Acta Cybernetica **24** (2019) 105–120.

Different Types of Search Algorithms for Rough Sets*

Dávid Nagy[a], Tamás Mihálydeák[b], and László Aszalós[c]

Abstract

Based on the available information in many cases, it can happen that two objects cannot be distinguished. If a set of data is given and in this set two objects have the same attribute values, then these two objects are called indiscernible. This indiscernibility has an effect on the membership relation because in some cases it makes our judgment uncertain about a given object. The uncertainty appears because if something about an object needs to be stated, then all the objects that are indiscernible from the given object must be taken into consideration. The indiscernibility relation is an equivalence relation which represents the background knowledge embedded in an information system. In a Pawlakian system this relation is used in set approximation. Correlation clustering is a clustering technique which generates a partition. In the authors' previous research, (see in [10, 11, 9]) the possible usage of correlation clustering in rough set theory was investigated. In this paper, the authors show how different types of search algorithms can affect the set approximation.

Keywords: rough set theory, set approximation, data mining

1 Introduction

Pawlak's indiscernibility relation (which is an equivalence relation) represents a limit of our knowledge embedded in an information system. This relation defines the base sets. They contain objects that are indiscernible from each other. In many applications, it is common to replace the equivalence relation with tolerance relation. In our previous study, we examined whether the clusters, generated by correlation clustering, can be understood as a system of base sets. Correlation clustering is a clustering method in data mining which creates a partition based on a tolerance relation. The groups, defined by this partition, contain similar

*This work was supported by the National Research, Development and Innovation Office of Hungary under Grant No. TÉT 16-1-2016-0193.

[a]Faculty of Informatics, University of Debrecen, E-mail: {nagy.david}@inf.unideb.hu

[b]Faculty of Informatics, University of Debrecen, E-mail: {mihalydeak.tamas}@inf.unideb.hu

[c]Faculty of Informatics, University of Debrecen, E-mail: {aszalos.laszlo}@inf.unideb.hu

DOI: 10.14232/actacyb.24.1.2019.8

objects. In our previous papers, we showed that it is worth to generate the system of base sets from the partition. This way, the base sets contain objects that are typically similar to each other and they are pairwise disjoint. To find the partition in reasonable time, search algorithms must be used. So the system of base sets is highly dependent on the used search algorithm. The structure of the paper is the following: A theoretical background about the classical rough set theory comes first. In section 3 we define correlation clustering mathematically. In section 4 we present our previous work. In section 5 we present the search algorithm used in our experiments. In section 6 we show a way to compare the algorithms. Finally we conclude the results.

2 Theoretical Background

In practice, a set is a collection of objects that are similar in some sense. A set is uniquely identified by its members. It means that if we would like to decide, whether an object belongs to this set, then we can give a precise answer which is yes or no. For instance, the set of even numbers is a crisp set because it can be decided if an arbitrary number is even or odd. However, in some computer science applications, researchers are interested in vague concepts. The notion of a brave warrior is vague because we cannot give two disjoint classes: brave and not brave warriors. One can consider a person brave due to their actions, but maybe someone else would consider this person as not brave. So bravery is a vague concept.

Rough set theory was proposed by professor Pawlak in 1982 (see in [12]). The theory offers a way to handle vague concepts. 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 distinguished. The indiscernibility relation generated in this way is the mathematical basis of rough set theory.

If we want to decide, whether an object belongs to an arbitrary set, based on the available data, then our decision affects the decision about all the objects that are indiscernible from the given object.

In this case, if we would like to check, whether an object is in an arbitrary set, then the following three possibilities appear:

- it is sure that the object is in the set if all the objects, that are indiscernible from the given object, are in the set;
- the object may be in the set if there some objects that are in the set and are indiscernible from the given object;
- it is sure that the object is not in the set if all the objects, that are indiscernible from the given object, are not in the set.

So the indiscernibility makes a set vague.

From the theoretical point of view, a Pawlakian approximation space (see in [12, 13, 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* l, 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 $B \cap S \neq \emptyset\}$.

