Artificial Intelligence and Human-Computer Interaction
Y. Ye and P. Siarry (Eds.)

doi:10.3233/FAIA240134

On the Application of Heuristics of the TSP for the Task of Restoring the DNA Matrix

Boris MELNIKOV [1] and Dmitrii CHAIKOVSKII
Shenzhen MSU–BIT University, Shenzhen, China
ORCiD ID: Boris Melnikov https://orcid.org/0000-0002-6765-6800
ORCiD ID: Dmitrii Chaikovskii https://orcid.org/0000-0002-0063-106X

Abstract. The traveling salesman problem (TSP) is a well-known optimization problem that seeks to find the shortest possible route that visits a set of points and returns to the starting point. In this paper, we apply some heuristics of the TSP for the task of restoring the DNA matrix. This restoration problem is often considered in biocybernetics. For it, we must recover the matrix of distances between DNA sequences, if not all the elements of the matrix under consideration are known at the input. We consider the possibility of using this method on the testing of distance calculation algorithms between a pair of DNA's to restore the partially filled matrix.

Keywords. Optimization Problems, DNA matrix, partially filled matrix, traveling salesman problem, heuristic algorithms.

1. Introduction

This paper is a continuation of our previous works [1-3]. We continue the consideration of the application multiheuristic approach to discrete optimization. In it, we add several auxiliary heuristic algorithms to the "usual" variants of the branch-and-bound algorithm, which are almost equally implemented in different subject areas. It is important to note that the greatest effect of these auxiliary heuristics is usually given by their simultaneous application, i.e. in the complex.

There is often necessary to calculate the distances between sequences of different nature. Fuzzy search algorithms are considered to be the basic systems for controlling spelling errors and full-format information search systems similar to Google or Yandex. The fuzzy search system acts as a very useful option for all search engines. But at the same time, its full-format execution will be significantly more complicated than performing a simple search using an exact match algorithm. The problem of fuzzy search is formulated as follows: there is a given word and we need to search in a text or dictionary of the given size, all words that match the given word (or start with the given word), given some possible differences.

There are various such algorithms for genomes, but the obvious disadvantage in calculating the distance between the same pair of DNA strings is obtaining different

[1] Corresponding Author: Boris Melnikov, Faculty of Computational Mathematics and Cybernetics, Shenzhen MSU–BIT University, China. E-mail: bormel@mail.ru.

results when using different algorithms. In the cited papers, we have considered some of these algorithms.

The other problem considered in biocybernetics is the recovering the matrix of distances between DNA sequences, when not all elements of the considered matrix are known at the input of the algorithm. The algorithms calculating the exact value of the distance between two sequences is not applicable, and to estimate the distance between such chains, one has to use heuristic algorithms, which give approximate results. At the same time, even such heuristic algorithms require large time costs: for example, to build a matrix of the order of 50×50, in which the distances calculated by the Needleman – Wunsch algorithm are recorded, it takes about 28 hours (with a processor clock frequency of the order of about 3 GHz).

Thus, to determine the distance between genomes the use of developed and researched was considered previously, where we used methods of comparative evaluation of algorithms calculate the distance between a pair of DNA chains for reconstructions of the partially filled distance matrix. The matrix restoration is considered as a result of multiple computer gateways. Assessment the unknown elements of the matrix are organized in a special way.

Continuing to improve the algorithms, we consider the use of the branch-and-bound method in it. To do this, for some known sequence of unfilled elements, we apply the algorithms we considered before, but now we choose the special sequences of elements. In our interpretation of the branch-and-bound method, all possible sequences of unknown elements of the upper triangular part of the matrix are taken as the set of admissible solutions. In each current subtask, any of the blank elements of the matrix is taken as the separating element, and the sum of the badness values for all triangles that have already been formed by the time this subtask is considered is taken as the bound. Thus, the definition of elements of an incompletely filled matrix occurs in such a sequence that the final badness value for all triangles is selected using greedy heuristics that fits completely into the framework of the classical variants of the description of the branch-and-bound method.

