# Similarity Based Rough Sets

Dávid Nagy(✉), Tamás Mihálydeák, and László Aszalós

Faculty of Informatics, University of Debrecen, Debrecen, Hungary
{nagy.david,mihalydeak.tamas,aszalos.laszlo}@inf.unideb.hu

**Abstract.** Pawlak's indiscernibility relation (which is an equivalence relation) represents a limit of our knowledge embedded in an information system. In many cases covering approximation spaces rely on tolerance relations instead of equivalence relations. In real practice (for example in data mining) tolerance relations may be generated from the properties of objects. A given tolerance relation represents similarity between objects, but the usage of similarity is very special: it emphasizes the similarity to a given object and not the similarity of objects 'in general'. The authors show that this usage has some problematic consequences. The main goal of the paper is to show that if one uses the method of correlation clustering then there is a way to construct a general (partial) approximation space with disjoint base sets relying on the similarity of objects generated by their properties. At the end a software describing a real life problem is presented.

**Keywords:** Rough set theory · Correlation clustering · Set approximation

## 1 Introduction

From the theoretical point of view a Pawlakian approximation space (see in [12–14]) can be characterized by an ordered pair $\langle U, \mathcal{R}\rangle$ where $U$ is a nonempty set of objects and $\mathcal{R}$ is an equivalence relation on $U$. In order to approximate an arbitrary subset $S$ of $U$ the following tools have to be introduced:

- *the set of base sets*: $\mathfrak{B} = \{B \mid B \subseteq U,$ and $x, y \in B$ if $x\mathcal{R}y\}$, the partition of $U$ generated by the equivalence relation $\mathcal{R}$;
- *the set of definable sets*: $\mathfrak{D}_{\mathfrak{B}}$ is an extension of $\mathfrak{B}$, and it is given by the following inductive definition:
  1. $\mathfrak{B} \subseteq \mathfrak{D}_{\mathfrak{B}}$;
  2. $\emptyset \in \mathfrak{D}_{\mathfrak{B}}$;
  3. if $D_1, D_2 \in \mathfrak{D}_{\mathfrak{B}}$, then $D_1 \cup D_2 \in \mathfrak{D}_{\mathfrak{B}}$.
- *the functions* $\mathsf{l}, \mathsf{u}$ form a Pawlakian approximation pair $\langle \mathsf{l}, \mathsf{u}\rangle$, i.e.
  1. $Dom(\mathsf{l}) = Dom(\mathsf{u}) = 2^U$
  2. $\mathsf{l}(S) = \bigcup\{B \mid B \in \mathfrak{B}$ and $B \subseteq S\}$;
  3. $\mathsf{u}(S) = \bigcup\{B \mid B \in \mathfrak{B}$ and $B \cap S \neq \emptyset\}$.


L. Polkowski et al. (Eds.): IJCRS 2017, Part II, LNAI 10314, pp. 94–107, 2017.
DOI: 10.1007/978-3-319-60840-2_7

$\mathcal{R}$ is called an indiscernibility relation. It represents a sort of limit of our knowledge embedded in an information system (or background knowledge). Indiscernibility has an affect 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 other objects which are indiscernible from the given object. Indiscernibility plays a crucial role in approximation process: if we are interested in whether $x \in S$ (where $S$ is the set to be approximated), then

1. the answer 'yes' (i.e. $x \in \mathsf{l}(S)$) means that not only $x \in S$ but all $y$, such that $x\mathcal{R}y$ are members of $S$;
2. the answer 'no' (i.e. $x \in \mathsf{l}(\overline{S})$, where $\overline{S}$ is the complement of $S$) means that not only $x \notin S$ but all $y$, such that $x\mathcal{R}y$ are not members of $S$;
3. the answer 'maybe' (i.e. $x \in \mathsf{u}(S) \setminus \mathsf{l}(S)$) means that there are $y_1, y_2$ such that $x\mathcal{R}y_1$ and $x\mathcal{R}y_2$ for which $y_1 \in S$ and $y_2 \notin S$.

