Some General Heuristics in the Traveling Salesman Problem and the Problem of Reconstructing the DNA Chain Distance Matrix

Boris Melnikov
bormel@mail.ru
Shenzhen MSU–BIT University
Shenzhen, Guangdong Province, China

Dmitrii Chaikovskii*
dmitriich@smbu.edu.cn
Shenzhen MSU–BIT University
Shenzhen, Guangdong Province, China

ABSTRACT

With all their differences, the two problems under consideration, namely the traveling salesman problem and the problem of restoring the DNA chain distance matrix, have a lot in common. This generality primarily consists in the following. For real problems and for standard methods of solving them, such as gradient descent, these problems can be formally solved, but in fact they are described by systems of equations with several dozen variables, and sometimes hundreds. In this regard, to solve them, we use sequential algorithms (step-by-step) for filling matrices, sometimes also using backtracking for the variables already considered. We show that such heuristics in the situations we are considering give acceptable anytime algorithms.

CCS CONCEPTS

• **Theory of computation** → **Design and analysis of algorithms**; • **Mathematics of computing** → **Statistical software**; • **Computer systems organization** → *Real-time systems.*

KEYWORDS

DNA matrix, traveling salesman problem, partially filled matrix, anytime algorithms, heuristics.

ACM Reference Format:
Boris Melnikov and Dmitrii Chaikovskii* . 2023. Some General Heuristics in the Traveling Salesman Problem and the Problem of Reconstructing the DNA Chain Distance Matrix. In *2023 7th International Conference on Computer Science and Artificial Intelligence (CSAI) (CSAI 2023), December 08–10, 2023, Beijing, China.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3638584.3638607

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. Similar algorithms are used in bioinformatics to compare sequenced genetic chains. Due to the large dimension of such chains, it is necessary to use heuristic algorithms that give approximate results.

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 results when using different algorithms. Therefore, there is a problem of assessing the quality of the used metrics (distances), the results of which can be concluded about the applicability of the algorithm to various studies.

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. We consider the possibility of using the developed and studied by us earlier method of comparative evaluation of algorithms for calculating distances between a pair of DNA strings to restore the partially filled matrix of distances. Matrix recovery occurs as a result of several computational passes. Estimation of unknown matrix elements are averaged 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%.

*Corresponding author.

CSAI 2023, December 08–10, 2023, Beijing, China

ACM ISBN 979-8-4007-0868-8/23/12
https://doi.org/10.1145/3638584.3638607

Here is a brief description of the content of those subsections that we consider the most important; we shall consider the following questions:

- the mathematical justification of the correctness of the constructions being made (Subsection 2.3);
- a brief statistical study of the problem of DNA matrix reconstruction of small dimensions (Subsection 3.1);
- the consideration of the application example of the method of the branch-and-bound algorithm in the problem of reconstructing a DNA template (Subsection 3.4).

2 SOME HEURISTICS FOR THE TRAVELING SALESMAN PROBLEM

The traveling salesman problem (TSP) is a well-known optimization problem where the goal is to find the shortest possible route that visits a given set of cities and returns to the origin city. Mathematically, it can be represented as follows:

$$\min \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} d_{ij} x_{ij}$$

subject to

$$\sum_{i=1, i\neq j}^{n} x_{ij} = 1, \quad j = 1, ..., n$$

$$\sum_{j=1, j\neq i}^{n} x_{ij} = 1, \quad i = 1, ..., n$$

$$u_i - u_j + n x_{ij} \leq n - 1, \quad 2 \leq i \neq j \leq n$$

where d_{ij} is the distance between city i and city j, x_{ij} is a binary variable that equals 1 if the path goes from city i to city j and 0 otherwise, and u_i are auxiliary variables to prevent sub-tours.

