Data & Knowledge Engineering 75 (2012) 78–98

Contents lists available at SciVerse ScienceDirect

Data & Knowledge Engineering

journal homepage: www.elsevier.com/locate/datak

An architecture for component-based design of representative-based clustering algorithms

Boris Delibašić [a], Milan Vukićević [a,*], Miloš Jovanović [a], Kathrin Kirchner [b], Johannes Ruhland [b], Milija Suknović [a]

[a] Faculty of Organizational Sciences, University of Belgrade, Jove Ilića 154, Belgrade, Serbia
[b] Faculty of Economics and Business Administration, Friedrich Schiller University of Jena, Carl-Zeiß Straße 3, Jena, Germany

ARTICLE INFO

Article history:
Received 23 December 2010
Received in revised form 29 March 2012
Accepted 29 March 2012
Available online 10 April 2012

Keywords:
Representative-based clustering algorithms
Architecture
Reusable component
Generic algorithm
K-means

ABSTRACT

We propose an architecture for the design of representative-based clustering algorithms based on reusable components. These components were derived from K-means-like algorithms and their extensions. With the suggested clustering design architecture, it is possible to reconstruct popular algorithms, but also to build new algorithms by exchanging components from original algorithms and their improvements. In this way, the design of a myriad of representative-based clustering algorithms and their fair comparison and evaluation are possible. In addition to the architecture, we show the usefulness of the proposed approach by providing experimental evaluation.

1. Introduction

Clustering in data mining is the process of grouping a set of objects into classes of similar objects. Berkhin [1], Xu and Wunsch [2] as well as Jain et al. [3] give good overviews of clustering algorithms from different perspectives. In this paper, we only consider representative-based algorithms that group objects into K partitions (clusters) where each partition is characterized by one representative. These algorithms try to find clusters by iteratively assigning objects to partitions and recalculating representatives.

Even though there are many new approaches for clustering data [4–8] that were proposed recently, representative-based clustering is still widely used in many emerging application areas like text mining (Steinbach et al. [9]; Mahdavi et al. [10]), bio-informatics (Chen et al. [11]; Vukicevic et al. [12,13]), medicine (Nanneti et al. [14]), document clustering (Bouras et al., [15], Chan et al., [16]), social network analysis (Liu, [17]) or image segmentation and retrieval (Murthy et al. [18]).

Due to its simplicity, fast execution time and intuitively understandable optimization mechanisms, K-means sets the foundation for further development of representative-based algorithm family more than 50 years ago (Jain, [19]; Bock, [20]). The original K-means suffers from many well-known weaknesses such as:

- Clustering is strongly dependant on initial representatives;
- A representative may be trapped in the local optimum during optimization;
- The presence of outliers influences clustering;
- It assumes that all attributes have equal importance for clustering;
- The number of clusters is user-dependant etc.

* Corresponding author at: Center for Business Decision Making, Faculty of Organizational Sciences, University of Belgrade, Jove Ilića 154, Belgrade, Serbia. Tel.: +381 698893542.
E-mail address: vukicevicm@fon.bg.ac.rs (M. Vukićević).

0169-023X/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.datak.2012.03.005

In order to deal with the aforementioned shortcomings many algorithms were proposed as K-means improvements [19]. Still, original K-means remained the most popular and most widely used clustering algorithm [1,21]. Sonnenburg et al. [22] attributed this problem to a large time gap between development of cutting edge algorithms (published in scientific papers) and their implementation in popular software. This gap was aroused due to "black-box" design and non-existence of standards for publishing algorithms and their parts and consequently, a lot of efforts are being made for re-implementation of existing algorithmic parts. This disabled the proposed algorithms and algorithm improvements to be tested and used by a wider community.

Sonnenburg et al. [22] also emphasized that "black-box" design slows down the development of algorithms because it doesn't allow interchange of good ideas between the existing algorithms. The question of whether the combination (re-use) of components from the existing algorithms could improve algorithm performance asked by [22], was answered positively for decision tree algorithms in Delibasic et al. [23]. Moreover, Wolpert [24] states that there is no one-fits-all algorithm, but for every dataset a best suited algorithm can be designed. Therefore, it is important to have a wider range of algorithms available to users and to define an intelligent strategy for selecting the best algorithm for the data at hand. All this created the motivation for development of architecture for component-based design of representative-based clustering algorithms.

Architecture presented in this paper aims to bridge the gap between development of new algorithms and their implementation in popular software by providing common code repository, and flexible way of adding new reusable components (RCs). Adapting such an architecture and its implementation in open source data mining software would allow algorithm designers and software developers collaborative development of new algorithms on existing resources with less re-implementation. RC-based architecture also allows more detailed examination of influence of partial algorithm improvements (e.g. [25]) on algorithm performance (i.e. examination of algorithm performance when single RC is added or removed).

Further, by encapsulating algorithm parts as reusable components (RCs), as reported by Tracz [26] and Coronato and Del Pietro [27], it is possible to combine RCs and produce a myriad of representative, component-based clustering algorithms. This provides data mining practitioners with large space of algorithms from which they can select the best suited one for data at hand as discussed by [22,24].

Delibasic et al. [28] analyzed a family of representative-based clustering algorithms in more detail and gave an overview of typical clustering algorithm sub-problems and their solutions. Following these ideas, we propose the architecture for component-based, white-box design and evaluation of representative-based clustering algorithms. The key contributions of this paper are threefold:

1. We propose an extended set of RCs for representative-based clustering algorithm design, compared to [28].
2. Additionally, we suggest the architecture for representative, component-based design that enables assembling RCs to clustering algorithms.
3. The first evaluation of the architecture is given on 84 RC-based algorithms which were tested on 10 benchmark data sets.

The remainder of the paper is organized as follows: Section 2 reviews previous related work in the literature and software. Section 3 proposes reusable components for representative-based clustering algorithms and the architecture for RC-based clustering algorithms design. A first evaluation of this architecture is given in Section 4, whereas Section 5 describes a possible method for search of adequate algorithm for data at hand in the space of RC-based clustering algorithms. Section 6 discusses the findings of this paper and gives an outlook on further research directions.

2. Related work

K-means was extended and improved in many ways, for e.g. by involving limits for cluster sizes, merging and splitting clusters while searching for the best K, or improved representative initialization. A systematic review of majority of those algorithms is presented in Bock [20] and Jain [19]. In this section, we review some popular and also some newly proposed K-means-like algorithm extensions, and shortly several frequently used data mining software and their available flexibilities (extensions) for clustering algorithms.

Global K-means and its modifications (Lai [29]; Bagirov [30]; Hansen et al. [31]; Likas et al. [32]) propose solutions for avoiding the local optima trap. Bouras et al. [15] use K-means with cosine similarity for document clustering. Tang et al. [33] developed new fuzzy *c*-means clustering model based on the data weighted approach. Mumtaz and Duraiswamy [34] propose DBK-means algorithm as a hybrid between DBScan and K-means. Žalik [35] proposed a K-means algorithm without the user having to define the exact number of clusters. This is achieved by minimizing the cost-function that improves the original mean-square-error. PCA analysis was used for dimension reduction and initialization improvement in (Tajunisha and Saravanan, [36]). Fahim et al. [37] and Yedla et al. [38] developed and evaluated techniques for identifying improved initial representatives for K-means. Ahmad and Day [39] proposed a k-mean clustering algorithm for mixed numeric and categorical data. Lühr and Lazarescu [40] used incremental connectivity based representative points for clustering of dynamic data streams. Jiang et al. developed class consistent k-means for application to face and action recognition [41]. An improved K-means based method for fingerprint segmentation with sensor interoperability was proposed by Yang et al. [42]. Ahmad and Dey [43] developed a k-means type clustering algorithm for subspace clustering of mixed numeric and categorical datasets.

During the whole clustering process, starting from pre-processing to cluster evaluation, several decisions have to be made by the user (Milligan and Cooper, [44]). Some of them are: decisions for choosing a distance measure, or how to initialize representatives. Following the ideas from [28], we name the common flexibilities of algorithms *sub-problems*, and these are the spots in algorithms where a solution has to be chosen. For each sub-problem there are generally plenty solutions. Table 1 shows common clustering sub-

Table 1
Sub-problems (flexibilities) in representative-based algorithms and proposed solutions.

Sub-problem	Solutions and abbreviations	Reference
Initialize representatives	Randomly (RANDOM)*	Lloyd [62]
	Like in an SPSS k-means cluster analysis algorithm (SPSS)	Hartigan [63]
	Using a hierarchical divisive clustering algorithm (DIANA)	Kaufman and Rousseeuw [64]
	Using hierarchical clustering with binary divisions on opposite sides of the cluster mean on a randomly chosen direction (XMEANS)	Pelleg and Moore [65]
	Using hierarchical clustering with binary divisions based on data's main principal component and corresponding eigenvalue (GMEANS)	Hammerly and Elkan [66]
	Using hierarchical clustering with binary divisions based on principal components analysis (PCA)	Ding and He [70]
	With a distance-based probabilistic distribution (KMEANS++)*	Arthur and Vassilvitskii [71]
Measure distance	With euclidean metric (EUCLID)*	e.g. Xu and Wunsch [2]
	With chebyshev metric (CHEBY)	e.g. Xu and Wunsch [2]
	With city block metric (CITY)	e.g. Xu and Wunsch [2]
	With correlation similarity (CORREL)	e.g. Xu and Wunsch [2]
	With cosine similarity (COSINE)*	e.g. Xu and Wunsch [2]
Update representatives	With arithmetic mean (MEAN)*	Hartigan and Wong [72]
	With median (MEDIAN)	Kaufman and Rousseeuw [64]
	With online processing with a decay learning rate dependent on the number of algorithms iterations (KOHONEN)*	Kohonen [73]
	With online processing with a learning rate dependent on the number of already assigned objects to clusters (BOTTOU)	Bottou [74]
	With rival-penalizing online processing (KSTAR)	Cheung [75]
Evaluate clusters	With the silhouette index (SILHOU)	Rousseeuw [76]
	With compactness, i.e. intra-cluster distance (COMPACT)*	Kaufman and Rousseeuw [64]
	With the XB index (XB)	Xie and Beny [77]
	With connectivity (CONN)	Ester et al. [78]
	With min–max cut (MMC)	Ding et al. [79]
	With Bayesian information criteria (BIC)	Pelleg and Moore [65]
Stop criterion	Representative stability (REPSTAB)	Hartigan and Wong [72]
	Number of iterations (NITER)*	Bottou [74]
	Cluster evaluation threshold (CETHR)	Pelleg and Moore [65]
	Time (TIME)	Kohonen [73]
	Membership stability (MEMBERS)*	Cheung [75]

problems and some popular solutions from literature. Abbreviations for RCs that will be used in Sections 3 and 4 are shown in brackets.

Besides solutions presented in Table 1, in the literature there are many more solutions for each sub-problem. For example, for "Initialize representatives" Astrahan [45] proposed searching for partitions with well-separated objects with a lot of neighbors, Bradley and Fayyad [46] use bootstrapping to find stable initial partitions, Faber [47] defines a method for choosing representatives from high density regions, Stanley [48] assigns objects to clusters with equal probabilities, Mirkin [49] applied Max–Min procedure, Belal and Daoud [50] propose using medians of the highest variance attribute, Maitra [51] generates a large number of local sub-clusters and obtains representatives from the most separated ones, etc. The proposed solutions from Table 1 can be formalized as RCs and further reused in clustering algorithms. Reusing solutions is not a new idea. According to Freeman [52], RCs are usually source code fragments, but they can also include design structures, module-level implementation structures, specifications, documentation or transformations.

