DRGraph: An Efficient Graph Layout Algorithm for Large-scale Graphs by Dimensionality Reduction

Minfeng Zhu, Wei Chen, Yuanzhe Hu, Yuxuan Hou, Liangjun Liu, and Kaiyuan Zhang

Abstract—Efficient layout of large-scale graphs remains a challenging problem: the force-directed and dimensionality reduction-based methods suffer from high overhead for graph distance and gradient computation. In this paper, we present a new graph layout algorithm, called DRGraph, that enhances the nonlinear dimensionality reduction process with three schemes: approximating graph distances by means of a sparse distance matrix, estimating the gradient by using the negative sampling technique, and accelerating the optimization process through a multi-level layout scheme. DRGraph achieves a linear complexity for the computation and memory consumption, and scales up to large-scale graphs with millions of nodes. Experimental results and comparisons with state-of-the-art graph layout methods demonstrate that DRGraph can generate visually comparable layouts with a faster running time and a lower memory requirement.

Index Terms—graph visualization, graph layout, dimensionality reduction, force-directed layout

1 INTRODUCTION

Graphs are common representations to encode relationships between entities in a wide range of domains, such as social networks [64], knowledge graph [59], and deep learning [62]. Node-link diagrams is an efficient method to depict the overall structure and reveal inter-node relations [36]. Nevertheless, the layout influences the understanding of the graph. For instance, it is typical to assume that two close nodes have high proximity even though no links exist among them [41]. Therefore, preserving the neighborhood is an essential concept of graph layout.

Over the past 50 years, numerous efforts have been exerted on graph layout. However, an efficient graph layout algorithm remains a challenging problem for large-scale data. Representatives include force-directed algorithms [16,31] and dimensionality reduction methods [34]. The force-directed methods solve the graph layout problem using a physical system with attractive and repulsive forces between nodes. Although force-directed methods are simple and easy to implement [3], they have a high computational complexity in calculating pairwise forces (the least one is $O(|V|^2 \log |V| + |E|)$) [22], where $|V|$ denotes the number of nodes and $|E|$ indicates the number of edges). However, preserving distances between pairs of widely separated nodes may result in a large contribution to the cost function. Thus, they are not good at preserving local structures [40] and may converge to local minima and unpleasing results [54].

As an alternative, studies applied dimensionality reduction methods, such as multidimensional scaling (MDS) [35], principal component analysis (PCA) [30] and t-distributed stochastic neighbor embedding (t-SNE) [40] for graph layout [5, 23, 34]. They usually minimize the difference between the node similarity (e.g., the shortest path distance) in the graph space and the layout proximity (e.g., Euclidean distance) in the layout space [65]. Nonlinear dimensionality reduction methods aim to preserve the local neighborhood structure which is analogous to the concept of graph layout. Though these methods can produce aesthetically pleasing results, they suffer from the high computational and memory complexity. For instance, tsNET [34] adopts t-SNE to capture local structures. However, tsNET is amenable for graphs with only a few thousand nodes due to the following reasons: (1) the computational complexity of the shortest path distance is $O(|V|^2(|V| + |E|))$; (2) pairwise node similarities require $O(|V|^2)$ computations; (3) the gradient requires $O(|V|^2)$ distances for pairwise layout proximities.

In this paper, we propose a new graph visualization algorithm, DRGraph, that enhances the dimensionality reduction scheme to achieve the efficient layout of large graphs. Our approach differs from conventional dimensionality reduction algorithms in three aspects. First, we utilize a sparse graph distance matrix to simplify the computation of node similarities by only taking the shortest path distances between a node and its neighbors into consideration. Second, we employ the negative sampling technique [43] to compute the gradient on the basis of a subset of nodes, efficiently reducing the computational complexity. Further, we present a multi-level process to accelerate the optimization process. By integrating these three techniques, DRGraph achieves a linear complexity for the computational and memory consumption (namely, $O(|E| + |V|)$ and $O(|E| + |V|)$, where $T$ denotes the number of iterations and $M$ is the number of negative samples).