The following two subsections include the possibility of applying in our case (that is, in the case of the pseudo-geometric traveling salesman problem) standard algorithms commonly used for the geometric case

2.1 Nearest neighbor algorithm

The nearest neighbor algorithm is a greedy heuristic that selects the nearest city to the current city at each step. The steps of the algorithm are as follows:

(1) Select a random city as the starting point.
(2) Find the nearest city to the current city that has not been visited yet.
(3) Move to the nearest city and mark it as visited.
(4) Repeat step 2 until all cities have been visited.
(5) Return to the starting city to complete the tour.

The objective function of this algorithm can be expressed as:

$$C_{\text{NN}} = \sum_{i=1}^{n} d(x_i, x_{i+1}) + d(x_n, x_1)$$

where C_{NN} is the total distance of the tour, x_i is the i-th city in the tour, and $d(x_i, x_{i+1})$ is the distance between city x_i and city x_{i+1}.

2.2 Simulated annealing

Simulated annealing is a probabilistic technique that allows for the possibility of accepting suboptimal solutions in the early stages of the search to escape local minima. The algorithm can be outlined as follows:

(1) Choose an initial solution and set the initial temperature T.
(2) Perform a small perturbation on the current solution to get a new solution.
(3) If the new solution is better, accept it. If it is worse, accept it with a probability $\exp\left(\frac{-\Delta E}{T}\right)$, where ΔE is the increase in the objective function value.
(4) Decrease the temperature according to a cooling schedule and go to step 2.
(5) Repeat steps 2-4 until the stopping criteria are met.

The objective function can be written similarly to the one in the nearest neighbor section, but the acceptance criterion involves the Metropolis criterion given by:

$$P(\Delta E) = \exp\left(\frac{-\Delta E}{T}\right)$$

where $P(\Delta E)$ is the probability of accepting a solution with an increase in energy ΔE, and T is the temperature parameter that decreases over time according to a cooling schedule.

2.3 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

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

in the following way.

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

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.

(1) The cubic functions pass through the points:

$$X_i(t_i) = x_{s_i},$$
$$X_i(t_{i+1}) = x_{s_{i+1}},$$
$$Y_i(t_i) = y_{s_i},$$
$$Y_i(t_{i+1}) = y_{s_{i+1}}.$$

(2) The first and second derivatives are continuous at the junctions:

$$X_i'(t_{i+1}) = X_{i+1}'(t_{i+1}),$$
$$X_i''(t_{i+1}) = X_{i+1}''(t_{i+1}),$$

$$Y_i'(t_{i+1}) = Y_{i+1}'(t_{i+1}),$$
$$Y_i''(t_{i+1}) = Y_{i+1}''(t_{i+1}).$$

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:

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

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 $\Phi[X, Y]$:

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

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.

2.4 Geometric approach to the pseudo-geometric problem: optimal and pseudo-optimal placement of the points

In this subsection, we delve deeper into the methodology developed for resolving the pseudo-geometric Travelling Salesman Problem (TSP). This method includes state-of-the-art techniques for classifying input data and establishing a pseudo-optimal placement of points which may have significant applications in various fields including logistics and network design.

The task of pseudo-recovering the original coordinates of a set of points bears a greater complexity than merely classifying them (i.e., determining their class based on the known cost matrix in the context of the TSP).

Our proposed algorithm addresses this by simplifying the problem to restoring the location of points in a geometric rendition of the TSP using a distance matrix defined by the function

$$c : E \to \mathbb{N}_0. \quad (3)$$

This problem can be tackled effectively using the following strategy:

- Select two arbitrary points v_1 and v_2 from the vertex set V.
- Position v_1 at the origin, and v_2 at the point $(0, c(v_1, v_2))$.
- Determine the coordinates of each subsequent point (denoted as u with coordinates (x, y)), select two points (v and w) that have already been positioned, and solve the equation set:

$$\begin{cases} (x - x_v)^2 + (y - y_v)^2 = (c(u, v))^2 \\ (x - x_w)^2 + (y - y_w)^2 = (c(u, w))^2. \end{cases} \quad (4)$$

It should be noted that applying this algorithm to the distance matrix described in equation (3) in the pseudo-geometric TSP context does not generally recover the original points' locations in the geometric TSP scenario. Furthermore, the potential violation of the triangle inequality in the provided distance matrix may render the algorithm inapplicable.