Although being a standard concept in software development (Frakes and Kang, [53]), the idea of RC design is not common in data mining today. Nevertheless, the idea of component-based (pattern) approach for design of data mining algorithms was presented in Delibasic et al. [54]. In clustering algorithm design, some efforts toward RC design are already made. E.g. Tajunisha and Saravanan [36] proposed a model for improving K-means algorithm with the flexibility of choosing different initialization techniques and distance measures. Lühr and Lazarescu [55] describe incremental clustering of dynamic data streams using connectivity-based representative points. A component-based approach for automatic feature selection was suggested by Mierswa and Morik [56]. A systematic evaluation of initialization methods for K-means is presented in Peterson et al. [57]. Xiong et al. [58] did a detailed analysis of clustering evaluation measures with respect to data distribution. Furthermore, there are several studies that are benchmarking K-means as algorithms that are differing in one or more parts (e.g. Altun et al. [59]). Following the clustering process proposed by Milligan and Cooper [44], Walesiak and Dudek [60] developed the "clusterSim" R package to determine the optimal clustering procedure for a data set by varying all combinations of distance measures, data scaling, and clustering algorithms. Achtert et al. [61] propose the ELKI framework which has flexibilities in working with arbitrary data types and using several distance and similarity measures for the given data types.

Still, all flexibilities mentioned above, are designed and evaluated on only a few sub-problems (e.g. representatives initialization or distance measure selection), while the other parts of algorithm were fixed. The framework proposed in this paper allows algorithm design by component interchange from all sub-problems from Table 1 (components from [28] are marked with *). Available

Table 2
Analyzed open-source software.

Software	Reference
RapidMiner	Mierswa et al. [80]
Weka	Hall et al. [81]
Tanagra	Rakotomalala [82]
Orange	Demsar et al. [83]
R	R Development Core Team [84]
KNIME	Berthold et al. [85]
JAVA-ML	Abeel et al. [86]
CLUTO	Karypis [87]

clustering algorithm flexibilities can be also analyzed from the aspect of data mining software. We, therefore, analyzed open-source (see Table 2) as well as some commercial software (IBM SPSS Modeler, Matlab).

A thorough analysis of the aforementioned software would be out of the scope of this paper. Therefore, we only give some general conclusions about them. All analyzed software offers various numbers of representative-based clustering algorithms.

JAVA-ML has several K-means-like algorithms (e.g. original K-means, K-medoids), offering flexibilities in distance measures, and various evaluation measures. Orange provides several distance measures for K-means. Tanagra implements flexibilities in using different "Update representatives" methods, namely "batch" (Forgy) and "online"[1] (MacQueen). Weka offers flexibilities for choosing different distance and similarity measures. It only allows combining EUCLID with MEAN, and automatically uses MEDIAN for other distance measures. RapidMiner does not provide much flexibility for K-means, but instead offers several distance measures for K-medoids that can be for numerical, categorical or for mixed data types. The authors of RapidMiner and Weka didn't allow combining the original K-means with other distance measures, as probably they are well aware that MEAN optimizes intra-cluster distance just for EUCLID metric space. Still, COSINE shows good performance when combined with MEAN for document clustering [8]. In experimental results we also showed that usage of MEAN with different distance measures from EUCLID can lead to good algorithm performance (Section 4). Cluto provides flexibilities in choosing various distance and similarity measures, where COSINE distance is the default option. It also offers hierarchical divisive binary partitioning, as well as different evaluation functions. R offers four different versions of K-means (Lloyd, MacQueen, Forgy, Hartigan and Wong).

Matlab gives several possibilities to initialize representatives and to handle empty clusters in K-means. More interestingly, Matlab uses "batch" and "online" clustering, but uses them sequentially, i.e. it first executes "batch" K-means, and afterwards performs "online" K-means on representatives received from "batch" K-means. On the other hand, IBM SPSS Modeler does not offer any flexibilities in K-means, i.e. only the number of iterations can be changed.

The general conclusion is that clustering algorithms, available in analyzed software, offer some flexibilities, especially a lot of distance measures. We conclude that algorithm flexibilities are not standardized, and that there isn't a software where all flexibilities, as identified in [28], can be combined in one algorithm. Still, in this paper, we propose such an architecture and show its benefits in Section 4.

3. Proposed clustering design architecture

The aim of our research was to extract interesting solutions for common sub-problems and encapsulate them as RCs that can be reused, for design of other algorithms. Formal design and implementation of RCs proposed in this paper comply with the definitions of reusable (software) components given by [27].

By defining RCs (Table 1) and a generic cluster (GC) algorithm (Section 3.4) that assembles RCs to algorithms, several flexibilities for representative-based clustering algorithms are achieved. Users can thus decide on several available RCs for resolving each clustering sub-problem.

3.1. General component-based algorithm architecture

To enable collaborative development (as suggested by Sonnenburg [22]) and usage of the proposed RC-based architecture, the framework should be easily extensible with RCs and generic algorithms. A proposal of such a framework can already be found in Vukicevic et al. [88] and on the WhiBo project homepage: www.whibo.fon.bg.ac.rs. Integration of architecture presented in this paper with WhiBo framework will be the subject of our future work. This should enable user-friendly algorithm design through GUI and hopefully integrated efforts in algorithm development from RapidMiner and WhiBo community.

The proposed architecture of the RC-based framework is presented with a layer diagram in Fig. 1. Layers are formed on different levels of abstraction and are independent. This allows easy extension of the proposed framework.

[1] "Batch" clustering calculates representatives after all instances have been assigned to clusters, while "online" clustering calculates representatives each time an instance has been assigned to a cluster.

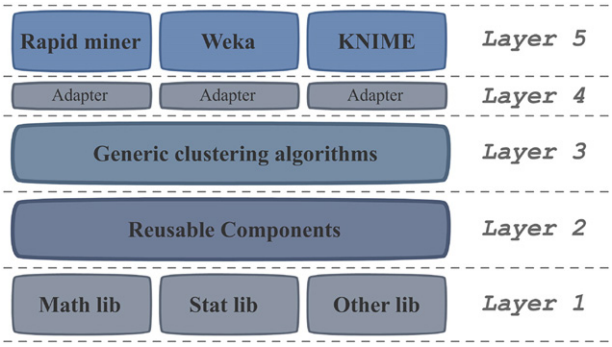

Fig. 1. The general architecture layer diagram of a component-based algorithm framework.

Layer 1 represents a repository of standard mathematical, statistical and other methods and operations (e.g. matrix calculus, statistical tests, etc.). RCs are stored in Layer 2. RCs can use methods from Layer 1, and are shaped to fit common sub-problem I/O structure (Table 3). The RCs are classified into different sub-problems (Fig. 4). Layer 3 stores generic algorithms that can assemble RCs to algorithms. Generic algorithms provide algorithm models that could be used in various software solutions. Layer 4 includes adapters that enable using generic algorithms in different softwares (Layer 5).

Fig. 2 shows a package structure that supports independence between the RC repository, generic algorithm and GUI. The GUI package should be designed to automatically recognize newly added RCs, and present it to the user. The "generic algorithm" package contains an RC repository and generic algorithms constructed to make specific clustering algorithms using RCs. The "algorithm definition" package should allow a complete description of a specified component-based algorithm, including choice of RCs and their parameter settings.

A user defines an algorithm through the GUI by assigning an RC for each sub-problem (step 1). After the algorithm is defined, it is recorded as an "algorithm definition", e.g. inside an XML file (step 2). Algorithm execution starts with call of generic algorithm (step 3), and concrete algorithm logic is defined when generic algorithm instantiates RCs from "algorithm definition" (step 4).

This separates the logic of the GUI from the domain logic (code implementing the generic algorithm), which is presented in Fig. 2. This way, the architecture is open for contributors to add new RCs for solving certain sub-problems and thus has the ability to evaluate and compare them with other algorithms.

3.2. Sub-problems and reusable components

Component-based design of algorithms asks for a *sub-problem/reusable-component* structure like in object-oriented software design. Standardized sub-problems with defined inputs and outputs (Table 3) and RCs (following these standards) enable exchanging various RCs in generic algorithms. We propose a sub-problem I/O structure in Table 3.

3.2.1. Initialize representatives

(IR) gets a dataset as an input and returns a set of cluster representatives as output (Table 3). This sub-problem allows using RCs from non-representative-based clustering algorithms (DIANA, PCA, etc.). E.g., the DIANA algorithm originally returns only object memberships, and therefore representatives must be calculated to use DIANA as an RC. In order to be consistent, the GC algorithm uses the selected "Update (Create) Representatives" RC to calculate representatives from the partitions identified by non-representative-based clustering algorithms. Milligan [89] proposed such hybridization of K-means algorithm with Ward's hierarchical initialization [90], and we extend this approach with following IR components originating from hierarchical algorithms: DIANA, GMEANS, PCA, and XMEANS.

Table 3
Sub-problems input/output (I/O) structure.

Sub-problem	Input	Output
Initialize representatives (IR)	Dataset	Representatives
Measure distance (MD)	Dataset with two objects	Distance
Update (Create) representatives (UR)	Dataset, representative (dataset)	Representative (representative)
Evaluate clusters (EC)	Cluster model (dataset, membership), representatives	Evaluation
Stop criterion (SC)	Algorithm state	Yes/no

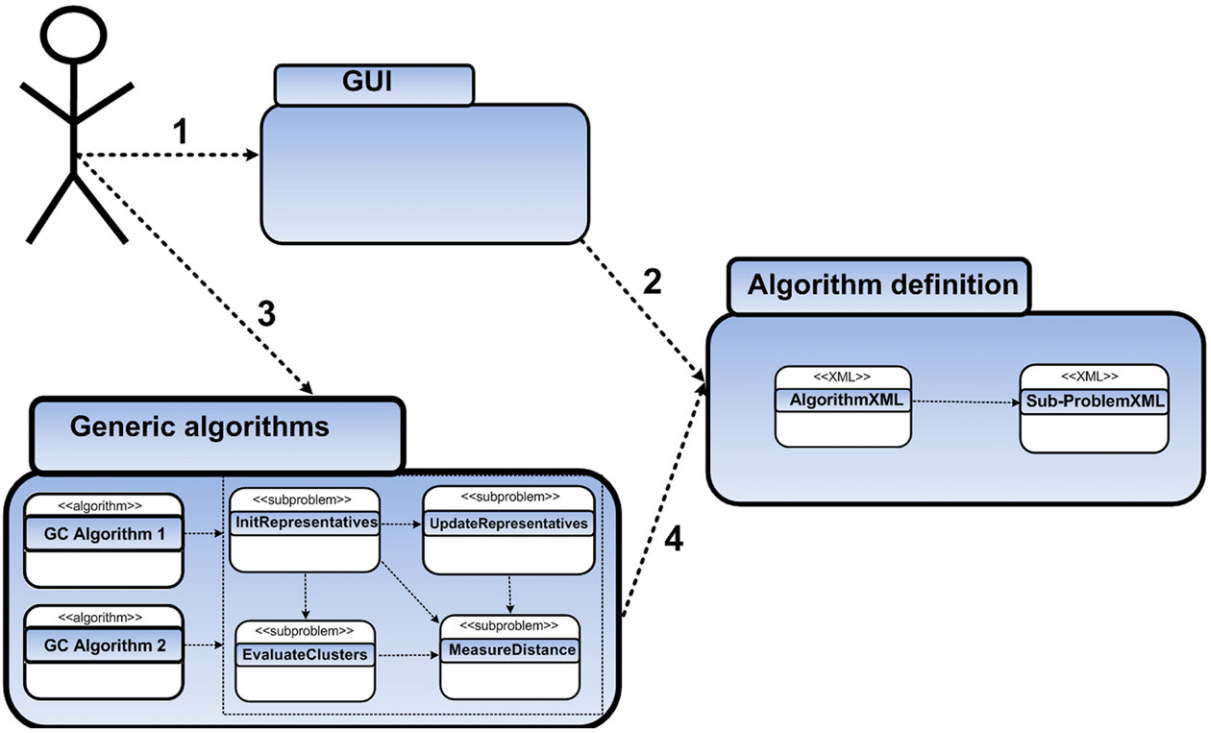

Fig. 2. Independence of GUI from domain logic code and algorithm definition.

