustering. The set to be approximated is shown in Fig. 2.7. The members of this set are denoted by the $\times$ symbols, and the other members are denoted by the cross symbol. The members were chosen randomly.

The approximation generated by the correlation clustering is displayed in Fig.2.8/A. The cardinality of the base sets is relatively great so the lower approximation consists of only a few members. (Only the members denoted by the empty circle and filled diamond are in the set.)

The approximation generated by the covering (based on a tolerance relation) is shown in Fig.2.8/B. Like in correlation clustering the lower approximation consists of only a few members. The two lower approximations have some difference, but they only differ in a set which has two members.

Between the upper approximations, a significant difference can be seen. The upper approximation defined by covering (based on a tolerance relation) contains many more objects, almost twice as much as the one defined by correlation clustering.

The approximation generated by the disjoint covering is shown in Fig.2.8/C. It can be seen that among the methods this generated the finest approximation (lower and upper approximation coincide). The reason is that almost all base sets are singletons. As mentioned before, if there were only singleton clusters, one would get the common set theory back.

2.3 Search Algorithms [3]

In a reasonable time, correlation clustering can only be solved by using search/optimization algorithms. However, each algorithm can provide different clusters. So the system of base

[3]The work described in this section was based on [38]

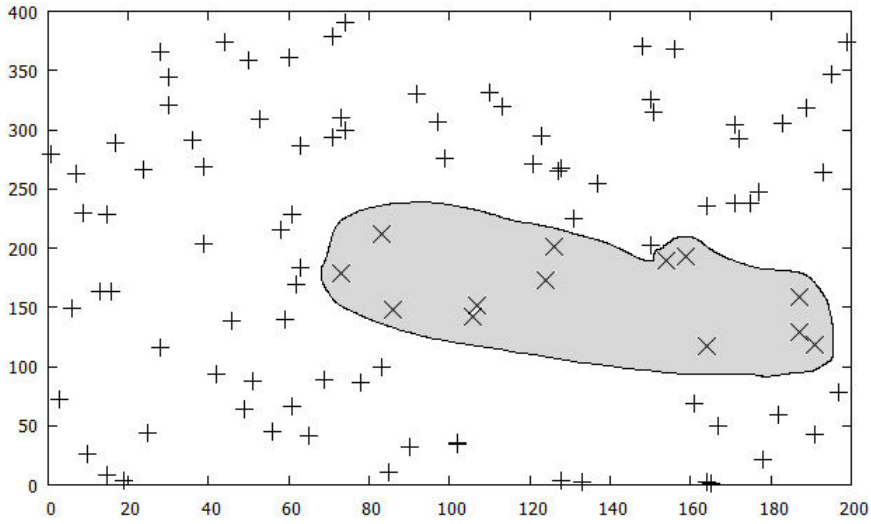


Figure 2.7. The set to be approximated

sets can also be different. It is a natural question to ask how the search algorithms can affect the structure of the base sets. As the approximation based on correlation clustering is a completely new way of approximating sets, so it is crucial to use the best possible search algorithm. In this section, some widely-known search algorithms are shown and also a comparison of how they work in the case of similarity based rough sets [38].

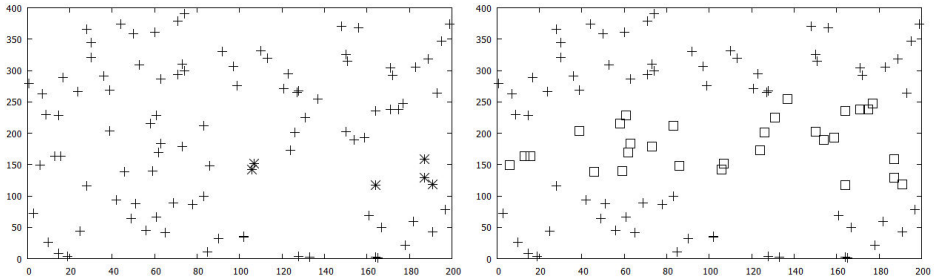
The following list shows the used algorithms in the experiments. Each of them can be downloaded from [3] and their description can be seen in Appendix A:

- Hill Climbing Algorithm
- Stochastic Hill Climbing Algorithm
- Tabu Search
- Simulated Annealing
- Parallel Tempering
- Genetic Algorithm
- Bees Algorithm
- Particle Swarm Optimization
- Firefly Algorithm

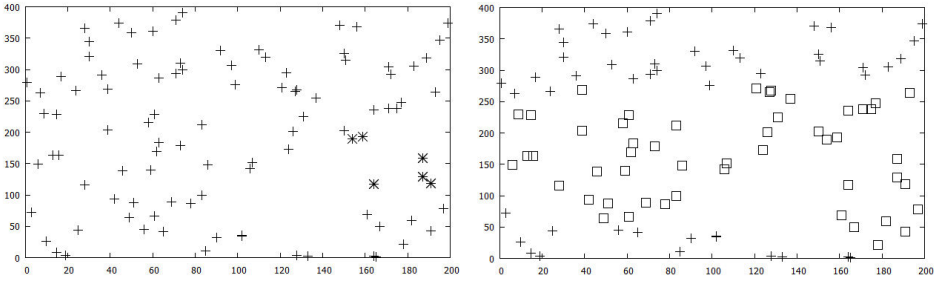
To compare the algorithms, the following values were computed:

- Number of singleton clusters

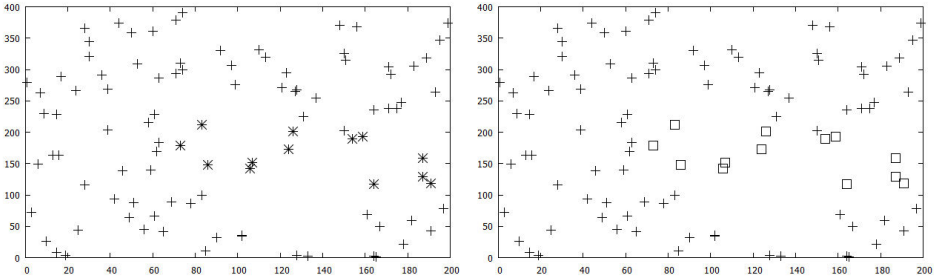
Figure 2.8. The outputs of the approximations by the software



A. The lower (left) and upper (right) approximation by correlation clustering



B. The lower (left) and upper (right) approximation by covering (based on a tolerance relation)



C. The lower (left) and upper (right) approximation by disjoint covering

- Standard deviation of the base set sizes
- The range of the base set sizes
- Execution time of the algorithm

Here, the cardinality of a base set is referred to as its size. As previously mentioned, singleton clusters mean little information. The greater their number is, the more unclear the information based on our knowledge becomes. For a search algorithm, the most optimal is, if it provides the least number of these clusters to have a precise result of the system.

The sizes of the base sets are also worth to be checked. For set approximations, it is more suitable if the sizes do not vary much. So the standard deviation of the base set sizes should be minimized as well as the range of the base set sizes.

An important parameter is the execution time of the search algorithms. It is especially crucial when there are a huge number of objects.

2.3.1 Results

Similarity based rough sets always requires a tolerance relation which represents similarity. In the experiments, random graphs were only used to define these tolerance relations.

Most of the algorithms have many parameters, and changing them can result in different outputs. Many possible combinations were tried during the research. Dozens of tables were generated and these tables are not present in this dissertation, but they can be downloaded from the following link: `https://bit.ly/2sO4UoD`

For comparing the parameters, the same graph (with 100 points, p = 0.6) were used for each algorithm. Each algorithm was run three times to exclude the randomness. In each case, the optimal parameter combination was the one that minimized the above mentioned 4 values. If the differences between the standard deviations, the ranges and the numbers of singletons were not significant, then the judgment was made by the execution time.

Algorithm 1 Erdős-Rényi random graph generating method

```
1:  procedure GENERATE(p)
2:      for i = 1, ..., N do
3:          for j = 1, ..., N do
4:              Generate a random x value between 0 and 1
5:              if x < p then
6:                  There is an edge between objects i and j
7:              end if
8:          end for
9:      end for
10: end procedure
```

In this part of the experiments, Erdős-Rényi graphs were used [24, 23]. This random graph generating method is very simple. Its pseudo-code can be seen in Algorithm 1. In the experiments, the following values were used $p = 0.5$, $p = 0.6$ and $p = 0.7$. Half of the generated edges denoted the similarity between the two objects and half of them the difference. Naturally, any other kind of graph can be used. 100, 200, 300 and 400 points

Table 2.1. Average standard deviations of the base set sizes for Erdős-Rényi graphs

Points	HC	SHC	TABU	SA	PT	GE	BEE	PSO	FF
200 q=0.5	37	31	32	13	12	40	15	34	27
400 q=0.5	63	61	51	15	24	82	21	66	49
200 q=0.6	37	35	37	18	19	39	15	35	31
400 q=0.6	69	63	61	21	23	83	26	65	49
200 q=0.7	37	31	26	18	20	38	15	39	30
400 q=0.7	68	61	66	14	19	78	18	63	55

were generated. For each test case, each algorithm was run 3 times on the same graphs, then the averages of the values, described in section 2.3, were determined. The results can be seen in the following tables and can be downloaded from the following link: `https://bit.ly/2sOqMA6`.

From Table 2.1 the average standard deviations of the base set sizes can be read. Even for a small number of points, the differences are apparent. The simulated annealing provided the best result in most cases. Its parallel version has almost the same output. The bees algorithm also returned a rather acceptable result.

In Table 2.2 the distance of the sizes of the biggest and the smallest base sets can be seen. The values show almost the same tendency as in the last table. The simulated annealing, the parallel tempering, and the bees algorithm proved to be the most acceptable. For 400 points, the other algorithms provided twice or three times as large values as the other 3 which is not suitable.

In Table 2.3 the average numbers of singletons are listed. Here, the differences are not so significant as before. The number of points does not affect it very much.

In Table 2.4 the average run-time of the algorithms can be seen in seconds. The values here vary the most. The simulated annealing was the least affected by the increasing number of points. For 200 points, the hill climbing algorithm and its stochastic version provided the fastest output. Although, as soon as the number of points was increased, they could not compete with the simulated annealing. For 400 points, the simulated annealing was more than 20-35 times faster than the other two. The particle swarm optimization was the slowest of all the algorithms. For a huge number of points, it is pointless to be used.

In this part of our experiments, random two-dimensional points were generated. The tolerance relation (based on similarity) is defined in equation 2.1. 100, 150, 200, 300, 500 points were generated and each algorithm was run 3 times for each point set and calculated the averages of the values described in section 2.3. In the following tables, the results can be seen.

In Table 2.5 the average standard deviations of the base set sizes can be seen. In the case

Table 2.2. Average ranges of the base set sizes for Erdős-Rényi graphs

Points	HC	SHC	TABU	SA	PT	GE	BEE	PSO	FF
200 q=0.5	105	92	80	48	45	98	49	92	82
400 q=0.5	176	189	144	57	78	213	65	186	193
200 q=0.6	105	94	97	46	46	98	47	89	92
400 q=0.6	192	173	178	87	62	209	81	193	197
200 q=0.7	104	94	74	52	64	104	49	104	88
400 q=0.7	198	179	191	50	53	228	81	186	210

Table 2.3. Average numbers of singletons for Erdős-Rényi graphs

Points	HC	SHC	TABU	SA	PT	GE	BEE	PSO	FF
200 q=0.5	0	0	0	1	0	1	0	1	1
400 q=0.5	0	0	0	2	2	2	0	2	7
200 q=0.6	1	1	1	1	1	1	0	1	2
400 q=0.6	0	0	0	3	3	1	1	1	3
200 q=0.7	1	0	0	1	1	1	0	1	2
400 q=0.7	0	1	0	2	3	1	1	1	4

Table 2.4. Average execution time for Erdős-Rényi graphs

Points	HC	SHC	TABU	SA	PT	GE	BEE	PSO	FF
200 q=0.5	5	7	163	3	16	82	53	569	274
400 q=0.5	142	181	1549	6	39	360	247	7971	1167
200 q=0.6	4	6	170	3	17	91	66	734	317
400 q=0.6	156	248	1665	6	40	457	274	8611	1287
200 q=0.7	3	8	156	3	17	87	59	818	283
400 q=0.7	138	256	1652	7	43	404	284	9878	1336

of a small number of points, the difference was not so considerable, but it became larger as the number of points was increased. In every case, the simulated annealing provided the most acceptable result. Interestingly, the parallel tempering fell short against the simulated annealing for a small number of points. However, in the 500 points test case the difference was negligible. Local search algorithms (hill climbing, its variant, tabu search) were rather good for a small number of points. In almost every situation, the firefly algorithm, genetic algorithm and particle swarm optimization provided the worst result.

Table 2.5. Average standard deviations of the base set sizes

Points	HC	SHC	TABU	SA	PT	GE	BEE	PSO	FF
100	7.6	8.1	7.5	7.1	7.6	8.9	7.6	7.7	7.6
150	11.8	10.2	10.4	7	11	14.5	8.7	12.2	11.5
200	13.1	16.2	12.8	10.3	13.2	17.8	12.7	13.8	13.3
300	25.7	28.9	27.2	17.1	18.4	31.4	23	29.1	28.9
500	46.6	34.4	39.4	26.5	27	51.4	34.2	47.8	48.4

Table 2.6 shows the ranges of the base set sizes. The outcome was quite similar as in the previous table. For a small number of points, the difference was not so high. For 500 points, it can be more noticeable. Like before, the firefly algorithm, genetic algorithm and particle swarm optimization ended up in the last places, and simulated annealing proved to be the most optimal.

Table 2.7 shows how many singleton clusters appeared. The results were quite the same in all cases. It is interesting that most of the algorithms were not affected by the increasing number of points. In the 500 points test case, some differences can be observed. In this case, the simulated annealing and its parallel version provided a rather inadequate result compared

Table 2.6. Average ranges of the base set sizes

Points	HC	SHC	TABU	SA	PT	GE	BEE	PSO	FF
100	23	24	22	21	26	23	23	26	23
150	27	25	35	20	31	37	21	37	33
200	38	38	38	30	41	47	37	40	35
300	67	68	70	48	61	74	64	71	68
500	111	94	99	73	81	119	103	117	116

Table 2.7. Average numbers of singletons

Points	HC	SHC	TABU	SA	PT	GE	BEE	PSO	FF
100	1	1	0	0	0	0	0	0	1
150	1	1	2	0	1	2	0	2	2
200	1	1	0	0	0	1	0	1	2
300	0	1	1	2	1	0	0	1	3
500	2	1	0	5	4	1	3	1	5

to the others. However, this difference was not so high.

In Table 2.8 the average execution time is listed in seconds. As expected, these values were the most dependent on the number of points. For less than 200 points, the hill climbing algorithm and its stochastic variant provided the fastest run-time. However, after 200 points they could not compete with the simulated annealing which could find the quasi-optimal partition in less than 5 seconds for each test case. The parallel tempering also proved to be quite fast, but not as fast as its single-threaded variant. The other algorithms executed in an unreasonable time which is unacceptable for a great number of points. Especially the particle swarm optimization proved to be very slow, it finished running after 3.5 hours for 500 points.

The experiments showed that the simulated annealing proved to be the best choice. In almost every test case, it provided the most suitable result. However, its most important property is that it was the least affected by the increasing number of points, so it can also finish in a reasonable time even for large amounts of points.

Table 2.8. Average execution time

Points	HC	SHC	TABU	SA	PT	GE	BEE	PSO	FF
100	0.2	0.3	15.6	1.1	5.4	15.7	15.5	32.5	54.1
150	0.5	0.4	43.5	0.7	7.3	14.5	11.9	130.8	58.3
200	6.4	9.5	172.4	3	16.8	92.7	66.3	564	281.1
300	40.1	43.5	647.4	4.7	25.2	207	161.1	1937.8	579.6
500	69.4	108.5	3550	3	50.8	207.4	135.9	13251	612.3

2.4 Similarity Based Rough Sets with Annotation [4]

Singleton clusters represent very little information because the system could not consider their members similar to any other objects without increasing the value of the cost function (see in section 2.1). As they mean little information, they can be left out. If the singleton clusters are not considered, then a partial system of base sets can be generated from the partition. Sometimes it can happen that an object does not belong to any cluster because the system could not consider it similar to any other objects based on the background information. This does not mean that this object is only similar to itself, but without proper information, the system could not insert it into any cluster in order not to increase the number of conflicts. In medical applications, it can occur that a patient has a similar disease as some other patients but has different data in the information system. In this case, the search algorithm would consider this patient different from the others and so the patient would not belong to any non-singleton cluster. Although, a doctor or an expert could recognize that the patient could belong to a non-singleton cluster. The original partial space was defined by the correlation clustering. However, the user has some background knowledge. They can use this knowledge to help the system by inserting the members of some singletons into base sets (non-singleton clusters). With the help of the annotation process, the user can put their own knowledge into the system. It also decreases the partiality by decreasing the number of singletons. After the annotation, a new approximation space appears.
Let S be the set to be approximated, $\{x\}$ a singleton gained from the correlation clustering and B a base set. The following cases can happen with the base set B after the annotation if $B \subseteq \mathsf{l}(S)$:

- If $x \in S$, then $B' = \{x\} \cup B$ and $B' \subseteq \mathsf{l}(S)$ This way the approximation of the set S becomes more precise.
- If $x \notin S$, then $B' = \{x\} \cup B$ and $B' \subseteq \mathsf{u}(S)$ but $B' \not\subseteq \mathsf{l}(S)$ This increases the uncertainty relative to the set S.

The following cases can happen with the base set B after the annotation if $B \subseteq \mathsf{u}(S)$:

- If $x \in S$, then $B' = \{x\} \cup B$ and $B' \subseteq \mathsf{u}(S)$
- If $x \notin S$, then $B' = \{x\} \cup B$ and $B' \subseteq \mathsf{u}(S)$

The following cases can happen with the base set B after the annotation if $B \subseteq \mathsf{u}(S) \setminus \mathsf{l}(S)$:

- If $x \in S$, then $B' = \{x\} \cup B$ and $B' \subseteq \mathsf{u}(S) \setminus \mathsf{l}(S)$
- If $x \notin S$, then $B' = \{x\} \cup B$ and $B' \subseteq \mathsf{u}(S) \setminus \mathsf{l}(S)$

In both cases, the upper approximation and the boundary region become larger. It can be said that the annotation depends on the set to be approximated. It could be useful if:

- $x \in S$, then the user could only choose from those B base sets which are in $\mathsf{l}(S)$.

[4] The work described in this section was based on [37]

- $x \notin S$, then the user could only choose from those B base sets which are in $\mathsf{l}(\overline{\mathsf{u}(S)})$, where $\overline{\mathsf{u}(S)}$ denotes the complement of the upper approximation.

This relative annotation looks very promising.

The order of the annotation is also worth to be checked. If the members x_1, x_2 of 2 different singletons were to be inserted into the same base set B, then the following question needs to be answered. Is it still relevant to insert x_2 into B after putting x_1 into B?

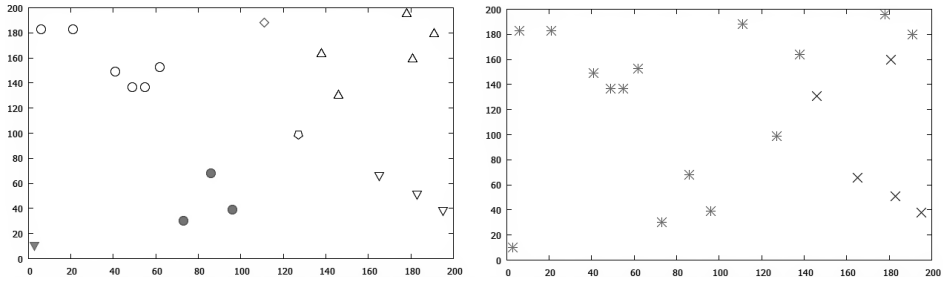
- If the answer is yes, then the two members are interchangeable. This means that x_1, x_2 has some sort of similarity that was hidden in the tolerance relation (based on similarity).
- If the answer is no, then the two members are not interchangeable. This means that annotating x_1 makes it irrelevant to insert x_2 into B.