In this research, our main focus and forthcoming work envision a refined algorithm conceived by us to address a specific TSP instance which employs the aforementioned algorithm for retrieving points' positions in a geometric TSP context. Moreover, this algorithm is versatile enough to be utilized in any TSP variant, although its application may not be recommended in most "random" or non pseudo-geometric special TSP cases.

Therefore, notwithstanding the general infeasibility of applying similar algorithms to the pseudo-geometric TSP variant, we endeavor to employ the same algorithms to address the city positioning problem, essentially tackling the minimization problem for a specially computed discrepancy or "badness" metric, defined as

$$\sqrt{\frac{2}{n \cdot (n-1)} \cdot \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left(c(u_i, u_j) - \tilde{c}(u_i, u_j)\right)^2}, \quad (5)$$

where:

- u_i ($i = 1, \dots, n$) are the points, with the coordinates of the i-th point represented by (x_i, y_i).
- $c(u_i, u_j)$ denotes the (i, j)-th element of the *given* cost matrix.
- $\tilde{c}(u_i, u_j)$ signifies the (i, j)-th element of the *obtained* cost matrix.

In a degenerate case (i.e., when $\sigma = 0$) and considering an ideal solution, a "badness" value of 0 should be achieved.

As a heuristic solution to this issue, we propose the following algorithm. Essentially, we are addressing the same minimization problem, albeit disregarding the geometric positioning of the points.

Algorithm for pseudo-optimal placement of points.
Input. Matrix $c : E \to \mathbb{N}_0$ (the matrix of weights of edges of a complete weighted graph with vertices $V = u_1, \dots, u_n$); the value $N \in \mathbb{N}$.
The parameter N represents the number of selectable pairs from the already allocated points. These pairs are selected to accommodate each new point, starting with the fourth one.

3 HEURISTICS IN THE PROBLEM OF RECONSTRUCTING THE DNA CHAIN DISTANCE MATRIX

The reconstruction of the DNA chain distance matrix is a critical problem in computational biology, which has implications in phylogenetics, molecular evolution, and other fields. In general, the problem entails determining the evolutionary distances between different DNA sequences.

A DNA molecule is a double helix consisting of four types of nucleotides: adenine (A), cytosine (C), guanine (G), and thymine (T). The sequential arrangement of these nucleotides forms a DNA sequence, which can be represented as a string of characters, e.g.,

$$S = \text{ACGTGACGTGGTAC}\ldots$$

A distance matrix is a matrix that contains the distances between all pairs of sequences in a set. If we have n sequences, the distance matrix D will be an $n \times n$ matrix, where D_{ij} represents the distance between sequence i and sequence j.

$$D = \begin{bmatrix} 0 & D_{12} & D_{13} & \ldots & D_{1n} \\ D_{21} & 0 & D_{23} & \ldots & D_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ D_{n1} & D_{n2} & D_{n3} & \ldots & 0 \end{bmatrix}$$

A common approach to computing the distances between sequences is to perform pairwise sequence alignments, using methods such as the Needleman-Wunsch algorithm.

The Needleman-Wunsch (NW) algorithm is a foundational global alignment method in bioinformatics, formulated by Saul B. Needleman and Christian D. Wunsch in 1970. This dynamic programming algorithm is primarily used to find the optimal alignment between two sequences, which includes identifying the similarities between them over their entire length. The mathematical formulation of the algorithm is as follows:

Given two sequences S_1 and S_2 of lengths m and n respectively, the NW algorithm constructs an $(m+1) \times (n+1)$ scoring matrix F where the entry $F(i, j)$ represents the optimal score for aligning the prefixes $S_1[1 \ldots i]$ and $S_2[1 \ldots j]$. The recursive definition of the scoring matrix is given by:

$$F(i,j) = \max \begin{cases} F(i-1, j-1) + s(S_1[i], S_2[j]), \\ F(i-1, j) + d, \\ F(i, j-1) + d \end{cases}$$

where:

- $s(a, b)$ is the score of aligning character a with b, often obtained from a substitution matrix.
- d is the gap penalty, representing the cost of inserting a gap in the alignment.

