Combinatorial Optimization Approaches for Data Clustering

Paola Festa

Abstract Target of cluster analysis is to group data represented as a vector of measurements or a point in a multidimensional space such that the most similar objects belong to the same group or cluster. The greater the similarity within a cluster and the greater the dissimilarity between clusters, the better the clustering task has been performed. Starting from the 1990s, cluster analysis has emerged as an important interdisciplinary field, applied to several heterogeneous domains with numerous applications, including among many others social sciences, information retrieval, natural language processing, galaxy formation, image segmentation, and biological data.

Scope of this paper is to provide an overview of the main types of criteria adopted to classify and partition the data and to discuss properties and state-of-the-art solution approaches, with special emphasis to the combinatorial optimization and operational research perspective.

1 Introduction

Given a finite set of *objects*, cluster analysis aims to group them such that the most similar ones belong to the same group or *cluster*, and dissimilar objects are assigned to different clusters. In the scientific literature, the objects are also called *entities* or *patterns* and are usually represented as a vector of measurements or a point in a multidimensional space. Clearly, it can be easily guessed that the greater the similarity within a cluster and the greater the dissimilarity between clusters, the better the clustering task has been performed.

Starting from the 1990s, cluster analysis has emerged as an important interdisciplinary field, involving different scientific research communities, including mathematics, theoretical and applied statistics, genetics, biology, biochemistry, computer science, and engineering. It has been applied to several domains with numerous applications, ranging from social sciences to biology. Starting from one

P. Festa (✉)
Department of Mathematics and Applications "R. Caccioppoli",
University of Napoli FEDERICO II, Compl. MSA, Via Cintia, 80126 Napoli, Italy
e-mail: paola.festa@unina.it

M.E. Celebi, K. Aydin (eds.), *Unsupervised Learning Algorithms*,
DOI 10.1007/978-3-319-24211-8_5

of the pioneering paper of Rao, which appeared in 1971 [60], more recent surveys on clustering algorithms and their applications can be found in [8–11, 21, 22, 40, 41, 49, 53, 53, 77], and very recently in [29, 45, 71]. Nice books and edited books are [12, 56, 57, 76].

In cluster analysis, the criterion for evaluating the quality of a clustering strongly depends upon the specific application in which it is to be used. The cluster task can be mathematically formulated as a constrained fractional nonlinear 0–1 programming problem, and there are no computationally efficient procedures for solving such a problem, which becomes computationally tractable only under restrictive hypotheses.

This paper surveys the main types of clustering and criteria for homogeneity or separation, with special emphasis to the optimization and operational research perspective. In fact, first a few mathematical models of the problem are reported under several different criterion adopted to classify the data and some classical state-of-the-art exact methods are described that use the mathematical model of the problem. Then, the most famous and applied state-of-the-art clustering algorithms are reviewed, underlying and comparing their main ingredients.

The remainder of this paper is organized as follows. The next section lists some among the most useful applications of the problem. In Sect. 3, the cluster analysis task is formally stated and the most used distance measures between the various entities are described. State-of-the-art mathematical formulations of the problem along with some classical state-of-the-art exact methods are described in Sect. 4. In Sect. 5, properties and state-of-the-art solution approaches are discussed. Concluding remarks are given in the last section.

2 Applications

Cluster analysis applies in several heterogeneous domains with numerous applications, whose number grows increasingly in recent years. As the increasingly sophisticated technology allows the storage of increasingly large amounts of data, the availability of efficient techniques for generating new information by examining the resulting large databases becomes ever more urgent.

Some of cluster analysis applications are listed in the following.

- **Social sciences**: in this context, clustering helps in understanding how people analyze and catalogue life experiences.
- **Information retrieval**: in this context, clustering is used to create groups of documents with the goal of improving the efficiency and effectiveness of the retrieval, such as in the case of thousands of Web pages retrieved as a result of a query to a search engine (see, e.g., [16–18, 74]).
- **Natural language processing**: typically, large vocabularies of words of a given natural language must be clustered w.r.t. corpora of very high size [70].

- **Business**: in this application, one can be interested in analyzing information about potential customers, in order to cluster them for some sort of marketing activities [58].
- **Galaxy formation**: in this context, a study has been conducted on the formation of galaxies by gas condensation with massive dark halos [75].
- **Image segmentation**: in this case, the segmentation is achieved by searching for closed contours of the elements in the image [73].
- **Biological data**: one among the most prominent interests of the biologists is the analysis of huge data containing genetic information, such as to find groups of genes having similar functions (see among others [3, 20, 28, 30, 50, 54, 55]).

3 Problem Definition and Distance Measures Definition

In cluster analysis, we are given

◇ a set of N objects (entities, patterns) $O = \{o_1, \ldots, o_N\}$;
◇ a set of M of pre-assigned clusters $S = \{S_1, \ldots, S_M\}$;
◇ a function $d : O \times O \mapsto R$ that assigns to each pair $o_i, o_j \in O$ a “metric distance” or “similarity” $d_{ij} \in R$ (usually, $d_{ij} \geq 0$, $d_{ii} = 0$, $d_{ij} = d_{ji}$, for $i, j = 1, \ldots, N$)

and the objective is to assigning the objects in O to some cluster in S while optimizing some distance criteria in such a way that the greater the similarity (or proximity, homogeneity) within a cluster and the greater the difference between clusters, the better or more distinct the clustering [38, 40].

According to [38] and [40], the problem involves the following five steps:

1. pattern representation (optionally including feature extraction and/or selection);
2. definition of a pattern proximity (or similarity) measure appropriate to the data domain;
3. clustering or grouping;
4. data abstraction (if needed), and
5. assessment of output (if needed).

In more detail, the first step, usually referred to as pattern representation, refers to the number of classes, the number of available patterns, and the number, type, and scale of the features available to the clustering algorithm. Typical of this first step are the process of *feature selection*, i.e., the identification of the most effective subset of the original features to use in clustering, and the process of *feature extraction*, i.e., the transformations of the input features to produce new salient features.

Pattern proximity (similarity) is usually measured by a distance function defined on pairs of patterns. In the scientific literature, the patterns or data objects are usually represented as a vector of measurements or a point in a multidimensional space.

Formally, a data object o_i, $i = 1, \ldots, N$ can be represented as the following numerical vector

$$\overrightarrow{A}_i = \{a_{ij} \mid 1 \le j \le L\},$$

where

- a_{ij} is the value of the jth feature for the ith data object and
- L is the number of features.

Then, the proximity d_{ij} between two objects o_i and o_j is measured by a proximity function d of corresponding vectors $\overrightarrow{A}_i$ and $\overrightarrow{A}_j$.

Several different scientific communities have used and discussed a variety of distance measures (see, for example, [4, 37, 38, 41]). Some of them are listed in the following.

3.1 Euclidean Distance

Given two objects o_i and $o_j \in O$, their Euclidean distance in L-dimensional space is defined as follows:

$$d_{ij} = \sqrt{\sum_{k=1}^{L} (a_{ik} - a_{jk})^2} = \|\mathbf{a}_i - \mathbf{a}_j\|_2. \tag{1}$$

Euclidean distance has an intuitive meaning and it is usually used to evaluate the proximity of objects in two- or three-dimensional space. In general, it works well when the data set has "compact" or "isolated" clusters [51].

3.2 Pearson's Correlation Coefficient

The Pearson's correlation coefficient measures the similarity between the *shapes of two patterns*, also known as *profiles*.

Given two objects o_i and $o_j \in O$, their Pearson's correlation coefficient is defined as follows:

$$d_{ij} = \frac{\sum_{k=1}^{L} [(a_{ik} - \mu_{o_i}) \cdot ((a_{jk} - \mu_{o_j}))]}{\sqrt{\sum_{k=1}^{L} a_{ik} - \mu_{o_i})^2} \cdot \sqrt{\sum_{k=1}^{L} a_{jk} - \mu_{o_j})^2}}, \tag{2}$$

where μ_{o_i} and μ_{o_j} are the mean value for $\overrightarrow{A}_i$ and $\overrightarrow{A}_j$, respectively.

