UDC 519.178+004.032.26 **doi: 10.32620/reks.2025.3.07**

**Olena LINNYK, Lyudmyla POLYAKOVA, Iryna ZARETSKA**

***V.N. Karazin Kharkiv National University, Kharkiv, Ukraine***

# ON KAMADA-KAWAI GRAPH LAYOUT WITH GRAPH NEURAL NETWORKS

*Traditional force-directed layout algorithms, such as the Kamada-Kawai (KK) method, are widely used for graph visualization due to their ability to produce aesthetically pleasing layouts. However, these algorithms can be computationally intensive for large graphs.* ***The subject matter*** *of the study is graph layout. The* ***aim*** *of this research is to explore the application of graph neural networks (GNNs) to improve the KK algorithm for graph layouts, resulting in a novel hybrid approach named KKNN. The key* ***tasks*** *addressed in this study include: 1. Development of the KKNN layout algorithm by integrating GNN-based reparameterization from the NeuLay-2 approach with the KK algorithm. 2. Evaluation of computational efficiency by comparing the computational performance of KKNN with both the original KK algorithm and the NeuLay-2. 3. Assessment of layout quality, particularly by examining the symmetry preservation, energy minimization, and aesthetic criteria such as minimal edge crossings and balanced node placement. 4. Testing on various graph types, including both random and highly structured (symmetric) graphs. The* ***methods*** *used are: GNN-based layout reparameterization, inspired by NeuLay-2; Kamada-Kawai graph layout algorithm; performance metrics, including time-to-convergence, energy minimization, and symmetry preservation. Our experiments demonstrate the following* ***results:*** *1. KKNN converges to the energy minimum faster and achieves a lower energy state compared to the original KK. 2. KKNN not only reduces computational time but also better preserves graph symmetries compared to NeuLay-2.* ***Conclusions.*** *This study underscores the potential of integrating neural networks with traditional graph layout algorithms, presenting a promising approach for efficient and high-quality graph visualization. KKNN not only enhances computational performance but also ensures visually interpretable layouts. This hybrid approach offers a pathway for future research in graph visualization, where combining deep learning techniques with classical algorithms may open new possibilities for handling complex, large-scale graphs in a visually coherent and computationally efficient manner.*

***Keywords:*** *network analysis; graph; layout; force-directed layout; neural networks.*

## 1. Introduction

### 1.1. Motivation

A network (or a graph) is a fundamental mathematical model that represents relationships between pairs of objects (nodes) connected by edges. Social, biological, information, transportation, and communication networks are the subjects of investigation across a wide range of scientific and engineering fields. Tackling data analysis challenges such as protein folding [1], information extraction [2], climate change [3], and COVID-19 forecasting [4, 5] and understanding [6] – requires the use of all modern algorithmic tools and machine learning techniques, including graph-theoretical methods. One key advantage of networks is their ability to be visualized. For instance, in data analysis, visualization transforms the data, given by a network, into an interpretable and insightful representation, enabling data analysts to leverage a powerful tool – namely, their own eyes.

The primary benefits of visual network representation include:

1. Understanding complex relationships, such as recognizing hidden patterns, detecting outliers, and revealing clusters, communities, and symmetries.

2. Facilitating interactive exploration, including exploratory data analysis, hypothesis generation, and experiment planning.

3. Simplifying massive and complicated datasets through dimensionality reduction, as most visualizations map network nodes to 2- or 3-dimensional space from a high-dimensional space.

To get the "drawing" of the network, layout algorithms compute the coordinates of the nodes, i.e., put the nodes into some points on the plane (2D layout) or in space (3D layout). The common opinion is that there is no best way to draw a graph, as different layouts can highlight different graph features. However, there are state-of-the-art layout algorithms – force-directed layouts or FDL – based on some physical interpretations of the network, where nodes are modeled as particles with attractive and repulsive forces acting between them or as elastic rings connected by springs along the edges.

The crucial aspects of layout algorithms are the computation time and the quality of the resulting drawing. High-quality layout criteria [7] include minimizing edge crossings; maintaining appropriate

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

distances between nodes (short for adjacent nodes and longer for non-adjacent ones), ensuring uniform node distribution, preserving inherent symmetries, and others.

Algorithms based on physical network interpretations can satisfy the aesthetic quality criteria by minimizing the total energy or the total sum of forces of the physical system. This makes FDL algorithms a standard tool for drawing graphs with up to a few thousand nodes (usually up to 1,000 nodes). For larger graphs (with more than 10,000 nodes), edge-related aesthetic criteria become less important due to the reduced visibility of individual edges. Instead, the correct placement of node communities and large topological features (like cycles or flares) becomes more significant. However, FDL algorithms are often time-consuming, requiring various computational improvement techniques such as predefined initial positions, multi-level approaches, or optimization tricks. Deep learning, specifically neural networks (NN), demonstrates strong potential for computing large graph layouts due to its ability to learn, capture, and leverage the internal structure of graphs. This paper contributes to the use of NNs for layout computation.

### 1.2. State of the Art

Originating from [8], the problem of graph drawing has led to the formation of a substantial graph drawing community, with numerous papers devoted to the subject and an annual Graph Drawing and Network Visualization symposium [9]. The state-of-the-art in graph layout algorithms is presented in [10, 11].

Note that the problem we address involves general undirected connected graphs. For specific graph types (such as hierarchical trees, labeled diagrams, or electrical circuits), there are specialized methods (see e.g. [11]). Additionally, the layout of a disconnected graph is typically obtained by combining, in a compact manner, the layouts of its connected components (see e.g. [12]).

Force-directed layout (FDL) algorithms, based on physical interpretations of graphs, are the most popular for general graph drawing. Two fundamental physical models within this category are the Kamada-Kawai (KK) spring model, considered in [13], and the force-directed placement (FDP) model, which originated from the Fruchterman and Reingold particle model, presented in [14]. The KK model treats the graph as a dynamically balanced system of rings (representing nodes) connected by springs of a specified length. Balancing the system minimizes its total energy, resulting in the optimal layout. The FDP model represents the network nodes as particles exerting attractive and repulsive forces on one another. The total energy of the physical system representing the network can be defined in various ways, leading to multiple FDL modifications. However, all these algorithms aim to compute the optimal layout by minimizing the total energy (or total force) of the system, via gradient descent or other optimization algorithms, often with techniques like Barnes-Hut optimization [15], simulated annealing [16, §10.9], or stress majorization [17]. One of the recent approaches to FDL optimization, based on the latent space model, is considered in [18].