U is the set of objects. $\mathfrak{B}$ is the system of base sets which represents the background knowledge. $\mathfrak{D}_{\mathfrak{B}}$ is the set of definable sets which defines how the base sets can be used in the set approximation. 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. The set $BN(S) = u(S) \setminus l(S)$ is called the boundary region of the set S. If $BN(S) = \emptyset$ then S is crisp, otherwise it is rough.

Table 1 shows a very simple table 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:

$$\mathfrak{B} = \{\{u_1\}, \{u_2\}, \{u_3\}, \{u_4\}, \{u_5, u_7\}, \{u_6, u_8\}\}$$

Let us suppose that based on some background knowledge we know that the patients u_1, u_2 and u_5 have the flu. Let S be the following set of these patients: $\{u_1, u_2, u_5\}$. The approximation of the set S is the following:

- $l(S) = \{\{u_1\}; \{u_2\}\}$
- $u(S) = \{\{u_1\}; \{u_2\}; \{u_5, u_7\}\}$
- $BN(S) = \{\{u_5, u_7\}\}$

Here, $l(S)$ contains those patients that surely have the flu. Patient u_5 is not in the lower approximation because there is one other patient, u_7, who is indiscernible from u_5, and we do not have information about, whether u_7 has the flu or not. So the base set $\{u_5, u_7\}$ can only be in the upper approximation.

Table 1: Information System

Object	Headache	Body Temp.	Muscle Pain
u_1	YES	Normal	NO
u_2	YES	High	YES
u_3	YES	Very high	YES
u_4	NO	Normal	NO
u_5	NO	High	YES
u_6	NO	Very high	YES
u_7	NO	High	YES
u_8	NO	Very High	YES

3 Correlation Clustering

Data mining is the process of discovering patterns and hidden information in large data sets. The goal of a data mining process is to extract information from a data set and transform it into an understandable structure for further use. Clustering is a data mining technique in which the goal is to group objects, so that the objects in the same group are more similar to each other than to those in other groups. In many cases, the similarity is based on the attribute values of the objects. In most of them, some kind of distance is used to define the similarity. However, sometimes only nominal data are given. In this particular case, distance can be meaningless. For example, what is the distance between a male and a female? In this case, a similarity relation can be used which is a tolerance relation. If this relation holds for two objects, we can say that they are similar. If this relation does not hold, then they are dissimilar. It is easy to prove that this relation is reflexive and symmetric. The transitivity; however, does not hold necessarily. Correlation clustering is a clustering technique based on a tolerance relation (see in [3, 4, 17]).

Let V a set of objects and T the similarity relation. The task is to find an $R \subseteq V \times V$ equivalence relation which is *closest* to the tolerance relation.

A (partial) tolerance relation T (see in [15, 8]) can be represented by a matrix M. Let matrix $M = (m_{ij})$ be the matrix of the partial relation T of similarity:

$$m_{ij} = \begin{cases} 1 & i \text{ and } j \text{ are similar} \\ -1 & i \text{ and } j \text{ are different} \\ 0 & \text{otherwise} \end{cases}$$

A relation is called partial if there exist two elements (i, j) such that $m_{ij} = 0$. It means that if we have an arbitrary relation $R \subset V \times V$ 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 $R_{true} \cup R_{false} \subset V \times V$. 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 $p(x) = p(y)$. We treat the following two cases conflicts for any $x, y \in V$:

- $(x, y) \in T$ but $p(x) \neq p(y)$
- $(x, y) \notin T$ but $p(x) = p(y)$

The goal is to minimize the number of these conflicts. If their number is 0, the partition is called *perfect*. Given the T and R, we call the number of conflicts 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.

The number of partitions can be given by the Bell number (see in [1]) which grows exponentially. For more than 15 objects, we cannot achieve the optimal partition by exhaustive search in reasonable time. In a practical case, a search algorithm can be used which can give a quasi-optimal partition.

4 Similarity based rough sets

In practical applications, indiscernibility relation is too strong. Therefore, Pawlakian approximation spaces have been generalized using tolerance relations (symmetric and reflexive) which are similarity relations. Covering-based approximation spaces (see [16]) generalize Pawlakian approximation spaces in two points:

1. $\mathcal{R}$ is 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 $[x] = \{y \mid y \in U, x\mathcal{R}y\}$.

The definitions of definable sets and approximation pairs are the same as before. In these covering systems, each object generates a base set.