In a real-world application, it can happen that an attribute value of an object is missing. This means that it can be unknown, unassigned or inapplicable (i.e. maiden name of a male). Coping with these data is usually a hard task. In many cases, these values are often substituted. It is common to replace a missing value with the mean or the most frequent value. Typically this gives a rather good result in many situations. In early-stage diabetes, it is not unusual that only the blood sugar level is higher than the normal level. If this value is missing for a patient, then it should not be replaced by the mean because the mean can be the normal blood sugar level. After the substitution, this patient can be treated as a healthy one. This type of substitution does not consider the information of an object itself but the information of a collection of objects, therefore it can lead to a false conclusion. The following method is proposed to handle missing data. If an object has a missing attribute value, then it cannot be treated as similar to any other object, so this entity forms a cluster alone. As mentioned earlier, these clusters cannot be treated as base sets. However, with the annotation, the user has the possibility to decide whether an object with missing data is similar to other objects or not. The user has some background knowledge that can be used this way to cope with the missing values. In this case, the information of an object itself is considered.

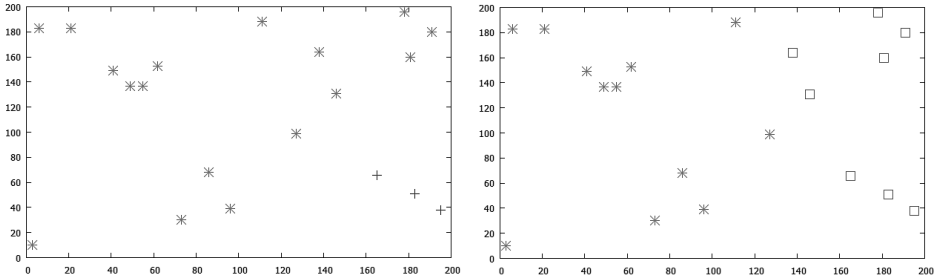
2.4.1 Annotation with Random Points

In this subsection, a possible example of the annotation process is shown. In the following figures, 20 points can be seen. The tolerance relation (based on similarity) is defined in equation 2.1. The similarity threshold $SIMM$ was set to 50, and $DIFF$ was set to 90. On the left side of Fig. 2.9/A the clusters generated by the correlation clustering can be seen. The singleton clusters contain the objects denoted by the $\diamond$ symbol, the $\pentagon$ symbol and the $\blacktriangledown$ symbol. Some points were selected for approximation. The members of this set are denoted by the $\times$ symbols, and the other members are denoted by the star symbol. The members were chosen randomly. This set can be seen on the right side of Fig. 2.9/A. In Fig. 2.9/B the reader can see the lower and upper approximation defined by the base sets gained from clustering after leaving the singletons out. The members of two singletons were inserted into two different base sets. The singleton denoted by the $\diamond$ symbol was merged with the base set denoted by the $\triangle$ symbol. The base set denoted by the $\bigtriangledown$ symbol was extended with the singleton denoted by the $\pentagon$ symbol. The result of the annotation can be seen in

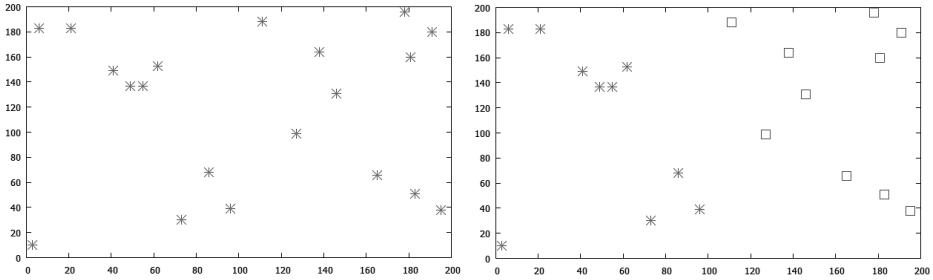
Figure 2.9. Annotation with random points



A. Clusters (left) and the set to be approximated (right)



B. The lower (left) and upper (right) approximation by clustering



C. The lower (left) and upper (right) approximation by clustering with annotation

Fig. 2.9/C. None of the members of the chosen singletons were members of the set to be approximated. This is the reason why the lower approximation became the empty set, and the upper approximation had more members.

2.5 Tools of the Annotation [5]

Annotation is a very useful improvement of the similarity based rough sets space because it creates a possible interaction between the system and the user. Naturally, this process is based on the user's expertise as they have to override the decision made by the system. Of course, this does not mean that there should not be any help or suggestions provided by the system itself. Two main techniques are introduced in the next part of the dissertation. The first one is a graphical method which tries to give a visual representation of the tolerance relation based on similarity. If two objects are close in this representation, then this indicates that those two objects should be treated as similar. If a member of a singleton is close to a base set, then maybe they should be merged. The second method is a mathematical way that aims to find those members in each cluster that can represent the entire cluster. During the computations, only the representative members should be considered. This way a lot of time and resources can be saved.

It is sometimes possible that there are more than one suitable base sets into which the user should insert the member of a singleton. In this case, the recommended base set should be the one whose representative member is the most similar to the member of the given singleton. In this way, there is no need to compare it to each member of each base set.

The annotation process can also be qualified as relevant or irrelevant regarding how it changes the representatives.

1. Relevant: After inserting a member of a singleton into a base set B, the representative member of the new base set B' is changed. In this case, some real information is implemented into the system. Let us assume that the objects are members of political parties and the representative members are the leaders of these parties. The annotation process is when a new member is elected to a party. If the annotation is relevant, then it means that the balance of the party has changed, and a new leader rises.

2. Irrelevant: After inserting a member of a singleton into a base set B, the representative member of the new base set B' is unchanged. In this case, the implemented information is not relevant because it does not alter the base sets gained from the correlation clustering.

In either case, the annotation can modify the set of possible representatives. As conclusion, one can say that if after the annotation something was changed, then the user had some useful information that was not embedded in the tolerance relation (based on similarity).

2.5.1 Visualization of Tolerance Relations [6]

Correlation clustering is based on similarity which can be represented by a tolerance relation. Visualizing these tolerance relations is sometimes problematic. In this chapter, an algorithm

[5] The work described in this section was based on [35, 6, 5, 7]

[6] The work described in this section was based on [5]

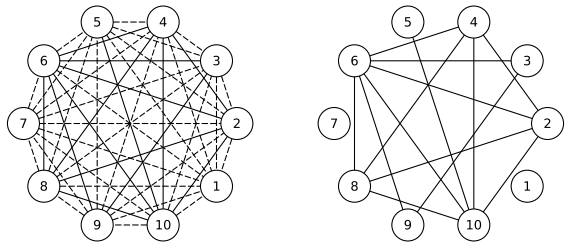


Figure 2.10. Tolerance relation based on GCD.

is presented [5] which can display similarity (based on a tolerance relation) in such a way that the user can easily interpret it. In the next paragraph, it can be seen why it is difficult to visualize these kinds of relations and why it is important to have a proper algorithm that can solve this issue. Let us suppose that two natural numbers are similar if their greatest common divisor is greater than 1. 1 is assumed to be similar to itself. And they are treated dissimilar if their greatest common divisor is 1. The analysis of this problem and its surprising solution can be found in [4].

If one takes this relation on numbers $1, 2, \ldots, 10$, then the picture on the left in Fig. 2.10 can be constructed. Here the circles representing the numbers are positioned on a circle, to make it easy to connect the numbers. In this picture, the similarity is denoted by solid lines and the dissimilarity by dashed ones. Even though each number is similar to itself (except for 1), it is not displayed in the figure. This picture is transparent, but imagine a similar picture denoting the tolerance relation of numbers $1, 2, \ldots, 100$! Why not leave out the dashed lines? The picture on right in Fig. 2.10 shows only the similarities. It is slightly more understandable, but it is hard to see the nexus. If the numbers are repositioned, everything becomes clearer. The numbers on the left of the circle have no similarities in the picture on the left of Fig. 2.11, and one can easily discover the groups 2-4-6-8-10, 3-6-9, 5-10. If the circle is disposed, the structure can become even clearer. The software yEd produced the picture in Fig. 2.11 on the right.

Numbers are easy to compare. However, it is not certain that 2 random objects are comparable. Or they are comparable, but nobody compared them yet. Hence, the relation can be partial. This means that there are three cases: two objects can be similar, dissimilar, or neutral. This can be visualized with three colors or three types of lines. If the lines for neutrality are not drawn, then the solid and dashed lines are enough.

In some graph visualization methods (force-directed graph) edges are modeled by springs, and the nodes are electrically charged particles. In these methods, the similarity (the edge between two nodes) is handled with springs, and the dissimilarity (the absence of the edge between two nodes) is handled with electricity. The graphs visualized with these methods are sparse graphs, i.e. the number of edges is a linear function of the number of nodes.

In this case, three different kinds of springs are used according to the three different values of the partial tolerance relation. With these three values, the graph of this relation is a dense one (all pairs of nodes are somehow connected), i.e. the number of edges is a quadratic function of the number of nodes.

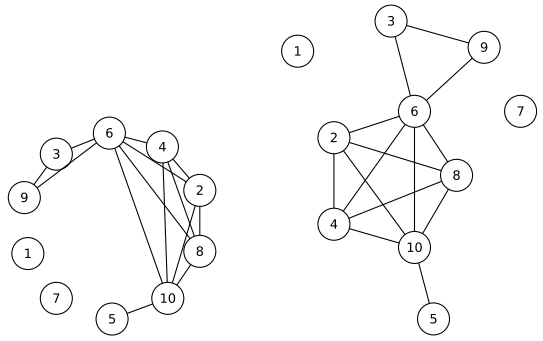


Figure 2.11. Repositioned tolerance relation of GCD

A physical metaphor is used to give a quasi-optimal solution of correlation clustering, where similar objects attract each other and dissimilar objects repulse each other. The same is used here, but the objects are not grouped but arranged on the plane (or in space). The requirements are:

- similar objects get close, and
- dissimilar objects get far from each other.

Néda et al. presented a model using electric particles to solve the problem of correlation clustering in [40]. In this model, the particles could move on a circle based on the superposition of the forces acting on the particle.

Here, imaginary springs are used. Each node of the graph moves by the superposition of the forces of its springs. To simplify the problem in this section total tolerance relations are used, i.e. any pair of objects are comparable (similar or dissimilar). To get a suitable location for each of the objects, the following constraints are defined:

- it is not appropriate, if some object hides the others, so an optimal (minimal) distance ($SIMM$) is fixed for similar objects,
- to get a finite picture, an optimal distance ($DIFF$) is determined for dissimilar objects.

It can be translated this for springs: there are short springs for similar objects and long springs for dissimilar objects. By Hooke's law, the force needed to extend or compress a spring by some distance d is proportional to that distance: $F = kd$.

Some placement of the objects can be treated as a network of these short and long springs. If two similar object (connected by a short spring) are closer than $SIMM$, then they repulse, and if they are farther than $SIMM$, then they attract each other. For the long springs, the same hold, but with $DIFF$ instead of $SIMM$, as Fig. 2.12 shows. Therefore, two functions are introduced: $f(\mathbf{d}) = (SIMM - |\mathbf{d}|)$ and $f'(\mathbf{d}) = (DIFF - |\mathbf{d}|)$, where $\mathbf{d}$ is the (distance) vector between two objects.

The periodic (sinusoidal) motion of a mass is used on a spring. One wants to get a location of objects and not a motion, so some attenuation is needed, a negative feedback.

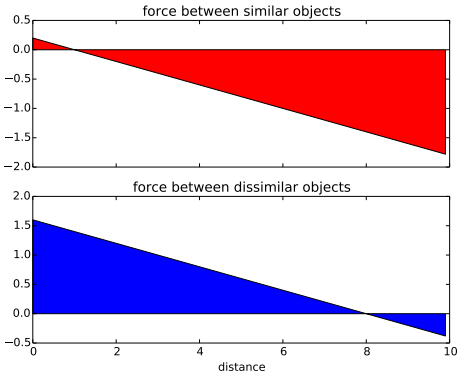


Figure 2.12. Spring functions

After some trial, the cube of the distance was found to be similar to Coulomb's rule in some sense. Finally, the resultant is the superposition of the forces:

$$\mathbf{F_i} = \sum_j \frac{f(\mathbf{d_{ij}})\mathbf{d_{ij}}}{|\mathbf{d_{ij}}|^3} + \sum_l \frac{f'(\mathbf{d_{il}})\mathbf{d_{il}}}{|\mathbf{d_{il}}|^3} \tag{2.2}$$

Here, the first part is summing for the objects similar, and the second part is for the objects dissimilar to object i.

Some relations are partial, where there are no constraints on the unrelated objects. It means that they can be at any distance from each other, so they can be positioned at the same place and partially or totally hide each other, or they can be very far from each other, so the picture can become very big. To solve these problems, a new spring function $\widehat{f}$ is introduced for the third type of springs (used for undefined values). This function is similar to the modified f', but the optimal distance is the interval $[SIMM, DIFF]$. If $d < SIMM$ then $\widehat{f}(d) > 0$, and if $d > DIFF$, then $\widehat{f}(d) < 0$, i.e. for small distances it repulses and for big distances it attracts.

After presenting the algorithm and its background, it is time to show it in practice. The algorithm (by László Aszalós) can be downloaded from
`https://github.com/aszalosl/visualize_tolerance`
The resulting image for some simple, but typical tolerance relation is shown. The first example is the snake, where adjacent objects are similar, and the others are dissimilar. One could think that the result is a straight line. Although the left side of Fig. 2.13 shows that this theory does not hold. If the dissimilar objects get too far from each other, then they attract, so an arch appears. In case of a non-defined relation for non-adjacent objects, a different image is obtained, because these objects do not repulse each other, so the snake can move in any direction, as right side of Fig. 2.13 shows.

Fig. 2.14 shows some total trees. Here each non-leaf node has exactly three successors. In the first two pictures, the non-adjacent nodes are dissimilar, and in the last one, the non-adjacent nodes are unrelated. The tree on the left is totally symmetric (if the difference of

Figure 2.13. Total and partial snake.

the size of nodes is omitted according to one and two digits). But not every run gives such a nice picture. The middle one was generated with the same parameters as the left one, but the nodes 8 and 13 got to the wrong place at the beginning, and since this layout is stable, these nodes cannot escape. In the last picture the non-adjacent nodes, since unrelated, do not need to get far from the other nodes, e.g. nodes 3 and 4 comply with the minimal distance constraint, so they are in a stable state.

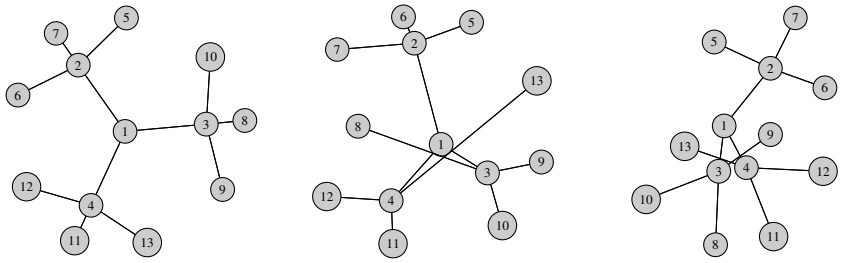


Figure 2.14. Total trees.

Of course, the shapes of these pictures can be formed by changing the values of parameters $SIMM$, $DIFF$. Several combinations were tested, to get specious pictures, e.g. in the case of Fig. 2.15. Here, the GCD relation was presented but did not connect the similar nodes. From the numbers, the reader can reconstruct these lines. The result of the correlation clustering for a few nodes gives a partition where each prime number (plus the 1) has a cluster, and each number gets into its smallest prime divisor's cluster. Although, the collision of nodes is inappropriate (when they partly hide each other), but in this case, it clarifies the picture.

In the middle of the picture are the even numbers. On the right are the multiplies of three. 5 is just below 25 (it can be known from the data of the image), as 7 is below 49. As the multiplicity of the divisors is not counted, the power of primes get to almost the same position (as 5 and 7 have shown). On the edge of the picture are the large prime numbers, because they differ from every other number, and as they are dissimilar to each other, they are positioned uniformly. It is worth to examine the subtleties of the picture: 35 is positioned between the numbers of clusters of 5 and 7, but it is slightly moved towards the even numbers because there are eight similar number (five numbers are divisible by five and three are divisible by seven), and three similar numbers in the cluster of 3. Similarly, the number 50 is slightly moved towards the numbers of the cluster of 5 and the number 48 towards the numbers of the cluster of 3. By zooming the picture, other subtleties could be explored, too.

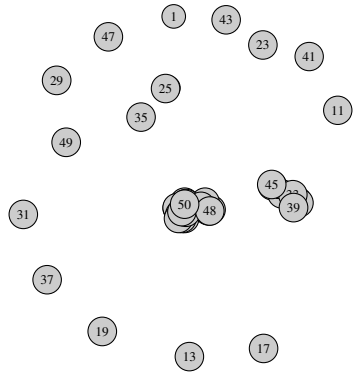


Figure 2.15. Visualization of the relation GCD.

Finally let us consider a picture that does not use abstract concepts of number theory, but comes from real life. Fig. 2.16 denotes the members of two departments. Two researchers are treated as similar, if they are co-authors, and dissimilar if there is no such third person who is co-author to both. Of course, this is a partial tolerance relation, because if x-y and y-z are pairwise similar, but x and z are not similar, then one cannot say that they are dissimilar according to the definition. Numbers 1-10 and 11-19 denote the members of the departments, respectively. The center of the picture is empty, hence the research areas are orthogonal. The numbers of the second department have higher densities, so their publication is stricter in the same themes mostly by the same co-authors. The numbers of the second department can be grouped into three clusters, and there are not many relations between these clusters.

Researchers 5, 6, 9 and 13 usually publish alone or with external colleagues, hence it is no wonder that they are alone in the picture. Researchers 1, 2, 7 and 8 wrote a common article, so they construct a strong core. As they publish with other authors too; they are positioned more widely on the picture according to the repulsion of other co-authors. From this group, 1 and 2 are the only co-authors of 4. Moreover 4 is the regular co-author of 10 and has no other co-author. Hence 7 and 8 repulse 10, who gets far from 4, and the attraction of 10 moves 4 away from its other co-authors. In the case of the other department, there is an attraction between 12 and 19 (they moved toward each other), but the chain of co-authors generates a repulsion, so they cannot get any closer.

This simple method could visualize complex systems. In this model, neither the quality and quantity of the common publications nor the date of these publications were taken into account.

2.5.2 Representative Members [7]

Instead of examining the entire population, polls usually only survey a small sample. This can be done because the results obtained are very close to what one would get by examining the entire population. However, the sample should be selected carefully. Many people think that the larger the sample, the better which is not true. The sample is representative in some

[7] The work described in this section was based on [35, 6, 7, 22]

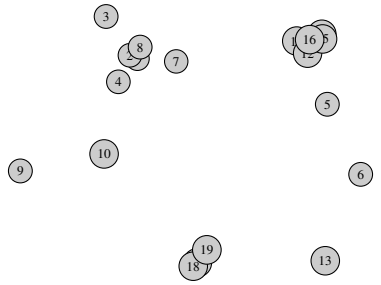


Figure 2.16. Research activities of two departments.

respects, i.e., the specific properties are as similar in the sample as in the entire population. The sample can be representative in one aspect, while not representative in another. There are various standard methods for determining a sample.

If the population is significantly inhomogeneous, i.e. it has high variability according to the survey, then the stratified (random) sampling can be used. In this case, the population is divided into several sub-populations (strata), where these sub-populations are homogeneous according to the examined criteria. From a homogeneous strata, the individuals can be randomly selected to be sampled (i.e., the representative of the group), typically in proportion to the size of the group.

If an object can be represented with a vector of numbers, the difference of vectors belonging to each object can be considered, where this difference/distance usually meets the requirements of metrics. Using this distance function, many clustering methods have been developed over the last sixty years. The most well-known is the k-means method in which a cluster is represented by its centre (one representative). The k-medoids algorithm is a version of this k-means method, and it replaces the cluster with the sample element closest to the cluster centre. The CURE method (clustering using representatives) goes one step further, replacing non-ellipsoid clusters with a maximum c sample elements. The k-means algorithm can be used in the k-nearest neighbours (k-NN) classification algorithm, where newly added objects are categorized into an existing cluster/class. Since comparing the new elements with all stored elements in a large database is a time costly task, by replacing the elements of the clusters with some of their representatives the complexity of the classification of new elements can be significantly reduced.