Graph drawing with FDL is supported across various software tools designed for different tasks, including Gephi [19], igraph (in R) [20], GraphViz [21], NetworkX (in Python) [22], Cytoscape [23], and others.

The successful application of neural networks to graph layouts initially stemmed from node embeddings – algorithms designed to represent individual nodes within a graph as unique vectors in a vector space. These embeddings effectively capture both the relational and structural properties of the graph. Prominent node embedding algorithms include DeepWalk, Node2Vec, GraphSAGE, and Verse, among others [24]. Typically, these methods learn node representations from random walks (considered as the node's "context") or by aggregating information from a node's local neighborhood.

Node embeddings can be used to solve a variety of machine learning tasks, as well as to construct graph layouts. Efficient processing of networks with millions of nodes, leveraging node embeddings, is implemented in LargeVis and GraphVite approaches [25, 26]. To generate a graph layout, node embeddings can be combined with classical dimensionality reduction techniques (such as PCA, MDS, UMAP, or t-SNE) for high-dimensional embeddings, or integrated with FDL algorithms for low-dimensional ones [27].

Node embeddings were not designed exactly for visualization purposes. The direct NN application started in the late 1990s in [28], but progress stalled until Graph Neural Networks (GNNs) enabled better handling of graph data and layout criteria.

Recent research explores the use of GNNs to generate aesthetically pleasing layouts. In [29], a GNN is used to produce layouts by balancing multiple pre-specified aesthetic criteria. In [30], Graph Neural Drawers are introduced – machines that can leverage different GNNs and loss functions (including those based on aesthetic criteria) to construct efficient layouts. The DeepFD algorithm presented in [31] is based on a graph-LSTM. It takes the FDL as the prototype to design the loss function and is trained on a dataset split by the Louvain community detection algorithm.

Another approach is discussed in [32], where a graph layout is reparameterized using a GNN to optimize the steps of an FDL algorithm on this transformed layout.

The reviewed literature highlights that node embedding algorithms, such as DeepWalk, Node2Vec, provide scalable ways of encoding structural and

relational properties of nodes and inspiring neural layout algorithms like NeuLay-2, which introduced GNN-based reparameterization of FDL energy optimization, achieving significant speedups compared to classical FDL methods. This algorithm serves as a conceptual and technical foundation for extending neural techniques to other graph layout models. However, these advances have not yet been systematically applied to the Kamada–Kawai model, which is particularly well-suited for capturing graph symmetries and flexible edge-length requirements. This gap motivates our research, which aims to extend neural reparameterization methods to the KK algorithm, resulting in the hybrid KKNN approach.

### 1.3. Objectives and the Approach

The objective of this study is to explore and analyze the Kamada-Kawai graph layout technique based on neural network reparametrization (KKNN) to improve the visualization and structural organization of graphs. We extend the NeuLay-2 approach proposed by A.-L. Barabási and others in [32] to the KK algorithm. The study aims to implement the KKNN algorithm, evaluate its performance compared to the NeuLay-2 and KK, assess its efficiency and readability, and propose enhancements for better graphical representation. Our motivation to explore the KK layout algorithm is twofold:

- The KK algorithm allows for adjustment of edge lengths enabling layouts with custom edge lengths, rather than the near-uniform lengths typically produced by other force-directed layout (FDL) algorithms.
- Experiments with NeuLay-2 have mostly been conducted on random graph models. However, it is important to evaluate if the layout algorithms accurately reflect the structure of graphs with various types of symmetries.

We use such quantitative metrics for layout assessment as: average runtime of the layout algorithm as well as ratios of running times for comparison; final minimized energy of the system (loss function value) to compare NN-parametrized and non-parametrized versions of the KK algorithm; coefficient of variation for edge length distribution to confirm uniformity of the distribution; ratio of angles between adjacent edges close to the expected theoretical values, quantifying how well structural symmetries are retained.

This paper is organized as follows. Section 2 outlines the problem, reviews the FDL and KK algorithms, and introduces the idea of GNN reparametrization. Section 3 presents the experiment results and discusses them. Section 4 presents the case study of cubic lattice layouts. Section 5 provides a summary and a description of further research steps.

## 2. Methods of Research

### 2.1. Framework for GNN-based Layout Method

In this study, we propose a hybrid algorithm, KKNN, which combines the Kamada-Kawai graph layout technique with neural network reparameterization. Our method is based on the idea of Barabási and others [32], which combines one of the FDL algorithms with GNN reparameterization. In FDL algorithms, the nodes of the graph are modeled as particles or bodies in a system where forces are applied. Attractive forces are acting between adjacent nodes, while all pairs of nodes repel each other. The energy of the system is given by

$$E_{FDL} = E_a + E_r \tag{1}$$

with the energy of attractive forces given by

$$E_a = \frac{1}{2} \sum_{(v_i, v_j)} A_{ij} \left\| x_i - x_j \right\|^2, \tag{2}$$

where $x_i$ represent the positions of graph nodes and $A$ is the adjacency matrix of the graph. The repulsive energy $E_r$ in (1) is usually chosen as a rapidly decreasing function, such as in the FDP algorithm from [14]:

$$E_r = \sum_{(v_i, v_j)} \frac{r_0}{\left\| x_i - x_j \right\|}, \tag{3}$$

or as considered in [32]:

$$E_r = \sum_{(v_i, v_j)} \exp\left( -\frac{\left\| x_i - x_j \right\|^2}{4 r_0^2} \right), \tag{4}$$

where $r_0$ is a constant parameter that regulates the distance of the action of repulsive forces.

The main technique used in [32] is to express the positions of the nodes as the output of a GNN. The authors introduce the NeuLay-2 algorithm, which starts with a high-dimensional embedding and passes it through a Graph Convolutional Network (GCN). Although GNN involves many more parameters than FDL, it converges faster. The architecture of GNN is presented in

Fig. *1*.

