ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ВЫСШАЯ ШКОЛА ЭКОНОМИКИ

Alexander Rubchinsky

A DIVISIVE-AGGLOMERATIVE CLASSIFICATION ALGORITHM BASED ON THE MINIMAX MODIFICATION OF FREQUENCY APPROACH

Препринт WP7/2010/07
Серия WP7
Математические методы
анализа решений в экономике,
бизнесе и политике

Москва
2010

УДК 519.8
ББК 22.18
R 88

R88 **Rubchinsky, A.**
A Divisive-Agglomerative Classification Algorithm Based on the Minimax Modification of Frequency Approach : Working paper WP7/2010/07 [Text] / A. Rubchinsky ; The University – Higher School of Economics. – Moscow: Publishing House of the University – Higher School of Economics, 2010. – 48 p. – 150 copies.

The conventional problem of automatic classification (AC for brevity) is considered. The suggested approach is based on the new combinations of known methods and their modifications. At first, consecutive dichotomies of the initial set are produced, whereby a family of classifications consisting of 2, 3, ..., *k* subsets is constructed, where *k* is a number certainly exceeding the assumed number of classes (the divisive stage). The dichotomy used is a new modification of the frequency method, which naturally includes elements of randomization. Second, each of the constructed classifications generates a new family of classifications by consecutively uniting the subsets that are the closest to one another (the agglomerative stage). After that, only non-coinciding classifications are left for further analysis. Finally, the process is repeated several times, with the result that most of the classifications turn out to be stochastically unstable. A stable classification with the maximal number of classes is declared to be the correct solution of the initial AC, while the absence of stable classifications is interpreted as the absence of cluster structure in the initial set.

УДК 519.8
ББК 22.18

Rubchinsky Alexander – Decision Analysis Laboratory at Higher School of Economics and International University “Dubna”

1. Introduction

The well-known problem of automatic classification (further referred to as AC, for brevity) consists in the division of a given set of objects into several non-intersecting subsets (usually called classes, aggregates, clusters, etc.). It is required that objects belonging to the same class be in one sense or another, closely connected or similar, whereas objects belonging to different classes should be as dissimilar as possible and could easily be discernible. The informal character of the AC problem, its various statements and applications, numerous approaches and methods of solution are comprehensively described in several monographs and reviews (see e.g. Aivazyan et al, 1989, Barseguyan et al, 2007, Braverman and Muchnik, 1983, Filippone et al, 2008, Gordon, 1999, Luxburg, 2007, Mirkin, 1996, Mirkin, 2005, Newman and Girvan, 2002, Newman, 2004).

In this paper, the AC problem is considered in the most conventional form. Namely, it is assumed that a set of objects is given such that for all pairs of these objects a ***degree of dissimilarity*** (or ***similarity***) has been already determined. The information on dissimilarity/ similarity is usually presented in one of the following three ways:

1) a pattern matrix (also called entity-to-variable data table and objects/parameters matrix);
2) a dissimilarity (similarity) matrix;
3) an undirected graph.

In order to solve the AC problem, a new algorithm is proposed. This algorithm finds the "correct" classes in diverse situations proceeding only from the initial data (the set of points in the Euclidean space, dissimilarity matrix, graph), with no additional assumptions of stochastic, geometric or other character, either explicit or implicit. Despite the large number of methods proposed for the solution of the AC problem, there are absolutely no methods that could find intuitively correct classifications even in simple cases of various characters. (An attempt to produce even a cursory review of these methods would make this paper several times longer.) Let us consider, as an example, the six sets shown in Fig.1. Even though it is possible to find a method coping with it for any of these sets, none of these methods is able to cope with all the sets.

In the suggested approach, the initial data on the problem are presented by the well-known neighborhood graph (see e.g. Luxburg,

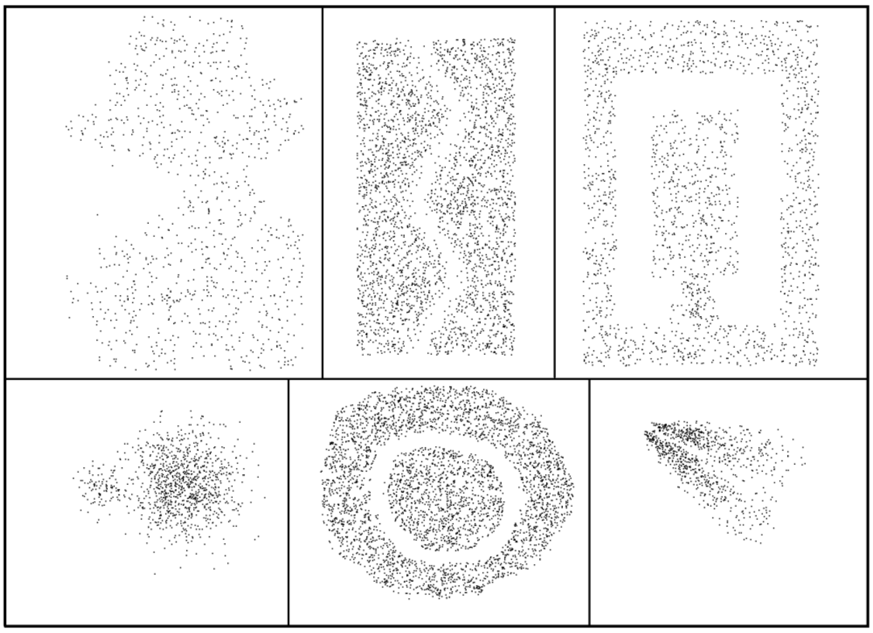


Fig.1. Simple two-dimensional classification problems

2007). Graph vertices are in one-to-one correspondence to the given objects. Any vertex *v* is connected to 4 or 5 other vertices, which correspond to the objects that are closest to the object corresponding to vertex *v*. The proximity of objects is determined either immediately by the given dissimilarity matrix or by the Euclidean distances between the objects, calculated from the given pattern matrix. It is worth noting that the presentation of AC problem by a graph is the most general description – exactly because it uses the "softest" (non-numerical) data on the connections between objects to be classified. In the framework of the suggested approach, only such essentially qualitative data on the connections are used.

Basically, throughout the paper I consider the sets of points on the plane (two-dimensional points). The idea is as follows. It is well known that AC is an informal problem. For two-dimensional sets considered, intuitively correct classifications are obvious. If a formal method cannot find them, then it is hardly probable that the same method is able to reveal cluster structures in more complicated multi-dimensional cases or in the problems presented by the dissimilarity matrix. Still, as will be further demonstrated, the suggested approach copes not only with two-dimensional problems (substantially more complicated than the problems shown in Fig.1), but also with multi-dimensional data and the dissimilarity matrix.

The analysis of known AC methods demonstrates that they successfully tackle some complicated AC problems and fail in other problems seemingly much simpler. Importantly, both types of problems are different for different methods. It is probably the dissatisfaction with this fact that triggers the emergence of more and more AC methods; still, the situation is practically not improving. One can therefore assume that no formal model could be proposed that could offer a sufficiently description of correct classification. Therefore the approach suggested here (unlike many others, including those ones on which our approach is based technically) does not attempt at finding the ***only*** correct classification using ***only one*** formal model. Instead, a multistage procedure is proposed. The result of each stage is a ***family*** of classifications which is first ***gradually expanded*** and then is ***gradually contracted*** so that the output of the entire procedure almost always amounts to one classification. From this viewpoint, the approach is close to genetic algorithms because the latter also deal with sets of solutions (populations) rather than with individual solutions. An additional advantage of this approach is that it is possible to stop after a particular stage and select one of the few remaining classifications using some of the known methods. Since the remaining classifications are reasonable enough, applying the known methods can prove essentially more efficient than applying them from the start, which requires hard calculations with no guarantee of results.

The material is structured as follows. In the Introduction, the particulars of the suggested approach are briefly described. In Section 2, the used version of the frequency method of dichotomy is presented in more detail. Section 3 outlines the construction of a family of classifications using a new divisive-agglomerative procedure (further, DAP for brevity). In Section 4 the most external cycle of the suggested general AC algorithm is presented: iterative constructions of families of classifications and the selection of stochastically stable classifications, i.e. classifications belonging to all families found by DAP under different initializations of the randomizer. Section 5 offers examples illustrating all stages of the suggested general AC algorithm. In Section 6, classifications found by the suggested algorithm are compared with those obtained by several known methods for the same initial data. In conclusion, the main features of the suggested approach to AC are summarized and some new statements are discussed.

2. Frequency Dichotomy Algorithms

2.1. Newman-Girvan Algorithm. In the article of Newman and Girvan, 2002, an entirely new approach to graph decomposition – and thereby to the AC problem – was suggested. A cut of the initial graph is found as a result of some operations with no prior optimization requirements or other conditions imposed on this cut. We will outline the essence by citing the article.

"We define the edge betweenness of an edge as the number of shortest paths between pairs of vertices that run along it. If there is more than one shortest path between a pair of vertices, each path is given equal weight such that the total weight of all the paths is unity. If a network contains communities or groups that are only loosely connected by a few intergroup edges, then all shortest paths between different communities must go along one of these few edges. Thus, the edges connecting communities will have high edge betweenness. By removing these edges, we separate groups from one another and so reveal the underlying community structure of the graph."

The formal algorithm for identifying communities is presented in the article as follows.

<u>Newman-Girvan Algorithm</u>

1. Calculate the betweenness for all edges in the network.
2. Remove the edge with the highest betweenness.
3. Recalculate betweennesses for all edges affected by the removal.
4. Repeat from step 2 until no edges remain.

It is clear that during the execution of the algorithm every increment (by 1) of the number of network connectivity components amounts to the division of one of the groups into two parts, which results in an emergence of a hierarchical structure of groups (or communities) determined only by the initial graph. The calculation of betweenness degree is reduced to the determination of shortest paths for all pairs of vertices; as it is well known, this is a computationally efficient operation with an upper estimation of n^2. Subsequently (see Newman, 2004) several modifications of this approach have been suggested, the most important of which being:

- use of random (instead of the shortest) paths for the calculation of edge betweenness;
- use of a relatively small part of pairs of vertices (instead of using all of them) for the estimation of edge betweenness;
- edge removal based on this estimation.

In view of the above, it seems more convenient to use, instead of the notion of "edge betweenness", the notion of "edge frequency" which should be understood as the number of occurrences of edges in the constructed paths. With these modifications, the algorithm of graph division into two parts can be described as follows.

<u>Generalized Newman-Girvan Algorithm</u>

1. Set the current frequency at every edge equal to zero.
2. Choose randomly two vertices of the graph.
3. Find by any method a path between vertices chosen at the previous step. If no such path exists, go to step 7.
4. Add 1 to frequencies at all edges included in the path found at step 3.
5. Under certain conditions return to step 2. Such conditions may include, e.g. the fact that steps 2 to 4 have been applied a certain large number of times, or that stochastic stability has been achieved, i.e. when the indices of edges with the maximal frequency have not changed for a long time (obviously, different realizations of this step are possible).
6. Remove the edge with the maximal frequency and return to step 1.
7. Stop. The graph is divided into two connectivity components that correspond to the required groups.

The above approach could be naturally referred to as ***frequency*** approach, since it is based on the calculation of frequencies of occurrence of graph edges in the consecutively constructed paths. It can be applied to every AC problem provided it is presented by a graph, in particular, by a neighborhood graph mentioned previously. An obvious drawback of Newman-Girvan algorithm (recognized by its authors) is that after the removal of an edge with the highest betweenness at step 2 all accumulated statistics of edge betweenness is deleted and, hence, cannot be used subsequently. Had it been possible to save these data for the consecutive steps, it could essentially accelerate the algorithm. The following is said on the issue in the cited article by Newman and Girvan (2002): “To try to reduce the running time of the algorithm further, one might be tempted to calculate the betweennesses of all edges only once and then remove them in order of decreasing betweenness. We find however that this strategy does not work well, because if two communities are connected by more than one edge, then there is no guarantee that all of those edges will have high betweenness – we only know that at least one of them will. By recalculating betweennesses after the removal of each edge we ensure that at least one of the remaining edges between two communities will always have a high value.”

It should be added that the same is true for the generalized Newman-Girvan algorithm. The next section shows how to avoid this trap.

2.2. Algorithm of constructing a uniform cut. Note that in the previously proposed frequency algorithms any path connecting a certain pair of vertices is produced <u>independently of all the paths already produced</u>. However, if all paths already produced are taken into account we can obtain cuts between two sets of vertices whose all edges have the <u>same maximal frequency</u>. Then concurrent removal of all edges with the maximal frequency, performed once, produces the desired dichotomy of the graph.

We will first present the algorithm itself and then give the necessary comments and examples of its performance in various situations. Though the algorithm belongs to frequency algorithms of classification, it can be considered new due to the presence of essential distinctive features. Importantly, unlike previously known versions of frequency algorithm, the algorithm under discussion finds an approximate solution of some graph optimization problem which offers a reasonable, if, as in other cases, incomplete, estimation of classification correctness (see the end of this section for details).

<u>Minimax frequency algorithm of graph dichotomy</u>. The input of the algorithm is an undirected connected graph *G*. The algorithm has two integer parameters:

- the maximal initial value *f* of edge frequency;
- the number of repetition *T* for the collection of statistics.

1. Preliminary stage. Frequencies at each edge of the graph are initialized by integer numbers uniformly distributed over the segment $[0, f - 1]$.
2. Cumulative stage. Operations of steps 2.1 – 2.3 are repeated T times:
   2.1. Random choice of a pair of vertices of graph G.
   2.2. Construction of a minimax path (path connecting the two vertices chosen at step 2.1, whose longest edge is the shortest one among all such paths) by Deikstra algorithm. The length of an edge is its current frequency.
   2.3. Modification of frequencies. 1 is added to each frequency of all edges belonging to the path found at the previous step, 2.2.
3. Final stage.
   3.1. The maximal value (achieved after T repetitions) of frequency $f_{\max}$ at graph edges is stored.
   3.2. Operations of steps 2.1 – 2.3 are executed once.
   3.3. The new maximal value of frequency f_{mod} at graph edges is determined.
   3.4. If $f_{\text{mod}} = f_{\max}$, return to step 3.2; otherwise go to the next step 3.5.
   3.5. Deduct 1 from frequencies in all edges forming the last found path.
   3.6. Remove all the edges at which the frequency is equal to $f_{\max}$.
   3.7. Find two connectivity components of the modified graph. The two constructed sets of vertices form the solution of the considered dichotomy problem.

We will first of all prove that immediately before the execution of step 3.6 the set of all edges whose frequency is equal to the maximal one indeed contains a cut of graph G.