This correlation coefficient views each object as a random variable with L observations and measures the similarity between two objects by calculating the linear relationship between the distributions of the two corresponding random variables. One drawback of the Pearson's correlation coefficient is that it assumes an approximate Gaussian distribution of the patterns and may not be robust for non-Gaussian distributions, as experimentally shown by Bickel in 2001 [7].

3.3 City-Block or Manhattan

City-block or Manhattan distance simulates the distance between points in a city road grid. It measures the absolute differences between two object attributes.

Given two objects o_i and $o_j \in O$, their City-block or Manhattan distance is defined as follows:

$$d_{ij} = \sum_{k=1}^{L} |a_{ik} - a_{jk}|. \tag{3}$$

3.4 Cosine or Uncentered Correlation

Cosine or uncentered correlation is a geometric correlation defined by the angle between two objects.

Given two objects o_i and $o_j \in O$, their cosine or uncentered correlation is defined as follows:

$$D_{ij} = \frac{\sum_{k=1}^{L} a_{ik} \cdot a_{jk}}{\sum_{k=1}^{L} a_{ik}^2 \sum_{k=1}^{L} a_{jk}^2}. \tag{4}$$

Note that,

- the larger the value of D_{ij}, the lower the angle between the objects;
- $D_{ij} \in [-1, 1]$: $D_{ij} = -1$ implies that the angle between vectors representing o_i and o_j is a right angle; while $D_{ij} = 1$ implies that the angle between o_i and o_j is 0;
- $d_{ij} = 1 - |D_{ij}|$.

4 Mathematical Formulations of the Problem

In 1971, Rao [60] presented several mathematical formulations of the clustering problem, depending on several different criteria adopted to classify the data. Some of them are reported in the following.

4.1 Minimize (Maximize) the Within (Between)-Clusters Sum of Squares

Defining a set of $N \times M$ Boolean decision variables $x_{ik} \in \{0,1\}$, $i = 1,\ldots,N$, $k = 1,\ldots,M$, such that in a solution

$$x_{ik} = \begin{cases} 1, \text{ if } o_i \in O \text{ is in cluster } S_k; \\ 0, \text{ otherwise,} \end{cases}$$

the problem admits the following fractional nonlinear mathematical formulation:

$$\begin{array}{lll}
(\mathrm{DC}-1) & \min \sum_{k=1}^{M} \left\{ \dfrac{\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} d_{ij}^2 \, x_{ik} \, x_{jk}}{\sum_{i=1}^{N} x_{ik}} \right\} & \\
& \text{s.t.} & \\
(\mathrm{DC}-1.1) & \sum_{k=1}^{M} x_{ik} = 1, & i = 1,\ldots,N \\
(\mathrm{DC}-1.2) & x_{ik} \geq 0 \text{ and integer}, & i = 1,\ldots,N,\ k = 1,\ldots,M.
\end{array}$$

Constraints (DC-1.1) assure that each o_i, $i = 1,\ldots,N$, belongs to only one cluster.

Since (DC-1) is a fractional nonlinear 0–1 programming problem, it is difficult to solve. Exceptions are the two cases described in the following two paragraphs.

4.1.1 Cardinality of Each Cluster A Priori Known

A special case of problem (DC-1) occurs when the cardinality of each cluster is a priori known, i.e., when

$$|S_k| = n_k, \text{ s.t. } \sum_{k=1}^{M} n_k = N.$$

In this case, mathematical formulation (DC-1) can be slightly modified to take into account this further information as follows:

$$\begin{array}{lll}
(\mathrm{DC}-2) & \min \sum_{k=1}^{M} \dfrac{1}{n_k} \left\{ \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} d_{ij}^2 \, x_{ik} \, x_{jk} \right\} & \\
& \text{s.t.} & \\
(\mathrm{DC}-2.1) & \sum_{k=1}^{M} x_{ik} = 1, & i = 1,\ldots,N
\end{array}$$

$$(\mathrm{DC}-2.2)\quad \sum_{i=1}^{N} x_{ik} = n_k, \qquad k = 1, \ldots, M$$

$$(\mathrm{DC}-2.3)\quad x_{ik} \geq 0 \text{ and integer}, \qquad i = 1, \ldots, N,\ k = 1, \ldots, M.$$

In formulation (DC-2), the objective remains to minimize (maximize) the within (between)-clusters sum of squares. As in (DC-1), constraints (DC-2.1) guarantee that each object o_i, $i = 1, \ldots, N$, belongs to only one cluster. The additional set of constraints (DC-2.2) impose that for each cluster S_k, $k = 1, \ldots, M$, the number of objects in S_k is equal to n_k.

(DC-2) is still a nonlinear 0–1 formulation, but its objective function has lost the fractional characteristics, since here $\{n_k\}_{k=1}^{M}$ are known in advance. Among the first approaches to solve this problem are to be counted the Boolean methods proposed by Hammer and Rudeanu in [35] to be applied once the clustering problem is interpreted as a constrained nonlinear Boolean programming problem. Another pioneering approach has been described by Rao in [60]. It first linearizes the objective function by adding a further set of constraints. Then, it solves the resulting problem by applying any known method for linear integer problems. The resulting linearized 0–1 problem admits the following formulation:

$$(\mathrm{DC}'-2)\quad \min \sum_{k=1}^{M} \frac{1}{n_k} \left\{ \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} d_{ij}^2\, y_{ij}^k \right\}$$

s.t.

$$(\mathrm{DC}'-2.1)\quad x_{ik} + x_{jk} - y_{ij}^k \leq 1, \qquad i = 1, \ldots, N-1,\ j = i+1, \ldots, N,\ k = 1, \ldots, M$$

$$(\mathrm{DC}'-2.2)\quad \sum_{k=1}^{M} x_{ik} = 1, \qquad i = 1, \ldots, N$$

$$(\mathrm{DC}'-2.3)\quad \sum_{i=1}^{N} x_{ik} = n_k, \qquad k = 1, \ldots, M$$

$$(\mathrm{DC}'-2.4)\quad x_{ik},\ y_{ij}^k \geq 0 \text{ and integer}, \qquad i,j = 1, \ldots, N,\ k = 1, \ldots, M.$$

It must be underlined that, although easier to solve formulation (DC′-2) presents a number of constraints that grows rapidly with N and M and therefore can only be used for small instances of the problem.

4.1.2 Bipartition of the Patterns

Another special case occurs when $M = 2$, i.e., when there are only two pre-assigned clusters S_1 and S_2. In this particular scenario, introducing N Boolean decision variables $x_i \in \{0, 1\}$, $i = 1, \dots, N$, such that in a solution

$$x_i = \begin{cases} 1, \text{ if } o_i \in O \text{ is in cluster } S_1; \\ 0, \text{ if } o_i \in O \text{ is in cluster } S_2, \end{cases}$$

the clustering task can be modeled as the following fractional nonlinear 0–1 programming problem with no additional constraints:

$$\text{(DC}-3) \quad \min \left\{ \frac{\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} d_{ij}^2 \, x_i \, x_j}{\sum_{i=1}^{N} x_i} + \frac{\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} d_{ij}^2 \, (1 - x_i)\,(1 - x_j)}{N - \sum_{i=1}^{N} x_i} \right\}$$

$$\text{s.t.}$$

$$\text{(DC}-3.1) \qquad x_i \in \{0, 1\}, \; i = 1, \dots, N.$$

Based on the above mathematical formulation, Rao [60] designed an exact method that he presented as a Branch and Bound algorithm, but that is quite a method of exploring the feasible region making use of appropriate lower bounds obtained by means of any relaxation. The root of the decision tree corresponds to the empty solution in which any object has been assigned yet. An object o_i is selected and a branch is emanated: for convenience, the assignment of o_i to cluster S_1 corresponds to the right branch and that to cluster S_2 to the left branch. At a generic iteration, each current leaf of the tree corresponds to a partial solution characterized by some objects assigned to S_1, some assigned to S_2, while others are yet to be assigned.

Generally speaking, in order to design any Branch and Bound approach, it is necessary to specify the following key issues:

- the branching rule;
- the bounding strategy;
- the fathoming criterion.