3.2.2. Measure distance

(MD) takes only two instances as an input and returns the distance (similarity) between those. RCs from this sub-problem are used throughout the algorithm in all RCs using MD.

3.2.3. Update (create) representatives

(UR) receives cluster representatives and a cluster model (data and membership) as an input and returns updated (or new) representatives. In case that "Initialize representatives" did not create representatives (just assigned memberships to dataset), the *create()* function (Fig. 2) is called to calculate cluster representatives.

3.2.4. Evaluate clusters

(EC) gets representatives and a cluster model as an input, and evaluates the model. RCs for this sub-problem are implementations of internal cluster evaluation measures. In the proposed GC algorithm (Algorithm 2 in Section 3.4), "Evaluate clusters" is used to make decisions on choosing the best model from the models generated through algorithm iterations and restarts. This way the best model, by a selected evaluation measure, is returned, and not the last one [91]. Furthermore, Fig. 4 shows that selected RCs for EC are used throughout the algorithm to choose best clustering models (e.g. in "Initialize representatives" all hierarchical RCs, like DIANA, PCA, XMEANS, and GMEANS use the "Evaluate cluster" RC to determine which cluster to split next).

3.2.5. Stop criteria

This follows a current algorithm state (e.g. NITER, REPSTAB, TIME etc.) and returns a signal for stopping generic algorithm execution.

The aforementioned sub-problems interact in the GC algorithms. Interactions between sub-problems are explained in the next subsection.

3.3. Generic clustering algorithm structure

The class diagram of the generic clustering algorithm (Level 3 in Fig. 1) is displayed in Fig. 3. The GC algorithm includes a composition of sub-problem interfaces. Every sub-problem (interface in object-oriented design) is implemented with a solution (RC). Standardized I/O structure (Table 3) allows implementation and reuse of RCs without changing the GC algorithm implementation.

In analyzed open-source software, we often found that distance measures are being used inconsistently throughout the algorithm. For example, some implementations allow choosing different distance measures as algorithm parameters, but selected distance

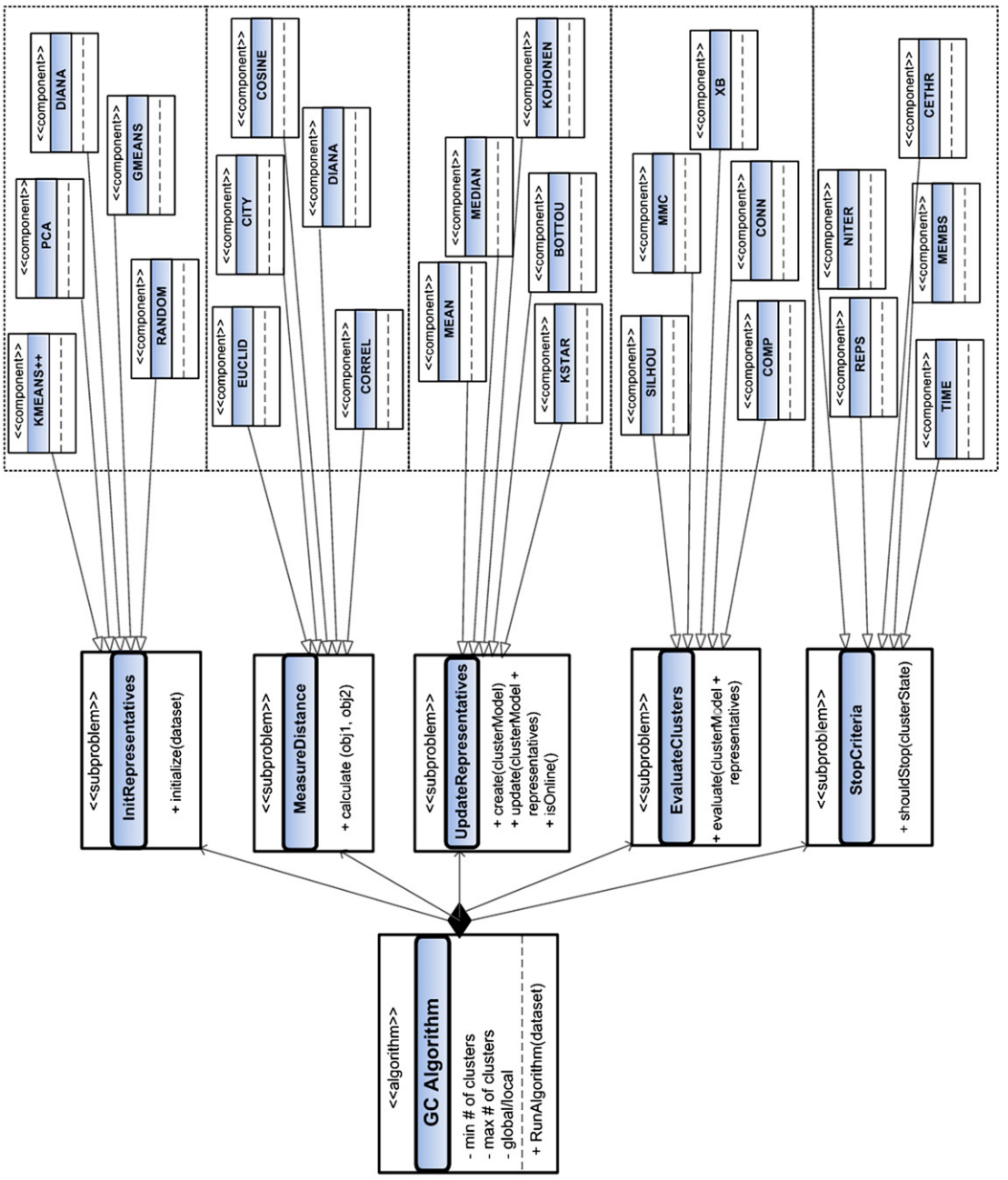

Fig. 3. Class diagram of generic clustering algorithm.

measure is used only in the "Update representatives" phase, but in "Initialize representatives" and "Evaluate clusters" some other distance measure is used (most often Euclidean distance). This can be attributed to non-existence of common code (RC) repository which produces the need for repetitions of code in collaborative development, and can lead to inconsistencies.

GC algorithm structure is solving this problem by providing a common code (RC) repository and abstraction of RCs on the sub-problem level. Common code repository allows keeping the code in one place and reduces the need for the repetition of code fragments and by that lowers the chance for errors and inconsistencies in implementations. Abstraction of RCs on the sub-problem level allows consistent interaction in usage of RCs throughout the algorithm (e.g. if the RC COSINE for "Measure Distance" is selected, it will be also used in "Initialize representatives", "Update representatives", and "Evaluate clusters") (Fig. 4).

3.4. Generic clustering algorithm flow

We propose a generic clustering algorithm that resembles the famous K-means algorithm structure. Pseudo-code of the original K-means algorithm designed with RCs described in Table 1 is shown in Algorithm 1.

Algorithm 1. Original K-means algorithm.

Input:
Dataset
Output:
Cluster model (Dataset with memberships) and cluster representatives
Parameters:
K: desired number of clusters
GCAlgorithm(Dataset, K)

```
1.  // Initialization
2.  "Initialize representatives" with RANDOM to initialize K representatives
3.  // Refinement
4.  repeat
5.      for each instance from dataset
6.          for each representative
7.              "Measure distance" with EUCLID (instance, representative)
8.          end
9.      end
10.     Assign instance to nearest representative by calculating mean
11. until REPSTAB "Stop criterion"
```

Based on the idea of component-based design, we propose a generic clustering algorithm (GC) that defines the generic flow for assembling RCs into algorithms. The algorithm is shown in Algorithm 2 and consists of three phases:

1. Initialization, where initial (minK) representatives are generated;
2. Refinement, where objects are iteratively assigned to representatives, and representatives recalculated; and
3. Hierarchical division, where the clusters are being split binary, until the desired number (maxK) of clusters has been achieved.

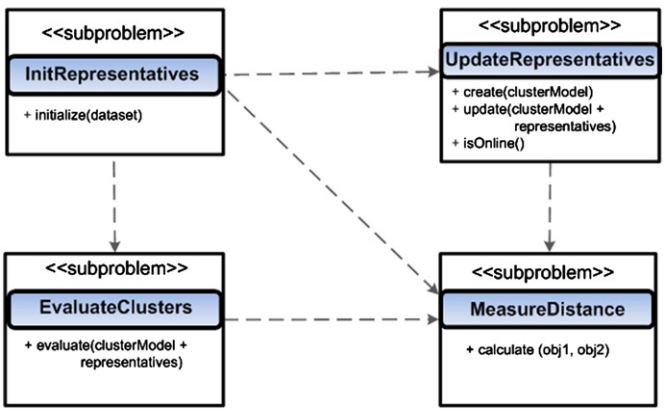

Fig. 4. Interactions between sub-problems.

Algorithm 2. Generic representative-based clustering algorithm.

Input:
Dataset
Output:
Cluster model (Dataset with memberships) and cluster representatives
Parameters:
minK: initial number of clusters
maxK: desired final number of clusters through hierarchical division
refinePartitions (local/global): refinement method in Hierarchical division

```
GCAlgorithm(Dataset, minK, maxK, refinePartitions)
1.  // Initialization
2.  Use "Initialize representatives" to initialize min_K representatives
3.  // Refinement
4.  repeat
5.      for each randomly (without replacement) sampled instance from dataset
6.          for each representative
7.              "Measure distance" (instance, representative)
8.          end
9.          Assign instance to nearest representative
10.         If "Update representatives".isOnline() then
11.             "Update representatives".update()
12.     end
13.     If not "Update representatives".isOnline() then
14.         "Update representatives".update()
15.  until"Stop criterion"
16.  // Hierarchical division
17.  If maxK > minK
18.      Split each cluster binary using "Initialize representatives" (minK = 2)
19.      repeat
20.          Do "Evaluate clusters" on child clusters and parent clusters
21.          Choose best difference between child and parent clusters evaluation
22.          If difference is positive
23.              If refinePartitions is local
24.                  Do Refinement (Parent cluster dataset, child centroids)
25.              If refinePartitions is global
26.                  Do Refinement (Whole dataset, all centroids)
27.      until number of clusters = maxK or no splitting in last loop
```

The GC algorithm has three global parameters. These are:

- minK: number of clusters that should be generated in the "Refinement" phase;
- maxK: number of clusters that should be produced after the "Hierarchical division" phase; and
- refinePartitions: defines how clusters should be refined in the "Hierarchical division" phase after a cluster has been split binary. There are two possibilities:
  - local: to refine child clusters based only on data from the parent cluster (as in X-means, [65]), and
  - global: to refine all clusters again, this time with two newly added child representatives (as in Hammerly and Elkan, [66]).

The proposed algorithm efficiently combines the refinement phases from "online" and "batch" [53] K-means like algorithms. This is achieved by randomly sampling the whole dataset which is a must for online clustering, but has no influence on batch clustering. It is also important to define for each "Update representative" RC whether it should be used "online" or "batch". This is accomplished by tagging each "Update representative" RC as "online" or "batch", since this affects how the component is used within the generic algorithm.