As a result of applying such an algorithm, we get the lowest possible badness (in the case of a completed version of the branch-and-bound method), or close to optimal ones. In our computational experiments, the running time of the algorithm practically coincides with the time of the algorithm considered before (it exceeds it by no more than 10%), and the badness value usually decreases by 20-40% from the initial value. Thus, we are able to quickly and efficiently restore the DNA matrix, often even if it is filled less than 40%. In this paper, we shall consider the following questions:

- a brief statistical study of the problem of DNA matrix reconstruction of small dimensions (Section 2);
- the mathematical justification of the correctness of the constructions being made (Section 3);
- the consideration of the application example of the method of the branch-and-bound algorithm in the problem of reconstructing a DNA template (Section 4).

2. A brief statistical study of the problem of DNA matrix reconstruction of small dimensions

In this section, we consider a problem of reconstruction of the matrix of distances between DNA sequences. This problem belongs to the field of biocybernetics was previously described by the first author in [4-5]. The task is to restore the elements of the distance matrix between DNA sequences. As a rule, about 50% of the matrix elements are known. To solve this problem, it is advisable to use the so-called anytime-algorithm [6], which would allow to track the gradual recovery of the elements of the matrix of distances. The paper [7] investigates algorithms for solving the problem of restoring a low-rank matrix with an arbitrarily damaged fraction of its elements. This problem can be considered as a reliable version of the classical principal component analysis [8] and occurs in a number of applications, including image processing, web data ranking and bioinformatics' data analysis.

The distance matrix reconstruction algorithm is based on the analysis of all possible triangles in the distance matrix. For each triangle, the value of badness is calculated, the tracking of which is part of the heuristic approach (see [4] for details). However, if we pass through the elements of the matrix "left to right" and "top to bottom", we can get unwanted results with a significant increase in badness for previously studied triangles. One approach to work around this problem is to form a sequence of subtasks by modifying the branch-and-bound algorithm [5]. In this case, the badness matrix value will act as the boundary value in the classical branch-and-bound algorithm [9-10]. Our goal is to consider various statistical indicators arising from the reconstruction of matrices described in [5] by the modified branch-and-bound algorithm. Below are the results of our statistical study of the DNA matrix of small dimensions for the problem of reconstruction. The results of our computational experiments are given in Table 1 (see some comments below). As in the case of the previous problem (TSP), we answered only one question: how often at least 1 pair of similar matrices occurs when applying the first few steps of the "classical" branch-and-bound algorithm. However, in comparison with the TSP and the previous section, we have replaced the word "identical" with the word "similar": in this case we consider similar matrices in which the sets of empty elements coincide. It is easy to make sure (also, for example, by computational experiments) that the further work of the branch-and-bound algorithm with two similar matrices in the vast majority of cases occur with the same order of selection of not yet filled matrix elements.

Table 1. The number of cases (out of 1000 considered), for which at least one pair was obtained.

	30	40	50	60	70
5	142/143	121/128	87/100		
10		196/401	173/363	166/334	
15			294/592	210/456	332/356

Generation of input data was carried out based on the matrix obtained working with matrices obtained by applying the algorithm of the Needleman – Wunsch [11]. We applied this algorithm to the mDNA chains of different animals taken from the NCBI data Bank [12], and sequenced mDNA chains were taken for one representative of each of the 28 mammalian orders (mammalian classification is chosen according to [13], other classification options were not considered). For this matrix, we randomly, applying a uniform distribution, chose the desired number of its rows / columns (5, 10 or 15), obtained a matrix of smaller dimension, which left the desired percentage of elements (from 30% to 70%). For each pair consisting of the dimension and the percentage of deleted items, we have done 1000 of these generations. Next, we ran the method of the

branch-and-bound algorithm, but, in contrast to the consideration of the TSP in the previous section, the last parameter was not the number of steps of the branch-and-bound algorithm, and the number of resulting subproblems: when we receive 10 (or 30) of the subproblems, we calculate the stop. Also, of course, we stopped the calculations and when we get two similar matrices, and it is the number of such options (out of 1000 possible) and is reflected in Table 1. The format of the submitted data is as follows.