The matrix F is initialized as follows:

$$\begin{aligned} F(0,0) &= 0, \\ F(i,0) &= i \cdot d, \quad i = 1, 2, \ldots, m, \\ F(0,j) &= j \cdot d, \quad j = 1, 2, \ldots, n. \end{aligned}$$

After computing the entire matrix F, the optimal alignment is obtained by backtracking from $F(m, n)$ to $F(0, 0)$, constructing the alignment by either matching or inserting gaps as dictated by the values in the matrix.

This global alignment algorithm ensures that the best alignment across the entire length of the sequences is found, providing a holistic view of the similarities and differences between them.

The time complexity of the Needleman-Wunsch algorithm is $O(mn)$, and the space complexity is also $O(mn)$, which may become a bottleneck for very long sequences.

3.1 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 text of your paper should be formatted as follows.

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 you pass through the elements of the matrix "left to right" and "top to bottom" you 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 the following table (see 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.

Generation of input data was carried out based on the matrix obtained working with matrices obtained by applying the algorithm

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	334/356

of the Needleman – Wunsch [11]. We applied this algorithm to the MDNA chains of different animals taken from the NCBI data Bank [16], and sequenced mDNA chains were taken for one representative of each of the 28 mammalian orders (mammalian classification is chosen according to [15], 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 you receive 10 (or 30) of the subproblems, we calculate the stop. Also, of course, we stopped the calculations and when you get two similar matrices, and it is the number of such options (out of 1000 possible) and is reflected in Tab. 1. In that table:

- 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 -– after receiving 30 subtasks.

3.2 Minimization technique

In the study of the matrix $M = \left(m_{i,j}\right)$ and its constituent elements $m_{i,j}$, it is conventionally assumed that

$$i, j, k \in 1, 2, \ldots, n \tag{6}$$

with the exception of diagonal elements. In this context, diagonal elements are the elements denoted by $m_{i,i}$, and any arithmetical expressions containing these elements are excluded from formulaic considerations. This presupposition aids in focusing on the pivotal elements contributing to the total error denoted by σ. The formal definition of σ is expressed as:

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sigma_{i,j}, \tag{7}$$

where different approaches exist for its calculation, one of which involves the sequential application of the following formulas:

- $r^{(1)} i, j, k = \max\left(mi, j, m_{k,j}, m_{k,i}\right)$,
- $r^{(2)} i, j, k = \min\left(mi, j, m_{k,j}, m_{k,i}\right)$,
- and the definition of $\sigma_{i,j}$ is:

$$\max_{1 \leqslant k \leqslant n \;\; k \neq i, k \neq j} \frac{2r^{(1)}_{i,j,k} + r^{(2)}_{i,j,k} - m_{i,j} - m_{k,i} - m_{k,j}}{r^{(2)}_{i,j,k}}. \tag{8}$$

Moving forward, we can rephrase the primary problem as an optimization task aiming to minimize the error value σ, a metric previously referred to as "badness" in earlier research. The procedure necessitates a sequential, stepwise insertion of missing elements, thereby streamlining the implementation of the corresponding algorithm.

By employing this approach, we generate a matrix populated with noisy data, denoted as $u^{\delta}_{i,j}$. Subsequently, we reconstruct the u^{ϵ} function through the resolution of previous equation. It is crucial to note that the noise level δ, which manifests during the restoration of missing values, can be quantitatively assessed through a meticulous analysis of the aberrations from the "isosceles triangle" principle, as detailed in [12].