Polls cannot ask too many questions from a person because their patience is finite. However, there are cases where one leaves behind a lot of information. Think for example our medical cards, our data stored at different kinds of service providers, or our digital footprint on the social network. In these cases, it is not worth transforming this information into a unified form in order to be able to define the differences between the data of objects. It is much easier to directly decide for two given objects whether they are similar or not.

In this section, a mathematical method is presented which—having an existing partition and tolerance relation (based on similarity) –determines which is the most typical object in a given cluster, i.e. which one can be considered representative. A real number, a rank is assigned to each of these objects which determines the "representativeness" of the objects.

Here, two possible ways of choosing the representatives are proposed.

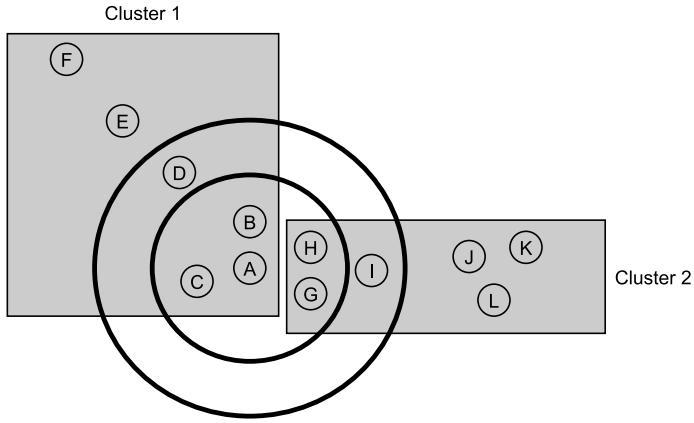


Figure 2.17. α and β values for the member A

2.5.3 First Method [8]

In this method, an object is called a representative if it is similar to most and different from the least of the members in its group.

For any member x four values are stored:

- α - the number of elements that are similar to x and are in the same cluster.
- β - the number of elements that are different from x and are the same cluster.
- γ - the number of elements that are similar to x and are in different clusters.
- δ - the number of elements that are different from x and are in different clusters.

Fig. 2.17 shows a very simple example to the method. In this example, the tolerance relation (based on similarity) is based on the Euclidean distance of the objects. The smaller circle denotes the similarity threshold and the greater one denotes the difference threshold. For member A the four values are:

- α = 3. Because there are four members (A,B,C) that are similar to A.
- β = 2. Because there are two members (E and F) that are different from A.
- γ = 2. Because there are two members (H and G) that are similar to A and are in a different cluster;
- δ = 3 Because there are three members (J, K and L) that are different from A and are in a different cluster.

[8] The work described in this section was based on [22]

In the first case, a member can be considered a possible representative if the following fraction (rank) is maximal:

$$r_1 = \frac{\alpha^w - \beta^v}{\alpha + \beta + 1} \, v, w \in \mathbb{R}, v, w \geq 1, w \geq v \tag{2.3}$$

In the second case, a member can be considered a possible representative if the following fraction (rank) is maximal:

$$r_2 = \frac{\alpha^w - \beta^v}{\alpha + \beta + u * \gamma + 1} \, u, v, w \in \mathbb{R}, v, w \geq 1, w \geq v \tag{2.4}$$

If two arbitrary objects have the same r_2 value, then the δ value decides.

The values r_1 and r_2 show how much an object represents a given cluster. In the case of r_1, only the similarities and differences in the given cluster are considered. Its value is high if the object is similar to the others in the cluster which means that there are only a few elements that are different from the given object. r_2 takes into account every cluster. The r_2 value of an object is high if it is similar to most of the elements in the same cluster and there are very few objects that are similar to the given object but are in different clusters. Those cases, when the object is similar to an element that is in a different cluster, are punished. That is why the γ value is only in the denominator. In some cases, it can happen that the similarity and the difference cannot be taken with the same weights. In the formulas, the u, v, w denote these weights.

The first method can be used when the members of the other groups do not matter. For example, let us assume that the objects are patients. Here the similarity is based on sharing some common symptoms. If the patient, who is the most similar to the others, needs to be found, then the patients from the other groups are irrelevant. For instance, if the task is to find a new possible way to cure a certain disease that a group of patients has, then it can be useful to test it on the representative patient first. In this case, the other group of patients is not relevant because they have different symptoms. The second method can be used when the members of the other groups matter. Let's assume that the objects are members of a political party. The similarity here can be based on the political view. Two politicians can be treated as similar if they share the same idea and different if they have different opinions. The leader of a party is expected to be similar to the others in the same party but different from the members of the other parties. Another good but a little extreme example is if the objects are members of an organized crime family. Two gangsters are similar if they like each other and different if they do not. The boss of the family should be liked in the family but disliked in the other families.

Fig 2.5.3 shows the difference between the two methods. In the left side of the figure, the first method was used. Member A was the representative of $cluster1$, because there are seven objects that are similar to A and no such ones that are different from A ($\alpha = 7, \beta = 0, \gamma = 2, \delta = 2, v = 2, w = 2, u = 1$). So its r_1 value is maximal. In the right side of the figure, the second method was used. Here the member F was the representative of $cluster1$. Its r_1 is less than that of member A, because it has only 6 similar objects. However, the r_2 value is higher than that of member A, because it has no such objects that are similar to it and are in a different cluster, while the member A has 2 objects (I, J) that are similar to it and are in $cluster2$ ($\alpha = 6, \beta = 0, \gamma = 0, \delta = 2, v = 2, w = 2, u = 1$).

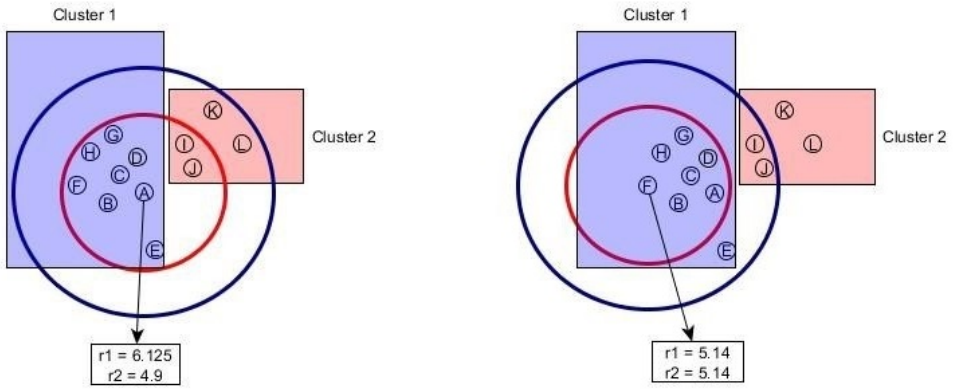


Figure 2.18. Difference between r_1 and r_2 maximization

The method was introduced because it is very simple and it can be easily used. However, every member has the same vote. It means that each object decreases or increases the rank of the other members exactly the same. Let us assume that object A and B, C are in different clusters and A is different from B and C. Let us also assume that the rank of B is much higher than that of C. In this method, B and C decrease the rank of A and this decrease is the same in both cases, even though that the rank of B is much higher than that of C (B should decrease it much more). To solve this issue, another method is proposed.

2.5.4 Second method – Ranking Algorithm [9]

In this section, a method is shown on how to describe relations using signed graphs. Each element in a relation is a vertex of the graph and two vertices are connected with an edge if and only if their two corresponding elements are in relation.

In the case of graphs, one can speak of the distance between two vertices (as the shortest path between the two vertices), but it carries much less information than the difference between two large vectors. Therefore, the similarity information should already be included in the graph, so the graph will correspond to a tolerance relation (based on similarity). As there are usually partial tolerance relations (based on similarity) in practice, there will be edges in the graph that denote the similarity and there will be ones that denote the dissimilarity. The partiality is represented by missing edges.

For example, links between individual websites or citations between scientific articles define a directed graph, i.e. a partial tolerance relation (based on similarity), but there is no representation of the dissimilarity.

Google's PageRank algorithm [2] is a great example of a ranking system on directed graphs. Considering the web pages (vertices) and the links between them (edges) as a directed graph, the boundary distribution of the random walk on the graph gives the rank of each page.

[9] The work described in this section was based on [6, 7]

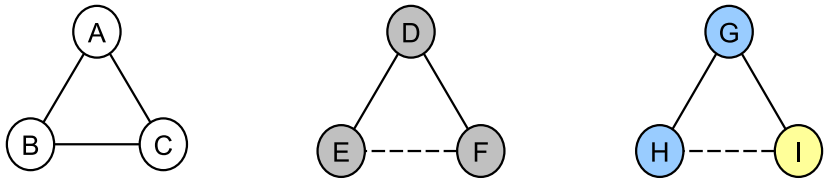


Figure 2.19. Simple ranking problems

For example, if the web page p is more likely to be accessed than the web page q, then the rank of p will be higher than that of q and will be ranked higher in the hit list. Here, if a page with a low rank refers to a page with a high rank, or a novice author refers to a well-known author in his article, it raises the rank of the page/author to a higher rank, but—through a non-symmetrical relation—this reference has no effect on the rank of the page/author with the lower rank.

However, if the graph is not directed, the edge between the two vertices affect the rank of both vertices. Since similarity is represented by a tolerance relation (reflexive, symmetric, but not necessarily transitive), the associated graph is not directed.

Let's see our (naive) expectations of a ranking method. In Fig. 2.19 on the left, there are three vertices (A, B and C) in a common cluster. In this figure, a cluster is represented by vertices of the same colour, while the similarity of vertices is denoted by a solid, and the difference by a dashed line. Because in this sub-graph each element is similar to each other, the same rankings are expected due to symmetry. In the middle graph of Fig. 2.19—where D, E and F are in a common cluster—a difference appears. This graph is called a minimal frustrated graph, because there is no such partition of vertices where similar elements are common, and different elements are clustered separately. In this graph, vertex D has only similar vertices, while vertices E and F both have similar and also different vertices. The fact that an object differs from an object in its own cluster reduces the rank of the object/vertex and thus the chance of being a representative of the cluster. Conversely, if an object is similar to an object in its cluster, then this increases its rank. Based on these, this cluster will be represented by vertex D because it has the highest rank. Moreover—according to the symmetry—the rank of vertices E and F should be the same.

Finally, take the graph on the right side of Fig. 2.19. Here, the vertices were divided into two clusters: $\{G,\ H\}$ and $\{I\}$. The fact that the vertices of G and I are similar, but are found in different clusters also reduces the rank of both vertices, because similarity to vertices in other clusters means deviation from the idealized characteristics of the group. The vertex h is similar to vertex G which belongs to H's cluster, and H is different from the vertex I belongs to another cluster. This latter also raises the rank of the vertex H and hence H becomes the representative of its own cluster. In the other cluster, the only vertex will be the representative.

Based on the examples above, the similar objects of the same cluster and dissimilar objects of other clusters can be called the fosterer of the object, while the similar objects of different clusters and dissimilar objects of the same cluster can be called the adversary of

the object. The fosterer objects help an object become a representative, while the adversary objects prevent it from happening.

Here, a list can be seen of what is expected from the rankings.

- Be symmetric, that is, if two vertices have the same number of vertices of the same rank in the same type of relation (fosterer or adversary), then their rank is the same.
- The rank of a particular item is immediately raised if:
  - one of its fosterer object increases in rank, or
  - one of its adversary object falls in rank, or
  - a new fosterer object appears.
- The rank of a particular item is immediately reduced if:
  - one of its adversary object increases in rank, or
  - one of its fosterer object falls in rank, or
  - a new adversary object appears.
- It does not directly change the rank of a particular item if:
  - another object that is not compared to it or is incomparable, appears in any cluster, or
  - the rank of such an object changes.

If a new object that is dissimilar to the current object, but similar to another object of the cluster, is added to this cluster, then it raises the rank of objects that are similar to it. This will have a ripple effect on objects that are similar to objects that are similar to the new object, and so on. Therefore, if this should be represented by an algorithm, then an iterative method, that would escalate these effects step by step, would be needed. On the other hand, since almost every object is related to every object, the rank of all objects should be treated altogether.

The ranking method

Let U denote the set of objects/vertices, and for simplicity, denote the objects with numbers: $U = \{1, 2, \ldots, n\}$. The set of clusters means a partition. This partition is interpreted as a function—denoted by p—that assigns a number to each object, so $p : U \to \mathbb{N}$. The objects x and y are in the same cluster, if $p(x) = p(y)$; and they are in different clusters, if $p(x) \neq p(y)$. Our tolerance relation (based on similarity) is a (possibly partial) tolerance relation $\mathcal{R}$—that is reflexive, symmetric, but not necessarily transitive.

Social ranking

A similar approach as PageRank or as various evaluation sites is used (accommodations, restaurants, marketplaces), where the rankings of individual websites, hotels, restaurants are summed up by aggregating individual ratings.

The rank of each object could be taken as the difference between the number of fosterer and adversary objects, but a more sophisticated method would be more appropriate in practice. There is a significant difference between the cases where an adversary object is the representative of another cluster, or it just a marginal object there. In the former case, it decreases the rank of the current object to a greater extent. A value proportional to the rank of the adversary or fosterer objects could define a good amount by which the rank of the current object can be decreased or increased.

The rank of object i is determined by its relation to all objects. Let m_{ij} indicate the relation (fosterer or adversary) between objects i and j. Each object is considered with its rank as weights, and so the following relation needs to be solved for values r_i: $r_i = \sum_j m_{ij} \cdot r_j$ for all $i \in U$.

These equations combine to $R = MR$ in matrix notation.

So that different values m_{ij} must not be used from task to task–depending on its size– a constant c is introduced to normalise the values m_{ij}. This changes the previous relation as follows $r_i = \sum_j (c \cdot m_{ij}) \cdot r_j$, but it can be transformed to $r_i = c \cdot \sum_j m_{ij} \cdot r_j$, which gives us $R = cMR$. If one takes the reciprocal of c to be denoted by λ, then a known relation: $MR = \lambda R$ appears, i.e. R is an eigenvector of the matrix M.

Taking into account the ideas from the previous chapter, the normalized value m_{ij} should be 1 for fosterer objects, -1 for adversary objects, and 0 for all other cases (where the tolerance relation is partial).

According to the much-cited example, a bald man does not resemble a hairy man, although an uprooted hairline does not change a person. As a person can have up to two million hairlines, the linear model of this example is reduced to only four states. Here 1 (bald) and 4 (hairy) correspond to the two end states, while 2 and 3 are two intermediate states. 1 and 2 are in a common cluster, and so are 3 and 4. Fig. 2.20 shows two graphs demonstrating this problem, where clusters are denoted by assigning different shades to the vertices. The similarities (denoted by solid lines) are the same in both cases, but the dissimilarities (denoted by dashed lines) have changed: dissimilarity appears between the second neighbours in the latter case. Hence, the first graph/relation is partial, and the second is total.

Two matrices based on the relations and partitions for each graph are:

$$M_1 = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & -1 & 0 \\ 0 & -1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix} \text{ and } M_2 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

The only adversary relation is between 2 and 3, because they are similar, but are in different clusters. In the first case, 1-3 and 2-4 are not comparable, so the corresponding values of the matrix are 0. In the second case, these pairs of objects are dissimilar, but they are in different clusters, so these are fosterer relations.

Calculating the eigenvalues and eigenvectors for the first graph, the following is obtained:

r_1	r_2	r_3	r_4	λ
0.7071	0.7071	0.3344	0.1368	2.4142
0.5000	-0.5000	0.6770	-0.5873	-0.4142
0.0000	0.0000	-0.6230	-0.6938	2.4132
0.5000	-0.5000	-0.2041	0.3939	-0.4142

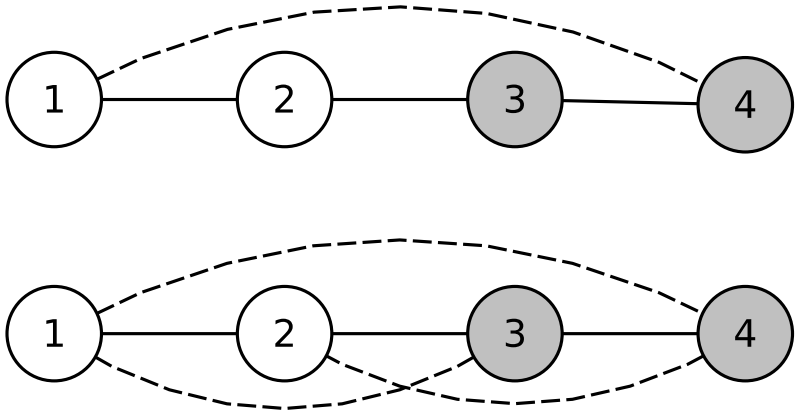


Figure 2.20. Two simple symmetric linear model

The graph is symmetric, so it is expected, that $r_1 = r_4$ and $r_2 = r_3$. Unfortunately, none of the eigenvectors satisfy this. Therefore, another method is needed.

Power-method

The algorithm of von Mises [34] for a diagonal matrix M results in the biggest eigenvalue (with the highest absolute value), and the corresponding eigenvector. The method starts with an arbitrary vector R_0 that in our case should be $\mathbf{1} = (1, \ldots, 1)^T$. Then R_{k+1} is determined as follows: the rank vector R_k is multiplied by the matrix M and normalized as shown by (2.5).

$$R_{k+1} \leftarrow \frac{MR_k}{||MR_k||} \tag{2.5}$$

Unfortunately, this iteration is converges slowly, but it is easy to use even for large sparse matrices. This is why it is used in PageRank implementation. If $R_i \approx R_{i+1}$, the method is stopped and the values in the vector R_i are considered the rank of the objects. If matrix M has an eigenvalue that is strictly greater in magnitude than its other eigenvalues, then R_i converges.

If this method is applied to the matrices shown in Fig. 2.20, then the result are: $R = (1, 0.414, 0.414, 1)^T$ and $R = (1, 0.618, 0.618, 1)^T$, where $r_1 = r_4$ and $r_2 = r_3$ as it was expected. These values also fit to the naive ideas: for the first graph, for object 1 both objects 2 and 4 are fosterers, while object 1 is a fosterer and object 2 is an adversary for object 2. So the expected relation $r_1 > r_2$ is fulfilled. In the second case, when there are more fosterer relations, the rank gained is also higher for objects 2 and 3.

Summarising it gives:

$$R_{k+1} \leftarrow \frac{MR_k}{||MR_k||} \approx \frac{M^{k+1}\mathbf{1}}{||M^{k+1}\mathbf{1}||}$$

Algorithm 2 Python implementation of the ranking method

```
def power_method(M, eps = 1e-9):
    N = M
    R = np.ones((len(M),))
    N2 = N@N
    N2 /= m = np.max(N2@R)
    while np.linalg.norm(N@R - N2@R) > eps:
        N, N2 = N2, N2@N2
        N2 /= np.max(N2@R)
    return N2
```

If k is a power of 2, then the same values can be calculated by repeated squaring using the following recurrence relation:

$$B_1 \leftarrow M \text{ and } N_{i+1} \leftarrow \frac{N_i B_i}{||N_i B_i||} \tag{2.6}$$

If $N_i\mathbf{1} \approx N_{i+1}\mathbf{1}$, then let $R \leftarrow N_i\mathbf{1}$. Not surprisingly, the two calculations give the same result.

Based on these, it is not difficult to write the ranking program in Python using the services of the Numpy package (Algorithm 2). Remark, that for Numpy, the operator `@` is the matrix multiplication operation.

Fig. 2.21 shows how each value r_i changes for a random matrix M. Here, the axis x represents the number of applications of (2.6), while the axis y represents the current values of the ranks. Due to normalisation, the highest rank is always 1, but as it can be seen from the chart, the rank of the objects changes from time to time. The algorithm is terminated when the ranks cease changing.

Ranks of numbers

As there is no standard tolerance relation (based on similarity) for larger databases, the various clustering/classification methods are usually tested on random graphs [39]. Here the following relation is used: let two numbers be similar if they have a non-trivial common divisor, i.e. $\gcd(x, y) > 1$, where $x, y \in \mathbb{N}^+$. 1 is considered to be similar to itself. Then $4\mathcal{R}6$ and $6\mathcal{R}9$ are fulfilled (the common divisors are 2 and 3), but $4\mathcal{R}9$ is not, so this relation is not transitive. If someone is interested in the correlation clustering of numbers $1, \ldots, n$, it can be easily formulated if $n < 111\,546\,435$, otherwise the situation becomes complicated [4].