- The lines specify dimension.
- The columns indicate the percentage of deleted items. For example, if the size is news 10, we have only 45 items located above the main diagonal; removing 40% means removing 18 items from them.
- In each of the filled cells-the results of calculations for this case: the number of cases (out of 1000 considered) for which at least one pair was obtained.
- In this case, the first value in the cell is given for calculations that stopped after receiving 10 subtasks, and the second is after receiving 30 subtasks.

3. The mathematical justification of the correctness of the constructions being made

As we said above, we use heuristics previously used to solve the traveling salesman problem. The possibility of their use is due to the fact that for a certain point of the distance matrix, which represents the distance between two species, we consider a triangle that is three distances between three species, and denote the remaining two sides of this triangle as two segments of the traveling salesman problem.

For each such pair of points (x_{s_i}, y_{s_i}) and $(x_{s_{i+1}}, y_{s_{i+1}})$, we can define two cubic functions in the following way.

$X_i(t)$and$Y_i(t)$ for $t \in [t_i, t_{i+1}] \equiv \Omega_i$

$$X_i(t) = a_{x,i} + b_{x,i}(t - t_i) + c_{x,i}(t - t_i)^2 + d_{x,i}(t - t_i)^3,$$
$$Y_i(t) = a_{y,i} + b_{y,i}(t - t_i) + c_{y,i}(t - t_i)^2 + d_{y,i}(t - t_i)^3.$$

The coefficients $a_{x,i}, b_{x,i}, c_{x,i}, d_{x,i}, a_{y,i}, b_{y,i}, c_{y,i}$, and $d_{y,i}$ can be determined by imposing the following conditions.

The cubic functions pass through the points:

$$\begin{aligned} X_i(t_i) &= x_{s_i}, \\ X_i(t_{i+1}) &= x_{s_{i+1}}, \\ Y_i(t_i) &= y_{s_i}, \\ Y_i(t_{i+1}) &= y_{s_{i+1}}. \end{aligned}$$

The first and second derivatives are continuous at the junctions:

$$\begin{aligned} X_i'(t_{i+1}) &= X_{i+1}'(t_{i+1}), \\ X_i''(t_{i+1}) &= X_{i+1}''(t_{i+1}), \\ Y_i'(t_{i+1}) &= Y_{i+1}'(t_{i+1}), \\ Y_i''(t_{i+1}) &= Y_{i+1}''(t_{i+1}). \end{aligned}$$

We denote the sets of piece-wise functions as:

$$X(t) = \begin{cases} X_1(t), \text{ for } t \in [t_1, t_2] \\ X_2(t), \text{ for } t \in [t_2, t_3] \\ \dots, \\ X_N(t), \text{ for } t \in [t_N, t_{N+1}] \end{cases}$$

and

$$Y(t) = \begin{cases} Y_1(t), \text{ for } t \in [t_1, t_2], \\ Y_2(t), \text{ for } t \in [t_2, t_3], \\ \dots, \\ Y_N(t), \text{ for } t \in [t_N, t_{N+1}]. \end{cases}$$

Then we define the functional $\Phi[X, Y]$ as follows:

$$\begin{aligned} \Phi[X,Y] \equiv \; & \frac{1}{N} \cdot \sum_{i=1}^{N} \left(\left(x_{s_i}^{\delta} - X(t_i) \right)^2 + \left(y_{s_i}^{\delta} - Y(t_i) \right)^2 \right) \\ & + \alpha \cdot \left(|X''(t)|_{L^2(\Omega)}^2 + |Y''(t)|_{L^2(\Omega)}^2, \right); \end{aligned} \tag{1}$$