About the branching rule, let us suppose that at a generic iteration t the algorithm is visiting node t of the tree that corresponds to a partial solution, where we have a set (possibly empty) of objects in S_1 and a set (possibly empty) of objects in S_2. Let $\overline{O} = O \setminus \{S_1 \cup S_2\}$ be the set of objects that are still to be assigned and let x_{best} be the incumbent solution, i.e., the best feasible solution found so far (at the beginning, x_{best} either is a known feasible solution or it is empty). If $\overline{O} = \emptyset$, then a complete solution has been found and x_{best} is eventually updated. Otherwise, a branching operation should be performed. Before performing a further branching, the so-called bounding criterion is tested, i.e., a lower bound for the objective function is computed (see hereafter how to compute a valid lower bound) and if the lower bound is greater than or equal to the objective function value corresponding to x_{best}, then no branching is performed since any feasible solution that can be generated from node t will be no better than the incumbent solution itself. Conversely, if the bounding criterion fails, a branching operation is performed by assigning to cluster S_1 an object $k \in \overline{O}$ such that

$$o_k = \arg\min_{j \in \overline{O}} \sum_{\hat{i} \in S_1} d_{\hat{i}j},$$

i.e., o_k is a not yet assigned object that corresponds to the minimum sum of the distances to all other objects already assigned to cluster S_1.

Generally speaking, the strategy for selecting the next sub-problem to be investigated determines how the Branch and Bound algorithm should proceed through the search tree and can have a significant effect on the behavior of the algorithm (fathoming rule). In [60], the proposed approach adopts a depth first search (DFS, for short) strategy, where the node with the largest level in the search tree is chosen for exploration. The algorithm continues branching until the end of the tree is reached or the bounding criterion indicates any branching is further needed. When this happens, the algorithm back-tracks along the tree until a node is reached with an unexplored branch to the left. A graphical representation of the branching rule is depicted in Fig. 1.

To obtain a lower bound, Rao [60] proposed several different strategies, based on the following further definitions and notation.

At a given node t of the branching tree, let x be a Boolean partial solution defined as in formulation (DC-3) and $\overline{O} = O \setminus \{S_1 \cup S_2\}$. Moreover, let

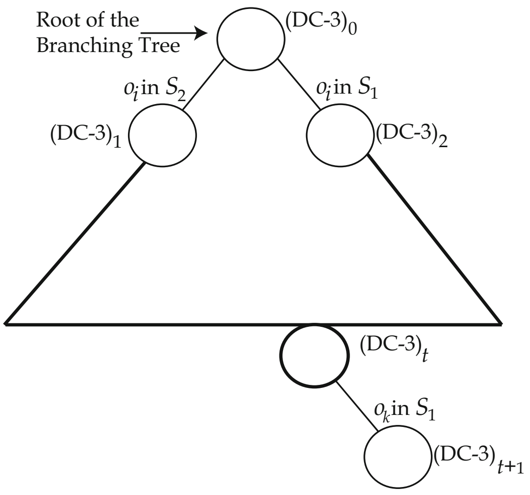

Fig. 1 Graphical representation of the branching rule

◇ $K_1 = \sum_{i,j \in S_1} d_{ij}$;
◇ $K_2 = \sum_{i,j \in S_2} d_{ij}$;
◇ D be the $n \times n$ symmetric distance matrix;
◇ F be a $|\overline{O}| \times |S_1|$ sub-matrix of D containing the distance between each currently not yet assigned object $k \in \overline{O}$ and an object in S_1;
◇ F_i, $i = 1, \ldots, |\overline{O}|$, be the sum of the distances in row i of matrix F;
◇ H be a $|S_2| \times |S_1|$ sub-matrix of D containing the distance between each object in S_2 and an object in S_1;
◇ H_i, $i = 1, \ldots, |S_1|$, be the sum of the distances in row i of matrix H;
◇ C be a $|\overline{O}| \times |\overline{O}|$ sub-matrix of D containing the distance between each pair of unassigned objects, whose diagonal elements are assigned a very high value $(+\infty)$.

Without loss of generality, matrices F and H are assumed arranged in such a way that

$$F_i < F_j, \text{ if } i < j;$$
$$H_i < H_j, \text{ if } i < j.$$

The objective function can be rewritten as follows:

$$Z = \min \left\{ \frac{K_1 + \sum_{i \in S_1, j \in \overline{O}} d_{ij}\, x_i\, x_j + \sum_{i,j \in \overline{O}} d_{ij}\, x_i\, x_j}{|S_1| + \sum_{i \in \overline{O}} x_i} \right.$$
$$\left. + \frac{K_2 + \sum_{i \in S_2, j \in \overline{O}} d_{ij}(1 - x_i)(1 - x_j) + \sum_{i,j \in \overline{O}} d_{ij}(1 - x_i)(1 - x_j)}{|S_2| + \left(|\overline{O}| - \sum_{i \in \overline{O}} x_i \right)} \right\}, \tag{5}$$

where $x_i \in \{0, 1\}$.

For any fixed value for $\overline{n} = \sum_{i \in \overline{O}} x_i$, by setting $n' = |\overline{O}| - \overline{n}$, $p = |S_1| + \overline{n}$, and $t = |S_2| + n'$, Z in (5) can be rewritten as follows:

$$Z' = \min t \cdot \left[\sum_{i \in S_1, j \in \overline{O}} d_{ij} x_i x_j + \sum_{i,j \in \overline{O}} d_{ij} x_i x_j \right]$$
$$+ p \cdot \left[\sum_{i \in S_2, j \in \overline{O}} d_{ij}(1 - x_i)(1 - x_j) + \sum_{i,j \in \overline{O}} d_{ij}(1 - x_i)(1 - x_j) \right], \tag{6}$$

given that the denominator (equal to pt), $K_1 t$, and $K_2 p$ are constant.

By varying $\overline{n}$ in its range from 0 to $|\overline{O}|$, $|\overline{O}|+1$ lower bounds for Z' (6) are computed and the minimum among them is kept as lower bound for Z (5). One way to obtain such a lower bound is the following:

$$Z' \geq u_1 + u_2 + u_3,$$

where

$$\begin{aligned}
u_1 &= t \cdot \min \sum_{i \in S_1, j \in \overline{O}} d_{ij} x_i x_j = t \cdot \sum_{i=1}^{\overline{n}} F_i;\\
u_2 &= p \cdot \min \sum_{i \in S_2, j \in \overline{O}} d_{ij}(1-x_i)(1-x_j) = p \cdot \sum_{i=1}^{n'} H_i;\\
u_3 &= \left[t \cdot \min \sum_{i,j \in \overline{O}} d_{ij} x_i x_j \right] + \left[p \cdot \min \sum_{i,j \in \overline{O}} d_{ij}(1-x_i)(1-x_j) \right].
\end{aligned}$$

Note that, the number of addenda in the summation to compute u_3 is $v = \frac{\overline{n}\cdot(\overline{n}-1)+n'(n'-1)}{2}$, lying either in the upper or in the lower triangular part of the symmetric matrix C. Consequently, a lower bound for u_3 can be obtained by multiplying $\min\{t, p\}$ by the sum of the v smallest elements of one of the triangle of matrix C. The reader interested in learning further techniques to compute a lower bound can refer to Rao's paper [60].

4.2 *Optimizing the Within Clusters Distance*

If one is interested in minimizing the total within clusters distance, the clustering task objective can be formulated as follows:

$$\begin{aligned}
&(\text{DC}-4) \quad \min \sum_{k=1}^{M} \left\{ \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} d_{ij}\, x_{ik}\, x_{jk} \right\}\\
&\qquad\qquad \text{s.t.}\\
&(\text{DC}-4.1) \quad \sum_{k=1}^{M} x_{ik} = 1, \qquad i = 1, \ldots, N\\
&(\text{DC}-4.2) \quad x_{ik} \geq 0 \text{ and integer}, \qquad i = 1, \ldots, N,\ k = 1, \ldots, M.
\end{aligned}$$

Note that, the objective function of (DC-4) is similar to the objective function of (DC-2), with the only differences that here the factors $\frac{1}{n_k}$, $k = 1, \ldots, M$, must not appear and coherently constraints (DC-2.2) do not require to be imposed.