In the following, the rank of each element is determined by using the optimal clustering of the set of numbers, except in the first case, where the numbers $1, \ldots, 12$ are placed in a common cluster.

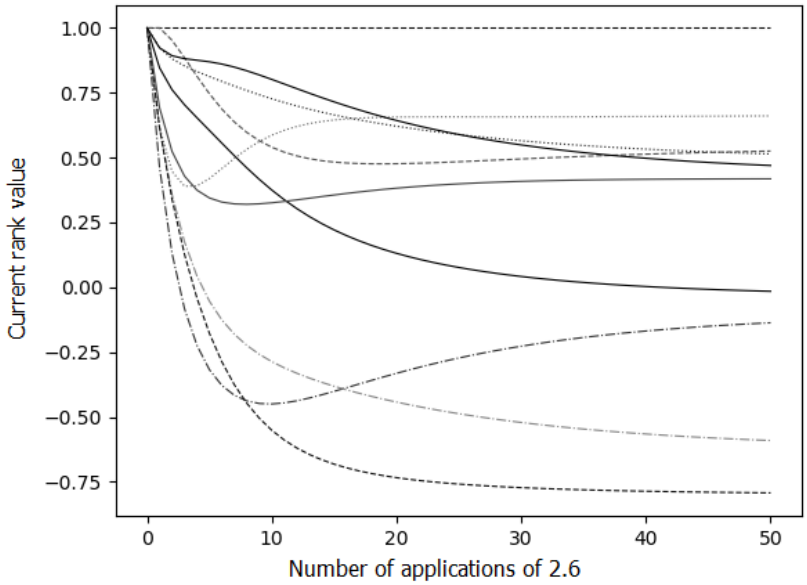


Figure 2.21. Changes in the rank of the objects in a random tolerance relation

As there is a single cluster in the equation (2.6), the matrix M can be replaced in the calculation with the matrix of the relation $\mathcal{R}$ in Fig. 2.22. In this matrix $\mathcal{R}$, the relation between the numbers i and j is given by the j^{th} number from the i^{th} row: similar numbers are denoted by 1, dissimilar numbers by -1. 12 and 6 are similar to the multiples of 2 and 3, these are eight numbers including themselves and different from four of them (1, 5, 7, and 11). The powers of 2 are similar to every even number (six numbers) and different from all odd numbers (six numbers). The powers of 3 (itself and 9) are similar to four numbers and different from eight numbers. 5 is similar to its duplicate, but there is no such number for 7 and 11 and for 1. Because in each case the numbers mentioned together are similar to the same numbers, so—according to the symmetry—their rank is the same (Table 2.9/A). In this table, the ranks are rounded to the closest hundreds, to fit on the page.

If one does the same calculation for numbers $1, \ldots, 100$ (Table 2.9/B), then the numbers will typically increase. There are many more even numbers which will increase the rank of even numbers, but they have the opposite effect on the rank of powers of 3. In this set, there are numbers similar to 7 or 11, so their rank increases; and there are more such numbers for 7, so its rank grows more.

Consider the partition of natural numbers obtained by correlation clustering [4]. The largest such cluster is the set of even numbers. This is followed by a set of odd numbers divisible by 3; next, the set of numbers which are divisible by 5, but not by 3 or 2, and so on. This can be formulated as $[2] = \{x \in U : 2|x\}$, $[3] = \{x \in U : 3|x\} \setminus [2]$, $[5] = \{x \in U :$

Table 2.9. Ranking numbers...

A) $1, \ldots, 12$ in a common cluster.

1	2	3	4	5	6	7	8	9	10	11	12
-0.62	0.97	0.06	0.97	-0.32	1.00	-0.62	0.97	0.06	0.89	-0.62	1.00

B) $1, \ldots, 100$ in a common cluster.

1	2	3	4	5	6	7	8	9	10	11	12	...
-0.55	0.99	-0.03	0.99	-0.28	1.00	-0.37	0.99	-0.03	0.95	-0.44	1.00	...

C) $1, \ldots, 12$ using optimal partition.

[1]	[2]						[3]		[5]	[7]	[11]
1	2	4	6	8	10	12	3	9	5	7	11
1.00	1.00	1.00	0.73	1.00	0.84	0.73	0.73	0.73	0.84	1.00	1.00

D) $1, \ldots, 15$ using optimal partition.

[1]	[2]							[3]			[5]	[7]	[11]	[13]
[1]	[2]							[3]			[5]	[7]	[11]	[13]
1	2	4	6	8	10	12	14	3	9	15	5	7	11	13
1.00	1.00	1.00	0.67	1	0.79	0.67	0.86	0.79	0.79	0.54	0.79	0.86	1.00	1.00

E) $1, \ldots, 12$ using common cluster and the weakened relation.

1	2	3	4	5	6	7	8	9	10	11	12
-0.53	1.00	0.06	0.83	-0.34	0.68	-0.53	0.70	-0.17	0.42	-0.53	0.81

F) $1, \ldots, 12$ using optimal partition and the weakened relation.

[1]	[2]						[3]		[5]	[7]	[13]
1	2	4	6	8	10	12	3	9	5	7	11
1.00	1.00	0.89	0.53	0.83	0.53	0.62	0.77	0.88	0.89	1.00	1.00

$$\mathcal{R} = \begin{pmatrix} 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ -1 & -1 & 1 & -1 & -1 & 1 & -1 & -1 & 1 & -1 & -1 & 1 \\ -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 \\ -1 & 1 & 1 & 1 & -1 & 1 & -1 & 1 & 1 & 1 & -1 & 1 \\ -1 & -1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ -1 & -1 & 1 & -1 & -1 & 1 & -1 & -1 & 1 & -1 & -1 & 1 \\ -1 & 1 & -1 & 1 & 1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 \\ -1 & 1 & 1 & 1 & -1 & 1 & -1 & 1 & 1 & 1 & -1 & 1 \end{pmatrix},$$

$$M = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 \\ 1 & 1 & -1 & 1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & 1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 \end{pmatrix}$$

Figure 2.22. tolerance relation (based on similarity) on numbers $1, \ldots, 12$, and the suitable matrix based on the optimal partition.

Table 2.10. Rank of numbers $1, \ldots, 1000$ using the optimal partition

x	$r(x)$	$r'(x)$
1	1.0	1.0
2	1.0	1.0
3	0.71587172	0.79465596
5	0.78447065	0.81182099
7	0.82770503	0.84160798
11	0.88038405	0.88841938
13	0.89607481	0.90279018
17	0.91498232	0.91946078
4	1.0	0.8102978
6	0.75571551	0.55292589
8	1.0	0.71548803
10	0.85121315	0.5761841

$5|x\} \setminus [2] \setminus [3], \ldots$. Of course, each prime number may have one cluster.

If the fosterer and adversary objects are considered based on the optimal partition and the tolerance relation (based on similarity) $\mathcal{R}$ in Fig. 2.22, then the matrix M presented in Fig. 2.22 is obtained. The partition mentioned above is optimal because it minimises the number of negative numbers in the matrix M. This, of course, also has an impact on rankings. While keeping the numbers in a common cluster, multiple ranks were negative due to dissimilarities, by using the optimal partition each cluster as a cluster by applying similarity (Table 2.9/C). For singletons (containing big primes and 1) the rank 1.0 is a proper value, as there are no similar numbers. In the set of even numbers, the rank of the powers of two will also be 1.0, as they are similar to all even numbers, and the $[2]$ cluster has only even numbers, and all even numbers are here. Numbers with other divisors will have similar numbers in other clusters. The more prime divisors a number has, the more clusters contain similar numbers, so its rank will be reduced. In the cluster $[3]$, the rank of the powers of three will be the highest, but it will not reach level 1.0, because there are nearly as many numbers that are dividable by 3—that is, similar—in the cluster $[2]$, as in the cluster $[3]$.

If the method on numbers $1, \ldots, 15$ is applied instead of on numbers $1, \ldots, 12$, then the ranks change (Table 2.9/D). The rank of 7 fell, as its adversary number (14) appeared. As number 15 is an adversary for both 5 and 10, it reduces the rank of both of them. The reader may be wondering how these ranks look when there are more numbers, e.g. $1 \ldots, 1000$ (Table 2.10, column r). Perhaps it is clear from above that in the case of an optimal clustering, the representatives of the individual clusters come from the powers of primes. Each power of a prime is given the same rank because it is completely symmetrical in terms of similarity.

$$\mathcal{R}' = \begin{pmatrix}
1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\
-1 & -1 & 1 & -1 & -1 & 1 & -1 & -1 & 1 & -1 & -1 & 1 \\
-1 & 1 & -1 & 1 & -1 & 0 & -1 & 1 & -1 & 0 & -1 & 1 \\
-1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 \\
-1 & 1 & 1 & 0 & -1 & 1 & -1 & 0 & 0 & 0 & -1 & 1 \\
-1 & -1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & -1 \\
-1 & 1 & -1 & 1 & -1 & 0 & -1 & 1 & -1 & 0 & -1 & 0 \\
-1 & -1 & 1 & -1 & -1 & 0 & -1 & -1 & 1 & -1 & -1 & 0 \\
-1 & 1 & -1 & 0 & 1 & 0 & -1 & 0 & -1 & 1 & -1 & 0 \\
-1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & -1 \\
-1 & 1 & 1 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & -1 & 1
\end{pmatrix}$$

$$M = \begin{pmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 & 1 & 1 & -1 \\
1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 \\
1 & 1 & -1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\
1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\
1 & 1 & 1 & 0 & -1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & -1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1
\end{pmatrix}$$

Figure 2.23. Weakened tolerance relation (based on similarity) on numbers $1, \ldots, 12$, and the suitable matrix based on the optimal partition.

Weakening the tolerance relation

The method was also executed on a different relation. Here two numbers are similar if one of the numbers is a divisor of the other number instead of the existence of a real common divisor. The partial relation here is made from a complete relation. If two numbers were dissimilar at the original tolerance relation, they will be dissimilar at the weakened relation, too. Moreover, numbers 4 and 6 were similar before, but not anymore. Therefore, the relation holds less often.

At first glance in Table 2.9/E the ranks of 2, 4, 8 are different, as well as ranks of 3 and 9. Once there is only one cluster, it is sufficient to count how many positive and negative values are in each row of the first matrix in Fig. 2.23. The number 2 is similar to every even number, that is, to every object in its cluster $[2]$, and there are no other similar numbers anywhere else. The number 4 is similar to half of the numbers of its cluster, but not dissimilar to any numbers in this cluster. The number 8 is similar to a quarter of the numbers of its cluster, etc. The fact that some ones were replaced with zeros, the symmetry disappears. The numbers 1, 7 and 11 are dissimilar from all the other numbers in this case, so now they have the same (negative) rank.

An interesting question can be to see what happens if the previous optimal partition (Table 2.9/F) is taken. The previous asymmetry remains. The primes have the highest rank, and the powers of primes have lower ranks. To see this tendency, let's see the outcome of ranking $1, \ldots, 1000$ (Table 2.10, column of r').

If the optimal partition of tolerance relation $\mathcal{R}$ is applied to the weakened tolerance relation $\mathcal{R}'$ (Table 2.9/F), the number of negative numbers in matrix M will also be significantly reduced. However, because of the change in the relation, the symmetry within the clusters is severely damaged which also affects the rankings. Therefore, when some ranks weaken in the $[2]$ cluster, this may affect elements in other clusters. If there exist fosterer elements in other clusters, these will in turn increase in rank.

Which partition could be optimal for this weakened relation? The number of similar and dissimilar numbers in the clusters should be taken into consideration according to some

given number. If the difference between these numbers not in their cluster is maximal, by moving the actual number to the maximal cluster, a better partition will be obtained. Let $U = \{1, \ldots, 1000\}$, and $n = 75$, which is $3 \cdot 5^2$. With the original relation $\mathcal{R}$, $[2]$ contains 267 numbers that are similar to n (which are divisible by 3 or 5), and 233 which are dissimilar to it. $[3]$ contains 167 similar numbers (all of them divisible by 3) and $[5]$ contains 67 similar numbers. At the weakened relation $\mathcal{R}'$, the previously dissimilar numbers remain dissimilar, and in $[2]$ there are no divisors of 75, just its even multiples. These, by definition, are similar to it, so $[2]$ contains 6 members, similar to 75. In $[3]$ there are three divisors of 75 (including itself), as well as 6 of its odd multiples, so $[3]$ contains 9 similar numbers. In $[5]$ there are two divisors of 75 and it does not contain any of its multiples (because any multiple needs to have a prime divisor 3, so it would either be in $[3]$, or in $[2]$ if the number is even). If the difference between similar and dissimilar numbers is taken for each cluster, this will be maximal in the number's cluster, as here for $[3]$. This means that this kind of partition is stable for $\mathcal{R}'$, and with a very high probability that it is the optimal partition, but this needs to be proven.

2.5.5 Selecting the Representatives [10]

In many applications, however, it might not be enough to have only one representative in each set of objects. Fig. 2.24 shows a very simple example of this problem. Clearly object A has the highest r value so it is the most representative object of the set. However, it is only similar to objects B, C, D, E, F and G and does not have any kind of connections with the rest of the objects. So the aforementioned property for samples is not satisfied as object A alone cannot represent the entire set.

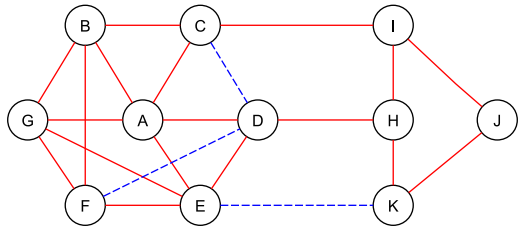


Figure 2.24. Multiple representatives are needed

Three possible ways are offered to generate more than one representative from which only the third option proves to be appropriate for real-world applications. A representative member x is said to cover the member y if x is similar to y.

1. The user gives a threshold value k. Then k percent of random objects are treated as representatives.

2. The user gives an interval for the rank values. If the rank of an object is in this interval, then it is considered as a representative.

3. Use an algorithm based on similarity to generate the necessary number of representatives.

The main issue with the first option is that the user must have some knowledge about the given set to choose an optimal k value. Due to randomness, critical information may get disregarded. Another problem is that sometimes one object can be enough to represent the whole set, but the user forces the system to choose additional representatives.

Similar problems arise with the second option. It can be hard to choose a proper interval. A more important issue is that the first few points selected will always be the ones with the highest rank. This way, some of the representatives may be picked from an already covered set of points. This is redundant, and some of the points may be left uncovered.

The pseudo-code of the third option can be seen in Algorithm 3. The input of the function is a set of data points D and the output is the set of representatives REP. It is an iterative method, in each step the algorithm keeps a record of the covered objects (i.e. the objects that are similar to one of the representatives) which is empty in the first iteration (line 6). In every iteration, the object with the highest rank is selected from the uncovered objects (line 10-15).

[10] The work described in this section was based on [35]

In line 16-20, the data points, that are covered by the currently selected representative, are inserted into the set C. At the end of each step, the chosen representative is moved into the set REP. The algorithm stops when there are no uncovered members left.

The strength of the method is that it uses the similarity between objects, and so it generates the optimal number of representatives. The other two methods can create too few or too many representatives. Another advantage is that it does not need any user-defined parameters. The algorithm can be treated as a directed sampling method which can be a very powerful tool in many applications.

A political party contains members that share a common political ideology. However, in some parties, it can happen that even though the members follow the same vision, there are some disagreements. So the group can be divided into smaller groups. In this case, one politician is not enough to represent the entire party. The aforementioned algorithm could be a solution as it takes into account the variety of the members.

Fig. 2.5.5 presents the steps of the algorithm for the data set shown in Fig 2.24. The grey ellipses contain the covered objects by a chosen representative member. In the first step, object A is chosen. In the second step, objects B, C, D, E, F and G are not considered as they are covered by A. The second method, mentioned at the beginning of this section, could have chosen B or G as possible representatives because they have the second-highest r value. Naturally, it is pointless to select them because the object A makes it redundant (both of them are similar to A). After four steps, the algorithm finishes and the four representatives are objects A, K, H, I. It can be easily seen that these 4 members share the diversity of the original data set.

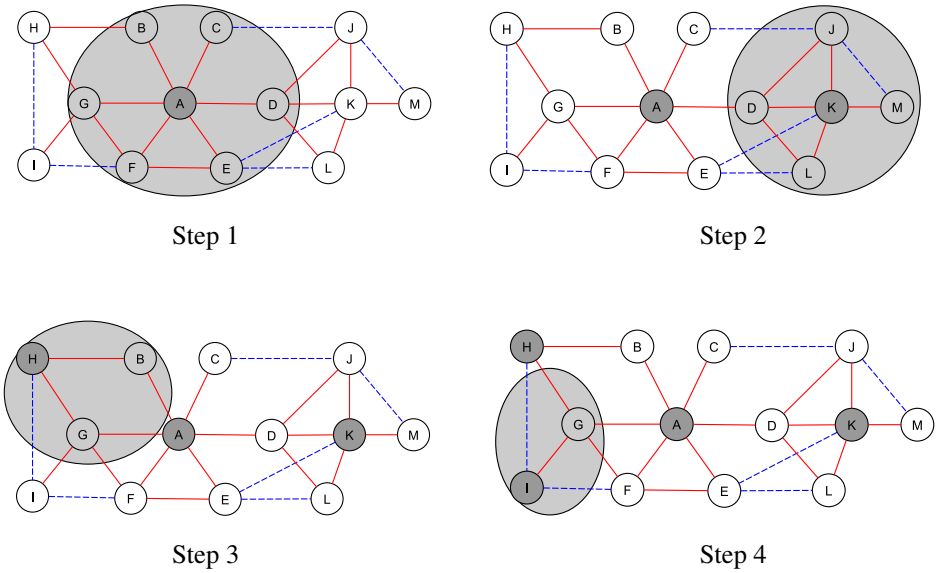


Figure 2.25. The execution of the algorithm

Algorithm 3 Selecting representatives

```
1: function SELECT REPRESENTATIVES(D)
2:     REP ← ∅
3:     for each p ∈ 𝒟 do
4:         calculate the rank of point p
5:     end for
6:     C ← ∅
7:     while C ≠ D do
8:         max ← −inf
9:         max_p ← None
10:        for each p ∈ (D \ C) do
11:            if rank of point p > max then
12:                max ← rank of point p
13:                max_p ← p
14:            end if
15:        end for
16:        for each p ∈ (D \ C) do
17:            if max_p covers p then
18:                C ← C ∪ {p}
19:            end if
20:        end for
21:        REP ← REP ∪ {max_p}
22:    end while
23:    return REP
24: end function
```

Chapter 3

Approximation Pairs Based on Representatives [1]

As mentioned before, samples and representatives play a very important role in data mining. In section 2.5.2 some possible ways were shown to select representatives for a group of objects. The importance of the representatives lies in reducing the execution time of the algorithms. So a natural question can be: can the representatives be used in the set-approximation process?

In the next two sections, two new approaches are introduced which try to answer the previous question.

1. Approximation Pairs Based on Similarity Based Rough Sets
2. Set-Based Approximation Pairs

3.1 Approximation Pairs Based on Similarity Based Rough Sets [2]

Similarity based rough sets is an approximation space which is based on the partition generated by correlation clustering. The lower approximation of a set S is the union of those base sets that are subsets of S. To get these base sets, every point in each base set must be considered. It can be a time-consuming task if the number of points is high. The effectiveness of the representatives lies in situations when the number of objects is very large. It can be practical to use the power of representatives in the approximation process. For each base set, let us consider only its representatives. Let $B \in \mathfrak{B}$ be a base set, and $REP(B)$ be the set of its representatives. The approximation functions are defined as the following:

- $\mathsf{l}_\mathsf{r}(S) = \bigcup\{B \mid B \in \mathfrak{B} \text{ and } \forall x \in REP(B) : x \in S\}$;
- $\mathsf{u}_\mathsf{r}(S) = \bigcup\{B \mid B \in \mathfrak{B} \text{ and } \exists x \in REP(B) : x \in S\}$.

