Int. J. Appl. Math. Comput. Sci., 2023, Vol. 33, No. 2, 297–312
DOI: 10.34768/amcs-2023-0022

GrDBSCAN: A GRANULAR DENSITY–BASED CLUSTERING ALGORITHM

DAWID SUCHY [a], KRZYSZTOF SIMINSKI [a,*]

[a]Department of Algorithmics and Software
Silesian University of Technology
ul. Akademicka 16, 44-100 Gliwice, Poland
e-mail: krzysztof.siminski@polsl.pl

Density-based spatial clustering of applications with noise (DBSCAN) is a commonly known and used algorithm for data clustering. It applies a density-based approach and can produce clusters of any shape. However, it has a drawback—its worst-case computational complexity is $O(n^2)$ with regard to the number of data items n. The paper presents GrDBSCAN: a granular modification of DBSCAN with reduced complexity. The proposed GrDBSCAN first granulates data into fuzzy granules and then runs density-based clustering on the resulting granules. The complexity of GrDBSCAN is linear with regard to the input data size and higher only for the number of granules. That number is, however, a parameter of the GrDBSCAN algorithm and is (significantly) lower than that of input data items. This results in shorter clustering time than in the case of DBSCAN. The paper is accompanied by numerical experiments. The implementation of GrDBSCAN is freely available from a public repository.

Keywords: granular computing, DBSCAN, clustering, GrDBSCAN.

1. Introduction

Clustering (cluster analysis) is a very important technique in data analysis and machine learning. The task aims at gathering objects into a collection in such a way that items in one cluster (group) are more similar to each other than to any item in other clusters. In very general terms, data in the same cluster are similar and data in different clusters are dissimilar. This very general rule results in a plethora of clustering algorithms. Below we shortly describe the main classes of clustering algorithms.

Density-based clustering algorithms discover regions with higher density of items. Algorithms in this class can produce arbitrarily shaped clusters. Items in regions of lower density are often classified as noise. In data sets with highly overlapping clusters, the algorithms do not always identify cluster borders correctly. Some influential algorithms in this group are DBSCAN (Ester *et al*., 1996), GDBSCAN (Sander *et al*., 1998), DENCLUE (Hinneburg and Keim, 1998), or OPTICS (Ankerst *et al*., 1999).

Grid-based clustering algorithms first split the input domain into a grid of orthogonal hyperboxes and then analyse each hyperbox. They are commonly applied to high-dimensional problems. Some of them can identify the importance of attributes in multi-dimensional spaces (subspace clustering). Important representatives are CLIQUE (Agrawal *et al*., 1998), ENCLUS (Cheng *et al*., 1999), and STING (Wang *et al*., 1997).

An important class of clustering algorithms is one based on minimisation of performance criteria. The value of the criterion drops if similar items are assigned to the same cluster. The properties of the produced clusters heavily depend on the criterion applied. The function is minimised iteratively. There is no guarantee that the resulting clusters are the optimal ones. Commonly the algorithms take the number of clusters to produce as a parameter. Notable algorithms in this family are fuzzy C-means (FCM) (Dunn, 1973), fuzzy C-medians (FCMed) (Jajuga, 1991), Gustafson–Kessel (Gustafson and Kessel, 1978), possibilistic FCM (Krishnapuram and Keller, 1993), FCOM (Leski, 2016), and RSFCM (Siminski, 2014).

Hierarchical algorithms start with a set of one-element clusters that are then merged into more complex ones. The merger depends on the distance between clusters to be merged. The process of clustering is graphically represented as a dendrogram.

*Corresponding author

Important representatives are SLINK (single linkage) (Sibson, 1973), complete linkage (Defays, 1977; Wu *et al*., 2021), CURE (Guha *et al*., 2001), and CHAMELEON (Karypis *et al*., 1999).

Biclustering algorithms use an interesting technique. They are clustered simultaneously. Objects and attributes are treated in the same way, and they can be transposed. Biclustering is used in (but is not limited to) bioinformatics. Some examples of biclustering algorithms are HroBi (Michalak and Stawarz, 2013), eBi (Stawarz and Michalak, 2012), or FuBi (Siminski, 2022a).

The above list of clustering algorithms is not exhaustive. More algorithms have been proposed, such as spectral (Shi and Malik, 2000), gravitational (Wright, 1977; Junlin and Hongguang, 2011) graph based (Hartuv and Shamir, 2000), and more.

The paper focuses on the DBSCAN algorithm. It is a density-based algorithm that can produce clusters (groups) of any shape. Unfortunately, the worst-case complexity of DBSCAN is $O(n^2)$ for a data set with n data items (for details of the complexity issue, see Section 3.1.2). In the paper, the granular computing paradigm is applied to reduce the complexity.

The paper is organized as follows. In Section 2 we briefly introduce the granular computing paradigm. In Section 3 we describe DBSCAN and GrDBSCAN––the new granular density-based clustering algorithm. In Section 4 we present experiments on GrDBSCAN.

2. Granular computing

Granular computing (GrC) is simultaneously an old and a new paradigm. It was introduced by Zadeh (1979) more than 40 years ago. However, for many years it seemed to hibernate and evoked a very sparse scientific response. The idea stated by Lotfi Zadeh was very innovative and reaching far into the future, whereas techniques, methods, and algorithms were scarce. This is why granular computing did not develop from the start. A few decades later, after many techniques, methods, paradigms, and algorithms had been developed and data had exploded in size, granular computing was revived and made huge progress (Yao *et al*., 2013). Now it is a new emerging field of research and study in data mining (Yao, 2007).

Zadeh (1997) enumerates three concepts of human cognition addressed by granular computing: decomposition of a whole into parts (granulation), integration of parts into a whole (organisation), and the cause–effect relation (causation).

Granular computing is an umbrella term and generalisation (Salehi *et al*., 2015) for various techniques, algorithms, and models. It is simultaneously a starting point for new concepts like computation with words (Zadeh, 2002). Granular computing is experiencing its revival nowadays when a plethora of algorithms, techniques, and models are at our disposal. They have a common denominator: granularity.

Granular computing may be a starting point for a human-centric data approach, because it mimics human cognition (Pedrycz *et al*., 2015). Humans commonly gather data into granules and then operate on them. This enables zooming-in when more details are needed, or zooming-out when details are not necessary or obfuscate the problem and general issues are in focus.

Yao (2016) proposed a three-way approach to granular computing (the triarchic theory of granular computing) with three perspectives, each supporting the other two. The philosophical perspective focuses on structured thinking. It handles the meronym-holonym concept with meronymy responsible for parts and holonymy—for a whole. The direction from a whole to parts addresses analysis whereas the direction from parts to a whole—synthesis. The methodological perspective focuses on structured problem solving. It handles methods, techniques, algorithms, and tools used for multilevel problem solving. The computational perspective focuses on information processing. It addresses the way information is handled from the methodological perspective. Granular computing commonly starts with creation of granules from data (granulation) followed by computing with granules. The latter is still a huge challenge for researchers.

2.1. Granules. A data granule is a crucial notion in granular computing. Commonly, it is defined as a collection of entities in the sense of similarity, likeness, proximity, indiscernibility, identity, or adjacency (Pedrycz, 2013; Yao and Zhong, 2007; Yao, 2008; Siminski, 2022b; 2021b; Shifei *et al.*, 2010). Data granules have two very important properties that make them different from data clusters. The first one is clear semantics of data granules. They can be tagged with semantically rich labels (Bargiela and Pedrycz, 2006). The second property is a hierarchy of granules. A data granule represents an entity and is composed of some elements. The elements may be represented by lower level granules. This leads to the conclusion that a granule is composed of (sub)granules and simultaneously a granule is a part of a (super)granule. Granularity enables operating on various levels of detail (Keet, 2008). When a more general view is needed, we simply climb up a hierarchy of granules to more general ones. On the other hand, when details are needed, we climb down a hierarchy to address a problem with more specialised granules (Yao, 2018; Ciucci, 2016).

2.2. Representation of granules. Representation of granules is diverse. Common representations are

intervals, sets, fuzzy sets, interval type-2 fuzzy sets (Pedrycz *et al*., 2012), rough sets (Skowron *et al*., 2016), intuitionistic sets, shadowed sets, clusters (Siminski, 2020), soft sets (Xia *et al*., 2022), and if-then rules (Siminski, 2021a). A set is a simple representation of a collection of related items. Fuzzy sets enable partial membership, whereas rough sets address the case when no "yes-no decision" can be made. They are a starting point for the three-way decision approach (Yao, 2009; 2011; Siminski, 2023).

Commonly, the first step in granular computing is granulation of data. The aim of this step is to form semantically rich granules from experimental data. Techniques used heavily depend on the representation of granules. Granules based on fuzzy sets may be produced with clustering algorithms. Granules represented with intervals may be produced through the mean and standard deviation, a minimum–maximum pair, etc. Other common techniques are discretization, quantization, aggregation, or transformation (Yao, 2020). Data items may be aggregated with binary relations producing binary granules (Qian *et al.*, 2011; 2010).

One more operation needs mentioning: degranulation. It is a process that produces data items from granules. Data granules have their internal structure, the internal representation of data entities. It is very important to highlight that a data granule is not an archive, a zip, or a bag of data items (Siminski, 2021a). This is why degranulation does not always produce exactly the same data that had been granulated into granules. The difference is called the degranulation error.

3. GrDBSCAN: Granular DBSCAN

In this section, we present a new granular density-based algorithm—GrDBSCAN. First, we briefly describe the DBSCAN algorithm in Section 3.1. Thereafter, Section 3.2 presents the GrDBSCAN algorithm.

