CYBERNETICS AND PHYSICS, VOL. 14, NO. 4, 2025, 356–362

THE PLANAR TRIANGULATION WITH NOISY DATA: THE DYNAMIC CHANGING THE ORDER OF POINTS

Boris Melnikov[1]
bormel@mail.ru

Bowen Liu[1]
liubowenmbu@163.com

Dang Van Vinh[2]
dangvvinh@hcmut.edu.vn

[1]Faculty of Computational Mathematics and Cybernetics, Shenzhen MSU-BIT University, Shenzhen, China
[2]Department of Applied Mathematics, Faculty of Applied Science,
Ho Chi Minh City University of Technology (HCMUT), Ho Chi Minh City, Vietnam

Article history:
Received 28.11.2025, Accepted 20.12.2025

Abstract

We continue to consider triangulation algorithms. To the subject areas in which triangulation is used, we can add, for example, the task of clustering sensor data by location (spatial clustering) is to group data points located in the same region based on their density. The goal is to identify clusters that are close to each other. In the previous paper, the aspects related to the possible parallelization of the heuristic algorithms proposed by us were mostly considered. The main purpose of this paper is different: the main heuristic considered here is to improve the already obtained preliminary location of a certain point in the "bad case", i.e., when we obtain from further calculations, that such an arrangement is unsuccessful; this is backtracking. We provide a general description of the triangulation algorithm we use. However, the algorithms we are considering are not limited to one description of backtracking. We use several options for choosing the next point to consider: either we take an arbitrary one (usually the next in number), or the one closest to the geometric center of the points already located, or the one furthest from this center. A simple example of such an algorithm is given. However, the main ideas of backtracking are noticeable in such a small example. Since we need to consider a wide variety of noise variants superimposed on the geometric arrangement of points, we, as in previous publications, consider geometric distances and multiply them by another implementation of independent identically distributed random variables with a mathematical expectation equal to 1 and different variants of the standard deviation. These standard deviation variants in our constructions vary from 0 (so-called geometric version) to 0.14. The end of the paper is devoted to a description of the experimental results and their understanding.

Key words

triangulation, noisy distance data, planar coordinate reconstruction, heuristic algorithm, backtracking

1 Introduction

In this paper, we continue to consider triangulation algorithms, the connection of which with mathematical models in physics (in particular, with models of black holes) was described in the previous paper [Melnikov and Liu, 2025] and is briefly repeated in the next section.

To the mentioned subject areas in which triangulation is used, we can add, for example, the task of clustering sensor data by location (spatial clustering) is to group data points located in the same region based on their density. The goal is to identify clusters that are close to each other. A possible approach to creating specific algorithms is the so-called hierarchical clustering, which creates a hierarchy of nested partitions of the original set of objects, while taking into account the distance between points and their surroundings. In our opinion, such ideas are similar to those used in this paper.

In the previous paper, the aspects related to the possible parallelization of the heuristic algorithms proposed by us were mostly considered. The main purpose of this paper is different: the main heuristic considered here is to improve the already obtained preliminary location of a certain point in the "bad case", i.e., when we obtain from further calculations, that such an arrangement is unsuccessful. From the point of view of artificial intelligence algorithms (more precisely, algorithms using the so-called rule-based approach), such actions can be called backtracking. Apparently, this term was introduced in 1950 by the well-known mathematician

D. H. Lehmer; we apply its description according to the classical monographs [Brassard and Bratley, 1995; Russell and Norvig, 2009]. Among the recent works related to backtracking algorithms, which are also interesting for our situation, we note, for example, [Isabel García-Planas, 2023; Emami et al., 2025; Dramé et al., 2025; Miao et al., 2026].

Thus, in this paper, we do not practically consider the issues of parallelization of the proposed algorithms. A general description of the tasks we are considering can be understood from the section content of the paper provided here. Therefore, we shall provide a detailed description of the content of the sections.

Section 2 is preliminary. We repeat from the previous article those possible applications of triangulation algorithms that we consider to be the most important or relevant to the description of the algorithms in this paper and add some additional applications of these algorithms.

