

# Clustering of Points in Multidimensional Space Based on Seurat Ideology

V. A. Lyubetsky<sup>a,\*</sup>, K. Yu. Gorbunov<sup>a,\*\*</sup>, S. A. Pirogov<sup>a,\*\*\*</sup>, G. A. Khaziev<sup>a,\*\*\*\*</sup>,  
and A. I. Aglamazova<sup>a,\*\*\*\*\*</sup>

<sup>a</sup>*Kharkevich Institute for Information Transmission Problems  
of the Russian Academy of Sciences, Moscow, Russia*

*e-mail:* \*lyubetsk@iitp.ru, \*\*gorbunov@iitp.ru, \*\*\*pirogov@iitp.ru, \*\*\*\*khaziev@iitp.ru,  
\*\*\*\*\*aglamazova.an@iitp.ru

Received September 9, 2025; revised November 17, 2025; accepted January 15, 2026

**Abstract**—We discuss modern applied problems of discrete optimization at a mathematical level, and for one of them, clustering points in multidimensional real space, we examine in detail the algorithm for solving it. A characteristic feature of these problems is the large size of the initial data, often represented by matrices with tens of thousands of rows and tens of thousands of columns. This leads to problems of processing large amounts of data in the context of a complex computational algorithm; moreover, it is often essential to obtain a reasonably accurate solution to the problem quickly, so the question of the algorithm complexity is of fundamental importance. Solving such a clustering problem leads to difficult mathematical problems: removing hidden parameters; identifying significant features; switching to optimal and informationally significant point coordinates; specific representation (manifold maps) of the vertices of a weighted graph; selection of a function depending on the current clustering of vertices, the maximization of which leads to the desired clustering; for each cluster, selection of features that individually characterize it; and, finally, reduction of the dimensionality of the source data.

*Key words:* optimal graph transformation, row evolution along a tree, optimal discrete clustering.

**DOI:** 10.1134/S0032946025040027

## 1. INTRODUCTION

In this article, we present a mathematical formulation of a number of modern (but already becoming classical) applied problems and, for one of them, the clustering of a set of points in a multidimensional real space, we describe a solution algorithm based on the Seurat ideology. This algorithm is an example of modern data processing aimed at solving a complex computational problem. Solutions to these problems are widely sought after and used in various applied fields, but we do not address the applied aspects themselves here. Although heuristic algorithms for solving the clustering problem under consideration are widely used, the authors are not aware of any comprehensive mathematical descriptions of its solution; to some extent, algorithms can be extracted from applied works, but there the descriptions are constructed in the context of the corresponding applied fields, which may cause difficulties for a mathematically oriented reader.

For such problems, besides the actual mathematical research, one needs to find an efficient algorithm and prove that the algorithm does indeed find the mathematically described solution; in addition, one needs to find an estimate of the algorithm run time (complexity); and, equally, find an efficient computer implementation of this algorithm. Since these tasks arise in an applied context,

it is not always realized that a computer program is preceded by a mathematical formulation and, at the very least, a mathematical understanding of the problem. In other words, it happens that an applied scientist writes a heuristic computer program before the mathematical formulation and investigation of the problem.

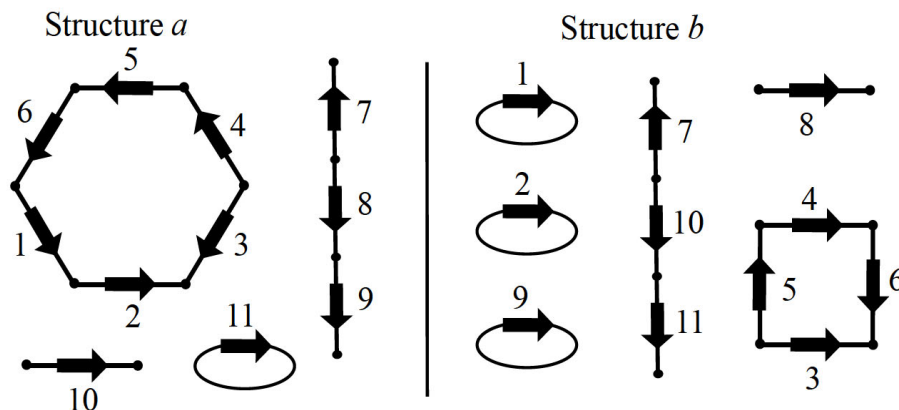
In conclusion, we would like to note that our scientific and methodological article is aimed at a wide range of readers, although it is preferable if they are familiar with the material covered in the first year of a mathematics degree program.

Finally, it is commonly agreed that algorithmic and computer processing of big data is a universal field of research and a method of work in literally all areas of natural sciences, engineering, and human sciences. Such processing relies on methods of modern mathematics (from algorithms/algebra/analysis to geometry) as well as on modern techniques of effective programming.

## 2. EXAMPLES OF PROBLEMS

We start with three examples of mathematical problems, although we do not discuss here methods for solving the first two (otherwise, the text would become much longer and the subject would become scattered), but their mathematical discussion is available by following the references.

1. We are given directed graphs with edge names, either without repeated names or with repetitions. Such graphs are sometimes called *structures*; essentially, we are given two pictures as shown in Fig. 1. Six *operations* are fixed that transform one structure into another; each operation is assigned a strictly positive rational (in general, real) number, which is called the *cost* of the operation. The cost shows how rarely a given operation is used in a process, which in turn is described as the minimum total cost of all operations used in it (including their repetitions): high frequency of use means low cost of the operation, and low frequency of use means high cost. This ensures the adequacy of the process. The six operations on structures mentioned above are well known and are not presented here; see, e.g., [1]. Thus, the task is to find an algorithm that, given two structures  $a$  and  $b$  and given the costs of operations, outputs one of the chains of operations that sequentially transform  $a$  into  $b$  and has the minimum total cost of operations.



**Fig. 1.** Given two structures, weighted directed graphs  $a$  and  $b$ , and costs of each of the six operations on the structures, it is required to find a chain (sequence) of operations, with repetitions, whose total cost is minimal compared to all possible chains starting with  $a$  and ending with  $b$ .

In [2], an algorithm was obtained that is linear in terms of the total number of edges in the data structures  $a$  and  $b$  and finds the minimum chain of operations given the costs. In general, linear complexity of an algorithm is very rare in real-world applied problems.

2. In this section, we are rather talking about a class of problems for which many heuristic solution algorithms are known. Given a rooted tree (sometimes, an acyclic network) for which edge lengths are specified; they correspond to a “discrete time” of transition from an “ancestor” at the beginning of an edge to a “descendant” at the end of the edge (the “beginning” is located closer to the root of the tree, and the “end” is farther from it) . The leaves contain current same-type data from the subject area. Non-leaf vertices are uniquely mapped to objects of the same type as those represented in the leaves; such a mapping is called an *arrangement* of objects over non-leaf vertices. Since the objects are assigned to the leaves in advance, the arrangement is defined on all vertices. In the space of all arrangements, a functional is defined that depends on the variable arrangement (in non-leaf vertices) and on the lengths of the edges. The task is to minimize the functional. In other words, the task is to describe the discrete evolution of a “progenitor”—an object at the root of the tree—along the entire tree up to the current data at the leaves. Examples of objects are sequences in a fixed alphabet, which, as is well known, can be used to encode any finite object; often graphs of a given type. Sometimes the problem is posed so that the tree itself is an argument of a functional or, conversely, a tree is given and all edges are considered to be of equal length, i.e., there are no lengths at all. We have found efficient computer solutions to a number of problems of this kind and proved the corresponding theorems, although in general this important class of problems is far from being mathematically covered; see, e.g., [3].

One of important approaches to defining such evolution is as follows. For given points in  $n$ -dimensional space, there is also specified a field of their velocities, which is interpreted as the development potential of a point/cell (see the beginning of Section 3 below). The velocity field is transformed into a Markov (or close to it) chain of transition probabilities of point/cell development. Fuzzy sets of cells are formed from the chain, which constitute initial, intermediate, and final macrostates, and the probabilities of transition from one macrostate to another are determined. From this, the evolution of macrostates is found, from the initial to the final ones, which are interpreted as clusters of cells [4].

3. Now we turn to the point clustering problem in multidimensional real space, solving which is the subject of the rest of this paper. We are given a numerical matrix whose rows are called *features* and whose columns are called *cells*, not specifically referring to biological cells but rather to any “isolated part of the world.” Hereinafter, columns will be considered as points in the corresponding space. A matrix element is a real number called the *expressiveness* of feature/row  $i$  in cell/column  $j$ . *Clustering* the columns of a matrix is dividing its columns into disjoint sets, each of which is called a *cluster*. It is required to find a clustering for which the columns within any cluster are most similar to each other compared to the similarity of columns between different clusters. The “similarity” of two columns is specified by a functional, which is not so easy to formulate; this will be done in the course of solving the problem, which, alas, deviates from the principle advocated in the Introduction.