Statement 1. Prior to the execution of step 3.6:
a) the maximal value of frequency over all the edges of the graph is equal to $f_{\max}$, where $f_{\max}$ is the number stored at step 3.1;
b) the set of all the edges whose frequency is equal to $f_{\max}$, contains a cut of graph G.

Proof. Step 3.2 refers to steps 2.1 – 2.3. If a new minimax path at step 2.2 is found, exactly one of the following two cases is possible:

1. There exists a minimax path connecting the vertices chosen at step 2.1 whose all edges have frequencies lower than $f_{\max}$.
2. No such path exists.

In the first case, after every addition of 1 (at step 2.3) to frequencies at all edges of the given path their maximal value (taken over all edges of the graph) does not exceed $f_{\max}$. On the other hand, at least at one edge its frequency increases by 1, whilst at no edge the frequency can decrease. Both these facts mean that after a certain finite number t of executions of steps 3.2→3.3→3.4→3.2 ($t \leq m \bullet f_{\max} + 1$, where m is the number of edges in the graph) at step 2.2 we encounter with case 2. In case 2, at any path connecting vertices chosen at step 2.1, there exists at least one edge whose frequency is no smaller than $f_{\max}$. Since up to now we have only encountered with case 1, then, as it was previously established, no frequency exceeds $f_{\max}$. Therefore at any path connecting vertices chosen at step 2.1, there exists at least one edge whose frequency is equal to $f_{\max}$. Hence, the set of all the edges whose frequency equals $f_{\max}$, contains a cut of graph G. Adding 1 to frequencies of all edges of the constructed path at step 2.3 and subsequently subtracting 1 at the same edges at step 3.5 does not change frequencies, which proves a) and b) and, hence, completes the proof of statement 1■

Figures 2a and 2b illustrate cases 1 and 2 considered in the proof of Statement 1. The cut itself, of course, depends upon the selection of pairs of vertices and the distribution of frequencies at edges that formed itself prior to the execution of step 3.1. This is the reason why the cumulative stage (which claims the most part of the time) should be carried out. As a result of this stage the required cut becomes stable in the sense that the edges that form the cut no longer depend upon the number T of the constructed minimax paths. Yet this cut can depend on the initialization of the random generator. So, the presence (or absence) of a dependence of the cut (and, hence, the corresponding dichotomy) upon the initialization of random generator turns out to be an important feature of the AC problem itself rather than the classification method used (see Section 2.3 for more details).

Let us consider the connections between the proposed algorithm and the known optimization statements for a balanced cut in a graph. We will first introduce the necessary notations. Let N be the number of vertices, M the number of executions of steps 2.1 – 2.3 together (excluding the last one) that take place at stages 2 and 3 of the algorithm, A and B denote any division of the set of graph vertices, $d(A, B)$ denote

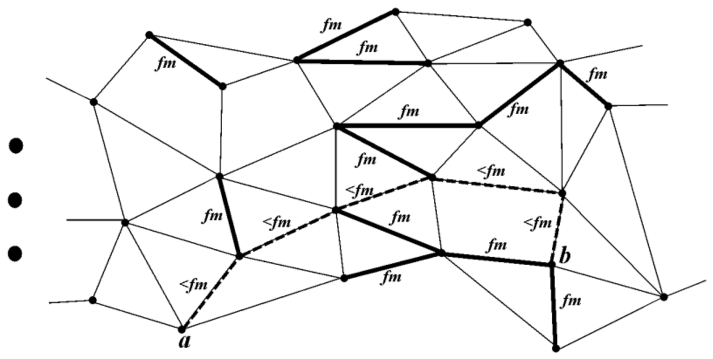


Fig.2a. The dashed line marks the path connecting vertices *a* and *b* in which all edges have frequencies lower than the maximal frequency f_m.

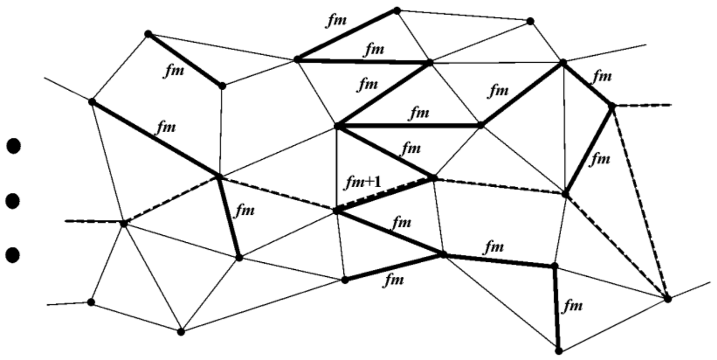


Fig.2b. The dashed line marks the path connecting vertices *a* and *b* located at either side of the cut, in which all edges have frequencies equal to the maximal frequency. Necessarily, such a path passes along an edge with the maximal frequency f_m.

the cardinality of cut (A, B). Note that M equals the number of all the constructed paths in the graph and $M \geq T$. Let us now consider all the paths (from among the constructed ones) whose one end belongs to A and the other end to B. The sum $S(A, B)$ of frequencies at all the edges from the cut (A, B) is no larger than the number of all such paths (to be denoted as $M(A, B)$). First, every path increases the sum of frequencies at least by one (if it intersects the cut (A, B) once, whereas some paths can intersect it several times); second, we need to add the initial frequency values (see the preliminary stage 1 of the algorithm). Since the vertices are chosen at random, the probability of the fact that one end of a path belongs to *A* and another to *B* is approximately equal to $(2\bullet|A|\bullet|B|) / N^2$. Therefore for the total number of such paths an approximate equality

$$M(A, B) \approx ((2\bullet|A|\bullet|B|)/N^2)*M \tag{1}$$

takes place. Assume (for the purpose of rough estimation) that any path from *A* to *B* intersects the cut (*A*, *B*) exactly once. Since the number of paths *M* significantly exceeds the maximal value of initial frequency *f*, the following rough estimation takes place:

$$S(A, B) \approx ((2\bullet|A|\bullet|B|)/N^2)*M. \tag{2}$$

Dividing both parts of this approximate equality by the number of edges in the cut (*A*, *B*), we receive

$$\bar{f}(A, B) = S(A, B)/d(A, B) \approx (((2\bullet|A|\bullet|B|)/N^2)*M)/d(A, B), \tag{3}$$

where $\bar{f}(A, B)$ is the <u>mean frequency</u> at edges belonging to the cut (*A*, *B*).

It is very important that the proposed algorithm finds such a cut (*A**, *B**) whose edges have the <u>same maximal frequency</u>. This means that for any other cut (*A*, *B*)

$$\bar{f}(A, B) \le \bar{f}(A^*, B^*). \tag{4}$$

Formulae (4) and (3) together mean that the cut (*A**, *B**) maximizes (approximately, in view of the assumptions made) the expression $(((2\bullet|A|\bullet|B|)/N^2)*M)/d(A, B)$ over the set of all cuts of the considered graph. Eliminating from the latter expression the constants 2, *N* and *M*, common for all the cuts, we obtain the expression

$$D(A, B) = \frac{|A|\times|B|}{d(A,B)}. \tag{5}$$

Let us call the function *D*(*A*, *B*) the ***decomposition function*** of a graph. The above deliberations suggest the following plausible conclusion: the cut (*A**, *B**) found by the algorithm approximately maximizes the decomposition function (5) of the considered graph. The fact that in some cases this cut depends upon the initialization of a random generator (for which reason alone it cannot strictly maximize function (5) defined only by the graph itself) expresses exactly the approximate character of the solution of this optimization problem. Relevant examples are given in the next Section 2.3.

The same optimization problem (named ***RatioCut Problem***) is considered in Luxburg, 2007, where its connection with spectral classification methods is demonstrated. Yet the essential issue concerning this *NP*-hard optimization problem does not consist in the search of its approximate solutions but, rather, in the elucidation of adequacy of this function to neighborhood graph decomposition, or, to be specific, does its maximization by the suggested algorithm allow finding intuitively correct classifications? Clearly, this question is informal and the answer can only be received through experiments.

2.3. Examples of Dichotomies Constructed by the Suggested Minimax Algorithm. Dichotomies obtained by the algorithm for all the six 2-dimensional sets shown in Fig.1, are presented in Fig.3. In this and

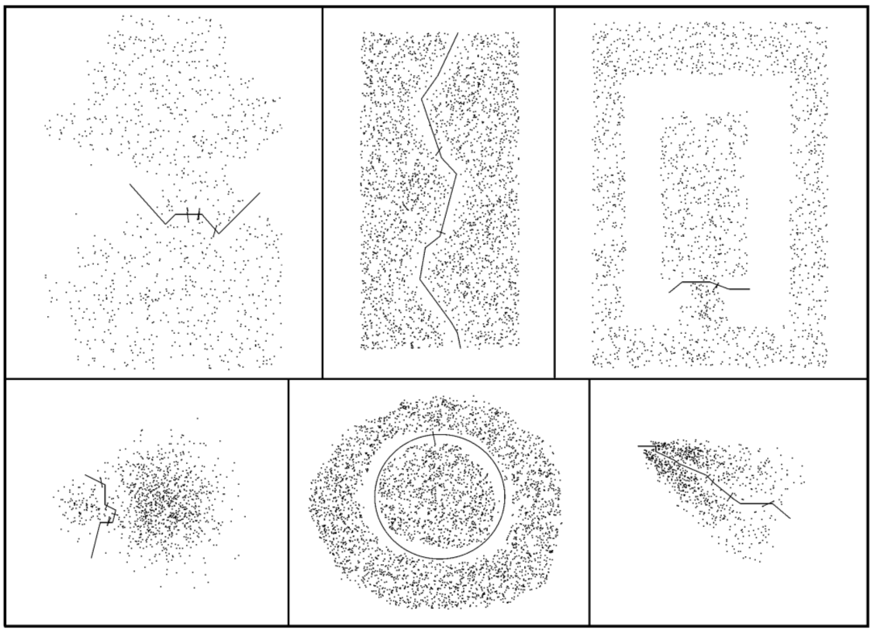


Fig.3. Solutions of six simple two-dimensional classification problems

all the subsequent figures that present classification results, only edges connecting different classes are shown; lines intersecting these edges separate the found classes. In all six cases, not only the same program was used but the few variable parameters remained the same: f = 10, T = 1000, in the construction of the neighborhood graph every vertex was connected to four closest vertices. The results do not depend on the initial seeds of the random generator. In no cases do they contradict the intuitive idea of the correctness of classification.

However this is not always the case, which set off the elaboration of the general AC algorithm described further in this work. In this algorithm, the suggested method of dichotomy is used as an essential step at the divisive stage (see Section 3). To understand the necessity of a deeper analysis, consider the following example.

Example 1. Two two-dimensial sets are shown in Fig.4a and 4c. The dichotomy result for the set of Fig.4a is shown in Fig.4b. Similarly to all six cases shown in Fig.1 and 3, the result does not depend on the initialization of the random generator. The cut, found by the minimax algorithm, maximizes the decomposition function (5) over the set of all cuts of the neighborhood graph and determines an intuitively correct classification into two classes.

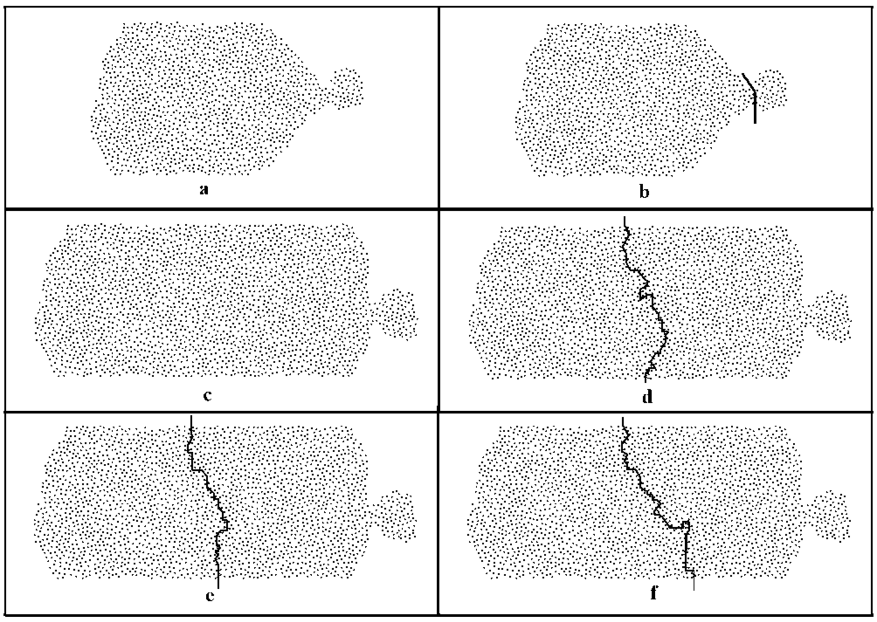


Fig.4. Stable and instable dichotomies

In contrast to the above, using the same algorithm for a similar set shown in Fig.4c, leads to results, perceptibly depending on the initialization of the random generator, as is clear from Fig.4d, 4e, and 4f. In these cases the found solutions do not coincide with the one intuitively obvious. Finally, the value of the decomposition function for the correct cut is equal to 31549, whereas for the incorrect cut found by the minimax algorithm and shown in Fig.4d it is equal to 40382. In two other cases this function is also substantially greater than its value on the correct cut. This simple example once again emphasizes the fact that we have to be cautious when contemplating sufficiently popular balanced criteria of classification (as well as other formal models of classification that are applied without good reason and with no clear identification of the type of AC problems for which the model is adequate).

The reason why criteria (5) fails in the case considered is clear enough. The quotient of the maximal and the minimal numbers of points belonging to correct classes in the set of Fig.4c is substantially greater than in the set in Fig.4a and in all sets in Fig.1. Therefore the numerator $|A|\times|B|$ in (5) is so small relative to the cardinality of product of approximately equal parts that it cannot be compensated by the denominator in (5), which is equal to the relatively small number of edges in the correct cut. The same phenomenon is true of other frequency algorithms of dichotomies (and even to a greater extent because it is manifested at a smaller ratio of cardinalities).

In order to retain the strong properties of the suggested method of dichotomy and to avoid its weakness, it is natural to consider consecutive dichotomies instead of one. For instance, using the same algorithm for the maximal of the two classes shown in Fig.4d (as far as the number of points is concerned), yields a division into three classes shown in Fig.5. If we now pool the two largest classes, we will obtain precisely the correct classification. In the next Section 3 the essential procedure of the suggested general AC algorithm is described; this procedure allows finding a family of classifications, including the correct one, by consecutive execution of (1) the divisive and (2) the agglomerative stages.