The title of Section 3 is "The general formal description of the algorithm". Once again, we primarily focus not on the possibility of parallelization, but on improving the algorithm through backtracking. This fact is the main one in the description of the main algorithm of the paper. However, the algorithms we are considering are not limited to one description of backtracking. We use several options for choosing the next point to consider: either we take an arbitrary one (usually the next in number), or the one closest to the geometric center of the points already located, or the one furthest from this center. It should also be noted that to average various potential options for the location of a point, we do not use the usual version of the arithmetic mean, but more complex algorithms that give better results, see [Melnikov and Liu, 2025].

A simple example of such an algorithm (considering not matrices of size 99×99, which are the main ones in this paper, but matrices of size 5×5) is given in Section 4. However, according to the authors, the main ideas of backtracking are noticeable in such a small example. So, in Section 4, we demonstrate the backtracking procedure step by step.

The title of Section 5 is "Organization of the calculations (setup of the experiments)". Since we need to consider a wide variety of noise variants superimposed on the geometric arrangement of points, we, as in previous publications, consider geometric distances and multiply them by another implementation of independent identically distributed random variables with a mathematical expectation equal to 1 and different variants of the standard deviation. These standard deviation variants in our constructions vary from 0 (this results in the so-called geometric version, and the triangulation task is quite simple) to 0.14.

The last two Sections (6 and 7) are devoted to the description of experimental results, their comprehension, and a very brief formulation of tasks to be performed in the future based on the results of this paper.

2 Preliminaries

In this paper, we repeat from the previous article those possible applications of triangulation algorithms that we consider to be the most important or relevant to the description of the algorithms in this paper, and add some other applications of these algorithms. Thus, in previous papers [Melnikov et al., 2006; Melnikov and Liu, 2025], we noted that distance-based triangulation is a fundamental technique for localization and mapping in two-dimensional spaces. Potential applications include wireless sensor networks, mobile robotics, and mapping systems, where real-time and reliable two-dimensional localization is essential. For example, significant results have been achieved in modeling black holes and the cosmological conditions under which conventional triangulation methods make it possible to understand what the plane of the event horizon may consist of; all of this connects quantum geometry with the concepts of space and time. In real-world applications, however, sensor measurements are often corrupted by noise and unpredictable errors, making it infeasible to reconstruct the true coordinates of points directly from raw distances.

In previous publications, we used heuristic algorithms to solve the triangulation problem. The algorithms also relate to a special formulation of the traveling salesman problem (the so-called pseudo-geometric version) and sometimes to some DNA analysis algorithms; see [Melnikov et al., 2022; Melnikov and Chaikovskii, 2023; Melnikov and Chaikovskii, 2024] etc.

Classical optimization-based approaches to restore the coordinates of the entire set of points can mitigate noise but usually incur high computational costs, as they attempt to minimize global error measures. To balance computational efficiency and reconstruction accuracy, we introduce a heuristic algorithm for planar point coordinate recovery.

In [Melnikov and Liu, 2025], we continued to describe algorithms for solving the triangulation problem with such noisy data, while in that work, the emphasis was on the possible parallel implementation of the proposed algorithms. The contents of the previous paper can be summarized in a bit more detail as follows. Firstly, we introduced the research background and formulated the reconstruction problem. Then, we presented a heuristic algorithm, followed by a description of its parallelization strategy; we applied several different small interrelated heuristics, which eventually brought successful results. After outlining the experimental setup, we reported and discussed the computational results. Finally, we highlighted possible directions for future work.

We noted in a previous paper that the experimental results demonstrate that our reconstruction algorithm performs reliably across different noise levels and problem sizes. In all tested configurations, the reconstruction error, as measured by the badness score, does not exceed approximately 200 % of the initial noise-induced deviation. The observed limitation is mainly due to the fact that the pseudo-noisy distance matrices cannot fully capture the true geometric structure of the original points, which inherently constrains the achievable accuracy.

3 The general formal description of the algorithm

Describing the main algorithm of the paper, we primarily focus not on the possibility of parallelization, but on improving the algorithm through backtracking.

However, the algorithms we are considering are not limited to one description of backtracking. We use several options for choosing the next point to consider: either we take an arbitrary one (usually the next in number), or the one closest to the geometric center of the points already located, or the one furthest from this center.

It should also be noted that to average various potential options for the location of a point, we do not use the usual version of the arithmetic mean, but more complex algorithms that give better results. A brief description of these algorithms is considered in our previous paper [Melnikov and Liu, 2025].

Algorithm 1.