Ultimately, this methodical strategy of sequentially populating missing matrix elements assures a progressive enhancement of the resultant solution. Theoretically, this rationalizes the forsaking of the more time-consuming branch and boundary method in favor of the greedy algorithm for discerning the value of individual elements, thereby expediting the overall computational process.

3.3 Quality assessment criteria for numerical solutions

Previously, we discussed the imperative of determining an effective method to gauge the quality of the solutions generated by our recovery algorithms. It remains clear that the existing computational model does not fully address the concerns pertaining to the quality assessment of matrix restoration. Consequently, a straightforward quality criterion could be the comparison, based on an appropriate metric, of the restored matrix with the actual derived distance matrix. This comparison could potentially be executed for small-scale examples, albeit restricted to a finite number of iterations, preferably during the preliminary phases of algorithm testing.

To this end, we introduce two plausible criteria for evaluating the numerical solutions to such restoration problems:

(1) The first criterion involves contrasting the matrix reconstructed by the current simplified algorithm with the matrix developed through the application of a comprehensive algorithm during the formation of each element, as documented in [13, 14]. This criterion value will be denoted by σ.
(2) The second criterion scrutinizes the discrepancy using a distinctive method, leveraging the same algorithms that function as supportive entities within the general recovery algorithm explored in this study. The criterion value in this case will be symbolized by δ (or d in some previous publications).

In both instances, the overarching aim remains to diminish the values deduced by the applied criteria.

We delineate the specific formulas as stated below:

(1) For σ, the typical formulation is

$$\sigma = \sqrt{\frac{2}{n \cdot (n-1)} \cdot \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (m_{i,j} - \widetilde{m}i, j)^2}, \tag{9}$$

where all the elements $\widetilde{m}i, j$ are retrieved via the implementation of the original algorithm (like the well-cited Needleman–Wunsch algorithm), excluding any element restoration. It is pertinent to mention that this method is seldom utilized, especially for extensive matrices generated through certain distance determination algorithms, highlighting the more universal application of the subsequent criterion, δ.

(2) Typically, δ is defined as

$$\delta = \sum_{i=1}^{n-2} \sum_{j=i+1}^{n-1} \sum_{k=j+1}^{n} \delta_{i,j,k}, \tag{10}$$

with a clear emphasis on not utilizing $\widetilde{m}i, j$ values. Each $\delta i, j, k$, where $1 \leqslant i, j, k \leqslant n$, and $i \neq j$, $i \neq k$, $j \neq k$ represents the "badness" of the respective triangle, typically calculated as follows:

(2a) Initially, we reassign the values of $m_{i,j}$, $m_{i,k}$, and $m_{j,k}$ to a, b, and c, satisfying $a \geqslant b \geqslant c$.

(2b) In instances where $a \geqslant b + c$, indicating a violation of the triangle inequality, a pre-determined constant ω (commonly, $\omega = 2$) is employed to establish

$$\delta_{i,j,k} = \max\left(\frac{a}{b+c}, \omega\right). \tag{11}$$

(2c) Conversely, for regular triangles, we calculate its angles, denoted as α, β, and γ, where $\alpha \geqslant \beta \geqslant \gamma$.

(2d) Subsequently, we define

$$\delta_{i,j,k} = \frac{\alpha - \beta}{\gamma}. \tag{12}$$

It is critical to acknowledge that δ can be computed swiftly, despite the necessity to analyze approximately n^3 triangles. Furthermore, we observe a correlation between these criteria and the task at hand: for instance, random matrices yield notably poorer results according to the δ criterion, even for minor dimensions. As evidence, a 13x13 random matrix recorded δ values within the 0.4 to 0.5 range, substantially exceeding the corresponding values for correct matrices of size 28x28 with a lower initial fullness percentage.

3.4 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 Fig. 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.