NeuLay-2 uses two GCN layers of the form

$$G(X) = \sigma(f(A)XW) \tag{5}$$

where A is the adjacency matrix of a graph, f is the

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

aggregation function, $\sigma = \mathrm{ReLU}$ is the activation function, and W is a matrix of trainable parameters. The aggregation function f is given by a symmetrized degree-normalized adjacency matrix

$$f = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2} \quad (6)$$

where $\tilde{A} = A + I$ and $\tilde{D}$ is the degree matrix of $\tilde{A}$ with $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$.

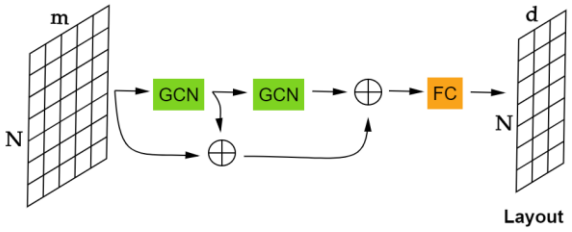



Fig. 1. NeyLay-2 architecture (represented from [32])

Initially, $Z=Z_{N\times m}$ is the embedding (usually random) of N points, corresponding to graph nodes, into m-dimensional space. The first layer takes Z as input and produces the output $G_1 = \sigma\left(f(A)ZW_1\right)$, which is then passed to the second layer of GNN, producing the output $G_2 = \sigma\left(f(A)G_1W_2\right)$. Here, matrices $W_1 \in \mathbb{R}^{m\times h_1}$ and $W_2 \in \mathbb{R}^{h_1\times h_2}$ are the weight matrices of the first and second GCN layers.

Finally, the high-dimensional embedding Z, along with the two-layer GCN outputs $G_1$ and $G_2$ are combined as $G_3 = \left[Z|G_1|G_2\right] \in \mathbb{R}^{N\times(m+h_1+h_2)}$ and passed to a fully connected (FC) layer with a linear activation function to project down to the d-dimensional layout positions $X \in \mathbb{R}^{N\times d}$. The matrix $W=[W_Z|W_1|W_2]$ from $\mathbb{R}^{(m+h_1+h_2)\times d}$ is the weight matrix of the FC layer.

The output of NeuLay-2, namely

$$X = G_3W + b = Z_WZ + G_1W_1 + G_2W_2 + b, \quad (7)$$

is then used as the input of the FDL algorithm. Instead of optimizing X directly, the NeuLay-2 parameters $\{Z, W, b\}$ are optimized.

### 2.2. KKNN Algorithm Description

In this research, we consider the Kamada-Kawai graph model, which models node-edge connections as rings connected by springs, instead of FDL, based on an attractive-repulsive forces system. The length of the spring between two nodes equals the graph-theoretic distance between them, i.e., the length of the shortest path. The total energy to be minimized is given by

$$E_{KK} = \frac{1}{2}\sum_{(v_i,v_j)} k_{ij}\left(\left\|x_i - x_j\right\| - l_{ij}\right)^2, \quad (8)$$

where $x_i$ is the position of the ring corresponding to the node $v_i$, $k_{ij}$ is the constant strength of the spring between $v_i$ and $v_j$, the length $l_{ij}$ of the spring between $v_i$ and $v_j$ corresponds to the desirable distance between them in the drawing and is proportional to $d_{ij}$ – the graph-theoretic distance between nodes $v_i$ and $v_j$ in the graph.

Consider that the attractive forces energy term in FDL-energy can be rewritten as

$$E_a = \frac{1}{2}\mathrm{Tr}\left[X^TLX\right], \quad (9)$$

where $L = D - A$ is the Laplacian of the graph with the degree matrix D and adjacency matrix A. While the KK-energy from (8) can be transformed as

$$E_{KK} = \frac{1}{2}\sum_{(v_i,v_j)} k_{ij}l_{ij}^2 - \sum_{(v_i,v_j)} k_{ij}l_{ij}\left\|x_i - x_j\right\| + \\ + \frac{1}{2}\sum_{(v_i,v_j)} k_{ij}\left\|x_i - x_j\right\|^2, \quad (10)$$

where the first term $\sum_{(v_i,\,v_j)} k_{ij}l_{ij}^2$ is a constant for the graph, independent of the layout, and the last term can be presented as

$$\frac{1}{2}\sum_{(v_i,v_j)} k_{ij}\left\|x_i - x_j\right\|^2 = \frac{1}{2}\mathrm{Tr}\left[X^TL^kX\right], \quad (11)$$

where $L^k$ is a weighted Laplacian with the ij-element given by

$$L_{ij}^k = \begin{cases} -k_{ij}, & i \neq j; \\ \sum_{i\neq t} k_{it}, & i = j. \end{cases} \quad (12)$$

Comparing energies to minimize in FDL and KK algorithms, we conclude that they have similar quadratic parts, containing (weighted) graph Laplacian, but KK-energy has another linear term, while the additional (repulsive) term of FDL-energy is not linear.

The hybrid approach KKNN is based on the same GNN reparameterization as in (5) and (6), but computes the layout positions using the KK model defined in (8),

optimized via gradient descent. The constant strength of the spring between $v_i$ and $v_j$ is taken as $k_{ij} = 1/d_{ij}^2$, as it was suggested in [17] for better drawings. Also, we consider both cases of unity edges ($l_{ij} = d_{ij} = 1$ for adjacent $v_i$ and $v_j$ non-adjacent) and weighted edges with the $l_{ij}$ taken as multiples of $d_{ij}$. Note that due to the parameter $l_{ij}$ adjustments in (8) the KK algorithm is capable to generate layouts with edges of the required lengths.

The steps of layout computation are as follows:

1. Nodes embedding initialization. We use the initialization of node positions in $d$-dimensional space ($d = 2$ for planar layouts and $d = 3$ for spatial) within the Kaiming uniform distribution.
2. GNN-based reparametrization as in (7).
3. KK energy model application by considering the loss function as KK-energy (8).
4. Minimizing the KK energy function via gradient descent with respect to GNN trainable parameters.
5. Evaluation and visualization.

## 3. Results and Discussion

In this section, we compare the performance of the KKNN algorithm with the original KK layout algorithm and NeuLay-2, both by speed and by performance.