3.1. DBSCAN. Density-based spatial clustering of applications with noise (DBSCAN) is an algorithm proposed by Ester *et al*. (1996). The algorithm picks a random point that belongs to no cluster. Then DBSCAN finds all neighbours of the point in a hyperball with radius ε. If the number of data items within the ball is lower than $P_{\min}$, then the starting point is labelled as noise. If the number of neighbours is large enough, the point is a seed of a new cluster. All its neighbours that do not belong to any cluster are now elements of a new cluster A. Cluster A is then extended (if possible) by including of neighbours for each neighbour of the seed. If any neighbours exist, they are added to cluster A and the extension procedure is run for them subsequently. The pseudocode of DBSCAN is presented in Algorithm 1.

3.1.1. Properties of DBSCAN. A very important feature of DBSCAN is the form of produced clusters. There is no limitation on cluster shapes—DBSCAN can elaborate clusters of any shape. Furthermore, DBSCAN

Algorithm 1. DBSCAN procedure.

Require: $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_X\}$: dataset
Require: ε, $P_{\min}$: DBSCAN parameters
Require: δ: metric
$i \leftarrow 0$; {i-th cluster}
$\mathbb{N} \leftarrow \emptyset$; {noise is an empty set}
for all data item $\mathbf{x}$ in $\mathbb{X}$ **do**
  $\mathbb{H} \leftarrow$ findNeighbours $(\mathbb{X}, \delta, \mathbf{x}, \varepsilon)$; {see Alg. 2}
  if $|\mathbb{H}| < P_{\min}$ **then**
    $\mathbb{N} \leftarrow \mathbb{N} \cup \{\mathbf{x}\}$;{too few neighbours: $\mathbf{x}$ is noise}
  else
    $i \leftarrow i + 1$; {the next cluster, a new cluster}
    $c_i \leftarrow c_i \cup \{\mathbf{x}\}$; {add data item to the i-th cluster}
    $\mathbb{X} \leftarrow \mathbb{X} \setminus \{\mathbf{x}\}$; {remove data item from data set}
    for all data item h in $\mathbb{H}$ **do**
      $c_i \leftarrow c_i \cup \{h\}$; {add neighbour to the i-th cluster}
      $\mathbb{X} \leftarrow \mathbb{X} \setminus \{h\}$; {remove neighbour from data set}
      if h in $\mathbb{N}$ **then**
        $\mathbb{N} \leftarrow \mathbb{N} \setminus \{h\}$; {$h$ is not noise}
      else
        $\mathbb{M} \leftarrow$ findNeighbours $(\mathbb{X}, \delta, h, \varepsilon)$; {find neighbours of a neighbour, Algorithm 2}
        if $|\mathbb{M}| \geq P_{\min}$ **then**
          $\mathbb{H} \leftarrow \mathbb{H} \cup \mathbb{M}$; {add a neighbour's neighbours to neighbours of the seed item in the cluster}
        end if
      end if
    end for
  end if
end for

Algorithm 2. DBSCAN: findNeighbours procedure.

Require: $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_X\}$: dataset
Require: ε, $P_{\min}$: DBSCAN parameters
Require: δ: metric
Require: $\mathbf{x}_p$: point to find neighbours for
$\mathbb{H} \leftarrow \emptyset$; {empty set of neighbours}
for all data item $\mathbf{x}$ in $\mathbb{X}$ **do**
  if $\delta(\mathbf{x}_p, \mathbf{x}) \leq \varepsilon$ **then**
    $\mathbb{H} \leftarrow \mathbb{H} \cup \{\mathbf{x}\}$; {$\mathbf{x}$ is a neighbour of $\mathbf{x}_p$}
  end if
end for
return $\mathbb{H}$; {neighbours of $\mathbf{x}_p$}

does not need an input parameter for the number of output clusters—their number is determined automatically. The number is influenced by modification of DBSCAN parameters (ε, $P_{\min}$, and the distance function).

Sensitivity to the value of ε is a disadvantage of DBSCAN. The value of ε is not easy to set. It is commonly specified by an expert with good understanding of a data set to a cluster. DBSCAN is not fully deterministic. Results may differ if the order of processed data changes.

3.1.2. Computational complexity of DBSCAN. Generally, for each data item all neighbours have to be found. Therefore, the computational complexity is $O(n^2)$, where n is the number of data points to cluster. However, the authors of DBSCAN (Ester *et al*., 1996) claim that it is possible to implement DBSCAN with $O(n \log n)$ time complexity. Gunawan (2013) doubts the line of reasoning of Ester *et al*. (1996) and estimates the complexity of DBSCAN as $O(n^2)$. Gan and Tao (2015) discuss the complexity issue of DBSCAN and state that linarithmic $O(n \log n)$ complexity is a misclaim and the true complexity is $O(n^2)$. Gunawan (2013) claims that "there is no known algorithm that can compute exactly the same clustering as the original DBSCAN algorithm and whose running time is $O(n \log n)$ in the worst case". Schubert *et al*. (2017) doubt linearithmic $O(n \log n)$ complexity of implementation based on index structures, criticise mentioning it by secondary sources, and state that "[t]his runtime complexity is formally incorrect since there is no theoretical guarantee for the assumed runtime complexity of range queries, and it should not be repeated in this form". Gunawan (2013) proposes a DBSCAN with linarithmic complexity, but only for 2D datasets.

3.2. GrDBSCAN. This section presents a novel granular density-based clustering algorithm. The algorithm granulates numerical input data into fuzzy granules and then runs density-based clustering on them. We shortly discuss the representation of fuzzy granules, fuzzy distance between granules, and the neighbourhood used in the GrDBSCAN algorithm.

3.2.1. Representation of granules. The proposed clustering algorithm works with fuzzy granules. They are represented with Gaussian type-1 fuzzy sets. Each attribute of a granule is defined by a pair of values: core m and fuzzification σ of the attribute. Thus, a granule in a D-dimensional space is represented by a vector of D pairs: cores and fuzzifications. The membership function for attribute d is defined as

$$u_d\left(x_d\right) = \exp\Big(-\frac{\left(x_d - m_d\right)^2}{2\sigma_{d^2}}\Big), \tag{1}$$

where m_d stands for the core of a set for the d-th attribute and σ_d for its fuzzification.

3.2.2. Distance between granules. DBSCAN works with points and GrDBSCAN with fuzzy granules. In the DBSCAN algorithm, any metric can be used. In the implementation we employ, the Euclidean metric is applied, but it can be easily changed to another. In the GrDBSCAN, the distance is fuzzy in each orthogonal dimension. There are some attempts to define a fuzzy Euclidean distance between fuzzy sets. However, they are commonly based on α-cuts and produce approximate results or have high computational time (Diamond and Körner, 1997; Chakraborty and Chakraborty, 2006), or are limited to interval-value fuzzy sets (Luo and Cheng, 2015). This is why we define a new distance measure for the GrDBSCAN algorithm. The function $\delta : \mathbb{G} \times \mathbb{G} \to \mathbb{T}$ denotes the distance between fuzzy granules. The function takes two granules, $g_a = (m_a, \sigma_a) \in \mathbb{G}$, $g_b = (m_b, \sigma_b) \in \mathbb{G}$, and outputs a triangular number $t \in \mathbb{T}$ represented by a triangular fuzzy set:

$$\delta\left((m_a, \sigma_a), (m_b, \sigma_b)\right) \to (s_{\min}, c, s_{\max}), \tag{2}$$

where c is the core, $s_{\min}$ is the minimum of the support, and $s_{\max}$ is the maximum of the support of the triangular fuzzy set. The values are elaborated for each attribute separately with the formulae

$$c = \left|m_b - m_a\right|, \tag{3}$$

$$s_{\min} = c - \max(\sigma_a, \sigma_b), \tag{4}$$

$$s_{\max} = c + \max(\sigma_a, \sigma_b). \tag{5}$$

3.2.3. Neighbourhood. GrDBSCAN finds neighbours for granule g. The neighbourhood $\mathbb{H}$ is defined as a set of granules whose distance from the central granule is less than ε. The distance between granules g_a and g_b is a fuzzy value elaborated with Eqns. (3)–(5). In the next step, the algorithm has to state whether granule g_b is in the neighbourhood $\mathbb{H}_a$ of granule g_a. The radius ε of g_a's neighbourhood is a crisp number. The membership $u^{(d)}_{\mathbb{H}_a}(g_b)$ of granule g_b to the neighbourhood $\mathbb{H}_a$ of granule g_a with regard to attribute d is a fuzzy value defined as

$$u^{(d)}_{\mathbb{H}_a}(g_b) = \begin{cases} 0, & \varepsilon \le s_{\min}, \\ \dfrac{(\varepsilon - s_{\min})^2}{(c - s_{\min})(s_{\max} - s_{\min})}, & s_{\min} < \varepsilon \le c, \\ 1 - \dfrac{(s_{\max} - \varepsilon)^2}{(s_{\max} - c)(s_{\max} - s_{\min})}, & c < \varepsilon < s_{\max}, \\ 1, & s_{\max} \le \varepsilon, \end{cases} \tag{6}$$

where $(s_{\min}, c, s_{\max})$ stands for the fuzzy distance between granules g_a and g_b with regard to attribute d.

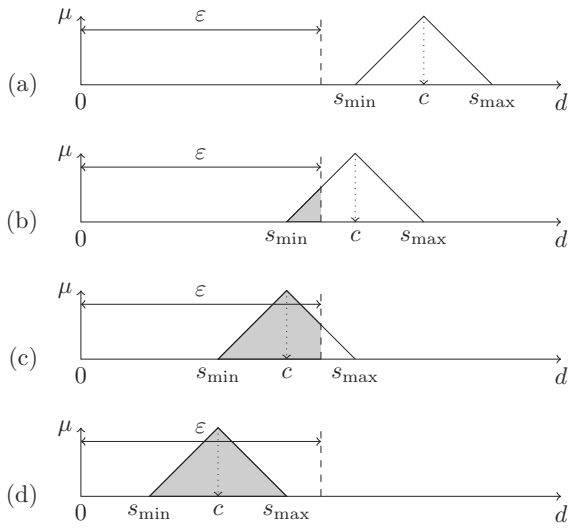


Fig. 1. Distance is represented with a triangular fuzzy set. The radius ε of a neighbourhood is a crisp value. The filled area denotes the degree to which the distance is within the radius ε. In the case (a) the distance is not within the radius ε, so the area is 0. In the case (b) the distance is partially within ε and the area is $1/8$. In the case (c) the distance is almost completely within ε and the area is $7/8$. In the case (d) the distance is completely within ε, so the area is 1.

