NEW ALGORITHMS FOR RESTORING DNA MATRIX AND THEIR STATISTICAL STUDY

Boris Melnikov[*]
Department of Computational Mathematics and Cybernetics,
Shenzhen MSU – BIT University, China
bormel@mail.ru

Article history:
Received 25.10.2024, Accepted 24.12.2024

Abstract

In the proposed paper, we continue to consider various heuristics for reconstructing distance matrices between DNA sequences; as before, we prefer to consider mitochondrial DNA. In the paper, we apply new heuristics. First, every time we receive a new matrix value, we return to the previously restored elements, select the one that gives the maximum value of badness, and try to improve it based on the newly obtained elements. Secondly, when choosing the final value of an element, we select for the first approximation several values close to the optimal one (in practice, there are up to 10 of them), after which we use correlation analysis to select the closest one.

We conducted computational experiments on the mitochondrial DNA of all 32 genera of monkeys. At the same time, we restored not one matrix, but 40 matrices: 10 times obtained for each of 4 different sparsity variants. We average the obtained calculation results in several different ways, in particular, by discarding the two smallest and two largest values, as well as using variants of the risk function.

The results obtained indicate that in order to obtain a matrix for all types of monkeys in the future, it is desirable to have about 10–12% of the data: simplifying the situation somewhat, we can say that 7–8% gives less adequate recovery results, and, vice versa, 15% and more do not improve the value of badness (compared to 10%). At the same time, obtaining exactly such a full matrix by the algorithm for calculating DNA distance should take about 20–25 days of computer operation.

Key words

DNA chains, distance matrix, optimization problem, restoring algorithm, greedy algorithm, heuristics, risk function, rank correlation.

[*]Corresponding author.

1 Introduction and motivation

First, we should note that the full long title of the proposed paper could be as follows: “Some new improved algorithms for restoring DNA matrix and their statistical study using the example of mitochondrial DNA of monkeys of various genera”.

In the paper, we continue to consider various heuristics for reconstructing distance matrices between DNA sequences. As before, we prefer to consider mitochondrial DNA, aiming in the near future to describe acceptable algorithms for restoring such a matrix for all species of monkeys: there are several hundred of these species, and, according to our calculations, obtaining a complete similar matrix using the Needleman-Wunsch method will take about a year of work on a conventional modern computer.

Therefore, to determine the distance between genomes, we need heuristic algorithms, and, if possible, they do not require too much time. In previous publications, we have repeatedly noted that there are different algorithms for such actions, but their obvious disadvantage is to obtain different results when using different heuristic algorithms, applied to calculating the distance between the same pair of DNA strings; however, sometimes the results vary greatly. As an auxiliary task, there is a problem of evaluating the quality of the used metrics (distances), and based on the results obtained in solving this problem, there is possible to draw conclusions about the applicability of a specific distance calculation algorithm to various applied research. A possible approach to determining the quality assessment of metrics has also been given in our previous publications and is briefly discussed in the Section 2.

Due to the very long calculation time of the complete distance matrix noted above, we are trying to apply different heuristics to restore it, calculating only some of its elements. In our first works on restoration, we preferred variants of the branches and boundaries algorithm, however, for square matrices of about 500 in size (by the number of species), this method will take a very long time and therefore is hardly acceptable. For this reason, as well as due to the successful operation of complex heuristics, in subsequent works we switched to such heuristics that do not use the method of branches and boundaries.

Thus, the main focus should be on the use of greedy heuristics. Speaking of them, we are in our latest publication [Abramyan, Melnikov, and Zhang, 2023] we considered a heuristic that took into account the order of the restored values of the matrix; and the earlier values were taken into account with greater weight. In the current paper, we abandoned it (because it takes a long time, although it gives good results) and applied two new heuristics. First, every time we receive a new value of the matrix, we return to the previously restored elements, select the one that gives the maximum value of badness, and try to improve it based on the newly obtained elements. Secondly, when choosing the final value of an element for the first approximation, we select several values close to the optimal one (in practice, there are up to 10 of them), after which we use correlation analysis to select the closest ones; we assume that the optimal value should be among them.