[1] The work described in this chapter was based on [35, 6]

[2] The work described in this section was based on [35]

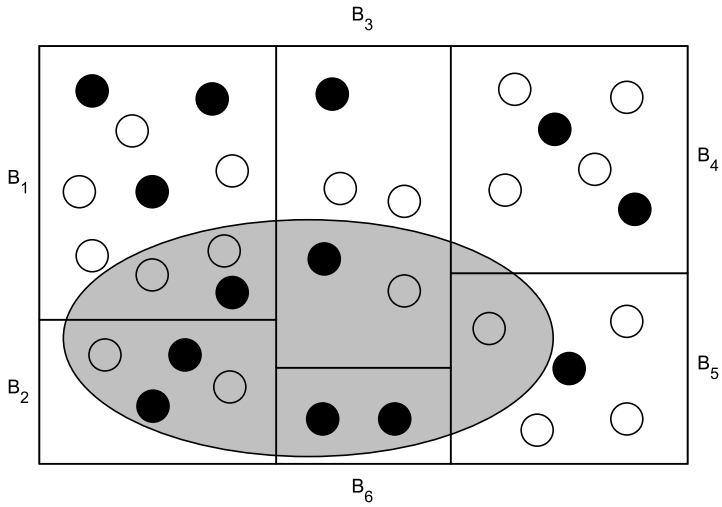


Figure 3.1. Approximation based on representatives

This way, the lower approximation of a set S becomes the union of those base sets for which every representative is a member of S. A base set belongs to the upper approximation if at least one of its representatives is in the set S. Naturally, the certainty of the lower approximation might be lost, but as the number of points is increasing a lot of resources can be saved. This a very important feature for which it can be worth giving up the certainty property.

In Fig. 3.1 a simple example is provided for the method. The base sets are denoted by solid-line rectangles, and the set to be approximated (S) is denoted by a grey ellipse. For each base set, the black circles symbolise the representatives.

The approximation of the set S is the following based on the representatives:

- $l_r(S) = B_2 \cup B_6$
- $u_r(S) = B_1 \cup B_2 \cup B_3 \cup B_6$

The approximation of the set S is the following based on the classical approximation pair:

- $l(S) = B_2 \cup B_6$
- $u(S) = B_1 \cup B_2 \cup B_3 \cup B_5 \cup B_6$

The lower approximation is the same in both cases which is, of course, not necessary. The upper approximation differs in one base set (B_5). When there is a huge number of points and there are several sets to be approximated, approximation using representatives is recommended. In this case, the method can reduce the run-time of the approximation significantly. Determining the approximation with the classical functions 32 objects needed

to be considered. Using the proposed method, only 13 of them had to be tested, so almost 60% of the original points were left-out. Of course, with 32 to 13 points is not a significant change, but in the case of millions of objects, it can be very useful. Working with only the representatives, time and resources can always be saved, because it is sure that the number of representatives is less than that of U. Proving this is very straightforward. Naturally, there cannot be more representatives than objects in the universe. Their numbers cannot be equal either because it could only happen if every object were a representative which implies that every cluster was singleton. Using this system is pointless because the system of base sets is empty (every singleton cluster is discarded).

3.1.1 Set Based Approximation Pairs [3]

In this subsection, two other new possible approximation pairs are proposed based on the representatives. Let $REP(S)$ denote the set of representatives of any arbitrary set S.

The two proposed approximation pairs can be given as $\langle \mathsf{l}, \mathsf{u}_1 \rangle$ and $\langle \mathsf{l}, \mathsf{u}_2 \rangle$, where

$$\mathsf{l}(S) = \bigcup_{\substack{x \in REP(S) \\ [x] \subseteq S}} \{[x]\}$$

$$\mathsf{u}_1(S) = \bigcup_{x \in REP(S)} \{[x]\}$$

$$\mathsf{u}_2(S) = \bigcup_{\substack{x \in REP(S) \\ \|[x] \cap S\| > \|[x] \setminus S\|}} \{[x]\}$$

In this space, a base set contains objects that are similar to a given representative, so each of them generates a base set. The representatives depend on the set to be approximated and so does the system of base sets. As a result, the base sets change as the set changes (they are relative to the set).

Fig. 3.2 shows how the lower and upper approximation of a set S looks like based on $\langle \mathsf{l}, \mathsf{u}_1 \rangle$ and $\langle \mathsf{l}, \mathsf{u}_2 \rangle$. In the middle, the lower approximation can be seen. On the left side, the upper approximation based on u_1 is shown and the right figure illustrates the upper approximation based on u_2 500 points were generated randomly in the unit square. The set to be approximated forms a triangle and its points are marked with larger squares. The base of the tolerance relation (based on similarity) was the Manhattan distance of these objects (d) this time. A similarity ($SIMM$) and a dissimilarity threshold ($DIFF$) were defined. $SIMM$ was set to 0.1 and $DIFF$ was set to 0.2. The objects that are similar to an object x are denoted by circles in Fig. 3.2. The colours of the circles refer to the process of approximation. The environments of the first representatives (objects that are similar to it) are represented by darker circles while the environments of the later representatives are represented by brighter ones.

The topmost circle—in the left picture—contains 4 objects from the set S, but 3 of them are similar to a former representative. The remaining one becomes a representative. However, this object is very similar to many non-set objects, so this circle is not included in the right picture, as this object is not a representative in this case.

[3] The work described in this section was based on [6]

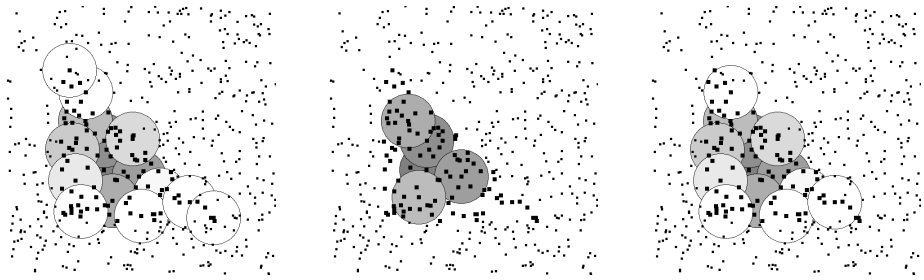


Figure 3.2. Three different set approximations of a triangle

3.1.2 Properties of approximation

Proposition 3.1. *None of the aforementioned approximation pairs are monotone.*

Proof. The first counterexample is presented in Fig. 3.4. In the figures, the solid/dashed edge between nodes x and y denotes that the relation $x\mathcal{R}y$ holds/does not hold, respectively. If there is no edge between some nodes, then the given relation is partial [32, 47], and the two nodes that aren't connected are neither similar nor dissimilar, i.e. they are incomparable or not have been compared yet. The ranks of the objects are for the two given sets:

S**:** $\langle 0.126, 0.626, 0.626, 0.126, -1.000, 0.547, 0.547\rangle$,

S'**:** $\langle 0.126, 0.626, 0.626, 0.126, 1.000, 0.547, 0.547\rangle$.

In the case of the set S $r_1 = r_4 < r_2 = r_3$, henceforth one possible $REP(S)$ can be $\{2, 4\}$, where $[2] = \{1, 2, 3, 5\}$ and $[4] = \{3, 4, 5, 7\}$. Object 4 is similar to 5 and 7, and 2 is similar to 5, hence $\mathsf{l}(S) = \varnothing$.

In the case of the set S' $r_1 = r_4 < r_2 = r_3 < r_5$, and object 5 which covers every member of S', so object 5 becomes the only representative. Hence $\mathsf{l}(S') = \{1, 2, 3, 4, 5\}$, thus $\mathsf{l}(S) \subseteq \mathsf{l}(S')$ holds.

However, although $S \subseteq S'$ $\mathsf{u}_1(S) \not\subseteq \mathsf{u}_1(S')$, where $\mathsf{u}_1(S) = \{1, 2, 3, 4, 5, 7\}$ and $\mathsf{u}_1(S') = \{1, 2, 3, 4, 5\}$. This proves that the approximation pair $\langle \mathsf{l}, \mathsf{u}_1\rangle$ is not monotone.

To disprove the monotonicity of function u_2 a different counterexample is needed. Fig. 3.3 show this tolerance relation and the sets S and S'. The ranks of the objects in this case are:

S**:** $\langle 0.585, 0.759, 0.698, 0.668, -1.000, -0.009, 0.318\rangle$,

S'**:** $\langle 0.585, 0.759, 0.698, 0.668, 1.000, -0.009, 0.318\rangle$.

For ranks in S $r_2 > r_3 > r_4 > r_1$, so at first the sizes of sets $[2] \cap S = \{1, 2, 3\}$ and $[2] \setminus S = \{5, 6\}$ need to be checked . The first of the two sets is bigger, so $2 \in REP(S)$. The set $S \setminus [2]$ only contains object 4, so the sizes of the sets $[4] \cap S = \{3, 4\}$ and $[4] \setminus S = \{5\}$ need to be checked. Again, the first is the bigger set, so $4 \in REP(S)$. Therefore $\mathsf{u}_2(S) = [2] \cup [4] = \{1, 2, 3, 4, 5, 6\}$. Summarising this: $S \subseteq S'$ but $\mathsf{u}_2(S) \not\subseteq \mathsf{u}_2(S')$. Hence, the approximation pair $\langle \mathsf{l}, \mathsf{u}_2\rangle$ is not monotone, too. ■

Proposition 3.2. *The weak approximation property holds for both approximation pairs.*

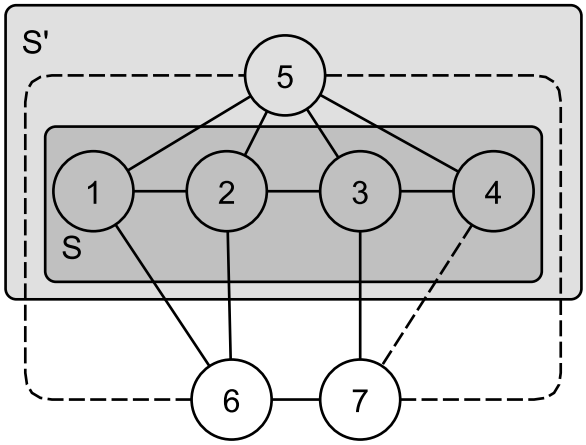


Figure 3.3. Monotonicity does not hold for u_2.

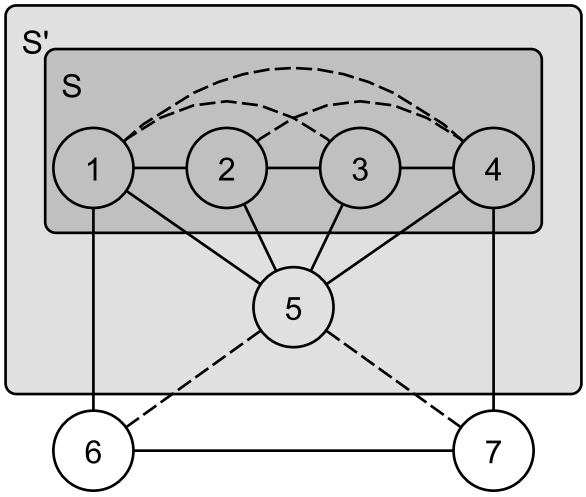


Figure 3.4. Monotonicity does not hold for u_1.

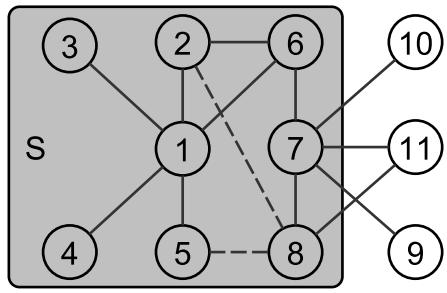


Figure 3.5. Strong approximation property does not hold for the second approximation pair

Proof.

In both cases, the functions use the same representatives based on the same order of ranks. Therefore the lower approximation cannot be a larger set than the upper approximation. ■

Proposition 3.3. *For* $\langle \mathsf{l}, \mathsf{u}_1 \rangle$ *the strong approximation property holds.*

Proof. By definition $\mathsf{l}(S) \subseteq S$ is always true. In the case of u_1, every member of the set S is covered by at least one of the representatives. Thus $S \subseteq \mathsf{u}_1(S)$ is also true. ■

Proposition 3.4. *For* $\langle \mathsf{l}, \mathsf{u}_2 \rangle$ *the strong approximation property does not hold.*

Proof. In Fig. 3.5 an example can be seen that contradicts this property. The ranks of the objects are the following:

$$r = \langle 1.00, 0.85, 0.36, 0.36, 0.65, 0.55, -0.32, -0.80, -0.11, -0.11, -0.40 \rangle.$$

Hence the possible representatives of the set S are the objects 1 and 7, where $[1] = \{1, 2, 3, 4, 5, 6\}$ and $[7] = \{6, 7, 8, 9, 10, 11\}$. The upper approximation of the set S is $\mathsf{u}_2(S) = \{1, 2, 3, 4, 5, 6\}$, because $[7] \cap S = \{6, 7, 8\}$ and $[7] \setminus S = \{9, 10, 11\}$, therefore object 7 cannot be a representative of S. So $S \not\subseteq \mathsf{u}_2(S)$, meaning that the strong approximation is not necessarily true.

■

Proposition 3.5. *The normality of the lower and upper approximation holds.*

Proof. The representatives are selected from the members of the set. If $S = \varnothing$ then there are no representatives, so $\mathsf{l}(S) = \mathsf{u}_1(S) = \mathsf{u}_2(S) = \varnothing$. So the normality of the lower approximation holds because an empty set does not have any representative members. The same holds for the normality of the upper approximation. ■

Chapter 4

A Novel Area of Application – Graph Approximation on Similarity Based Rough Sets

Rough set theory is a possible way to handle uncertainty using set-approximations. Similarity based rough sets is a new way for the same problem and it is based on the similarity of objects. A tolerance relation (based on similarity) can be represented by a signed graph. This graph is complete if the relation is total, and it is not complete if the relation is partial. Because of the symmetry, it must be an undirected graph. Due to the reflexivity, every vertex in the graph has a self-loop edge. If two objects are similar, then a positive edge runs between them, and if they are different, then the edge is negative. Every graph can be represented by a set that contains ordered pairs. In this case, it can be represented by a set of 3-tuples. If a similarity graph can be treated as a set, then a natural question arises: can this graph be also approximated? In this chapter, a possible way is shown to define the lower and upper approximation of a graph and also how this method can be applied in data pre-processing.

The graph in Fig. 4.1 can be represented by the following set:

$$\left\{\begin{array}{lllll} \langle A,A,+\rangle, & \langle B,B,+\rangle, & \langle C,C,+\rangle, & \langle D,D,+\rangle, & \langle A,B,+\rangle, \quad \langle A,C,+\rangle, \\ \langle A,D,-\rangle, & \langle B,A,+\rangle, & \langle B,D,-\rangle, & \langle B,E,-\rangle, & \langle C,A,+\rangle, \quad \langle C,D,+\rangle, \\ \langle D,A,-\rangle, & \langle D,B,-\rangle, & \langle D,C,+\rangle, & \langle E,B,-\rangle & \end{array}\right\}$$

The main goal of this part of the dissertation is to extend the principles of similarity based rough sets to graphs that represent tolerance relation (based on similarity). As mentioned before, from the theoretical point of view, a Pawlakian approximation space can be characterized by an ordered pair $\langle U, \mathcal{R}\rangle$, where U denotes the universe (a nonempty set of objects) and $\mathcal{R}$ denotes an equivalence relation based on indiscernibility. In the similarity based rough sets, $\mathcal{R}$ is a tolerance relation (based on similarity). In the case of graph approximation, the universe is a complete undirected signed graph in which every node is connected to every node by 2 edges (one positive and one negative) and every node has also 2 self-loop edges. Formally $G = U \times U \times \{+,-\}$.

Let $G_1 \subseteq G$ ($G_1 \neq G$) be a graph representing an arbitrary tolerance relation (based on

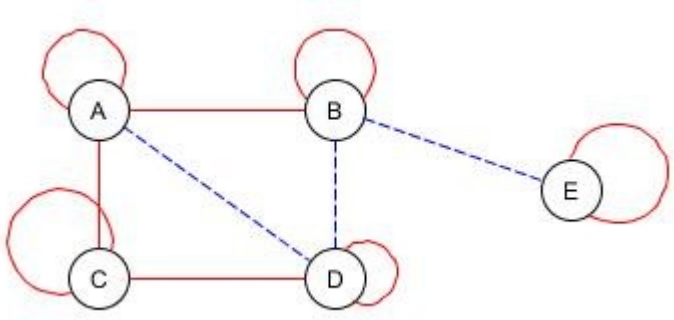


Figure 4.1. A tolerance relation (based on similarity) represented by a signed graph

similarity). This graph defines the background knowledge. The relation $\mathcal{R}$ is also defined by G_1. For all objects $x, y \in U$ if $(x, y, +)$, then the objects are similar and if $(x, y, -)$,then they are different. In rough set theory (and also in its similarity based version), the base sets provide the background knowledge. Here, there are base graphs, and they represent the same. The system of base graphs is also determined by the correlation clustering, and it can be given by the following formula:

Definition 4.1.

$$\mathfrak{B}_G = \{g \mid g \subseteq G_1, \textit{ and } \langle x, y, s\rangle \textit{ if } p(x) = p(y) \textit{and } s \in \{+, -\}\},$$

where p is the partition gained from the correlation clustering.

Definition 4.2.
Let $G_2 \subseteq G_1$ be an arbitrary subgraph of G_1. The lower and upper approximation of G_2 can be given by the following way:

$$\mathsf{l}(G_2) = \bigcup \{g \mid g \in \mathfrak{B}_G \textit{ and } g \subseteq G_2\}$$

$$\mathsf{u}(G_2) = \bigcup \{g \mid g \in \mathfrak{B}_G \textit{ and } g \cap G_2 \neq \varnothing\}$$

The lower approximation is the disjoint union of those base graphs that are subgraphs of G_2. The upper approximation is the disjoint union of those base graphs for which there exists a graph which is a subgraph of both G_1 and G_2.

Definition 4.3. *The accuracy of the approximation can be calculated by the following fraction, where $N(G)$ denotes the number of self-loop edges in an arbitrary graph G:*

$$\alpha_{G_2} = \frac{|\mathsf{l}(G_2)/2| - N(\mathsf{l}(G_2))}{|\mathsf{u}(G_2)/2| - N(\mathsf{u}(G_2))}$$

Graph approximation uses the same concepts as the set approximation. However, it is a stricter method, as it takes into consideration not only the objects but the edges too.

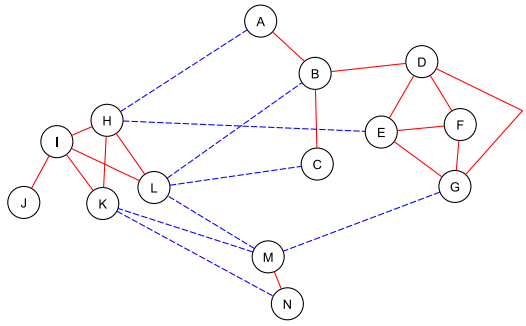


Figure 4.2. G_1 graph representing a tolerance relation (based on similarity) based on a set of attributes

4.1 Attribute Reduction with Graph Approximation

The three main steps of data mining are pre-processing, knowledge discovery and post-processing. Data pre-processing is a very important step in the data mining process which involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, noisy, and is likely to contain many errors. Data pre-processing is a method of resolving such issues. One of the main issues with raw data is that there can be too many attributes. So a natural question can be if some of these attributes can be removed from the system preserving its basic properties (whether some attributes can be considered as superfluous). In this subsection, a method is proposed to measure the dependency between two attributes (or set of attributes). If this dependency value is above a threshold, then one of the attributes can be removed. Let $IS = (U, A)$ an information system and $A', A'' \subseteq A$ two sets of attributes. Let G_1 be the graph representing the tolerance relation (based on similarity), which is based on the attribute set A'. Let G_2 be the graph representing the tolerance relation (based on similarity) which is based on the attribute set A''. To measure the dependency between A' and A'' the following method is proposed:

1. Determine the system of base graphs based on G_1;
2. Approximate G_2using the base graphs defined in the first step;
3. Calculate the accuracy of approximation;
4. If the accuracy is higher than a threshold, then A'' can be treated as superfluous.