An example of applications of the formula (6) above is presented in Fig. 1. In the case (a) the distance is not within the radius ε, so the area is 0. In the case (b) the distance is partially within ε and the area is $1/8$. In the case (c) the distance is almost completely within ε and the area is $7/8$. In the case (d) the distance is completely within ε, so the area is 1.

The membership $u_{\mathbb{H}_a}(g_b)$ of granule g_b to the neighbourhood of granule g_a is elaborated as the t-norm $\odot$ of memberships elaborated for all attributes $1, 2, \ldots, D$:

$$u_{\mathbb{H}_a}(g_b) = u^{(1)}_{\mathbb{H}_a}(g_b) \odot \ldots \odot u^{(D)}_{\mathbb{H}_a}(g_b) = \bigodot_{d\in\mathbb{D}} u^{(d)}_{\mathbb{H}_a}(g_b). \quad (7)$$

3.2.4. General idea of GrDBSCAN. The general idea of the proposed algorithm is presented in Algorithm 3. The first step is the granulation of input data (line 1). In this paper, GrDBSCAN uses one of two algorithms: FCM (Dunn, 1973) or FCOM (Leski, 2016), but any technique that produces Gaussian granules described in Section 3.2.1 can be applied. The goal of this step is the reduction in the number of data items for the second step, which is density-based clustering of fuzzy granules (line 2).

Algorithm 3. GrDBSCAN.

Require: $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_X\}$: data set
Require: ε, $P_{\min}$, ξ, ψ: GrDBSCAN parameters
Require: δ: distance function
1: $\mathbb{G} \leftarrow$ granulate($\mathbb{X}$); {data granulation}
2: $\mathbb{C} \leftarrow$ clusterGranules($\mathbb{G}$); {clustering of granules with Algorithm 4}

3.2.5. Clustering of granules. Granules have fuzzy memberships to clusters. In order to handle fuzzy neighbourhoodness, the algorithm introduces two new parameters:

- ξ: threshold membership below which a data item is regarded as not belonging to any cluster;
- ψ: threshold membership above which a data item is regarded as a neighbour of the seed of a cluster.

The relation between the values is $\xi < \psi$. If a granule is a neighbour of the seed of a cluster, it belongs to a cluster and its membership $u > \psi$. In such a case the granule is not going to be a seed of a new cluster because that is possible only if $u < \xi$.

The elaborated clusters are presented with membership matrix $\mathbf{U}$. Each row of the matrix represents a cluster and each column—a granule. Each value in a row is a membership value of a granule to a cluster. Matrix $\mathbf{U} = (u_{cg})$ is composed of membership values u_{cg} of granule g to cluster c.

Procedure 'clusterGranules'. This is a core procedure of GrDBSCAN (Algorithm 4). The first step is a random selection of a seed granule for the first cluster (line 3 in Algorithm 4). Then, a loop is started (lines 4–12). First the algorithm finds neighbours – it elaborates membership values of all granules to the neighbourhood of the seed granule (line 7). Then a neighbour with the highest membership to the neighbourhood (the "best" neighbour) is selected (line 8). Then follows an expansion of the neighbourhood of the "best" neighbour (line 9). This approach is repeated until there are no more "best" neighbours. When neighbourhood expansion is no longer possible, a new cluster is completed and added to the set of clusters (line 10). Finally, a new seed granule is chosen and the loop is repeated. If no more seed granules can be found, the last step of the algorithm removes clusters with too few granules inside (line 13).

Below we provide a more detailed description of subprocedures used in form of Algorithm 4.

Procedure 'findNeighboursMemberships'. This procedure (Algorithm 5, line 7) yields a value 1 for true neighbours, 0 for true non-neighbours, and partial

membership for granules within boundary region. It applies Eqns. (3)–(5), (6), and (7).

Algorithm 4. clusterGranules.

Require: $\mathbb{G} = \{g_1, g_2, \ldots, g_G\}$: data set
Require: ε, $P_{\min}$, ξ, ψ: parameters
Require: δ: distance function
1: {initialize algorithm}
2: $\mathbf{U} \leftarrow [\,]$; {empty membership matrix}
3: $g_s \leftarrow$ a random granule in $\mathbb{G}$; {seed granule for a new cluster}
4: **while** $g_s \neq$ null **do**
5: $\mathbf{p} \leftarrow$ [**false**]; {flags for each granule denoting being processed}
6: $\mathbf{p}[g_s] \leftarrow$ true; {currently processed}
7: $\mathbf{u}_s \leftarrow$ findNeighboursMemberships$(g_s, \mathbb{G})$; {find neighbours of g_s, Algorithm 5}
8: $g_h \leftarrow$ getBestNeighbour$(\mathbb{G}, \mathbf{u}_s, \mathbf{p})$; {get the neighbour with the highest membership, Algorithm 6}
9: $\mathbf{u}_s \leftarrow$ expandCluster$(g_h, \mathbf{u}_s, \mathbf{p})$; {expand the cluster c with g_h's neighbours, Algorithm 7}
10: $\mathbf{U} \leftarrow [\mathbf{U}\ \mathbf{u}_s]^{\mathrm{T}}$; {$\mathbf{u}_s$ represents membership of all granules to a completed cluster, we add a row representing a cluster to membership matrix}
11: $g_s \leftarrow$ findNewSeed($\mathbf{U}$); {find a new seed granule for a new cluster, Algorithm 9}
12: **end while**
13: $\mathbf{U} \leftarrow$ pruneClusters($\mathbf{U}$); {Alg. 10}

Algorithm 5. findNeighboursMemberships.

Require: $\mathbb{G} = \{g_1, g_2, \ldots, g_G\}$: set of granules
Require: ε, $P_{\min}$: DBSCAN parameters
Require: g_s: seed granule to elaborate neighbours for
Require: tnorm
1: {initialize algorithm}
2: **for all** granule g in $\mathbb{G}$ **do**
3: $u \leftarrow 1$;
4: **for all** d in g's attributes **do**
5: {applied to each attribute d}
6: $l_d \leftarrow \delta(g_s, g)$; {distance between g_s and g with regard to the d-th attribute, Eqns. (3), (4), (5)}
7: $u_d \leftarrow$ calculate membership for l_d and ε with Eqn. (6);
8: $u \leftarrow$ tnorm (u, u_d); {Eqn. (7)}
9: **end for**
10: $\mathbf{u}[g] \leftarrow u$; {membership of granule g to the neighbourhood of seed granule g_s}
11: **end for**
12: **return** $\mathbf{u}$; {memberships of all granules to the neighbourhood of seed granule g_s}

Procedure 'getBestNeighbour'. This procedure (Algorithm 6) is responsible for identification of a neighbour with the highest membership to the neighbourhood. It finds an unprocessed neighbour with the highest membership to the neighbourhood and whose membership is greater than the threshold value ψ (cf. Section 3.2.5).

Procedure 'expandCluster'. This procedure (Algorithm 7) is responsible for expansion of a new cluster. It takes membership values $\mathbf{u}_s$ of all granules to the seed granule of the cluster that is being expanded and a vector $\mathbf{p}$ of flags that signify if a granule has already been processed. It also takes a granule g_h that belongs to the cluster and tests if it is possible to expand the cluster with g_h's neighbours. Therefore, neighbours of g_h are elaborated (line 2 in Algorithm 7). Their membership

Algorithm 6. getBestNeighbour.

Require: $\mathbb{G} = \{g_1, g_2, \ldots, g_G\}$: set of granules
Require: $\mathbf{u}_s$: neighbours memberships
Require: $\mathbf{p}$; flags for each granule denoting being processed
1: $v_{\max} \leftarrow 0$; {maximal value}
2: $i_{\max} \leftarrow$ null; {index of maximal value}
3: **for** $k \leftarrow 1$ **to** G **do**
4: {for each granule}
5: $u \leftarrow \mathbf{u}_s[k]$;
6: **if** $u > v_{\max}$ **and** $u > \psi$ **and** $\mathbf{p}[k]$ **= false then**
7: $v_{\max} \leftarrow u$;
8: $i_{\max} \leftarrow k$;
9: **end if**
10: **end for**
11: **return** $g_{i_{\max}}$;

Algorithm 7. expandCluster.

Require: $\mathbb{G} = \{g_1, g_2, \ldots, g_G\}$: data set
Require: g_h: a neighbour of seed granule g_s whose neighbours are used to expand cluster
Require: $\mathbf{u}_s$: membership values to the neighbourhood of the seed granule of the cluster
Require: $\mathbf{p}$; flags for each granule denoting being processed
1: **while** $g_h \neq$ null **do**
2: $\mathbf{u}_h \leftarrow$ findNeighboursMemberships$(\mathbb{G}, \varepsilon, g_h)$; {find neighbour's neighbours, Algorithm 5}
3: update $\mathbf{u}_s$ with $\mathbf{u}_h$; {Algorithm 8}
4: $\mathbf{p}[g_h] \leftarrow$ **true**; {g_h is processed and will not be processed again}
5: $g_h \leftarrow$ getBestNeighbour$(\mathbb{G}, \mathbf{u}_s, \mathbf{p})$; {get the neighbour with the highest membership, Algorithm 6}
6: **end while**

to the neighbourhood of g_h is stored in vector $\mathbf{u}_h$. Then memberships to the seed granule have to be updated (line 3). If a granule is a neighbour of a neighbour of the seed granule, it is also a neighbour of the seed granule.