3. Four-stages Divisive-Agglomerative Procedure

This section describes the central part of the proposed classification algorithm, which we will refer to as divisive-agglomerative procedure (DAP). The procedure, whose flowchart is shown in Fig. 6, consists of four stages. The input is a neighborhood graph, constructed according to the initial data. The parameters of the procedure are the parameters of dichotomy (listed immediately before the minimax algorithm in Section 2.2) and an additional parameter k introduced below,

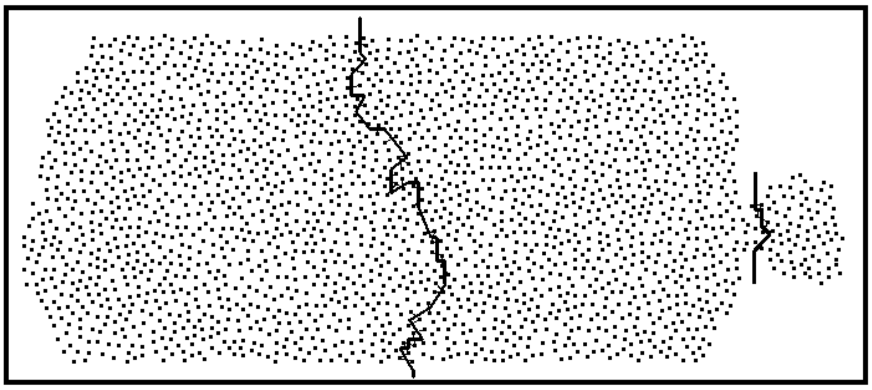


Fig.5. Result of two consecutive dichotomies

which is the only one that is related to DAP itself. The output is a family of classifications of the initial set.

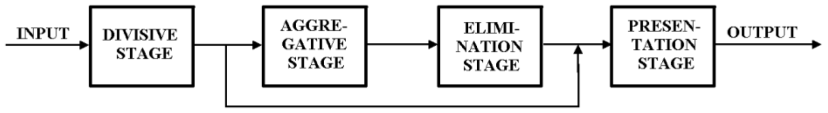


Fig.6. Flow-chart of the divisive-agglomerative procedure

Before we describe the stages of DAP, we will consider an auxiliary modification algorithm of for the matrix of class connections in the case of uniting two classes. This algorithm is used as a separate repeated step in the divisive and agglomerative algorithms presented below.

Algorithm of connection matrix modification. The input of the algorithm consists of a symmetric $m \times m$ matrix D as well as indices i and j ($i < j$). The output of the algorithm is a similar matrix D' for $(m-1)$ classes, where the "new" i-th class is the union of the "old" i-th class and the j-th class. It is assumed that the value of connection between any one of the remaining classes and the new united class is equal to the sum of values of connection between this particular class and the i-th and the j-th classes, while all the other connections remain the same. This assumption is natural in the considered situation, where the value of connection between two classes is equal to the number of edges connecting the corresponding subgraphs of the initial graph.

1. For all the elements of i-th row of matrix D let $d_{ik} = d_{ik} + d_{jk}$.

2. For all the elements of i-th column of matrix D let $d_{ki} = d_{ki} + d_{kj}$.
3. Let $d_{ii} = 0$.
4. Shift up all the rows starting with (j+1)-th row: $d_{st} = d_{s+1,t}$ ($s = j, \ldots, m-2$; $t = 0, 1, \ldots, m-1$).
5. Shift left all the columns starting with (j+1)-th column: $d_{ts} = d_{t,s+1}$ ($s = j, \ldots, m-2$; $t = 0, 1, \ldots, m-1$).
6. Delete the last row and the last column of matrix D. The obtained matrix D' is the algorithm output.

We will describe three stages of DAP separately, illustrating the description by an example.

3.1. Divisive stage. Remember that a ***divisive*** AC algorithm consists in the consecutive division of the initial set: first the whole set is divided into two parts, whereupon one of the two parts is divided into two parts once again, and so on, until some classification is obtained that seems satisfactory. It is clear that the answers to the most essential questions – how to divide a set into two parts, and which of the already constructed parts is selected for the next division – determine the essence of the algorithm being designed. These important issues are reflected in the following algorithm.

Divisive algorithm. The input of the algorithm is an undirected connected graph G. The only algorithm parameter is a counting number k, equal to the maximal number of parts in the graph division. The output of the algorithm is described below.

1. Initialization. Define the integer variable d (the number of the current dichotomy); a one-dimensional array P of length k–1 (the array of indices of subgraphs, consecutively chosen for division); an array S of length k, whose components are subgraphs. Let $d = 0$; $S[d] = G$, where G is the given input graph. Note that $P[0] = 0$ by construction.
2. From subgraphs $S[i]$ ($0 \leq i \leq d$), select a subgraph $S[im]$ with the maximal number of vertices.
3. Let $P[d] = im$.
4. Divide subgraph S [im] into two subgraphs using the algorithm of dichotomy described in Section 2.2; denote these subgraphs as S_a and S_b.
5. Let $S[im] = S_a$, $S[d+1] = S_b$.
6. Let $d = d+1$.

7. If $d < k–1$, go to step 2.
8. <u>Construction of a family of classification of set $A = \{0, 1, \dots, k–1\}$</u>.
    8.1. Define a classification C_k^{k-2} of the set $A = \{0, 1, \dots, k–1\}$ into k classes: $A_k^{k-2}[j] = \{j\}$ ($j = 0, 1, \dots, k–1$).
    8.2. For $i = k–3, \dots, 0$ recurrently define classifications C_{i+2}^{i} of the set $A = \{0, 1, \dots, k–1\}$ into (i+2) classes as follows:
    $A_{i+2}^{i}[j] = A_{i+3}^{i+1}[j]$ ($j = 0, 1, \dots, i+1; j \neq P[i]$), (6)
    $A_{i+2}^{i}[P[i]] = A_{i+3}^{i+1}[P[i]] \cup A_{i+3}^{i+1}[i+2]$. (7)
9. Define a symmetric matrix D of dimension $k \times k$ whose element d_{ij} is equal to the number of edges connecting the i-th and the j-th subgraphs from array S ($i, j = 0, 1, \dots, k–1;\ i \neq j$). For this purpose, introduce an array a of length N, where N is the number of vertices of the initial graph G. For all vertices v of the subgraph $S[i]$ let $a[v] = i$ ($i = 0, 1, \dots, k–1$). Checking consecutively all the edges of the initial graph (specified as an array of pairs of adjacent vertices), add one to the element of matrix D with indices (p, q), where $p = a[v]$, $q = a[w]$, v and w are the ends of the next edge, and $p < q$.
10. Construction of the family of matrices.
    10.1. Let $D_k^{k-2} = D$.
    10.2. For $i = k–3, \dots, 0$, recurrently define a new matrix D_{i+2}^{i} proceeding from the matrix D_{i+3}^{i+1} and indices $P[i+1]$ and (i+2) by using the above considered algorithm of modification of the connection matrix.
11. Stop.

The output of the divisive algorithm is constituted by the array S of subgraphs; the family of classifications $\{D_{i+2}^{i}\}$ ($i = k–2, \dots, 0$) of the set $A = \{0, 1, \dots, k–1\}$ into (i+2) classes; the family of matrices $\{D_{i+3}^{i+1}\}$ ($i = k–2, \dots, 0$). We will henceforth refer to these classifications and matrices as ***basic***. Subgraphs $S[0], S[1], \dots, S[k – 1]$, which are the components of array S, are henceforth referred to as ***blocks***, because all classes in all subsequently constructed classifications consist only of these components and/or their unions.

Example 2. Consider the set shown in Fig.7a. Assume $k = 4$. The consecutive dichotomies are shown in Fig.7b – 7d. In this case $P = (0, 0, 1)$. The array S consists of four blocks, shown in Fig.7d. The basic

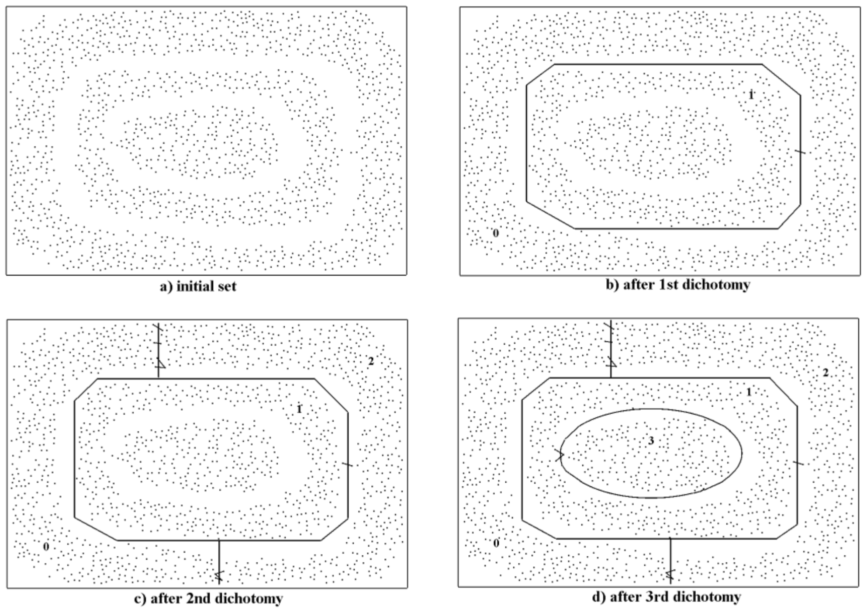


Fig.7. Consecutive dichotomies

classifications of set A = {0, 1, 2, 3}, in accordance with steps 8.1 and 8.2 of the divisive algorithm, are:

$C_2^0 = \{\{0,2\},\{1,3\}\}$, $C_3^1 = \{\{0\},\{1,3\},\{2\}\}$, $C_4^2 = \{\{0\},\{1\},\{2\},\{3\}\}$. (8)

In accordance with steps 9 and 10.1 of the divisive algorithm we obtain

$$D_4^2 = D = \begin{pmatrix} 0 & 0 & 7 & 0 \\ 0 & 0 & 1 & 2 \\ 7 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 \end{pmatrix}. \tag{9}$$

Further, for i = 1 we have $P[i+1]$ = 1, i+2 = 3 and in accordance with step10.2 of the divisive algorithm and the algorithm of connection matrix modification from D_4^2 we obtain the matrix

$$D_3^1 = \begin{pmatrix} 0 & 0 & 7 \\ 0 & 0 & 1 \\ 7 & 1 & 0 \end{pmatrix}.$$

Similarly, for i = 0 we have $P[i+1]$ = 0, i+2 = 2, and we obtain from D_3^1 the matrix $D_2^0 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. Thus, the following three basic matrices are found:

$$D_4^2 = \begin{pmatrix} 0 & 0 & 7 & 0 \\ 0 & 0 & 1 & 2 \\ 7 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 \end{pmatrix}, D_3^1 = \begin{pmatrix} 0 & 0 & 7 \\ 0 & 0 & 1 \\ 7 & 1 & 0 \end{pmatrix}, D_2^0 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \qquad (10)$$

The arrays S of four blocks, the families of classifications (8) and matrices (10) together form the output of the divisive algorithm in the considered case.

3.2. Agglomerative stage. The set A = {0, 1, …, k–1} is considered as the initial set. The element d_{ij} of the matrix D_4^2, found at the divisive stage, is considered as the value of connection between the objects i and j; the value of connection between two subsets of A is defined as the sum of numbers d_{ij} over all pairs of objects belonging to these subsets. Remember that by construction this number d_{ij} coincides with the number of edges connecting the i-th and j-th subgraphs from the array S. In the agglomerative algorithm described below, for every i varying from k–2 to 0 the basic matrix D_{i+2}^i and the classification C_{i+2}^i determine the new family of matrices and classifications, which we will refer to as **adjoint** ones.

Agglomerative algorithm. The input of the algorithm consists of: the family of basic classifications $\{C_{i+2}^i\}$ (i = k–2, …, 0) of set A = {0, 1, …, k–1} into (i+2) classes; the family of basic matrices $\{D_{i+2}^i\}$ (i = k–2, …, 0). The output of the algorithm is a family of $\frac{k(k-1)}{2}$ of the same set A = {0, 1, …, k–1}.

1. Construction of adjoint matrices and classifications. For every i = k–2, …, 0 (external cycle), the following operations are executed:

   1.1. For i+1, …, 2 (internal cycle) the next matrix D_j^i is defined by the previous matrix D_{j+1}^i as follows:

      1.1.1. Find a pair of indices s and t ($s < t$), such that the element d_{st} of the matrix D_{j+1}^i is maximal.

      1.1.2. Construct the next matrix D_j^i from the matrix D_{j+1}^i and indices s and t, using the algorithm of connection matrix modification.

      1.1.3. Define the new classification C_j^i of set A = {0, 1, …, k–1} into j classes from the classification C_{j+1}^i as follows:

      $A_j^i[p] = A_{j+1}^i[p]$ (p = 0, 1, …, t–1; $p \neq s$), (11)

$A_j^i[p] = A_{j+1}^i[p+1]$ ($p = t, \ldots, j-1$), (12)

$A_j^i[s] = A_{j+1}^i[s] \cup A_{j+1}^i[t]$. (13)

2. Presentation of the output. Present all constructed classifications in the following order, writing in one row all classifications into the same number of classes:

$C_2^0, C_2^1, \ldots, C_2^{k-3}, C_2^{k-2};$
$C_3^1, \ldots, C_3^{k-3}, C_3^{k-2};$
$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$ (14)
$C_{k-1}^{k-3}, C_{k-1}^{k-2};$
$C_k^{k-2}.$

In (14) the first classification in every row is basic; all of them are constructed from the initial classification C_2^{k-2} by formulae (6) and (7). All other classifications in (14) are the adjoint ones. The last classifications in every row (starting from the penultimate row) are adjoint to C_k^{k-2}, penultimate classifications in every row (starting from the third row from the end) are adjoint to C_{k-1}^{k-3}, and so on, up to the classification C_2^1, adjoint to C_3^1. In all cases the lower index is equal to the number of classes, while the upper index is the number of the group written diagonally; this group contains one basic classification (the first one in the diagonal), and all those adjoint to it as found by the agglomerative algorithm. The output of this algorithm consists of the family of classifications of the set $A = \{0, 1, \ldots, k-1\}$, ordered as shown in (14); the basic classifications are determined by formulae (6), (7) and the adjoint ones by formulae (11) – (13)

Example 2. Continuation. In this case three basic matrices are specified by (10) and three basic classifications are specified by (8). We will now come to construct adjoint matrices and classifications following the agglomerative algorithm. Fixing $i = 2$ we obtain for $j = 3$ (see step 1.1.1) $d_{st} = 7$, $s = 0$, $t = 2$. In accordance with step 1.1.2, the matrix is $D_3^2 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 2 \\ 0 & 2 & 0 \end{pmatrix}$, and in accordance with step 1.1.3 the classification $C_3^2 = \{\{0,2\},\{1\},\{3\}\}$. Further, at $i=2$ we obtain for $j=2$ $d_{st} = 2$, $s = 1$, $t = 2$; the matrix $D_2^2 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$; $C_2^2 = \{\{0,2\},\{1,3\}\}$. Return to the external cycle and set i to 1. For the only $j = 2$ we obtain (see matrix D_3^1) $d_{st} = 7$, s

= 0, t = 2, which gives $D_2^1 = D_2^0 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ and $C_2^1 = C_2^2 =\{\{0,2\},\{1,3\}\}$. Finally, for $i = 0$ the required j satisfying the conditions $1 \geq j \geq 2$, does not exist and no adjoint matrices or classifications are constructed.