We conducted computational experiments on the mitochondrial DNA of all 32 genera of monkeys. At the same time, we restored not one matrix only, but 40 matrices: 10 times obtained for each of 4 different sparsity variants. The obtained calculation results are averaged in several different ways, in particular, by discarding the two smallest and two largest values, as well as using variants of the risk function, for more information, see the Section 4.

The results obtained indicate that in order to obtain a matrix for all types of monkeys in the future, it is desirable to have about $10 - 12\,\%$ of the data: simplifying the situation somewhat, we can say that $7 - 8\,\%$ gives less adequate recovery results, and, vice versa, $15\,\%$ and more do not improve the value of badness (compared to $10\,\%$). At the same time, obtaining exactly such a full matrix by the algorithm for calculating DNA distance should take about $20 - 25$ days of computer operation.

At the end of the introduction, we note the following. Like we said in [Abramyan, Melnikov, and Zhang, 2023], the connection of some algorithms for studying DNA chains with various tasks of experimental physics was demonstrated in some of our previous publications; to them, it is worth adding [Sergeenko, Granichin, and Yakunina, 2020], which, among other things, examines the connection with Hamiltonian cycles, and, consequently, with the traveling salesman problem (TSP), also considered in some publications of the author of this paper, see [Melnikov, Zhang, and Chaikovskii, 2022] and some others.

2 Preliminaries

Thus, for a chosen algorithm for calculating the distances between two DNA sequences, we have a distance matrix computed using this algorithm. We consider the set of species for which the matrix is being constructed as predetermined, and thus we assume that a certain set of genomes is established. The elements of such a matrix, i.e., the distances between genomes, are used by biologists in both scientific and popular science publications.

Let us note that the methods and algorithms we propose are applicable for any initial algorithm used to determine the distances between genomes, for any particular part of the genomes, and for any set of species.

Now, we provide a brief description of some aspects of our previous papers ([Melnikov, Pivneva, and Trifonov, 2017; Melnikov, Zhang, and Chaikovskii, 2022; Abramyan, Melnikov, and Zhang, 2023] etc.) that are necessary for this study.

In many previous publications, we have presented the following example. Let us consider three species: human, chimpanzee, and bonobo. According to biologists:

- the ancestors of both apes and humans diverged approximately 7 000 000 years ago;
- the ancestors of chimpanzees and bonobos diverged approximately 2 500 000 years ago.

At the same time, *the exact values are not particularly important*; what matters is that *the triangles* formed by the corresponding distances between the three species should ideally be *acute-angled isosceles*. Moreover, this condition must hold *for any three species*.

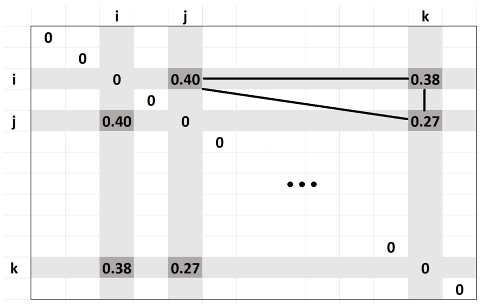

Figure 1. The distance matrix and one of its triangles

Thus, the distance matrix derived from algorithms (such as Needleman–Wunsch) forms specific triangles representing the distances between genomes. The total number of such triangles is

$$\frac{n \cdot (n-1) \cdot (n-2)}{6},$$

which equals 4960 pieces for a matrix with dimension $n = 32$.

For each such triangle, . . .

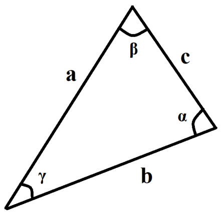

Figure 2. A triangle and notation for its sides and angles

. . . we determine the numerical value of the so-called "badness". In previous calculations, we considered several variants of badness, examples of which are shown in the following Table 1 for specific triangles. Below, we use the badness value labeled as "Bad. (0)", which we consider the most appropriate.