In practical applications indiscernibility relation is too strong: we have to handle indiscernible objects in the same way but in real practice we have to consider them as similar objects. Pawlakian approximation spaces have been generalized using tolerance relations (instead of equivalence ones), which are similarity relations and so they are symmetric and reflexive. Covering-based approximation spaces (see for instance [16]) generalize Pawlakian approximation spaces in two points:

1. $\mathcal{R}$ is (only) a tolerance relation;
2. $\mathfrak{B} = \{[x] \mid [x] \subseteq U, x \in U \text{ and } y \in [x] \text{ if } x\mathcal{R}y\}$, where $[x] = \{y \mid y \in U, x\mathcal{R}y\}$.

These spaces use the definitions of definable sets and approximation pairs of Pawlakian approximation spaces.

Covering approximation spaces use similarity relations instead of equivalence relations, but the usage of similarity relations (which are tolerance relations from the mathematical point of view) is very special. It emphasizes the similarity to a given object and not the similarity of objects 'in general'. We can recognize this feature when we try to understand the precise meaning of the answer coming from an approximation relying on a covering approximation space. If we are interested in whether $x \in S$ (where $S$ is the set to be approximated), then (see Fig. 1)

1. the answer 'yes' (i.e. $x \in \mathsf{l}(S)$) means that there is an object $x'$ such that $x'\mathcal{R}x, x' \in S$ and all $y$ for which $x'\mathcal{R}y$ are members of $S$;
2. the answer 'no' (i.e. $x \in \mathsf{l}(\overline{S})$) means that there is an object $x'$ such that $x'\mathcal{R}x, x' \notin S$ and all $y$ for which $x'\mathcal{R}y$ are not members of $S$;
3. otherwise the answer is 'maybe' (informally $x$ is a member of the border of $S$) means that there is no $x'$ for which $x'\mathcal{R}x, [x'] \subseteq U$, and there is no $x''$ for which $x''\mathcal{R}x, [x''] \cap U \neq \emptyset$.

Some practical problems of covering approximation spaces:

1. The former answers show, that generally the lower and upper approximations are not close in the following sense (see Fig. 2):