By writing out all found classifications in accordance with step 3 of the agglomerative algorithm we obtain

$$C_2^0 = \{\{0,2\},\{1,3\}\},\ C_2^1 = \{\{0,2\},\{1,3\}\},\ C_2^2 = \{\{0,2\},\{1,3\}\},$$
$$C_3^1 = \{\{0\},\{1,3\},\{2\}\},\ \ C_3^2 = \{\{0,2\},\{1\},\{3\}\}, \tag{15}$$
$$C_4^2 = \{\{0\},\{1\},\{2\},\{3\}\}.$$

Note that the adjoint classification $C_3^2 = \{\{0,2\},\{1\},\{3\}\}$ is the only correct classification (see Fig.8). This classification cannot be constructed directly by any number of consecutive dichotomies (e.g. it is not a basic classification for any parameter k), which is one of the reason why we had to introduce adjoint classifications.

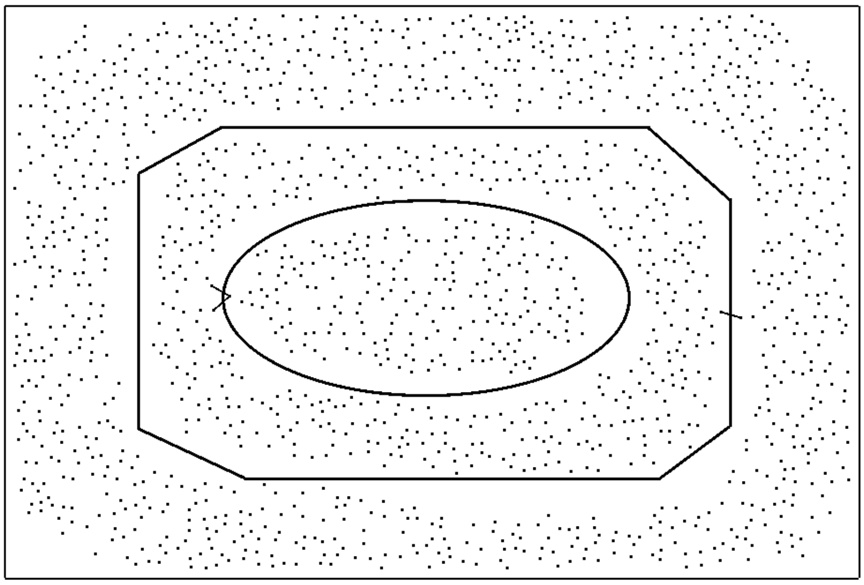


Fig.8. A correct adjoint classification C_3^2

3.3. Elimination stage. The output of the agglomeration stage is the family of classification of the set $A = \{0, 1, \ldots, k–1\}$ in form (14). Since the number of classifications in the i-th row from the bottom is equal to i, and the total number of rows is equal to k, it can be seen that the total number of constructed classifications is $\frac{k(k-1)}{2}$. Some of them may be identical. This DAP stage is designed to eliminate these (certainly redundant) classifications. The algorithm itself is almost obvious, the more so as the dimension of the problem is not large (the number k of blocks is assumed to be small enough – about 10 to 20).

Yet the comparison of two classifications of one and the same set can be easily made even for much greater k. It is sufficient to list the numbers in every class in the increasing order. The comparison of classifications is obviously reduced to the comparison of sets, whereas for ordered subsets of the same set it is reduced to consecutive comparison of elements having the same indices – up to the first discrepancy or to the end of the exhaustion. The output of this stage is a family of different classifications.

Example 2. Continuation. Of the 6 classifications (15) of the set $A = \{0, 1, 2, 3\}$, only 4 are different:

$$C_2^0 = \{\{0,2\},\{1,3\}\},$$
$$C_3^1 = \{\{0\},\{1,3\},\{2\}\},\quad C_3^2 = \{\{0,2\},\{1\},\{3\}\}, \tag{16}$$
$$C_4^2 = \{\{0\},\{1\},\{2\},\{3\}\}.$$

These four classifications are shown in Fig.9.

3.4. Presentation stage. The output of the elimination stage is the family (to be further denoted as F) of different classifications of the set $A = \{0, 1, \dots, k–1\}$. Yet the output of the whole DAP is, as was pointed out at the beginning of the Section, the family of classifications of the initial set of objects (e.g. vertices of the given graph G), rather than the set of indices of blocks $A = \{0, 1, \dots, k–1\}$. Transition from one family to the other is the goal of this stage.

Of course, every block (e.g. a subgraph of the initial graph) is uniquely defined by its index (see the description of the output of the divisive algorithm). Therefore every classification of block indices uniquely defines some classification of the initial set. However, considering the subsequent operations of the suggested general AC algorithm, it seems expedient at this stage to present every classification into t classes as a set of t binary vectors ($t = 2, \dots, k$) of length N, where N is the number of vertices of graph G.

Presentation algorithm. The input of the algorithm is the array S of blocks of length k, constructed at the divisive stage (see the description

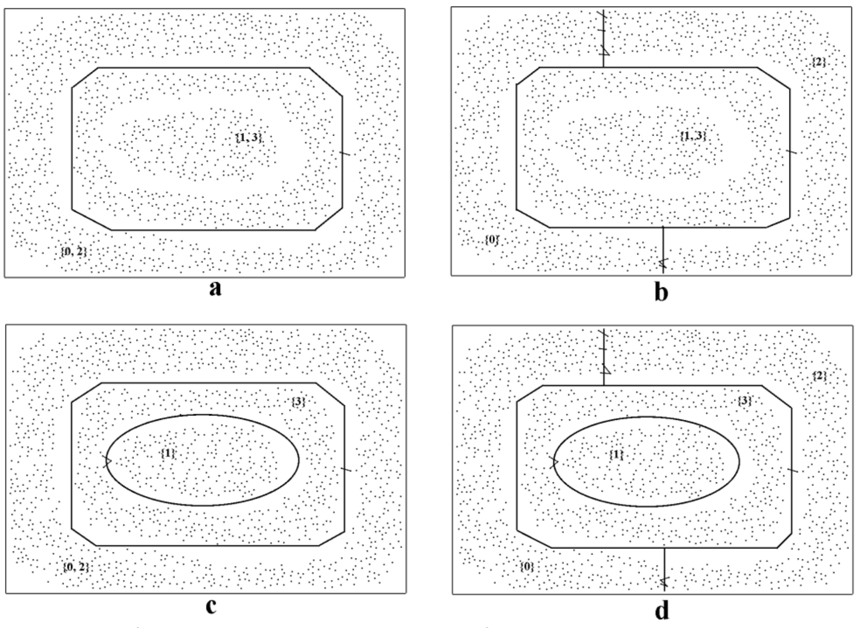


Fig. 9. Classification a into two classes; classifications b and c into three classes; classification d into four classes

in Section 3.1), and a family F of classifications of the set $A = \{0, 1, \ldots, k–1\}$, whose elements correspond to the indices of subgraphs (or blocks) in the array S. The output of the algorithm is the family R of classifications of set of vertices of G, denoted below as V.

```
1. Let $R = Ø$.
2. For every classification $C \in F$ execute the following operations (the
external cycle by the given classifications of $A = \{0, 1, \ldots, k–1\}$):
    2.1. Let $B = Ø$ ($B$ denotes the set of binary vectors which defines
    the classification of set $V$ corresponding to $C$).
    2.2. For every class $X$ from classification $C$ execute the following
    operations (cycle by classes from the fixed classification $C$):
        2.2.1. Define a vector $x$ with $N$ components and let $x_i = 0$ ($i = 0,
        1, \ldots, N – 1$).
        2.2.2. For every index $p \in X$ execute the following operations
        (cycle by indices from class $X$):
```

2.2.2.1. Consider the set of vertices of subgraph $S[p]$ (it is given as an integer array z of length n_p, whose components are indices of vertices belonging to subgraph $S[p]$).

2.2.2.2. For every component y of array z let $x[y] = 1$ (internal cycle by vertices from one block).

2.2.3. Add the vector x to the set B: $B = B\cup\{x\}$.

2.3. Add the set of vectors B to the set R: $R = R\cup B$.

Thus, the output of the presentation algorithm and, hence, the output of the whole DAP, is the family of classifications of the set of vertices of graph G; every one of which is presented by a set of binary vectors – one vector for each class..

To conclude this Section 3.4, we will give some comments. It may seem possible to significantly simplify the suggested DAP. Specifically, choose a sufficiently large value of the parameter k, make $(k–1)$ consecutive dichotomies using the divisive algorithm and thus find one basic classification C_k^{k-2} (and the corresponding basic matrix $D_k^{k-2} = D$). After that, find all classifications adjoint to C_k^{k-2}, one of which is correct. Indeed, this is how conventional agglomerative algorithms work, starting with the initial classification, all classes of which consist of one object. In many cases this simpler procedure also proves successful in the framework of the suggested approach. However it is not always the case.

Example 3. Consider the set shown in Fig.10a (two two-dimensional normal distributions consisting of 200 and 1000 points). The basic classification C_3^1, i.e. at $k = 3$, for some initializations of the random generator is shown in Fig.10b. The only corresponding adjoint classification C_2^1, shown in Fig.10c, is not correct. The correct classification in this case is the basic classification C_2^0, shown in Fig.10d. This simple example demonstrates that we need to analyze all basic classifications. It is not a problem, however because computation time needed for all operations in the agglomerative algorithm is negligibly small as compared to the time required by $(k – 1)$ dichotomies in the divisive algorithm.

There is nothing drastic in this example. Generally, for sufficiently large k values, the classes are getting small, so that the number of edges connecting some classes with the remaining ones may be smaller than in the correct cut. Since in the aggregation the sets to be united are connected by the maximal number of edges, the correct classification does not coincide with any classification, adjoint to the basic classification C_k^{k-2} for large k. In some cases (in particular, shown in example 3) this phenomenon can even happen for sufficiently small k. In Fig.10b the wrong cut consists of 6 edges, while the correct cut has 7 edges. Therefore in the construction of an adjoint classification the sets selected for the union are wrong, which is demonstrated by Fig.10c.

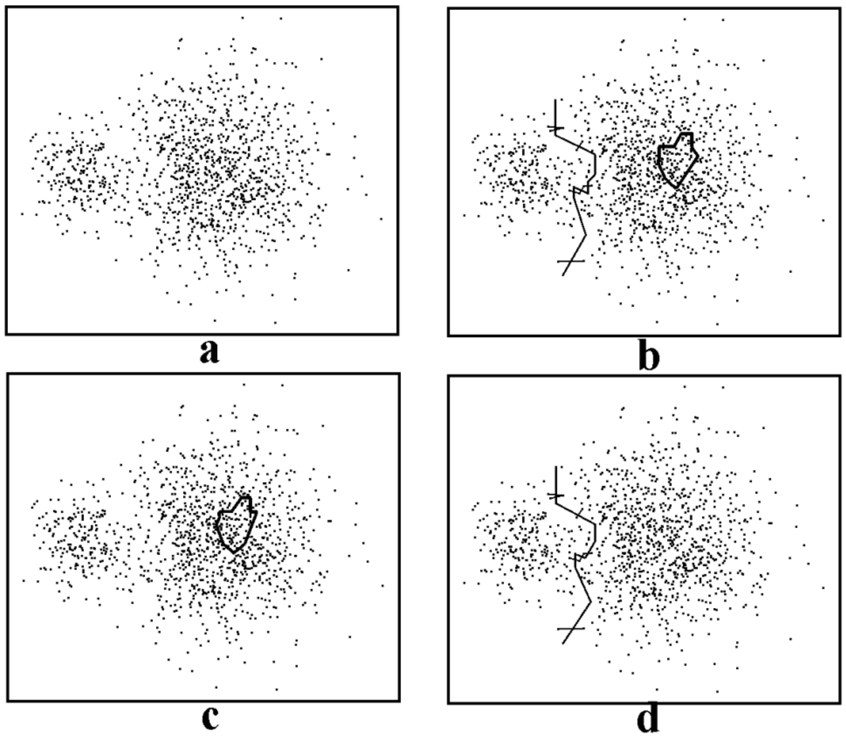


Fig.10. Example of a missed basic classification

Numerous examples demonstrate that correct classifications are indeed present among the few classifications found by the suggested DAP. The method of their selection is considered in the next Section 4.

4. Stable Classifications

Since the suggested algorithm of dichotomy contains randomized steps (including the determination of the initial frequencies and the choice of every pair of vertices), the result of dichotomy is, generally speaking, random. Examples considered in Section 2 point to the fact that correctness and stochastic stability are closely connected. An important fact could be experimentally established: in all cases when the result of dichotomy does not depend on the initialization of a random generator (which **is** stochastic stability), the constructed classification into two classes is correct, e.g. it does not contradict geometric intuition. Conversely, in all cases (reported in this work as well as in dozens of others) the absence of stability implies incorrectness of the constructed classification. It is exactly this deliberation that underlies the proposed algorithm of selection of correct classifications from among all found by the suggested DAP.

Since the result of each dichotomy is random, all classifications constructed by DAP and even their number are random. We propose to repeat this procedure several times for different initializations of the random generator. It turned out that only very few of the classifications found belong to all of the constructed families, whereas the most of classifications do not belong to at least some of them and, hence, can be considered as random, instable and inessential. Classifications entering into all the families will be referred to as ***stable classifications***. A ***correct classification*** is defined as a stable classification with the maximal number of classes (of course, this number cannot exceed a preset number *k* of blocks, which is the only parameter of DAP). Note that it is possible to choose *k* with a safety margin, essentially greater than the assumed maximally possible number of classes.