Algorithm 8 and Fig. 2 present the idea of the membership update. Let g_a be a seed of a new cluster. For all granules we elaborate memberships to g_a's neighbourhood. Let the values for granules g_b and g_c be $u_a(b) = 1$ and $u_a(c) = 0.3$, respectively. In the next step, we try to extend the cluster and elaborate neighbours of granule g_b (which is a neighbour of granule g_a). Let the elaborated value of the membership of granule g_c to the neighbourhood of granule g_b be $u_b(c) = 0.8$. Now we update the membership $u_a(c) = 0.3$ with the formula

$$u_a(c) \leftarrow \max\left(u_a(c), u_a(b) \odot u_b(c)\right), \tag{8}$$

where $\odot$ stands for a t-norm. In the example, the original value $u_a(c) = 0.3$ is updated to $u_a(c) \leftarrow 0.8$ (the minimum t-norm is used).

The last two steps of the 'expandCluster' procedure are line 4 where we set the processed flag for granule g_h and line 5 where we choose a new neighbour to expand the cluster with. When no more expansion is possible, the procedure ends and returns to line 10 in the 'clusterGranules' procedure (Algorithm 4). The row representing the cluster is elaborated and added to the membership matrix $\mathbf{U}$ (line 10).

Procedure 'findNewSeed'. The next step is a selection of a new seed granule for a next cluster (Algorithm 9). The procedure chooses a granule with a minimal membership to all clusters. For each granule its memberships to all clusters are aggregated with an s-norm. Granules with aggregated membership greater than or equal to ξ cannot be candidates for a new seed granule (cf. Section 3.2.5). A candidate cannot also be a seed of any already existing cluster.

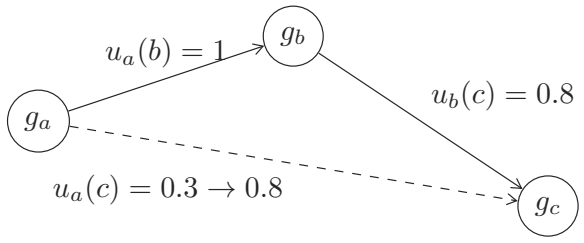


Fig. 2. Update of the membership value of g_c to the cluster whose seed is g_a. The membership is updated from original 0.3 to 0.8.

Algorithm 8. updateMembership.

Require: $\mathbf{u}_{g_s}$: membership values to the neighbourhood of the seed granule of the cluster

Require: $\mathbf{u}_{g_h}$: membership values to the neighbourhood of the neighbour of the seed granule

1: **for all** g **do**
2: $u_{g_s}(g) \leftarrow \max\left(u_{g_s}(g), u_{g_s}(g_h) \odot u_{g_h}(g)\right)$;
3: **end for**

Procedure 'pruneClusters'. The very last step in the GrDBSCAN algorithm is the pruning of clusters (Algorithm 10). If the cardinality of a cluster is lower than $P_{\min}$, it is removed from a set of clusters.

3.2.6. Complexity of GrDBSCAN. Before we analyse the complexity of the GrDBSCAN algorithm, let us recall the notation: X stands for the number of input data and G is the number of granules. GrDBSCAN consists of two parts: the first one is granulation (line 1 in Algorithm 3), the second—the clustering of fuzzy granules (line 2 in Algorithm 3).

In the experiments for granulation, the FCM algorithm is used (although the implementation is ready for other clustering algorithms). It is based on minimisation of a criterion function. The function is minimised in the Picard iteration procedure. FCM holds data in a membership matrix. The matrix has G rows (a separate row for each granule) and X columns (each column for each object). The value in the g-th row and the x-th column denotes the membership of the x-th data item to the g-th granule. Each iteration of the FCM algorithm has three steps: (i) determination of granule centres with time complexity $O(XGD)$, (ii) computation of distances of data items from granule centres: $O(XGD)$, and (iii) modification of the partition matrix $O(XGD)$. If the algorithm is run I times, its time complexity is

Algorithm 9. findNewSeed.

Require: $\mathbf{U}$

1: $\mathbf{u} \leftarrow$ aggregate each column if $\mathbf{U}$ separately with an s-norm; {each column represents memberships of one granule to all cluster}
2: $g \leftarrow$ choose a granule (that is not a seed granule for any cluster) with minimal aggregated value in $\mathbf{u}$ that is less than ξ;
3: **return** g;

Algorithm 10. pruneClusters.

Require:

1: **for all** rows in $\mathbf{U}$ **do**
2: $w \leftarrow$ sum up values in a row;
3: **if** $w < P_{\min}$ **then**
4: remove the row from $\mathbf{U}$;
5: **end if**
6: **end for**

$O(XGDI)$. This is the complexity of the first step of the algorithm.

The idea of the granule clustering part of the GrDBSCAN algorithm is similar to that of DBSCAN. However, its complexity requires a more in-depth discussion. Similarly to the complexity issue of DBSCAN, theoretical complexity is difficult to find. Let us focus on the worst-case scenario. In each iteration all neighbours of an item in question are found. This step has $O(G^2)$ complexity. The number of iterations equals that of final clusters. It is (in the worst-case) G, thus resulting in $O(G^3)$ complexity for the clustering part.

The complexity of the clustering part of the GrDBSCAN algorithm is higher than that of the DBSCAN algorithm since no items can be discarded from further analysis after a membership assignment. In the new granular algorithm, one object can determine a member of multiple clusters with different memberships. For each granule we have to determine distances to all other granules. Furthermore, when expanding the neighbourhood, the distances are determined once again for every neighbour granule. In the worst case, the number of neighbours approaches that of granules. Finally, the operations mentioned will be repeated for every core granule, that is, for every output cluster (including that pruned in Algorithm 10). In the worst case, the number of output clusters approaches that of granules. All the above operations seem to require worst-case $O(G^3)$ time. The reasoning above should be expanded, though. A granule can be a seed granule for a cluster only once. If a granule is in a cluster (the granule's membership to the cluster is no less than ψ), it will not be a seed of a new cluster (cf. Section 3.2.5). Thus, even worst-case complexity will be lower. In many cases the number of clusters is lower than the number of items to cluster – this is the idea of clustering. Therefore, the complexity of this part of the GrDBSCAN algorithm is lower than $O(G^3)$ and is $O(G^2 \cdot K)$, where K stands for the final number of clusters.

Finally, the complexity of the GrDBSCAN algorithm is $O(XDIG + G^2K)$. It is worth noting that the complexity of GrDBSCAN is linear with respect to the size X of the input data set and quadratic (cubic in the worst-case scenario) for the number of granules G. Commonly, the number of granules is lower than that of input data $G < X$ (or even $G \ll X$), and G is a parameter of the GrDBSCAN.

4. Experiments

The GrDBSCAN algorithm has been implemented as part of a library for fuzzy and neuro-fuzzy systems (Siminski, 2019) freely available from the GitHub repository.[1] GrDBSCAN works with fuzzy values. It applies t-norms and s-norms for modelling 'and' and 'or' operators, respectively. Theoretically, any t-norm and s-norm can be used. In our experiments, the minimum t-norm and maximum s-norm are used.

4.1. Data sets. Both synthetic and real-life data sets are used in the experiments. The synthetic data sets with known numbers of clusters are inspired by Karami and Johansson (2014) as well as Starczewski *et al*. (2020). The data sets (2D "lagoon", "two crescents", "four angles", "two circles", "various sizes", and 3D "spheres") are visualised in Fig. 3. Each data set holds 1 000, 2 000, 5 000, 10 000, 20 000, 30 000, 40 000, and 50 000 data items. The real-life data sets with an unknown number of clusters are available from public websites: "gowalla"[2] and "brightkite".[3] These are two-dimensional data sets with 6.4 million ("gowalla") and 4.7 million ("brightkite") entries (Cho *et al*., 2011).

4.2. Granulation. The GrDBSCAN algorithm has two steps: the granulation of input data and the clustering of granules. For the first step any algorithm that yields granules represented by Gaussian fuzzy sets can be used. In the experiments, the FCM (Dunn, 1973) and FCOM (Leski, 2016) algorithms are applied. Both share two parameters: the exponent for the distances and the number of groups to elaborate. The exponent value is set to $m = 2$. This value is supported by many clustering experiments run by various researchers. The value is also applicable for the granulation step of GrDBSCAN. However, higher values of this parameter (e.g., $m = 4$) were also used to get more concise granules (e.g., for the "gowalla" and "brightkite" data sets).

FCM and FCOM cannot find an optimal number of groups and this value has to be passed as an input parameter. The experiments were run for various numbers of groups. The FCOM algorithm has one more parameter: the loss function. The original paper on FCOM (Leski, 2016) describes several loss functions but without any recommendation. In the experiments, the logarithmic-linear loss function was used. This choice was supported by the remarks stated by Siminski (2017). FCM and FCOM are iterative algorithms. In the experiments, the number of iterations is set to 100.

4.3. Preliminary experiments. The preliminary experiments focused on parameters ξ and ψ. They were run on the "two circles" data set for the values $0.1 \leq \xi \leq \psi \leq 0.9$. High values of ψ produce smaller clusters and

[1] http://github.com/ksiminski/neuro-fuzzy-library.

[2] https://snap.stanford.edu/data/loc-gowalla.html.

[3] https://snap.stanford.edu/data/loc-brightkite.html.