Table 1. Some triangles and their badness

Sides	Angles	Bad. (0)	Bad. (5)
a, b, c	α, β, γ	$(\alpha - \beta)/\gamma$	$(a - b)/c$
1 1 1	60 60 60	0	0
5 5 4	66 66 47	0	0
42 41 28	72 68 39	0.10	0.04
19 18 17	66 60 55	0.11	0.06
40 38 27	74 66 40	0.19	0.07
10 9 8	72 59 50	0.26	0.13
6 5 5	74 53 53	0.39	0.20
13 12 5	90 67 23	1.00	0.20
5 4 3	90 53 37	1.00	0.33
12 6 5	–	1.09	
20 6 5	–	1.81	

Some comments.

[1] The angles are rounded to the nearest degree, which may result in the sum not being exactly 180.

[2] $a \geqslant b \geqslant c$, $\alpha \geqslant \beta \geqslant \gamma$.

[3] Other columns (i.e., Bad. (1) . . . Bad. (4)), that sometimes were used in our previous papers and take into account the badness on sides and corners, are not shown here.

[4] The triangles in the table are ordered in ascending order of their corresponding Bad. (0) values.

[5] To compare with the average values, we often use triangles whose sides make up an arithmetic progression with a difference of 1; some of such triangles are given in the table.

[6] Some of the triangles shown in the table are shown in the following Fig. 3; the proportions are observed.

Thus, each triangle has a badness value ranging from 0 to 1 (or from 1 to 2 if the three points do not form a triangle). The total badness value is typically considered as the sum of the badness values for all triangles.

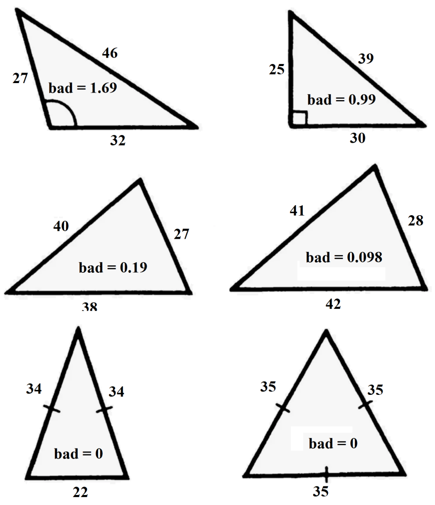

Figure 3. A triangle and notation for its sides and angles

Based on such matrices, we can calculate the total badness of all triangles and assert that algorithms with lower badness values are better than those with higher values. However, this particular analysis is beyond the scope of this paper and has been the subject of previous work.

3 On the data organization in the data archive

The archive of the data used for computational experiments can be found in a convenient format on the following cloud server: `https://disk.yandex.ru/d/2zZNVPmlsZi4RQ`

As already noted, unlike previous publications, we consider matrices of a slightly larger size; this is a size of 32×32 instead of 28×28 in previous publications. This is due to the fact that recently we have managed to find the mitochondrial DNA of 4 more species of monkeys that do not belong to any of the previously considered genera; note that biologists describe 34 genera of monkeys in total, so we consider almost all genera. However, of course, the total number of monkey species (according to various classifications, from 525 to more than 800, [Cibelli et al., 2014; Lennarz and Lane, 2013] etc.) is still far away, and, in addition, building a complete table of the distances between the mitochondrial DNA of all monkey species is one of the immediate goals of the work; at the same time, we hope that more successful algorithms for restoring small matrices will also be more successful in the case of very large dimensions.

Now we shall provide the list of species corresponding to the numbers used in the archive matrices; these species are given in the following Table 2.