In the following figures, a very simple example is shown with 14 objects. In Fig. 4.2 a graph can be seen, which denotes a similarity based on a set of attributes. In the figure, the solid lines denote the similarity, while the dashed ones denote the difference between objects. In Fig. 4.3 the base graphs can be observed which were generated by the correlation clustering. In this example, there are three base graphs.

Fig. 4.4 shows another graph, which also illustrates another set of attributes. It is important that the similarity is based on the same objects as before. The difference between G_1 and

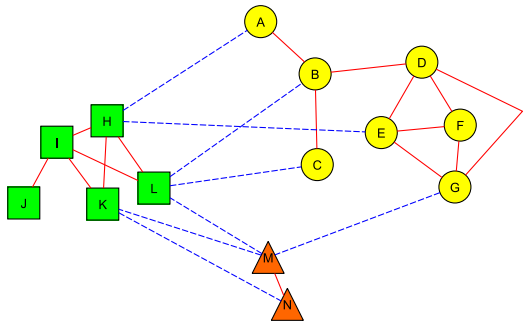


Figure 4.3. The base graphs generated by the correlation clustering

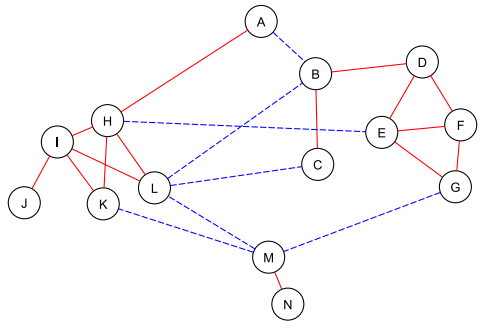


Figure 4.4. G_2 graph representing a tolerance relation (based on similarity) based on a set of attributes with the same objects

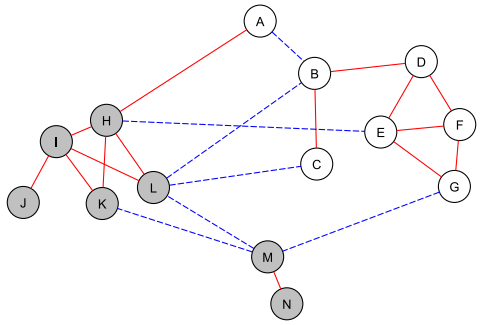


Figure 4.5. The lower approximation of G_2

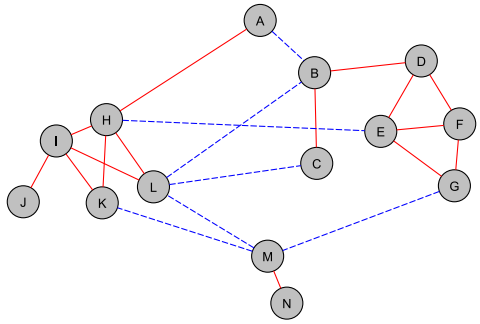


Figure 4.6. The upper approximation of G_2

G_2 is negligible. It can be observed only in 4 edges. Between objects A and H, the negative edge was replaced by a positive one. Between objects K and N, the edge was deleted as well as between D and G. Between A and B the positive edge was changed to negative.

In Fig. 4.5 the lower approximation and in Fig. 4.6 the upper approximation can be seen. It is interesting that even though there are only slight differences between the two graphs the lower approximation contains only the two smallest base graphs. The reason is that graph approximation is stricter because it takes into consideration the edges. As objects A and B became different, the base graph containing them cannot be in the lower approximation, only in the upper approximation.

The accuracy of the approximation is: $\alpha_{G_2} = \frac{14}{29} = 0.48$

This means that based on G_1 the percentage of the available information about G_2 is only 48% even though they are almost equivalent. This method takes into account the real similarity among objects; therefore it can give appropriate results in situations, where other algorithms proved to be a dead end. For example, in mathematical statistics a common method to measure the dependency between attributes is correlation. Although, it works only if there is a linear relationship between the attributes. Our proposed method can work in various applications for any type of relationship.

Chapter 5

Summary

The main result of my research was developing a completely new approximation space which was based on a tolerance relation (based on similarity) . In this space, the system of base sets is generated by the correlation clustering. Correlation is a clustering technique that is based on a tolerance relation. The proposed space has many good qualities and it is different from the covering spaces induced by a tolerance relation (based on similarity). The main difference between our space and these covering spaces is that ours considers the real similarity among the objects while the covering spaces generated by a tolerance relation only consider similarity to a distinguished member. An algorithm was successfully developed, based on physics, by which tolerance relations (based on similarity) can easily be visualized. In data mining, to reduce the execution time of an algorithm, it is common to use samples. A sample contains points from the original data set. There are numerous ways to choose a part of the input data set which can be treated as a sample. However, in every method, it is crucial that the chosen objects must represent the entire population. In this case, representativeness means that the specific properties are as similar in the sample as in the entire set. Without this property, important information might be disregarded. In this dissertation, a method was provided to generate the representatives of a given set. One of the good properties of this algorithm is that it chooses the necessary number (not too small or big) of representatives. The importance of the representatives lies in reducing the execution time of the algorithms. In our research, it was applied to set approximation. The lower approximation of a set is the union of those base sets that are subsets of the given set. In order to get these base sets, every object in each base set must be considered. It can be a time-consuming task if the number of points is high. In this situation, it could be helpful to consider only the representatives for each base set. New approximation pairs have also been proposed based on the representatives. Feature selection is an important part of machine learning. Feature selection refers to the process of reducing the number of attributes in a dataset, or of finding the most meaningful attributes. This process improves the quality of the model and it also makes the model more efficient. In the last section of the dissertation, our graph-approximation method is described. It is based on the fact that any tolerance relation (based on similarity) can be represented by a signed graph and any graph can be represented by a set. Therefore the proposed method (similarity based rough sets) can be used for graphs as well. In the last part of this work, a possible way is shown to use graph approximation in feature selection.

6. fejezet

Összefoglalás

Napjainkban az adatok mennyisége robbanásszerűen növekszik. A keletkező adatok azonban gyakran hiányosak esetleg inkonzisztensek. A hiányosságnak számos oka lehet. Például nem ismert az adott érték vagy éppen nem értelmezhető. Inkonzisztenciáról akkor beszélünk, ha az adatok közt valamilyen ellentmondás mutatkozik. Ezen problémák számos nemkívánatos eseményt idézhetnek elő (rossz előrejelzés, nem megfelelő döntéshozatal stb.). Az informatikai tudományokban az ilyen jellegű pontatlanságok elkerülésére számos módszert alkottak.

Az életlen halmazok elmélete egy viszonylag új elmélet, melynek alapjait Pawlak professzor a 80-as években fektette le [42, 43, 45]. A pawlaki terek az adatokban rejlő bizonytalanságot egy megkülönböztethetetlenségen alapuló reláció által teszik kezelhetővé. A rendelkezésre álló információk alapján számos esetben előfordul, hogy két objektumot nem tudunk megkülönböztetni egymástól. Két objektum megkülönböztethetetlennek tekinthető, ha minden figyelembe vett, releváns tulajdonságuk megegyezik. Ezen megkülönböztethetetlenségi reláció ekvivalencia reláció, mely a háttértudásunkat, illetve annak korlátozottságát reprezentálja. A pawlaki terek e relációt veszik alapul a halmazközelítéseknél. A megkülönböztethetetlenség hatással van az eleme reláció használatára, hiszen bizonyos esetekben bizonytalanná teszi a reláció megítélését, életlenné teszi a halmazt azáltal, hogy egy adott objektumra vonatkozó döntés kihat a tőle megkülönböztethetetlen objektumokra vonatkozó hasonló döntésekre. Tulajdonképpen kénytelenek vagyunk bizonyos objektumokat ugyanúgy kezelni. Ezt a bizonytalanságot halmaz-approximációs eszközökkel reprezentáljuk. Ha nagy méretű információs rendszerekből szeretnénk minél több hasznosítható információt kinyerni, akkor elkerülhetetlen a megkülönböztethetetlenség kezelése. Az életlen halmazok elméletében arra szeretnénk választ kapni, hogy hogyan jellemezhetőek bizonyos halmazok, illetve, hogy egyes tulajdonságok által generált halmazba beletartozik-e egy bizonyos objektum vagy sem.

Az elmúlt 40 évben a pawlaki tereknek több általánosítása is napvilágot látott. Egyes esetekben a gyakran szigorúnak tekinthető ekvivalencia relációt egy hasonlóságot reprezentáló tolerancia relációval helyettesítik [25, 47, 44]. Léteznek ezen kívül a valószínűségszámításra alapuló modellek is [27, 46, 48, 51, 52]. Végül, de nem utolsósorban meg kell említeni a fuzzy halmazelmélettel való hibridizált rendszereket is [54, 19, 20].

Az adatbányászat az adatok hihetetlen mértékű növekedése miatt az informatikának egy rendkívül fontos és folyamatosan fejlődő ágává vált. Az adatbányászat egy olyan technika,

mellyel hasznos információ nyerhető ki automatikus módon nagy mennyiségű adatokban. Célja az adattárak átkutatása annak érdekében, hogy olyan új és hasznos mintázatokat találjanak, amelyek egyébként ismeretlenek maradnának. Az élet számos területén alkalmazhatóak az adatbányászati módszerek. Olyan kérdésekre kaphatunk választ, mint például: igaz-e, hogy azok a vásárlók, akik pelenkát vásárolnak, azok sört is fognak. Természetesen nem minden információ feltárási feladat tekinthető adatbányászatnak. Például egyes rekordok kikeresése egy adatbázis-kezelő rendszer segítségével, vagy bizonyos weblapok megtalálása egy internetes keresőprogram segítségével, az információkeresés (information retrieval) területével vannak kapcsolatban. Az adatbányászatnak 3 fő lépését szokás megkülönböztetni: előfeldolgozás (pre-processing), adatbányászat, utófeldolgozás (post-processing). Az előfeldolgozás célja, hogy a nyers input adatokat megfelelő formátumba alakítsa. Lépései tartalmazzák az adatok több forrásból való egyesítését, az adatok tisztítását a zaj és a redundáns megfigyelések eltávolításával, és azon rekordok és változók kiválasztását, amelyek lényegesek az aktuális feladatnál. Ezután az adott adatbányászati algoritmus megkapja az előfeldolgozott adatokat bemenetként. Végeredményül egy mintázatot, modellt, "tudást" kapunk. Gyakran azonban ezek a mintázatok nem értelmezhetőek vagy önmagukban haszontalanok. Ezért van szükség az ún. utófeldolgozásra, mely különböző vizualizációs és kiértékelő techinkák segítségével segít a döntéshozatalban.

Az életlen halmazok elmélete kulcsfontosságú lehet az adattudományokban [10], hiszen nagy mennyiségű adatok esetén a bizonytalanság megfelelő kezelése elengedhetetlen. Az adatok előfeldolgozása esetén számos esetben alkalmaznak életlen halmazokra épülő módszereket.

Diszkretizációnak nevezzük azt a folyamatot mikor egy folytonos értékű attribútomot nominális attribútumra képzünk le az eredeti tartomány intervallumokra történő vágásával. Minden ilyen intervallumot egy-egy nominális értéknek tekinthetünk. Életlen halmazokon alapuló diszkretizációs technikák lásd például [31, 41].

Az ún. jellemzők szelekciója az előfeldolgozás azon folyamata mikor lényegtelen jellemzőket (attribútumokat) eltávolítunk a rendszerből. Az attribítumok növekedésével nő az algoritmusok futási ideje és erőforrásigénye. Az életlen halmazokon alapuló jellemző-szelekciós módszerek az ún. reductok fogalmán alapszik. Egy reduct attribútumoknak egy olyan részhalmaza, mely ugyanazokat az alaphalmazokat eredményezi, mint amelyket az összes attribútum használata esetén kapnánk [49, 50, 30].

Egy további fontos előfeldolgozási technika az ún. példány-szelekció. Az adatbányászatban az ún. felügyelt tanítási módszerek (pl. osztályozás) a bemeneti adathalmazt két részre osztják: tanuló és teszt adathalmaz. A tanuló adathalmaz segítségével taníthatóak be a modellek, míg a teszt adathalmaz segítségével lehet őket kiértékelni. Példány-szelekció esetén a tanulóhalmaz méretét kívánják csökkenti úgy, hogy a hatékonyság ne csökkenjen. Így egy új tanulóhalmazt kapunk, mellyel növelhető a modell hatékonysága valamint csökkenthető az adathalmaz mérete is [12].

Természetesen nem csak az adatok előfeldolgozásánál érdemes az életlen halmazok elméletét segítségül hívni. Számos adatbányászati területen is alkalmazható, mint például: szabály alapú osztályozás [26, 28], társítási (asszociációs) szabályok generálása [29, 17], klaszterezés [18] stb.

A pawlaki tér az objektumok megkülönböztethetetlenségét reprezentáló ekvivalencia reláción alapul. A tér általánosításaként megjelentek olyan approximációs terek, melyek nem egy ekvivalencia reláción, hanem egy hasonlóságot reprezentáló tolerancia reláción alapul-

nak. Ezek a terek megőrzik a pawlaki tér azon tulajdonságát, hogy az approximációban résztvevő alaphalmazok uniója kiadja a tér univerzumát. Feladják viszont azt a követelményt, hogy az alaphalmazok páronként diszjunktak legyenek. Az alaphalmazok generálása itt úgy történik, hogy minden objektumhoz képezzük a vele tolerancia relációban álló objektumok halmazát. Ez azt jelenti, hogy egy adott objektumhoz való hasonlóságot vesszünk figyelembe. Azzal a gyakorlatban is előforduló nehézséggel szembesülhetünk, hogy az alaphalmazok száma szélsőséges esetben megegyezhet az objektumok számával. Ez a közelítések meghatározása esetén jelentősen növeli a számításigényt és nagy adatbázisoknál korlátozza az ilyen terek hatékony használatát. Ennek a problémának egy lehetséges megoldására dolgoztunk ki a hasonlóságon alapuló életlen halmazok rendszerét, ahol a korrelációs klaszterezés segítségével az alaphalmazok képzése a hasonlóságon magán és nem az egy objektumhoz való hasonlóságon alapszik. Az így létrejövő tér egyrészt jól reprezentálja az értelmezett hasonlóságot, másrészt az alaphalmazok száma kezelhető méretűre csökken. Ennek a térnek a tulajdonságaival és az alkalmazhatóságával foglalkozik a dolgozat, bemutatva mindazokat az előnyöket, amit a korrelációs klaszterezés alapján nyerhetünk.

A dolgozat struktúrája a következő. Az 1. fejezetben az életlen halmazok elméletének alapjai valamint a fontosabb approximációs terek kerülnek lefektetésre. A második fejezetben a hasonlóság alapú életlen halmazok approximációs tér [36] kerül bemutatásra. Ebben a fejezetben ezenkívül néhány eszköz és fejlesztés is bemutatásra kerül [37, 35, 6, 5, 7]. A hasonlóság alapú életlen halmazok gráfokra is általánosíthatóak, melyet jellemzők szelekciójára is lehet alkalmazni. Ezzel a módszerrel a dolgozat a 4. fejezetben foglalkozik.

6.1. Életlen halmazok elmélete

A gyakorlatban egy halmazt objektumok kollekciójaként értelmezzük, melyet elemeivel lehet azonosítani. Ha egy objektumról szeretnénk eldönteni, hogy benne van-e egy adott halmazban vagy sem, akkor a kérdésre igennel vagy nemmel lehet válaszolni. Jó példa lehet a hárommal osztható számok halmaza, hiszen egy számról könnyen eldönthető, hogy osztható-e 3-mal vagy sem. Természetesen ehhez szükség van a maradékos osztás műveletének ismeretére. Ezt a fajta ismeret egy háttértudásként tekinthető, mellyel eldönthető egy számról, hogy benne lesz-e a halmazban vagy sem. Nem mindenki tudja azonban megfelelően használni a maradékos osztás műveletét bármely számra. Néhány alsótagozatos diák kizárólag a 100-as számkörben képes műveleteket végezni. Ők nem tudnánk eldönteni, hogy a 142 osztható-e 3-mal. Számukra ez a szám lehet, hogy osztható is 3-mal és lehet, hogy nem osztható. Tehát a háttértudásuk miatt egyfajta bizonytalanság lép fel. Az életlen halmazok elméletének matematikai alapjait Pawlak professzor fektette 1982-ben [42]. Az elmélet a háttértudásból származó bizonytalanságot (homályosságot) kívánja kezelni. Az informatika mindennapjaiban, a különböző informatikai alkalmazásokban az objektumok adatbázisokban adottak számunkra. Van egy egyedi azonosítójuk, és bizonyos attribútum-értékek hozzájuk vannak rendelve. Az egyedi azonosító egy technikai eszköz, nem mond semmit magáról az objektumról: két objektum a szó valódi értelmében csak akkor különböztethető meg egymástól, ha legalább egy attribútum-értékük különbözik. Ez a szemlélet jól tükrözi azt a mindennapi gyakorlatot, amelyben két objektumot akkor tudunk egymástól megkülönböztetni, ha legalább egy (ismert) tulajdonságuk különbözik. Ha egy tetszőleges halmaz esetén a rendelkezésre álló információk alapján el szeretnénk dönteni, hogy egy objektum eleme-e egy hal-

maznak, akkor döntésünk kihat az adott objektumtól megkülönböztethetetlen objektumokra is:

- bizonyosak lehetünk abban, hogy egy objektum eleme egy halmaznak, ha minden, tőle megkülönböztethetetlen objektum is eleme a halmaznak;
- lehetséges, hogy egy objektum eleme egy halmaznak, ha van olyan tőle megkülönböztethetetlen objektum, amely eleme a halmaznak;
- bizonyosak lehetünk abban, hogy egy objektum nem eleme egy halmaznak, ha egyik tőle megkülönböztethetetlen objektum sem eleme a halmaznak.

Összefoglalásképpen azt mondhatjuk, hogy a rendelkezésünkre álló információk alapján fellépő megkülönböztethetetlenség „elkeni" a halmazok határát, az egyébként éles határral rendelkező halmazokat „életlenné" teszi.

Egy általános approximációs tér a következő módon definiálható:

6.1. Definíció. *Egy általános approximációs tér egy rendezett 5-ös* $\langle U, \mathfrak{B}, \mathfrak{D}, \mathsf{l}, \mathsf{u}\rangle$ *ahol:*

- $U \neq \varnothing$ *objektumoknak egy halmaza (univerzum)*
- $\mathfrak{B}$ *az alaphalmazok halmaza, melyekre a következőek teljesülnek:*
  - $\mathfrak{B} \neq \varnothing$
  - *ha* $B \in \mathfrak{B}$*, akkor* $B \subseteq U$ *és* $B \neq \varnothing$
- $\mathfrak{D}$ *a definiálható halmazokat tartalmazza, mely egy induktív definícióval adható meg, melynek bázisa* $\{\varnothing\} \cup \mathfrak{B}$
- $\mathsf{l}, \mathsf{u} : 2^U \to \mathfrak{D}$ *egy approximációs párt alkot.*

Informálisan az U halmaz az approximáció univerzuma, azaz az U részhalmazait közelítjük. A $\mathfrak{B}$ nem üres halmazoknak egy családja: elemeit alaphalmazoknak nevezzük, és azt a háttértudást reprezentálja, amelyre támaszkodva a halmazok közelítését elvégezzük. A $\mathfrak{D}$ halmazcsalád a definiálható halmazokat tartalmazza, rögzíti, hogy milyen módon használhatjuk fel az alaphalmazokat a közelítés során. Ennek alapján az alaphalmazok a közelítés eszközeinek tekinthetők. Az l, u függvények megadják egy tetszőleges halmaz alsó illetve felső közelítését. Az alaphalmazok családjának megválasztásával, az alaphalmazok felhasználásának a megadásával (azaz a definiálható halmazok családjának értelmezésével), valamint az approximációs párra vonatkozó követelmények specifikálásával az általános approximációs tér számos különböző típusát lehet megadni.