Input:

- the distance matrix; it may have been obtained by a pseudogeometric method, but this fact is not used for the operation of the algorithm;
- the method of selecting the next point is one of three possible ones: either the next in number, or the one closest to the center of the system from the points already placed, or the one furthest from this center.

Step 1. Place the 1st point at the origin.

Step 2. Select the next point according to the originally set selection method.

Step 3. If these are the 2nd or 3rd points, then place them according to the obvious algorithms, and then proceed to Step 2.

Step 4. For each pair of points that have already been placed, select the coordinates of the new point as the third vertex of a triangle with known 3 sides.

Step 5. Average the coordinates of all pairs according to Weiszfeld's Algorithm [Venceslau et al., 2016; Neumayer et al., 2020; Melnikov and Liu, 2025].

Step 6. Calculate the real distance matrix for the points that have already been placed.

Step 7. Compare all the elements with the original distance matrix, and calculate for each element the badness as the modulus of the difference with the corresponding value of the original matrix.

Step 8. Calculate the amount of badness for each row.

Step 9. Select the maximum value of the total badness and fix this element.

Step 10. For a fixed element, recalculate the coordinates according to Steps 4 and 5 of the algorithm.

Step 11. If the point in question is not the last, then go to Step 2.

Output:

- placed elements (their coordinates). □

Note to step 5 of the algorithm. The reason for choosing this geometric center search algorithm is as follows. Let there be a set of measured positions of the same point distorted by a random error. In this case, regions with a high density of these dimensions contain fewer distorted coordinates than regions with a low density, especially if the latter are removed from the "cluster" of dimensions. Consequently, the contribution of the error to the final position of the reconstructed point turns out to be less inside the dense region and more outside it. The algorithm automatically takes this into account: the reconstructed point is shifted to the region with the highest measurement density, thereby reducing the overall effect of noise on the result.

Note to step 7 of the algorithm. Sometimes, we use the square of the difference. For example, this is what we do in the examples discussed below.

4 A simple example of the backtracking

A simple example of such an algorithm (considering not matrices of size 99×99, which are the main ones in this paper, but matrices of size 5×5) is given in this section. According to the authors, the main ideas of backtracking are noticeable in such a small example.

Below is the demonstration of the backtracking procedure, step by step.

4.1. Original distance matrix. Let the given matrix be as follows:

$$\mathsf{M}_{\text{orig}} = \begin{bmatrix} \mathsf{A} & \mathsf{B} & \mathsf{C} & \mathsf{D} & \mathsf{E} \\ 0.000 & 3.592 & 5.045 & 2.165 & 4.326 \\ 3.592 & 0.000 & 3.795 & 4.575 & 1.352 \\ 5.045 & 3.795 & 0.000 & 4.149 & 2.679 \\ 2.165 & 4.575 & 4.149 & 0.000 & 4.694 \\ 4.326 & 1.352 & 2.679 & 4.694 & 0.000 \end{bmatrix}$$

In this section, we shall give all numerical values with an accuracy of 3 digits after the decimal point.

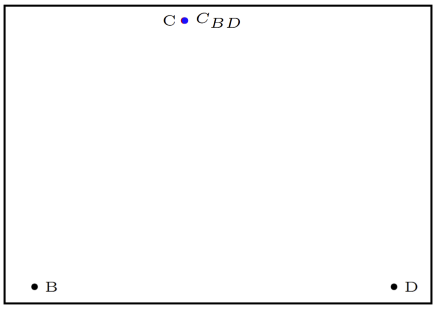
4.2. Matrix with noise. Using the last matrix, we obtained the following matrix with noise:

$$\mathsf{M}_{\text{err}} = \begin{bmatrix} \mathsf{A} & \mathsf{B} & \mathsf{C} & \mathsf{D} & \mathsf{E} \\ 0.000 & 4.010 & 5.248 & 2.222 & 4.137 \\ 4.010 & 0.000 & 3.723 & 4.379 & 1.366 \\ 5.248 & 3.723 & 0.000 & 4.123 & 2.625 \\ 2.222 & 4.379 & 4.123 & 0.000 & 4.625 \\ 4.137 & 1.366 & 2.625 & 4.625 & 0.000 \end{bmatrix}$$

4.3. Phase 1: Start with B and D.

Starting points: B and D are fixed. Point B is placed at the origin, and point D is placed at the desired distance from it along the OX axis.