Table 2. Species of monkeys corresponding to the data of the matrix archive

No.	Species
1	Allenopithecus nigroviridis
2	Ateles belzebuth
3	Brachyteles arachnoides
4	Cacajao calvus
5	Callimico goeldii
6	Callithrix jacchus
7	Carlito syrichta
8	Cebuella pygmaea
9	Cephalopachus bancanus
10	Cercocebus atys
11	Cercopithecus albogularis
12	Chlorocebus sabaeus
13	Colobus angolensis
14	Erythrocebus patas
15	Galago moholi
16	Gorilla gorilla
17	Lagothrix lagotricha
18	Leontopithecus rosalia
19	Macaca fascicularis
20	Macaca fuscata
21	Mandrillus leucophaeus
22	Nasalis larvatus
23	Nycticebus coucang
24	Papio anubis
25	Presbytis melalophos
26	Pygathrix nemaeus
27	Rhinopithecus roxellana
28	Saguinus oedipus
29	Saimiri boliviensis
30	Semnopithecus entellus
31	Tarsius dentatus
32	Theropithecus gelada

Thus, the size of the matrices in question is 32×32 instead of 28×28 in previous publications. However, the following is much more important. Previously, we restored only one sparse matrix in all publications (some values were forcibly removed from the original matrix, while about 15 % of the elements remained in the matrix), and now we consider 10 matrices for each of several different sparsity values (exactly, 30 %, 20 %, 12 %, and 8 %).

Let us describe the archive of matrices. The initial table is `ta00.txt`. In all the archive matrices, the deleted values are replaced by -1. Titles of matrices are the following:

- from `ta10.txt` to `ta19.txt` correspond to a sparsity of 12 (that is, approximately 12 % of the elements remain in the matrix);
- from `ta20.txt` to `ta29.txt` correspond to a sparsity of 20;
- from `ta30.txt` to `ta39.txt` correspond to a sparsity of 30;
- from `ta40.txt` to `ta49.txt` correspond to a sparsity of 8.

4 The brief description of the algorithms used and the basis results of computational experiments

Once again, we note that the approach proposed in the paper, the methods and algorithms, are possible for any options for the development of tasks:

- for any initial algorithm used to determine the distances between genomes, see [Levenshtein, 1966; Needleman and Wunsch, 1970; Munekawa, Ino, and Hagihara, 2008; Polavarapu et al., 2011] etc.;
- for any particular part of the genome; in addition to mitochondrial DNA, for mammals (including human) in such a situation, the tail on the Y chromosome or the main histocompatibility complex are often considered; see [Tian and Li, 2024; Miao et al., 2025] etc.;
- and for any set of species.

In the simple DNA matrix restoring method we developed earlier [Melnikov, Zhang, and Chaikovskii, 2022; Abramyan, Melnikov, and Zhang, 2023], unknown elements of the matrix are considered and determined “from the left to the right and from the top to the bottom” (in such sequence only), and the recovery itself occurs as a result of several computational passes. On each of the passes, different estimates are obtained for some of the still unfilled (i.e., unknown) elements of the matrix; these estimates are specially averaged, and the result of averaging is taken as the value of the unknown element.

The use of the described method to fill in the matrix of distances between DNA sequences allows, first of all, to significantly reduce the time for filling it, and the deviation of the matrix elements obtained in this way from the values calculated, for example, using the Needleman – Wunsch algorithm averaged less than 2 %, and about 10 % in the worst case.

Thus, the following Table 3 shows the results corresponding to the application of *the previously presented algorithms to the new values*. We repeat that now we have restored not the only matrix, but 10 matrices for each of the 4 sparsity values.

Table 3. Applying old algorithms for new data

Old	δ		σ	
100 %	0.2223		0	
30 %	0.2223	0.2104	0.0682	0.0683
	0.1975	0.2179	0.0759	0.0769
20 %	0.1998	0.1661	0.0862	0.0782
	0.1538	0.1813	0.0651	0.0679
12 %	0.1176	0.1335	0.1021	0.1101
	0.1247	0.1257	0.0269	0.0913
8 %	0.1215	0.1328	0.1073	0.1133
	0.1469	0.1459	0.1077	0.1152

Like the previous paper,

- δ is the average badness of the restoring matrices; as we already said, we used Bad. (0) value;
- σ is the discrepancy, i.e., for $n \times n$ matrix,

$$\sigma = \frac{\sqrt{\sum\limits_{\substack{1 \leqslant i \leqslant n-1 \\ i+1 \leqslant j \leqslant n}} (a_{ij} - \overline{a_{ij}})^2}}{(n \cdot (n-1))/2},$$

  where a_{ij} and $\overline{a_{ij}}$ are the corresponding elements of the original and reconstructed matrices, respectively.