We present a multi-threaded implementation of DRGraph and evaluate our DRGraph on various datasets quantitatively and qualitatively. Generally, the single-thread version of DRGraph is roughly two times faster than GPU-accelerated tsNET while producing comparable layouts on moderate-sized graphs. For large-sized graphs, DRGraph yields more expressive results, whereas tsNET cannot handle them without special optimizations. DRGraph runs at a comparable speed like FM$^3$ [22] and can preserve more topologically neighbors. For the Flan_1565 graph with 1,564,794 nodes and 56,300,289 edges, DRGraph consumes only 7 GB memory, whereas FM$^3$ requires approximately 44 GB memory. Thus, DRGraph can easily scale up to large graphs with millions of nodes on commodity computers. The source code of DRGraph is available at https://github.com/ZJUVAG/DRGraph.

2 RELATED WORK

2.1 Dimensionality Reduction

Dimensionality reduction methods convert a high-dimensional dataset into a low-dimensional space. As a fundamental means for visualization, dimensionality reduction has been applied in a broad range of fields and ever-increasing datasets [47]. Classical techniques include PCA [30], Sammon mapping [53] and MDS [35]. Researchers employ linear discriminant analysis [14] to reveal label information when data have associated class labels. However, linear dimensionality reduction fails to detect nonlinear manifolds in high-dimensional space. Nonlinear dimensionality reduction algorithms aim to preserve local structures of nonlinear manifolds. Isomap [56] estimates the geodesic distance instead of Euclidean distance to minimize the pairwise distance error.

Recently, stochastic neighbor embedding (SNE) [24] based approaches transform Euclidean distance into probability to measure similarities among the data points. $r$-distributed stochastic neighbor embedding (r-SNE) is proposed to solve the crowding problem [40]. Though r-SNE shows its significant advantage in generating low-dimensional...
embedding, high computational complexity prevents it from being applied to large datasets. Barnes-Hut-SNE (BH-SNE) [57] reduces the computational complexity from $O(N^2)$ to $O(N \log N)$ by leveraging a tree-based method. Tang et al. [55] presented LargeVis to construct the $k$-nearest neighbor graph and accelerated optimization using the negative sampling technique [43]. A-tSNE [47] progressively computes the approximated $k$-nearest neighborhoods and updates the embedding without restarting the optimization. Nowadays, GPUs are widely employed for further acceleration [7, 48]. Though GPGPU-SNE [48] has a linear computational complexity, t-SNE-CUDA [7] outperforms GPGPU-SNE due to the highly-optimized CUDA implementation. Given the non-convexity of the objective function, t-SNE-based algorithms may end up in local minima and unpleasing layouts. The multi-level concept has been widely used to address this problem [1]. The multi-level representation is created by clustering [42], decomposition [44], anchor point [29], and Monte Carlo process (e.g., HSNE) [46]. However, these methods suffer from high computation cost for generating the multi-level representation. DRGraph introduces an enhanced multi-level scheme with a linear computational complexity.

2.2 Graph Layout

Graph layout algorithms map nodes of a given graph to 2D or 3D positions [38]. The goal is to compute positions for all nodes according to the topological structure of the given graph. Graph layout algorithms can be categorized into two classes, namely, force-directed and dimensionality reduction-based methods [21].

Force-directed methods. Most graph layout methods adopt the force-directed drawing algorithm because they are simple to understand and easy to implement. There are two main classes of force-directed methods: spring-electrical and energy models [21].