4.4. Phase 2: Restore C.



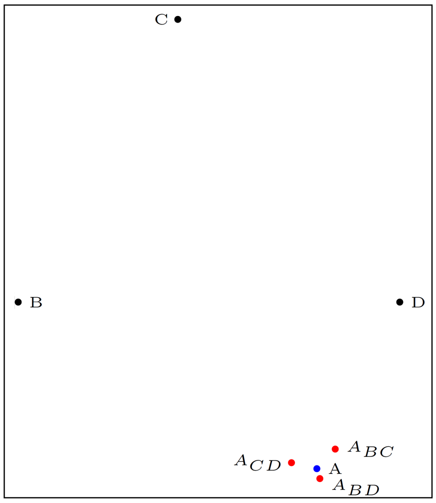
Here and further, the subscripts (always including 2 points) will symbolize the base of the triangle along which the coordinates of the selected point are plotted (in this first example, these are points B and D). The final point placement options will be indicated without subscripts, usually in blue. However, as follows from the previous one, the word “final” is not entirely correct: all coordinates may still change later when backtracking is applied.

Certainly, we cannot have the backtracking effect (because only three points are used).

4.5. Phase 3: Restore A.

We have the following stages here.

4.5.1. Calculated coordinates and values



4.5.2. $\mathrm{sub\_M_{err}}$.

Now, we compute the badness matrix and row sums, then select A for backtracking.

$$\mathrm{M_{err}} = \begin{bmatrix} \mathrm{A} & \mathrm{B} & \mathrm{C} & \mathrm{D} \\ 0.000 & 4.010 & 5.248 & 2.222 \\ 4.010 & 0.000 & 3.723 & 4.379 \\ 5.248 & 3.723 & 0.000 & 4.123 \\ 2.222 & 4.379 & 4.123 & 0.000 \end{bmatrix}$$

Note that this matrix is just a kind of “grid” of the original matrix $\mathrm{M_{err}}$.

4.5.3. Restored_matrix

The following matrix is obtained according to Algorithm 1 described above, namely Steps 4, 5, and 6 of this algorithm. We do not provide details of the calculations, but only the final values of the matrix elements.

$$\mathrm{Restore} = \begin{bmatrix} \mathrm{A} & \mathrm{B} & \mathrm{C} & \mathrm{D} \\ 0.000 & 3.924 & 5.393 & 2.134 \\ 3.924 & 0.000 & 3.723 & 4.379 \\ 5.393 & 3.723 & 0.000 & 4.123 \\ 2.134 & 4.379 & 4.123 & 0.000 \end{bmatrix}$$

4.5.4. Badness (subtract of the two matrices, then square each element)

$$\mathrm{Badness} = \begin{bmatrix} 0.000 & 0.007 & 0.021 & 0.008 \\ 0.007 & 0.000 & 0.000 & 0.000 \\ 0.021 & 0.000 & 0.000 & 0.000 \\ 0.008 & 0.000 & 0.000 & 0.000 \end{bmatrix}$$

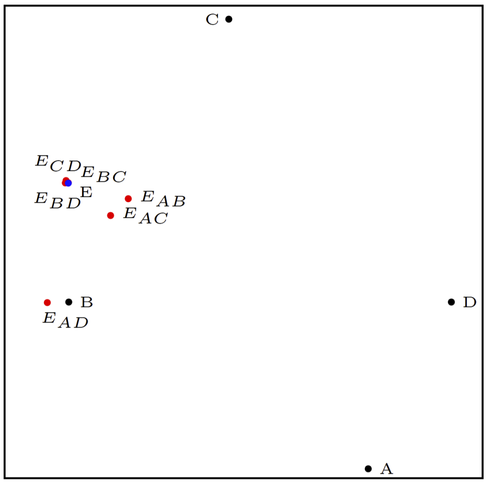
4.5.5. Sums of rows of badness

$$\mathrm{Sum} = \begin{bmatrix} 0.036 \\ 0.007 \\ 0.021 \\ 0.008 \end{bmatrix}$$

4.6. Phase 4: Restore E.

We have the following stages here.

4.6.1. Calculated coordinates and values



4.6.2. $\mathrm{sub\_M_{err}}$.

Now, we compute the badness matrix and row sums. After that, we shall select A for backtracking once again, because in the vector that we will construct next, the value corresponding to A will be the maximum.