It is clear that the σ value for large matrices can be calculated very rarely, and for small matrices often; it, like δ, is used to check the quality of the recovery algorithm.

We can see, each cell in the table contains not one value, but 4 values, usually approximately equal to each other; we shall provide detailed information about *the different averaging* options *in the next section*: this may be of interest for deeper research, which we have not conducted in this paper.

Let us turn to the consideration of substantially new calculations. In the current paper, as already mentioned in the introduction, we apply new heuristics, primarily the following two.

First, every time we receive a new matrix value, we return to the previously restored elements, select the one that gives the maximum value of badness, and try to improve it based on the newly obtained elements. This value of badness (for some specific element under consideration, and not set initially, but filled in during our calculations) is calculated for all pairs of elements already present in the matrix; and this element in question with this pair forms a triangle for which badness is calculated.

Secondly, when choosing the final value of an element for the first approximation, we select several values close to the optimal one (in practice, there are up to 10 of them), after which we use correlation analysis to select the closest ones; this is done as follows. In the examples of algorithms and programs given in our recent publications, several "preliminary" possible values for the generated element were explicitly indicated; Namely, it was primarily the value found by a very fast and very greedy algorithm, as well as its product by some predefined coefficients, and these coefficients were usually (but not always) selected from a certain R to $1/R$ with some selected negative steps, and R was often chosen to be from 4.0 to 5.0 (therefore, for example, this coefficient could be in the segment from 0.25 to 4.0).

While doing this, we discovered the following fact. Very often in the real calculations, the values for different variants of the coefficients almost coincided (which, apparently, can be explained by the small dependence of this particular new element on those already constructed earlier), as a result of which the choice of the final value could in principle be ambiguous. Note in this regard that we have observed a similar fact when using the branch and bound method, not only in the problem discussed in this article, but also in many other discrete optimization problems. As a result, we propose to describe a similar possible heuristic for a whole class of problems in the future.

The calculation of unknown elements of an incompletely filled matrix based on the use of the value of the badness indicator is based on the calculation that it should ideally be zero, and determining the elements "from left to right and from top to bottom", i.e., in a strictly defined sequence, can lead to the fact that for previously defined triangles, the value of the badness indicator is significantly it will increase. One of the forms of smoothing out such a situation was the use of a risk function that performs a special averaging, namely, the use of risk functions [Melnikov, 2001], which we shall discuss in the next section in a completely different context.

Thus, as a result of the calculations, we got the following Table 4 for the new heuristics (and, certainly, for *the same* new data).

All relevant values have been improved compared to the previous table 3. In each of the cells, the first (upper-left) value is obtained by simply averaging all 10 values, and the second (upper-right) value is obtained by averaging 6 values: not counting the 2 smallest and 2 largest (the so-called "median averaging"). An explanation of the 2 remaining averaging options (also for 10 experiments in each cell of the table) will be given in the next section.

Table 4. Applying new algorithms for new data

New	δ		σ	
100 %	0.2223		0	
30 %	0.1319	0.1258	0.0318	0.0309
	0.1366	0.1379	0.0327	0.0328
20 %	0.1167	0.1129	0.0387	0.0375
	0.1204	0.1204	0.0400	0.0401
12 %	0.0857	0.0868	0.0475	0.0474
	0.0890	0.0899	0.0484	0.0488
8 %	0.0848	0.0857	0.0573	0.0533
	0.0897	0.0902	0.0608	0.0609

5 The auxiliary algorithms and the interpretation of the results

Let us note right away, that the detailed title of this section could be as follows: "The auxiliary algorithms (the risk functions and the rank correlation) and the interpretation of the computational results".