The hierarchical division phase is often recommended for finding the right K in K-means (e.g. X-means [65]). It is used whenever the user isn't sure of the right number of clusters, but can give information about the range of K (from minK through maxK).

We transformed several RCs that originally could only be used for K = 2 initialization. We transformed them in order to initialize K representatives by applying a hierarchical binary divisive partitioning procedure. The hierarchical divisive initialization procedure used for DIANA, PCA, XMEANS, and GMEANS RCs is shown in Algorithm 3.

Algorithm 3. Initialize representatives RCs hierarchical divisive algorithm.

Input:
Dataset
Output:
Representatives
Parameter:
K:number of clusters
Hierarchical division (Dataset, K)
1. Binary split (Dataset)
2. **repeat**
3. **for** each cluster
4. parentEC = "Evaluate cluster" (current cluster)
5. childCluster = Binary split (current cluster)
6. children EC = "Evaluate cluster" (childCluster)
7. Do **Refinement** (childCluster dataset, childCluster centroids)
8. **end**
9. Choose best improvement between parent and child clusters evaluation
10. Replace parent cluster with child clusters
11. **until** K clusters

3.5. Generic clustering algorithm complexity

In order to identify the scalability of the generic algorithm we identified the complexity on the level of sub-problems, since complexity also depends on selected RCs for algorithm execution. Complexity of most RCs is described in papers referenced in Table 1.

Partitioning part of algorithm (lines 1–15 of Algorithm 2) executes initialization and refinement phase sequentially. Initialization is executed only once while refinement phase (assigning instances to representatives and recalculation of representatives) is repeated until stop criterion is met. After this phase minK clusters are identified. Hierarchical divisioning part of algorithm (lines 16–27 of Algorithm 2) is repeating the procedure described above on every cluster (with fixed K = 2) until maxK of clusters is created or until there is no improvement of the cluster model quality. In every step divisive part of the algorithm divides every cluster on two partitions. Child clusters that improve overall cluster model quality the most are kept and refined (after every partitioning number of clusters is increased by 1). So, maximum execution time of the whole algorithm is:

$$\text{time}(\text{Init}+\text{Refine})+\frac{\text{maxK}-\text{minK}}{2}*(\text{maxK}+\text{minK})*\text{time}(\text{Init})+(\text{maxK}-\text{minK})*\text{time}(\text{Refine}).$$

It is important to notice that complexity of the generic algorithm on the level of sub-problems does not exceed the complexity of X-means and G-means algorithm, which have similar division strategies.

With respect to the number of instances, complexity of initialization and refinement is linear or quadratic, when using RCs described earlier. Therefore, the overall complexity of the generic algorithm does not exceed quadratic complexity, and often has linear complexity. On the other side, a lot of efforts were invested to lower the complexity and to allow scalability of existing clustering algorithms e.g. [67–69], so these improvements could be integrated as RCs in the generic framework.

4. Experimental evaluation

The aim of the experiments was to provide some evidence whether component reuse and interchange of good ideas between algorithms can lead to performance improvement of representative-based clustering algorithms as suggested in [22,23]. We designed 84 RC-based clustering algorithms for experimental evaluation. These algorithms were made by combining RCs from 3 sub-problems (A more detailed characterization of these RCs is given in the Appendix of this paper). These are:

- Initialize representatives: DIANA, GMEANS, KMEANS++, PCA, RANDOM, SPSS, XMEANS
- Measure distance: CITY, CORREL, COSINE, EUCLID
- Update representatives: BOTTOU, MEAN, MEDIAN.

The "Evaluate cluster" and "Stop criterion" RCs were constant in all algorithms, namely COMPACT (K-means default) and MEMBERS, respectively. The algorithms were tested on 10 UCI datasets [92]. These datasets and their basic characteristics are shown in Table 4. We set K equal to number of classes in datasets as it is common in literature [93,94]. All algorithms were run 10 times and the reported results are shown on average.

Besides testing the main hypothesis, i.e. whether a newly built RC-based clustering algorithm can lead to better results on a given dataset than common well-known clustering algorithms, we also examined the influence of one RC and combinations of RCs on each dataset and on all datasets on average. All algorithms were evaluated with silhouette index value [76] because it is widely used, fairly simple and can be used for easy comparison of clustering quality between different datasets. Silhouette index takes values between

Table 4
Datasets used in experiments.

No.	Dataset	Records	Attributes	Classes
1	breast	683	9	2
2	dermatology	358	34	6
3	german	1000	20	2
4	glass	214	9	6
5	heart	297	13	5
6	ionosphere	351	33	2
7	iris	101	16	3
8	segmentation	210	19	7
9	soybean	266	35	15
10	vowel	990	10	11

Table 5
Comparison of component-based with well-known algorithms.

Algorithm/ dataset	K-means (RAND-EUCLID-MEAN)	K-medians (RAND-EUCLID-MEDIAN)	K-means++ (KMEANS++ EUCLID-MEAN)	G-means (GMEANS-EUCLID-MEAN)	Best RC-based alg.	Worst RC-based alg.	Best-Worst
Breast	0.59	0.59	0.59	0.59	0.65 (21)	0.22	0.43
Dermatology	0.30	0.15	0.32	0.35	0.62 (1)	0.13	0.49
German	0.37	0.18	0.26	0.45	0.53 (3)	0.02	0.51
Glass	0.42	0.22	0.44	0.46	0.72 (3)	0.21	0.51
Heart	0.30	0.19	0.32	0.32	0.54 (1)	0.14	0.40
Ionosphere	0.33	0.37	0.36	0.37	0.55 (5)	0.20	0.35
Iris	0.62	0.60	0.61	0.62	0.82 (2)	0.53	0.29
Segmentation	0.40	0.15	0.40	0.45	0.64 (3)	0.05	0.59
Soybean	0.20	0.16	0.21	0.24	0.40 (1)	0.09	0.31
Vowel	0.41	0.40	0.42	0.42	0.69 (1)	0.25	0.44

−1 and 1, where values closer to 1 indicate that objects are soundly clustered. By using only one evaluation measure, comparison of algorithms is adequate and consistent, as explained by Estivill-Castro [95]. For other evaluation measures best algorithms and components could vary.

4.1. Comparison with well known algorithms

In the first experiment we compared newly developed RC-based algorithms with well-known algorithms that can be reconstructed by combining RCs. The benchmark algorithms were K-means, K-medians, K-means++ and G-means (as partitioning algorithm). In this way, we provided a fair testing environment as stated in [22] (on the same platform, where all components had the same implementation). Table 5 shows silhouette values for ten datasets.

Last three columns in Table 5 show silhouette values for best algorithms (number in brackets indicates the number of algorithms with the same, best result), worst RC-based algorithms and their difference, respectively.

As hypothesized, newly developed RC-based algorithms gave better results on every analyzed dataset than original algorithms did. Differences in silhouette values and performance of well-known algorithms compared to the best and the worst RC-based algorithms are visualized in Fig. 5.

Run times of popular algorithms as well as best and worst RC-based algorithms from Table 5 are shown in Table 6. Although RC-based algorithm can, in general, achieve better silhouette index values, the trade-off is that RC-based algorithms sometimes have increased run time, and this is certainly an important aspect in RC-based design. On the other hand, one can always add to the generic framework RCs that are computationally more effective, and have a good trade-off between quality of clustering and run time.

It is interesting to mention that on most datasets more than one RC-based algorithm have shown equally best performance on silhouette value. Best algorithms on every dataset are shown in Table 7.

We identified that for some datasets only one sub-problem influenced the performance difference. On *breast* dataset 21 algorithms gave the best result of silhouette index value (0.65). Each of those algorithms had CITY for distance measure. This indicates that this dataset silhouette value didn't depend on other two sub-problems. This is also verified in the results from Table 8 in the next subsection.

COSINE and CORREL showed good results on several datasets. For *ionosphere* dataset there were 40 best algorithms that contained COSINE or CORREL with performance between 0.55 and 0.56. On *heart* dataset the best algorithm was DIANA-CORREL-MEAN (0.54) but 17 more algorithms had the same silhouette value (SV)[2] greater than 0.50. Each of those included COSINE or CORREL distance measures. On *soybean*, these distance measures gave the best results on 15 algorithms with SV between 0.36 and 0.40. These are good

[2] Note that there is a distinction between SILHOU and SV abbreviations, as SV is used on a clustering model to evaluate its goodness, whereas SILHOU is used as an RC in the GC algorithm during clustering model building.

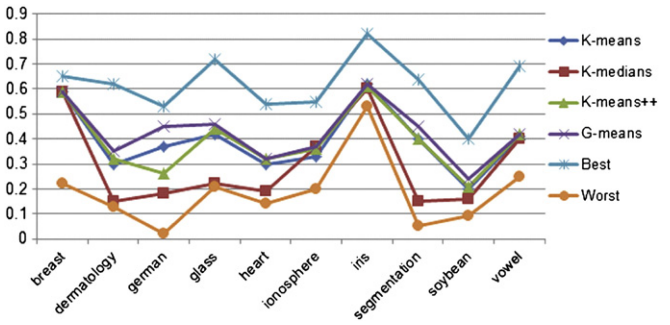

Fig. 5. Visualization of silhouette value differences of well-known algorithms compared to best and worst RC-based algorithms.

results since the best well-known algorithm from Table 5 had 0.24. Dataset *dermatology* had 4 best algorithms that contained COSINE, with SV = 0.62. Good performance of COSINE is not surprising because of good metric properties, Zhu et al. [96].

On the other hand, on some datasets, an RC combination produced better results, i.e. when those RCs were used individually in algorithms the performance was much worse. For example on *german* dataset there were three best algorithms (Table 7). In all three cases DIANA and EUCLID were part of the best algorithm. Since the next best algorithm with DIANA RC had SV = 0.47, we can conclude that for this dataset the difference was made because of synergy of RCs. Similar behavior is spotted on *dermatology* where PCA and COSINE gave the best result (in 3 algorithms) for every "update representatives" RC. On *iris* dataset the synergy of COSINE and BOTTOU produced the two best algorithms. On *segmentation* dataset, SPSS and EUCLID gave the best results for all three "update representatives" RCs (SV = 0.64). The next best algorithm with SPSS initialization had SV = 0.58 and the next best algorithm with EUCLID had SV = 0.54. Therefore, we conclude that these RCs work best in pair on this dataset.

4.2. Detection of well performing components

To test whether an RC makes a significant impact on algorithm performance, we used the following procedure. First, we picked a sub-problem (e.g. "Measure distance"), and made groups of algorithms differing in RCs of that sub-problem. Then we tested if differences between those groups were significant, using Wilcoxon signed-rank paired test with 95% confidence. A paired test is appropriate because we can pair algorithms from different groups that differ only in the components we test, while the rest of the algorithm is the same. Next, we separated RCs in two groups, "best" and "rest", similar to Wijaya et al. [97]. All the components are in the "best" group, unless they are proved significantly worse than the best RC. Components in the "best" group are recommended for use in an RC-based algorithm that should cluster the selected dataset in a good way. The "best" RC groups for each sub-problem are shown in Table 8.

Several interesting patterns emerge. First, we see that, for some datasets, one RC performed better than all the others (boldfaced in Table 8). These RCs are best recommendation for these datasets. For example, on *breast* dataset, CITY distance measure was significantly better than the other distance measures, but we could not differentiate on the best "initialize representatives" RC. In this way, we pointed out good parts of an algorithm which constitute a well-performing algorithm.

On the other hand, sometimes no clear distinction could be made on which RCs are good for a dataset, which is the case with IR sub-problem for *breast*, *iris* or *segmentation* datasets. This means IR RCs haven't consistently made an algorithm part of "best" algorithms for these datasets.