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

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

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

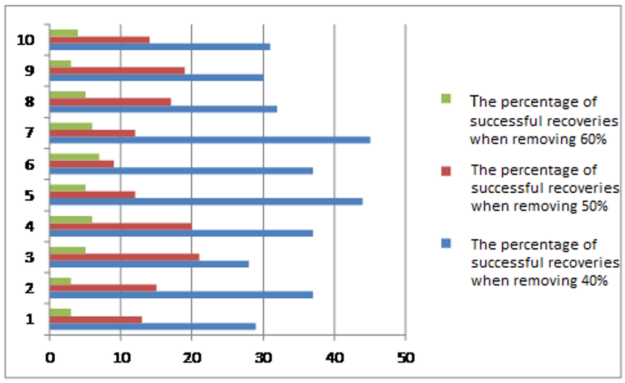


Figure 2: The average percentage of successful recoveries, 5x5.

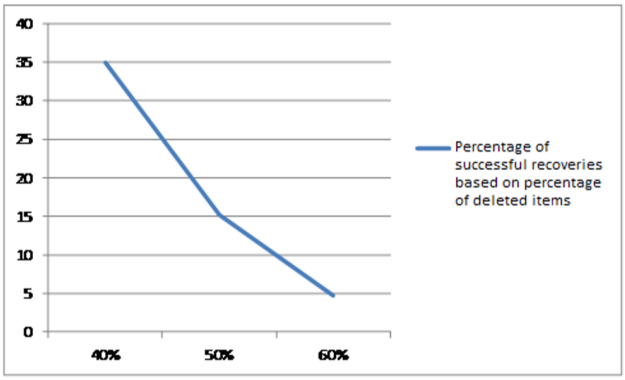
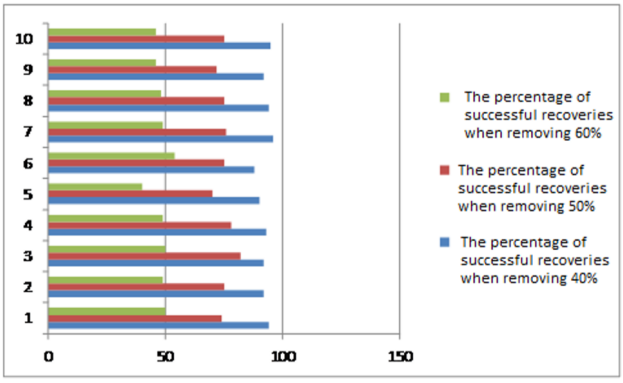
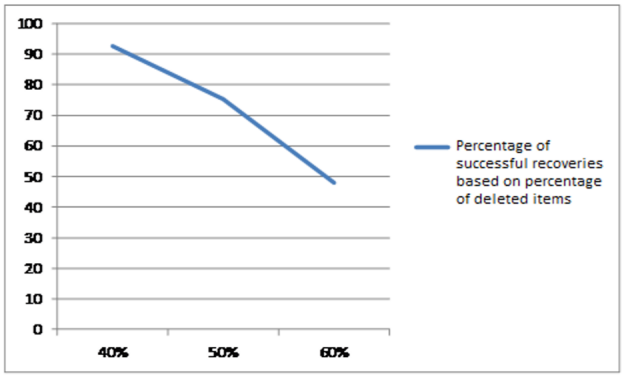


Figure 3: The results of numerical experiments for 10×10 matrices.



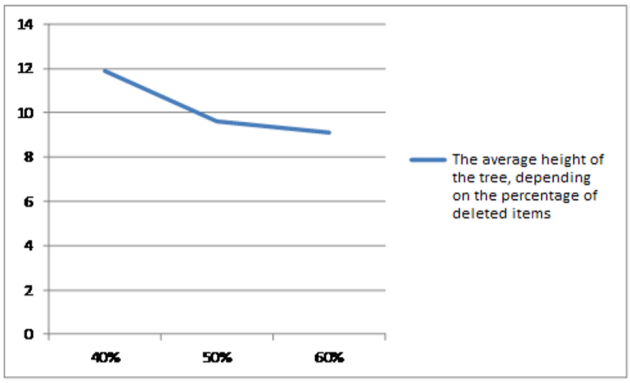
We also present a graph of the average number of successful recoveries, Fig. 4.