Numerous heuristic solutions to this problem are quite complex; as an example, we refer to its solution in the context of applied bioinformatics, see [5]. Below, we outline in several steps our original solution, for which we developed an effective computer program used by us.

In conclusion, we note that in Sections 3–10, we follow the general outline of the Seurat–Louvain–Leiden method. Our goal is to draw the attention of the mathematically oriented reader to numerous mathematical problems that arise in this method, which is far from mathematical not only in its justification, but even partly in its understanding. This equally applies to the problems formulated in Sections 1 and 2.

The next section (Section 3) is elaborated in detail in Sections 4–10; readers who are less familiar with this subject may choose to skip Section 3 at first.

### 3. PLAN FOR SOLVING THE CLUSTERING PROBLEM FOR COLUMNS OF A NUMERICAL MATRIX

Thus, a cell is a point in the  $N$ -dimensional space  $\mathbb{R}^N$  of real numbers. In other words, the *coordinates* of a cell  $j$  are the column of numbers in this matrix, which can be conveniently thought of as “located under the cell”  $j$ . The number of columns in this and subsequent matrices does not change and is equal to  $M$ . Let us present a solution plan that is more or less general for different approaches to this problem. The mathematical justification and complexity estimation of the algorithm proposed below are far from being solved, although they are being actively studied. If desired, the steps of this plan can be read after the rest of the article.

1. In a given numerical  $N \times M$  matrix  $X = (x_{ij})$ , we replace each element  $x_{ij}$  with its share relative to the entire column  $j$ ; we denote the resulting matrix also by  $X$ . For each row  $i$  of the matrix  $X$ , we calculate the mean  $x_i$  and the sample variance  $\sigma_i^2$ , and transform the matrix  $X$  element by element into a new matrix  $Y$ , in which  $\sigma_i^2$  already has a weak dependence on  $x_i$ . Recall that for any  $N \times M$  matrix, the mean of the  $i$ th row is

$$x_i = \frac{1}{M} \sum_{j=1}^M x_{ij},$$

the variance of the row is

$$\sigma_i^2 = \frac{1}{M-1} \sum_{j=1}^M (x_{ij} - x_i)^2$$

and the standard deviation of the row is  $\sigma_i$ .

2. Based on matrix  $Y$ , we form a new matrix  $Z$ . To do this, for each row  $i \in Y$ , we construct a regression  $f$  for the set  $\{(y_\ell, \sigma_\ell^2)\}$  of points on the plane: a point is the variance  $\sigma_\ell^2$  together with the mean  $y_\ell$  of row  $\ell$ , where  $y_\ell$  is taken only from a given neighborhood  $I_i$  of the mean  $y_i$ . Then the matrix  $Z$  is defined as

$$Z = \frac{y_{ij} - y_i}{f(y_i)}.$$

In  $Z$ , we select a prescribed number of rows with the largest new variance  $\delta = \frac{\sigma_i^2}{f(i)}$ . Let  $\tilde{Z}$  denote the part of the matrix  $Z$  containing only these rows.

3. For the columns of  $\tilde{Z}$ , we find the best coordinates, the coordinates in the basis of the left singular vectors of this matrix. Then we project the columns of this matrix onto the subspace of the  $n$  most informative such vectors, where informativeness depends on the corresponding singular numbers. Let us denote the resulting matrix by  $Z^*$ , its size is  $n \times M$ . The value of  $n$  itself is determined by the the fraction of variance unexplained for  $\tilde{Z}$ . Note that the squares of the singular numbers and the left singular vectors coincide with the eigenvalues and eigenvectors, respectively, of the  $M \times M$  covariance matrix  $\frac{1}{M-1} \tilde{Z}^T \cdot \tilde{Z}$  of features.
4. The columns of the matrix  $Z^*$  are simultaneously points in  $\mathbb{R}^n$  (there are  $M$  columns/points) in the  $n$ -dimensional real space  $\mathbb{R}^n$  and, at the same time, they are also *vertices* of the following graph  $G$ . For each of these points  $j$ , we find a list of the  $k$  nearest neighbors of  $j$  based on the Euclidean distance to other points (this  $j$ -list starts with  $j$  itself) and consider the *ranks* of the points from the  $j$ -list in ascending order of distance from  $j$ .
5. In the graph  $G$  we draw an edge between vertices  $j$  and  $\ell$  whenever their lists have a common  $k$ -neighbor. We assign weights to the edges, which completes the definition of the weighted undirected graph  $G$ . We choose some initial clustering  $\mathcal{C}_0$  of the vertices in  $G$ .
6. We will iteratively find the *maximum of the modularity function*, whose argument is the current clustering  $\mathcal{C}$  of vertices in  $G$ ; as a result, we will find the final clustering of vertices in  $G$  (vertices are columns in the matrix  $Z^*$ ).

7. Now we return to the matrix  $Y$ , which has  $n$  rows and  $M$  columns, but now also has its cells clustered (we call this the *clustering matrix*). In each cluster, we will find *differentially expressed rows* (DE rows or, equivalently, *DE features*) using the Mann–Whitney U test and the Benjamini–Hochberg procedure.
8. The graph  $G$  (with coordinates in the  $n$ -space) can be approximated by another graph  $G_1$ , whose vertices are in a  $K$ -space, where  $K$  is much smaller than  $n$ , for example,  $K = 2$ . This is done using a nonlinear function called *UMAP*; the graph  $G_1$  in the  $K$ -space is found as the minimum of the *Kullback–Leibler divergence*. The dimensionality reduction problem solved here (i.e., reducing the dimension from  $n$  to  $K$ ) is in itself one of the central problems in working with big data. Generally speaking, such a dimensionality reduction can be performed from any dimension, for example, from  $N$  to  $K$ .

#### 4. log-TRANSFORMATION OF THE MATRIX $X$

In applied studies, each row  $i$  of an initial matrix  $X$  is often considered as a sample of its random variable  $\mathbf{X}_i$  with mathematical expectation  $u_i$  and variance  $v_i$ , where  $v_i = v(u_i)$ , and  $v(u)$  is a positive quadratic *polynomial* independent of  $i$ . Then the aim is to find a function  $g(x)$ , where  $x$  is any element of the matrix  $X$ , such that the matrix  $Y = \{y_{ij}\}$ , where  $y_{ij} = g(x_{ij})$ , for each row  $i$  has empirical variance  $\sigma_i^2$  depending weaker (as weakly as possible) on its mean  $y_i$ , and in the limit not depending on it. This setting, as well as its subsequent solution, are heuristic, although widely used, as discussed in the Introduction.

Thus, the  $i$ th rows in matrices  $X$  and  $Y$  correspond to random variables  $\mathbf{X}_i$  and  $\mathbf{Y}_i = g(\mathbf{X}_i)$ , where the index  $i$  will be omitted for brevity in what follows. We expand  $\mathbf{Y}$  in powers of  $\mathbf{X} - u$ :

$$\mathbf{Y} = g(u) + g'(u) \cdot (\mathbf{X} - u) + g''(u) \cdot (\mathbf{X} - u)^2 + \dots,$$

and keep only the affine part in the decomposition; we obtain

$$\mathbf{E}(\mathbf{Y}) \approx g(u) \quad \text{and} \quad \text{Var}(\mathbf{Y}) \approx g'(u)^2 \cdot v.$$

Thus, we arrive at the differential equation

$$g'(u)^2 \cdot v(u) = 1.$$

We apply the resulting function  $g$  to each element of each row in  $X$  to obtain the desired matrix  $Y$ . For a monotonic function  $g$ , different row means in  $X$  are transformed into different row means in  $Y$ , which correspond to approximately equal row variances in  $Y$ ; in this sense, the new variance depends weakly on the new mean.

In biochemical and other applications, including the experimental determination of gene expressiveness in cells, the variance  $v$  is often a positive quadratic polynomial of the mean  $u$ . For example, the measured value may be of the form  $\alpha + \mu e^\eta + \varepsilon$ , where  $\mu$  is its exact (true, but unknown to us) value,  $\alpha$  is the data shift, and  $\varepsilon$  and  $\eta$  are additive and multiplicative measurement errors, which are independent random variables, for example, normal with zero mean. Then the variance  $v$  of such a measurement depends on the mean  $u$  precisely as the above polynomial:

$$v = \frac{\text{Var}(e^\eta)}{\mathbf{E}(e^\eta)^2} \cdot (u - \alpha)^2 + \text{Var}(\varepsilon) = (c_1 u + c_2)^2 + c_3.$$

Thus, for this polynomial (and any other polynomial of the type mentioned), the above equation leads to a hyperbolic arcsine transformation in the role of  $g$ . The explicit form of  $g$  for an element

of the matrix  $X$  and for the polynomial written in this way is as follows:

$$x_{ij} \rightarrow y_{ij} = \begin{cases} \frac{1}{c_1} \cdot \operatorname{arsinh}\left(\frac{c_2}{\sqrt{c_3}} + \frac{c_1}{\sqrt{c_3}} \cdot x_{ij}\right), & c_3 > 0, \\ \frac{1}{c_1} \cdot \ln(c_2 + c_1 \cdot x_{ij}), & c_3 = 0, \end{cases}$$

where the hyperbolic arcsine is defined as

$$\operatorname{arsinh}(z) = \ln(z + \sqrt{z^2 + 1}) = \int_0^z \frac{1}{\sqrt{t^2 + 1}} dt,$$

and the constants  $c_1$ ,  $c_2$ , and  $c_3$  are chosen based on the initial data. More precisely, these constants are chosen, whenever possible, from the condition of minimum correlation between the row variances and row means in the matrix  $Y$ .