A further useful possible target of a clustering task is to minimize the maximum within cluster distance. In this case, one is basically interested in minimizing the maximum distance within clusters and the problem can be modeled as a linear integer programming problem as follows:

$$
\begin{array}{lll}
(\mathrm{DC}-5) & \min Z & \\
& \text{s.t.} & \\
(\mathrm{DC}-5.1) & d_{ij}\, x_{ij} + d_{ij}\, x_{jk} - Z \le d_{ij}, & i = 1,\dots,N-1 \\
& & j = i+1,\dots,N \\
& & k = 1,\dots,M \\
(\mathrm{DC}-5.2) & \sum_{k=1}^{M} x_{ik} = 1, & i = 1,\dots,N,\ k = 1,\dots,M \\
(\mathrm{DC}-5.3) & x_{ik} \ge 0 \text{ and integer}, & i = 1,\dots,N,\ k = 1,\dots,M \\
(\mathrm{DC}-5.4) & Z \ge 0 \text{ and integer}. &
\end{array}
$$

In the case of minimizing the distance between the objects inside the same cluster, in 2010 Nascimento et al. [55] slightly modified Rao's model (DC-4) as follows:

$$
\begin{array}{lll}
(\mathrm{DC}-6) & \min \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} d_{ij} \sum_{k=1}^{M} x_{ik} \cdot x_{jk} & \\
& \text{s.t.} & \\
(\mathrm{DC}-6.1) & \sum_{k=1}^{M} x_{ik} = 1, & i = 1,\dots,N \\
(\mathrm{DC}-6.2) & \sum_{i=1}^{N} x_{ik} \ge 1, & k = 1,\dots,M \\
(\mathrm{DC}-6.3) & x_{ik} \in \{0,1\}, & i = 1,\dots,N,\ k = 1,\dots,M.
\end{array}
$$

Note that, a set of additional constraints (DC-6.2) are imposed to guarantee that each cluster S_k, $k = 1,\dots,M$, contains at least one object. In the attempt of remedying to the nonlinearity of the objective function, Nascimento et al. proposed a linearization of the model (DC-6). In more detail, by defining a further decision vector $y \in \mathscr{R}^{N\times N}$, the linearized version of formulation (DC-6) is the following:

$$\text{(LDC}-6) \quad \min \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} d_{ij} \cdot y_{ij}$$

s.t.

$$\text{(LDC}-6.1) \quad \sum_{k=1}^{M} x_{ik} = 1, \qquad i = 1, \ldots, N$$

$$\text{(LDC}-6.2) \quad \sum_{i=1}^{N} x_{ik} \geq 1, \qquad k = 1, \ldots, M$$

$$\text{(LDC}-6.3) \quad x_{ik} \in \{0, 1\}, \qquad i = 1, \ldots, N, \ k = 1, \ldots, M$$

$$\text{(LDC}-6.4) \quad y_{ij} \geq x_{ik} + x_{jk} - 1, \ i = 1, \ldots, N, j = i + 1, \ldots, N, \ k = 1, \ldots, M$$

$$\text{(LDC}-6.5) \quad y_{ij} \geq 0, \qquad i = 1, \ldots, N, j = i + 1, \ldots, N.$$

Constraints (LCD-6.4) and (LCD-6.5) guarantee that $y_{ij} = 1$ if $x_{ik} = x_{jk} = 1$, i.e., if objects $o_i, o_j \in O$ are in the same cluster. Therefore, it can be easily seen that the objective function of model (LCD-6) aims at minimizing the distance between objects in the same cluster. Note that, model (LCD-6) has $\frac{N \cdot (N-1) \cdot (M+1)}{2}$ more constraints than (DC-6) but it is linear and therefore "easier" to be solved.

Exact methods [35, 60] and the above-described mathematical formulations of the problem can be used only for small- sized instances, since the number of constraints characterizing the models increases very rapidly with both the number of objects N and the number of pre-assigned clusters M.

5 A Review of the Most Popular Clustering Techniques

According to Jain et al. [40] (see the taxonometric representation of clustering methods in Fig. 2), state-of-the-art clustering algorithms can be mainly divided into two families: *partitioning* and *hierarchical algorithms*.

A partitioning method partitions the set of data objects into non-overlapping clusters such that each data object belongs to exactly one cluster. Instead, in a hierarchical approach a cluster is permitted to have subclusters and the result of the clustering task is a set of nested clusters that can be organized in a tree. Each node of the tree corresponds to a cluster and it is the union of its children (subclusters). Clearly, the leaves have no subclusters and the root node represents the cluster containing all the objects.

Besides *exclusive/non-overlapping* versus *overlapping clustering*, in the literature several different further types of clusterings can be found, such as *fuzzy clustering* and *probabilistic clustering*.

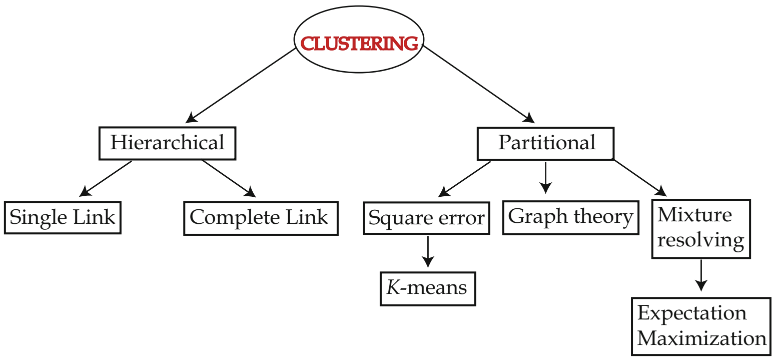

Fig. 2 A taxonomy of clustering methods

In a fuzzy clustering, clusters are viewed as fuzzy sets and a *membership weight function* $W : O \times S \mapsto [0, 1]$ is defined such that for each object $o_i \in O$, $i = 1, \ldots, N$, and for each cluster S_k, $k = 1, \ldots, M$, W_{ik} measures a "level" of membership of o_i to cluster S_k. Clearly, $W_{ik} = 0$ corresponds to "absolutely $o_i \notin S_k$, while $W_{ik} = 1$ corresponds to "absolutely $o_i \in S_k$.

Similarly to fuzzy clustering, a probabilistic clustering requires the computation of the probability p_{ik} with which each object $o_i \in O$, $i = 1, \ldots, N$, belongs to each cluster S_k, $k = 1, \ldots, M$. Clearly, it must be imposed that these probabilities must sum to 1. In general, probabilistic clustering are converted to an exclusive/non-overlapping clustering, where each object is assigned to the cluster in which its probability is highest.

5.1 Hierarchical Clustering Algorithms

The most popular hierarchical clustering algorithms are the single-link algorithm [65], complete-link [46], and minimum-variance algorithms [72]. The single-link and the complete-link approaches differ in how they define the similarity between a pair of clusters: in the single-link approach, this distance is the minimum of the distances between all pairs of patterns drawn from the two clusters (one pattern from the first cluster, the other from the second); in a complete-link algorithm, this distance is the maximum of all pairwise distances between patterns in the two clusters. In either case, two clusters are merged to form a larger cluster based on minimum distance criteria.

5.2 Partitioning Clustering Algorithms

The most popular partitioning clustering algorithms are the squared error algorithms (among them the most famous are the k-means method [52] and the k-medoid method [43, 44]), graph-theoretic algorithms [78], and mixture-resolving and mode-seeking algorithms [38].

5.2.1 Squared Error Algorithms and the k-Means/k-Medoid Algorithms

The squared error criterion is the most intuitive and used criterion among partitioning clustering techniques. As the Euclidean distance, it tends to work well with "isolated" and "compact" clusters.

Given a clustering S of a set of patterns O containing M clusters, the squared error for S, also known as *scatter*, is defined as

$$E^2(O,S) = \sum_{j=1}^{M} \sum_{i=1}^{n_j} \|\mathbf{a}_i^{(j)} - \mathbf{c}_j\|^2,$$

where

◇ n_j is the number of objects in cluster $j \in \{1,\ldots,M\}$;
◇ $\mathbf{a}_i^{(j)}$ the ith pattern belonging to cluster $j \in \{1,\ldots,M\}$;
◇ $\mathbf{c}_j$ is the centroid of cluster $j \in \{1,\ldots,M\}$.

The framework of a generic squared error algorithm is described in Fig. 3.