Thus, we compute the badness matrix again and perform another backtracking.

Next, we shall write significantly fewer comments: they practically coincide with the above ones.

$$M_{\text{err}} = \left[\begin{array}{ccccc} A & B & C & D & E \\ 0.000 & 4.010 & 5.248 & 2.222 & 4.137 \\ 4.010 & 0.000 & 3.723 & 4.379 & 1.366 \\ 5.248 & 3.723 & 0.000 & 4.123 & 2.625 \\ 2.222 & 4.379 & 4.123 & 0.000 & 4.625 \\ 4.137 & 1.366 & 2.625 & 4.625 & 0.000 \end{array}\right]$$

4.6.3. Restored_matrix

$$\text{Restore} = \left[\begin{array}{ccccc} A & B & C & D & E \\ 0.000 & 3.924 & 5.393 & 2.134 & 4.743 \\ 3.924 & 0.000 & 3.723 & 4.379 & 1.362 \\ 5.393 & 3.723 & 0.000 & 4.123 & 2.628 \\ 2.134 & 4.379 & 4.123 & 0.000 & 4.592 \\ 4.743 & 1.362 & 2.628 & 4.592 & 0.000 \end{array}\right]$$

4.6.4. Badness (subtract of the two matrices, then square each element)

$$\text{Badness} = \begin{bmatrix} 0.000 & 0.007 & 0.021 & 0.008 & 0.367 \\ 0.007 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.021 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.008 & 0.000 & 0.000 & 0.000 & 0.001 \\ 0.367 & 0.000 & 0.000 & 0.001 & 0.000 \end{bmatrix}$$

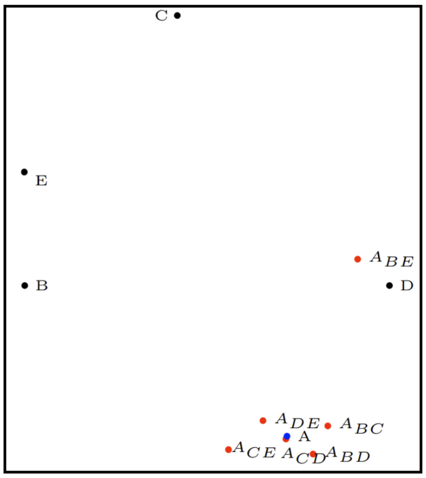
Note that there are very small values in this matrix, which give 0 with the rounding we use.

4.6.5. Sums of rows of badness

$$\text{Sum} = \begin{bmatrix} 0.402867 \\ 0.007431 \\ 0.021098 \\ 0.008835 \\ 0.367756 \end{bmatrix}$$

In this example, when restoring the fifth point, the reverse stroke already changes one of the previous points, and not the last restored point, so the advantage of the method becomes noticeable precisely at the fifth point.

4.7. Phase 4: Restore A once again.



As before, the exact values of specific coordinates are of little interest. They are understandable based on the subsequent data that calculates the value of badness.

4.8. Comparison. The restored matrix without backtracking:

$$M_{\text{no-bt}} = \left[\begin{array}{ccccc} A & B & C & D & E \\ 0.000 & 3.924 & 5.393 & 2.134 & 4.743 \\ 3.924 & 0.000 & 3.723 & 4.379 & 1.362 \\ 5.393 & 3.723 & 0.000 & 4.123 & 2.628 \\ 2.134 & 4.379 & 4.123 & 0.000 & 4.592 \\ 4.743 & 1.362 & 2.628 & 4.592 & 0.000 \end{array}\right]$$

The restored matrix with backtracking:

$$M_{\text{bt}} = \left[\begin{array}{ccccc} A & B & C & D & E \\ 0.000 & 3.632 & 5.220 & 2.188 & 4.473 \\ 3.632 & 0.000 & 3.723 & 4.379 & 1.362 \\ 5.220 & 3.723 & 0.000 & 4.123 & 2.628 \\ 2.188 & 4.379 & 4.123 & 0.000 & 4.592 \\ 4.473 & 1.362 & 2.628 & 4.592 & 0.000 \end{array}\right]$$

$$\begin{gathered} \text{Badness}(M_{\text{err}} \text{ and } M_{\text{orig}}) = 0.783, \\ \text{Badness(no-bt)} = 0.963, \\ \text{Badness(bt)} = 0.473. \end{gathered}$$

The first value of badness is the distance between the original matrix and the matrix obtained after exposure to noise; the distance is understood as the square root of the sum of the squares of the corresponding elements. Apparently, this value is less interesting.