Note that when  $c_3 = 0$ , we obtain a function that is close to the widely accepted function  $\log_2(1 + x)$  (or, as is often written in applied publications,  $\log_2(x)$ ). Of course, there is no reason to believe that  $c_3 = 0$  always holds for the initial data.

## 5. SELECTING ROWS WITH HIGH VARIANCE

For the matrix  $Y$  and each of its rows  $i$ , we calculate the mean  $y_i$  and variance  $\sigma_i^2$ . Let us construct a low-degree polynomial regression  $f(\cdot)$  for the set of points  $\{(y_i, \sigma_i^2)\}$ , where  $y_i$  is in the middle of the interval  $\mathcal{O}_i$ , and the intervals  $\{\mathcal{O}_i\}$  are chosen so that the number of points in each of them is approximately the same, see Fig. 2. Let  $f(i) = f(y_i)$  be the value of the regression  $f(\cdot)$  at  $y_i$ . The regression refers to the interval  $\mathcal{O}_i$ , and in this sense it is called local, and the number  $f(i)$  can be called the *regression variance* of row  $i$  (sometimes it is called the mean–variance ratio). It characterizes the “dispersion” of row  $i$  without taking into account its individuality. In other words, we will be interested in rows for which the variance  $\sigma_i^2$  does not obey the general rule given by the regression  $f(\cdot)$ , that is,  $\sigma_i^2$  and  $f(i)$  differ significantly,  $f(i)$  being significantly less than  $\sigma_i^2$ .

Now we form the matrix  $Z = \frac{y_{ij} - y_i}{f(i)}$ . For its rows  $i$ , the mean is  $z_i = 0$ , and the variance is  $\delta_i = \frac{\sigma_i^2}{f(i)}$ . In  $Z$ , variances that are too large are considered to be artifacts of the original data; therefore, rows for which  $\delta_i^2 > \sqrt{M}$  are removed from  $Z$ . In the resulting matrix  $Z$ , we keep only rows with the largest regression variance  $\delta_i$  in a given quantity. We denote by  $\tilde{Z}$  the part of the matrix  $Z$  containing only the selected rows (for example, 2000); then its size is  $2000 \times M$ .

Thus, rows of the matrix  $\tilde{Z}$  weakly depend on the mean and do not obey the regression rule, having a large variance. This allows us to clearly distinguish columns by their coordinates.

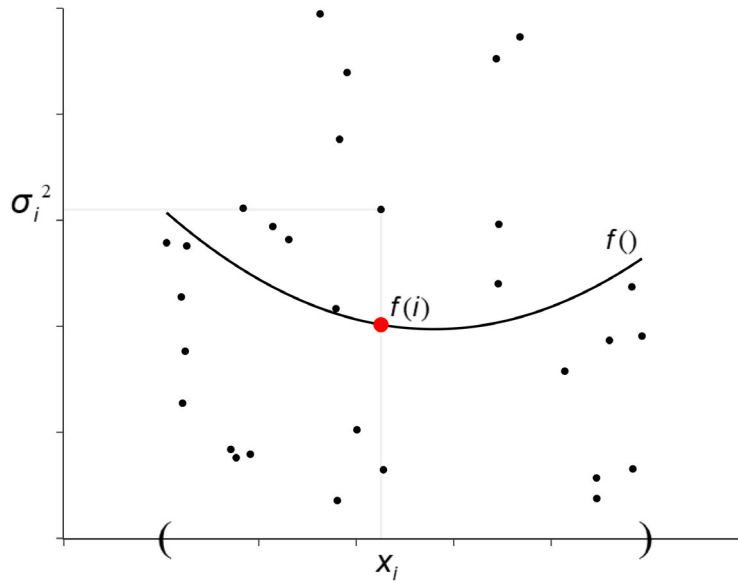
As a result, each cell/column acquires 2000 coordinates representing a point in the 2000-dimensional real space  $\mathbb{R}^{2000}$ . We obtain a set  $\mathcal{M}$  of  $M$  points in this space.

## 6. CHOOSING THE BEST COORDINATES FOR POINTS IN A SET $\mathcal{M}$

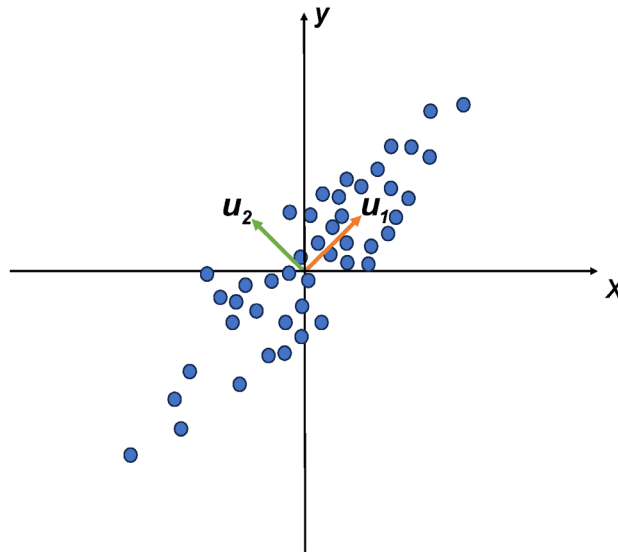
Now we chose even better, *new coordinates* for cell  $j$ , which are linear combinations of its old coordinates, the columns of the matrix  $\tilde{Z}$ . This is done in two steps.

Step 1. For the matrix  $\tilde{Z}$ , we calculate the singular values and left singular vectors  $u_1, u_2, \dots$  of unit length, and switch to the *orthonormal* basis in  $\mathbb{R}^{2000}$  consisting of these vectors, see Fig. 3. It is convenient to assume that the singular values are arranged in descending order

$$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_{2000},$$



**Fig. 2.** Quadratic regression, local on the interval  $\mathcal{O}_i$  with center  $y_i$ , which determines the regression variance  $f(i)$  of row  $i$  of the matrix  $Y$ . In this interval, points on the plane are defined as the mean  $y_\ell$  (abscissa) of row  $\ell$  and its variance  $\sigma_\ell^2$  (ordinate).



**Fig. 3.** For cells  $j$ , we show the transition from their old coordinates, columns of the matrix  $\tilde{Z}$ , to their new coordinates, columns of the matrix  $\tilde{Z}'$ .

as well as the corresponding rows of the matrix and the new coordinate axes. The new coordinates of cell  $j$  (its old coordinates being the elements of the column  $\tilde{Z}_j$ ) are  $(u_k, \tilde{Z}_j)$ , where the  $k$ th coordinate is specified,  $1 \leq k \leq 2000$ . The result of Step 1 is a  $2000 \times M$  matrix, which we denote by  $\tilde{Z}$ .

Step 2. We keep only  $n$  new and most informative coordinates in  $\tilde{Z}'$ , which we call *reduced* (or most informative); they form a matrix  $Z^*$  of size  $n \times M$ , which is obtained from the  $2000 \times M$  matrix  $\tilde{Z}'$  of Step 1.

The choice of  $n$  is based on the following consideration of informativeness of the new coordinates. For brevity, we omit the symbols  $\sim'$  in what follows. Replacing the old coordinates  $Z_j$  of cell  $j$  with

its projection onto the first  $n$  coordinates,

$$Z_j \rightarrow Z_j^* = \sum_{k=1}^n (u_k, Z_j) \cdot u_k,$$

introduces a *mean square error* per cell equal to

$$\frac{1}{M} \sum_{j=1}^M \|Z_j - \widetilde{Z}_j'\|^2 = \sum_{k=n+1}^N \lambda_k,$$

where  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_{2000}$ . This sum is called the *residual variance*.