Pawlak a megkülönböztethetetlenséget egy ekvivalencia-relációval reprezentálta. A klasszikus pawlaki terekben az alaphalmazok az ekvivalencia osztályok, tehát egy alaphalmaz az egymástól megkülönböztethetetlen objektumokat tartalmazza. Az approximációs pár pedig a következőképpen definiálható:

- Azon elemek, amelyek bizonyosan elemei egy adott halmaznak, alkotják a halmaz alsó közelítését

- Azon elemek, amelyek lehet, hogy elemei egy adott halmaznak, alkotják a halmaz felső közelítését

Gyakorlati alkalmazások során a megkülönböztethetetlenség túl szigorú lehet. Számos esetben elégséges lehet csupán objektumok hasonlóságáról beszélni. A pawlaki terek általánosításaként olyan terek jelentek meg, melyek egy tolerancia relációra támaszkodnak. Ezekeben a terekben az alaphalmazok azokat az elemeket tartalmazzák, melyek egy bizonyos elemhez hasonlóak. Így tulajdonképpen minden objektum egy-egy alaphalmazt generál.

További általánosított terek is léteznek. Egyes terek esetén az univerzumnak bármely részhalmaza is alaphalmazt alkothat. Egyetlen kritériumnak kell teljesülnie, hogy az alaphalmazok uniójának az univerzummal kell egyenlőnek lennie. Az ún. parciális terekben ez a kritérium is elhagyható. Tehát az univerzumnak bármely nemüres halmaza alkothat alaphalmazt.

6.2. Hasonlóság Alapú Életlen Halmazok

A pawlaki tér általánosításaként megjelenő egyes lefedő (covering) terek néhány esetben egy tolerancia (reflexív, szimmetrikus) relációra támaszkodnak. Ilyen relációkat generálhatunk az objektumaink tulajdonságai alapján. Az így megadott tolerancia reláció két objektum közti hasonlóságot reprezentálja. A tolerancia relációra támaszkodó lefedő terek tipikus alakjában egy objektum a hozzá hasonló objektumokkal alkot egy alaphalmazt (az adott objektumhoz való viszonyuk vonatkozásában megkülönböztethetetlenek egymástól). Ily módon minden elem egy alaphalmazt fog generálni. Ezen alaphalmazrendszer viszont számos problémát generál. Az egyik, hogy nem magát az objektumok közti hasonlóságot veszi alapul, hanem csupán egy adott objektumhoz való hasonlóságot. Másik fontos gyengesége, hogy a kapott alaphalmazok nem páronként diszjunktak. Természetes kérdés, hogy nem lehetne-e a tolerancia reláció alapján az objektumok közötti valódi hasonlóság alapján generálni az alaphalmazokat?

6.2.1. Korrelációs klaszterezés

Az adatbányászat egyik fő eszköze a klaszteranalízis. Itt az objektumainkból olyan csoportokat képzünk, melynél az egy csoportban lévő objektumok hasonlítanak egymáshoz, míg a különböző csoportban lévők különböznek egymástól. Általában a hasonlóság és különbözőség az objektumot leíró adatok számszerű különbségén alapul. Vannak olyan esetek azonban, amikor az objektumaink nem írhatóak le számokkal, mégis dönthetünk azok hasonlóságáról illetve különbségéről. Gondoljunk csak például személyekre, ahol a külső nem feltétlenül írható le számokkal, mégis teszünk állításokat, hogy két személy hasonló-e vagy sem. Természetesen ezek az állítások személyfüggőek, valaki tekintheti ugyanazt a két személyt hasonlónak, míg más különbözőnek.

Ha matematika segítségével szeretnénk a hasonlóságot megragadni, akkor egy tolerancia relációra lesz szükségünk. Ha ez a reláció teljesül két objektum között, akkor azt mondjuk, hogy a két objektum hasonló, míg ha nem teljesül, akkor pedig azt mondjuk, hogy különbözőek. Természetesen egy objektum saját magához mindenképpen hasonlít, így ez a reláció természetszerűen reflexív, és hasonlóan belátható, hogy szimmetrikus is. Viszont ennél tovább már nem mehetünk, például a tranzitivitás nem teljesül. Ha tekintjük az embert és az

egeret, akkor a belső felépítés alapján hasonlóak, ezért is használnak az orvosi kutatásokban egeret. Másrészt az ember és a próbabábú is hasonló a külső alapján, ezért is láthatunk az üzletek kirakataiban felöltöztetett próbabábukat, mellyel az egyes ruhadarabokat kívánják eladni. Viszont semmi hasonlóságot nem lehet felfedezni egy egér és egy próbabábú között.

A tolerancia reláción alapuló klaszterezés Bansal vetette fel (és adott rá egy közelítő megoldást) [8], de Zahn is megfogalmazta ugyanezt a problémát egy más szempontból [55]. A korrelációs klaszterezés célja egy olyan ekvivalencia reláció megtalálása, mely a tolerancia relációhoz a legközelebbi. A módszer megoldása egy költségfüggvény minimalizálását jelenti, mely az ún. negatív eseteket vagy konfliktusokat számolja: Konflikusnak nevezhetőek a következő esetek:

- Két objektum különböző mégis egy klaszterbe kerültek
- Két objektum hasonló, de különböző klaszterbe kerültek

A költségfüggvény értéke tekinthető a két reláció távolságának.

A korrelációs klaszterezés eredménye tehát egy partíció, melynek elemei (klaszter) a hasonló objektumokat tartalmazza. Érdemes lehet tehát az így generált csoportokat a halmazapproximáció alaphalmazaiként kezelni. A generált partíció számos jó tulajdonsággal rendelkezik:

- a hasonlóságon magán alapszik és nem csak egy bizonyos objektumhoz való hasonlóságon;
- az alaphalmazok páronként diszjunktak így az alsó és felső közelítések zártak;
- megfelelő számú alaphalmaz jelenik meg;
- az alaphalmazok számossága nem túl nagy és nem túl kicsi.

A korrelációs klaszterezés NP teljes probléma, hiszen a lehetséges partíciók száma exponenciálisan növekszik. 15 feletti elemszám esetén szinte lehetetlen belátható időn belül megtalálni a megoldást, ha az összes lehetséges partíciót megvizsgáljuk. A gyakorlatban ezért keresőalgoritmusok használata ajnálott. Az alaphalmazrendszer tehát nagyban függhet az alkalmazott keresőalgoritmustól. A 2.3. szakaszban számos algoritmus került összehasonlításra véletlenül generált adatokon. A kapott eredményekből egyértelműen látszik, hogy ezeken az adathalmazokon a szimulált hűtés teljesített legjobban. Az összehasonlítás alapját az alábbi tulajdonságok képezték: egyelemű (szingleton) klaszterek száma, alaphalmazok méretének szórása és terjedelme, valamint az algortimus futási ideje.

6.2.2. Hasonlóság alapú életlen halmazok annotációval

A korrelációs klaszterezés során előfordulhat olyan eset, hogy egy elem önmagában alkotott klasztert, mert a konfliktusok számának növelése nélkül a rendszer nem tudta egy klaszterbe sem besorolni. Az ilyen szingleton klaszterek a hasonlóság szempontjából nem hordoznak információt, ezért e klasztereket nem tekintetjük alaphalmazoknak. Ezáltal egy parciális approximációs tér jön létre.

Előfordulhat azonban olyan eset is, hogy egy elemet azért nem tudott a rendszer egyetlen klaszterbe sem besorolni, mert a rendelkezésre álló információ nem volt elegendő. Tegyük

fel, hogy az objektumaink orvosi felvételek páciensekről. Elképzelhető, hogy egy páciens hasonló betegségben szenved, mint egy csoportnyi másik beteg, de a felvétel zajossága miatt az algoritmus a konfliktusok számának csökkentése érdekében nem sorolta be a megfelelő klaszterbe. Egy orvos vagy szakértő azonban észrevenné a problémát, és tudná, hogy az adott páciens hasonló a többi esethez. Érdekes lehet, ha a felhasználónak lehetősége lenne egyes szingleton klaszterek elemének egy tetszőleges alaphalmazhoz történő besorolására. Ezzel a felhasználó a saját tudását tudná bevinni a rendszerbe, valamint a parcialitás mértéke is csökkenne.

6.2.3. Annotáció támogatása

Az annotáció egy igen hasznos továbbfejlesztése a hasonlóság alapú életlen halmazok terének, hiszen egy lehetséges interakciót szolgáltat a felhasználó és a rendszer között. Természetesen a folyamat az adott felhasználó szakértelmén alapszik, hiszen a rendszer által hozott döntést kell felülbírálnia. Természetesen ez nem azt jelenti, hogy a felhasználónak nem adható valamilyen segítség. A dolgozatban két fő módszer került bemutatásra. Az első egy vizuális techinika, mellyel hasonlóságot reprzentáló tolerancia relációkat lehet könnyedén ábrázolni. Ha két objektum közel van ebben a reprezentációban, akkor az azt jelenti, hogy valószínüleg hasonlóak. Ha egy szingleton klaszter eleme valamely másik objektumhoz közel van, akkor érdemes lehet az elemet tartalmazó klaszterbe beleilleszeni. A másik módszer egy matematikai technika, mely klaszterenként azokat az elemeket kívánja megtalálni, mely az egész csoportot reprezentálja. Időt lehetne spórolni, ha a számítások során csupán ezek a reprezentatív objektumok lennének használva. Előfordulhat, hogy több lehetséges klaszterbe is beilleszthető lenne a szingleton klaszter eleme. Érdemes lehet abba a csoportba betenni az szingleton elemet, melynek reprezentatív eleme a leginkább hasonló a kérdéses objektumhoz.

6.3. Reprezentatív elemeken alapuló approximációs párok

A reprezentatív elemek jelentősége az algoritmusok erőforrás igényének csökkentésében rejlik. A halmazok közelítésénél az alaphalmazok minden elemét figyelembe kell venni, mely nagyméretű halmazok esetén költséges lehet. A dolgozatban három approximációs pár került bemutatásra, melyek a reprezentatív elemeken alapszanak. Ezeket a párok két főbb csoportba sorolhatóak:

- Hasonlóság alapú életlen halmazokra támaszkodó approximációs párok
- Halmaz központú approximációs párok

6.3.1. Hasonlóság alapú életlen halmazokra támaszkodó approximációs párok

Legyen $B \in \mathfrak{B}$ egy alaphalmaz és $REP(B)$ a reprezentatív elemeinek halmaza. A közelítő függvények a következőképpen alakulnak:

- $\mathsf{l}_\mathsf{r}(S) = \bigcup\{B \mid B \in \mathfrak{B} \text{ és } \forall x \in REP(B) : x \in S\}$;
- $\mathsf{u}_\mathsf{r}(S) = \bigcup\{B \mid B \in \mathfrak{B} \text{ és } \exists x \in REP(B) : x \in S\}$.

Így az alsó közelítés azon alaphalmazok uniójából tevődik össze, melyek összes reprezentatív eleme benne van a közelítendő halmazban. A felső közelítésbe pedig azok az alaphalmazok tartoznak, melyeknek legalább egy reprezentatív eleme benne van a közelítendő halmazban. Természetesen az alsó közelítésben rejlő bizonyosság kis mértékben sérül, de az adathalmaz növekedésével csökkenthető a közelítések kiszámításához szükséges erőforrások.

6.3.2. Halmaz központú approximációs párok

Legyen $[x] = \{y \mid x\mathcal{R}y\}$ az x objektumhoz hasonlatos elemek halmaza és $REP(S)$ az S halmaz reprezentatív elemeinek halmaza. Az S halmazon alapuló két approxmációs pár a következőképpen adható meg: $\langle \mathsf{l}, \mathsf{u}_1 \rangle$ és $\langle \mathsf{l}, \mathsf{u}_2 \rangle$, ahol

$$\mathsf{l}(S) = \bigcup_{\substack{x \in REP(S) \\ [x] \subseteq S}} \{[x]\}$$

$$\mathsf{u}_1(S) = \bigcup_{x \in REP(S)} \{[x]\}$$

$$\mathsf{u}_2(S) = \bigcup_{\substack{x \in REP(S) \\ \|[x] \cap S\| > \|[x] \setminus S\|}} \{[x]\}$$

Ebben a térben egy alaphalmaz azon objektumokat tartalmazza, amelyek hasonlóak egy bizonyos reprezentatívhoz. Így minden reprezentatív egy alaphalmazt fog generálni. Tehát az alaphalmazok rendszere mindig a közelítendő halmaztól függ.

6.4. Gráf-Approximáció hasonlóságon alapú életlen halmazokkal

Az életlen halmazok elmélete egy lehetséges utat kínál a bizonytalanság kezelésére a halmazapproximáció segítségével. Az hasonlóságon alapuló appropximációs terek az objektumok közötti tolerancia relációra támaszkodnak. Egy hasonlóságot reprezentáló tolerancia reláció megadható egy előjeles gráffal. Ez a gráf teljes, ha a reláció is teljes és nem teljes, ha a reláció parciális. A reflexivitás miatt minden csúcsból mutat egy hurokél önmagába. Ha két objektum hasonló, akkor egy pozitív él húzódik köztük, míg ha különbözőek akkor ez az él negatív. Minden gráf reprezentálható egy rendezett párokból álló halmazzal. Ebben az esetben rendezett hármasokat tartalmazó halmazzal. Ha egy hasonlóságot reprezentáló gráf egy halmazzal megadható, akkor felmerülhet a következő kérdés. Lehet-e ezt a halmazt közelíteni? A kérdésre a választ a dolgozat a 4. fejezetben tárgyalja.

6.5. Konklúzió

A dolgozat legfontosabb eredménye egy oylan teljesen új approximációs tér kifejlesztése volt, mely egy tolerancia relación alapul és az alaphalmazok a korrelációs klaszterezés eredményeként jönnek létre. A bemutatott tér (hasonlóság alapú életlen halmazok) számos jó tulajdonsággal rendelkezik és különbözik a tolerancia relación alapuló lefedő terektől. Sikeresen

kifejlesztésre került egy olyan módszert, mellyel könnyedén lehet tolerancia/hasonlósági relációkat ábrázolni. Elkészítettünk továbbá egy olyan algoritmust, mellyel minden klaszterből ki lehet választani azon elemeket, melyek a leginkább hasonlítanak a többihez. Az ilyen objektumokat reprezentatív elemeknek neveztük. A módszer jó tulajdonsága, hogy mindig megfelelő számú (nem túl sok és nem túl kevés) reprezentánst válogat ki így csökkentve a számolás során figyelembe veenedő objektumok számát. A reprezentatív elemekre támaszkodva több új approximációs-párt is definiáltunk. A hasonlóság alapú életlen halmazokra támaszkodó gráf-approximációs folyamattal pedig egy olyan algoritmus került bemutatásra, mellyel egy adathalmaz két attribútumhalmaza közti szorosságot lehet vizsgálni és az adatok előfeldolgozásában használatos ún. jellemző kinyerésben is alkalmazható.

Köszönetnyilvánítás

Szeretném megköszönni páromnak, szüleimnek, bátyámnak valamint nagyszüleimnek, hogy biztató szavaikkal hozzájárultak a dolgozat elkészüléséhez.

Köszönettel tartozom témavezetőmnek Dr. Mihálydeák Tamásnak mindazon szakmai iránymutatásért és támogatásért, melyek nélkül a dolgozat nem jöhetett volna létre.

Hálával tartozom kollégáimnak Dr. Aszalós Lászlónak és Dr. Várterész Magdának, akik egyetemi tanulmányaim alatt számos alkalommal nyújtottak segítséget és útmutatást.

Végül szeretném megköszönni Dr. Kádek Tamás kollégámnak a dolgozat megírása során adott számos ötletét és tanácsát.

Appendix A

Algorithms

Between 2006 and 2016, "Advanced Search Methods" was a compulsory subject for some Computer Science master students. Initially, students learned about the well known NP-hard problems (SAT, NLP, TSP, etc.) and various popular optimization methods. Later, to help some physicists from University Babes-Bolyai in Cluj, one co-author began to research the problem of correlation clustering. This problem can easily be formulated (which equivalence relation is the closest to a given tolerance relation?), can be quickly understood, is freely scalable, but NP-hard, and if there are more than 15 objects in a general case an approximate solution can only be provided. That is why this problem got a central role from 2010. In this year, the students with the co-author's lead implemented the learned algorithms, and used correlation clustering to test and compare them. There were several didactic goals of this development: they worked as a team, where the leader changed from algorithm to algorithm, whose duty was to distribute the subtasks among the others and compose/finalize their work. The implementations together gave thousand of LOCs, so the students got experience with a real-life size problem. When designing this system as a framework, the OOP principles were of principal importance, to be able to apply it for other optimization problems. The system was completed with several refactorisations and extended with methods developed directly for correlation clustering, and several special data structures which allowed to run programs several magnitudes faster. Finally, the full source of the whole system with detailed explanations was published at the Hungarian Digital Textbook Repository [3]. According to the students' requests, the system was written in Java. Jason Brownlee published a similar book using Ruby [11].

In the next subsections, there is some brief information about the used algorithms and their parameters. The whole descriptions can also be seen in [3].

A.1 Hill Climbing Algorithm

This method is very well-known. Each state in the search plane represents a partition. A state is considered better than another state if its number of conflicts is less than that of the other state. In each step, it is checked, whether there is a better state in the neighborhood of the actual state. If there is not, then the algorithm stops. If there is, then the next step goes from this point.

A.2 Stochastic Hill Climbing Algorithm

The original hill climbing search is greedy, it always moves to the best neighbor. Stochastic hill climbing is a variant, where the algorithm chooses from the neighbors in proportion to their goodness, allowing the algorithm to move in a worse direction as well.

A.3 Tabu Search

Each state in the search plane represents a partition like in the previous algorithms. The tabu search defines a list of banned states or directions to where it cannot move at a time. This is called a tabu list or memory. There are many types of memories. In the experiments, a short-term memory was used with a size of 50.

The neighborhood of the actual state consists of banned and permitted states. In each step, there are two possibilities:

- If one of the neighbors is so good that it is better than the best state so far, then it should go there even if it is banned. This is called the aspirant condition.
- The algorithm moves to the best-permitted neighbor of the actual state.

If the new state is better than the best state so far, then this state will be the new best. The previous state will also be added to the tabu list, so the algorithm could not go backwards immediately. If the list is full, then the last state will be deleted. The algorithm stops if it reaches the 1000th step and it returns the best state.

A.4 Simulated Annealing

Each state in the search plane represents a partition. In each step, the algorithm chooses a neighbor of the actual state. Let f denote the number of conflicts in the actual state and f' denote the number of conflicts in the neighbor. If $f' < f$, then the algorithm moves to the neighbor. If not, then it chooses this state with the probability of $\frac{e^{f-f'}}{T}$. The value T is the temperature value which is a crucial parameter. It should decrease in each step. Determining the starting temperature value is a hard task. The common method for the issue is heating. In each temperature (starting from 1), 500 attempts are made to move to a neighbor, and the number of successful movements are counted. If the ratio of the number of successful movements and the number of attempts reaches 0.99, then the heating procedure stops, otherwise the temperature value is increased. After the heating, the annealing (search) step comes. It is important that how much time the algorithm spends in each temperature value. A minimal step count was defined and it increases each time the temperature is decreased until it reaches a maximal value when the algorithm stops and returns the best state. The minimal step count was set to 100 and the maximal was set to 1000. The temperature values are always decreased by 97%.

A.5 Parallel Tempering

The simulated annealing runs on a single thread. This is a parallelized version, where threads can cooperate. In this method, 3 threads were used.

A.6 Genetic Algorithm

In this algorithm, each partition is represented by an entity. In each search step, there is a population of entities with a size of 100. In the beginning, each entity represents a random partition. This population contains the actual generation of entities and the best entities from the old generation. In each step, the best 25 entities stay in the population. The rest of the spaces are filled with the descendants of the entities of the old generation. In step 1, the algorithm defines the new generation. Step 2 is the reproduction step. A descendant is created in this step with the crossover of 2 parent entities. In the experiments, one-point crossover was used. For the crossover, not a random or the best element is chosen but an element from a set defined by a parameter. The size of this set was 4. After step 2, each child entity goes through a mutation phase (step 3) whose probability was 2/3. After each child entity is created, the actual generation is overwritten by the new generation, and the algorithm goes back to step 1. The algorithm stops when it reaches the 2000th generation and returns the best entity of the population.