However, this describes an RC's average behavior, and does not say that a particular RC could not constitute a good algorithm if combined with specific RCs from other sub-problems. This dependence and synergy of specific RCs are partially explored in the next subsection.

Table 6
Comparison of component-based with well-known algorithms by run time (in seconds).

Dataset/ algorithm	K-means (RAND-EUCLID-MEAN)	K-medians (RAND-EUCLID-MEDIAN)	K-means++ (KMEANS++ EUCLID-MEAN)	G-means (GMEANS-EUCLID-MEAN)	Best RC-based alg. by silhouette index
Breast	0.41	0.28	0.7	1.02	0.24
Dermatology	0.21	0.11	0.41	0.52	0.74
German	0.63	0.38	0.65	0.96	0.95
Glass	0.22	0.15	0.6	0.7	0.58
Heart	0.13	0.09	0.34	0.35	0.36
Ionosphere	0.24	0.10	0.73	0.37	0.53
Iris	0.18	0.13	0.29	0.49	0.18
Segmentation	1.54	0.56	2.71	3.16	2.29
Soybean	1.98	0.59	3.69	3.27	2.86
Vowel	1.95	0.56	3.37	4.15	3.76

Table 7
RC based algorithms that had the best (equal) silhouette index value on a dataset.

Dataset	Best algorithms
Breast	All algorithms that contain CITY distance measure
Dermatology	PCA-COS-MEDIAN
German	DIANA-EUCLID-MEDIAN, DIANA-EUCLID-BOTTOU, DIANA-EUCLID-MEAN
Glass	SPSS-COSINE-MEDIAN, SPSS-COSINE-MEAN, SPSS-COSINE-BOTTOU
Heart	DIANA-CORREL-MEAN
Ionosphere	SPSS-COSINE-MEDIAN, DIANA-COSINE-MEDIAN, SPSS-COSINE-MEAN, GMEANS-COSINE-BOTTOU, PCA-COSINE-MEAN
Iris	GMEANS-COSINE-BOTTOU, DIANA-COSINE-BOTTOU
Segmentation	SPSS-EUCLID-BOTTOU, SPSS-EUCLID-MEDIAN, SPSS-EUCLID-MEAN
Soybean	PCA-CORREL-MEAN
Vowel	DIANA-CORREL-MEDIAN

Further, we analyzed the effect of employing and taking one component away on average for all datasets. Fig. 6 shows silhouette values for every "Initialize representatives" and "Distance measure" RC, where "Update representatives" is fixed. It can be seen that employing DIANA initialization produces the best results for any distance measure used. So, for the fixed algorithm, removing this RC leads to worse results. Another interesting thing is that using CITY or EUCLID RCs reduces algorithm performance for any combination of RCs. Similar results were obtained for MEAN and ONLINE fixed so we don't show these two graphs (complete results are available upon request).

The influence of employing or taking one component away, however, needs more thorough examination. One of the advantages of the component-based design approach is that it gives further possibilities for more detailed analysis of RC individual and synergic performance on a dataset which is analyzed in the following subsections.

4.3. Average performance of algorithms

Using the procedure described in Section 4.2, we identified the "best" and "rest" groups based on average performance on 10 datasets. We explored why algorithms were tagged as "best" or "rest" on average.

We used a decision tree algorithm and an association rule algorithm to extract rules for classifying an algorithm as "best" or "rest". Our findings are given in Fig. 7 and Table 9. For example, Rule 1 from Fig. 6 shows that algorithms with Update Representatives (UR) = MEAN or BOTTOU, Initialize representatives (IR) = DIANA or GMEANS or PCA or SPSS, and Measure distance (MD) = CITY or COSINE or EUCLID perform as "best" on average with a 100% confidence.

The first four columns of Fig. 7 are ordered by attributes selected for tree growth. It is interesting to notice that MEAN and BOTTOU behave more similarly in comparison to MEDIAN. Algorithms containing MEAN and BOTTOU have 40/56 = 71% of "best" algorithms, where algorithms with MEDIAN only have 4/28 = 14% of "best" algorithms. In that case, only DIANA combined with MEDIAN (Rule 7) produced "best" algorithms.

In Table 9 nine association rules are shown with confidence greater than 80%. For example, Rule 3 shows that utilization of DIANA produces mostly "best" algorithms (confidence 92%), while using RANDOM (Rule 8) results in "rest" algorithms (confidence 83%). CITY + BOTTOU (Rule 1) and CITY + MEAN (Rule 2) showed a synergic effect producing always the "best" algorithms, where their individual usage didn't guarantee a "best" algorithm.

4.4. Searching for the right number of clusters (K)

In this section we explore the capabilities of the hierarchical division part of the generic clustering algorithm. We set minK to 2, and maxK to 2K.

Compared to previous experiment, besides using COMPACT for "Evaluate clusters", we also used BIC (Bayesian Information Criteria), and also included "global" and "local" strategies for division part of algorithm, resulting in a total of 336 algorithms. This was

Table 8
Groups of "best" components for each sub-problem that showed significantly better performance on each dataset.

Dataset	Initialize representatives (best)	Measure distance (best)	Update reps (best)
Breast	DIANA, GMEANS, KMEANS++, PCA, SPSS, XMEANS	**CITY**	**MEAN**
Dermatology	DIANA, PCA, SPSS	COSINE, CORREL	**MEAN**
German	**DIANA**	EUCLID, CITY, COSINE	BOTTOU, MEAN
Glass	DIANA, PCA, SPSS	**COSINE**	BOTTOU, MEAN
Heart	DIANA, KMEANS++, PCA, RANDOM, SPSS	CORREL, COSINE	MEAN, BOTTOU
Ionosphere	DIANA, GMEANS, KMEANS++, PCA, Spss	**COSINE**	MEAN, MEDIAN
Segmentation	DIANA, GMEANS, KMEANS++, PCA, SPSS, XMEANS	CORREL, COSINE	**MEAN**
Soybean	**DIANA**	COSINE, CORREL	**MEAN**
Vowel	DIANA, SPSS	COSINE, EUCLID, CORREL	**MEDIAN**
Iris	SPSS, DIANA, GMEANS, KMEANS++, PCA, RANDOM, XMEANS	**COSINE**	MEAN, MEDIAN

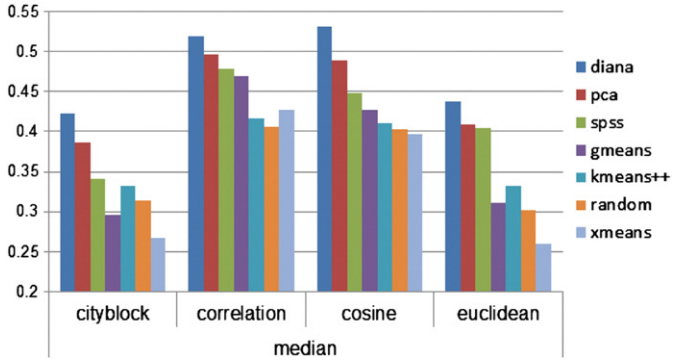

Fig. 6. Visualization of silhouette value average on all datasets.

	UR	IR	MD	UR	Type	No. Alg.	Confidence
Rule 1	MEAN BOTTOU	DIANA GMEANS PCA SPSS	CITY COS EUCLID		BEST	24	100%
Rule 2			COR	MEAN	BEST	4	100%
Rule 3				BOTTOU	REST	4	75%
Rule 4		KMEANS+ + RANDOM XMEANS	CITY		BEST	6	100%
Rule 5			COR COS		REST	12	92%
Rule 6			EUCLID		BEST	6	50%
Rule 7	MEDIAN	DIANA			BEST	4	100%
Rule 8		GMEANS KMEANS+ + RANDOM XMEANS			REST	16	100%
Rule 9		PCA SPSS			REST	8	63%

Fig. 7. A decision tree for explaining algorithms being classified as “best” or “rest”.

done as there are several proposals in the literature to use BIC instead of COMPACT in hierarchical division (e.g. [65], [66]) as decisions on hierarchical cluster division based on BIC are supposed to be more reliable. We will also discuss results for both local and global refinement strategies of child clusters during hierarchical division.

All algorithms consisting of stochastic RCs were run 10 times and the average K results are reported. As expected, these results were dependent on the RCs an algorithm has.

In Table 10 we show the results for algorithms grouped by “Initialize representatives” RCs on all datasets using the global refinement strategy. We report average and standard deviation of K from all algorithms that use jointly an IR and EC RC. E.g. 3.2 ± 0.89 is the average (with standard deviation) number of clusters found by 12 (3 UR*4 MD RCs) algorithms that use DIANA for “Initialize representatives” and BIC for “Evaluate clusters”. IR RCs have high influence in the hierarchical division part of the generic clustering algorithm, as for every new division IR RCs are used.

On all datasets, except *vowel*, algorithms containing specific IR and EC RCs managed to find the true number of clusters. On *vowel* no algorithm group managed to find the right cluster, and DIANA + BIC proposed the nearest value to K (according to mean

Table 9
Rules for explaining algorithms being “best” or “rest”.

	IR	MD	UR	Type	No. Alg.	Confidence
Rule 1		CITY	BOTTOU	BEST	7	100%
Rule 2		CITY	MEAN	BEST	7	100%
Rule 3	DIANA			BEST	12	92%
Rule 4		EUCLID	BOTTOU	BEST	7	86%
Rule 5		COSINE	MEDIAN	REST	7	86%
Rule 6		CORREL	BOTTOU	REST	7	86%
Rule 7		CORREL	MEDIAN	REST	7	86%
Rule 8	RANDOM			REST	12	183%

Table 10
Average and standard deviation of algorithms' K found with global refinement strategy and different IR RCs and BIC and COMPACT used for "Evaluate clusters". Bolded are values of algorithm groups that managed to find the right number of cluster. In bolded italic are values where right K wasn't found, so the nearest match is highlighted.