The quantity

$$\frac{1}{M} \sum_{j=1}^M \left\| \sum_{k=1}^n (u_k, Z_j)^2 \cdot u_k \right\|^2 = \frac{1}{M} \sum_{j=1}^M \sum_{k=1}^n (u_k, Z_j)^2 = \sum_{k=1}^n \lambda_k$$

is referred to as the *explained variance*. The *relative error*  $\delta_n$  is defined as the ratio of the residual variance to the sample variance; this fraction is called the *unexplained variance*:

$$\delta_n = \frac{\lambda_{n+1} + \dots + \lambda_{2000}}{\lambda_1 + \lambda_2 + \dots + \lambda_{2000}}.$$

We choose  $n$  from the condition  $\delta_n \leq \varepsilon_0$ , where  $\varepsilon_0$  is a given threshold.

### 7. CLUSTERING OF CELLS BASED ON THEIR NEW REDUCED COORDINATES (COLUMNS IN THE MATRIX $Z^*$ )

Now we form a graph  $G$ : its vertices are the cells/points  $j$  with their  $n$  reduced and new (most informative) coordinates, which are the columns  $Z_j^*$  of the matrix  $Z^*$ . This graph is called the *kNN* (*k*-Nearest-Neighbors) graph, and the coordinates of the cell/point/vertex  $j$  are called its *PCA coordinates*.

For each vertex  $j$ , based on the Euclidean distance between points in the  $n$ -dimensional space, we compose a list of the  $k$  vertices closest to it (*k*-neighbors, with  $j$  itself being the first in this  $j$ -list, where  $k$  is a parameter). The *j-neighborhood* in  $G$  is defined as the  $j$ -list together with all edges connecting any two vertices from it. Note that in this way we pass to new neighborhoods for all vertices  $j$ , i.e., to maps represented by one-dimensional simplices on the manifold of all cells.

We enumerate the list with successive natural numbers (called *ranks*), starting from 1, in ascending order of Euclidean distance. Let  $\text{rank}(v, j)$  denote the rank of vertex  $v$  in the  $j$ -list, which starts with vertex  $j$ . Vertices that are at the same distance from  $j$  are numbered with fractional numbers: for example, vertices at distances 1, 2, 7, 7, 8, ... have ranks 1, 2, 3.5, 3.5, 5, ...

We connect vertices  $j$  and  $\ell$  with an edge if  $j \neq \ell$ ,  $\ell$  is a neighbor in the  $j$ -list, and  $j$  is a neighbor in the  $\ell$ -list.

The weight  $A_{j\ell}$  of edge  $e = (j, \ell)$  is defined as the maximum over all common  $k$ -neighbors  $v$  for  $j$  and  $\ell$ :

$$A_{j\ell} = \max\{k - 0.5(\text{rank}(v, j) + \text{rank}(v, \ell)) \mid v\}.$$

Let us find some initial clustering  $\mathcal{C}_0$  of the vertices of the graph  $G$  (a partition of its vertices). The choice of  $\mathcal{C}_0$  is important and significantly affects the final result of the clustering; there are several known methods for making this choice. The resulting clusters may allow edges between vertices from different clusters; such edges are said to be *intercluster* (or “*between clusters*”); conversely, edges whose both ends belong to the same cluster are said to be *intracluster* (or “*within a cluster*”).

Starting from the initial clustering  $\mathcal{C}_0$ , we iteratively find the local maximum of the function  $H(\mathcal{C})$ , which is called the *modularity*  $\mathcal{C} = \mathcal{C}(G)$  of the clustering of  $G$  and characterizes the quality of the partition of vertices in the graph  $G$ . The modularity function is defined as

$$H(\mathcal{C}) = \frac{1}{2m} \sum_{j,\ell} \left( A_{j\ell}(e) - \gamma \cdot \frac{k_j \cdot k_\ell}{2m} \right) \rightarrow \max.$$

Here,  $A_{j\ell}$  is the weight of edge  $e$ , which runs through *all edges* in all clusters of the current partition  $\mathcal{C} = \mathcal{C}(G)$ ,  $m$  is the sum of the weights of all edges in  $G$ , and  $k_j$  is the sum of the weights of all edges in  $G$  incident to vertex  $j$ ;  $\gamma$  is a parameter called *granularity*: the minimum total weight of edges in a cluster (relative to the total weight of all edges) over all clusters, divided by the maximum of the similar fraction of the total weight of edges between all pairs of different clusters.

Regarding the parameter  $\gamma$ , we note the following. As it increases, the number  $e$  of negative edges may increase, and it becomes advantageous to divide the cluster into several clusters so that the negative edges are concentrated between the clusters and are thus excluded from  $H$ . Then the number of clusters increases, and the clusters themselves become smaller. Conversely, when  $\gamma$  decreases, the number of positive edges may increase, so it becomes advantageous to merge several clusters into one in order to include intercluster edges that have become positive in  $H$ . Then the number of clusters decreases and they become larger.

A nontrivial choice of the function  $H(\mathcal{C}(G))$  fundamentally affects the clustering result; note the following considerations regarding the choice and justification of the modularity function. Let  $r \in \mathcal{C}$  denote a cluster of a given clustering  $\mathcal{C}$ . The weights  $A_{j\ell}$  of the edges in  $G$  can be interpreted as a weightless multigraph: vertices  $j$  and  $\ell$  are connected by  $A_{j\ell}$  edges. We denote by the same symbol  $A_{j\ell}$  the symmetric incidence matrix of the resulting loopless *multigraph*, which we again denote by  $G$  (the diagonal contains zeros, the other numbers are equal to the weight  $\frac{A_{j\ell}}{2}$ ; as usual, the weight of a loop is doubled). Then  $A_{j\ell} = A_{\ell j}$ , and summing over all edges  $j, \ell \in r$  leads to the doubled sum of the number of edges in  $r$ . In the formula for  $H$ , we switch from summing over edges  $e = (j, \ell) \in G$  to summing over clusters  $r \in G$  to obtain

$$H(\mathcal{C}(G)) = \frac{1}{m} \sum_r \left[ \left( \frac{1}{2} \cdot \sum_{(j,\ell) \in r} A(j, \ell) \right) - \frac{k(r)^2}{4m} \right],$$

where  $m$  is the number of edges in  $G$  and  $\sum_{(j,\ell) \in r} A(j, \ell)$  is twice the number of edges within cluster  $r$ . Here,  $k(j)$  is the incidence degree of vertex  $j$  in the graph  $G$  calculated using the incidence matrix  $A_{j\ell}$ , and  $k(r) = \sum_{j \in r} k(j)$  is the incidence degree of cluster  $r$ . It is easily seen that

$$\sum_{j \in G} k(j) = 2m.$$

Thus, the minuend is the number of edges in  $r \in \mathcal{C}(G)$ , and we want to interpret the subtrahend.

This can be achieved as the average number of edges in a family of random graphs with the same vertices (and parameters) as  $G$  and the same clustering of these vertices. However, we provide a simpler combinatorial explanation. On the set of vertices of the multigraph  $G$  with its clustering, consider an “ideal” (ordinary, not multi-) graph  $G'$  with the same clustering and the same incidence degrees  $k(j)$  as for the vertices of the multigraph  $G$ . Recall that in  $G$  and  $G'$ , the incidence degree is defined as the sum of weights of all edges incident to it. Namely,  $G'$  is a complete graph with loops, and we assume that the weight of the edge  $(j, \ell)$  in it is

$$\frac{k(j)k(\ell)}{2m} \quad \text{if } j \neq \ell, \quad \text{and} \quad \frac{k(j)k(\ell)}{4m} \quad \text{if } j = \ell.$$

In  $G'$ , for any vertex  $j$ , the incidence degree is

$$\sum_{\ell \neq j} \frac{k(j)k(\ell)}{2m} + \frac{2k(j)^2}{4m} = k(j).$$

Furthermore, in  $G'$ , the sum of the weights of all edges in  $r$  is

$$\sum_{\substack{j \neq \ell \\ \{j, \ell\} \in r}} \frac{k(j)k(\ell)}{2m} + \sum_j \frac{k(j)^2}{4m} = \frac{k(r)^2}{2m}.$$