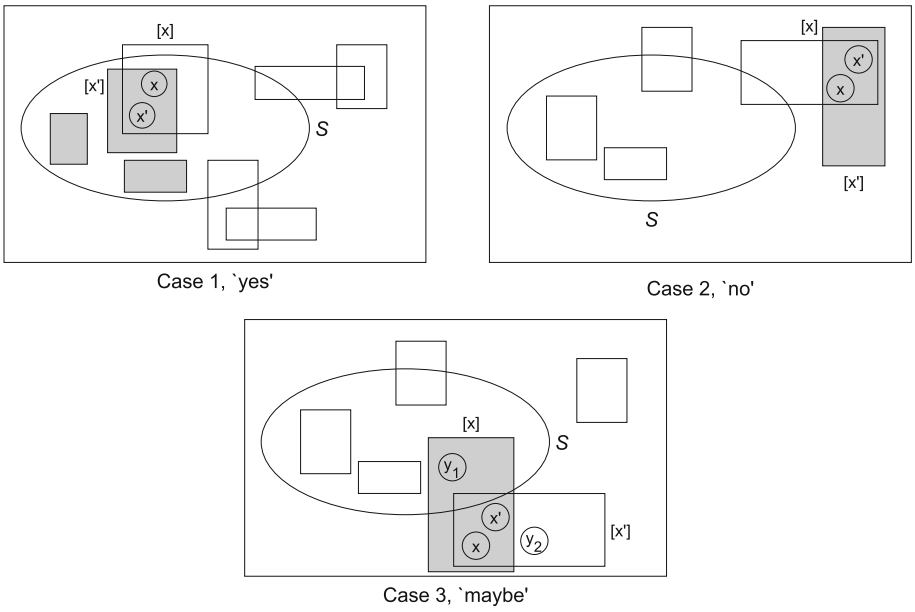



**Fig. 1.** Some base sets in covering cases

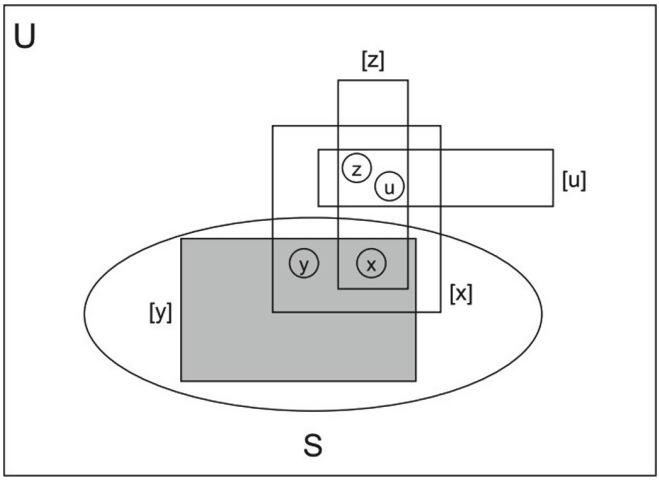
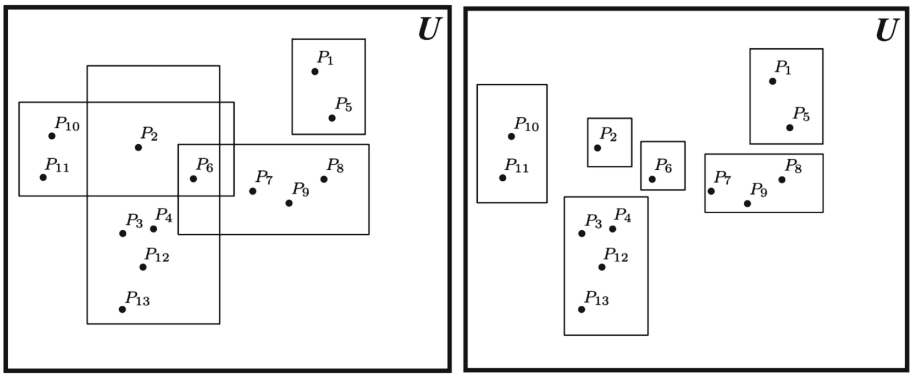



**Fig. 2.** In covering the lower and upper approximations are not closed

(a) If $x \in \mathsf{l}(S)$, then we cannot say that $[x] \subseteq S$.
(b) If $x \in \mathsf{u}(S)$, then we cannot say, that $[y] \cap S \neq \emptyset$ for all $y \in [x]$.

2. The number of base sets is not more than the number of members of $U$, so we have too many base sets for practical applications.

If we want to avoid these problems we can generate a Pawlakian approximation space by constructing a system of disjoint base sets (see in [7]) (see Fig. 3). If we have two base sets $B_1, B_2$, such that $B_1 \cap B_2 \neq \emptyset$, then we substitute them with the following three sets: $B_1 \setminus B_2, B_2 \setminus B_1, B_1 \cap B_2$. Applying this iteratively we can get the reduction. 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 base sets we have, the closer we are to the classical set theory. (If all base sets are singleton, then there is no difference between the approximation space and classical set theory.)




(a) Covering (b) Partition

**Fig. 3.** Covering and its reduction to a partition

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

- In Pawlak's original system 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 all members of a given base set are similar to a distinguished object (which is used to generate the given base set).

Further generalization is possible (see): General (partial) Pawlakian approximation spaces can be obtained by 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 we can use very different properties in order to define different base sets. The members of the base set can be handled in the same way relying on their common property. In this case there is no way to give a corresponding relation

which is able to 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 we use this relation in order to get the system of base sets, the result can be totally different from our original base system (see Fig. 4).

In Fig. 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 we used only two base sets, but it is the same when we have more.)