Thus, let us move on to a brief description of averaging using the risk function. Let us repeat that we apply risk functions in two cases:

- for the final choice of the current calculated value: for the algorithms briefly described in the previous section;
- and for the option of averaging various results; such special averaging may be more adequate than the usual arithmetic average.

Using risk functions is just averaging with specially selected weighting coefficients; for more information, see [Melnikov, 2001] etc. So, for values

$$x_1, x_2, \ldots, x_n$$

and functions f, the risk-averaged value is assumed to be equal to

$$\frac{\sum\limits_{1\leqslant i\leqslant n} f(x_i)\cdot x_i}{\sum\limits_{1\leqslant i\leqslant n} f(x_i)},$$

as can be seen in both further figures, i.e., Fig. 4 and 5: the average value in both situations is the arithmetic mean of the lengths of the vertical segments corresponding to some various values of the considered variable.

Here is a very brief information about the specific variants of the risk function f used in the previous formula: first, we talk about the so-called *static risk functions*. Frequently used specific examples of such functions are shown in Fig. 4 and 5. Note that in all the cases necessary for the paper (that is, two use cases and two different functions), we consider large values to be "bad" and small values to be "good" ones; as a result, both functions are increasing, decreasing risk functions are constructed in exactly the same way.

Since normalization can be assumed to have been performed, the lower value (the abscissa is indicated by min in the figures) is 0.5 in both cases, and the higher value (the abscissa is indicated by max) is 1. We need sufficient detail in which only convexity or concavity is important, then in both cases we construct parabolas, and one of the points, min or max, is the vertex of this parabola; the coefficients for the parabola are easily selected according to these figures.

Specifically, Fig. 4 shows such a risk function in which *large* values are considered *"bad"*, and at the same time, *the worst case* is more important for the algorithm using these functions.

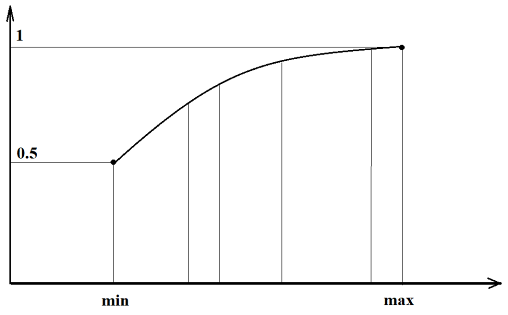

Figure 4. An example of risk function; large values are "bad"; the worst case is more important.

Vice versa, Fig. 5 shows such a risk function in which *large* values are also considered *"bad"*, but, at the same time, *the average case* is more important for the algorithm using these functions.

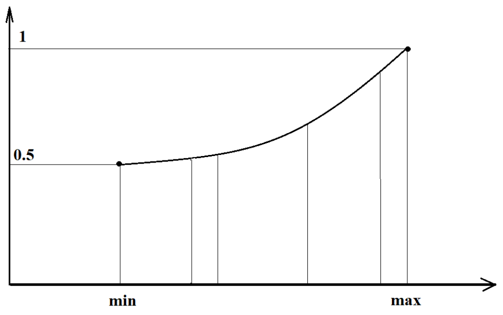

Figure 5. An example of risk function; large values are "bad"; the average case is more important.

Thus, it is these two averaging options, with these specific risk functions, that are applied to all the cells in both tables of experimental results: the lower left and right values, respectively.

So, these were static risk functions. More complex are the dynamic risk functions that generalize them, which we first applied to programming nondeterministic games, [Melnikov, 2001] etc.[1] We shall not write in detail about dynamic risk functions (their application to the material of this article is expected in the future), we shall only say the following. First, they try to automatically adjust to the situation, that is, depending on the values of the variable, choose the direction of the convexity of functions. Secondly, after the intellectual games, we applied these functions in discrete optimization problems, and this is the option that is supposed to be applied in the future for the problems considered here.