The idea of basing the choice of classification on stochastic stability is not new. Yet the efficiency of this idea depends on how natural and strong the connection is between quality of classification and its stability. The experiments demonstrate that, within the suggested approach, this connection is practically decisive. The matter is that stochastic stability is analyzed for very few classifications, already constructed on the basis of the reasonable dichotomies, and the considered random families already contain correct classifications. Therefore none of the experiments yield partial coincidences (50-90%) for correct classifications. Either the coincidence is practically full (with a small number of declining points), or it does not reach 90%. This high stability (backed by intuitive obviousness of the results) is the strongest argument in favor of the suggested approach. In the case of absence of stable classifications it may be reasonably inferred that the initial set does has no cluster structure.

Before we give a formal description of the selection algorithm of stable classifications we will introduce necessary formal notions and describe auxiliary algorithms. Two finite sets A and B coincide at level α ($0 \leq \alpha \leq 1$), if

$$|A \cap B| / |A \cup B| \geq \alpha \qquad (17)$$

($|X|$ denotes the cardinality of the finite set X). Two classifications P and Q of the same finite set M into the same number of classes m coincide at level α if it is possible to enumerate their classes $P_1, P_2, \ldots, P_m$ and $Q_1, Q_2, \ldots, Q_m$ so that sets P_i and Q_i coincide at level α ($i = 1, 2, \ldots, m$). The following almost evident statement takes place.

Statement 2. Let two classifications P and Q coincide at level α, and $\alpha > 0.5$. Then the one-to-one correspondence between their classes mentioned in the previous definition is determined uniquely.

Proof. It is sufficient to prove that any class from P may coincide at level α only with one class from Q.
Assume that it is wrong and, hence, a class A from P coincides at level α with two different classes B и C from Q. Let $A' = A \cap B$, $A'' = A \cap C$. Since $B \cap C = \emptyset$, $A' \cap A'' = \emptyset$. Two conditions of coincidence (for B and C) can be written as:
$|A'| \geq (|A''| + |B|)\alpha$,
$|A''| \geq (|A'| + |C|)\alpha$.
Summing up these two inequalities, after simple rearrangements we have

$$(|A'|+|A''|)\beta \geq (|B| +|C|)\alpha, \qquad (18)$$

where $\beta = 1-\alpha$.
At the same time by the definition of A' and A''
$|A'| \leq |B|$, $|A''| \leq |C|$, hence

$$(|A'|+|A''|) \leq (|B| +|C|). \qquad (19)$$

Since by condition $\alpha > 0.5$, it implies that $\alpha > \beta$. Therefore (18) implies
$(|A'|+|A''|)\alpha > (|B| +|C|)\alpha$,
hence $(|A'|+|A''|) > (|B| +|C|)$, which contradicts (19). The contradiction completes the proof ■

Algorithm of checking the coincidence at level α of two subsets of the same set U. The algorithm is as follows. In our case every set is presented by a binary vector with $m = |U|$ components (see the presentation algorithm in Section 3.4).

```
1. By one simultaneous survey of both vectors calculate:
    1.1. The number of components with two 1s.
    1.2. The number of components with at least one 1.
2. Divide the first number by the second one.
3. Compare the result with the given number $\alpha$. If it exceeds $\alpha$, then
   two sets coincide at level $\alpha$; otherwise, they do not coincide.
```

Algorithm of checking the coincidence at level α of two classifications of the same set into *m* classes. It is assumed that $\alpha > 0{,}5$.

```
1. Let $i = 0$.
2. Let $i = i + 1$.
3. If $i > m$, then the classifications coincide. Stop the algorithm.
4. Let $j = 0$.
5. Let $j = j + 1$.
6. If $j > m$, then the classifications do not coincide. Stop the algorithm.
7. Check the coincidence at level $\alpha$ of sets $P_i$ and $Q_j$. In the case of
   coincidence go to step 2, otherwise go to step 5.
```

Statement 2 guarantees the correctness of the algorithm. The matter is that no analysis of different one-to-one correspondences between the classes is required. It is sufficient to find, for every class from the 1st classification, a class from the 2nd classification that coincides with it.

Algorithm of selection of stable classifications. This algorithm is the external cycle in the considered general AC algorithm. The input of the algorithm is the given graph *G*; the output is a set (it may be an empty set or a set consisting of more than one element) of correct classifications in the AC problem, presented by the graph *G*. Parameters of the algorithms are two numbers *r* and α: *r* is equal to the number of repetitions of DAP; α sets a minimal level of coincidence of classifications for which the conclusion is made with regard to stochastic stability (the typical value of α lies in segment [0.90 – 0.95]). The remaining parameters have been described above. Classifications found by the algorithm will be called (r, α)-stable.

```
1. Initialization. Execute DAP once. Let $F = C$, where $C$ is the set of
   classifications found by DAP; $F = \{F_1, \dots, F_n\}$. Let $t = 0$.
2. Let $t = t + 1$. If $t = r$, go to 4.
3. Execute DAP once and denote the set of found classifications by $C$
   $= \{C_1, \dots, C_{m_t}\}$.
    3.1. Let $i = 0$.
    3.2. Let $i = i + 1$.
    3.3. If $i > n$, then go to 2
    3.4. Let $j = 0$.
    3.5. Let $j = j + 1$.
    3.6. If $j \le m_t$, then go to 3.9.
    3.7. Delete the classification $F_i$ from the list $F = \{F_1, \ldots, F_n\}$;
    reduce the indices of all classifications from $(i + 1)$ by 1 and let $n = n–1$.
    3.8. If $n = 0$, then $(r, \alpha)$-stable classifications do not exist. Stop.
    Otherwise, go to step 3.2.
    3.9. Check the coincidence at level $\alpha$ of sets $F_i$ and $C_j$. (using the
    above described algorithm). In the case of coincidence go to 3.2;
    otherwise go to 3.5.
4. The current set $F = \{F_1, \ldots, F_n\}$ is the set of all $(r, \alpha)$-stable
classifications. Stop.
```

Some comments to the last algorithm are needed. If classification F_i does not coincide with any of new classifications C_j ($j = 1, \ldots, m_t$), this means that F_i is not a stable one. Therefore at step 3.7 this classification is deleted from the list F. If at this step all classifications from F are deleted, then the algorithm stops at step 3.8. But if for classification F_i a classification C_j coinciding with it has been found (which is established at step 3.9), then F_i is still considered a stable one, and after returning to step 3.2 the next classification F_{i+1} is checked. The return to step 2 after step 3.2 means the transition to the next iteration in the external cycle over new executions of DAP.

Example 2. Continuation. A single execution of DAP yields 4 classifications shown in Fig.9. From the viewpoint of stability, the most crucial one is the classification into 3 classes shown in Fig.9b, which is the basic classification $C_3^1 = \{\{0\},\{1,3\},\{2\}\}$. Let the number r of repetitions of DAP be equal to 4. In four executions of DAP the classifications, corresponding to C_3^1, are shown in Fig.11a – 11d.

The classifications in Fig.11a and 11b coincide at level $\alpha \approx 0{,}85$ (class {2} consists of 483 points in the first classification and of 410 points in the second one. It is clear that at the level 0.9 or higher these classifications are not stable. The same is also true of the classification into 4 classes corresponding to the basic classification $C_4^2 = \{\{0\},\{1\},\{2\},\{3\}\}$ (see Fig.9d). Yet the classifications into 2 and 3 classes shown in Fig.9a and 9c are stable at level 1. Therefore the classification from Fig.9c is taken as a formal solution of AC problem in the considered case, because it is a stable classification with the maximal number of classes (under the restriction k = 4). Note that the algorithm finds the same single classification stable at level 1 under any k > 4, too.

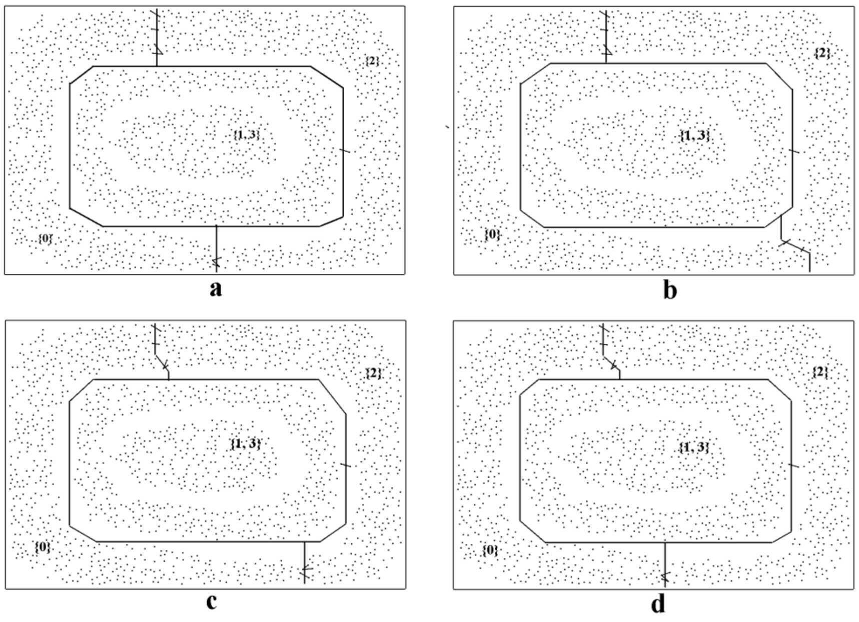


Fig.11.Example of instability of classification

Stability parameters r and α are connected in the same way as the number of random trials and significance level are connected in statistics: the larger values of r correspond to the lower values of coincidence level α. Yet in the considered situations the coincidence level α decreases with the growth of r for unstable classifications. For stable classifications the coincidence level soon approaches a number sufficiently close to 1, and does not vary further with the growth of r. This allows us to assume that the number r of repetitions stays within the limit 5 – 15. Intermediate situations, when stability is lost after 20 – 30 repetitions, may happen (though for very few initial sets and rare initializations of the random generator). Natural modifications of the notion of stability that span these very rare situations are disregarded here for the sake of brevity.

5. Examples

In this section several examples are considered in the increasing order of complexity (the notion of complexity is informal). In the first six examples 4 – 9, the initial sets are two-dimensional sets of points. The figures show only the final results of the suggested general AC algorithm. Parameters α, *f* and *T* were the same in all considered cases: α = 0.95; *f* = 10; *T* = 1000. In examples 4 – 6, 10, 11 *r* = 10 and *k* = 6; in example 7 *r* = 5 and *k* = 10; in example 8 *r* = 10 and *k* = 15.

Example 4. The initial set is the union of two two-dimensional sets of points, each normally distributed along the *x* axis and uniformly along the *y* axis. The result of classification is shown in Fig.12. Note that the number of classes is not defined in advance – there is only the upper bound *k*.

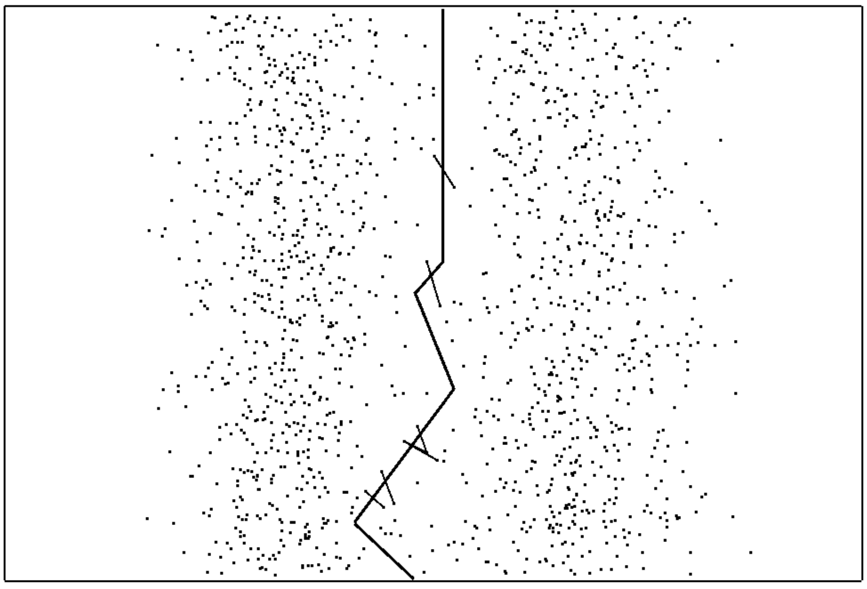


Fig.12. Classification result in example 4 (1397 points)

Example 5. The initial set consists of two contacting rings with a few points inside. The result of classification is shown in Fig.13.

Example 6. The initial set consists of four two-dimensional slightly damaged normal distributions. The result of classification is shown in

Fig.14.

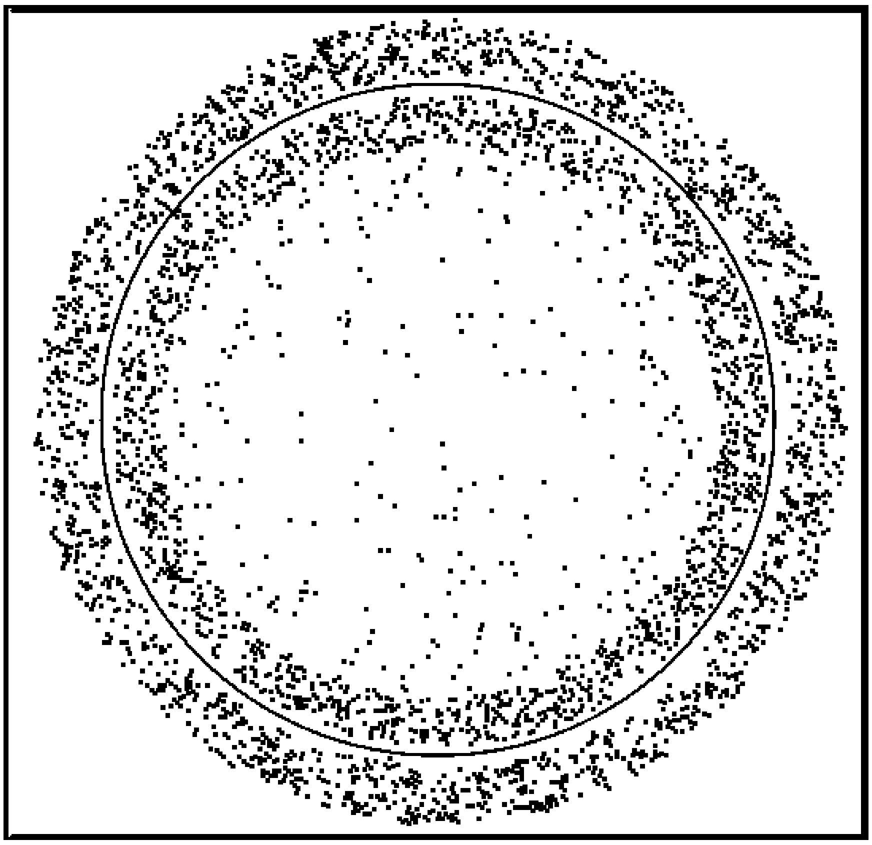


Fig.13. Classification result in example 5 (3228 points)

Example 7. The initial set resembles a spiral with two narrow isthmuses approximately in the middle and close to one of the ends. The result of classification is shown in Fig.15.