$$\begin{aligned} \Phi[X,Y] \equiv \; & \frac{1}{N} \cdot \sum_{i=1}^{N} \left(\left(x_{s_i}^{\delta} - X(t_i) \right)^2 + \left(y_{s_i}^{\delta} - Y(t_i) \right)^2 \right) \\ & + \alpha \cdot \left(|X''(t)|_{L^2(\Omega)}^2 + |Y''(t)|_{L^2(\Omega)}^2 \right); \end{aligned}$$

here, the domain Ω is the union of all the segments' domains, i.e.,

$$\Omega = \bigcup_{i=1}^{N} \Omega_i$$

Let α be such that the minimizing elements $X_\alpha(t)$ and $Y_\alpha(t)$ of $\Phi[X, Y]$ satisfy:

$$\frac{1}{N} \cdot \sum_{i=1}^{N} \left(\left(x_{s_i}^{\delta} - X_\alpha(t_i) \right)^2 + \left(y_{s_i}^{\delta} - Y_\alpha(t_i) \right)^2 \right) = \delta^2.$$

We find the minimizing elements $X_\alpha(t)$ and $Y_\alpha(t)$ of $\Phi[X, Y]$:

$$(X_\alpha(t), Y_\alpha(t)) = \arg \min_{X,Y} \Phi[X, Y]. \tag{2}$$

These minimizing cubic spline functions $X_\alpha(t)$ and $Y_\alpha(t)$ will provide the best approximation of the true path, given the noisy data and the chosen smoothness parameter α..

4. The consideration of the application example of the branch-and-bound algorithm

Each matrix has a number of characteristics that affect the outcome of the branch-and-bound method. One of these characteristics included in this list is the percentage of deleted items. On Figure 1, there is a selection of the results of the method for the problem discussed in the previous sections, for 5×5 matrices with the removal of 40%, 50% and 60% of the elements of the original matrix.

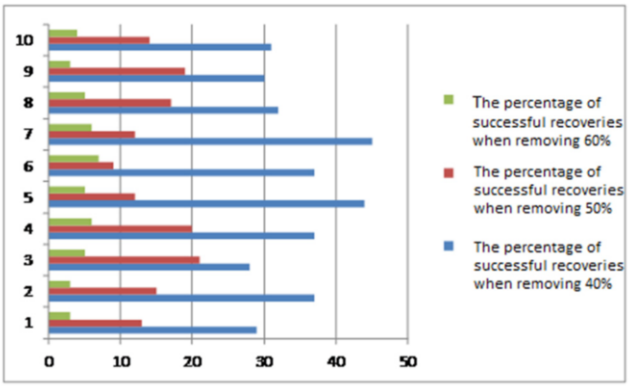

Figure 1. The selection of the results, 5x5.

The following graph (Figure 2) shows the average percentage of successful recoveries based on the percentage of deleted items.

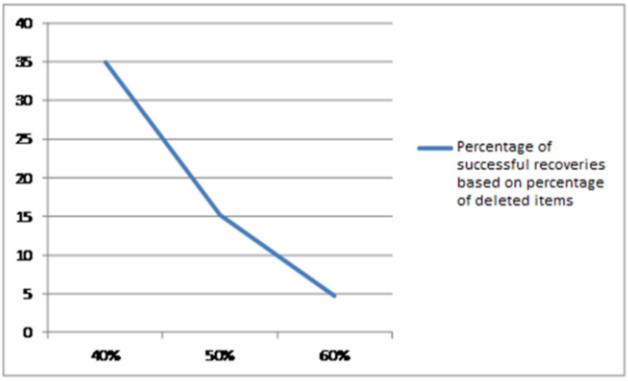

Figure 2. The average percentage of successful recoveries, 5x5.

The results of similar numerical experiments for 10×10 matrices are given on Figure 3.

We also present a graph of the average number of successful recoveries, Figure 4.

Compared to the previous results for 5×5 matrices, it is noticeable that the percentage of successful recoveries has increased significantly. This seems to be due to the fact that there is more room for triangle selection and element recovery during the computational process. With a further increase in the dimension of the matrices, we

observe the tendency of the percentage of successful recoveries to 100 with the specified percentage of deleted elements (40%, 50%, 60%), see Figure 5.