Dataset	IR	BIC	COMPACT	Dataset	IR	BIC	COMPACT
breast	DIANA	3.2 ± 0.89	3.92 ± 0.23	*ionosphere*	DIANA	3.25 ± 0.85	3.8 ± 0.57
(k = 2)	GMEANS	2.57 ± 0.57	3.37 ± 0.61	*(k = 2)*	GMEANS	3 ± 0.82	3.02 ± 0.81
	KMEANS++	2.78 ± 0.58	3.23 ± 0.26		KMEANS++	2.69 ± 0.58	2.5 ± 0.45
	PCA	3.21 ± 0.89	4 ± 0		PCA	4 ± 0	4 ± 0
	RANDOM	**2.18 ± 0.21**	2.98 ± 0.45		RANDOM	3.18 ± 0.21	**2.41 ± 0.39**
	SPSS	2.79 ± 0.54	3.04 ± 0.38		SPSS	2.78 ± 0.53	**2.47 ± 0.39**
	XMEANS	**2.09 ± 0.12**	3.99 ± 0.03		XMEANS	3.58 ± 0.24	4 ± 0
dermatology	DIANA	7.65 ± 4.17	10.07 ± 1.96	*iris*	DIANA	4.4 ± 1.77	5.5 ± 0.97
(k = 6)	GMEANS	2.9 ± 0.86	**6.09 ± 3.02**	*(k = 3)*	GMEANS	**2.77 ± 0.84**	4.31 ± 1.29
	KMEANS++	2.89 ± 1.48	3.46 ± 1.44		KMEANS++	**2.55 ± 0.8**	**2.96 ± 0.79**
	PCA	6.85 ± 3.65	9.95 ± 1.02		PCA	4.3 ± 1.7	5.83 ± 0.19
	RANDOM	2.02 ± 0.04	3.25 ± 1.13		RANDOM	2.06 ± 0.07	**3.12 ± 0.96**
	SPSS	2.73 ± 0.77	3.31 ± 0.6		SPSS	**2.8 ± 0.66**	3.23 ± 0.67
	XMEANS	2.01 ± 0.03	11.13 ± 0.77		XMEANS	2.03 ± 0.05	6 ± 0
german	DIANA	3.17 ± 0.86	3.97 ± 0.08	*segmentation*	DIANA	8.94 ± 4.69	11.13 ± 3.83
(k = 2)	GMEANS	**2.2 ± 0.26**	3 ± 0.77	*(k = 7)*	GMEANS	4.83 ± 2.37	**6.68 ± 4.19**
	KMEANS++	2.51 ± 0.62	2.78 ± 0.71		KMEANS++	3.92 ± 0.93	3.4 ± 0.85
	PCA	3.2 ± 0.89	4 ± 0		PCA	10.58 ± 2.12	12.48 ± 1.65
	RANDOM	**2.12 ± 0.16**	2.66 ± 0.61		RANDOM	4.37 ± 0.46	3.28 ± 1.18
	SPSS	**2.48 ± 0.62**	2.61 ± 0.62		SPSS	3.78 ± 0.69	3.3 ± 0.76
	XMEANS	**2.06 ± 0.12**	3.95 ± 0.09		XMEANS	6.14 ± 1.28	13.1 ± 1.21
glass	DIANA	7.73 ± 4.26	9.85 ± 3.04	*soybean*	DIANA	11.71 ± 8.55	**15.08 ± 8.02**
(6)	GMEANS	2.75 ± 1.1	**5.73 ± 3.27**	*(k = 15)*	GMEANS	2.62 ± 0.71	5.84 ± 4.46
	KMEANS++	2.75 ± 1.32	3.26 ± 1.4		KMEANS++	4.15 ± 1.92	4.13 ± 2.28
	PCA	6.5 ± 3.9	9.5 ± 2.74		PCA	8.14 ± 4.72	11.18 ± 4.02
	RANDOM	2.04 ± 0.05	3.27 ± 1.83		RANDOM	3.1 ± 0.89	3.55 ± 2.03
	SPSS	2.83 ± 0.86	3.23 ± 1.04		SPSS	4.08 ± 1.54	3.18 ± 1.68
	XMEANS	2.03 ± 0.07	11.2 ± 0.77		XMEANS	3.16 ± 0.72	20.14 ± 5.84
heart	DIANA	6.72 ± 3.49	8.32 ± 1.83	*vowel*	DIANA	***11.93 ± 7.77***	16.08 ± 6.8
(5)	GMEANS	3.18 ± 1.56	5.53 ± 2.11	*(k = 11)*	GMEANS	3.68 ± 2.22	8.63 ± 7.45
	KMEANS++	3.4 ± 2.2	4.23 ± 2.24		KMEANS++	2.58 ± 0.53	2.18 ± 0.21
	PCA	**5.38 ± 2.54**	7.93 ± 0.78		PCA	13.5 ± 5.97	17.28 ± 5.07
	RANDOM	2.08 ± 0.11	4.15 ± 2.08		RANDOM	2.98 ± 0.57	2.2 ± 0.34
	SPSS	3.31 ± 1.68	4.08 ± 1.86		SPSS	2.82 ± 0.87	2.2 ± 0.29
	XMEANS	2.01 ± 0.03	9.59 ± 0.56		XMEANS	4.33 ± 0.96	18.84 ± 3.44

K retrieved). From Table 9 we see that algorithms using BIC RC more often find the right K, still algorithms using COMPACT are sometimes also capable of finding the right K.

Analysis on the algorithm level revealed that on all datasets some algorithms guessed the right number of clusters. On breast 47 algorithms, on dermatology 5 algorithms, on german 71 algorithms, on glass 6 algorithms, on heart 3 algorithms, on ionosphere 46 algorithms, on iris 28 algorithms, on segmentation 4 algorithms, on soybean 5 algorithms, and on vowel 7 algorithms out of 168 with global refinement strategy managed to detect the desired number of clusters.

When using the local refinement strategy results were somewhat different. On *breast* RANDOM + BIC and XMEANS + BIC found 2.31 ± 0.32 and 2.07 ± 0.09 clusters, respectively. On *dermatology* DIANA + BIC and KMEANS++(+)BIC returned 5.92 ± 2.98 and 5.73 ± 2.8 clusters, respectively. On *german* the following algorithms found the right K: RANDOM + BIC (2.09 ± 0.07) and XMEANS + BIC (2.07 ± 0.12). On *glass* DIANA + BIC (6.2 ± 3.14) and GMEANS + BIC (5.63 ± 2.74) found the right K. On *heart* PCA + BIC (4.77 ± 2.05) and SPSS + BIC (5.44 ± 2.45) found K. On *segmentation* KMEANS++(+)BIC (7.34 ± 1.77) and GMEANS + BIC (7.33 ± 1.66) proposed the right number of K. On *ionosphere*, *iris*, *soybean*, and *vowel* the algorithms didn't manage to detect the right K. On the analyzed datasets it was more beneficial to use the global refinement strategy, as on more datasets algorithms using the global strategy managed to find the right K.

We should, however, look at these conclusions with caution, as analysis on synthetic datasets, where the right number of K is more justified, compared to UCI benchmarks datasets, could give a clearer picture of the algorithms that perform best. However, we argue that through combining RCs one can always find a best suited algorithm for a specific dataset and task (e.g. performing good on silhouette index value, or for detecting the right K, etc.).

5. Algorithm space search

In previous section we showed that component interchange from the original algorithms can produce better results than achieved by original algorithms. Still, manual search or automatic brute search for the best algorithm in such a large space is almost impossible. This implies an important research task: definition of intelligent strategy for automatic algorithm selection in the space of RC based clustering algorithms. This problem is addressed in [98] where evolutionary algorithm (EA) is used for

Gene (sub-problem)	Initialize representatives	Measure distance	Update representatives	Evaluate clusters	Stop criterion
Alleles	RANDOM	EUCLID	MEAN	COMPACT	NITER (10)

Fig. 8. Genotype for K-means algorithm composed from RCs.

automatic search through the space of RC-based decision trees and showed noteworthy performance. Here we show how the approach from [82] could be adopted for the search in the space of RC based clustering algorithms.

Algorithm search space over all possible RC combinations of sub-problems defined in Table 1 counts 5250 algorithms (7*5*5*6*5) and 10,500 (with local/global strategy for hierarchical division). Since there are no constraints in combining clustering RCs, the search space size grows super-linearly with addition of new RCs. Furthermore, as RCs can have parameters as well, this broadens the algorithm search space even more.

Algorithm space search with EA as defined in [98] does not need any prior knowledge about the behavior of the algorithms, but relies on algorithm evaluation for the problem at hand. In case of clustering algorithms fitness function should have some internal clustering evaluation measure. Since clustering evaluation measures over the algorithmic space topology have no known theoretical properties (e.g. linearity, convexity), it is reasonable to use the meta-heuristic approach when searching for the best solution.

Genetic coding of chromosomes proposed in [98], as well as genetic operators, is fully customized to conform to the component-based algorithms. Each sub-problem is represented by a custom gene, whose values (alleles) represent an RC with its parameters (if available). Here we show a possible way of coding RC based clustering algorithms (analogous to [98]). Fig. 8 shows the gene representation of the K-means algorithm reconstructed with the components. It uses RANDOM initialization, measures distance with EUCLID, updates representatives with MEAN, and evaluates cluster quality with COMP and stopping criteria is defined with NITER (number of iterations) with parameter 10. Fig. 9 shows a genotype of an RC-based algorithm.

6. Discussion and outlook

In this paper we proposed architecture for component-based design of representative-based clustering algorithms. Including this architecture in open source data mining frameworks should enable at least three benefits for algorithm designers and software developers:

- An easy way to develop and test partial improvements (RCs) and collaborative development of representative-based clustering,
- Reduced need for re-implementation of representative-based clustering algorithms and their parts,
- Better understanding of the algorithm structure and behavior (e.g. for educational purposes).

Further, the component repository (Table 1) and GC algorithm (Algorithm 2) allow design of 5250 (with a selected division) and 10,500 (with local/global strategy) RC-based algorithms by combining RCs on every sub-problem. This number increases by adding new RCs to the repository. Initial evaluation was conducted where we wanted to show that re-assembling of components from the existing algorithms can produce better algorithms than the ones that RCs originate from (as hypothesized in [22]).Through the experimental section, we made several conclusions:

- RC-based algorithms achieved better results on silhouette index value than original algorithms did on the 10 benchmark datasets.
- Sometimes only one RC made the difference for the “best” algorithm.
- On the other hand, sometimes only the synergy of RCs produced the “best” algorithm.
- On average, it is better to use MEAN or BOTTOU than MEDIAN for “Update representatives” in partitioning clustering.
- Using DIANA improved the probability of a partitioning algorithm being the “best”, whereas using RANDOM improved the probability of an algorithm being the “rest”.
- Algorithms using BIC and global refinement strategy for hierarchical division tend to better detect the true number of clusters.

This showed that using RC based algorithms can allow data mining practitioners to achieve better adaptation for data at hand. Still, manual search or automatic brute search for the best algorithm in such a large space is very hard. This implies an important research task: definition of intelligent strategy for automatic algorithm selection in the space of RC based clustering algorithms. This problem is

Gene (sub-problem)	Initialize representatives	Measure distance	Update representatives	Evaluate clusters	Stop criterion
Alleles	DIANA	COSINE	BOTTOU	SILHOU	MEMBERS(5)

Fig. 9. Genotype of an RC-based algorithm.

addressed in [98] where evolutionary algorithm is used for automatic search through the space of RC based decision trees and showed cutting edge performance.

A further approach which could help for intelligent algorithm selection is meta-learning. It relates performance of algorithms to data characteristics and enables automatic selection of algorithms for a given problem. Although this is a common approach for selection/ranking of supervised learning algorithms [99], in the area of clustering this is a relatively new and unexplored topic [100]. Still there are initial researches in the area of microarray data clustering [100,101] which gave very promising results.

After development of intelligent strategy for automatic algorithm selection in the space of RC based clustering algorithms, we plan to apply this approach in different application areas like: analysis of web usage data [102,103], management models [104,105], document data [106–108] etc.

Appendix. RC characterization

The following tables propose some characteristics of RCs on the level of sub-problems that could be used for meta-learning. The characteristics are given for "Initialize representatives", "Measure distance", and "Update representatives" in Tables 1 through 3 respectively.

The RCs for "Update representatives" are described with the following characteristics:

- Binary division: defines whether the RC uses hierarchical binary division for initializing K representatives.
- Stochastic: defines whether the RC includes stochastic, i.e. whether the RC can propose different representatives when ran several times.
- Order dependent: defines whether the order of objects in a dataset has influence on initial representatives.
- PCA-based: defines whether principal component analysis is used for representatives' calculation.
- Discrete: defines whether the proposed representatives are from the discrete object space, or not.

Table 1
Characterization of initialize representatives RCs.

Name	Binary divisions	Stochastic	Order dependent	PCA-based	Discrete
DIANA	X				
SPSS			X		X
PCA	X			X	
GMEANS	X			X	
KMEANS++		X			X
XMEANS	X	X			
RANDOM		X			X

One attribute is given for "Measure distance" in Table 1. It defines whether the RC is a distance measure, or a similarity measure.

Table 2
Characterization of measure distance RCs.