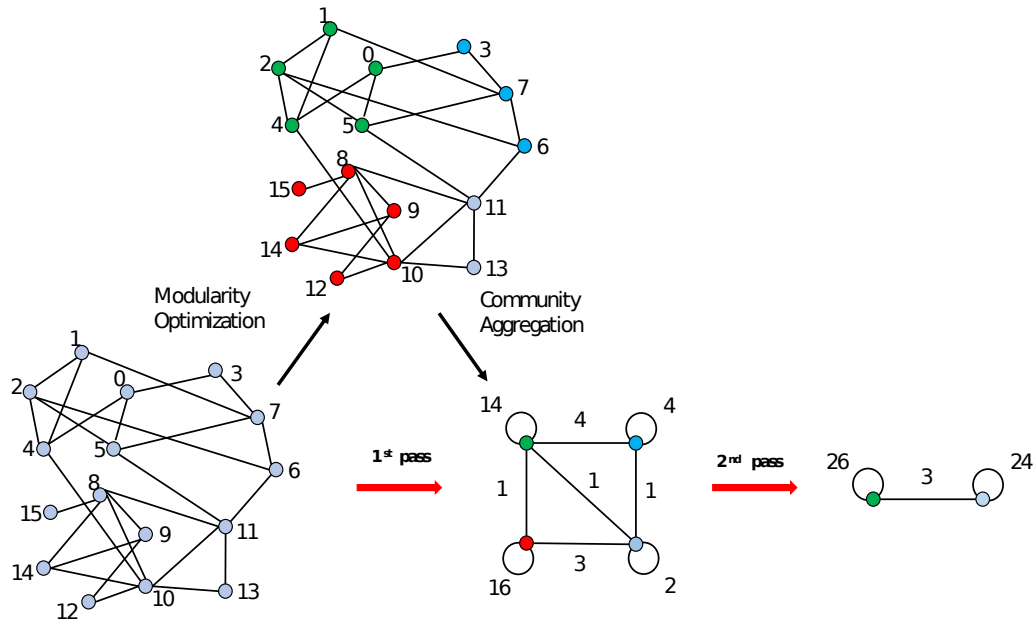
The latter is verified by directly expanding the brackets in  $k(r)$ . We repeat that the graphs  $G$  and  $G'$  have as common parameters the clustering  $\mathcal{C}$  itself, the number of vertices and edges, and the incidence degrees  $\{k(j)\}$  for all vertices  $j$ . The values of  $k(j)$  can be considered as characteristics of graph nonuniformity, which are the same for  $G$  and  $G'$ : for them, the edge density is increased or decreased (compared to the average density) in the same areas.

### 7.1. Leiden Iterative Clustering Algorithm

The algorithm described above can be refined as follows. For the initial graph  $G$ , an initial clustering  $\mathcal{C}_0$  is selected and the modularity function  $H(\mathcal{C}(G))$  is maximized. Based on the resulting graph, we construct a new graph  $G_1$ , in which all vertices of a single cluster in  $G$  are represented by a single new vertex in  $G_1$ . All edges within a cluster in  $G$  are replaced by a loop in  $G_1$  at this new vertex, and all edges between two different clusters in  $G$  are replaced by a single new edge in  $G_1$  between the corresponding new vertices. We assume that the weight of the loop is equal to the total weight of the original corresponding intracluster edges, and the weight of the new edge is equal to the total weight of the original corresponding intercluster edges. After that, we maximize the function  $H(\mathcal{C}(G_1))$ . Thus, the process continues until the maximum of  $H(G_k)$  on the next graph  $G_k$  increases, see Fig. 4. Then, by successive merges, we return to the initial graph  $G$ . Specifically, if vertices in  $G_{k-1}$  belong to the same cluster in  $G_k$ , then the corresponding clusters are merged into a single cluster in  $G_{k-1}$ . By backtracking, we arrive at a clustering of the initial graph  $G$ . We obtain a clustering of vertices in the original  $G$ . At each transition from  $k-1$  to  $k$ , the quality of the corresponding clustering of the graph  $G_k$  does not decrease.

### 7.2. Towards the Justification of the Louvain–Leiden Iterative Clustering Algorithm

As an initial clustering, we can choose *singleton clustering*, which consists of all vertices of the graph  $G_k$  considered as one-element sets (“singletons”). However, a larger clustering called *neighborhood clustering* is usually taken as the initial clustering; we denote it by  $\mathcal{C}_0$ . In what follows, we will denote the graph  $G_k$  by  $G$ , for example, this is the graph  $G$  defined at the beginning of this section, from which the iterative algorithm starts. To construct this  $\mathcal{C}_0$ , we use the  $j$ -neighborhoods of all vertices in  $G$  described at the beginning of this section and the weights of the edges of the graph  $G$ . Let us describe the cluster  $c \in \mathcal{C}_0$ . Each cluster is formed around some vertex  $j \in G$  and in this sense is denoted by  $c_j$  (a cluster with a “center” at  $j$ ). Clusters  $c_j$  are formed iteratively, each starting from the  $j$ -neighborhood. In the subgraph  $c_j$ , let  $d(v)$  be the sum of weights of the edges in  $c_j$  incident to  $v$ ; this is the degree of the vertex  $v$  relative to  $c_j$ . We thin out each  $c_j$  by removing from it the vertices  $v \in c_j$  together with all edges incident to them if  $\frac{d(v)}{|c_j|} < r$ , where  $r \in (0, 1]$  is a threshold and  $|c_j|$  is the sum of the weights of all edges in  $c_j$ . The removal of such vertices from  $c_j$  depends on the order in which the vertices in  $c_j$  are viewed; we continue the removal



**Fig. 4.** All edges have unit weights. The steps of the iterative Leiden algorithm are shown. The example is taken from [6].

until such a vertex exists. Removals are performed independently for each  $j$ . If  $|c_j| = 0$ , then we keep the vertex  $j$ ; if  $v$  is removed from all  $j$ -neighborhoods in  $G$ , then we leave  $v$  as a singleton.

After that, in the resulting set of subgraphs  $c_j$ , we eliminate their intersections by vertices: let  $v$  belong to several  $c_j$ . For each  $c_j$ , we calculate the *average weight* of the edges connecting  $v$  to all vertices in  $c_j$ . We keep the vertex  $v$  in the subgraph  $c_j$  for which the average weight is the largest, and remove it from the other  $c_j$ . The resulting subgraphs  $c_j$  form the clusters of the initial clustering  $C_0$  in the graph  $G$ .

Let us outline the proof that for each transition from  $k$  to  $k + 1$ , the quality of the corresponding clustering of the graph  $G_k$  does not decrease.

For brevity, we prove here the higher quality of the  $C_0$  clustering compared to the singleton clustering in a simplified case, where only  $k$  edges are accounted for at each vertex, vertices from the neighborhoods are not removed and the closest neighbor of two vertices from the  $j$ -neighborhood is the point  $j$  itself, and ranks are pairwise distinct integers. Let  $G$  be a complete graph with loops: this is the case if the missing edges are assigned a weight of 0. Recall that the value of  $H$  on any clustering in  $G$  (up to a multiplicative constant) is equal to the sum of differences of weights of the edges lying inside a cluster in  $G$  and in the ideal graph  $G'$ . Loops lie inside the cluster in any clustering, and therefore contribute equally to  $H$  for singleton and neighborhood clusterings. Thus, we need to estimate the sums of the weights in  $G$  and  $G'$  of the edges that lie inside a cluster in the neighborhood clustering but not in the singleton clustering. These are all edges that are not loops in the clusters  $c_j \in C_0$ .

Therefore, in  $G$ , the sum of the weights of the edges from  $c_j \in C_0$  that are not loops is

$$\frac{k^2(k-1)}{2} - \frac{1}{2} \sum_{i \neq \ell} (i + \ell) = \frac{k^2(k-1)}{2} - \frac{(k-1)k(k+1)}{4} = \frac{k(k-1)^2}{4};$$

here,  $i$  and  $\ell$  are the ranks of vertices in  $c_j$ , which take values from 1 to  $k$ . Let us estimate the sum  $k_i$  of the weights of edges incident to a vertex of rank  $i$  from  $c_j \in C_0$ :

$$k_i = k(k - 1) - \frac{1}{2} \sum_{\ell \neq i} (i + \ell) = \frac{3k^2 - 5k - 2ik + 4k}{4} \leq \frac{3k(k - 1)}{4}.$$

For  $\gamma = 1$ , in the ideal graph  $G'$ , the sum of the edge weights from  $c_j$  is

$$\frac{1}{2m} \sum_{i \neq \ell \in c_j} k_i k_j \leq \frac{1}{2m} \left( \frac{3}{4} k(k - 1) \right)^2 \frac{k(k - 1)}{2} = \frac{9}{64m} k^3 (k - 1)^3.$$

Thus, the quality of neighborhood clustering is greater than that of singleton clustering if

$$\frac{k(k - 1)^2}{4} > \frac{9}{64m} k^3 (k - 1)^3,$$

i.e.,

$$m > \frac{9}{16} k^2 (k - 1).$$

Usually, graphs with  $m > 15000$  and  $k$  no greater than 30 are processed, so this inequality is satisfied.

**Theorem 1.** *Let  $G$  be a complete graph with loops,  $C_k$  be a final clustering of level  $k$ , and  $C_{k+1}$  be a clustering of level  $(k + 1)$  in which all clusters are singletons. Then  $H(C_k) \leq H(C_{k+1})$ .*

**Proof.** Let us call the clusters in  $C_{k+1}$  of the  $(k + 1)$ st level large, the clusters in  $C_k$  of the  $k$ th level medium, and the clusters in  $C_{k-1}$  of the  $(k - 1)$ st level small. Similarly, we will name the clusterings themselves in these graphs.