The k-means algorithm [52] and the k-medoid algorithm [43] are the simplest and maybe most famous approaches that use a squared error criterion. k-means relies on the definition of a centroid usually as the mean of a group of objects and it is typically applied to objects in a continuous n-dimensional space. Conversely, k-medoid algorithm relies on the definition of a *medoid* as the most representative object for a group of objects and in principle can be used in a wider range of contexts compared to k-means, since it only requires a suitable definition of a proximity

```
algorithm squared-error ($O = \{o_1,\ldots,o_N\}$, $M$)
1 Select an initial partition $S = \{S_1,\ldots,S_M\}$ of the patterns in $O$ with a fixed
  number $M$ of clusters and cluster centers.
2 Assign each pattern $o_i$, $i = 1,\ldots,N$, to its closest cluster center and compute
  the new cluster centers as the centroids of the clusters.
  Repeat this step until convergence is achieved, i.e., until the cluster
  membership is stable.
3 Merge and split clusters based on some heuristic information, optionally
  repeating step 2.
end squared-error
```

Fig. 3 Framework of a general squared error algorithm

```
algorithm k-means ($O = \{o_1,\dots,o_N\}$, $M$)
1 Select from $O$ a set of $M$ patterns as initial centroids.
2 Repeat
        Form $M$ clusters by assigning each pattern to its closest centroid.
        Re-compute the centroid of each cluster.
3 until a stopping criterion is met.
end k-means
```

Fig. 4 Framework of the *k*-means algorithm

```
algorithm bisection k-means ($O = \{o_1,\dots,o_N\}$, $M$, n)
1 $S := \{S_1\} := \{O\}$
2 Repeat
      Select and remove a cluster $S_t$ from the set $S$
      For n trials, bisect $S_t$ in $S_t^1$ and $S_t^2$ using k-means and retaining the best
      bisection $S_t^{b1}$ and $S_t^{b2}$
      $S := S \cup \{S_t^{b1}, S_t^{b2}\}$
3 until $|S| = M$
end bisection k-means
```

Fig. 5 Framework of the bisection *k*-means algorithm

measure among objects. It has to be furthermore underlined that in a *k*-medoid algorithm, the medoid is by definition an object of the given set of objects O, while in a *k*-means approach this property is not necessarily satisfied by a centroid.

A typical *k*-means algorithm is described in Fig. 4. It starts with a random initial partition of the data objects and keeps reassigning objects to "close" clusters until a convergence criterion is met. The most used stopping criteria are no (or minimal) reassignment of patterns to new cluster centers or minimal decrease in squared error. In this latter case, when the stopping criterion is related to the squared error, to evaluate a clustering it is generally used as objective function a function that takes into account the squared error: among different clustering of the same set of objects it is preferred the one corresponding to the minimum squared error, since this means that its centroids better represent the objects in their own cluster.

The main drawback of the *k*-means algorithm is that the quality of the solution it returns strongly depends upon the selection of the initial partition. If the initial partition is not properly chosen, the approach may converge to a local minimum of the criterion function value. In the attempt to overcome this important drawback of the *k*-means approach, a variant of the method has been proposed called *bisection k*-means [67, 68]. The *bisection k*-means initially splits the objects into two clusters, then further splits one of the just created clusters, and so on, iteratively, until M clusters are individuated (obtaining, as side effect, hierarchical clusters).

A typical bisection *k*-means approach is described in Fig. 5. In line 1, the algorithm initializes the set of clusters to contain only one cluster S_1 containing all objects. Then, in the loop at lines 2–3, until a set S of M clusters is attained, the algorithm iteratively selects and removes from the current set S a cluster S_t and for a given in input number n of trials it bisects S_t into two clusters, retaining the best bisection S_t^{b1} and S_t^{b2} (corresponding, for example, to the lowest squared error) and adding it to the set S under construction.

The study and the design of efficient initialization methods for the k-means technique is still a research topic that attracts the efforts by various scientific communities. A recent survey and comparative study of the existing methods has been conducted by Celebi et al. [15], who cited as particularly interesting two hierarchical initialization methods named Var-Part and PCA-Part proposed in 2007 by Su and Dy [69]. These two methods are not only linear, deterministic, and order-invariant. Besides these nice characteristics, in a recent paper by Celebi and Kingravi [13], a discriminant analysis-based approach has been proposed that addresses a common deficiency of these two methods. A deep experimental analysis showed that the two methods are highly competitive with state-of-the-art best random initialization methods to date and that the proposed approach significantly improves the performance of both hierarchical methods. Finally, in [14], Celebi and Kingravi presented an in-depth comparison of six linear, deterministic, and order-invariant initialization methods.

5.2.2 Graph-Theoretic Algorithms

Most of the graph-theoretic algorithms are divisive approaches, i.e., they start with one cluster that contains all the objects and then at each step they split a cluster. Specularly, agglomerative algorithms start with each pattern in a distinct (singleton) cluster, and successively merge clusters together until a stopping criterion is satisfied.

The graph-theoretic algorithms use a graph representation of the Data sets. Formally, given the set of objects $O = \{o_1, \ldots, o_N\}$ and the distance function $d : O \times O \mapsto R$, a weighted undirected graph $G = (V, E, w)$ can be defined such that

- $V = O$;
- edges in E indicate the relationship between objects;
- $w_{ij} = d_{ij}$, $\forall\ i, j \in V$ (i.e., $o_i, o_j \in O$).

The graph G is usually called *proximity graph*. It is very easy to see that each cluster can correspond to a *connected component* of G, i.e., a subset of nodes/objects that are connected to one another [63]. A further stronger possible graph-theoretic approaches family looks for *cliques* in G, i.e., sets of nodes/objects in the graph that are completely connected to each other [6, 55].

The easiest and maybe most famous graph-theoretic divisive clustering algorithm is based on the construction of a minimal spanning tree (MST) of the graph G [78]. Once obtained a MST, it generates clusters by deleting from the MST the edges with the largest weights.

5.2.3 Mixture-Resolving Algorithms

The idea behind this family of clustering algorithms is that the objects in O are drawn from one of several distributions (usually, Gaussian) and the goal is to identify the parameters of each distribution (e.g., a maximum likelihood estimate). Traditional approaches to this problem involve obtaining (iteratively) a maximum likelihood estimate of the parameter vectors of the component densities.

Among these techniques, it has to be cited the Expectation Maximization (EM) algorithm [19], a general purpose maximum likelihood algorithm for missing-data problems. The EM algorithm has been applied to the problem of parameter estimation.

5.3 Efficient Metaheuristic Approaches

Metaheuristics for data clustering have been designed only in the last 20 years. They include artificial neural networks [39, 64], evolutionary approaches such as genetic algorithms [42, 59], simulated annealing [47], and tabu search [2, 31].

Starting from 2010, GRASP (Greedy Randomized Adaptive Search Procedure) algorithms [24, 30, 55] have also been proposed that model the problem of pattern clustering as a combinatorial optimization problem defined on the weighted graph $G = (V, E, w)$ representing the data sets, as also used by graph-theoretic algorithms. GRASP, originally proposed by Feo and Resende [23] for set covering, has been applied in a wide range of problem areas [25–27, 61, 62]. It is a multi-start process, where at each iteration, a greedy randomized solution is constructed to be used as a starting solution for local search. Local search repeatedly substitutes the current solution by a better solution in the neighborhood of the current solution. If there is no better solution in the neighborhood, the current solution is returned as a local minimum and the search stops. The best local minimum found over all GRASP iterations is output as the final solution.

In 2010, Nascimento et al. [55] proposed a GRASP algorithm to cluster biological data sets. The greedy randomized solution is iteratively built, starting from a complete graph $\left(|E| = \frac{N\cdot(N-1)}{2}\right)$, indicating that the data set forms a unique cluster. Then, at each iteration of the construction procedure edges are gradually eliminated from the graph, creating unconnected full subgraphs (cliques), each representing a cluster. The edge elimination follows a greedy randomized criterion that selects at random one edge in a subset (RCL—Restricted Candidate List) of higher weighted edges. Once a greedy randomized clustering $S = \{S_1, \ldots, S_M\}$ has been obtained, a local search procedure is applied starting from S to attempt to improve it. The local search works in an iterative fashion, until no better solution is found in the neighborhood. At each iteration, it replaces the current solution S by a better solution in the neighborhood $N(S)$ obtained by transferring in S an object from a cluster to another one.