The spring-electrical model assigns attractive and repulsive forces between nodes. The model moves each node along the direction of the composition force until the composition force on each node is zero. Eades replaced nodes by steel rings and replaced edges with springs [13]. To draw nodes evenly, Fruchterman and Reingold [16] modeled nodes as atomic particles and added repulsive forces between all nodes. However, previous algorithms are time-consuming to visualize large-scale graphs due to high computational complexity. Thus, previous studies employed simulated annealing techniques to optimize the spring-electrical model [15, 26]. ForceAtlas2 [28] combines an adaptive-cooling schedule and a local temperature technique to produce continuous layouts. To further expedite spring-electrical methods, the computational complexity of attractive and repulsive forces must be reduced. Researchers adopt Barnes-Hut technique to accelerate the force calculation [26]. Multi-level method [17, 19, 22] has been used widely in many graph layout methods. An initial layout is generated for the next larger graph that is drawn afterward [42, 68].

The energy model formulates the graph layout problem as an energy system and optimizes the system by searching a state with minimum energy [18]. A previous study generated a graph layout by solving the partial differential equations on the basis of the energy function [50]. The concept of the KK algorithm proposed by Kamada and Kawai [31] is that Euclidean distances in the layout space should approximate graph-theoretic distances, i.e., the shortest path distance. Incremental methods [9] accelerate the optimization by arranging a small portion of the graph before arranging the rest. Stress majorization is employed to improve the computation speed and graph layout quality [20]. Stress function can be reformulated to draw graphs with various constraints [11, 25, 61], including length, non-overlap, and orthogonal constraints [32, 51, 66]. Pivot MDS [5] first places anchor nodes and then locates other nodes on the basis of their distances to anchor nodes.

Dimensionality reduction based methods. Graph layout by dimensionality reduction aims to preserve graph structures. Utilizing dimensionality reduction techniques to study graph layout requires further exploration. Recent works [34, 39, 65] pursued this line of thought and illustrated how to use dimensionality reduction for graph layout. Graph layout by dimensionality reduction can be classified into projection and distance-based methods.

Projection-based methods have two stages: first, embed graph nodes into a high-dimensional space and then project vectors into low-dimensional space. High-dimensional embedding (HDE) [23] adopts PCA to project the graph. Koren et al. [33] improved HDE by replacing PCA with subspace optimization. Zaorálek et al. [67] compared several different dimensionality reduction methods for graph layout. More recently, powerful deep neural networks are also utilized to learn how to draw a graph from training examples [37, 60]. However, the pairwise similarity loss of deep-learning methods commonly has a quadratic computational complexity with respect to the number of nodes. Distance-based methods adopt graph-theoretic distance instead of the distance in high-dimensional space used by projection-based methods. s-SNE [39] is developed by considering spherical embedding and resolves the “crowding problem” by eliminating the discrepancy between the center and the periphery. tsNET [34] utilizes neighborhood-preserving t-SNE technique for graph layout. Dimensionality reduction approaches with high efficiency can be employed to accelerate tsNET. The single-thread version of DRGraph is faster than tsNET accelerated by the t-SNE-CUDA algorithm [7]. DRGraph optimizes the objective function with the negative sampling technique [43] which reduces the computational complexity to linear. Also, we employed an efficient multi-level representation to propagate gradient information and draw graphs from coarse to fine.

3 Method

3.1 Background on Graph Layout with tsNET

Our approach takes a similar framework as that of tsNET [34]. Formally, let $G = (V,E)$ be an undirected unweighted graph with a set of nodes $V = \{v_1, v_2, ..., v_{|V|}\}$ and a set of edges $E = \{e_1, e_2, ..., e_{|E|}\}$. Each edge is a connection between two nodes: $e = (v_i, v_j) \in V \times V$. Then, the graph layout problem is formulated as embedding a given graph to 2D or 3D space: $\phi : G \rightarrow Y, Y = \{y_1, y_2, ..., y_{|V|}\}$, where $y_i$ is the layout position of node $v_i$.