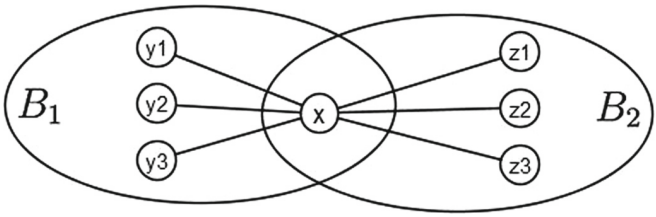



**Fig. 4.** Base sets by properties of objects

The main goal of the present paper is the following: the authors want to show that there is a way to construct a general (partial) approximation space with disjoint base sets relying on the properties of our objects. These spaces are very useful in data mining. At the very beginning we have a general (partial) approximation space $\langle U, \mathfrak{B} \rangle$. A base set is a collection of objects with the same (practically useful) property. Common properties represent similarity between objects, and the generated tolerance relation can be used to define the system of disjoint base sets with the help of correlation clustering. The final general (partial) Pawlakian approximation space is the core notion of similarity based on rough set theory. This space 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 we have to use an acceptable number of base sets);
- the size of base sets is not too small, or too big.

At first the authors overview the most important points of correlation clustering, and then they deal with how to apply correlation clustering in rough set theory. At the end an implementation of similarity based rough set theory is showed.

## 2 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 we can still say something about their similarity or dissimilarity. Think of the humans for example. It is hard to detail someone's looks by a number, but we still make 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 we want to formulate the similarity and dissimilarity by using mathematics, we need a tolerance relation. If this relation holds for two objects, we can say that they are similar. If this relation does not hold, we say that 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 we cannot go any further, e.g. the transitivity does not hold necessarily.

If we take a human and a mouse, 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 (see in [5,6,17]).

Our task is to find an $R \subseteq V \times V$ equivalence relation *closest* to the tolerance relation. A (partial) tolerance relation $\mathcal{R}$ (see in [10,15]) 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 we have an arbitrary relation $R \subseteq V \times V$ we have two sets of pairs. Let $R_{true}$ be the set of those pairs of elements for which the $R$ holds, and $R_{false}$ be the one for which $R$ does not hold. If $R$ is partial then $R_{true} \cup R_{false} \subseteq V \times V$. If $R$ is total then $R_{true} \cup R_{false} = V \times V$.

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 $\delta$ is the Knockecker delta symbol (see [11]). 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 $R$ we call the value $f$ 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 similarity relation. Take the relation on the left of Fig. 5. The dashed line denotes dissimilarity and the normal line similarity. On the right, Fig. 5 shows all the partition of these objects, where rectangles indicate the clusters. The thick lines denote the pairs which are counted in the cost function. In the upper row the value of the cost function is 1 (in each case), while in the two other cases it is 2 and 3, respectively.

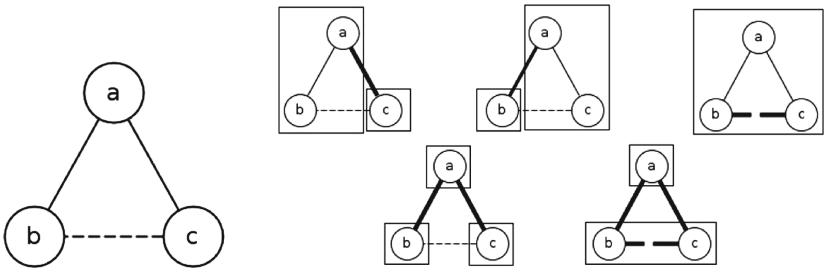



**Fig. 5.** Minimal frustrated similarity graph and its partitions

The number of partition can be given by the Bell number (see [1]), which grows exponentially. Hence, in general — even in the case of some dozens of objects — the optimal partition cannot be determined in reasonable time, thus a search algorithm which 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.

## 3 Correlation Clustering in Rough Set Theory

When we would like to define the base sets we use the background knowledge embedded in a given information system. If we have a Pawlakian system then we call two objects indiscernible if all of their known attribute values are the same. In many cases covering systems rely on a similarity (tolerance) relation. As we mentioned earlier some problems can come up using these covering systems. A base set contains members which are similar to a distinguished member.

This means that covering does not consider the similarity relation itself but the similarity with respect to a distinguished object. As a result of the correlation clustering based on the tolerance relation we obtain a partition of the universe [2–4]. The clusters contain elements which are usually similar to each other (not just to a distinguished member). So the partition can be understood as a system of base sets. Singleton base sets represent very little information (its member is only similar to itself). Without increasing the number of conflicts we cannot consider its member similar to any objects. By deleting singleton base sets we get a partial system of base sets.