Correlation clustering defines a partition. The clusters contain objects that are typically similar to each other. In our previous work ([10]), we showed that this partition can be understood as the system of base sets which results in a completely new approximation space. The approximation space also has several good properties. The most important one is that it focuses on the similarity (the tolerance relation) itself, and it is different from the covering type approximation space relying on the tolerance relation.

Singleton clusters represent very little information because the system could not consider its member similar to any other objects without increasing the number of conflicts. As they mean little information, we can leave them out. If we do not consider the singleton clusters, then we can generate a partial system of base sets from this partition where singleton clusters are not base sets (see in [11, 9]).

In reasonable time, correlation clustering can only be solved by using search algorithms. However, each algorithm can provide different clusters. So the system of base 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, it is crucial to use the best possible search algorithm.

5 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 we have more than 15 objects in a general case we can only provide an approximate solution. That is why this problem got 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 [2]. According to the students' request the system was written in Java. Jason Brownlee published a similar book using Ruby [5].

The following list shows the used algorithms in our experiments. Each of them can be downloaded from [2]:

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

In the next subsections, there are some brief information about the used algorithms and their parameters. The whole descriptions can also be seen in [2].

5.1 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.

5.2 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.

5.3 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 our experiments, we used a short-term memory with the 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 1000th step and it returns the best state.

5.4 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 were 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%.

5.5 Parallel Tempering

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

5.6 Genetic Algorithm

In this algorithm, each partition is represented by an entity. In each search step, there is a population of entities with the size of 100. In the beginning, each entity represents a random partition. This population contains the actual generation of entities and 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 this paper, we have used one-point crossover. 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 the step 1. The algorithm stops when it reaches the 2000th generation and returns the best entity of the population.

5.7 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 our 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 our 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.

5.8 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.

5.9 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 which is represented by the brightest firefly. The number of fireflies was set to 50.

6 Comparing Algorithms

To compare the algorithms, we calculated the following values:

- Number of singleton clusters
- Standard deviation of the base set sizes
- Interquartile range of the base set sizes
- Execution time of the algorithm

In this paper, the authors refer to the cardinality of a base set as its size. As previously mentioned, singleton clusters mean little information. The greater their number is, the more unclear our knowledge becomes. For a search algorithm, the

most optimal is, if it provides the least number of these clusters in order to have a precise knowledge 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 interquartile 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.

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

For comparing the parameters, the same graph (with 100 points, q = 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 which minimized the above mentioned 4 values. If the differences between the standard deviations, the interquartile ranges and the numbers of singletons were not significant, then the judgment was made by the execution time.

7 Results

7.1 Erdős-Rényi graphs

Algorithm 1 Erdős-Rényi random graph generating method

Procedure ER(N, p)

```
for i = 1, ..., N do
  for j = 1, ..., N do
    Generate a random x value between 0 and 1
    if x < p then
      There is an edge between objects i and j
    end if
  end for
end for
```

In this part our experiments, we used Erdős-Rényi graphs (see in [7, 6]). This random graph generating method is very simple. Its pseudo-code can be seen in Algorithm 1. In our experiments, we 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. The graphs are only used for defining a similarity relation. Any other kinds of graphs can be used. 100, 200, 300 and 400 points 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 6, were determined. The results can

Table 2: 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

be seen in the following tables and can be downloaded from the following link: `https://bit.ly/2sOqMA6`.

From Table 2 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 3 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 the algorithms provided twice or three times as large values as the other 3 which is not suitable.

In Table 4 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 5 the average run-time of the algorithms can be seen in seconds. The values here vary the most. It is obvious that 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 basically pointless to be used.

7.2 Random two-dimensional points

In this part of our experiments, random two-dimensional points were generated. The base of the tolerance relation was the Euclidean distance of these objects (d). We defined a similarity (S) and a dissimilarity threshold (D). S was set to 50 and

Table 3: Average interquartile 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 4: 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 5: 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

D was set to 90. The tolerance relation $\mathcal{R}$ can be given this way for any objects A, B:

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

We generated 100, 150, 200, 300, 500 points and each algorithm was run 3 times for each point set and calculated the averages of the values described in section 6. In the following tables, we can see the results.