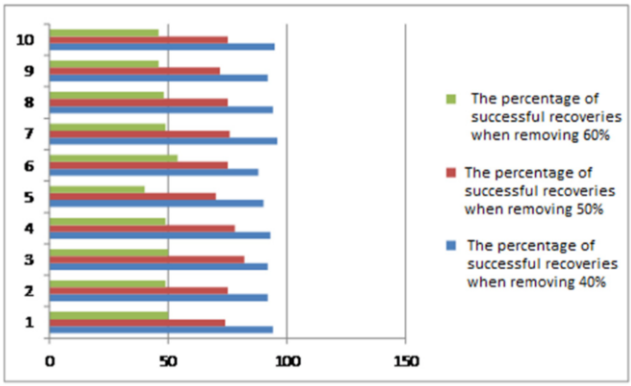

Figure 3. The results of numerical experiments for 10×10 matrices.

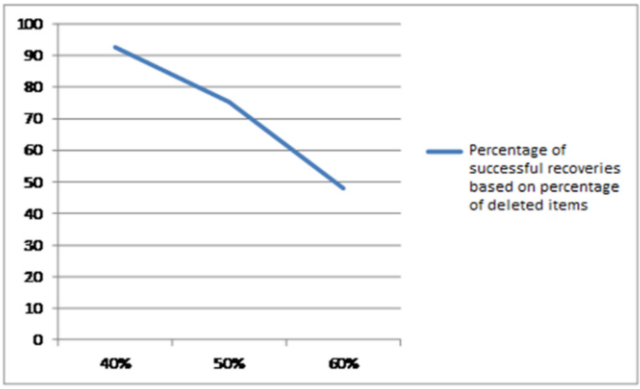

Figure 4. The average percentage of successful recoveries, 10x10.

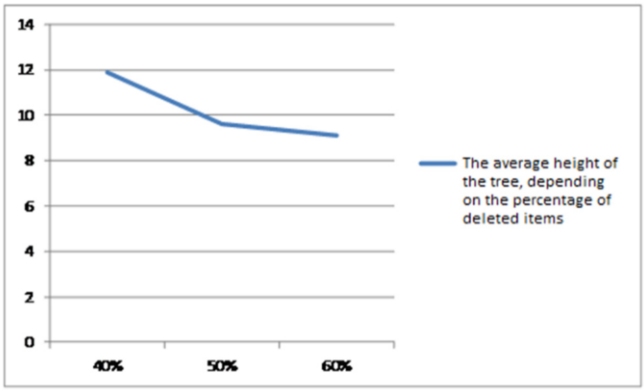

Figure 5. The tendency of the percentage of successful recoveries.

The next interesting characteristic is the height of the decision tree (i.e., the maximum path length from the tree root to the leaf). In the presented graphs it is clearly seen that the average value of the height of the decision tree for the matrix 5×5 is not too varies with the percentage of deleted elements.

It also makes sense to look at the change in the height of the decision tree depending on the dimension of the matrix with a fixed number of deleted elements. The two diagrams below illustrate this comparison, see Figure 6 and Figure 7.

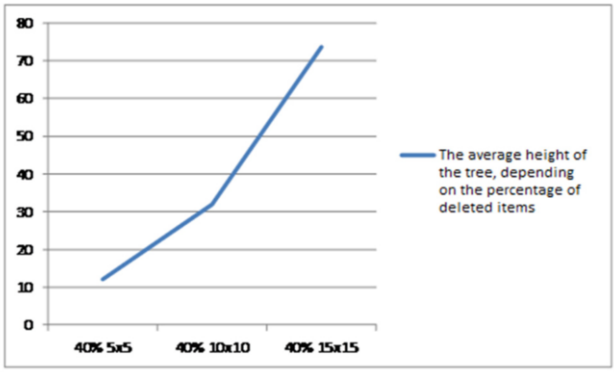

Figure 6. The height of the decision tree, 40% of deleted elements.