From the definition of the weight of an intercluster edge, it follows that the total weight  $m$  of all intercluster edges does not change when moving to the next clustering level (regardless of whether the clusters of the next level are singletons). It also follows that if large clusters are singletons, then the total weight of all edges lying within such a singleton (which is the weight of the loop at the corresponding medium cluster) is equal to the total weight of the edges between small clusters lying within the medium cluster. Summing this weight over singletons, we obtain that the sum  $\sum A_{j\ell}$  from the formula for  $H$  for large clustering is equal to this sum for the medium one.

It remains to consider the sum of the products  $k_j \cdot k_\ell$  for pairs  $(j, \ell)$  of vertices lying within a cluster of the given clustering. Any large singleton cluster  $a$  consists of one medium cluster, which we will denote by  $a'$ . For the large clustering, a term in the above-mentioned sum is the squared weight of the loop at  $a'$ , which we will call the first sum. For the medium clustering, this is the sum over all pairs  $(j, \ell)$  of small clusters (in  $a'$ ) of the products  $k_j \cdot k_\ell$ ; we will call it the second sum.

The first sum is not greater than the second. Indeed, when expanding the brackets in the first sum, we obtain the products  $w(e_1) \cdot w(e_2)$  of the weights of edges  $e_1$  and  $e_2$  between small clusters lying inside  $a'$ , and each product occurs twice if  $e_1 \neq e_2$  and once otherwise. In the second sum, when the brackets are expanded, the products  $w(e_1) \cdot w(e_2)$  of the weights of edges  $e_1$  and  $e_2$  will also occur, but not necessarily lying inside  $a'$ . However, each such product for edges lying inside  $a'$  occurs at least twice if  $e_1 \neq e_2$  and at least once otherwise. Therefore, the part of the sum  $\sum k_j k_\ell$  in the formula for  $H$  related to a single cluster  $a$  for the large clustering is not greater than the corresponding part of the sum (related to  $a'$ ) for the medium clustering. Summing over all singletons and taking into account that the sum  $\sum k_j k_\ell$  enters  $H$  with a minus sign, we obtain the result.  $\triangle$

However, there is no guarantee that the function  $H$  does not decrease during the backward pass of the procedure. Therefore, we started this pass with a final clustering of each level, not necessarily the last one, and from all the clusters obtained in this way in  $G$ , we selected the cluster with the maximum value of  $H$ .

## 8. DIFFERENTIALLY EXPRESSED FEATURES

We have already obtained the clustering (partitioning)  $\{G_s\}$  of the cells/columns of the original  $N \times M$  matrix  $Y$ , which has undergone log-normalization but not scaling-centering; the matrix consists of old coordinates, before the transition to the PCA coordinates and the reduction of the less informative of them; here,  $s$  runs through all the clusters. For each cell cluster  $G$  and each row  $i$ , we define differentially expressed features. Such features/rows are abbreviated as *DE features* (or *DE rows*). Intuitively speaking, a feature/row  $i$  is called *differentially expressed* if the expressions in cells  $j \in G$  are statistically significantly different from the expressions in all other cells  $\ell \notin G$  of the same row  $i$ . This understanding of *DE rows* for a given cluster  $G$  is clarified below: first, using the Mann–Whitney U test [7–9], then using the Benjamini–Hochberg procedure [10], as well as a number of other conditions.

Thus, a row  $i$  is fixed (its name and the corresponding index need not be mentioned), and a cluster  $G$  is fixed, which induces a partition of the row into two sets, which we will denote by the same symbols  $G$  and  $\neg G$ , still calling them clusters. Let us arrange all expressions of the row  $i$  in ascending order; and assign numbers in the resulting sequence of expressions with *ranks*, i.e., natural numbers, also in ascending order, starting with 1. To identical numbers, we assign their average rank. Let  $R_1$  be the sum of the ranks of expressions in a given cluster  $G$ , and  $R_2$  be the sum of the ranks of expressions outside this cluster, i.e., in  $\neg G$ ; let  $m_1$  and  $m_2$  be the cardinalities of the sets  $G$  and  $\bar{G}$ , respectively;  $m_1 + m_2 = M$ . Since

$$R_1 + R_2 = M \cdot \frac{1 + M}{2},$$

we can operate with only  $R_1$  or only  $R_2$ .

It is assumed that  $G$  and  $\bar{G}$  are *independent* samples from two continuous distributions,  $\mathfrak{F}$  and  $\mathfrak{L}$ , which either *coincide* or *do not coincide*. For experimental-mathematical samples  $G$  and  $\neg G$ , the question of their independence is difficult to resolve, so it remains at the calculator’s discretion; in our situation, this is an assumption. If  $\mathfrak{F} = \mathfrak{L}$ , then for the expressions  $y_j \in G$  and  $y_\ell \in \neg G$ , we have  $\mathbf{P}(y_j < y_\ell) = 1/2$ . This property is called the absence of “*dominance*” of  $G$  and  $\neg G$  over each other. The converse is not true: for example, if  $\mathfrak{F}$  and  $\mathfrak{L}$  are two normal distributions with the same mathematical expectations but different variances, then there is no dominance:  $\mathbf{P}(y_j - y_\ell < 0) = \frac{1}{2}$ , since the difference between these distributions has zero mathematical expectation.

Thus, we want to verify whether the *null hypothesis*, which consists of the “absence of domination of  $G$  and  $\neg G$  over each other,” is satisfied. If the null hypothesis is rejected, row  $i$  is called a *DE feature*. If the null hypothesis is rejected, then we have  $\mathfrak{F} \neq \mathfrak{L}$  at the same confidence level.

### 8.1. Mann–Whitney–Wilcoxon test (Wilcoxon rank-sum test, Mann–Whitney U test, and finding the $p$ -value; abbreviated as MWW)

By row  $i$  of the matrix  $Y$ , we form a random variable

$$U = R_2 - \frac{m_2(m_2 + 1)}{2} = \sum_{j=1}^{m_1} \sum_{\ell=1}^{m_2} I(x_j < x_\ell),$$

where  $x_j$  and  $x_\ell$  are expressions in the cluster  $G$  and outside it, i.e., in  $\neg G$ , respectively; and  $I(\cdot)$  is the characteristic function of the condition given in parentheses. If the null hypothesis is true, then its mathematical expectation and variance are

$$\mathbf{E}(U) = \frac{m_1 \cdot m_2}{2} \quad \text{and} \quad \sigma(U)^2 = m_1 \cdot m_2 \frac{M+1}{12}.$$

In Section 6, the matrix  $Y$  has been centered and normalized. Similarly, this general procedure is applied to  $U$  to obtain the statistics (random variable)

$$Z = \frac{U - m_1 m_2 / 2}{\sqrt{m_1 m_2 (M+1) / 12}}.$$

If in the sequence of ranks there is asymmetry (with respect to the linear order of numbers) of expressions in favor of  $G$  or, conversely, in favor of  $\neg G$ , then  $Z$  deviates to the right or left of zero, more or less. If this deviation  $Z$  is greater than a given threshold  $Z_0$ , then we consider the feature string  $i$  to be *DE*; let us explicate this approach.

Once again, let  $G$  and  $\neg G$  be samples of two random variables. Once again, the null hypothesis is that

$$\mathbf{P}(y_j < y_\ell) = \frac{1}{2} \quad \text{for} \quad y_j \in G \quad \text{and} \quad y_\ell \in \bar{G}$$

(in other words, “ $y_j$  does not dominate  $y_\ell$ , and  $y_\ell$  does not dominate  $y_j$ ”). This hypothesis expresses the “absence of order asymmetry” between  $G$  and  $\bar{G}$ . Note that the distribution for  $Z$  is given by a measure on  $\mathbb{R}$ , which is defined as  $Z^{-1}$  of the measure on the  $M$ -space induced by the distributions  $\mathfrak{F}$  and  $\mathfrak{L}$ .

When the null hypothesis is satisfied and  $m_1, m_2 \rightarrow \infty$ , the distribution of the statistic  $Z$  tends toward the 0–1 normal distribution, denoted by  $N(0, 1)$ . Because of this, the approximate random variable  $Z$  has distribution  $N(0, 1)$ ; i.e., instead of the usually unknown distributions  $\mathfrak{F}$  and  $\mathfrak{L}$ , the probability, denoted by  $\mathbf{P}$  throughout what follows, will be calculated with respect to the normal distribution  $N(0, 1)$ .

Although this is not used below, note that if the expression in  $G$  dominates the expression in  $\neg G$ , i.e.,  $\mathbf{P}(y_j < y_\ell) > 1/2$ , then  $Z$  tends to  $+\infty$ , and if the expression in  $G$  dominates the expression in  $\neg G$  in the same sense, then  $Z$  tends to  $-\infty$  with  $m_1$  and  $m_2$  also increasing.