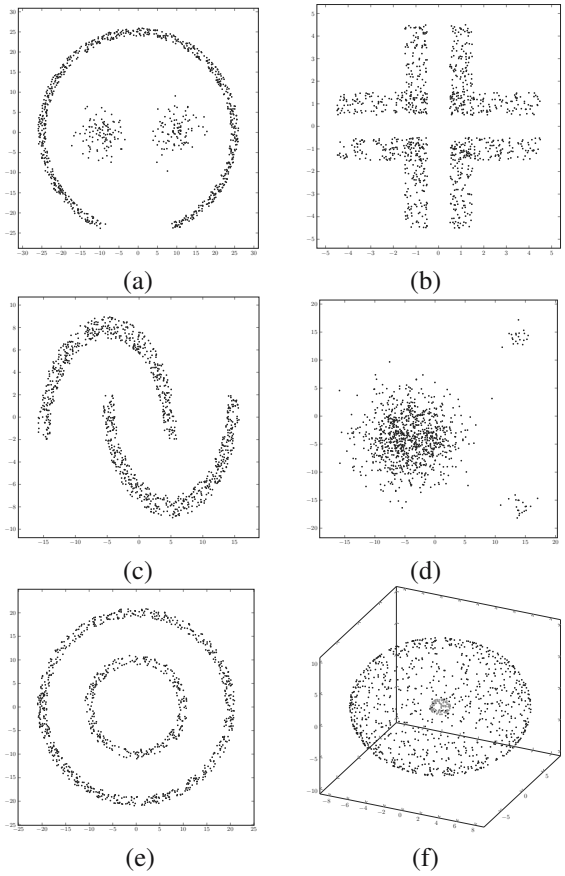


Fig. 3. Visualisation of data sets: "lagoon" (a) "four angles" (b) "two crescents" (c) "various sizes" (d) "two circles" (e) "spheres"(f).

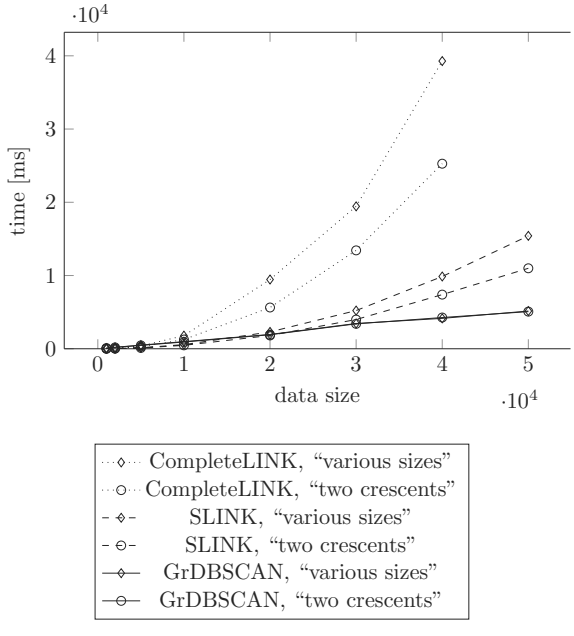


Fig. 4. Comparison of execution times of the GrDBSCAN, SLINK, and CompleteLINK algorithms for the "various sizes" and "two crescents" data sets.

low values of ψ result in larger overlapping clusters. High values of parameter ξ tend to produce more clusters. while low values of parameter ξ tend to produce fewer clusters, but more overlapping ones, The experiments reveal that GrDBSCAN is not very sensitive to these parameters (ε is more influential). The values $\xi = 0.4$ and $\psi = 0.6$ are used in the following experiments.

4.4. Running time. This experiment has two parts for (a) a varying number of data items and (b) a varying number of granules.

Number of data items. The experiments focus on running time of DBSCAN and GrDBSCAN. They are run for all data sets with 1 000, 2 000, 5 000, 10 000, 20 000, 30 000, 40 000, and 50 000 data items.

The comparison of execution times of DBSCAN and GrDBSCAN is presented in Tables 1 and 2 for the "two crescents" and "brightkite" data sets, respectively. The symbol '[−]' means that the execution time was very long – for example, the estimated time of DBSCAN for 3 000 000 data items is approximately 17.5 days and for 4 000 000 it is about 31.25 days.

Figures 5–9 visualise execution times of DBSCAN and GrDBSCAN for the "two crescents", "two circles", "two crescents", "spheres", and "gowalla" data sets, respectively. Figure 9 presents the execution time for small numbers of data items. For those, DBSCAN runs faster than GrDBSCAN. However, the square time complexity of DBSCAN surpasses the execution time of GrDBSCAN for bigger data sets.

Figure 4 compares the execution time of GrDBSCAN with the SLINK (Sibson, 1973) and CompleteLINK (Defays, 1977) algorithms. We used these algorithms for the comparison because they can elaborate groups of any shape, whereas algorithms in the FCM family produce only hyperellipsoidal groups. It can be observed that both SLINK and CompleteLINK have superlinear time complexity. For small data sets, SLINK is faster; however, for large data sizes, its execution time surpasses that of GrDBSCAN. For this experiment we used the `scipy` Python implementation. The absolute execution time values may not be suitable for comparison, as the implementation languages differ, but the observed trend for the SLINK and CompleteLINK algorithms is superlinear, whereas it is linear for GrDBSCAN.

The results for all data sets follow the same pattern. This is why the results are presented only for selected ones. The execution time of the DBSCAN algorithm follows $O(n^2)$ complexity. The complexity of the GrDBSCAN is in concordance with the theoretical analysis given in Section 3.2.6.

Table 1. Clustering time for the "two crescents"data set.

data size	running time [ms] DBSCAN	GrDBSCAN
1000	119	98
2000	450	185
5000	2845	462
10000	11838	925
20000	47709	1914
30000	116582	3402
40000	208036	4265
50000	332864	5071

Table 2. Execution time for the "brightkite" data set.

data size	execution time [ms] DBSCAN	GrDBSCAN
1000	122	238
2000	509	476
3000	1 008	689
4000	1 839	906
5000	2 995	1 199
10000	12 086	2 396
20000	49 827	4 686
30000	114 728	7 224
40000	211 008	9 744
50000	342 312	13 238
100000	1 417 806	25 388
200000	[−]	53 661
300000	[−]	80 650
400000	[−]	107 662
500000	[−]	139 438
1000000	[−]	277 234
2000000	[−]	557 470
3000000	[−]	835 631
4000000	[−]	1 144 224

Table 3. Execution time for the "two crescents" data set.

number of granules	execution time [ms] granulation	clustering
20	1063.7	0.8550
50	2610.9	0.9547
100	5303.5	1.5423
200	10753.2	3.4178
400	21105.8	12.7719
500	27169.2	20.4674
600	31688.6	31.1268
800	48575.4	57.8904
1000	57786.0	98.6146

Table 4. Clustering quality matrix for the "two crescents"data set.

		GrDBSCAN groups 1	2
DBSCAN groups	1	248.505	0.498
	2	0.779	224.891

Number of granules. This section presents results for the "two crescents" data set with a constant number of data items: 10 000. The parameter values are $\varepsilon = 5$, $\xi = 0.4$, and $\psi = 0.6$. The execution time is measured separately for two steps of GrDBSCAN: granulation and clustering of granules (cf. Algorithm 4). This is necessary because each step operates on a significantly different number of input objects: granulation runs for 10 000 data items, whereas clustering runs for a much smaller number of granules. Therefore, the granulation time is much longer and overshadows the clustering time. The times for both steps are presented in Table 3 and Fig. 10 for the clustering step only. The results are in concordance with theoretically elaborated complexity of GrDBSCAN (Section 3.2.6). If we set a constant number of granules, GrDBSCAN runs in linear time with regard to the size of input data. However, sometimes (depending on the specific problem to solved) an increase in input data may need a higher number of granules. In such a case the quadratic component of the time complexity of GrDBSCAN may more strongly influence the execution time (Fig. 10).

4.5. Clustering quality measure. This section defines a clustering quality measure to quantify the quality of clustering with GrDBSCAN in comparison with DBSCAN. We treat clusters produced by the DBSCAN algorithm as the reference for the results of GrDBSCAN.

Let the value $u_c(x)$ be a membership of point x to GrDBSCAN's cluster c. It is evaluated as a maximum for all granules of a product of membership $u_g(x)$ of point x to granule g and of membership $u_c(g)$ of granule g to cluster c:

$$u_c(x) = \max_{g \in \mathbb{G}} u_c(g) \cdot u_g(x). \tag{9}$$

Having produced the values with Eqn. (9), the algorithm fills a correspondence matrix. Each column represents a cluster elaborated by GrDBSCAN, and each row represents a cluster elaborated by DBSCAN. Matrix cell l_{ab} represents the number of items assigned to cluster a by DBSCAN and to cluster b by GrDBSCAN. The value in the cell is calculated with the formula

$$l_{ab} = \sum_{k=1}^{X} u_b(x_k) \, [x_k \in a] \,, \tag{10}$$

where $[\cdot]$ is the Iverson bracket defined as

$$[x] = \begin{cases} 1, & x \text{ is true,} \\ 0, & x \text{ is false.} \end{cases} \tag{11}$$

Notation $[x_k \in a]$ evaluates to 1 if point x_k belongs to the a-th cluster elaborated by DBSCAN, otherwise its value is 0.

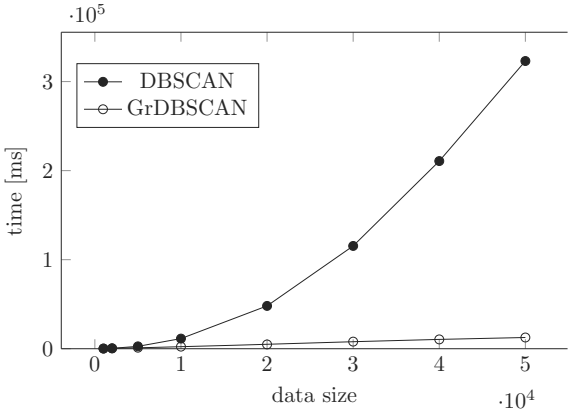


Fig. 5. Comparison of execution times with regard to the number of data items for the "two circles" data set.