Example 8. The initial set resembles a lake with two peninsulas. The smaller peninsula contains relatively few points (223 of 4501), while the number of edges, connecting it to the other part is equal to 6, i.e. it is not too small. Therefore this peninsula is revealed only after a considerable number of consecutive dichotomies and therefore *k* was taken to be large enough (*k*=15). The result of classification is shown in Fig.16.

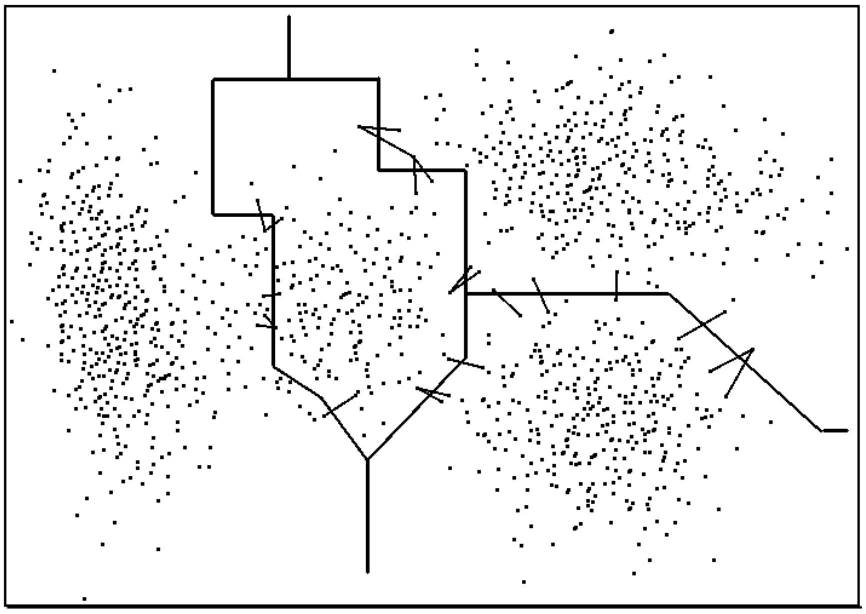


Fig.14. Classification result in example 6 (1279 points)

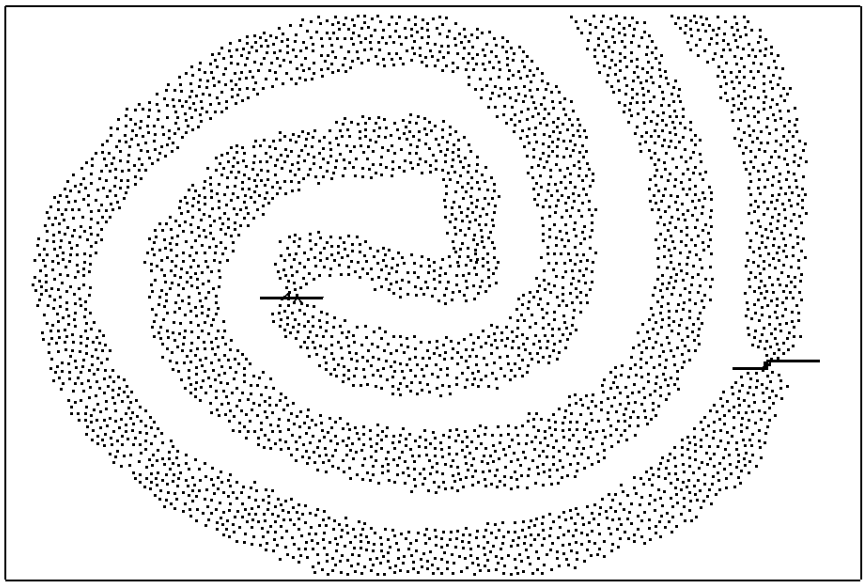


Fig.15. Classification result in example 7 (4272 points)

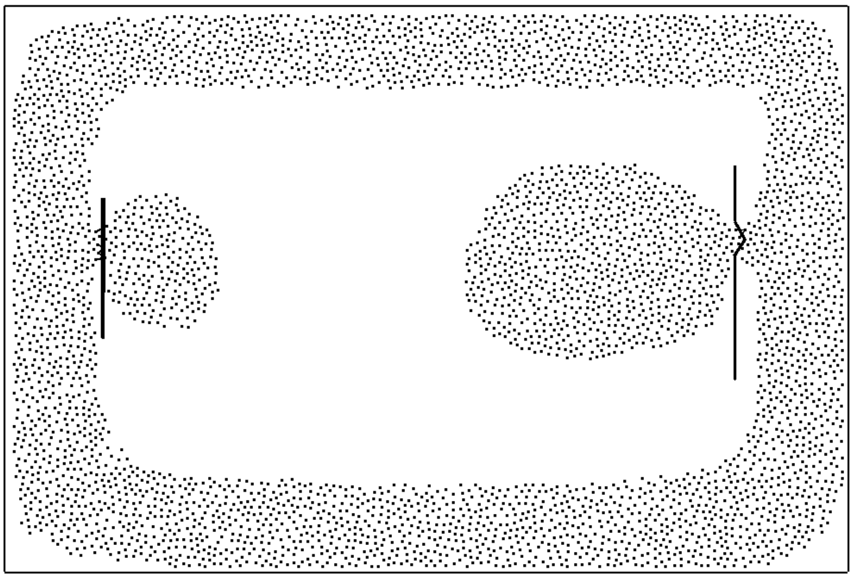


Fig.16. Classification result in example 8 (4501 points)

Example 9. The initial set is shown in Fig.17. No cluster structure is observed. The suggested algorithm confirms the fact: stable classifications are lacking even at level 0.6 for the number of repetitions r = 4.

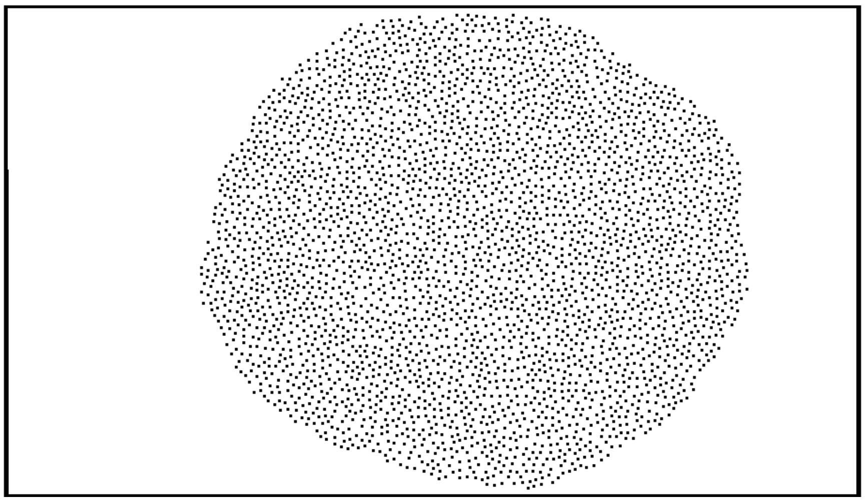


Fig.17. Example of absence of any reasonable classification

Example 10. Multidimensional data: a matrix of objects and parameters matrix. Unlike all previous examples, in this case the initial set is not two-dimensional. There are 15 objects and 20 parameters. Three classes are selected. The first class consists of objects with numbers 0, 3, 6, 7, 10, 13; the second class consists of objects with numbers 1, 4, 5, 8, 14; the third class consists of objects with numbers 2, 9, 11, 12. Two groups of parameters are selected. The first group consists of parameters 0, 2, 3, 5, 8, 9, 11, 14, 16, 17, 19; the second group consists of parameters 1, 4, 6, 7, 10, 12, 13, 15, 18.

The objects from the first class have "large" values (between 44 and 90) for the parameters of the first group and "small" values (between 10 and 56) for the parameters of the second group. The objects from the second class have "large" values (between 44 and 90) for the parameters of the second group and "small" values (between 10 and 56) for the parameters of the first group. The objects from the third class have "intermediate" values (between 27 and 73) for the parameters of both groups. All numbers in the pattern matrix are randomly selected within the corresponding ranges (large, small, intermediate):

The suggested general algorithm correctly finds the given classification into 3 classes. The parameters from the second group are marked grey; matrix rows are rearranged in such a way that the objects from the same class are written consecutively. Thus, in this example the correct classification is revealed despite significant intersections of large, small, and intermediate ranges:

```
      0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
  ______________________________________________________________________________
  0 | 69  16  56  76  27  45  54  52  50  82  39  56  45  21  51  50  80  88  55  51
  3 | 72  21  46  69  33  49  49  12  72  50  24  54  47  38  58  23  63  65  54  50
  6 | 71  31  86  71  47  74  16  48  76  80  19  62  50  26  86  20  83  54  43  55
  7 | 70  15  64  69  11  50  28  42  44  68  17  56  13  46  86  21  62  80  31  45
 10 | 49  12  83  65  14  55  44  10  76  79  12  47  36  28  71  51  54  84  24  71
 13 | 54  24  78  54  12  70  48  39  86  46  19  55  49  24  89  53  49  54  17  65
---------------------------------------------------------------------------------
  1 | 34  57  47  32  74  40  58  59  48  28  47  21  70  79  28  87  28  39  67  17
  4 | 35  50  47  27  84  15  59  72  40  12  57  36  85  88  21  46  53  26  72  15
  5 | 33  53  43  25  46  15  77  57  19  54  79  19  56  52  52  57  50  48  71  31
  8 | 39  71  48  10  84  55  65  77  35  12  67  29  57  74  48  79  17  24  64  31
 14 | 53  81  55  44  73  14  66  49  44  16  77  26  82  69  13  63  50  47  47  33
---------------------------------------------------------------------------------
  2 | 48  55  43  52  71  56  44  43  72  28  37  42  30  29  47  48  40  60  39  63
  9 | 35  68  42  37  63  33  51  38  42  53  58  72  64  64  35  66  72  52  30  34
 11 | 46  46  59  29  67  57  29  46  27  58  62  34  55  47  60  28  70  37  54  45
 12 | 29  58  38  45  55  44  34  27  50  47  30  70  43  55  29  44  66  71  64  62
```

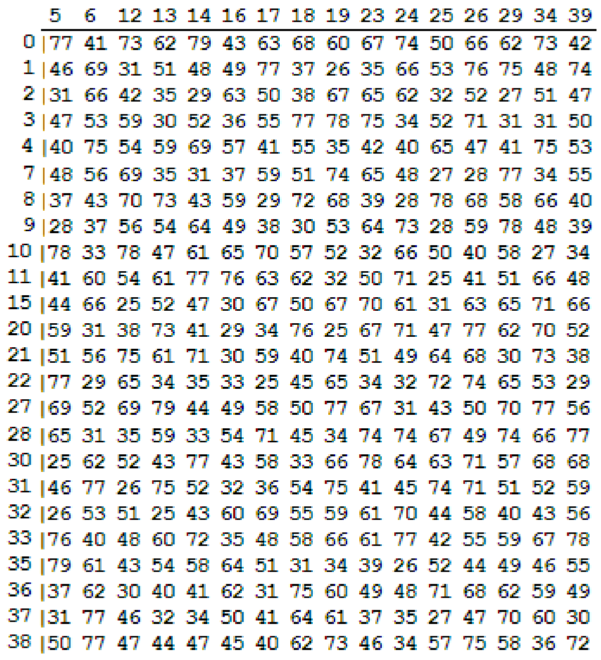
Example 11. Dissimilarity matrix. There are 40 objects. Two classes are selected. The first class consists of objects with numbers 0, 1, 2, 3, 4, 7, 8, 9, 10, 11, 15, 20, 21, 22, 27, 28, 30, 31, 32, 33, 35, 36, 37, 38; the second class consists of objects with numbers 5, 6, 12, 13, 14, 16, 17, 18, 19, 23, 24, 25, 26, 29, 34, 39. Dissimilarities between objects belonging to the same classes are random numbers from 10 to 60, while dissimilarities between objects belonging to different classes are random numbers from 25 to 80.

The suggested general algorithm finds correctly the initial classification into 2 classes. Matrix rows and columns are rearranged in such a way that the objects from the same class are written together. The data is presented by 4 matrices. The 1-st and the 4-th ones contain dissimilarities between objects from the 1-st and 2-nd class, respectively: there are no numbers larger than 60. The 2-nd and the 3-rd ones contain dissimilarities between objects from different classes: there are no number lesser than 25.