We select  $Z_0$  from the condition  $2\mathbf{P}(x > Z_0) = \alpha$ . Then  $Z_0$  is called the *quantile of significance level*  $\alpha$ ;  $\alpha$  itself is specified by a calculator, for example,  $\alpha = 0.1$ .

The  $p$ -value for a given row  $i$ , denoted by  $p_{\text{val}}(i)$ , is defined by the distribution  $N(0, 1)$  as twice the area under the distribution to the right of the number  $|Z|$ , i.e.,

$$p_{\text{val}}(i) = 2\mathbf{P}(x > |Z|);$$

the empirical value  $Z = Z(i)$  is calculated from the given row  $i$ .

Note that

$$|Z(i)| > Z_0 \iff p_{\text{val}}(i) < \alpha.$$

The number  $p_{\text{val}}(i)$  is called the *current significance level* with a significance level of  $\alpha$ ; for example,  $\alpha = 0.05$ .

If  $p_{\text{val}}(i) < \alpha$ , then the row/feature  $i$  is included in the *preliminary list* of DE features of the given cell cluster  $G$  (and the null hypothesis is rejected). Otherwise, the null hypothesis for  $i$  is accepted.

Further reduction of the *preliminary list* of DE features is related to the possible randomness of  $p_{\text{val}}(i)$  itself.

8.2. *Supplementing the MWW with the Benjamini–Hochberg procedure. Finding the  $p_{\text{adj}}$ -value*

Let us rearrange the rows/features  $i$ ,  $1 \leq i \leq N$ , in the matrix  $Y$  in ascending order of the numbers  $p_{\text{val}}(i)$  themselves:  $p_{\text{val}}(1) \leq \dots \leq p_{\text{val}}(n)$ , where the former row  $i$  acquires some number  $k$ .

By the definition:

$$p_{\text{adj}}(k) = \min \left\{ \frac{n}{k} \cdot p_{\text{val}}(k), p_{\text{adj}}(k + 1) \right\}, \quad p_{\text{adj}}(n) = p_{\text{val}}(k).$$

Then

$$p_{\text{adj}}(k) \leq p_{\text{adj}}(k + 1) \quad \text{and} \quad 0 < p_{\text{val}}(k) \leq p_{\text{adj}}(k) \leq 1.$$

Thus, a *DE feature* is again preliminarily called a row  $i$  (with a new number  $k$ ) that satisfies the condition  $p_{\text{adj}}(k) < \alpha$  as the first step in Section 8.1. A very weak variant is  $\alpha = 0.1$ .

By backward induction, it is easy to verify the following: for row  $k$  if  $p_{\text{val}}(k) > \alpha$  (i.e., row  $k$  is not a DE feature with respect  $p_{\text{val}}$ ), then  $p_{\text{adj}}(k) > \alpha$  (the same row is not a DE feature with respect to  $p_{\text{adj}}$ ).

On the DE features selected in this way, three additional conditions are imposed in accordance with the parameters  $q$ ,  $q_1$ , and  $q_2$  (for example,  $q = 2$  and  $q_1 = q_2 = 0.25$ , or a very weak variant of the parameters:  $q = 0.1$ ,  $q_1 = 0.01$ , and  $q_2 = 0$ ).

The first condition on the DE features:

$$\log_2 \text{FC}(i) = \left| \log_2 \left( \frac{1}{m_1} \cdot \sum_{j \in G} x_j \right) - \log_2 \left( \frac{1}{m_2} \cdot \sum_{\ell \notin G} x_\ell \right) \right| > q.$$

The second and third conditions on the DE features: for feature  $i$ , we denote by pct.1 and pct.2 the fractions of cells with nonzero expression in cluster  $G$  and in its complement  $-G$ , respectively; and we set

$$\max\{\text{pct.1}, \text{pct.2}\} \geq q_1, \quad |\text{pct.2} - \text{pct.1}| \geq q_2.$$

The final determination of a DE feature consists in fulfilling all of the above conditions.

A marker is a DE feature for which  $p_{\text{adj}}(i) < \beta$  with a particularly small value  $\delta \ll \alpha$  (for example,  $\beta = 10^{-20}$  or less); the value of  $\beta$  is selected based on the data.

9. EXPLANATION TO THE BENJAMINI–HOCHBERG PROCEDURE

Until the end of this section, we assume the null hypothesis, denoting it by (\*).

Recall that the distribution for  $Z$  only approaches the normal distribution; and  $\mathbf{P}(x)$  is here more convenient to understand as  $\mathbf{P}(x) = \int_x^\infty e^{-\frac{t^2}{2\pi}} dt$ .

Then the probability of *rejecting* the null hypothesis is

$$\mathbf{P}(p_{\text{val}}(i) < \alpha) = \mathbf{P}(x > \mathbf{P}^{-1}(\alpha/2)),$$

where  $\mathbf{P}^{-1}(\alpha/2) = Z_0$  and  $\mathbf{P}^{-1}$  is the inverse function of  $\mathbf{P}$ . Thus, if the null hypothesis is true, then  $\mathbf{P}(p_{\text{val}}(i) < \alpha) = \alpha$ . The matrix  $Y$  has  $N$  rows, so

$$\mathbf{P}(\text{“the null hypothesis is accepted for all rows”}) = (1 - \alpha)^N,$$

and

$$\mathbf{P}(\text{“the null hypothesis is rejected for some rows”}) = 1 - (1 - \alpha)^N,$$

which for large  $N$  seems paradoxical under assumption (\*). The renumbering of rows (from  $i$  to  $k$ ) mentioned in Section 8.2 does not affect this seeming paradox.

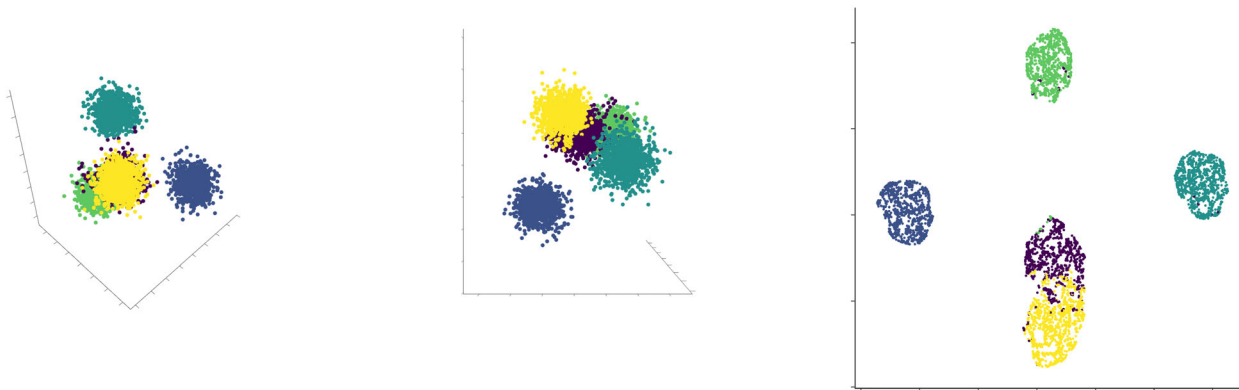
For the Benjamini–Hochberg procedure, it has been proved (not presented here) that replacing the condition  $p_{\text{val}}(i) < \alpha$  with the condition  $p_{\text{adj}}(i) < \alpha$  yields

$$\mathbf{P}(\{k \mid p_{\text{adj}}(i) < \alpha\} = \emptyset) \geq 1 - \alpha,$$

which reduces the combination of an unlikely event with the assumption (\*). Note that replacing  $p_{\text{val}}$  with  $p_{\text{adj}}$ , on the one hand, removes falsely differentially expressed features (“false positives”), and, on the other hand, can lead to the loss of truly differentially expressed features (“false negatives”).

## 10. DATA DIMENSIONALITY REDUCTION

If a clustered set of points (data) in a high-dimensional space  $\mathbb{R}^n$  is mapped to the space  $\mathbb{R}^K$ , where  $K < n$ , for example, to a plane ( $K = 2$ ), by ordinary (linear) projection, then projections of the clusters in  $\mathbb{R}^n$ , i.e., clusters in  $\mathbb{R}^K$ , may overlap or be closer in the projection than in the Euclidean distance in  $\mathbb{R}^n$ . Such an inadequate situation is aimed to be avoided by a nonlinear “projection” called UMAP, see Fig. 5. Usually, data in  $\mathbb{R}^n$  and  $\mathbb{R}^K$  are represented by graphs  $G$  and  $G_1$ , whose vertices uniquely correspond to each other and to points in the corresponding spaces. Thus, the dimensionality reduction problem consists in transitioning from data in  $\mathbb{R}^n$  to the corresponding data in  $\mathbb{R}^K$ . More precisely, from graph  $G$  to graph  $G_1$ , which have the same cells/vertices.