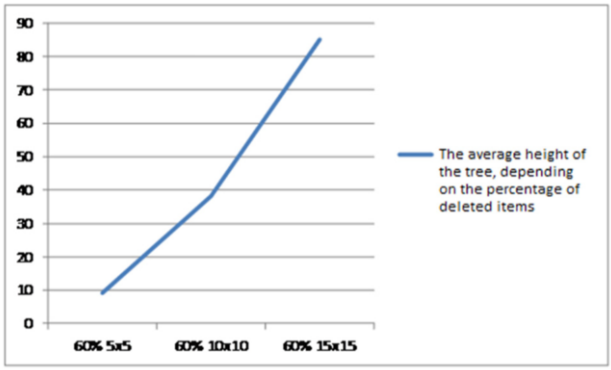

Figure 7. The height of the decision tree, 60% of deleted elements.

5. Conclusion

The statistical regularities obtained by us in this article are actually the probability of a successful situation, which makes it possible not to calculate the separating element once again. According to our calculations, obtained also in the course of computational experiments, the choice of the separating element in some variants of the branch-and-bound algorithm is spent more than 99% of the time of the program. Therefore, the results provide a rationale for the application of clustering situations in the development of algorithms for solving discrete optimization problems using the branch and bound method. This application gives easily observed improvements of the algorithm; it is this

variant, from our point of view, reflects the representativeness of the data in many real problems.

Acknowledgement

The authors were supported by a grant from the scientific program of Chinese universities “Higher Education Stability Support Program” (section “Shenzhen 2022 – Science, Technology and Innovation Commission of Shenzhen Municipality”).

References

[1] Melnikov B, Melnikova E, Pivneva S, Trenina M. An approach to analysis of the similarity of DNA-sequences. CEUR Workshop Proceedings, 2018;2212, p. 63–72.
[2] Melnikov B, Trenina M. On possible methods for solving the problem of reconstructing the matrix of distances between DNA strings. CEUR Workshop Proceedings, 2018;2258, p. 11–20.
[3] Melnikov B, Trenina M, Kochergin A. On one problem of reconstructing matrix distances between chains of DNA. IFAC – Papers Online, 2018;51(32): 378–383.
[4] Melnikov B, Trenina M. On a problem of the reconstruction of distance matrices between DNA sequences. International Journal of Open Information Technologies, 2018; 6(6):1–13.
[5] Melnikov B, Trenina M. The application of the branch and bound method in the problem of reconstructing the matrix of distances between DNA strings. International Journal of Open Information Technologies, 2018; 6(8):1–13.
[6] Melnikov B. Multiheuristic approach to discrete optimization problems. Cybernetics and Systems Analysis, 2006, vol. 42, nom. 3, P. 335–341.
[7] Ganesh A, Lin Z, Wright J, Wu L, Chen M. Fast algorithms for recovering a corrupted low-rank matrix. Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 3rd IEEE International Workshop; 2009; p. 213–216.
[8] Bro R, Smilde A. Principal component analysis. Analytical Methods, 2014;6(9):2812.
[9] Goodman S, Hedetniemi S. Introduction to the Design and Analysis of Algorithms. NY: McGraw-Hill;1977. p. 344.
[10] Hromkovi˘c J. Algorithmics for hard problems-introduction to combinatorial optimization, randomization, approximation, and heuristics. Berlin: Springer; 2003. p. 538.
[11] Melnikov B, Trenina M, Kochergin A. The approach to improving algorithms for calculating distances between DNA strings (using the example of the Needleman-Wunsch algorithm). Proceedings of Higher Educational Institutions. Volga Region. Physical and Mathematical Science. 2018;45(1):46–59.
[12] Nucleotide – NCBI [Electron. resource]. – Access mode: free, https://www.ncbi.nlm.nih.gov/nuccore.
[13] Ayala F, Kayger J. Modern genetics. V. 1. California, Menlo Park; 1980; p. 295