Matrix 1

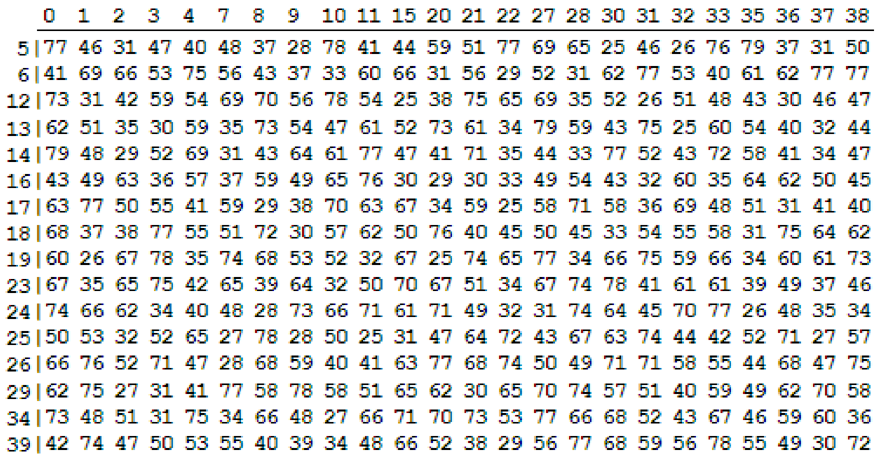
```
     0  1  2  3  4  7  8  9 10 11 15 20 21 22 27 28 30 31 32 33 35 36 37 38
   ------------------------------------------------------------------------
 0 |-- 52 51 35 23 25 55 21 49 20 37 13 37 53 22 44 21 28 48 55 51 16 48 28
 1 |52 -- 49 45 43 49 19 39 42 14 41 21 60 52 48 24 39 13 13 58 21 35 30 60
 2 |51 49 -- 47 11 57 48 11 34 36 60 59 28 57 44 37 29 46 27 30 45 12 37 49
 3 |35 45 47 -- 11 13 15 28 21 30 35 27 36 26 39 38 32 35 37 25 48 15 31 46
 4 |23 43 11 11 -- 17 46 24 31 21 28 56 52 25 22 60 33 55 58 15 36 41 47 45
 7 |25 49 57 13 17 -- 53 19 54 59 30 60 14 58 50 54 57 16 28 36 52 60 28 28
 8 |55 19 48 15 46 53 -- 56 14 56 47 34 56 24 46 24 25 27 12 11 19 20 56 33
 9 |21 39 11 28 24 19 56 -- 58 30 38 37 28 52 44 37 26 40 27 31 42 55 17 54
10 |49 42 34 21 31 54 14 58 -- 57 49 50 39 33 38 46 21 36 24 55 40 60 19 56
11 |20 14 36 30 21 59 56 30 57 -- 28 28 60 48 44 31 36 50 53 29 32 50 44 25
15 |37 41 60 35 28 30 47 38 49 28 -- 42 32 20 19 28 30 41 56 58 20 23 57 42
20 |13 21 59 27 56 60 34 37 50 28 42 -- 19 57 14 40 36 20 50 18 27 20 53 33
21 |37 60 28 36 52 14 56 28 39 60 32 19 -- 42 14 50 50 35 54 33 33 39 11 54
22 |53 52 57 26 25 58 24 52 33 48 20 57 42 -- 27 23 40 31 19 60 32 16 28 54
27 |22 48 44 39 22 50 46 44 38 44 19 14 14 27 -- 60 52 17 17 19 35 16 24 27
28 |44 24 37 38 60 54 24 37 46 31 28 40 50 23 60 -- 13 34 17 41 19 13 25 29
30 |21 39 29 32 33 57 25 26 21 36 30 36 50 40 52 13 -- 43 16 17 50 25 35 43
31 |28 13 46 35 55 16 27 40 36 50 41 20 35 31 17 34 43 -- 31 17 28 40 45 18
32 |48 13 27 37 58 28 12 27 24 53 56 50 54 19 17 17 16 31 -- 45 17 23 18 21
33 |55 58 30 25 15 36 11 31 55 29 58 18 33 60 19 41 17 17 45 -- 57 49 19 24
35 |51 21 45 48 36 52 19 42 40 32 20 27 33 32 35 19 50 28 17 57 -- 15 51 18
36 |16 35 12 15 41 60 20 55 60 50 23 20 39 16 16 13 25 40 23 49 15 -- 55 23
37 |48 30 37 31 47 28 56 17 19 44 57 53 11 28 24 25 35 45 18 19 51 55 -- 32
38 |28 60 49 46 45 28 33 54 56 25 42 33 54 54 27 29 43 18 21 24 18 23 32 --
```

Matrix 2



	5	6	12	13	14	16	17	18	19	23	24	25	26	29	34	39
0	77	41	73	62	79	43	63	68	60	67	74	50	66	62	73	42
1	46	69	31	51	48	49	77	37	26	35	66	53	76	75	48	74
2	31	66	42	35	29	63	50	38	67	65	62	32	52	27	51	47
3	47	53	59	30	52	36	55	77	78	75	34	52	71	31	31	50
4	40	75	54	59	69	57	41	55	35	42	40	65	47	41	75	53
7	48	56	69	35	31	37	59	51	74	65	48	27	28	77	34	55
8	37	43	70	73	43	59	29	72	68	39	28	78	68	58	66	40
9	28	37	56	54	64	49	38	30	53	64	73	28	59	78	48	39
10	78	33	78	47	61	65	70	57	52	32	66	50	40	58	27	34
11	41	60	54	61	77	76	63	62	32	50	71	25	41	51	66	48
15	44	66	25	52	47	30	67	50	67	70	61	31	63	65	71	66
20	59	31	38	73	41	29	34	76	25	67	71	47	77	62	70	52
21	51	56	75	61	71	30	59	40	74	51	49	64	68	30	73	38
22	77	29	65	34	35	33	25	45	65	34	32	72	74	65	53	29
27	69	52	69	79	44	49	58	50	77	67	31	43	50	70	77	56
28	65	31	35	59	33	54	71	45	34	74	74	67	49	74	66	77
30	25	62	52	43	77	43	58	33	66	78	64	63	71	57	68	68
31	46	77	26	75	52	32	36	54	75	41	45	74	71	51	52	59
32	26	53	51	25	43	60	69	55	59	61	70	44	58	40	43	56
33	76	40	48	60	72	35	48	58	66	61	77	42	55	59	67	78
35	79	61	43	54	58	64	51	31	34	39	26	52	44	49	46	55
36	37	62	30	40	41	62	31	75	60	49	48	71	68	62	59	49
37	31	77	46	32	34	50	41	64	61	37	35	27	47	70	60	30
38	50	77	47	44	47	45	40	62	73	46	34	57	75	58	36	72

Matrix 3



	0	1	2	3	4	7	8	9	10	11	15	20	21	22	27	28	30	31	32	33	35	36	37	38
5	77	46	31	47	40	48	37	28	78	41	44	59	51	77	69	65	25	46	26	76	79	37	31	50
6	41	69	66	53	75	56	43	37	33	60	66	31	56	29	52	31	62	77	53	40	61	62	77	77
12	73	31	42	59	54	69	70	56	78	54	25	38	75	65	69	35	52	26	51	48	43	30	46	47
13	62	51	35	30	59	35	73	54	47	61	52	73	61	34	79	59	43	75	25	60	54	40	32	44
14	79	48	29	52	69	31	43	64	61	77	47	41	71	35	44	33	77	52	43	72	58	41	34	47
16	43	49	63	36	57	37	59	49	65	76	30	29	30	33	49	54	43	32	60	35	64	62	50	45
17	63	77	50	55	41	59	29	38	70	63	67	34	59	25	58	71	58	36	69	48	51	31	41	40
18	68	37	38	77	55	51	72	30	57	62	50	76	40	45	50	45	33	54	55	58	31	75	64	62
19	60	26	67	78	35	74	68	53	52	32	67	25	74	65	77	34	66	75	59	66	34	60	61	73
23	67	35	65	75	42	65	39	64	32	50	70	67	51	34	67	74	78	41	61	61	39	49	37	46
24	74	66	62	34	40	48	28	73	66	71	61	71	49	32	31	74	64	45	70	77	26	48	35	34
25	50	53	32	52	65	27	78	28	50	25	31	47	64	72	43	67	63	74	44	42	52	71	27	57
26	66	76	52	71	47	28	68	59	40	41	63	77	68	74	50	49	71	71	58	55	44	68	47	75
29	62	75	27	31	41	77	58	78	58	51	65	62	30	65	70	74	57	51	40	59	49	62	70	58
34	73	48	51	31	75	34	66	48	27	66	71	70	73	53	77	66	68	52	43	67	46	59	60	36
39	42	74	47	50	53	55	40	39	34	48	66	52	38	29	56	77	68	59	56	78	55	49	30	72

Matrix 4

	5	6	12	13	14	16	17	18	19	23	24	25	26	29	34	39
5	--	13	60	46	14	22	28	19	18	53	46	55	31	19	44	21
6	13	--	43	52	37	41	42	41	27	39	57	31	29	48	31	39
12	60	43	--	16	23	22	42	12	55	55	39	58	44	15	43	12
13	46	52	16	--	29	59	40	18	19	36	13	12	59	23	31	50
14	14	37	23	29	--	46	49	36	31	48	28	58	49	33	34	51
16	22	41	22	59	46	--	53	16	60	27	36	14	53	15	23	31
17	28	42	42	40	49	53	--	54	46	24	57	55	57	33	23	40
18	19	41	12	18	36	16	54	--	12	25	40	15	37	33	15	15
19	18	27	55	19	31	60	46	12	--	13	19	48	25	56	26	16
23	53	39	55	36	48	27	24	25	13	--	19	54	56	58	44	35
24	46	57	39	13	28	36	57	40	19	19	--	47	52	12	38	28
25	55	31	58	12	58	14	55	15	48	54	47	--	18	36	47	42
26	31	29	44	59	49	53	57	37	25	56	52	18	--	29	48	33
29	19	48	15	23	33	15	33	33	56	58	12	36	29	--	22	50
34	44	31	43	31	34	23	23	15	26	44	38	47	48	22	--	58
39	21	39	12	50	51	31	40	15	16	35	28	42	33	50	58	–

6. Comparison with Other Methods

6.1. Comparison with SPSS. In this Section the six sets shown in Fig.1 are considered. All of them are correctly divided by the suggested algorithm of dichotomy (see classifications in Fig.3). The results of classification of the same sets by classification methods offered in the well known software package SPSS are presented below. The results of the *K*-mean method are shown in Fig.18. The results of the hierarchical algorithm (for the between-group linkage version) are shown in Fig.19. The results of other versions of hierarchical classifications are approximately the same. The best results are obtained for Ward’s method, yet only three sets of six are classified correctly (see Fig.20). In a slightly more complicated situation, carefully considered in example 2 with the continuations, Ward’s method gives unsatisfactory results (see Fig.21).

6.2. Other Comparisons. To ascertain that many well-known methods are inoperative (at least, in some simple cases), no formal check of them (i.e. running of the respective programs for the sets, in which the suggested general AC algorithm gives correct classifications) is required at all. Their descriptions imply that in many cases (including rather simple) they will not work properly. Let us dwell on several well-known methods in more detail.

1. <u>Methods minimizing the mixture of variances</u>. It is well known that in the case of two classes this mixture reaches its minimal value for a of non-convex classes a correct classification will not be obtained.

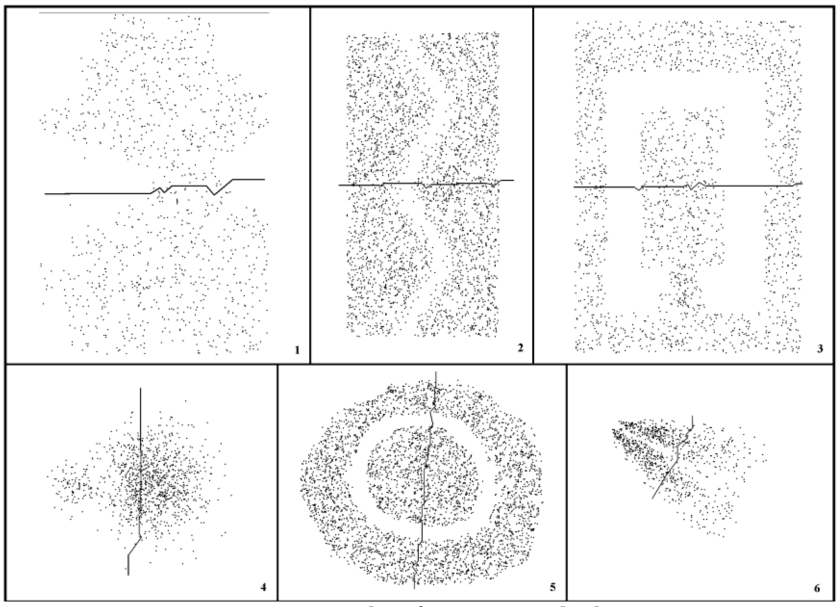


Fig.18. Results of *K*-mean method

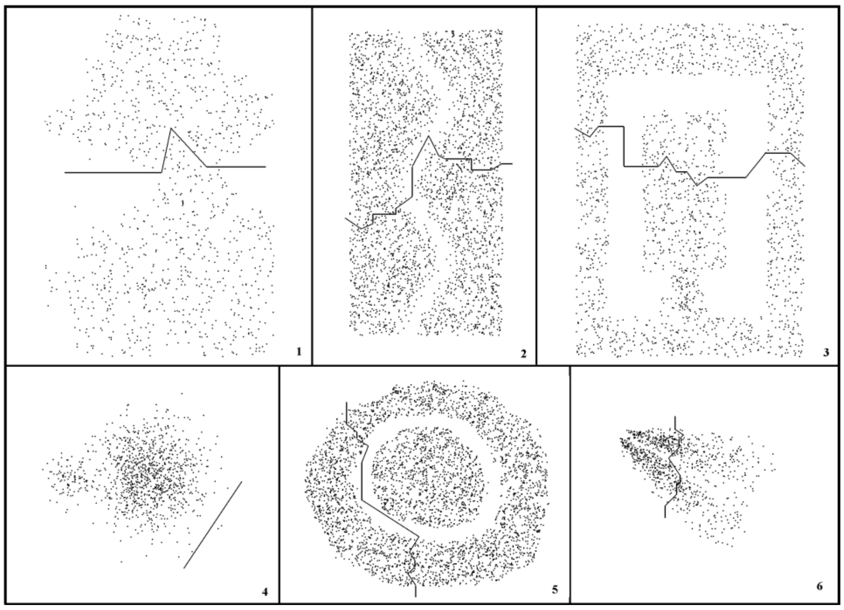


Fig.19. Results of hierarchical classification with between-groups linkage

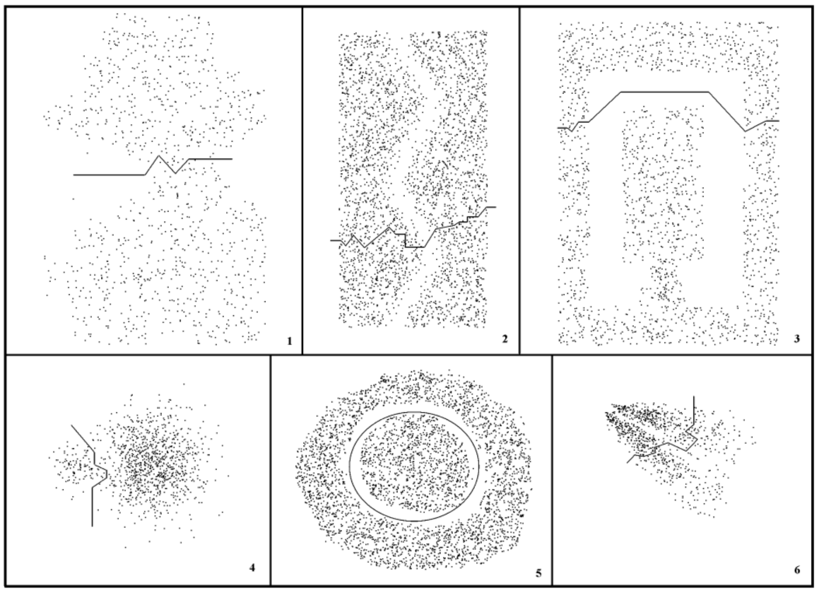


Fig.20. Results of hierarchical classification with Ward's method. Sets 1, 4 and 5 are classified correctly