Name	Distance
EUCLID	X
COSINE	
CORREL	
CHEBY	X
CITY	X

In Table 3, RCs for updating representatives are shown. Their proposed characteristics are:

- Online: defines whether the component works online or not (batch), and
- Discrete: defines whether the component works with discrete objects or in the continuous space.

Table 3
Characterization of update representatives RCs.

Name	Online	Discrete
MEAN		
BOTTOU	X	
MEDIAN		X

References

[1] P. Berkhin, Grouping Multidimensional Data — Recent Advances in Clustering, Springer-Verlag, 2005, pp. 127–160.
[2] R. Xu, D. Wunsch, Survey of clustering algorithms, IEEE Transactions on Neural Networks 16 (3) (2005) 654–678.
[3] A. Jain, M. Murty, P. Flynn, Data clustering: a review, ACM Computing Surveys 31 (3) (1999) 264–323.
[4] J. Shao, C. Plant, Q. Yang, C. Böhm, Detection of arbitrarily oriented synchronized clusters in high-dimensional data, Proc. of 11th IEEE International Conference on Data Mining (ICDM), 2011, pp. 607–616, doi:10.1109/ICDM.2011.50.
[5] E. Shaham, D. Sarne, B. Ben-Moshe, Sleeved co-clustering of lagged data, Knowledge and Information Systems (2011), doi:10.1007/s10115-011-0420-6.
[6] H.-X. Dang, J. Bailey, A hierarchical information theoretic technique for the discovery of non linear alternative clusterings, Proc. of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2010, pp. 573–582.
[7] T. De Bie, An information theoretic framework for data mining, Proc. of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2011, pp. 564–572.
[8] A. Da Silva, R. Chiky, G. Hébrail, A clustering approach for sampling data streams in sensor networks, Knowledge and Information Systems (2011), doi:10.1007/s10115-011-0448-7.
[9] M. Steinbach, G. Karypis, V. Kumar, A comparison of document clustering techniques, Proc. Workshop on Text Mining, 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, 2000.
[10] M. Mahdavi, H. Abolhassani, Harmony K-means algorithm for document clustering, Data Mining and Knowledge Discovery 18 (2009) 370–391.
[11] B. Bhen, J. He, S. Pellicer, Y. Pan, Using Hybrid Hierarchical K-means (HHK) clustering algorithm for protein sequence motif Super-Rule-Tree (SRT) structure construction, International Journal of Data Mining and Bioinformatics 4 (3) (2010) 316–330.
[12] M. Vukicevic, B. Delibasic, M. Jovanovic, M. Suknovic, Z. Obradovic, Internal evaluation measures as proxies for external indices in clustering gene expression data, Proc. 2011 IEEE International Conference on Bioinformatics and Biomedicine, Atlanta, GA, Nov. 2011, doi:10.1109/BIBM.2011.97.
[13] M. Vukicevic, B. Delibasic, Z. Obradovic, M. Jovanovic, M. Suknovic, A method for design of data-tailored partitioning algorithms for optimizing the number of clusters in microarray analysis, Proc. 2012 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology, San Diego, CA, May 2012.
[14] L. Nanetti, L. Cerliani, V. Gazzola, R. Renken, C. Keysers, Group analyses of connectivity-based cortical parcellation using repeated K-means clustering, NeuroImage 47 (4) (2009) 1666–1677.
[15] C. Bouras, V. Tsogkas, Advanced knowledge-based systems, Invited Session of the 14th International Conference on Knowledge-based and Intelligent Information & Engineering Systems, Cardiff Wales, UK, 2010, pp. 379–388.
[16] C.-L. Chen, F. Tseng, T. Liang, An integration of WordNet and fuzzy association rule mining for multi-label document clustering, Data & Knowledge Engineering 69 (11) (2010) 1208–1226.
[17] J. Liu, Comparative analysis fork-means algorithms in network community detection, advances in computation and intelligence, Lecture Notes in Computer Science 63 (82) (2010) 158–169.
[18] V. Murthy, E. Vamsidhar, J. Kumar, Content based image retrieval using Hierarchical and K-means clustering techniques, International Journal of Engineering, Science and Technology 2 (3) (2010) 209–212.
[19] A. Jain, Data clustering: 50 years beyond K-means, Pattern Recognition Letters 31 (8) (2010) 651–666.
[20] H.H. Bock, Clustering methods: a history of the K-means algorithm, in: P. Brito, et al., (Eds.), Selected Contributions in Data Analysis and Classification, Springer Verlag, Heidelberg, 2007, pp. 161–172.
[21] X. Wu, V. Kumar, J.R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G.J. McLachlan, A.F.M. Ng, B. Liu, P.S. Yu, Z.-H. Zhou, M. Steinbach, D.J. Hand, D. Steinberg, Top 10 algorithms in data mining, Knowledge and Information Systems 14 (1) (2007) 1–37, doi:10.1007/s10115-007-0114-2.
[22] S. Sonnenburg, M. Braun, C.S. Ong, S. Bengio, L. Bottou, G. Holmes, Y. LeCun, K.R. Müller, F. Pereira, C.E. Rasmussen, G. Rätsch, B. Schölkopf, A. Smola, P. Vincent, J. Weston, R. Williamson, The need for open source software in machine learning, Journal of Machine Learning Research 8 (2007) 2443–2466.
[23] B. Delibašić, M. Jovanović, M. Vukićević, M. Suknović, Z. Obradović, Component-based decision trees for classification, Intelligent Data Analysis 15 (5) (2011) 671–693, doi:10.3233/IDA-2011-0489.
[24] D.H. Wolpert, The lack of a priori distinctions between learning algorithms, Neural Computation 8 (7) (1996) 1341–1390.
[25] E. Baralis, G. Bruno, A. Flori, Measuring gene similarity by means of the classification distance, Knowledge and Information Systems 29 (2011) 81–101, doi:10.1007/s10115-010-0374-0.
[26] W. Tracz, Where does reuse start? ACM SIGSOFT Software Engineering Notes 15 (2) (1990) 42–46.
[27] A. Coronato, G. De Pietro, Formal design and implementation of constraints in software components, Advances in Engineering Software 41 (2010) 737–747.
[28] B. Delibasic, K. Kirchner, J. Ruhland, M. Jovanovic, M. Vukicevic, Reusable components for partitioning clustering algorithms, Artificial Intelligence Review 32 (2009) 59–75, doi:10.1007/s10462-009-9133-6.
[29] J. Lai, Fast global K-means clustering using cluster membership and inequality, Pattern Recognition 43 (5) (2010) 1954–1963.
[30] A. Bagirov, Modified global kK-means algorithm for minimum sum-of-squares clustering problems, Pattern Recognition 41 (10) (2008) 3192–3199.
[31] P. Hansen, E. Ngai, B.K. Cheung, N. Mladenovic, Analysis of global K-means, an incremental heuristic for minimum sum-of-squares clustering, Journal of Classification 22 (2) (2005) 287–310.
[32] A. Likas, M. Vlassis, J. Verbeek, The global K-means clustering algorithm, Pattern Recognition 36 (2) (2003) 451–461.
[33] C. Tang, S. Wang, W. Xu, New fuzzy c-means clustering model based on the data weighted approach, Data & Knowledge Engineering 69 (9) (2010) 881–998.
[34] K. Mumtaz, K. Duraiswamy, A novel density based improved K-means clustering algorithm–Dbkmeans, International Journal on Computer Science and Engineering 2 (2) (2010) 213–218.
[35] K. Žalik, An efficient k′-means clustering algorithm, Pattern Recognition Letters 29 (9) (2008) 1385–1391.
[36] N. Tajunisha, V. Saravanan, An Increased performance of clustering high dimensional data using principal component analysis, First International Conference on Integrated Intelligent Computing, 2010, pp. 17–21.
[37] A. Fahim, A. Salem, F. Torkey, G. Saake, M. Ramadan, An efficient K-means with good initial starting points, Computer Science and Telecommunications 2 (19) (2009) 47–57.
[38] M. Yedla, S.R. Pathakota, T.M. Srinivasa, Enhancing K-means clustering algorithm with improved initial centers, International Journal of Computer Science and Information Technologies 1 (2) (2010) 121–125.
[39] A. Ahmad, L. Dey, A k-mean clustering algorithm for mixed numeric and categorical data, Data and Knowledge Engineering 63 (2) (2007) 503–527, doi:10.1016/j.datak.2007.03.016.
[40] Sebastian Lühr, Mihai Lazarescu, Incremental clustering of dynamic data streams using connectivity based representative points, Data and Knowledge Engineering 68 (1) (2009) 1–27, doi:10.1016/j.datak.2008.08.006.
[41] Z. Jiang, Z. Lin, L.S. Davis, Class consistent k-means: application to face and action recognition, Computer Vision and Image Understanding 116 (6) (2012) 730–741, doi:10.1016/j.cviu.2012.02.004.
[42] Z.G. Yang, Y. Li, G.P. Yang, An improved K-means based method for fingerprint segmentation with sensor interoperability, Applied Mechanics and Materials 135–136 (2011) 237–243, doi:10.4028/www.scientific.net/AMM.135-136.237.
[43] A. Ahmad, L. Dey, A k-means type clustering algorithm for subspace clustering of mixed numeric and categorical datasets, Pattern Recognition Letters 32 (7) (2011) 1062–1069, doi:10.1016/j.patrec.2011.02.017.
[44] G.W. Milligan, M.C. Cooper, Methodology review: clustering methods, Applied Psychological Measurement 11 (4) (1987) 329–354.