The second and third values are the distances from the resulting matrix to the original one when we do not use (or, vise versa, use) backtracking. We can say that working without backtracking is the work of algorithm 1 described above, skipping Steps 6 through 10.

Apparently, no further comments are needed.

Of course, many similar situations, analogous to the artificial example discussed above, arise when processing real matrices, primarily with large dimensions.

5 Organization of the calculations (setup of the experiments)

The authors believe that the organization of computational experiments was carried out in accordance with many modern articles; let us mention, for example, [Chen et al., 2026; Wang et al., 2026].

Since we need to consider a wide variety of noise variants superimposed on the geometric arrangement of points, we, as in previous publications, consider geometric distances and multiply them by another implementation of independent identically distributed random variables with a mathematical expectation equal to 1 and different variants of the standard deviation. These standard deviation variants in our constructions vary from 0 (this results in the so-called geometric version, and the triangulation task is quite simple) to 0.14.

All the original data, as well as the results of computational experiments, are in the archive, which is available at `https://disk.yandex.ru/d/M_OhYBXHKR4fsQ`. Next, we shall describe the organization of calculations in more detail.

We select 5 variance values; in other words, σ value is from 0 to 0.14.

For each of the sigma values, we consider different dimensions: 25, 33, 49, and 99; that is, we can consider the 5×4 table.

For each cell of the table, we conduct 20 computational experiments; in the cell, we write the average value of the values obtained using the following steps:

- we make 4 different matrices of the required dimension; note that these matrices are geometric;
- for each of them, we get 5 pseudogeometric matrices based on the σ value known to us; note that there will be $4 \cdot 5 = 20$ experiments for the cell in question;
- we restore the main values; that is, we get the pseudo-location of the points; we use Algorithm 1, and for each experiment 2 times, that is, we either use backtracking or do not use it;
- we calculate and average the values of badness for this pseudo-location of points.

In each cell of the table, the value is "in the numerator" without using backtracking, and "in the denominator" using it.

In this case, we get 3 such tables; recall that the method of selecting the next point is one of the three following possibilities:

- either the next in number;
- or the one closest to the center of the system from the points already placed;
- or the one furthest from this center.

6 Experimental results

Thus, the first table is for the "next" selection:

Next

sigma / N	**25**	**33**	**49**	**99**
0	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
0.02	0.229 / 0.209	0.393 / 0.355	0.398 / 0.473	0.793 / 0.742
0.06	0.784 / 0.768	1.432 / 1.207	2.164 / 1.564	3.861 / 3.731
0.10	1.614 / 1.321	2.053 / 2.731	3.189 / 3.166	7.576 / 6.968
0.14	2.268 / 2.089	3.568 / 3.143	5.518 / 5.402	10.956 / 9.659

As expected, the values improve after applying backtracking.

The second table is for the "nearest" selection:

Nearest

sigma / N	**25**	**33**	**49**	**99**
0	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
0.02	0.338 / 0.276	0.245 / 0.263	0.305 / 0.351	1.065 / 0.618
0.06	2.877 / 1.835	2.660 / 2.154	3.306 / 3.486	11.107 / 13.818
0.10	4.221 / 3.968	6.211 / 6.562	9.932 / 9.341	18.035 / 20.953
0.14	5.126 / 5.334	9.131 / 8.164	13.144 / 12.715	24.857 / 25.396

It is clear that this strategy ("nearest") is worse than the previous one ("next"). Moreover, there are even cells in the table (we marked them with a fill), in which the use of backtracking worsens the results; we can explain this situation, but this explanation is unlikely to be interesting for the subject of the paper.

The last table is for the "farthest" selection:

Farthest

sigma / N	**25**	**33**	**49**	**99**
0	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
0.02	0.503 / 0.471	0.316 / 0.443	0.988 / 0.645	0.967 / 1.024
0.06	1.060 / 0.925	1.348 / 1.301	1.667 / 1.798	3.522 / 4.032
0.10	1.624 / 1.467	1.883 / 1.797	2.979 / 2.873	5.181 / 5.158
0.14	2.027 / 2.013	2.703 / 2.865	3.758 / 4.228	8.956 / 8.915

7 Conclusion

From the experimental results given, we draw the following conclusions.