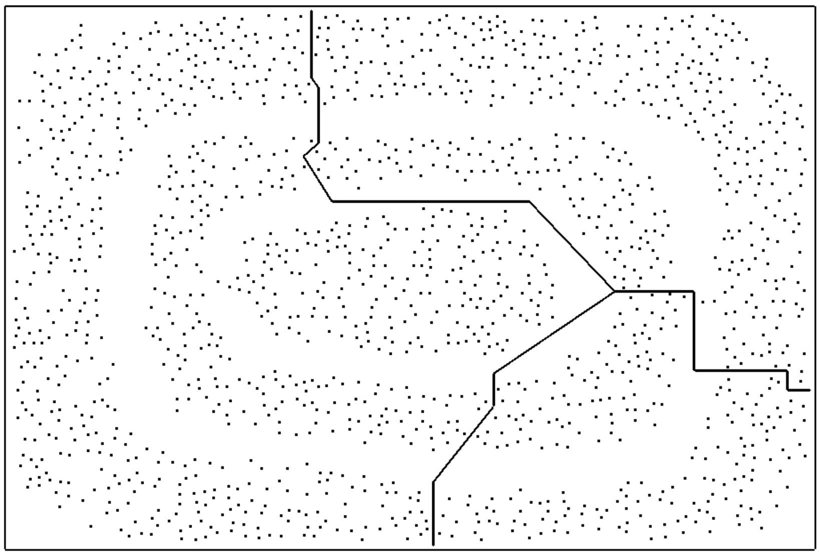


Fig.21. Example of the wrong classification by Ward's method

2. Minimal linkage methods. Assume that A and B form an arbitrary division of an initial set into two subsets. Define
$d(A, B) = \min_{i \in A, j \in B} d_{ij}$.
A division maximizing $d(A, B)$ is considered to be a solution of the AC problem. Example 5, in which the distance between classes is very close to 0, demonstrates that these methods do not work in such cases.

3. Generalized Newman-Girvan Algorithm. The frequency methods are perhaps the strongest AC methods (from the viewpoint of diversity and incidence of problems solved by them). The algorithm copes with all test sets 1 – 6, except for set 4. The result of classification of this set is shown in Fig.22. The reason (as in example 1) consists in a large quotient of cardinalities of correct classes.

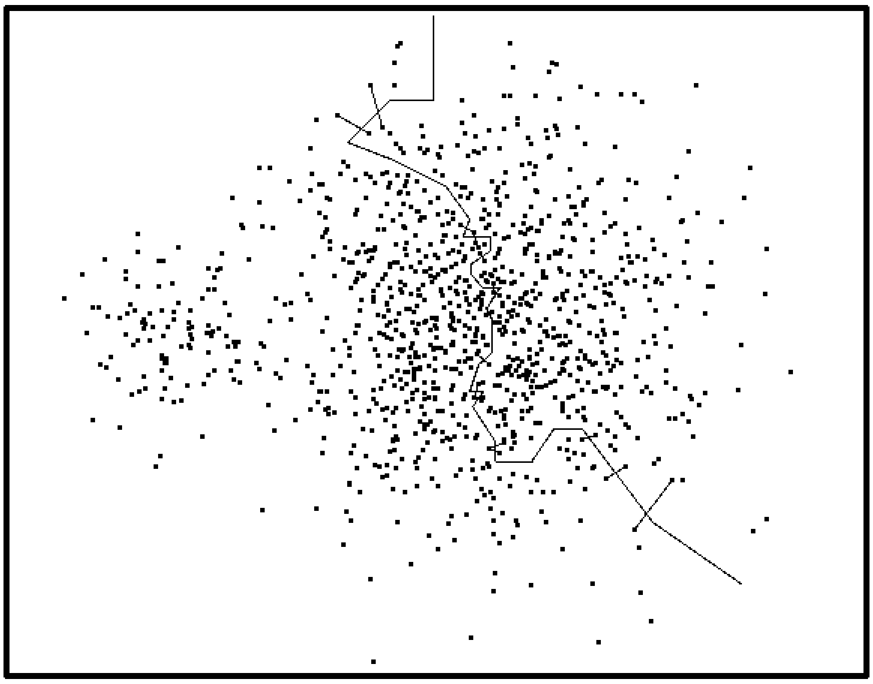


Fig.22. Example of wrong classification by generalized Newman-Girvan algorithm

4. Balanced Cut Criterion (5). In the above mentioned example 1 a discrepancy was noted between this criterion and some AC problems

(whose correct solutions do not correspond to cuts maximizing the criterion). Yet this example implies much more. The review by Luxburg (2007), devoted to Spectral Clustering, discusses the formal correspondence between the optimization problem with criterion (5), on the one hand, and a version of spectral algorithm, on the other hand. In the review of Filippone et al, 2008 attention is paid to the formal correspondence between three approaches: balance criteria optimization, spectral methods and kernel clustering methods. Yet the balance criterion optimization, as was revealed in several cases (not only in example 1 and test set 4), does not lead to correct classifications. Hence, the same conclusion is true for spectral and kernel methods that have been very popular and actively developing in the last few years.

7. Conclusion

The methods mentioned at the end of the previous section demonstrate diversity and depth of their mathematical foundations. However, the essentially simpler scheme suggested in this work is more efficient – first of all from the viewpoint of diversity and difficulty of problems being solved. It seems that practically all of the known methods have a drawback that at the first glance is not very important but in fact proves crucial. They do not contain any instructive description (to say nothing of a formal one) of a class of successfully solved problems. In the book of Braverman and Muchnik published quite a long time ago (1983) there is an honest confession of the authors, who say (in page 140): "It is natural that any specific functional cannot encompass the wide diversity of possible views of what a "good aggregation of elements" is. It is more likely that the biologist, the engineer, the geologist and the economist have different opinions on this subject. Therefore we can only be surprised that some functionals, suitable for solving a wide class of problems from various different fields, could be suggested." However, it is desirable to have a more exact description.

Let us dwell in more detail on the features of the suggested approach.

1. The description of feasible initial sets is sufficiently clear-cut. Specifically, we consider AC problems, presented by undirected graphs. A class corresponds to a subgraph containing <u>many vertices compared to the number of edges connecting the subgraph to other ones</u>. No other data on mutual arrangement of subgraphs, weight of vertices, length of edges,

and so on are used. Note that just as the number of vertices in a subgraph, so the number of edges connecting this subgraph to other ones, can vary essentially depending on the subgraph. It is only important that the quotient of these numbers should be large enough.

2. The character of the scheme does not presuppose the use of a single new idea, but rather a new combination of known ideas and their modifications. In the course of action, the family of constructed classifications first grows and then drops down to one classification.

3. The number of classes is not given in advance. It is limited by a given sufficiently large number.

4. There are relatively (for such a universal scheme) few parameters, all of which are meaningful. Two parameters f and T of the suggested algorithm of frequency dichotomy have little effect on the result. All the above presented results are obtained for $f = 10$ and $T = 1000$. But for $f = 20$ and $T = 1500$, as well as for other arbitrary changes of these parameters within limits 5 – 25 and 500 – 3000, the results practically remain the same. The only parameter of the divisive-agglomerative procedure is the number k of blocks. This parameter is essential. Roughly speaking, for too small values of k correct classifications may be lost (in example 8 for $k = 10$ the smaller isthmus is not found), while for too large values of k redundant stable classifications may appear. Yet for all the intermediate k the same correct classification is found. Finally, at the external cycle of stability check two parameters are given: the number r of random trials and coincidence level α. It is possible to choose arbitrary $r \in \{5, \dots, 10\}$ and $\alpha \in [0{,}95; 0{,}99]$; for the variations within these limits the resulting stable classification remains unchanged.

5. The result of application of the suggested AC algorithm is clear-cut and unambiguous. Of course, preliminary knowledge of the considered AC problem can be very helpful – but rather at the interpretation stage than at the computational one.

Of course, no universal methods of classification exist. The above formulated requirements to the feasibility of initial sets clearly point to the "local" character of the considered classifications because they are defined by local connections between classes. Not all classifications are local, though. In the set shown in Fig.23, the cluster structure evidently consists of two rings. But in this case it would be wrong to say that the classes (in the corresponding neighborhood graph) are connected by a small number of edges. The suggested method leads to an obviously wrong result, even though the classes are connected by only 5 edges. It is only appropriate to repeat the above cited deliberation from the book by Braverman and Muchnik (1983) that one should not be surprised if a method fails in some case but rather, if it proves suitable for many various situations because it holds true for the suggested new AC algorithm.

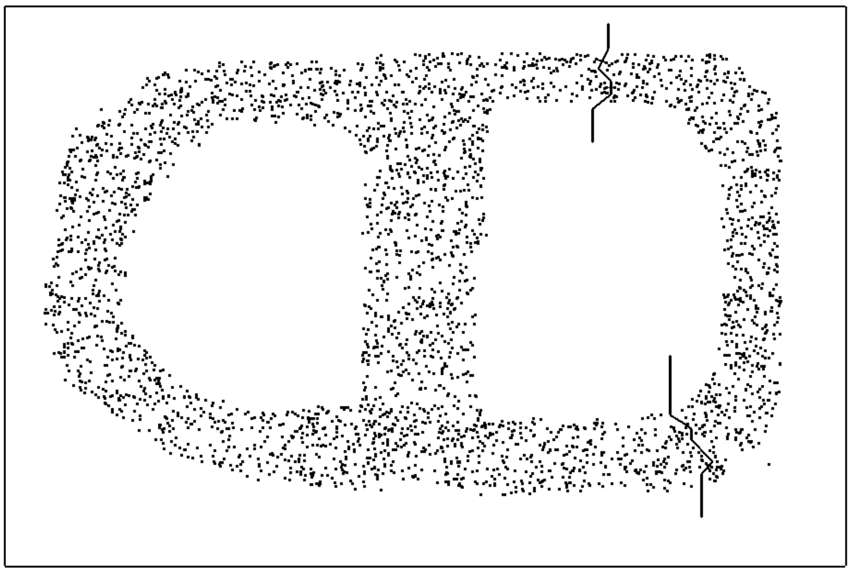


Fig.23. Example of non-local classification

The author is grateful to professors F.T. Aleskerov and B.G. Mirkin for their attention and numerous helpful observations, and his colleagues K.B. Pogorelskiy, E.G. Galitskiy and E.B. Galitskiy for the help in experimenting on SPSS. The author thanks the Decision Analysis Laboratory at SU-HSE for the financial support of the work.

References

Aivazyan S.A., Buchstaber V.M., Yenyukov I.S., Meshalkin L.D. Classification and Reduction of Dimensionality (in Russian). – Moscow: Finansy i Statistika, 1989.

Barseguyan A.A., Kupriyanov M.S., Stepanenko V.V., Holod I.I. Tecnologies of Data Analysis: Data Mining, Visual Mining, Text Mining, OLAP. 2-nd Edition (in Russian) – St.Peterburg.: BHV-Peterburg, 2007.

Braverman E.M., Muchnik I.B., Structured Methods of Experimental Data Processing (in Russian) – Moscow: Nauka, 1983.

Filippone M., Camastra F., Masulli F. Rovetta S. A Survey of Kernel and Spectral Methods for Clustering. – Pattern Recognition, 41(1):176-190, January 2008.

Gordon, A.D. Classification. – Chapman & Hall/CRC, 1999..

Luxburg, U. A Tutorial on Spectral Clustering. – Statistics and Compu-ting 17(4): 395-416, 2007.

Mirkin, B. Mathematical Classification and Clustering. – Kluwer Academic Publishers, 1996.

Mirkin, B., Clustering for Data Mining: A Data Recovery Approach. – Chapman & Hall/CRC, 2005.

Newman, M.E.J., Girvan, M. Community structure in social and biological networks. – Proc. Natl. Acad. Sci. USA **99**, 7821–7826, 2002.

Newman, M.E.J. Detecting community structure in networks. – Eur. Phys. J. B **38**, 321–330, 2004.

Рубчинский, А. А.
Дивизимно-агломеративный алгоритм классификации на основе минимаксной модификации частотного подхода : Препринт WP7/2010/07 [Текст] / А. А. Рубчинский ; Гос. ун-т – Высшая школа экономики. – М.: Изд. дом Гос. ун-та – Высшей школы экономики, 2010. – 48 с. – 150 экз.

В работе рассматривается традиционная задача автоматической классификации (АК). Предложенный подход состоит в новой комбинации достаточно известных методов и их модификации. Сначала осуществляются последовательные дихотомии исходного множества и тем самым строится семейство классификаций на 2, 3, ..., k подмножеств, где k – некоторое число, заведомо превосходящее предполагаемое число классов (дивизимный этап). Используемая дихотомия относится к частотным методам, представляя собой их новую модификацию; она естественно включает в себя элементы рандомизации. Затем из каждой из полученных классификаций строится новое семейство классификаций путём последовательного объединения наиболее близких подмножеств (агломеративный этап). После этого для дальнейшего анализа оставляются только несовпадающие классификации. Наконец, весь процесс повторяется несколько раз, в результате чего большинство из оставшихся классификаций оказываются стохастически неустойчивыми. Устойчивая классификация с максимально возможным числом классов и объявляется решением исходной задачи АК, а отсутствие устойчивых классификаций интерпретируется как отсутствие кластерной структуры в исходном множестве.

Рубчинский А.А. – Лаборатория анализа и выбора решений Государственного университета – Высшей школы экономики и Международный университет «Дубна»

Препринт WP7/2010/07
Серия WP7
Математические методы анализа решений
в экономике, бизнесе и политике

Рубчинский Александр Анатольевич

Дивизимно-агломеративный алгоритм классификации на основе минимаксной модификации частотного подхода

(*на английском языке*)

Публикуется в авторской редакции

Зав. редакцией оперативного выпуска *А.В. Заиченко*
Технический редактор *Н.С. Петрин*

Отпечатано в типографии Государственного университета –
Высшей школы экономики с представленного оригинал-макета
Формат 60×84 $^{1}/_{16}$. Тираж 150 экз. Уч.-изд. л. 2,9
Усл. печ. л. 2,79. Заказ №   . Изд. № 1190

Государственный университет – Высшая школа экономики
125319, Москва, Кочновский проезд, 3
Типография Государственного университета – Высшей школы экономики

Тел.: (495) 772-95-71; 772-95-73