In 2011, Frinhani et al. [30] described several hybrid GRASP with path-relinking heuristics [32, 48, 62] for data clustering. In a GRASP with path-relinking [1, 48, 62], at each GRASP iteration an elite set of good-quality solutions is stored and eventually updated. In fact, the local optimal current solution is combined with a randomly selected solution from the elite set using the path-relinking operator. The best of the combined solutions is a candidate for inclusion in the elite set and is added to the elite set if it meets quality and diversity criteria. The path-relinking procedure proposed in [30] applies to a pair of solutions S_s (starting solution) and S_t (target solution). Initially, the procedure computes the symmetric difference $\Delta(S_s, S_t)$, i.e., the set of moves needed to reach S_t (target solution) from S_s (initial solution). A path of solutions in the solution space is generated linking S_s and S_t and the best solution S^* in this path is returned by the algorithm. At each step, the procedure examines all moves $m \in \Delta(S, S_t)$ from the current solution S and selects the one corresponding to the least cost solution, i.e., the one which minimizes the objective function evaluates in $S \oplus m$, the solution resulting from applying move m to solution S. The best move m^* is made, producing solution $S \oplus m^*$. The set of available moves is updated. If necessary, the best solution S^* is updated. The procedure terminates when S_t is reached, i.e., when $\Delta(S, S_t) = \emptyset$.

In 2013 [28], a Biased Random-Key Genetic Algorithm (BRKGA) has been proposed for data clustering. It is well known that in the attempt of finding good quality solutions for a combinatorial optimization problem, Genetic Algorithms (GAs) [33, 36] implement the concept of *survival of the fittest*, making an analogy between a solution and an *individual* in a *population*. In particular, each individual of the current population represents a feasible solution, that is encoded by a corresponding *chromosome* that consists of a string of *genes*. Each gene can take on a value, called an *allele*, from some alphabet. For each chromosome it is possible to evaluate its *fitness level*, which is clearly correlated to the corresponding objective function value of the solution it encodes. GAs keep proceeding over a number of iterations, called *generations*, evolving a population of chromosomes. This *evolution* is implemented by simulating the process of natural selection through mating and mutation.

Genetic algorithms with random keys, or *random-key genetic algorithms* (RKGA), were introduced by Bean [5]. In a RKGA, chromosomes are represented as vectors of randomly generated real numbers in the interval $(0, 1]$. A deterministic algorithm, called a *decoder*, takes as input a solution vector and associates with it a solution of the combinatorial optimization problem for which an objective value or fitness can be computed.

A RKGA evolves a population of random-key vectors over a number of generations. The initial population is made up of $p > 0$ vectors of random-keys. Each component of the solution vector is generated independently at random in the real interval $(0, 1]$. After the fitness of each individual is computed by the decoder in generation t, the population is partitioned into two groups of individuals: a small group of p_e *elite* individuals, i.e., those with the best fitness values, and the remaining set of $p - p_e$ *non-elite* individuals. To evolve the population, a new generation of individuals must be produced. All elite individuals of the population

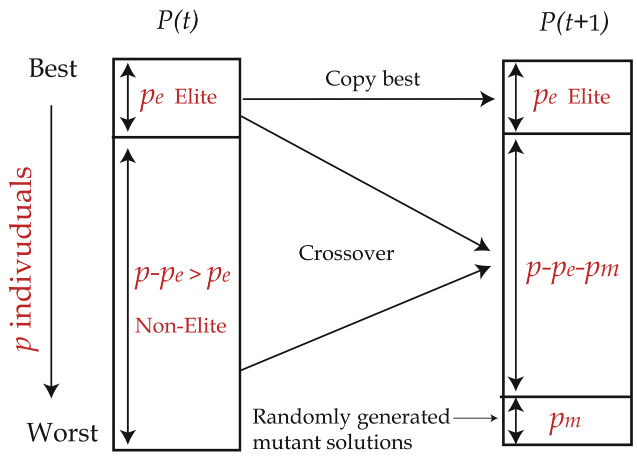

Fig. 6 BRKGA: generation $t+1$ from generation t

of generation t are copied without modification to the population of generation $t+1$. RKGAs implement mutation by introducing *mutants* into the population. A mutant is simply a vector of random keys generated in the same way that an element of the initial population is generated. At each generation (see Fig. 6), a small number (p_m) of mutants are introduced into the population. With the p_e elite individuals and the p_m mutants accounted for in population $k+1$, $p-p_e-p_m$ additional individuals need to be produced to complete the p individuals that make up the new population. This is done by producing $p-p_e-p_m$ offspring through the process of mating or crossover.

Bean [5] selects two parents at random from the entire population to implement mating in a RKGA. A *biased random-key genetic algorithm*, or BRKGA [34], differs from a RKGA in the way parents are selected for mating. In a BRKGA, each element is generated combining one element selected at random from the elite partition in the current population and one from the non-elite partition. Repetition in the selection of a mate is allowed and therefore an individual can produce more than one offspring in the same generation. As in RKGA, *parametrized uniform crossover* [66] is used to implement mating in BRKGAs. Let $\rho_e > 0.5$ be the probability that an offspring inherits the vector component of its elite parent. Let m denote the number of components in the solution vector of an individual. For $i = 1, \ldots, m$, the ith component C_i of the offspring vector C takes on the value of the ith component E_i of the elite parent E with probability ρ_e and the value of the ith component $\bar{E}_i$ of the non-elite parent $\bar{E}$ with probability $1-\rho_e$.

When the next population is complete, i.e., when it has p individuals, fitness values are computed by the decoder for all of the newly created random-key vectors and the population is partitioned into elite and non-elite individuals to start a new generation.

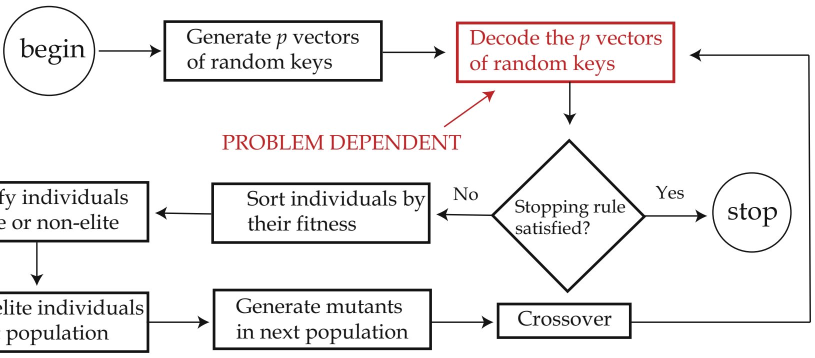
begin
Generate p vectors of random keys
Decode the p vectors of random keys
PROBLEM DEPENDENT
Stopping rule satisfied?
Yes
stop
No
Sort individuals by their fitness
Classify individuals as elite or non-elite
Copy elite individuals to next population
Generate mutants in next population
Crossover

Fig. 7 BRKGA: flow chart

A BRKGA searches the solution space of the combinatorial optimization problem indirectly by searching the continuous m-dimensional hypercube, using the decoder to map solutions in the hypercube to solutions in the solution space of the combinatorial optimization problem where the fitness is evaluated. As underlined in [34], it is evident then that any BRKGA heuristic is based on a general-purpose metaheuristic framework (see Fig. 7), in which there is a portion totally independent from the specific problem to be solved. The only connection to the combinatorial optimization problem being solved is the problem-dependent portion of the algorithm, where the decoder produces solutions from the vectors of random-keys and computes the fitness of these solutions. Therefore, to describe a BRKGA for a specific combinatorial optimization problem, one needs only to show how solutions are encoded as vectors of random keys and how these vectors are decoded to feasible solutions of the optimization problem. We report in the following encoding and decoding schemes proposed in [28] for data clustering problems.

A solution is denoted as $x = (x_1, \dots, x_N)$, such that for all $i = 1, \dots, N$, $x_i \in \{1, \dots, M\}$, i.e.,

$$\forall\, i = 1, \dots, N, \quad x_i = k, \text{ iff } o_i \in S_k,\ k \in \{1, \dots, M\}.$$

5.4 Encoding

A solution is encoded as a random-key vector $X = (X_1, X_2, \dots, X_N)$, where for $i = 1, \dots, N$, component X_i is a real number in the interval (0, 1].