In conclusion of the section, let us talk about the use of rank correlation algorithms in the tasks under consideration. It was said above that we pre-select several adequate values close to the one that we previously considered optimal (as we noted, there are up to 10 of them). We apply these values to the sequences of badness of all new triangles formed (the estimate of the total number of triangles in the matrix was given above); we believe (this is exactly the heuristic of using correlation algorithms in this problem) that the value that will be definitively declared optimal has the maximum sum of the correlation coefficients with other values. Note that we used the version described in [Melnikov and Lysak, 2024] as an algorithm for calculating rank correlation. The results obtained do indeed indicate a slight improvement in the final values, compared with the variants when auxiliary algorithms for applying rank correlation were not used.

As a small analysis of the obtained calculation results, let us say the following.

- Firstly, any of the 4 averaging options gives approximately the same improvement, that is, simplifying the situation, the distribution of values is “very simple”, and therefore we can use the simplest averaging.
- Secondly, the final improvement is very big: it is significantly higher than the results obtained in our previous paper [Abramyan, Melnikov, and Zhang, 2023] in comparison with the results preceding that paper.
- Thirdly, it seems that it is really necessary to improve greedy heuristics in this problem, rather than implement variants of the branch and bound method; however, such a statement should also be definitively confirmed by further computational experiments.

[1]The computer program for playing backgammon, implemented under the guidance of the author of this paper, successfully participated in the ICGA World Intellectual Computer Games Championship. However, several strong programs were not presented at the championship (the authors from Europe did not come to Southeast Asia), so the author did not consider himself entitled to be called the winner of this championship.

6 Conclusion

In general, our research on DNA analysis can be divided into the following interconnected areas.

- Evaluating the relative quality of algorithms for determining the distance between two DNA sequences.
- The *task of restoring incomplete distance matrices*; as noted earlier, some of our previous work focuses on this problem, and *we continue this line of inquiry in this paper*.
- Improving the initial algorithms for determining distances between sequences based on the results obtained in the previous steps.
- Addressing related problems in the statistical analysis of the results obtained.

We also note the similarity between the incomplete distance matrix problem discussed here and the traveling salesman problem, particularly its pseudo-geometric version. This connection is reflected in many recent articles by the author, among which we note the ones already quoted above [Melnikov, Pivneva, and Trifonov, 2017; Melnikov, Zhang, and Chaikovskii, 2022; Abramyan, Melnikov, and Zhang, 2023]. In both types of problems:

- either one can use the branch-and-bound method, in which the main task of algorithm optimization is the development of auxiliary heuristics for this method;
- or alternatively, the focus can be on developing greedy heuristics, which can be considered as the main subject of this paper.

Let us repeat with a few additions what was said above. The results obtained indicate that in order to obtain a matrix for all types of monkeys in the future, it is desirable to have about $10-12\,\%$ of the data: simplifying the situation somewhat, we can say that $7-8\,\%$ gives less adequate recovery results. I.e., δ value it turns out to be too small, which in the limit, i.e. with a very small percentage of known matrix cells, can give a zero value; this may correspond to equal values of the matrix elements, which, of course, is not interesting for the problems under consideration. Vice versa, $15\,\%$ and more do not improve the value of badness (compared to $10\,\%$). At the same time, obtaining exactly such a full matrix by the algorithm for calculating DNA distance should take about $20-25$ days of computer operation. *What is said above in this paragraph can be considered the formulation of the most important direction for further research.*

At the end of the paper, we shall provide a few more links related to various studies of distance matrices between genomes. The use of a rather original approach to the study of the genomes of various species of monkeys is described in the most recent studies, [Young and Gilles, 2025]. Among the earlier works, we note [Ballester and Richards, 2007; Bodenhofer et al., 2015].

Acknowledgement

This work was supported by the grant of the Scientific program of Chinese universities “Higher Education Stability Support Program” (chapter “Shenzhen 2022 – Science, Technology and Innovation Commission of Shenzhen Municipality”).