- The strategy of choosing the most remote element is the best. It is slightly better than the "random" strategy, and they both far outperform the selection of the closest element.
- Backtracking improves the algorithm, but not always. In the "bad" situations (large dimension, large standard deviation of the initial data), we observe such an improvement much more and more clearly.
- If we randomly select a point, backtracking always provides an improvement.
- Therefore, some further improvement of specific implementations of backtracking algorithms is required: we have described only the simplest version of it in the paper.

Thus, in the next paper, we propose to consider:

- the improvement of the backtracking algorithm,
- its relationship to parallelism, [Sun, 2005; Sovrasov and Barkalov, 2020; Melnikov and Liu, 2025] etc.,
- as well as a statistical analysis of the results obtained.

References

Brassard, G. and Bratley, P. (1995). Fundamentals of algorithmics.

Chen, Y., He, Q., Chen, X., and Zheng, L. (2026). Dynamic link prediction in construction innovation networks: An integrated framework of topological and content attributes. *Expert Systems with Applications*, **299**.

Dramé, M., Garcia-Rodriguez, F.-J., Ershov, D., Martyn, J. E., Tinevez, J.-Y., Buchrieser, C., and Escoll, P. (2025). Backtracking metabolic dynamics in single cells predicts bacterial replication in human macrophages. *Nature Communications*, **16** (1).

Emami, H., Emami, S., and Rezaverdinejad, V. (2025). A backtracking search-based extreme gradient boosting algorithm for soil moisture prediction using meteorological variables. *Earth Science Informatics*, **18** (2).

Isabel Garcıa-Planas, M. (2023). Controllability properties for multi-agent linear systems. a geometric approach. *Cybernetics and Physics*, **12** (1), pp. 28 – 33.

Melnikov, B. and Chaikovskii, D. (2023). Some general heuristics in the traveling salesman problem and the problem of reconstructing the dna chain distance matrix. p. 361 – 368.

Melnikov, B. and Chaikovskii, D. (2024). Pseudogeometric version of the traveling salesman problem, its application in quantum physics models and some heuristic algorithms for its solution. vol. 446, p. 391 – 401.

Melnikov, B. and Liu, B. (2025). A parallelizable heuristic algorithm for planar triangulation with noisy data. *Cybernetics and Physics*, **14** (3), pp. 267 –274.

Melnikov, B., Radionov, A., Moseev, A., and Melnikova, E. (2006). Some specific heuristics for situation clustering problems. vol. 2, p. 272 – 279.

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.

Miao, F., Ma, X., and Li, H. (2026). High-precision parameter estimation of photovoltaic models via a novel adaptive elite-biased backtracking search algorithm. *Measurement: Journal of the International Measurement Confederation*, **258**.

Neumayer, S., Nimmer, M., Setzer, S., and Steidl, G. (2020). On the robust pca and weiszfeld's algorithm. *Applied Mathematics and Optimization*, **82** (3), pp. 1017 – 1048.

Russell, S. and Norvig, P. (2009). Artificial intelligence: A modern approach.

Sovrasov, V. and Barkalov, K. (2020). Parallel global optimization algorithm with uniform convergence for solving a set of constrained global optimization problems. In Olenev, N., Evtushenko, Y., Khachay, M., and Malkova, V., editors, *Advances in Optimization and Applications*, Cham, Springer International Publishing, pp. 38–52.

Sun, W. (2005). A scalable parallel algorithm for global optimization based on seed-growth techniques. In *Lecture Notes in Computer Science. Conference: High Performance Computing and Communications, First International Conference, HPCC 2005, Sorrento, Italy*, 09, pp. 839–844.

Venceslau, H. M., Karam Venceslau, M. B., Xavier, A. E., and Maculan, N. (2016). A geometric perspective of the weiszfeld algorithm for solving the fermat-weber problem. *RAIRO - Operations Research*, **50** (1), pp. 157 – 173.

Wang, J., MacCormac, O., Rochford, W., Kujawa, A., Shapey, J., and Vercauteren, T. (2026). Tree-based semantic losses: Application to sparsely-supervised large multi-class hyperspectral segmentation. *Lecture Notes in Computer Science*, **15967 LNCS**, pp. 584 – 594.