5.5 Decoding

Given an encoded solution $X = (X_1, X_2, \ldots, X_N)$ and representing the set S as an array of $|S| = M$ elements, decoding consists of three steps and produces a string $x = (x_1, x_2, \ldots, x_N)$.

In the first step, the string x is computed such that, for each $i = 1, \ldots, N$,

$$x_i = \left\lceil X_i \cdot \frac{1}{\Delta} \right\rceil, \qquad \Delta = \frac{1}{M}.$$

In the second step, starting from x, a locally optimal solution $\hat{x} = (\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_N)$ with respect to the 2-swap neighborhood is computed.

Finally, in the third step, a chromosome adjustment is made to the encoded solution such that the resulting chromosome $\hat{X}$ decodes directly to $\hat{x}$ using only the first step of the decoding process.

For $j = 1, \ldots, |S| = M$, let

$$\begin{aligned} l_j &= (j-1) \cdot \Delta; \\ u_j &= j \cdot \Delta. \end{aligned}$$

Then, to complete the adjustment, for each $i = 1, \ldots, N$, it is simply computed

$$\hat{X}_i = l_j + X_i \cdot \Delta,$$

where j is the index of the subset in S to which object o_i belongs according to solution $\hat{x}$.

6 Concluding Remarks

The scope of this paper is to provide an overview of the main types of clustering and criteria for homogeneity or separation, with special emphasis to the optimization and operational research perspective, providing a few mathematical models of the problem under several different criterion adopted to classify the data.

Besides mathematical models that can be efficiently used to find exact solutions only in case of small-sized instances, there are hundreds of approximate techniques, proposed by researchers from several heterogenous communities, and more or less efficient, depending on the input data and the type of information they contain.

The conclusion is that unfortunately there is no single best approach that wins in every aspect. Given the intrinsic difficulties to be faced when clustering data, a person interested in performing this task should select a group of algorithms that seem to be the most appropriate, taking into account also the specific data set under analysis.

References

1. Aiex, R.M., Binato, S., Resende, M.G.C.: Parallel GRASP with path-relinking for job shop scheduling. Parallel Comput. **29**, 393–430 (2003)
2. Al-Sultan, K.S.: A tabu search approach to clustering problems. Pattern Recognit. **28**, 1443–1451 (1995)
3. Alon, U., Barkai, N., Notterman, D.A., Gish, K., Ybarra, S., Mack, D., et al.: Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. In: Proceedings of the National Academy of Sciences, pp. 6745–6750 (1999)
4. Anderberg, M.R.: Cluster Analysis for Applications. Academic, New York (1973)
5. Bean, J.C.: Genetic algorithms and random keys for sequencing and optimization. ORSA J. Comput. **6**, 154–160 (1994)
6. Ben-Dor, A., Shamir, R., Yakhini, Z.: Clustering gene expression patterns. J. Comput. Biol. **6**(3/4), 281–297 (1999)
7. Bickel, D.R.: Robust cluster analysis of DNA microarray data: an application of nonparametric correlation dissimilarity. In: Proceedings of the American Statistical Association (2001)
8. Boginski, V., Butenko, S., Pardalos, P.M.: Network models of massive datasets. Comput. Sci. Inf. Syst. **1**(1), 75–89 (2004)
9. Boginski, V., Butenko, S., Pardalos, P.M.: Mining market data: a network approach. Comput. Oper. Res. **33**(11), 3171–3184 (2006)
10. Busygin, S., Prokopyev, O.A., Pardalos, P.M.: Feature selection for consistent biclustering via fractional 0–1 programming. J. Comb. Optim. **10**(1), 7–21 (2005)
11. Busygin, S., Prokopyev, O.A., Pardalos, P.M.: Biclustering in data mining. Comput. Oper. Res. **39**(9), 2964–2987 (2008)
12. Butenko, S., Chaovalitwongse, W.A., Pardalos, P.M. (eds.): Clustering Challenges in Biological Networks. World Scientific, Singapore (2009)
13. Celebi, M.E., Kingravi, H.A.: Deterministic initialization of the *k*-means algorithm using hierarchical clustering. Int. J. Pattern Recogn. Artif. Intell. **26**(7), 1250018 (2012)
14. Celebi, M.E., Kingravi, H.: Linear, deterministic, and order-invariant initialization methods for the *k*-means clustering algorithm. In: Celebi, M.E. (ed.) Partitional Clustering Algorithms, pp. 79–98. Springer, Cham (2014)
15. Celebi, M.E., Kingravi, H.A., Vela, P.A.: A comparative study of efficient initialization methods for the *k*-means clustering algorithm. Expert Syst. Appl. **40**(1), 200–210 (2013)
16. Charikar, M., Chekuri, C., Feder, T., Motwani, R.: Incremental clustering and dynamic information retrieval. In: Proceedings of the 29th Annual ACM Symposium on Theory of Computing, pp. 626–634 (1997)
17. Cutting, D.R., Karger, D.R., Pedersen, J.O., Tukey, J.W.: Scatter/gather: a cluster-based approach to browsing large document collections. In: Fifteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 318–329 (1992)
18. Cutting, D.R., Karger, D.R., Pedersen, J.O.: Constant interaction-time scatter/gather browsing of very large document collections. In: Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 126–134 (1993)
19. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. J. R. Stat. Soc. **B39**, 1–38 (1977)
20. Ding, C.H.Q., Dubchak, I.: Multi-class protein fold recognition using support vector machines and neural networks. Bioinformatics **17**(4), 349–358 (2001)
21. Fan, N., Pardalos, P.M.: Multi-way clustering and biclustering by the Ratio cut and Normalized cut in graphs. J. Comb. Optim. **23**(2), 224–251 (2012)
22. Fan, Y.-J., Chaovalitwongse, W.A., Liu, C.-C., Sachdeo, R.C., Iasemidis, L.D., Pardalos, P.M.: Optimisation and data mining techniques for the screening of epileptic patients. Int. J. Bioinform. Res. Appl. **5**(2), 187–196 (2009)

23. Feo, T.A., Resende, M.G.C.: A probabilistic heuristic for a computationally difficult set covering problem. Oper. Res. Lett. **8**, 67–71 (1989)
24. Feo, T.A., Resende, M.G.C.: Greedy randomized adaptive search procedures. J. Glob. Optim. **6**, 109–133 (1995)
25. Festa, P., Resende, M.G.C.: GRASP: an annotated bibliography. In: Ribeiro, C.C., Hansen, P. (eds.) Essays and Surveys on Metaheuristics, pp. 325–367. Kluwer, Norwell (2002)
26. Festa, P., Resende, M.G.C.: An annotated bibliography of GRASP – part I: algorithms. Int. Trans. Oper. Res. **16**(1), 1–24 (2009)
27. Festa, P., Resende, M.G.C.: An annotated bibliography of GRASP – part II: applications. Int. Trans. Oper. Res. **16**(2), 131–172 (2009)
28. Festa, P.: A biased random-key genetic algorithm for data clustering. Math. Biosci. **245**(1), 76–85 (2013)
29. Festa, P.: On data clustering: exact and approximate solutions. In: Butenko, S., et al. (eds.) Examining Robustness and Vulnerability of Networked Systems, pp. 65–82. IOS Press, Fairfax (2014)
30. Frinhani, R.M.D., Silva, R.M.A., Mateus, G.R., Festa, P., Resende, M.G.C.: Grasp with path-relinking for data clustering: a case study for biological data. In: Proceedings of the 10th International Symposium on Experimental Algorithms, SEA'11, pp. 410–420. Springer, Berlin/Heidelberg (2011)
31. Glover, F.: Tabu search and adaptive memory programing – advances, applications and challenges. In: Barr, R.S., Helgason, R.V., Kennington, J.L. (eds.) Interfaces in Computer Science and Operations Research, pp. 1–75. Kluwer, Boston (1996)
32. Glover, F., Laguna, M., Martí, R.: Fundamentals of scatter search and path relinking. Control Cybern. **39**, 653–684 (2000)
33. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Boston (1989)
34. Gonçalves, J.F., Resende, M.G.C.: Biased random-key genetic algorithms for combinatorial optimization. J. Heuristics **17**(5), 487–525 (2011)
35. Hammer, P.L., Rudeanu, S.: Boolean Methods in Operations Research and Related Areas. Springer, Heidelberg (1968)
36. Holland, J.H.: Adaptation in Natural and Artificial Systems. MIT Press, Cambridge (1975)
37. Hubert, L., Arabie, P.: Comparing partitions. J. Classif. **2**, 193–218 (1985)
38. Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data. Prentice-Hall, Englewood Cliffs (1988)
39. Jain, A.K., Mao, J.: Neural networks and pattern recognition. In: Zurada, J.M., Marks, R.J., Robinson, C.J. (eds.) Computational Intelligence: Imitating Life, pp. 194–212. IEEE Press, New York (1994)
40. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Comput. Surv. **31**(3), 264–323 (1999)
41. Jiang, D., Tang, C., Zhang, A.: Cluster analysis for gene expression data: a survey. IEEE Trans. Knowl. Data Eng. **16**(11), 1370–1386 (2004)
42. Jones, D., Beltramo, M.A.: Solving partitioning problems with genetic algorithms. In: Proceedings of the Fourth International Conference on Genetic Algorithms, pp. 442–449 (1991)
43. Kaufman, L., Rousseeuw, P.J.: Statistical Data Analysis Based on the L1-Norm and Related Methods. North-Holland, Amsterdam (1987)
44. Kaufman, L., Rousseeuw, P.J.: Finding Groups in Data: An Introduction to Cluster Analysis. Wiley, New York (2005)
45. Kocheturov, A., Batsyn, M., Pardalos, P.M.: Dynamics of cluster structures in a financial market network. Phys. A Stat. Mech. Appl. **413**, 523–533 (2014)
46. King, B.: Step-wise clustering procedures. J. Am. Stat. Assoc. **69**, 86–101 (1967)
47. Klein, R.W., Dubes, R.C.: Experiments in projection and clustering by simulated annealing. Pattern Recogn. **22**, 213–220 (1989)