The implementation of NeuLay-2 was taken from [33]. The KK algorithm was implemented by the authors based on [13] to ensure a fair comparison. Experiments were performed on hardware equipped with a CPU (Intel Core i5-9400F, 8GB of RAM). Due to hardware limitations, the tests were restricted to graphs with up to 3300 nodes and 7000 edges. For the same reason, we report only relative comparisons of running times, omitting absolute values in seconds, since GPU acceleration could potentially speed up the layout calculations. The absolute values can be imagined by the following: the running time ranged from 1.5–220 s for KKNN and 1.9–840 s for NeuLay-2 (for 100–3300 nodes).

To evaluate the algorithms' running times, we used several graph generation models, including both random and highly symmetric graphs. The random models include the Barabási–Albert preferential attachment model [34] (denoted on figures as "ScaleFree"), Watts–Strogatz small-world graphs [35] (denoted as "SmallWorld" ), Random Geometric Graphs [36] (denoted as "Geometric"), and Internet Autonomous System networks [37] (denoted as INet(Internet)). By varying parameters, we generate families of random graphs with node sets ranging from 100 to 3000 nodes and diverse topological properties. The symmetric graphs (denoted on figures as "Symmetric") include planar quadratic and hexagonal lattices, balanced and binary trees, as well as cubic and pyramidal lattices, and a quadratic lattice folded into a tube, which is best represented using a 3D layout. Overall, more than 25 graphs were included in the experiment. The realizations of these models are obtained from NetworkX [22] or created directly by the authors (pyramid, hexagonal lattices, tube graphs) and can be referenced in [38].

The layouts of symmetric graphs produced by KKNN are demonstrated in

Fig. 2: a) Tube with 1000 nodes and 1980 edges; b) HGrid (hexagonal grid) with 930 nodes and 1350 edges; c) BalT (balanced tree) with 3280 nodes and 3279 edges; d) WeightedCone10 with 220 nodes and 990 edges. The edges of the last graph have decreasing lengths from top to bottom.

In [32], it is stated that NeuLay-2 offers a 10- to 100-fold improvement in speed compared to FDL. To assess the difference between KKNN and NeuLay-2 speeds, we measured the average running time of both algorithms and evaluated the ratio of KKNN's average running time (timeKKNN) to that of NeuLay-2 (timeNeuLay2).

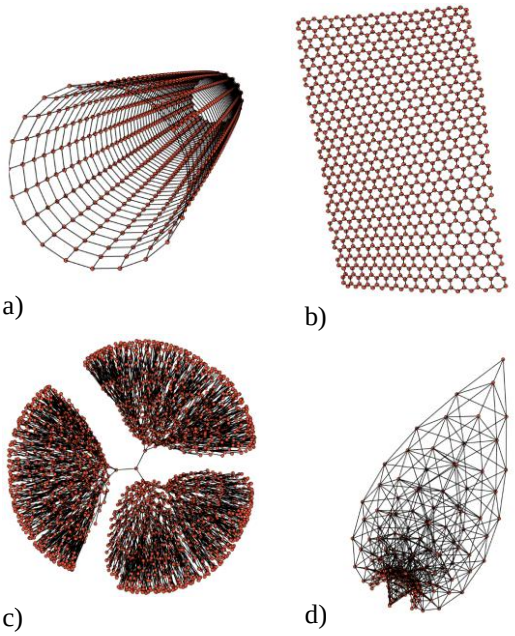



Fig. 2. Layouts in 3D by KKNN obtained for
a) Tube; b) HGrid; c) BalT(3,7); d)WeightedCone10

The experiments demonstrate that the KKNN algorithm calculates layout faster than NeuLay-2, with a running time ratio ranging from 0.2 to 0.9 and a mean ratio value of 0.53. This means that KKNN computes the layout on average twice as fast. The relative running times for some of the experimental graphs are shown in Figure 3a. The points, labeled by graphs with different shapes, correspond to various graph types. The ratio of

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

KKNN execution time to Neulay-2 execution time is plotted along the ordinate axis. As can be seen, the ratio is less than 1 for all considered graphs. It may also be noted that speedup increases with the number of nodes. For graphs with more than 1000 nodes, KKNN to NeuLay-2 average improvement becomes 0.45. The other observation is that average ratios in geometric and symmetric graphs are smaller than in ScaleFree, INet, and SmallWorld models: 0.49 vis 0.57.

As shown above, FDL and KK energies have similar quadratic terms (see (9), (11)), but KK energy has an additional linear term, while FDL energy has a nonlinear one (see (3), (10)). This may cause faster processing of KK energy by the neural network.

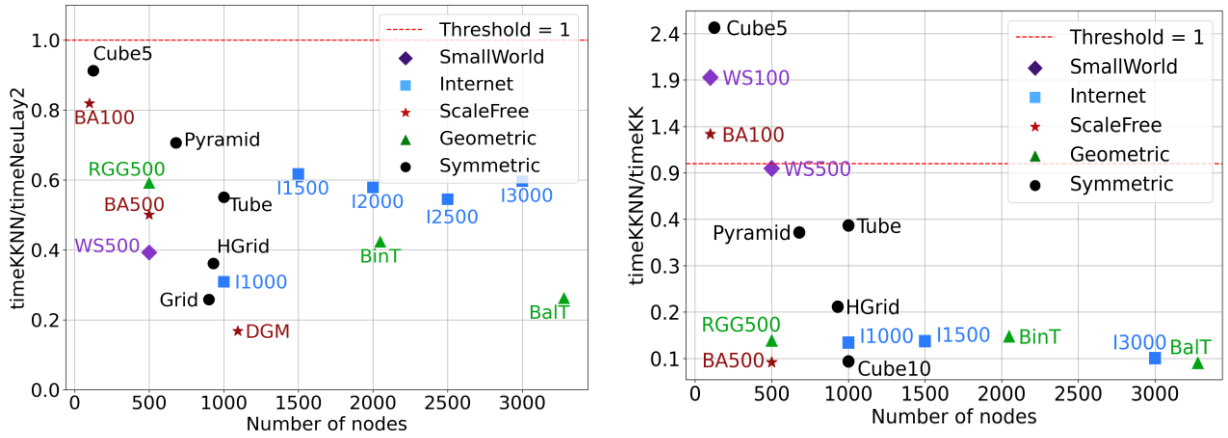
The reasons for the speed-up of NN algorithms compared to non-NN in calculating layouts for different random graph models are discussed for NeuLay in [32]. The authors analytically demonstrated that outlier eigenvalues of the adjacency matrix accelerate layout calculation. The same reasoning applies to the KKNN.