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



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%), Fig. 5.

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, Fig. 6 and Fig. 7.

We can see that the increase in the height of the decision tree with the growth of the dimension of the matrix occurs at a high speed. Also, with increasing the dimension of the matrix significantly

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

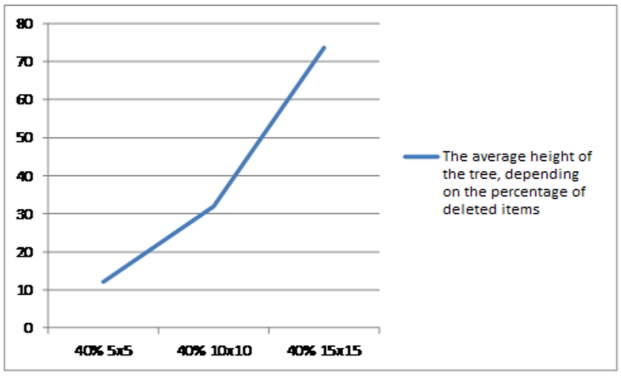
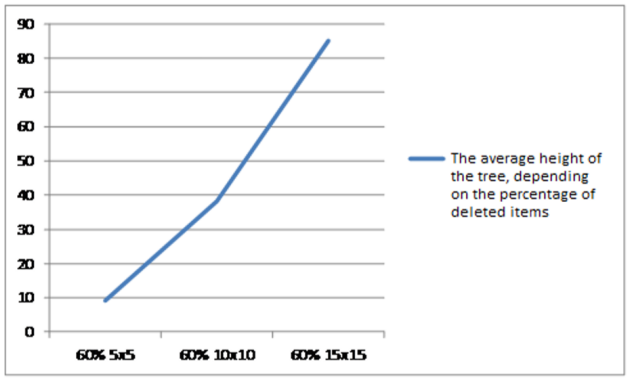


Figure 7: The height of the decision tree, 60% of deleted elements.



increases the number of subtasks of the same level, and therefore the search for problems takes much longer.

4 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.

ACKNOWLEDGMENTS

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") and National Natural Science Foundation of China (No. 12350410359).

REFERENCES

[1] Melnikov B., Melnikova E., Pivneva S., Trenina M. An approach to analysis of the similarity of DNA-sequences. *CEUR Workshop Proceedings*, 2018, vol. 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, vol. 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, vol. 51, issue 32, P. 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, vol. 6, nom. 6, P. 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, vol. 6, nom. 8, P. 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, 344 p.

[10] Hromkovič J. *Algorithmics for Hard Problems. Introduction to Combinatorial Optimization, Randomization, Approximation, and Heuristics*. Berlin: Springer, 2003, 538 p.

[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, vol. 45, nom. 1, P. 46–59.

[12] Melnikov, B., Pivneva, S., Trifonov, M.: Various algorithms, calculating distances of DNA sequences, and some computational recommendations for use such algorithms. CEUR Workshop Proceedings, **1902**, pp. 43–47 (2017),

[13] Melnikov, B., Trenina, M.: On a problem of the reconstruction of distance matrices between DNA sequences. International Journal of Open Information Technologies, **6**(6), pp. 1–13 (in Russian) (2018)

[14] Melnikov, B., Trenina, M.: Application of the branches and boundaries method in a problem of the reconstruction of distance matrices between DNA sequences. International Journal of Open Information Technologies, **6**(8), pp. 1–13 (in Russian) (2018)

[15] Ayala F., Kayger J. *Modern genetics. V. 1.* California: Menlo Park, 1980, 295 p.

[16] Nucleotide – NCBI [Electron. resource]. – Access mode: free, `https://www.ncbi.nlm.nih.gov/nuccore`.