## 4 Program

The authors of this article wrote a software, which represents the theory in real life problems. The software can be downloaded from:
https://arato.inf.unideb.hu/aszalos.laszlo/covering/.

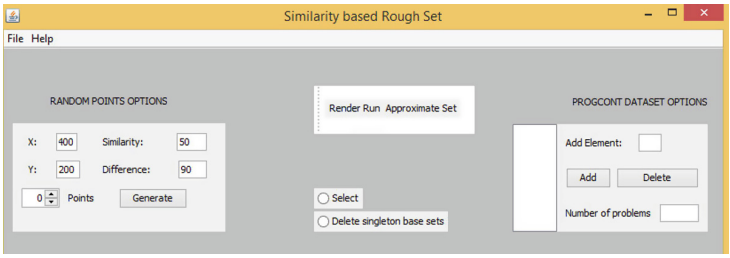



**Fig. 6.** Graphical user interface

Figure 6 illustrates the graphical user interface of the program. For giving the input datasets we have two options.

1. Generating random points
2. Reading a predetermined formatted dataset

1. *Random points*
   At first 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 (These options can be given on the left panel of the user interface). The base of the tolerance relation is the Euclidean distance of the objects ($d$). We defined a similarity ($S$) and a dissimilarity threshold ($D$). The tolerance relation $\mathcal{R}$ can be given this way for any objects $A, B$:

$$A\mathcal{R}B = \begin{cases} +1 & d(A,B) \leq S \\ -1 & d(A,B) > D \\ 0 & \text{otherwise} \end{cases}$$

2. *Predetermined formatted dataset*
   The so called ProgCont system (see in [9]), which was developed at the Faculty of Informatics at the University of Debrecen evaluates the programming competitions and midterms. Our software can read and handle data, generated by the ProgCont system. Each record consists of the following attributes: competitor id, problem id, solution id and the id of the programming language. Let $A, B$ be two arbitrary competitors. Let $S_A$ and $S_B$ the sets of the solutions of the problems made by the competitors $A$ and $B$. So the tolerance relation $\mathcal{R}$ for any competitors $A, B$ is the following:

$$A\mathcal{R}B = \begin{cases} +1 & |S_A \Delta S_B| \leq S \\ -1 & |S_A \Delta S_B| \geq D \\ 0 & \text{otherwise} \end{cases}$$

   The similarity and difference are defined by the cardinality of the symmetric difference ($\Delta$) of the given sets.

**Algorithm 1.** Run method

```
1: procedure RUN(N)
2:     best_partition ← FindBestPartition(N)
3:     covering_base_sets ← GetCovering()
4:     disjoint_covering_base_sets ← MakeDisjointSets(covering_base_sets)
5:     print best_partition
6:     print covering_base_sets
7:     print disjoint_covering_base_sets
8: end procedure
```

So in our program two competitors are similar to each other, if among the same solutions there is a difference less than or equal to $S$, and they are treated as different if this difference is greater than $D$. They are neutral otherwise. In our algorithm we used 1 as $S$. We thought that if two persons have only one different solution, then it does not imply that they have different knowledge. The $D$ threshold was set to 3.

After reading/generating the data, the software finds the quasi optimal partition (see Algorithm 1). Whereas numerous algorithms can be used for finding the optimal clustering, we used a genetic search algorithm (see Algorithm 2 in [8]). This algorithm is simple, it can be easily implemented, and it gives a relatively optimal solution for the correlation clustering's problem.

The set to be approximated can also be defined in two ways. The user can select the points of this set manually, or if we have a ProgCont dataset, then we can give IDs of the problems. The algorithm managing the approximation checks which competitors solved the given problems, and adds the points representing them to the set.

**Algorithm 2.** Genetic algorithm

```
1: function FIND BEST PARTITION(N)
2:     population ← random_population
3:     while exit condition false do
4:         sort(population)
5:         for i ← 1, N do
6:             new_population.add(population.get(i))
7:         end for
8:         p1 ← select_parents()
9:         p2 ← select_parents()
10:        children ← crossover(p1, p2)
11:        if small probability then
12:            mutation(children)
13:        end if
14:        new_population.add(children)
15:        population ← new_population
16:        max ← find_max(population)
17:    end while
18:    return max
19: end function
```

As mentioned in the previous sections the singleton base sets hold little information about the similarity. In our software there is an option to throw the singleton base sets away. The base sets, we got this way, are partial because their union does not cover the universe.