To compare KKNN with the original KK, we investigate the behavior of the loss (energy) function depending on the number of iterations of the KK algorithm. The experiment demonstrates that KKNN converges to the energy minimum faster than KK, and the minimum achieved is deeper than that of KK. For all graphs, the final energy in the case of the KKNN layout was lower, on average by 10%. For all graphs with more than 300 nodes (except the Pyramid lattice with 680 nodes), KKNN converged 1.2 to 7 times faster, on average by 3.4 times. An example plot of the energy descent process is demonstrated in Section 4.

The running time comparison of KK and KKNN was made in the same manner as for KKNN and Neulay-2. It indicates that for graphs with a small or medium number of nodes, the original KK algorithm has a shorter running time. However, for large graphs, neural network reparametrization in KKNN speeds up the layout calculation. Running time ratios for graphs with more than 500 nodes are all under 1 and have a mean value of 0.45. The relative running time for some of the experimental graphs can be seen in Figure 3b). The graphs of type INet show consistent KKNN improvement with up to 10 times faster than KK.

Next, we investigate the symmetry performance of the KKNN and NeuLay-2 algorithms. We consider graphs with different symmetries: square, cubic, hexagonal, and tetrahedral pyramid lattices, as well as regular polyhedra, and examine the distributions of edge lengths and angles between edges. The coefficient of variation $CV = \frac{\sigma}{\mu}$, where $\mu$ is a mean value and $\sigma$ is a standard deviation for edge length distribution (ELCV) is used to measure the uniformity of the distribution. Also, we calculate the ratio of angles between adjacent edges close to the expected theoretical values (AR). The expected theoretical values for square and cubic lattices are multiples of $90^0$, for tetrahedral pyramid and hexagonal lattices – the multiples of $60^0$, for regular polyhedra $\frac{360^0 \cdot (n-2)}{n}$, where $n$ is the number of face nodes. The tolerance was taken as $5^0$. Table 1 presents the results of measurements for some graphs: the cubical lattice (Cube06) with 2744 nodes, the square grid (Grid) with 900 nodes, the hexagonal grid (HGrid) with 930 nodes, which can be seen in Figure 2, b), and the tetrahedral pyramid (Pyramid) with 680 nodes.

Overall, more than 10 graphs with up to 3000 nodes and up to 7000 edges were tested. The results demonstrate that KKNN produces layouts with almost equal edges (ELCV is close to 0 for all experimental graphs), while the lengths of edges, generated by Neulay-2, vary a lot (0.2 < ELCV < 0.35). The angle distribution in the KKNN case is close to expected (AR is close to 1), but for Neulay-2, angles are noisy (0.18 < AR < 0.4).




a) b)

Fig. 3. KKNN compared with a) NeuLay-2 and b) KK by running time.
The ratios of timeKKNN to timeNeuLay2 and timeKK are pointed out.
The points are labeled by graphs and marked by different types

Table 1

Edge length uniformity (ELCV) and angles preservation (AR) for NeuLay-2 and KKNN

| Graph | ELCV | | AR | |
|---|---|---|---|---|
| | NeyLay | KKNN | Neylay | KKNN |
| Cube14 | 0.35 | **0.03** | 0.10 | **0.94** |
| Grid | 0.23 | **0.02** | 0.14 | **0.96** |
| HGrid | 0.26 | **0.01** | 0.30 | **0.96** |
| Pyramid | 0.21 | **0.03** | 0.40 | **0.99** |

Example calculations of theoretical expected angle values, as well as histograms of edge length and angle distributions are demonstrated in Section 4.

The effectiveness of KKNN in symmetry preservation may be caused by the fact that many distances between nodes in well-represented lattice-like layouts are measured along straight lines and are, in fact, equal to the graph-theoretic distances between these nodes, as assumed in the KK algorithm.

## 4. Case study

Regular network and lattice layout computation is an important tool, e.g., for chemical compound or crystal lattice simulation tools (see [39]). In this section, we discuss in detail the layout of cubic lattices.

An $n$-cubic lattice (Cube_n) may be defined as a graph whose nodes are integer points $(i, j, k)$ with $0 \le i, j, k < n$ and edges connecting each node to its immediate neighbor in one coordinate direction (i.e., a node $(i, j, k)$ is connected to the nodes obtained by increasing exactly one of its coordinates by 1). An $n$-cubic lattice has $n^3$ nodes and $3n^2(n-1)$ edges.

A canonical symmetric 3D layout of a cube lattice may place the nodes into integer points $(i, j, k)$ and draw edges along grid lines. Taking into account the boundary "faces" and "edges" of a cube lattice, one can immediately calculate the total number of angles in the canonical layout as $12n^2(n+1)$ for $90^0$ and as $3n(n^2-1)$ for $180^0$. The numbers of nodes, edges, right and straight angles for cubic lattices are given in Table 2.

We calculate the layouts of Cube_n (for $n = 6,8,10,12,14$) and investigate the execution time, energy descent, and symmetry preservation, as well as give a visualization of layouts to compare the layout with a canonical one.

The results of the KKNN, the NeuLay-2, and the KK layouts comparison for cubic lattice graphs are as follows. The KKNN converges faster to the energy minimum compared to KK (on average, iteration 200 vis 450). KKNN achieves the lower minimum (10% lower on average) on cubic lattices. The example energy descent curves for Cube10 are shown in Figure 4.

Table 2

Characteristics of $n$-cubic lattices

| n | Nodes number | Edges number | Number of $90^0$ angles | Number of $180^0$ angles |
|---|---|---|---|---|
| 6 | 216 | 540 | 3024 | 630 |
| 8 | 512 | 1344 | 6912 | 1512 |
| 10 | 1000 | 2700 | 13200 | 2970 |
| 12 | 1728 | 4752 | 22464 | 5148 |
| 14 | 2744 | 7644 | 35280 | 8190 |