In Table 6 the average standard deviations of the base set sizes can be seen. In case 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. It is interesting that 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 7 shows the interquartile 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 8 shows how many singleton clusters appeared. The results were quite the same in all cases. In 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

Table 6: 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 7: Average interquartile 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

a rather inadequate result compared to the others. However, this difference was not so high.

In Table 9 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

Table 8: 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

Table 9: 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

points.

8 Conclusion

In [10] the authors introduced a partial approximation space relying on a similarity relation (a tolerance relation technically). The genuine novelty of approximation spaces is the systems of base sets: it is the result of correlation clustering, and so similarity is taken into consideration generally. Singleton clusters have no real information in approximation process, these clusters cannot be taken as base sets, therefore the approximation spaces are partial in general cases (the unions of base sets are proper subsets of the universes.) The partition, and so the system of base sets, gained from correlation clustering depends on the used search algorithm. In the present paper, we used several algorithms and we showed a way to compare them. In our experiments, we used two different types of random graphs. For these types of graphs, the simulated annealing proved to be 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 reasonable time even for large amounts of points.

References

[1] Aigner, Martin. Enumeration via ballot numbers. *Discrete Mathematics*, 308(12):2544 – 2563, 2008. DOI: 10.1016/j.disc.2007.06.012.

[2] Aszalós, László and Mária, Bakó. *Advanced Search Methods.* Educatio Társadalmi Szolgáltató Nonprofit Kft., 2012. in Hungarian.

[3] Bansal, Nikhil, Blum, Avrim, and Chawla, Shuchi. Correlation clustering. *Machine Learning*, 56(1-3):89–113, 2004.

[4] Becker, Hila. A survey of correlation clustering. *Advanced Topics in Computational Learning Theory*, pages 1–10, 2005.

[5] Brownlee, Jason. *Clever Algorithms: Nature-Inspired Programming Recipes*. Lulu.com, 1st edition, 2011.

[6] Erdös, P. and Rényi, A. On random graphs i. *Publicationes Mathematicae Debrecen*, 6:290, 1959.

[7] Erdös, P. and Rényi, A. On the evolution of random graphs. In *Publication of the Mathematical Institute of the Hungarian Academy of Sciences*, pages 17–61, 1960.

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

[9] Mihálydeák, Tamás. Logic on similarity based rough sets. In Nguyen, Hung Son, Ha, Quang-Thuy, Li, Tianrui, and Przybyła-Kasperek, Małgorzata, editors, *Rough Sets*, pages 270–283, Cham, 2018. Springer International Publishing.

[10] Nagy, Dávid, Mihálydeák, Tamás, and Aszalós, László. 10.1007/978-3-319-60840-2_7, *Similarity Based Rough Sets*, pages 94–107. Springer International Publishing, Cham, 2017.

[11] Nagy, Dávid, Mihálydeák, Tamás, and Aszalós, László. Similarity based rough sets with annotation. In Nguyen, Hung Son, Ha, Quang-Thuy, Li, Tianrui, and Przybyła-Kasperek, Małgorzata, editors, *Rough Sets*, pages 88–100, Cham, 2018. Springer International Publishing.

[12] Pawlak, Zdzisław. Rough sets. *International Journal of Parallel Programming*, 11(5):341–356, 1982.

[13] Pawlak, Zdzislaw et al. Rough sets: Theoretical aspects of reasoning about data. *Systern Theory, Knowledge Engineering and Problem Solving, Kluwer Academic Publishers, Dordrecht, 199l*, 9, 1991.

[14] Pawlak, Zdzisław and Skowron, Andrzej. Rudiments of rough sets. *Information sciences*, 177(1):3–27, 2007.

[15] Skowron, Andrzej and Stepaniuk, Jaroslaw. Tolerance approximation spaces. *Fundamenta Informaticae*, 27(2):245–253, 1996.

[16] Yao, Yiyu and Yao, Bingxue. Covering based rough set approximations. *Information Sciences*, 200:91 – 107, 2012. DOI: `http://dx.doi.org/10.1016/j.ins.2012.02.065`.

[17] Zimek, Arthur. Correlation clustering. *ACM SIGKDD Explorations Newsletter*, 11(1):53–54, 2009.