## 5 Results

The execution time of the algorithm managing the set-approximation can be seen in Fig. 7. The axis $x$ represents the number of points, and the axis $y$ represents the execution time in milliseconds.

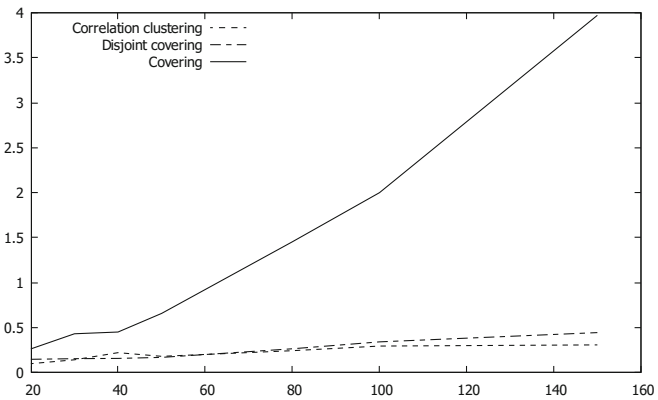



**Fig. 7.** Execution time

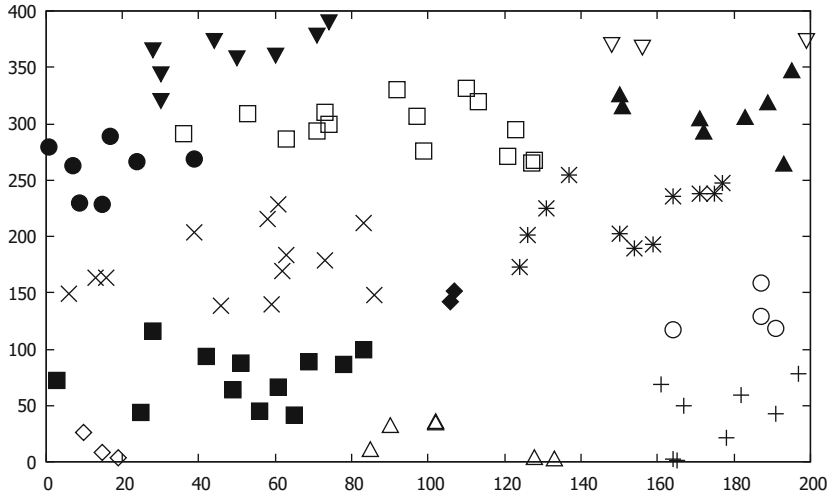



**Fig. 8.** The clusters

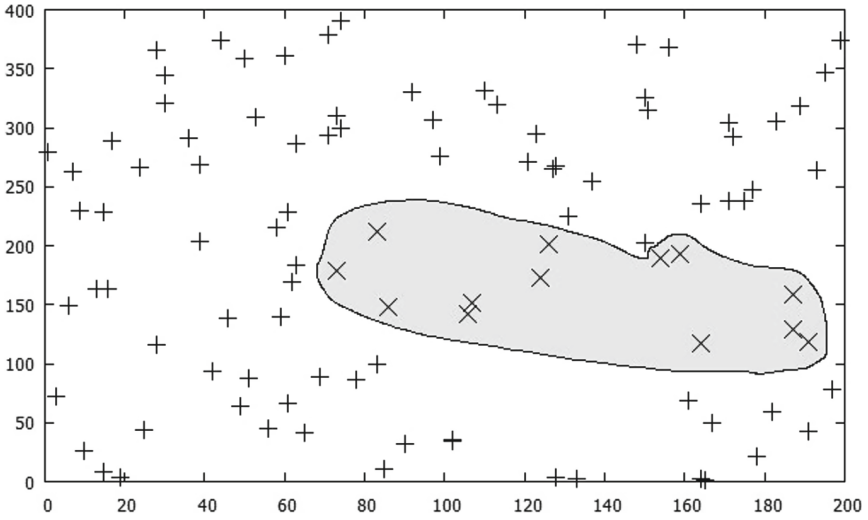
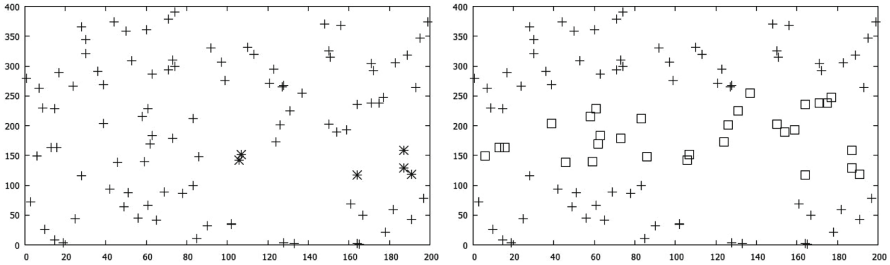
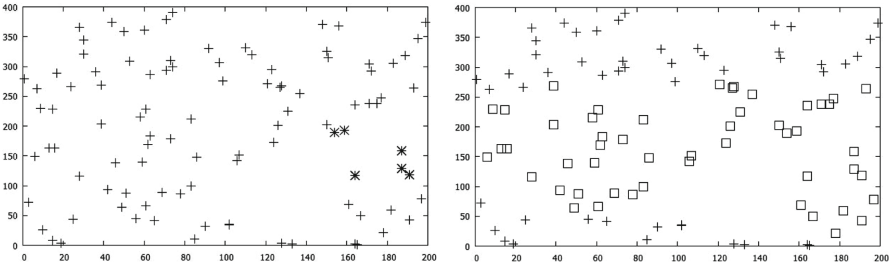



**Fig. 9.** The set to be approximated

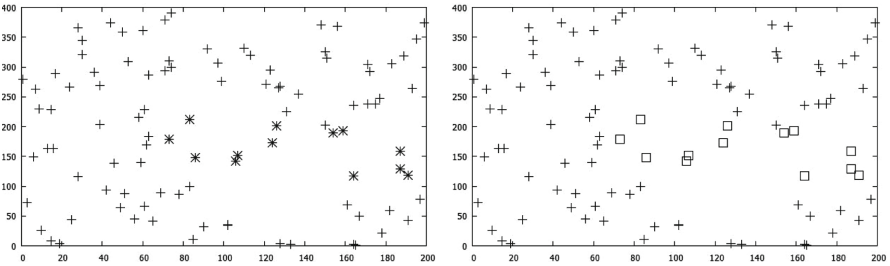
If we take a look at the figure we can see that the approximation by covering is the slowest. This was expected, because there are a lot of base sets to work with. Between the disjoint covering and the correlation clustering there is no significant difference. Nevertheless, as the number of points increases, the correlation clustering gives the fastest way to approximate. It is an interesting fact that there is such a great difference between the covering and its disjoint variant. Despite the fact that a disjoint covering has the largest number of base



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



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



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

**Fig. 10.** The outputs of the approximations by the software

sets, their cardinality is much less (most of them are singleton) than in the case of a regular covering.

The following figures show the output of our software for 100 random points. The similarity threshold $S$ was set to 50, and $D$ was set to 90. Figure 8 represents the clusters (base sets) created by the correlation clustering. The set to be approximated is shown in Fig. 9. The members of this set are denoted by the

× 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. 10A. 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 is shown in Fig. 10B. 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 we can see a significant difference. The upper approximation defined by covering contains much more objects, almost twice as much as the one defined by correlation clustering.

The approximation generated by the disjoint covering is shown in Fig. 10C. We can see that among the methods this generated the finest approximation (lower and upper approximation coincide). The reason is that almost all base sets are singleton. As mentioned before if we have only singleton base sets we get the common set theory back.