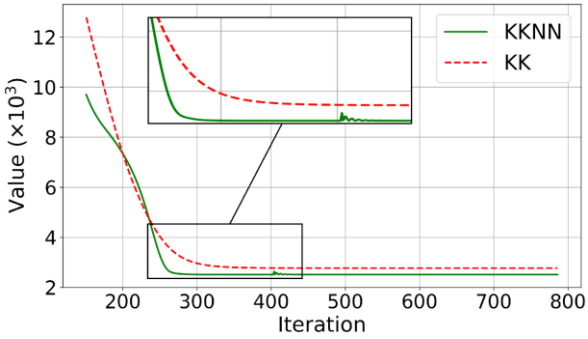
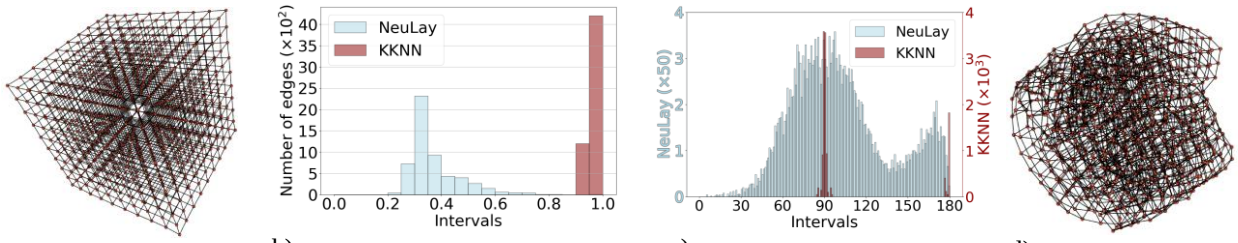



Fig. 4. KKNN compared with KK by energy descent

The KKNN to Neulay-2 improvement in running time is on average 70%. The length uniformity ELCV is 0.03 on average compared to 0.26 for NeuLay-2, and the expected angles ratio AR is 0.98 on average compared to 0.15 for NeuLay-2. The visual comparison of NeyLay-2 and KKNN layout quality, as well as edge length and angle distributions, can be done in Figure 5.

The results for cubic lattices confirm those obtained for other graphs.




a) b) c) d)

Fig.5. Symmetry performing by KKNN and NeuLay-2 for Cube10: a) KKNN layout; b) edge length distribution; c) angles between edges distribution; d) NeuLay-2 layout

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

## 5. Conclusions

The main contribution of this research lies in the proposal, implementation, and assessment of the KKNN algorithm – a hybrid approach for graph layout calculation. The basis of the KKNN is the previously known NeuLay-2 algorithm, which was constructed to accelerate FDL algorithms.

Our results demonstrate that KKNN improves the efficiency of the traditional KK algorithm by reducing the time required to reach the energy minimum and achieving a lower overall energy state. Compared to NeuLay-2, KKNN performs better in terms of speed and the preservation of graph symmetries, especially for relatively large and complex networks. The KKNN can be used for calculating the layout of medium-sized graphs (up to 10,000 nodes), especially those including symmetrical structures (grids, regular networks, graphs for chemical compounds, topological models of the data), to reduce the layout algorithm's working time while preserving the aesthetic and readability.

Future research directions are as follows:

– exploring the application of different neural network models and more advanced architectures for layout optimization;

– applying the mechanism of accelerating convergence by NN-reparametrization to other optimization problems on graphs (e.g., resource balancing, optimal layouts in electric circuits and chip designs, systems equilibrium positions determination);

– investigating the effectiveness of NN layout algorithms in presenting other graph properties, such as cluster separation, preserving of community structure, sparseness, or denseness visualization.

Overall, this research contributes to the growing field of NN-based graph drawing and opens new avenues for efficient and accurate visualization of networks.

**Contributions of authors:** conceptualization, methodology – **Lyudmyla Polyakova, Iryna Zaretska;** formulation of tasks, analysis of results – **Lyudmyla Polyakova;** development of model, software, verification, visualization – **Olena Linnyk;** writing – **Olena Linnyk, Lyudmyla Polyakova, Iryna Zaretska.**

### Conflict of Interest

The authors declare that they have no conflict of interest in relation to this research, whether financial, personal, authorship, or otherwise, that could affect the research and its results presented in this paper.

### Financing

This study was conducted without financial support.

### Data Availability

All data supporting the plots presented in this paper, other findings of the study, as well as the source code, are available in the data repository [38].

### Use of Artificial Intelligence

The authors confirm that they did not use artificial intelligence methods while creating the presented work.

### Acknowledgments

We thank Vadym Kaidalov for his help with software adjustment and code optimization.

All the authors have read and agreed to the published version of this manuscript.

## References

1. Jumper, J., Evans, R., Pritzel, A., Green, P., Figurnov, D., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S., Ballard, A., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., Back, B., Petersen, S., Reiman, D., Clancy, E., Zielinski, M., Steinegger, M., Pacholska, M., Berghammer, T., Bodenstein, S., Silver, D., Vinyals, O., Senior, A.W., Kavukcuoglu, K., Kohli, P., & Hassabis, D. Highly accurate protein structure prediction with AlphaFold. *Nature*, 2021, vol. 596, pp. 583–589. DOI: 10.1038/s41586-021-03819-2.

2. Zhou, D., Li, S., Dong, L., Chen, R., Peng, X., & Yao, H. C-KGE: Curriculum learning-based Knowledge Graph Embedding. *Computer Speech & Language*, 2025, vol. 89, article no. 101689. DOI: 10.1016/j.csl.2024.101689.

3. Liu, Q., Kim, Y., & Hemsley, J. Climate Change Skeptics and the Power of Negativity. *Proceedings of the Association for Information Science and Technology*, 2024, vol. 61, iss 1, pp. 999-1001. DOI: 10.1002/ pra2.1166.

4. Chumachenko, D., Meniailov, I., Bazilevych, K., Chumachenko, T., & Yakovlev, S. Investigation of Statistical Machine Learning Models for COVID-19 Epidemic Process Simulation: Random Forest, K-Nearest Neighbors, Gradient Boosting. *Computation*, 2022, vol. 10, no. 6, article no. 86. DOI: 10.3390/ computation10060086.

5. Mohammadi, A., Meniailov, I., Bazilevych, K., Yakovlev, S., & Chumachenko, D. Comparative study of linear regression and SIR models of COVID-19 propagation in Ukraine before vaccination. *Radioelectronic and computer systems*, 2021, no. 3, pp. 5–18. DOI: 10.32620/reks.2021.3.01.