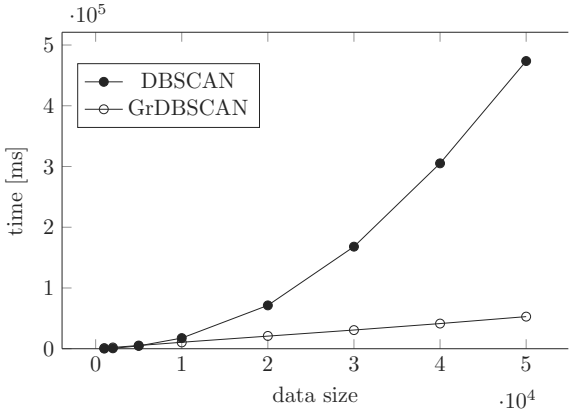


Fig. 7. Comparison of execution times with regard to the number of data items for the "spheres" data set.

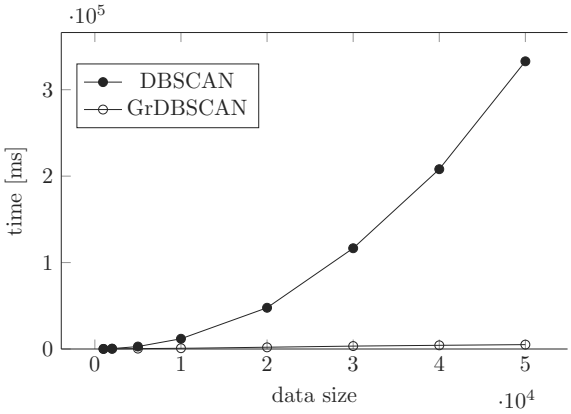


Fig. 6. Comparison of execution times with regard to the number of data items for the "two crescents" data set.

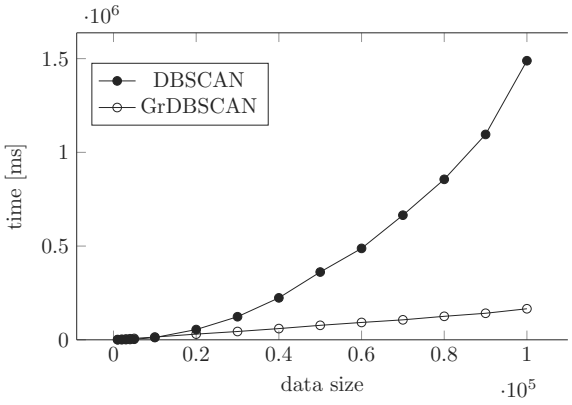


Fig. 8. Comparison of execution times with regard to the number of data items for the "gowalla" data set.

DBSCAN and GrDBSCAN form clusters independently, so the correspondence between clusters has to be found. In order to do this, first all columns are sorted with respect to the maxima of each column, then all rows are sorted with respect to the maxima of each row. This step aims at maximising values on the diagonal of the matrix. If the numbers of rows and columns are not the same, extra rows or columns with zeros are added to make the matrix a square. The quality index is determined as the multiclass Matthews correlation coefficient $\varphi \in [-1, 1]$ (Matthews, 1975; Chicco and Jurman, 2020).

4.6. Quality of clustering. When measuring the quality of clustering, DBSCAN is used as a reference algorithm. Our aim is to reduce the computational complexity with granulation of data and keep the elaborated clusters similar to those produced by DBSCAN. Quality matrices are presented in Tables 4–9 for the "two crescents", "four angles", "lagoon", "various sizes", "spheres", and "two circles" data sets; Tables 10 and 11 gather the values of the quality index φ. The matrices show that GrDBSCAN can reproduce clusters elaborated by the DBSCAN algorithm. Visualisations of clustering results are presented in Figs. 11 and 16.

Granules are represented in a symbolic way. They are modelled with Gaussian fuzzy sets and are plotted as ellipses whose orthogonal sizes are fuzzifications of attributes. The larger the ellipse, the more fuzzy granule it represents.

4.7. Robustness to noise. In this section, we analyse the robustness of the GrDBSCAN algorithm to noise. Uniform noise has been added to the "two crescents" data set with 1000 informative (non-noisy) data items. The number of granules was set to 20. The results are presented in Fig. 17. The first panel (a) presents results for a data set with 200 noise points. The original crescents were correctly identified. For 700 noise points (panel (b)), two additional regions were identified as clusters despite being built of noise points. Two crescents were identified. In the panel (c), 1500 noise points distort the results: one

Table 5. Clustering correspondence matrix for the "four angles"data set.

		GrDBSCAN groups			
		1	2	3	4
DBSCAN groups	1	144.10	0.00	0.79	0.20
	2	0.00	143.06	0.00	0.06
	3	1.19	0.00	137.00	0.00
	4	0.20	0.19	0.00	135.70

Table 6. Clustering correspondence matrix for the "lagoon"data set.

		GrDBSCAN groups		
		1	2	3
DBSCAN groups	1	374.957	0.000	0.000
	2	0.000	58.4854	0.000
	3	0.000	0.000	56.2111

Table 7. Clustering correspondence matrix for the "various sizes"data set.

		GrDBSCAN groups		
		1	2	3
DBSCAN groups	1	520.593	0.000	0.000
	2	0.000	10.309	0.000
	3	0.000	0.000	9.830

Table 8. Clustering correspondence matrix for the "spheres"data set.

		GrDBSCAN groups	
		1	2
DBSCAN groups	1	316.861	0.000
	2	0.000	95.823

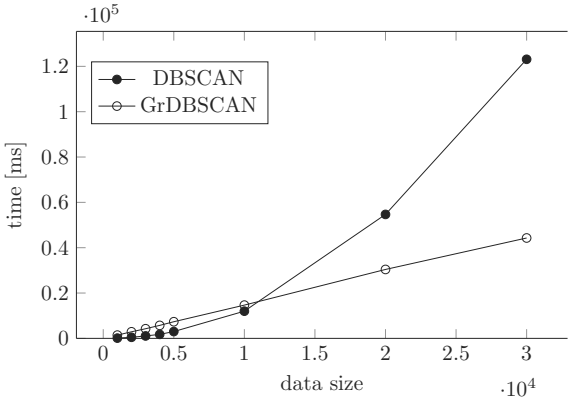


Fig. 9. Comparison of execution times with regard to the number of data items for a small number of data items in the "gowalla" data set.

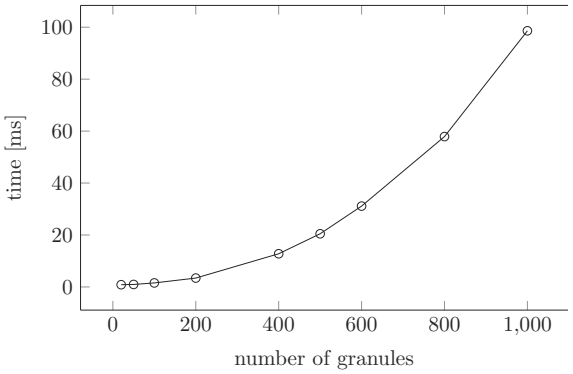


Fig. 10. Clustering time (second step of GrDBSCAN) with regard to the number of granules for the "two crescents" data set with 10 000 data items.

of the crescents is split into two clusters. For 2500 noise points, the results are even poorer. In order to get better results, the number of granules needs to be increased. In Fig. 17(e), for a data set with 2500 noise items, the number of granules is set to 100. The output clusters are filtered using the input parameter $P_{\min}$. Two of the biggest clusters cover the informative data. Figure 17(f) presents a similar case, but with 5000 noise items (and 100 granules). The two biggest clusters cover the informative data (crescents). The values of the quality index φ for experiments with granulation with the FCM and FCOM (Leski, 2016) algorithms are presented in Table 12.

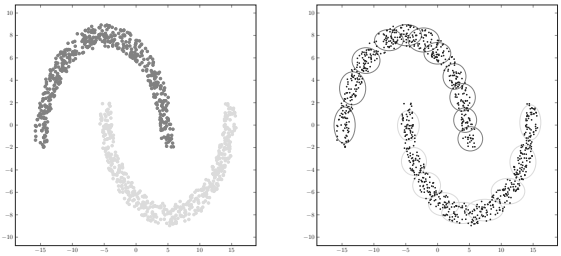
5. Conclusions

The paper presented a novel granular density-based clustering algorithm—GrDBSCAN. The main motivation was reduction in the execution time while preserving the advantages of a density-based clustering approach at the same time. In order to achieve this goal, the algorithm is split into two parts: granulation and the clustering of granules. The first step (granulation) produces fuzzy granules from input data. The second step (the clustering of fuzzy granules) produces fuzzy clusters of the elaborated granules. The proposed GrDBSCAN algorithm can elaborate clusters in shorter time than its non-granular counterpart DBSCAN. When the number of granules is constant, the GrDBSCAN algorithm is linear with regard to the input data size. GrDBSCAN can yield similar clusters to those discovered by the DBSCAN algorithm. The implementation of GrDBSCAN is freely available from a public repository.[4]

Acknowledgment

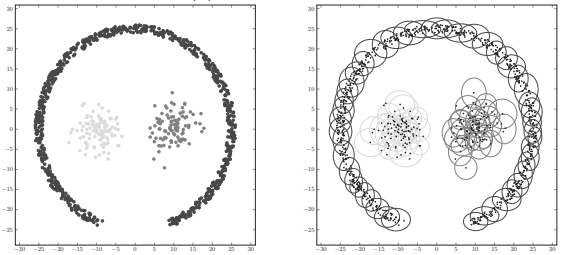
This research is funded by the Silesian University of Technology (Poland) under the grant number 02/080/RGJ22/0025.

[4] http://github.com/ksiminski/neuro-fuzzy-library.