A.7 Bees Algorithm

This algorithm is based on the society of honey bees. Each partition is represented by a "bee". There are two types of bees: scout bees and recruit bees. Scout bees scout the area and they report back to the hive about their findings. Then the necessary number (in proportion to the goodness of the finding) of recruit bees go to the area to forage. In this case, the scouts are scattered across the search plane and recruit bees were assigned only to the best of them. These bees are called elites. The rest of the scout bees wander in the plane. It changes dynamically which scout bees are considered as elite and how many recruits are assigned to them. The recruits search around the elite bee to which they were assigned, and if they find a better state, then the scout bees move to that position. In the experiments, the number of scout bees was set to 50 and the number of elites was 5 and 1000 recruit bees follow the elites. In the beginning, the scout bees start from a random position. The algorithm stops when it reaches the 2000th step and it returns the partition represented by the best bee.

A.8 Particle Swarm Optimization

In this algorithm, each partition is represented by an insect (particle). Each insect knows its best position and the best position of the swarm. The size of the swarm was set to 50. In each step, each insect moves in the search plane. In the beginning, the insects start from a random position. There are 3 possibilities for them to move:

- Randomly move

- Move towards its best position
- Move toward the best position of the swarm

The possibilities of the moves was set to 0.2, 0.3, 0.5 respectively. After reaching the 6000th step, the algorithm stops and returns the insect with the best position.

A.9 Firefly Algorithm

In this algorithm, each partition is represented by a firefly. The fireflies are unisex and their brightnesses are proportionate to the goodness of the partition they represent. In the beginning, the fireflies start from a random position, and in each step, each firefly moves to its brightest neighbor. If the brightest neighbor of a firefly is itself, then it moves randomly. Brightness is dependent on the distance of the insects. The intensity of a firefly is defined by the following formula: $I_d = \frac{I_0}{1+\gamma d^2}$, where I_0 denotes the starting intensity, γ is the absorption coefficient (was set to 0.03) and d is the distance between the two fireflies. After 10000 steps the algorithm stops, and the result is the partition that is represented by the brightest firefly. The number of fireflies was set to 50.

Appendix B

Software

I developed a program that helps us approximate different types of sets using the proposed new approximation space (similarity based rough sets). The software can be downloaded from:
`https://github.com/Nagy-David/Similarity-Based-Rough-Sets`. For giving the input datasets, the user has two options:

1. Generating random coordinate points
2. Reading a dataset from a file

1. *Random Points*

   The user gives the number of points, and then the points are generated in a 2-dimensional interval which is also given by the user In this option, the base of the tolerance relation (representing the similarity) is the Euclidean distance of the objects (d). A similarity ($SIMM$) and a dissimilarity threshold ($DIFF$) were defined. The tolerance relation $\mathcal{R}$ can be given this way for any objects x, y:

$$x\mathcal{R}y = \begin{cases} +1 & d(x,y) \leq SIMM \\ -1 & d(x,y) > DIFF \\ 0 & \text{otherwise} \end{cases} \tag{B.1}$$

2. *Continuous Data*

   Each row represents a single entity. In the software, there is an option to normalize the data in the way described below. Let A be an attribute and v the value to be normalized. After the normalization:

$$v = \frac{v - min(A)}{max(A) - min(A)} \tag{B.2}$$

   The similarity is defined in two steps.

   (a) step: Let $A_1, A_2 \dots A_n$ be some attributes, $t_1, t_2 \dots t_n$ be threshold values and x, y be two objects. Let $z(A_i)$ denote the attribute value of A_i for any object

$(i = 1 \dots n)$. If $\exists i \in \{1 \dots n\} : |x(A_i) - y(A_i)| \geq t_i$, then the objects x and y are treated as different.

(b) step: If the condition in the first step does not hold, then the tolerance relation $\mathcal{R}$ can be defined in the following way for any objects x, y using a similarity threshold $SIMM$ and a dissimilarity threshold $DIFF$:

$$x\mathcal{R}y = \begin{cases} +1 & d(x,y) \leq SIMM \\ -1 & d(x,y) > DIFF \\ 0 & \text{otherwise} \end{cases} \tag{B.3}$$

The d "distance"' value is calculated for any objects x, y by the following method:

$$d(x,y) = \sqrt{\sum_{i=1}^{n} (x(A_i) - y(A_i))^2} \tag{B.4}$$

The necessity of the first step can be explained by the following simple example. Let us assume that the objects are patients. Two patients may differ only in the blood pressure level and the other attribute values are relatively close to one another. So the distance between these two entities can be a small value. However, the patients cannot be treated as similar because a high blood pressure level can indicate an illness. This fact remains hidden without the first step, because the similarity value can be small for the two patients. The same holds for normalized data.

After getting the input points the software runs a search/optimization algorithm that finds a quasi-optimal partition. As mentioned earlier, the singleton clusters mean little information, so the software leaves them out and creates the system of base sets. After defining the base sets, the user can select a set of points for approximation.

In the software. the user has the option to insert the members of the left-out singleton clusters to any base set. Two singleton clusters cannot be merged together due to the tolerance relation based on similarity (their members are different). It was mentioned earlier that there are two types of singletons:

- Its member is different from most of the objects so it forms a cluster alone.
- Due to the background knowledge the system decided that this object cannot be a member of any other group.

The software does not examine for a singleton to which type it belongs, so there is no mandatory annotation for any singleton. It is up to the user to decide.

Irodalomjegyzék

[1] AIGNER, M. Enumeration via ballot numbers. *Discrete Mathematics 308*, 12 (2008), 2544 – 2563.

[2] ALTMAN, A., AND TENNENHOLTZ, M. Ranking systems: the pagerank axioms. In *Proceedings of the 6th ACM conference on Electronic commerce* (2005), ACM, pp. 1–8.

[3] ASZALÓS, L., AND BAKÓ, M. Advanced search methods (in Hungarian). `http://morse.inf.unideb.hu/~aszalos/diak/fka`, 2012.

[4] ASZALÓS, L., HAJDU, L., AND PETHŐ, A. On a correlational clustering of integers. *Indagationes Mathematicae 27*, 1 (2016), 173–191.

[5] ASZALÓS, L., AND NAGY, D. Visualization of tolerance relations. In *Proceedings of the 10th International Conference on Applied Informatics* (2018), K. Gábor, Kusper; Roland, Ed., pp. 15–22.

[6] ASZALÓS, L., AND NAGY, D. Iterative set approximations based on tolerance relation. In *Rough Sets* (Cham, 2019), T. Mihálydeák, F. Min, G. Wang, M. Banerjee, I. Düntsch, Z. Suraj, and D. Ciucci, Eds., Springer International Publishing, pp. 78–90.

[7] ASZALÓS, L., AND NAGY, D. Selecting representatives. In *Communication Papers of the 2019 Federated Conference on Computer Science and Information Systems* (2019), M. Ganzha, L. Maciaszek, and M. Paprzycki, Eds., vol. 20 of *Annals of Computer Science and Information Systems*, PTI, pp. 13–19.

[8] BANSAL, N., BLUM, A., AND CHAWLA, S. Correlation clustering. *Machine Learning 56*, 1-3 (2004), 89–113.

[9] BECKER, H. A survey of correlation clustering. *Advanced Topics in Computational Learning Theory* (2005), 1–10.

[10] BELLO, R., AND FALCON, R. *Rough Sets in Machine Learning: A Review*. Springer International Publishing, Cham, 2017, pp. 87–118.

[11] BROWNLEE, J. *Clever Algorithms: Nature-Inspired Programming Recipes*, 1st ed. Lulu.com, 2011.

[12] CABALLERO, Y., BELLO, R., ALVAREZ, D., GAREIA, M. M., AND PIZANO, Y. Improving the k-nn method: Rough set in edit training set. In *Professional Practice in Artificial Intelligence* (Boston, MA, 2006), J. Debenham, Ed., Springer US, pp. 21–30.

[13] CIUCCI, D., MIHÁLYDEÁK, T., AND CSAJBÓK, Z. E. *On Definability and Approximations in Partial Approximation Spaces*. Springer International Publishing, Cham, 2014, pp. 15–26.

[14] CSAJBÓK, Z., AND MIHÁLYDEÁK, T. Partial approximative set theory: A generalization of the rough set theory. *International Journal of Computer Information Systems and Industrial Management Applications 4* (12 2012), 437–444.

[15] CSAJBÓK, Z., AND MIHÁLYDEÁK, T. A general set theoretic approximation framework. In *Advances on Computational Intelligence*, S. Greco, B. Bouchon-Meunier, G. Coletti, M. Fedrizzi, B. Matarazzo, and R. Yager, Eds., vol. 297 of *Communications in Computer and Information Science*. Springer Berlin Heidelberg, 2012, pp. 604–612.

[16] CSAJBÓK, Z., AND MIHÁLYDEÁK, T. From vagueness to rough sets in partial approximation spaces. In *JRS2014* (2014). Submitted.

[17] DELIC, D., LENZ, H.-J., AND NEILING, M. Improving the quality of association rule mining by means of rough sets. In *Soft Methods in Probability, Statistics and Data Analysis* (Heidelberg, 2002), P. Grzegorzewski, O. Hryniewicz, and M. Á. Gil, Eds., Physica-Verlag HD, pp. 281–288.

[18] DO PRADO, H. A., ENGEL, P. M., AND FILHO, H. C. Rough clustering: An alternative to find meaningful clusters by using the reducts from a dataset. In *Rough Sets and Current Trends in Computing* (Berlin, Heidelberg, 2002), J. J. Alpigini, J. F. Peters, A. Skowron, and N. Zhong, Eds., Springer Berlin Heidelberg, pp. 234–238.

[19] DUBOIS, D., AND PRADE, H. Twofold fuzzy sets and rough sets—some issues in knowledge representation. *Fuzzy Sets and Systems 23*, 1 (1987), 3 – 18. Fuzzy Information Processing in Artificial Intelligence and Operations Research.

[20] DUBOIS, D., AND PRADE, H. Rough fuzzy sets and fuzzy rough sets*. *International Journal of General Systems 17*, 2-3 (1990), 191–209.

[21] DÜNTSCH, I., AND GEDIGA, G. *Approximation Operators in Qualitative Data Analysis*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003, pp. 214–230.

[22] DÁVID NAGY, LÁSZLÓ ASZALÓS, T. M. Finding the representative in a cluster using correlation clustering. *Pollack Periodica 14*, 1 (2019), 15– 24.

[23] ERDÖS, P., AND RÉNYI, A. On random graphs i. *Publicationes Mathematicae Debrecen 6* (1959), 290.

[24] ERDŐS, P., AND RÉNYI, A. On the evolution of random graphs. In *PUBLICATION OF THE MATHEMATICAL INSTITUTE OF THE HUNGARIAN ACADEMY OF SCIENCES* (1960), pp. 17–61.

[25] FILIBERTO, Y., CABALLERO, Y., LARRUA, R., AND BELLO, R. A method to build similarity relations into extended rough set theory. In *2010 10th International Conference on Intelligent Systems Design and Applications* (Nov 2010), pp. 1314–1319.

[26] GOGOI, P., BHATTACHARYYA, D. K., AND KALITA, J. K. A rough set-based effective rule generation method for classification with an application in intrusion detection. *Int. J. Secur. Netw. 8*, 2 (Aug. 2013), 61–71.

[27] GRECO, S., MATARAZZO, B., AND SŁOWIŃSKI, R. Parameterized rough set model using rough membership and bayesian confirmation measures. *International Journal of Approximate Reasoning 49*, 2 (2008), 285 – 300. Special Section on Probabilistic Rough Sets and Special Section on PGM'06.

[28] GRZYMALA-BUSSE, J. W. *LERS-A System for Learning from Examples Based on Rough Sets*. Springer Netherlands, Dordrecht, 1992, pp. 3–18.

[29] GUAN, J. W., BELL, D. A., AND LIU, D. Y. The rough set approach to association rule mining. In *Third IEEE International Conference on Data Mining* (Nov 2003), pp. 529–532.

[30] HU, Q., YU, D., LIU, J., AND WU, C. Neighborhood rough set based heterogeneous feature subset selection. *Information Sciences 178*, 18 (2008), 3577 – 3594.

[31] LENARCIK, A., AND PIASTA, Z. *Discretization of Condition Attributes Space*. Springer Netherlands, Dordrecht, 1992, pp. 373–389.

[32] MANI, A. Choice inclusive general rough semantics. *Information Sciences 181*, 6 (2011), 1097–1115.

[33] MIHÁLYDEÁK, T. Logic on similarity based rough sets. In *Rough Sets* (Cham, 2018), H. S. Nguyen, Q.-T. Ha, T. Li, and M. Przybyła-Kasperek, Eds., Springer International Publishing, pp. 270–283.

[34] MISES, R. V., AND POLLACZEK-GEIRINGER, H. Praktische verfahren der gleichungsauflösung . *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik 9*, 2 (1929), 152–164.

[35] NAGY, D., AND ASZALÓS, L. Approximation based on representatives. In *Rough Sets* (Cham, 2019), T. Mihálydeák, F. Min, G. Wang, M. Banerjee, I. Düntsch, Z. Suraj, and D. Ciucci, Eds., Springer International Publishing, pp. 91–101.

[36] NAGY, D., MIHÁLYDEÁK, T., AND ASZALÓS, L. *Similarity Based Rough Sets*. Springer International Publishing, Cham, 2017, pp. 94–107.

[37] NAGY, D., MIHÁLYDEÁK, T., AND ASZALÓS, L. Similarity based rough sets with annotation. In *Rough Sets* (Cham, 2018), H. S. Nguyen, Q.-T. Ha, T. Li, and M. Przybyła-Kasperek, Eds., Springer International Publishing, pp. 88–100.

[38] NAGY, D., MIHALYDEAK, T., AND ASZALOS, L. Different types of search algorithms for rough sets. *Acta Cybernetica 24*, 1 (May 2019), 105–120.

[39] NÉDA, Z., SUMI, R., ERCSEY-RAVASZ, M., VARGA, M., MOLNÁR, B., AND CSEH, G. Correlation clustering on networks. *Journal of Physics A: Mathematical and Theoretical 42*, 34 (Aug. 2009), 345003.

[40] NÉDA, Z., SUMI, R., ERCSEY-RAVASZ, M., VARGA, M., MOLNÁR, B., AND CSEH, G. Correlation clustering on networks. *Journal of Physics A: Mathematical and Theoretical 42*, 34 (2009), 345003.

[41] NGUYEN, H. S. Discretization problem for rough sets methods. In *Rough Sets and Current Trends in Computing* (Berlin, Heidelberg, 1998), L. Polkowski and A. Skowron, Eds., Springer Berlin Heidelberg, pp. 545–552.

[42] PAWLAK, Z. Rough sets. *International Journal of Parallel Programming 11*, 5 (1982), 341–356.

[43] PAWLAK, Z., ET AL. Rough sets: Theoretical aspects of reasoning about data. *Systern Theory, Knowledge Engineering and Problem Solving, Kluwer Academic Publishers, Dordrecht, 199l 9* (1991).

[44] PAWLAK, Z., AND SKOWRON, A. Rough sets: Some extensions. *Information Sciences 177*, 1 (2007), 28 – 40. Zdzisław Pawlak life and work (1926–2006).

[45] PAWLAK, Z., AND SKOWRON, A. Rudiments of rough sets. *Information sciences 177*, 1 (2007), 3–27.

[46] PAWLAK, Z., WONG, S., AND ZIARKO, W. Rough sets: probabilistic versus deterministic approach. *International Journal of Man-Machine Studies 29*, 1 (1988), 81 – 95.

[47] SKOWRON, A., AND STEPANIUK, J. Tolerance approximation spaces. *Fundamenta Informaticae 27*, 2 (1996), 245–253.

[48] ŚLĘZAK, D., AND ZIARKO, W. The investigation of the bayesian rough set model. *International Journal of Approximate Reasoning 40*, 1 (2005), 81 – 91. Data Mining and Granular Computing.

[49] SWINIARSKI, R. W., AND SKOWRON, A. Rough set methods in feature selection and recognition. *Pattern Recognition Letters 24*, 6 (2003), 833 – 849.

[50] YANG, Y., CHEN, D., AND DONG, Z. Novel algorithms of attribute reduction with variable precision rough set model. *Neurocomputing 139* (2014), 336 – 344.

[51] YAO, Y. Decision-theoretic rough set models. In *Rough Sets and Knowledge Technology* (Berlin, Heidelberg, 2007), J. Yao, P. Lingras, W.-Z. Wu, M. Szczuka, N. J. Cercone, and D. Ślęzak, Eds., Springer Berlin Heidelberg, pp. 1–12.

[52] YAO, Y., GRECO, S., AND SŁOWIŃSKI, R. *Probabilistic Rough Sets*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015, pp. 387–411.

[53] YAO, Y., AND YAO, B. Covering based rough set approximations. *Information Sciences 200* (2012), 91 – 107.

[54] YAO, Y. Y. *Combination of Rough and Fuzzy Sets Based on α-Level Sets*. Springer US, Boston, MA, 1997, pp. 301–321.

[55] ZAHN, JR, C. Approximating symmetric relations by equivalence relations. *Journal of the Society for Industrial & Applied Mathematics 12*, 4 (1964), 840–847.

[56] ZAKOWSKI, W. Approximations in the space (u,π). *Demonstratio Mathematica 16*, 3 (1983), 761–770.

[57] ZIMEK, A. Correlation clustering. *ACM SIGKDD Explorations Newsletter 11*, 1 (2009), 53–54.

Publikációs lista

1. **Different Types of Search Algorithms for Rough Sets** (Dávid Nagy, Tamás Mihálydeák and László Aszalós), *In Acta Cybernetica*, volume 24, pp. 105-120, 2019.
2. **Finding the representative in a cluster using correlation clustering** (Dávid Nagy, László Aszalós and Tamás Mihálydeák), *In Pollack Periodica*, volume 14, pp. 15- 24, 2019.
3. **Iterative Set Approximations Based on Tolerance Relation** (László Aszalós and Dávid Nagy), In Rough Sets (Tamás Mihálydeák, Fan Min, Guoyin Wang, Mohua Banerjee, Ivo Düntsch, Zbigniew Suraj and Davide Ciucci, ed.), Springer International Publishing, pp. 78-90, 2019.
4. **Approximation Based on Representatives** (Dávid Nagy and László Aszalós), In Rough Sets (Tamás Mihálydeák, Fan Min, Guoyin Wang, Mohua Banerjee, Ivo Düntsch, Zbigniew Suraj and Davide Ciucci, ed.), Springer International Publishing, pp. 91-101, 2019.
5. **Selecting representatives** (László Aszalós and Dávid Nagy), *In Communication Papers of the 2019 Federated Conference on Computer Science and Information Systems (Maria Ganzha, Leszek Maciaszek, Marcin Paprzycki, eds.)*, Annals of Computer Science and Information Systems, PTI, volume 20, pp. 13-19, 2019.
6. **Visualization of tolerance relations** (László Aszalós and Dávid Nagy), *In Proceedings of the 10th International Conference on Applied Informatics (Gábor Kusper and Roland Király ed.)*, pp. 15-22, 2018.
7. **Similarity Based Rough Sets with Annotation** (Dávid Nagy, Tamás Mihálydeák and László Aszalós), In Rough Sets (Hung Son Nguyen, Quang-Thuy Ha, Tianrui Li and Małgorzata Przybyła-Kasperek, ed.), Springer International Publishing, pp. 88-100, 2018.
8. **Similarity Based Rough Sets** (Dávid Nagy, Tamás Mihálydeák and László Aszalós), In Rough Sets (Lech Polkowski, Yiyu Yao, Piotr Artiemjew, Davide Ciucci, Dun Liu, Dominik Ślęzak and Beata Zielosko, ed.), Springer International Publishing, pp. 94-107, 2017.
9. **Conjectures on phase transition at correlation clustering of random graphs** (László Aszalós, János Kormos and Dávid Nagy), *In Annales Univ. Sci. Budapest., Sect. Comp.*, pp. 37-54, 2014.
10. **Graph Approximation on Similarity Based Rough Sets** (Dávid Nagy, Tamás Mihálydeák and László Aszalós), *In Pollack Periodica* [ACCEPTED FOR PUBLICATION]