6. Drach, K., Glushakov, S., & Kotenko, I. Understanding the spread of COVID-19 in the United States using topology and machine learning for time series. *Proceedings of the Pharmaceutical Users Software Exchange 2020 (Phuse US Connect, November 8-11, 2020, virtual).* Available at lexjansen.com/

phuse/2020/rw/PAP_RW03.pdf (accessed 31 August 2024).

7. Di Bartolomeo, S., Crnovrsanin, T., Saffo, D., Puerta, E., Wilson, C., & Dunne, C. Evaluating Graph Layout Algorithms: A Systematic Review of Methods and Best Practices. *Computer Graphics forum*, 2024, vol. 43, iss. 6, article no. e15073. DOI: 10.1111/cgf. 15073.

8. Tutte, T. W. How to draw a graph. *Proc. London Math. Soc.*, 1963, vol. 3, iss. 1, pp. 743-768. DOI: 10.1112/ PLMS/S3-13.1.743.

9. *Drawing G.: Graph drawing website*, 2024. Available at: http://www.graphdrawing.org/ (accessed 31.08.2024).

10. Gibson, H., Faith, J., & Vickers, P. A survey of two-dimensional graph layout techniques for information visualization. *Information Visualization*, 2013, vol. 12, pp. 324-357. DOI: 10.1177/ 1473871612455749.

11. Pinki, P., & Shekhawat, K. An annotated review on graph drawing and its applications. *AKCE Int. Journal of Graphs and Combinatorics*, 2023, vol. 20, iss. 3, pp. 258-281. DOI: 10.1080/09728600.2023. 2218459.

12. Freivalds, K., Dogrusoz, U., & Kikusts, P. Disconnected graph layout and the polyomino packing approach. *Proc. Symp. Graph Drawing GD'01* in *Lecture Notes in Computer Science*, 2002, vol. 2265, pp. 378-391. Available at: https://link.springer.com/chapter/10.1007/3-540-45848-4_30 (accessed 31.08.2024)

13. Kamada, T., & Kawai, S. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 1989, vol. 31, iss. 1, pp. 7-15. DOI: 10.1016/0020-0190(89)90102-6.

14. Fruchterman, T. J. & Reingold, E. M. Graph drawing by force-directed placement. *Software: Practice and Experience*, 1991, vol. 21, iss. 11, pp. 1129-1164. DOI: 10.1002/spe.4380211102.

15. Barnes, J. & Hut, P. A hierarchical O(NlogN) force-calculation algorithm. *Nature*, 1986, vol. 324, iss. 6096, pp. 446-449. DOI: 10.1038/324446A0.

16. Press, W. H., Teukolsky, S. A., & Flannery, B. P. *Numerical Recipes in C.* Second ed. Cambridge, USA, Cambridge University Press, 1992. 996 p.

17. Gansner, E. R., Koren, Y., & North, S. Graph Drawing by Stress Majorization. In: J. Pach, ed. *Graph Drawing*. Berlin, Heidelberg: Springer, 2005. pp. 239-250. DOI: 10.1007/978-3-540-31843-9_25.

18. Gaisbauer, F., Pournaki, A., Banisch, S., & Olbrich, E. Grounding force-directed network layouts with latent space models. *Journal of Computational Social Science*, 2023, vol. 6, iss 2, pp. 707-739. DOI: 10.1007/s42001-023-00207-w.

19. Bastian, M., Heymann, S., & Jacomy, M. Gephi: An Open Source Software for Exploring and Manipulating Networks. *International AAAI Conference on Weblogs and Social Media*, 2009, vol. 3, iss. 1, pp. 361-362. DOI: 10.1609/icwsm.v3i1.13937

20. Csardi, G., & Nepusz, T. The igraph software package for complex network research. *InterJournal, Complex Systems*, 2006, vol. 1695, iss. 5., pp. 1-9. Available at https://igraph.org/ (accessed 01.06.2024)

21. Ellson, J., Gansner, E., Koutsofios, L., North, S., & Woodhull, G. Graphviz – Open Source Graph Drawing Tools. In: P. Mutzel, M. Junger & S. Leipert, eds. *Graph Drawing*, Springer Berlin Heidelberg, 2002, pp. 483--484. Available at: https://graphviz.org/ (accessed 31.08.2024).

22. Hagberg, A., Swart, P., & Chult, D. Exploring network structure, dynamics, and function using NetworkX, *Los Alamos National Lab. (LANL)*, Los Alamos, NM (United States), 2008. Available at: https://networkx.org/ (accessed 31.08.2024).

23. Shannon, P., & et al. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome research*, 2003, vol. 13, iss. 11, pp. 2498-2504. Available at: https://cytoscape.org/ (accessed 31.08.2024).

24. Khosla, M., Setty, V., & Anand, A. A Comparative Study for Unsupervised Network Representation Learning, 2020, Available at https://arxiv.org/abs/1903.07902 (accessed 31.08.2024).

25. Tang, J., Liu, J., Zhang, M., & Mei, Q. Visualizing Large-scale and High-dimensional Data, 2016. Available at https://arxiv.org/abs/1602.00370 (accessed 31.08.2024). DOI: 10.1145/2872427. 2883041.

26. Zhu, Z., Xu, S., Qu, M., & Tang, J. GraphVite: A High-Performance CPU-GPU Hybrid System for Node Embedding. In: *The World Wide Web Conference*, 2019, pp. 2494-2504. DOI: 10.1145/3308558.3313508.

27. Shen, L., Tai, Z., Shen, E., & Wang, J. Graph Exploration with Embedding-Guided Layouts, 2023. Available at: https://arxiv.org/abs/2208.13699 (accessed 31.08.2024).

28. Cimikowski, A., & Shope, P. A neural network algorithm for a graph layout problem. *IEEE Trans Neural Netw.*, 1996, vol. 7, iss. 2, pp. 341-345. DOI: 10.1109/ 72.485670.

29. Wang, X., Yen, K., Hu, Y., & Shen, H.-W. DeepGD: A Deep Learning Framework for Graph Drawing Using GNN, 2021. Available at https://arxiv.org/abs/2106.15347 (accessed 01.06.2024).

30. Tiezzi, M., Ciravegna, G., & Gori, M. Graph Neural Networks for Graph Drawing. *IEEE Transactions on Neural Networks and Learning Systems*, 2024, vol. 35, iss. 4, pp. 4668–4681. DOI: 10.1109/tnnls.2022. 3184967.