The author also expresses gratitude to post-graduate students Li Jiamian and Mu Jingyuan (Shenzhen MSU – BIT University) who received values of the initial table `ta00.txt` of the archive.

References

Abramyan M., Melnikov B., and Zhang Y. (2023) Some more on restoring distance matrices between DNA chains: reliability coefficients. *Cybernetics and Physics*, **12**(4), pp. 237–251.

Ballester P.J. and Richards W.G. (2007) Ultrafast shape recognition to search compound databases for similar molecular shapes. *Journal Comput. Chem.*, **28**. DOI: 10.1002/jcc.20681.

Bodenhofer U., Bonatesta E., Horejs-Kainrath C., and Hochreiter S. (2015) Msa: an R package for multiple sequence alignment. *Bioinformatics*, **31**(24), pp. 3997–3999. DOI: 10.1093/bioinformatics/btv494.

Cibelli J., Wilmut I., Jaenisch R. et al. (Eds) (2014). *Principles of Cloning*. Academic Press, New York.

Lennarz J. and Lane M. (Eds) (2013). *Encyclopedia of Biological Chemistry*. Elsevier Publ., Amsterdam.

Levenshtein V. (1966) Binary codes capable of correcting. Deletions, insertions, and reversals. *Soviet Physics Doklady*, **10**, pp. 707–710.

Melnikov B. (2001) Heuristics in programming of nondeterministic games. *Programming and Computer Software*, **27**(5), pp. 277–288

Melnikov B., Pivneva S., and Trifonov M. (2017). Various algorithms, calculating distances of DNA sequences, and some computational recommendations for use such algorithms. *CEUR Workshop Proceedings*, **1902**, pp. 43–50.

Melnikov B., Zhang Y., and Chaikovskii D. (2022). An inverse problem for matrix processing: an improved algorithm for restoring the distance matrix for DNA chains. *Cybernetics and Physics*, **11**(4), pp. 217–226.

Melnikov B. and Lysak T. (2024). On some algorithms for comparing models of femtosecond laser radiation propagation in a medium with gold nanorods. *Cybernetics and Physics*, **13**(3), pp. 261–267.

Miao L., Liu Sh., Pan K.-P., Jiao R.-L., Zhang Q., Xu T.-Y., Tong Sh.-Y., Kang K.-L., Zhao J., Zhang Ch., Wang Kai-Di, and Ji A.-Q. (2025). Improved understanding of sequence polymorphisms at 42 Y chromosome short tandem repeats for the Chinese Han population. *Cybernetics and Physics*, **75**, no. 103181, DOI: 10.1016/j.fsigen.2024.103181.

Munekawa Y., Ino F., and Hagihara K. (2008). An inverse problem for matrix processing: an improved algorithm for restoring the distance matrix for DNA chains. *8th IEEE International Conference on BioInformatics and BioEngineering, BIBE-2008*, DOI: 10.1109/BIBE.2008.4696721.

Needleman S. and Wunsch Ch. (1970). A general method is applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, **48**(3), pp. 443–453.

Polavarapu N., Arora G., Mittal V., and McDonald J. (2011). Characterization and potential functional significance of human-chimpanzee large INDEL variation. *Mob DNA*, Oct 25:2:13.

Sergeenko A., Granichin O., and Yakunina M. (2020). Hamiltonian path problem: the time consumption comparison of DNA computing and branch and boundary method. *Cybernetics and Physics*, **9**(1), pp. 121–127.

Tian W. and Li L. (2024). Full genomic sequence shows HLA-B*40:186:02 differs from HLA-B*40:186:01 by a cytosine substitution in exon 2. *HLA (Tissue Antigens)*, Published by John Wiley & Sons Ltd, `https://www.scopus.com/sourceid/21100775663`.

Young S. and Gilles J. (2025) Use of 3D chaos game representation to quantify DNA sequence similarity with applications for hierarchical clustering. *Journal of Theoretical Biology*, **5967**(111972). DOI: 10.1016/j.jtbi.2024.111972.