**Fig. 5.** Example of the nonlinear UMAP projection of data (not presented here) in high dimensions onto a plane ( $K = 2$ ).

To be more precise, we are given cells/points  $j$  in  $n$ -dimensional space with their PCA coordinates (and simultaneously  $j$  are vertices of the graph  $G$ ). As in Section 7, for each point/cell  $j$ , we construct a list of  $k$ -neighbors, and  $\rho(j, \ell)$  is the Euclidean distance from  $j$  to its  $k$ -neighbor  $\ell$ . Let  $\rho_j$  be the smallest nonzero distance from  $j$  to its nearest  $k$ -neighbor. For a given  $j$ , we calculate the normalizing number  $\sigma_j$  given by the equation

$$\sum_{\ell} \exp\left(-\frac{\rho(j, \ell) - \rho_j}{\sigma_j}\right) = \log_2 k,$$

where  $\ell$  runs through all  $k$ -neighbors of the cell/point  $j$ .

A new graph  $G_1$  is constructed, which has the same vertices/cells, edges, and clustering as the graph  $G$ ; but the vertices of  $G_1$  correspond uniquely to points in the  $K$ -dimensional space. Thus,

the vertices/points in  $G$  are mapped to the vertices/points in  $G_1$ ; this mapping is what is called UMAP; see [11].

Now we proceed to constructing the *weights* of edges and the *coordinates* of vertices in the graph  $G_1$ . But first, we change the weights in the graph  $G$  and denote the resulting graph by  $G_{01}$ . In it, vertices/cells  $j$  and  $\ell$  are connected by an undirected edge whenever they are  $k$ -neighbors of each other, as was the case in Section 7. But we set the weight of the edge  $e = (j, \ell)$  in  $G_{01}$  to be

$$W(j, \ell) = w(j \rightarrow \ell) + w(\ell \rightarrow j) - w(j \rightarrow \ell) \cdot w(\ell \rightarrow j), \quad 0 < W(j, \ell) < 1,$$

where the auxiliary one-sided weight is

$$w(j \rightarrow \ell) = \exp\left(-\frac{\rho(j, \ell) - \rho_j}{\sigma_j}\right).$$

The weight  $W(j, \ell)$  can be thought of as the probability of at least one edge existing between vertices  $j$  and  $\ell$ .

Thus, we obtain a new (in terms of weights) undirected loaded graph  $G_{01}$  in the  $n$ -dimensional space of “high dimensionality,” where the same cells  $j$  are vertices with the same edges  $e$  and new weights  $W(e)$ . For brevity, let  $K = 2$ .

Let us proceed to constructing the *actual graph*  $G_1$  on the plane: it has the same vertices  $j$  and edges  $e$  as the graphs  $G$  and  $G_{01}$ , but again with new weights. Let  $x(j)$  denote the coordinates on the plane of the cell/vertex  $j \in G_1$ . We define in  $G_1$  the weight of the same edge  $e = (j, \ell)$  as in  $G_{01}$ , setting

$$S_{j\ell} = \left(1 + a\|x(j) - x(\ell)\|_2^{2b}\right)^{-1}, \quad 0 < S_{j\ell} < 1,$$

where  $a$  and  $b$  are parameters (“UMAP weights”). The weights  $S_{j\ell}$  in  $G_1$  are close to the inverse Euclidean distance between  $x(j)$  and  $x(\ell)$ .

It remains to compute the coordinates  $x(j)$  of all vertices  $j$  in such a way that  $G_{01}$  and  $G_1$  are, in a sense, close to each other. The UMAP problem is to find all coordinates  $x(j)$  from the condition of proximity of two distributions on the same edges in  $G_{01}$  and  $G_1$ . Namely, to find them from the minimum condition for the following function, called the Kullback–Leibler divergence:

$$L(x) = \sum_e L(x(j_1), \dots, x(j_M)) = \sum_e \left[ W(e) \log\left(\frac{W(e)}{S(e)}\right) + (1 - W(e)) \log\left(\frac{1 - W(e)}{1 - S(e)}\right) \right] \rightarrow \min,$$

minimizing over the coordinates  $x = x_1, x_2; \dots; x_{M-1}, x_M$  of all vertices in  $G_1$ .

The minimization result will determine the desired 2-dimensional graph  $G_1$  on the plane. Once again: the clustering from the graph  $G$  in the  $n$ -dimensional space is carried over to it.

As usual, numerical minimization depends significantly on the choice of a starting point. The initial coordinates of the vertices of graph  $G_1$  can be chosen as the first two most informative coordinates of points in the  $n$ -dimensional space.

We rewrite the minimized functional:

$$L(x) = \sum_j \sum_{\ell \neq j} \left( W_{j\ell} \cdot \log \frac{W_{j\ell}}{S_{j\ell}} + (1 - W_{j\ell}) \cdot \log \frac{1 - W_{j\ell}}{1 - S_{j\ell}} \right) \rightarrow \min.$$

We obtain the following minimization problem:

$$\begin{aligned} \min_x L(x) &= \min_x \left( - \sum_j \sum_{i \neq j} \left( W_{j\ell} \cdot \log S_{j\ell} + (1 - W_{j\ell}) \cdot \log(1 - S_{j\ell}) \right) \right) \\ &= \min_{x,y} \sum_j \sum_{\ell \neq j} \left( W_{j\ell} \cdot A_{j\ell}(S) + (1 - W_{j\ell}) \cdot R_{j\ell}(S) \right), \end{aligned}$$

where

$$A_{j\ell} = -\log S_{j\ell}, \quad R_{j\ell} = -\log(1 - S_{j\ell}).$$

To minimize over  $x(j)$ , the gradient descent or stochastic gradient descent algorithms can be used, in which iterations are terminated if the minimized function begins to change little.

#### FUNDING

This research was supported by the Russian Science Foundation under grant no. 24-44-00099 (<https://rscf.ru/project/24-44-00099/>).

#### CONFLICT OF INTEREST

The authors of this work declare that they have no conflicts of interest.

#### REFERENCES

1. Gorbunov, K.Yu. and Lyubetsky, V.A., An Almost Exact Linear Complexity Algorithm of the Shortest Transformation of Chain-Cycle Graphs, <https://arXiv.org/abs/2004.14351> [math.CO], 2020.
2. Gorbunov, K. and Lyubetsky, V., Linear Time Additively Exact Algorithm for Transformation of Chain-Cycle Graphs for Arbitrary Costs of Deletions and Insertions, *Mathematics*, 2020, vol. 8, no. 11, p. 2001 (28 pp.). <https://doi.org/10.3390/math8112001>
3. Gorbunov, K. and Lyubetsky, V., Algorithms for the Reconstruction of Genomic Structures with Proofs of Their Low Polynomial Complexity and High Exactness, *Mathematics*, 2024, vol. 12, no. 6, p. 817 (26 pp.). <https://doi.org/10.3390/math12060817>
4. Weiler, P., Lange, M., Klein, M., Pe'er, D., and Theis, F., CellRank 2: Unified Fate Mapping in Multiview Single-Cell Data, *Nat. Methods*, 2024, vol. 21. P. 1196–1205. <https://doi.org/10.1038/s41592-024-02303-9>
5. Seurat – Guided Clustering Tutorial, Satja Lab., 2023, [https://satijalab.org/seurat/articles/pbmc3k\\_tutorial.html](https://satijalab.org/seurat/articles/pbmc3k_tutorial.html) (accessed June 13, 2025).
6. Blondel, V.D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E., Fast Unfolding of Communities in Large Networks, *J. Stat. Mech.: Theory Exp.*, 2008, vol. 10, p. P10008 (12 pp). <https://doi.org/10.1088/1742-5468/2008/10/P10008>.
7. Borovkov, A.A., *Matematicheskaya statistika. Dopolnitel'nye glavy* (Mathematical Statistics: Advanced Chapters), Moscow: Nauka, 1984.
8. Hájek, J. and Šidák, Z., *Theory of Rank Tests*, New York: Academic, 1967.
9. Chibisov, D.M., *Lektsii po asimptoticheskoi teorii rangovykh kriteriev* (Lectures on the Asymptotic Theory of Rank Tests), Lekts. Kursy NOC, vol. 14, Moscow: Steklov Math. Inst., 2009.
10. Benjamini, Y. and Hochberg, Y., Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing, *J. Roy. Statist. Soc. Ser. B*, 1995, vol. 57, no. 1, pp. 289–300. <https://doi.org/10.1111/j.2517-6161.1995.tb02031.x>
11. Härdle, W.K., Simar, L., and Fengler, M.R., *Applied Multivariate Statistical Analysis*, Berlin: Springer, 2024, 6th ed. <https://doi.org/10.1007/978-3-031-63833-6>

**Publisher's Note.** Pleiades Publishing remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

AI tools may have been used in the translation or editing of this article.