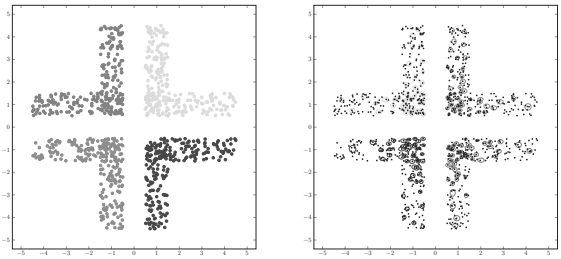
(a) (b)

Fig. 11. Clustering results produced by DBSCAN (a) and GrDBSCAN (b) for the "two crescents" data set.



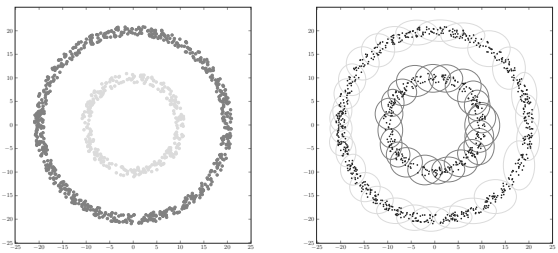
(a) (b)

Fig. 12. Clustering results produced by DBSCAN (a) and GrDBSCAN (b) for the "lagoon" data set.



(a) (b)

Fig. 13. Clustering results produced by DBSCAN (a) and GrDBSCAN (b) for the "four angles" data set.



(a) (b)

Fig. 14. Clustering results produced by DBSCAN (a) and GrDBSCAN (b) for the "two circles" data set.

Table 9. Clustering correspondence matrix for the "two circles"data set.

		GrDBSCAN groups	
		1	2
DBSCAN	1	316.135	8.022
groups	2	7.067	182.800

Table 10. Clustering quality index φ.

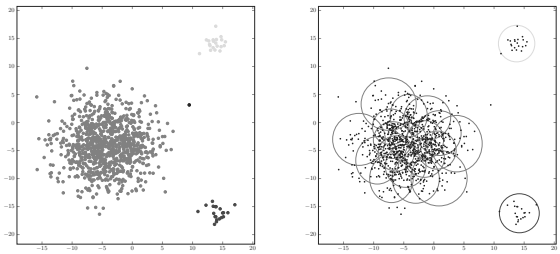
dataset	quality index φ
"lagoon"	1.000
"various sizes"	1.000
"spheres"	1.000
"two circles"	0.937
"four angles"	0.994
"two crescents"	0.995

Table 11. Clustering quality index φ for the "gowalla" and "brightkite" data sets.

number of	quality index φ	
items	"gowalla"	"brightkite"
1000	1.000	0.996
2000	1.000	1.000
5000	0.999	1.000

Table 12. Clustering quality index φ for the "two crescents" data set with noise produced with GrDBSCAN with the granulation algorithms FCM and FCOM.

number of		quality index φ	
noise items	granules	FCM	FCOM
100	20	0.997	0.983
200	20	0.993	0.887
700	20	0.868	0.525
1500	20	0.593	0.269
2500	20	0.455	0.181
2500	100	0.625	0.938
5000	100	0.448	0.925



(a) (b)

Fig. 15. Clustering results produced by DBSCAN (a) and GrDBSCAN (b) for the "various sizes" data set.

References

Agrawal, R., Gehrke, J., Gunopulos, D. and Raghavan, P. (1998). Automatic subspace clustering of high dimensional data for data mining applications, *ACM SIGMOD Record* **27**(2): 94–105.

Ankerst, M., Breunig, M., Kriegel, H.-P. and Sander, J. (1999). Optics: Ordering points to identify the clustering structure, *ACM SIGMOD'99: International Conference on Management of Data, Philadelphia, USA*, pp. 49–60.

Bargiela, A. and Pedrycz, W. (2006). The roots of granular

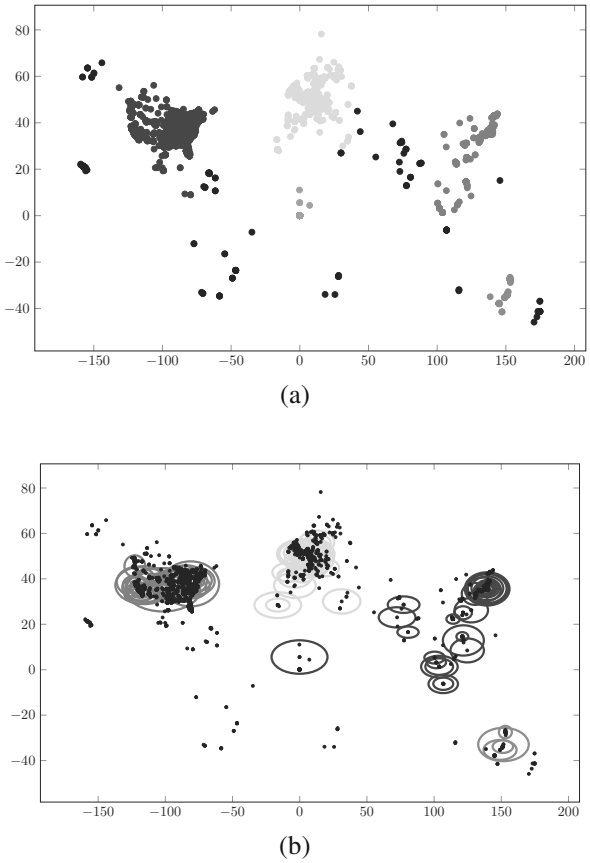


Fig. 16. Clustering results for the "brightkite" data set with 4000 data items produced by DBSCAN (a) and GrDBSCAN (b) (100 granules, $m = 3.5$).

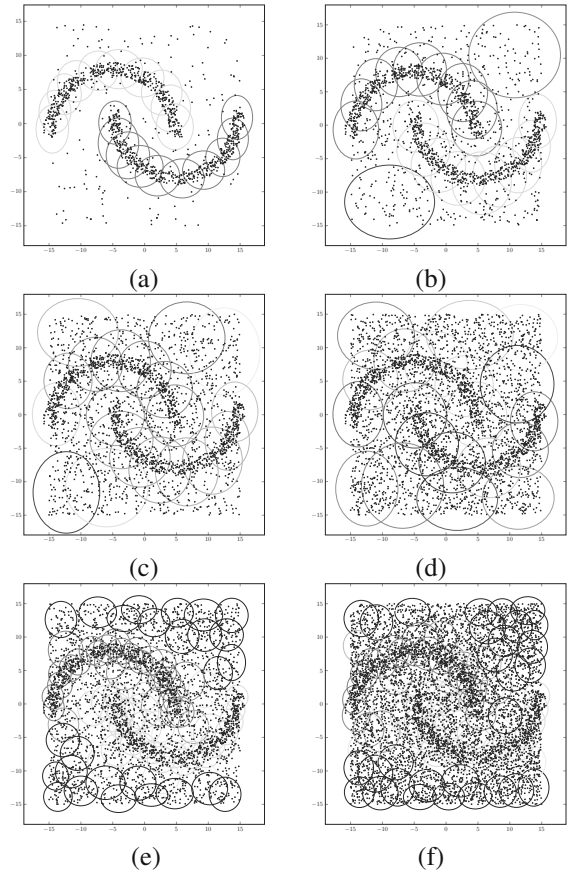


Fig. 17. Visualisation of clustering results of the "two crescents" data set with noise: 200 noise points and 20 granules (a), 700 noise points and 20 granules (b), 1500 noise points and 20 granules (c), 2500 noise points and 20 granules (d), 2500 noise points and 100 granules (e), 5000 noise points and 100 granules (f).

computing, *2006 IEEE International Conference on Granular Computing, Atlanta, USA*, pp. 806–809.

Chakraborty, C. and Chakraborty, D. (2006). A theoretical development on a fuzzy distance measure for fuzzy numbers, *Mathematical and Computer Modelling* **43**(3): 254–261.

Cheng, C.-H., Fu, A.W. and Zhang, Y. (1999). Entropy-based subspace clustering for mining numerical data, *KDD'99: Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, USA*, pp. 84–93.

Chicco, D. and Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation, *BMC Genomics* **21**(6): 1–13.

Cho, E., Myers, S.A. and Leskovec, J. (2011). Friendship and mobility: User movement in location-based social networks, *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'11, San Diego, USA*, pp. 1082–1090.

Ciucci, D. (2016). Orthopairs and granular computing, *Granular Computing* **1**: 159–170.

Defays, D. (1977). An efficient algorithm for a complete link method, *The Computer Journal* **20**(4): 364–366.

Diamond, P. and Körner, R. (1997). Extended fuzzy linear models and least squares estimates, *Computers & Mathematics with Applications* **33**(9): 15–32.

Dunn, J.C. (1973). A fuzzy relative of the ISODATA process and its use in detecting compact, well separated clusters, *Journal Cybernetics* **3**(3): 32–57.

Ester, M., Kriegel, H.-P., Sander, J. and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise, *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, USA*, pp. 226–231.

Gan, J. and Tao, Y. (2015). DBSCAN revisited: Mis-claim, un-fixability, and approximation, *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, SIGMOD'15, Melbourne, Australia*, pp. 519–530.

Guha, S., Rastogi, R. and Shim, K. (2001). CURE: An efficient clustering algorithm for large databases, *Information Systems* **26**(1): 35–58.

Gunawan, A. (2013). *A Faster Algorithm for DBSCAN*, Master's thesis, Eindhoven University of Technology, Eindhoven.

Gustafson, D.E. and Kessel, W.C. (1978). Fuzzy clustering with a fuzzy covariance matrix, *1978 IEEE Conference on Decision and Control Including the 17th Symposium on Adaptive Processes, San Diego, USA*, pp. 761–766.

Hartuv, E. and Shamir, R. (2000). A clustering algorithm based on graph connectivity, *Information Processing Letters* **76**(4): 175–181.