48. Laguna, M., Martí, R.: GRASP and path relinking for 2-layer straight line crossing minimization. INFORMS J. Comput. **11**, 44–52 (1999)
49. Liu, C.-C., Chaovalitwongse, W.A., Pardalos, P.M., Uthman, B.M.: Dynamical feature extraction from brain activity time series. In: Encyclopedia of Data Warehousing and Mining, pp. 729–735. IDEA Group, Hershey (2009)
50. Ma, P.C.H., Chan, K.C.C., Yao, X., Chiu, D.K.Y.: An evolutionary clustering algorithm for gene expression microarray data analysis. IEEE Trans. Evol. Comput. **10**(3), 296–314 (2006)
51. Mao, J., Jain, A.K.: A self-organizing network for hyperellipsoidal clustering (hec). IEEE Trans. Neural Netw. **7**, 16–29 (1996)
52. McQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the 5.th Berkeley Symposium on Mathematical Statistics and Probability, pp. 281–297 (1967)
53. Mucherino, A., Papajorgji, P., Pardalos, P.M.: A survey of data mining techniques applied to agriculture. Oper. Res. **9**(2), 121–140 (2009)
54. Nascimento, M.C.V.: Uma heurística GRASP para o problema de dimensionamento de lotes com múltiplas plantas. PhD thesis, USP (2007)
55. Nascimento, M.C.V., Toledo, F.M.B., de Carvalho, A.C.P.L.F.: Investigation of a new GRASP-based clustering algorithm applied to biological data. Comput. Oper. Res. **37**(8), 1381–1388 (2010)
56. Pardalos, P.M., Hansen, P. (eds.) : Data Mining and Mathematical Programming. American Mathematical Society, Providence (2008)
57. Pardalos, P.M., Coleman, T.F., Xanthopoulos, P. (eds.): Optimization and Data Analysis in Biomedical Informatics. Springer Series: Fields Institute Communications, vol. 63, 150 p. Springer, New York (2012). ISBN 978-1-4614-4132-8
58. Porter, M.: Location, competition, and economic development: local clusters in a global economy. Econ. Dev. Q. **14**(1), 15–34 (2000)
59. Raghavan, V.V., Birchand, K.: A clustering strategy based on a formalism of the reproductive process in a natural system. In: Second International Conference on Information Storage and Retrieval, pp. 10–22 (1979)
60. Rao, M.R.: Cluster analysis and mathematical programming. J. Am. Stat. Assoc. **66**(335), 622–626 (1971)
61. Resende, M.G.C., Ribeiro, C.C.: Greedy randomized adaptive search procedures. In: Glover, F., Kochenberger, G. (eds.) Handbook of Metaheuristics, pp. 219–249. Kluwer, New York (2002)
62. Resende, M.G.C., Ribeiro, C.C.: GRASP with path-relinking: recent advances and applications. In: Ibaraki, T., Nonobe, K., Yagiura, M. (eds.) Metaheuristics: Progress as Real Problem Solvers, pp. 29–63. Springer, New York (2005)
63. Shamir, R., Sharan, R.: CLICK: a clustering algorithm for gene expression analysis. In: Proc. Eighth Int'l Conf. Intelligent Systems for Molecular Biology (ISMB '00) (2000)
64. Sethi, I., Jain, A.K. (eds.): Artificial Neural Networks and Pattern Recognition: Old and New Connections. Elsevier, New York (1991)
65. Sneath, P.H.A., Sokal, R.R.: Numerical Taxonomy. Freeman, San Francisco (1973)
66. Spears, W.M., DeJong, K.A.: On the virtues of parameterized uniform crossover. In: Proceedings of the Fourth International Conference on Genetic Algorithms, pp. 230–236 (1991)
67. Steinbach, M., Karypis, G., Kumar, V.: On the virtues of parameterized uniform crossover. In: Proceedings of World Text Mining Conference, KDD2000 (2000)
68. Steinbach, M., Karypis, G., Kumar, V.: A comparison of document clustering techniques. In: Proceedings of KDD Workshop on Text Mining Conference, KDD2000 (2000)
69. Su, T., Dy, J.G.: In search of deterministic methods for initializing *k*-means and Gaussian mixture clustering. Intell. Data Anal. **11**(4), 319–338 (2007)
70. Ushioda, A., Kawasaki, J.: Hierarchical clustering of words and application to NLP tasks. In: Ejerhed, E., Dagan, I. (eds.) Fourth Workshop on Very Large Corpora, pp. 28–41. Association for Computational Linguistics, Cambridge (1996)

71. Valery, K.A., Koldanov, A.P., Pardalos, P.M.: A general approach to network analysis of statistical data sets In: Pardalos, P.M., Resende, M.G.C., Vogiatzis, C., Walteros, J.L. (eds.) Learning and Intelligent Optimization. 8th International Conference, Lion 8, Gainesville, FL, 16–21 February 2014. Revised Selected Papers. Lecture Notes in Computer Science, vol. 8426, pp. 88–97. Springer, Berlin/Heidelberg (2014)
72. Ward Jr., J.H.: Hierarchical grouping to optimize an objective function. J. Am. Stat. Assoc. **58**, 236–244 (1963)
73. White, S.D.M., Frenk, C.S.: An optimal graph theoretic approach to data clustering: theory and its application to image segmentation. Astrophys. J. **379**(Part 1), 52–72 (1991)
74. Willet, P.: Recent trends in hierarchic document clustering: a critical review. Inf. Process. Manag. **24**(5), 577–597 (1988)
75. Wu, Z., Leahy, R.: Galaxy formation through hierarchical clustering. IEEE Trans. Pattern Anal. Mach. Intell. **15**(11), 1101–1013 (1993)
76. Xanthopoulos, P., Pardalos, P.M., Trafalis, T.B.: Robust Data Mining. Springer Briefs in Optimization, vol. XII, 59 p. Springer, New York (2013)
77. Xu, R., Wunsch, D.: Survey of clustering algorithms. IEEE Trans. Neural Netw. **16**(3), 645–678 (2005)
78. Zahn, C.T.: Graph-theoretical methods for detecting and describing gestalt clusters. IEEE Trans. Comput. **C-20**, 68–86 (1971)