[45] M.M. Astrahan, Speech Analysis by Clustering, or the Hyperphome Method, Stanford Artificial Intelligence Project Memorandum AIM-124, Stanford University, Stanford, CA, 1970.
[46] P.S. Bradley, U.M. Fayyad, Refining initial points for K-means clustering, Proc. 15th International Conf. on Machine Learning, Morgan Kaufmann, San Francisco, CA, 1998, pp. 91–99.
[47] V. Faber, Clustering and the continuous K-means algorithm, Los Alamos Science 22 (1994) 138–144.
[48] D. Steinley, Local optima in K-means clustering: what you don't know may hurt you, Psychological Methods 8 (3) (2003) 294–304.
[49] B. Mirkin, Clustering for Data Mining: A Data Recovery Approach, Chapman and Hall, London, 2005.
[50] M. Belal, A. Daoud, A new algorithm for cluster initialization, Proc. World Academy of Science, Engineering and Technology, 2005, pp. 74–76.
[51] R. Maitra, Initializing partition–optimization algorithms, IEEE/ACM Transactions on Computational Biology and Bioinformatics 6 (1) (2009) 144–157.
[52] P. Freeman, Reusable software engineering: concepts and research directions, in: P. Freeman (Ed.), Tutorial: Software Reusability, IEEE Computer Society Press, 1987, pp. 10–23.
[53] W. Frakes, K. Kang, Software reuse research: status and future, IEEE Transactions on Software Engineering 31 (7) (2005) 529–536.
[54] B. Delibašić, K. Kirchner, J. Ruhland, A pattern based data mining approach, in: Burkhardt Preisach, Schmidt-Thieme Decker (Eds.), Data Analysis, Machine Learning and Applications, Springer Verlag, 2008, pp. 327–334, doi:10.1007/978-3-540-78246-9_39.
[55] S. Lühr, M. Lazarescu, Incremental clustering of dynamic data streams using connectivity based representative points, Data & Knowledge Engineering 68 (2009) 1–27.
[56] I. Mierswa, K. Morik, Automatic feature extraction for classifying audio data, Machine Learning 58 (2–3) (2005) 127–149.
[57] A.D. Peterson, A.P. Ghosh, R. Maitra. A systematic evaluation of different methods for initializing the k-means clustering algorithm. Technical Report 07, Iowa State University, Department of Statistics, Ames, IA, 2010. 2010.
[58] H. Xiong, J. Wu, J. Chen, K-means clustering versus validation measures: a data-distribution perspective, IEEE Transactions on Systems, Man, and Cybernetics. Part B, Cybernetics: A Publication of the IEEE Systems, Man, and Cybernetics Society 39 (2) (2009) 318–331.
[59] O. Altun, N. Dursunoglu, M. Amasyali, Clustering application benchmark, IEEE International Symposium on Workload Characterization, 2006, pp. 178–181.
[60] M. Walesiak, A. Dudek, Symulacyjnaoptymalizacjawyboruproceduryklasyfikacyjnejdladanegotypudanych - charakterystykaproblemu, ZeszytyNaukoweUniwersytetuSzczecinskiego, 450, 2007, pp. 635–646.
[61] E. Achtert, H. Kriegel, A. Zimek, ELKI: a software system for evaluation of subspace clustering algorithms, Proc. of 20th International Conference on Scientific and Statistical Database Management, 2008, pp. 580–585.
[62] S.P. Lloyd, Least squares quantization in PCM [Pulse-Code Modulation.], IEEE Transactions on Information Theory 28 (1982) 129–137, doi: 10.1109/TIT.1982.1056489.
[63] J.A. Hartigan, Clustering Algorithms (Probability & Mathematical Statistics), John Wiley & Sons Inc, 1975.
[64] L. Kaufman, P.J. Rousseeuw, Finding Groups in Data: An Introduction to Cluster Analysis, John Wiley & Sons, 1990.
[65] D. Pelleg, A.W. Moore, X-means: extending K-means with efficient estimation of the number of clusters, Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Morgan Kaufmann, 2000, pp. 727–734.
[66] G. Hammerly, C. Elkan, Learning the k in k-means, Proceedings of the Seventeenth Annual Conference on Neural Information Processing Systems, 2003, pp. 281–288.
[67] O. Younis, S. Fahmy, FlowMate: scalable on-line flow clustering, IEEE/ACM Transactions on Network 13 (2) (2005) 288–301, doi: 10.1109/TNET.2005.845532.
[68] Amit Kumar, Yogish Sabharwal, Sandeep Sen, Linear-time approximation schemes for clustering problems in any dimensions, Journal of the ACM 57 (2) (2010), doi:10.1145/1667053.1667054.
[69] A. Ene, S. Im, B. Moseley, Fast clustering using MapReduce, Proc. of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2011, pp. 681–689.
[70] C. Ding, X. He, K-means clustering via principal component analysis, Proceedings of the Twenty-first International Conference on Machine Learning, ACM, New York, NY, 2009, p. 29.
[71] D. Arthur, S. Vassilvitskii, K-Means++: the advantages of careful seeding, Proc. Eighteenth Annual ACMSIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics, New Orleans, Louisiana, 2007, pp. 1027–1035.
[72] J. Hartigan, M. Wong, A k-means clustering algorithm, Applied Statistics 28 (1979) 100–108.
[73] T. Kohonen, Self-Organizing Maps, third ed. Springer Verlag, Berlin, 2001.
[74] L. Bottou, Convergence properties of the K-means algorithms, Advances in Neural Information Processing Systems, 7, MIT Press, 1995, pp. 585–592.
[75] Y. Cheung, k*-Means: a new generalized k-means clustering algorithm, Pattern Recognition Letters 24 (2003) 2883–2893.
[76] J. Rousseeuw, Silhouettes: a graphical aid to the interpretation and validation of cluster analysis, Journal of Computational and Applied Mathematics 20 (1987) 53–65.
[77] X.L. Xie, G. Beni, A validity measure for fuzzy clustering IEEE Trans, Pattern Analysis and Machine Intelligence 13 (8) (1991) 841–847.
[78] M. Ester, H. Kriegel, J. Sander, X. Xiaowei, Proc. of 2nd International Conference on Knowledge Discovery and Data Mining, 1996, pp. 226–231.
[79] C. Ding, X. He, H. Zha, M. Gu, H. Simon, A min–max cut algorithm for graph partitioning and data clustering, Proc. IEEE Int'l Conf. Data Mining, 2001.
[80] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, T. Euler, YALE: rapid prototyping for complex data mining tasks, Proc. 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-06), 2006.
[81] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. Witten, The WEKA data mining software: an update, ACM SIGKDD Explorations Newsletter 11 (1) (2009) 10–18, doi:10.1145/1656274.1656278.
[82] R. Rakotomalala, TANAGRA: un logicielgratuit pour l'enseignement et la recherche, Actes de EGC'2005, RNTI-E-3, vol. 2, 2005, pp. 697–702.
[83] J. Demšar, B. Zupan, G. Leban, T. Curk, Orange: from experimental machine learning to interactive data mining, Knowledge Discovery in Databases: PKDD, 2004, pp. 537–539.
[84] R Development Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria3-900051-07-0, 2008 URL http://www.R-project.org.
[85] M. Berthold, N. Cebron, F. Dill, G. Di Fatta, T. Gabriel, F. Georg, T. Meinl, P. Ohl, C. Sieb, B. Wiswedel, KNIME: The Konstanz Information Miner, Proc. of the Workshop on Multi-Agent Systems and Simulation MAS&S, 4th Annual Industrial Simulation Conference (ISC), Palermo, Italy, Jun. 2006, pp. 58–61.
[86] T. Abeel, Y.V. de Peer, Y. Saeys, Java-ML: a machine learning library, Journal of Machine Learning Research 10 (2009) 931–934.
[87] G. Karypis, CLUTO a Clustering Toolkit, Technical Report 02-017, Dept. of Computer Science, University of Minnesota, 2002.
[88] M. Vukićević, M. Jovanović, B. Delibašić, M. Suknović, WhiBo — RapidMiner plug-in for component based data mining algorithm design, Proc. of the 1st RapidMiner Community Meeting and Conference, Dortmund, Germany, 2010, pp. 33–38.
[89] G. Milligan, An examination of the effect of six types of error perturbation on fifteen clustering algorithms, Psychometrika 45 (1980) 325–342.
[90] J.H. Ward, Hierarchical grouping to optimize an objective function, Journal of the American Statistical Association 58 (301) (1963) 236–244.
[91] B. Dom, An Information Theoretic External Cluster Validity Measure, IBM Research Report RJ 10219, IBM's Almaden Research Center, San Jose, CA, 2001.
[92] A. Asuncion, D.J. Newman, UCI Machine Learning Repository, University of California, School of Information and Computer Science, 2007 [www.ics.uci.edu/~mlearn/MLRepository.html].
[93] A. Ahmad, L. Dey, A k-mean clustering algorithm for mixed numeric and categorical data, Data & Knowledge Engineering 63 (2) (2007) 503–527.
[94] G. Forestier, P. Gançarski, C. Wemmert, Collaborative clustering with background knowledge, Data & Knowledge Engineering 69 (2) (2010) 211–228.
[95] V. Estivill-Castro, Why so many clustering algorithms, SIGKDD Explorations 4 (1) (2002) 65–75.
[96] S. Zhu, J. Wu, H. Xiong, G. Xia, Scaling up top-*K* cosine similarity search, Data & Knowledge Engineering 70 (1) (2011) 60–83.
[97] D. Wijaya, A. Kalousis, M. Hilario, Predicting classifier performance using data set descriptors and data mining ontology, Proc. of the 3rd Planning to Learn Workshop, ECAI, 2010.

[98] M. Jovanović, B. Delibašić, M. Vukićević, M. Suknović, Optimizing performance of decision tree component-based algorithms using evolutionary algorithms in Rapid Miner, Proc. of the 2ndRapidMiner Community Meeting and Conference, Dublin, Ireland, 2011.
[99] K. Smith-Miles, Towards insightful algorithm selection for optimisation using meta-learning concepts, Proceedings of the IEEE International Joint Conference on Neural Networks, 2008, pp. 4118–4124.
[100] M. de Souto, R. Prudencio, R. Soares, D. Araujo, I. Costa, T. Ludermir, A. Schliep, Ranking and selecting clustering algorithms using a meta-learning approach, Proc. of the IEEE International Joint Conference on Neural Networks, 2008, pp. 3729–3735.
[101] A. Nascimento, R. Prudencio, M. de Souto, I. Costa, Mining rules for the automatic selection process of clustering methods applied to cancer gene expression data, Proceedings of the 19th International Conference on Artificial Neural Networks: Part II, Springer-Verlag Berlin, Heidelberg, 2009.
[102] M. Wan, A. Jönsson, C. Wang, L. Li, Y. Yang, Web user clustering and Web prefetching using Random Indexing with weight functions, Knowledge and Information Systems (2011), doi:10.1007/s10115-011-0453-x.
[103] M. Milovanović, M. Minović, V. Štavljanin, M. Savković, D. Starčević, Wiki as a corporate learning tool: case study for software development company, Behaviour & Information Technology (2012), doi:10.1080/0144929X.2011.642894.
[104] G. Savoiu, O. Jaško, M. Čudanov, Diversity of specific quantitative, statistical and social methods, techniques and management models in management system, Management 14 (52) (2010) 5–13.
[105] O. Sedlak, V. Kocic-Vugdelija, M. Kudumovic, C. Besic, D. Djordjevic, Management of family farms — implementation of fuzzy method in short-term planning, Technics Technologies Education Management — TTEM 5 (4) (2010) 710–718.
[106] V. Balachandran, D. Khemani, Interpretable and reconfigurable clustering of document datasets by deriving word-based rules, Knowledge and Information Systems (2011), doi:10.1007/s10115-011-0446-9.
[107] A. Kalogeratos, A. Likas, Document clustering using synthetic cluster prototypes, Data & Knowledge Engineering 70 (3) (2011) 284–306 doi: j.datak.2010.12.002.
[108] C.-L. Chen, F.S.C. Tseng, An integration of WordNet and fuzzy association rule mining for multi-label document clustering, Data & Knowledge Engineering 69 (11) (2010) doi:j.datak.2010.08.003.

Boris Delibašić is assistant professor at the University of Belgrade, Serbia, at the Faculty of Organizational Sciences, at the Department for Business Decision Making since 2007. He holds a PhD for "Formalization of the Business Decision Making identification of reusable components to support the decision-making process.

Milan Vukićević is teaching and research assistant at the Faculty of Organizational Sciences, University of Belgrade, within the Center for Business Decision Making. He is currently a PhD student. His main research field interests are clustering and classification algorithm design, data mining, decision support, and meta learning.

Miloš Jovanović is teaching and research assistant at the Faculty of Organizational Sciences, University of Belgrade, within the Center for Business Decision Making. He is currently PhD student. His main research field interests are data mining, decision support systems, data warehousing, optimization and artificial intelligence.

Kathrin Kirchner is a postdoctoral researcher and lecturer at the department of Business Information Systems, Friedrich Schiller University of Jena, Germany. For her PhD she developed a spatial decision support system for the rehabilitation of gas pipeline networks. Her research interests include data mining, decision support and knowledge management.

Johannes Ruhland has been a full professor at Friedrich Schiller University of Jena since 1994. Before that he was a full professor at Universities of Ulm and Munich. Professor Ruhland's research addresses data mining algorithms and data mining applications in marketing and analytical CRM as well as business process management and economical and managerial aspects of energy economy.

Milija Suknović is professor at the University of Belgrade, Serbia, at the Faculty of Organizational Sciences, at the Department for Business Decision Making since 1991. He holds a PhD for “Development of Support Methodology for Group Decision Making”. His research interests are focused on data mining, decision support systems, group decision making and data warehousing.