Hinneburg, A. and Keim, D.A. (1998). An efficient approach to clustering in large multimedia databases with noise, *Proceedings of the 4th International Conference on Knowledge Discovery and Datamining (KDD'98), New York, USA*, pp. 58–65.

Jajuga, K. (1991). L_1-norm based fuzzy clustering, *Fuzzy Sets and Systems* **39**(1): 43–50.

Junlin, L. and Hongguang, F. (2011). Molecular dynamics-like data clustering approach, *Pattern Recognition* **44**(8): 1721–1737.

Karami, A. and Johansson, R. (2014). Choosing DBSCAN parameters automatically using differential evolution, *International Journal of Computer Applications* **91**(7): 1–11.

Karypis, G., Han, E.-H. and Kumar, V. (1999). Chameleon: A hierarchical clustering algorithm using dynamic modeling, *IEEE Computer* **32**: 68–75.

Keet, C.M. (2008). *A Formal Theory of Granularity*, PhD thesis, Free University of Bozen-Bolzano, Bolzano.

Krishnapuram, R. and Keller, J. (1993). A possibilistic approach to clustering, *IEEE Transactions on Fuzzy Systems* **1**(2): 98–110.

Leski, J.M. (2016). Fuzzy *c*-ordered-means clustering, *Fuzzy Sets and Systems* **286**: 114–133.

Luo, M. and Cheng, Z. (2015). The distance between fuzzy sets in fuzzy metric spaces, *12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD'15), Zhangjiajie, China*, pp. 197–201.

Matthews, B. (1975). Comparison of the predicted and observed secondary structure of T4 phage lysozyme, *Biochimica et Biophysica Acta (BBA)—Protein Structure* **405**(2): 442–451.

Michalak, M. and Stawarz, M. (2013). HRoBi—The algorithm for hierarchical rough bi-clustering, *in* L. Rutkowski *et al.* (Eds), *Artificial Intelligence and Soft Computing*, Lecture Notes in Computer Science, Vol. 7895, Springer, Berlin/Heidelberg, pp. 194–205.

Pedrycz, A., Hirota, K., Pedrycz, W. and Dong, F. (2012). Granular representation and granular computing with fuzzy sets, *Fuzzy Sets and Systems* **203**: 17–32.

Pedrycz, W. (2013). *Granular Computing: Analysis and Design of Intelligent Systems*, CRC Press, Boca Raton.

Pedrycz, W., Succi, G., Sillitti, A. and Iljazi, J. (2015). Data description: A general framework of information granules, *Knowledge-Based Systems* **80**: 98–108.

Qian, Y.H., Liang, J.Y., Yao, Y.Y. and Dang, C.Y. (2010). MGRS: A multi-granulation rough set, *Information Sciences* **180**(6): 949–970.

Qian, Y., Liang, J., Zhi Z. Wu, W. and Dang, C. (2011). Information granularity in fuzzy binary GRC model, *IEEE Transactions on Fuzzy Systems* **19**(2): 253–264.

Salehi, S., Selamat, A. and Fujita, H. (2015). Systematic mapping study on granular computing, *Knowledge-Based Systems* **80**: 78–97.

Sander, J., Ester, M., Kriegel, H.-P. and Xu, X. (1998). Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications, *Data Mining and Knowledge Discovery* **2**: 169–194.

Schubert, E., Sander, J., Ester, M., Kriegel, H.P. and Xu, X. (2017). DBSCAN revisited, revisited: Why and how you should (still) use DBSCAN, *ACM Transactions on Database Systems* **42**(3): 1–21.

Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**(8): 888–905.

Shifei, D., Li, X., Hong, Z. and Liwen, Z. (2010). Research and progress of cluster algorithms based on granular computing, *International Journal of Digital Content Technology and its Applications* **4**(5): 96–104.

Sibson, R. (1973). SLINK: An optimally efficient algorithm for the single-link cluster method, *The Computer Journal* **16**(1): 30–34.

Siminski, K. (2014). Rough fuzzy subspace clustering for data with missing values, *Computing & Informatics* **33**(1): 131–153.

Siminski, K. (2017). Fuzzy weighted c-ordered means clustering algorithm, *Fuzzy Sets and Systems* **318**: 1–33.

Siminski, K. (2019). NFL—Free library for fuzzy and neuro-fuzzy systems, *in* S. Kozielski *et al.* (Eds), *Beyond Databases, Architectures and Structures. Paving the Road to Smart Data Processing and Analysis*, Springer International Publishing, Cham, pp. 139–150.

Siminski, K. (2020). GrFCM—Granular clustering of granular data, *in* A. Gruca *et al.* (Eds), *Man-Machine Interactions 6*, Springer International Publishing, Cham, pp. 111–121.

Siminski, K. (2021a). GrNFS: A granular neuro-fuzzy system for regression in large volume data, *International Journal of Applied Mathematics and Computer Science* **31**(3): 445–459, DOI: 10.34768/amcs-2021-0030.

Siminski, K. (2021b). An outlier-robust neuro-fuzzy system for classification and regression, *International Journal of Applied Mathematics and Computer Science* **31**(2): 303–319, DOI: 10.34768/amcs-2021-0021.

Siminski, K. (2023). 3WDNFS—Three-way decision neuro-fuzzy system for classification, *Fuzzy Sets and Systems* **466**, Article ID: 108432, DOI: 10.1016/j.fss.2022.10.021.

Siminski, K. (2022a). FuBiNFS—Fuzzy biclustering neuro-fuzzy system, *Fuzzy Sets and Systems* **438**: 84–106.

Siminski, K. (2022b). Prototype based granular neuro-fuzzy system for regression task, *Fuzzy Sets and Systems* **449**: 56–78.

Skowron, A., Jankowski, A. and Dutta, S. (2016). Interactive granular computing, *Granular Computing* **1**: 95–113.

Starczewski, A., Goetzen, P. and Er, M.J. (2020). A new method for automatic determining of the DBSCAN parameters, *Journal of Artificial Intelligence and Soft Computing Research* **10**(3): 209–221.

Stawarz, M. and Michalak, M. (2012). eBi—The algorithm for exact biclustering, *in* L. Rutkowski *et al.* (Eds), *Artificial Intelligence and Soft Computing*, Lecture Notes in Computer Science, Vol. 7268, Springer, Berlin/Heidelberg, pp. 327–334.

Wang, W., Yang, J. and Muntz, R. (1997). STING: A statistical information grid approach to spatial data mining, *Proceedings of the 23rd International Conference on Very Large Data Bases, San Francisco, USA*, pp. 186–195.

Wright, W. (1977). Gravitational clustering, *Pattern Recognition* **9**(3): 151–166.

Wu, C., Peng, Q., Lee, J., Leibnitz, K. and Xia, Y. (2021). Effective hierarchical clustering based on structural similarities in nearest neighbor graphs, *Knowledge-Based Systems* **228**: 107295.

Xia, S., Chen, L., Liu, S. and Yang, H. (2022). A new method for decision making problems with redundant and incomplete information based on incomplete soft sets: From crisp to fuzzy, *International Journal of Applied Mathematics and Computer Science* **32**(4): 657–669, DOI: 10.34768/amcs-2022-0045.

Yao, J.T., Vasilakos, A.V. and Pedrycz, W. (2013). Granular computing: Perspectives and challenges, *IEEE Transactions on Cybernetics* **43**(6): 1977–1989.

Yao, Y. (2007). The art of granular computing, *in* M. Kryszkiewicz *et al.* (Eds), *Rough Sets and Intelligent Systems Paradigms*, Springer, Berlin/Heidelberg, pp. 101–112.

Yao, Y. (2008). Granular computing: Past, present and future, *2008 IEEE International Conference on Granular Computing, GrC 2008, Hangzhou, China*, pp. 80–85.

Yao, Y. (2009). Three-way decision: An interpretation of rules in rough set theory, *in* P. Wen *et al.* (Eds), *Rough Sets and Knowledge Technology*, Springer, Berlin/Heidelberg, pp. 642–649.

Yao, Y. (2011). The superiority of three-way decisions in probabilistic rough set models, *Information Sciences* **181**(6): 1080–1096.

Yao, Y. (2016). A triarchic theory of granular computing, *Granular Computing* **1**: 145–157.

Yao, Y. (2018). Three-way decision and granular computing, *International Journal of Approximate Reasoning* **103**: 107–123.

Yao, Y. (2020). Three-way granular computing, rough sets, and formal concept analysis, *International Journal of Approximate Reasoning* **116**: 106–125.

Yao, Y. and Zhong, N. (2007). Granular computing, *in* B. Wah (Ed.), *Wiley Encyclopedia of Computer Science and Engineering*, Wiley, Hoboken, DOI: 101002/9780470050118.essse468.

Zadeh, L.A. (1979). Fuzzy sets and information granularity, *in* N. Gupta *et al.* (Eds), *Advances in Fuzzy Set Theory and Applications*, North-Holland Publishing Co., Amsterdam, pp. 3–18.

Zadeh, L.A. (1997). Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic, *Fuzzy Sets and Systems* **90**(2): 111–127.

Zadeh, L.A. (2002). From computing with numbers to computing with words—From manipulation of measurements to manipulation of perceptions, *International Journal of Applied Mathematics and Computer Science* **12**(3): 307–324.

Dawid Suchy holds BSc and MSc degrees from the Faculty of Automatic Control, Electronics and Computer Science at the Silesian University of Technology (Gliwice, Poland). His recent research is focused on granular computing.

Krzysztof Siminski holds a DSc degree from the Faculty of Automatic Control, Electronics and Computer Science at the Silesian University of Technology (Gliwice, Poland), where he currently works. His main scientific interests focus on data analysis, data mining, machine learning, computational intelligence, granular computing, and formal languages.

Received: 13 September 2022
Revised: 24 December 2022
Accepted: 23 January 2023