31. Zhang, S., Xu, R., Zhang, Q., Quan, Y., & Liu, Q. DeepFD: a deep learning approach to fast generate force-directed layout for large graphs. *Journal of Visualization*, 2024, vol. 27, pp. 925–940. DOI: 10.1007/s12650-024-00991-1.

32. Both, C., Dehmamy, N., Yu, R., & Barabási, A.-L. Accelerating network layouts using graph neural networks. *Nature Communications*, 2023, vol. 14, article no. 1560. DOI: 10.1038/s41467-023-37189-2.

33. Both, C. *NeuLay*, 2023. Available at: https://github.com/csabath95/NeuLay (accessed 01.06.2024).

34. Barabási, A.-L., & Albert, R. Emergence of Scaling in Random Networks. *Science*, 1999, vol. 286,

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

iss. 5439, pp. 509-512. DOI: 10.1126/science. 286.5439.509.

35. Watts, D. J., & Strogatz, S. H. Collective dynamics of ‘small-world’ networks. *Nature*, 1998, vol. 393, pp. 440-442. DOI: 10.1038/30918.

36. Penrose, M. *Random Geometric Graphs.* Oxford, Oxford University Press, 2003. 344 p.

37. Elmokashfi, A. M., Kvalbein, A., & Dovrolis, C. On the Scalability of BGP: The Role of Topology Growth. *IEEE Journal on Selected Areas in Communications*, 2010, vol. 28, pp. 1250-1261. DOI: 10.1109/JSAC.2010.101003.

38. Linnyk, O. *KamadaNN*, 2024. Available at: https://github.com/OlenaLinnyk/KamadaNN (accessed 31 August 2024).

39. Thompson, A. P., Aktulga, H. M., Berger, R., Bolintineanu, D. S., Brown, W. M., Crozier, P. S., Veld, P. J., Kohlmeyer, A., Moore, S. G., Nguyen, T. D., Shan, R., Stevens, M. J., Tranchida, J., Trott, C., & Plimpton, S. J. LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales. *Comp Phys Comm*, 2022, vol. 271, article no. 10817. DOI: 10.1016/j.cpc.2021.108171.

*Received 01.10.2024, Accepted 25.08.2025*

## ПРО УКЛАДАННЯ ГРАФІВ АЛГОРИТМОМ КАМАДА-КАВАЇ ЗА ДОПОМОГОЮ ГРАФОВИХ НЕЙРОННИХ МЕРЕЖ

***О. С. Лінник, Л. Ю. Полякова, І. Т. Зарецька***

Традиційні алгоритми силоспрямованого укладання, такі як метод Камада-Каваї (КК), широко використовуються для візуалізації графів завдяки їхній здатності створювати естетично привабливі зображення. Однак ці алгоритми можуть бути обчислювально складними для великих графів. **Предметом** цього дослідження є укладання графів. **Метою** цього дослідження є застосування графових нейронних мереж (GNN) для вдосконалення алгоритму КК для укладання графів, результатом чого є новий гібрідний підхід під назвою KKNN. Основні **завдання** цього дослідження включають: 1. Розробку алгоритму укладання KKNN шляхом інтеграції репараметризації на основі GNN з підходу NeuLay-2 з алгоритмом КК. 2. Оцінку обчислювальної ефективності через порівняння продуктивності KKNN та оригінального алгоритму КК, а також NeuLay-2. 3. Оцінку якості укладання, зокрема, аналіз збереження симетрії, мінімізації енергії та естетичних критеріїв, таких як мінімізація перетину ребер і збалансоване розташування вершин. 4. Тестування на різних типах графів, включаючи як випадкові, так і структуровані (симетричні) графи. Використані такі **методи**, як репараметризація укладання на основі GNN з алгоритму NeuLay-2; алгоритм укладання графа Камади-Каваї та визначення показників продуктивності, включаючи час досягнення мінімуму, мінімізацію енергії та збереження симетрії. Наші експерименти демонструють наступні **результати**: 1. KKNN швидше збігається до мінімуму енергії та досягає меншого локального мінімуму енергії порівняно з вихідним КК. 2. KKNN не тільки скорочує час обчислень, але й краще зберігає симетрії графів порівняно з NeuLay-2. **Висновки.** Це дослідження підкреслює потенціал інтеграції нейронних мереж з традиційними алгоритмами укладання графів, пропонуючи перспективний підхід для ефективної та високоякісної візуалізації графів. KKNN не тільки покращує обчислювальну продуктивність, але й забезпечує візуально інтерпретовані укладання. Цей гібридний підхід відкриває можливості для майбутніх досліджень у сфері візуалізації графів, де поєднання методів глибокого навчання з класичними алгоритмами може створити нові перспективи для обробки складних, великомасштабних графів у візуально узгодженій й обчислювально ефективній формі.

**Ключові слова:** аналіз мереж; граф; укладання графу; силоспрямоване укладання; нейронні мережі.

**Лінник Олена Степанівна** – магістрантка каф. теоретичної і прикладної інформатики, Харківський національний університет імені В.Н. Каразіна, Харків, Україна.

**Полякова Людмила Юріївна** – канд. фіз.-мат. наук, доц. каф. теоретичної і прикладної інформатики, Харківський національний університет імені В.Н. Каразіна, Харків, Україна.

**Зарецька Ірина Тимофіївна** – канд. фіз.-мат. наук, доц. каф. теоретичної і прикладної інформатики, Харківський національний університет імені В.Н. Каразіна, Харків, Україна.

**Olena Linnyk** – Student, Department of Theoretical and Applied Computer Science, V.N. Karazin Kharkiv National University, Kharkiv, Ukraine, e-mail: linnyk2021mf12@student.karazin.ua.

**Lyudmyla Polyakova** – PhD in Physics and Mathematics, Associate Professor of the Department of Theoretical and Applied Computer Science, V.N. Karazin Kharkiv National University, Kharkiv, Ukraine,
e-mail: l.yu.polyakova@karazin.ua, ORCID: 0000-0002-6674-1958, Scopus Author ID: 23028761300.

**Iryna Zaretska** – PhD in Physics and Mathematics, Associate Professor of the Department of Theoretical and Applied Computer Science, V.N. Karazin Kharkiv National University, Kharkiv, Ukraine,
e-mail: zaretskaya@karazin.ua, ORCID: 0000-0001-8747-2737.