## 6 Conclusion and Future Work

The authors introduced a new method to define base sets in a general approximation space. The most important novelty of the introduced method is the usage of similarity relation of objects. It emphasizes and so relies on the similarity of objects 'in general' (and not on the similarity to a given object). Correlation clustering is a possible way to define a system of disjoint base sets corresponding to a given similarity 'in general'. There are many different algorithms of correlation clustering. In the application presented in the paper the authors used a genetic algorithm. It worked well, but in the near future other algorithms have to be checked, and a comparative (empirical and theoretical) study seems to be very important in order to determine the properties of different algorithms. Relying on the results the whole method can be useful in data mining and deep learning.

## References

1. Aigner, M.: Enumeration via ballot numbers. Discret. Math. **308**(12), 2544–2563 (2008). http://www.sciencedirect.com/science/article/pii/S0012365X07004542
2. Aszalós, L., Mihálydeák, T.: Rough clustering generated by correlation clustering. In: Ciucci, D., Inuiguchi, M., Yao, Y., Ślęzak, D., Wang, G. (eds.) RSFDGrC 2013. LNCS, vol. 8170, pp. 315–324. Springer, Heidelberg (2013). doi:10.1007/978-3-642-41218-9_34. http://dx.doi.org/10.1109/TKDE.2007.1061
3. Aszalós, L., Mihálydeák, T.: Rough classification based on correlation clustering. In: Miao, D., Pedrycz, W., Ślęzak, D., Peters, G., Hu, Q., Wang, R. (eds.) RSKT 2014. LNCS, vol. 8818, pp. 399–410. Springer, Cham (2014). doi:10.1007/978-3-319-11740-9_37

4. Aszalós, L., Mihálydeák, T.: Correlation clustering by contraction. In: 2015 Federated Conference on Computer Science and Information Systems (FedCSIS), pp. 425–434. IEEE (2015)
5. Bansal, N., Blum, A., Chawla, S.: Correlation clustering. Mach. Learn. **56**(1–3), 89–113 (2004)
6. Becker, H.: A survey of correlation clustering. In: Advanced Topics in Computational Learning Theory, pp. 1–10 (2005)
7. Ciucci, D., Mihálydeák, T., Csajbók, Z.E.: On definability and approximations in partial approximation spaces. In: Miao, D., Pedrycz, W., Ślęzak, D., Peters, G., Hu, Q., Wang, R. (eds.) RSKT 2014. LNCS, vol. 8818, pp. 15–26. Springer, Cham (2014). doi:10.1007/978-3-319-11740-9_2
8. Goldberg, D.E., Holland, J.H.: Genetic algorithms and machine learning. Mach. Learn. **3**(2), 95–99 (1988). http://dx.doi.org/10.1023/A:1022602019183
9. Kádek, T., Kósa, M., Pánovics, J.: Experiences of programming competitions supported by the ProgCont system (in Hungarian). In: New Technologies in Science, Research and Education, pp. 152–157 (2012)
10. Mani, A.: Choice inclusive general rough semantics. Inf. Sci. **181**(6), 1097–1115 (2011)
11. Néda, Z., Sumi, R., Ercsey-Ravasz, M., Varga, M., Molnár, B., Cseh, G.: Correlation clustering on networks. J. Phys. A: Math. Theor. **42**(34), 345003 (2009). http://www.journalogy.net/Publication/18892707/correlation-clustering-on-networks
12. Pawlak, Z.: Rough sets. Int. J. Parallel Prog. **11**(5), 341–356 (1982)
13. Pawlak, Z., Skowron, A.: Rudiments of rough sets. Inf. Sci. **177**(1), 3–27 (2007)
14. Pawlak, Z., et al.: Rough Sets: Theoretical Aspects of Reasoning About Data. System Theory Knowledge Engineering and Problem Solving, vol. 9. Kluwer Academic Publishers, Dordrecht (1991)
15. Skowron, A., Stepaniuk, J.: Tolerance approximation spaces. Fundamenta Informaticae **27**(2), 245–253 (1996)
16. Yao, Y., Yao, B.: Covering based rough set approximations. Inf. Sci. **200**, 91–107 (2012). http://www.sciencedirect.com/science/article/pii/S0020025512001934
17. Zimek, A.: Correlation clustering. ACM SIGKDD Explor. Newsl. **11**(1), 53–54 (2009)