Graph layout methods are tied by an optimization problem [65] that minimizes the difference between the graph space and the layout space. The node similarity (NS) is defined as the pairwise similarity between two nodes in the graph space. The layout proximity (LP) is defined as the pairwise proximity between two nodes in the layout space. Connected nodes with high node similarity should preserve high layout proximity in the layout space. A loss function $D(NS,LP)$ models the difference between the node similarity and the layout proximity. The Kullback-Leibler divergence $D_{KL}(\cdot \| \cdot)$ formulates the graph layout problem as optimizing the following objective function:

$$Y^* = \arg \min_y D_{KL}(NS||LP) = \arg \min_y \sum y_{ij} \log y_{ij} - NS_{ij} \log LP_{ij},$$

where $NS_{ij}$ is the node similarity between $v_i$ and $v_j$, and $LP_{ij}$ is the layout proximity between $y_i$ and $y_j$, and $Y^*$ is the optimal graph layout. Given that the first term is a constant, the problem is equivalent to the following optimization problem:

$$Y^* = \arg \max_y \sum y_{ij} \log LP_{ij},$$

The node similarity is computed by graph-theoretic distance $GD$ in the graph space. We compute a graph distance matrix by leveraging the shortest path distance (SPD) using a breadth-first search: $GD_{ij} = SPD(v_i,v_j)$. $SPD(v_i,v_j)$ is the shortest path distance between nodes $v_i$ and $v_j$. The node similarity matrix $NS$ is given by transforming the graph distance using a similarity function (e.g., Gaussian distribution):

$$NS_{ij} = \frac{\exp(-GD_{ij}^2/2\sigma^2)}{\sum_{i,j} \exp(-GD_{ij}^2/2\sigma^2)}$$

and

$$NS_{ij} = (NS_{ij} + NS_{ji})/(2|V|),$$

where $\sigma$ is the variance of Gaussian distribution on node $v_i$.

The layout proximity is measured by the layout distance $LD$ between nodes’ positions in the layout space. We can compute layout
distance $LD$ using Euclidean distance: $LD_{ij} = \|y_i - y_j\|^2$. Then, the layout proximity $LP$ is measured by a proximity function (e.g., Student’s $t$-distribution). The proximity function captures important locality properties in the layout space, providing an appropriate scale to connect the node similarity and the layout proximity. The layout proximity of the pair $(y_i, y_j)$ in the layout space can be formulated as follows:

$$LP_{ij} = \frac{1}{1 + LD_{ij}} - 1,$$

(5)

where $LP_{ij}$ denotes the layout proximity, $LD_{ij}$ denotes the layout distance between $y_i$ and $y_j$, and $b$ is a parameter to control the shape of the distribution. When $b = 1$, $LP$ is equivalent to the normalized Student’s $t$-distribution (a single degree of freedom) used by tsNET.

tsNET modifies the objective function with two additional cost terms, and tsNET* further assigns initial values by Pivot MDS (PMDS) [5]. The tsNET algorithm is useful for neighborhood preservation. However, tsNET is amenable for graph data with only a few thousand nodes due to the following reasons. First, tsNET must measure graph distances between all node pairs to construct node similarity. All pairwise shortest path distances require $O(|V|(|V| + |E|))$ computations using the breadth-first search. Second, computing the node similarity needs $O(|V|^2)$ computations, because computing the normalization terms needs to sum over $|V|^2$ graph distances. Third, the gradients require $|V|^3$ pairs of Euclidean distances in each iteration. Thus, tsNET has a quadratic computational and memory complexity:

$$C_{\text{computation}}^{\text{tsNET}} = O(|V|(|V| + |E|) + |V|^2 + |E||V|)$$

$$C_{\text{memory}}^{\text{tsNET}} = O(|V|^2)$$

where $T$ is the number of iterations.

### 3.2 DRGraph

We seek to overcome the performance overhead of tsNET in three aspects. Particularly, our approach utilizes a sparse graph distance matrix to simplify the computation of pairwise node similarities, the negative sampling technique to approximate the gradient on the basis of a subset of nodes and a multi-level process to accelerate the optimization process. By integrating them into a new pipeline, called DRGraph, a linear complexity for the computation and memory consumption is achieved. Figure 1 compares the framework of tsNET and DRGraph. In addition to the new layout pipeline, three new components are highlighted in blue font, namely, sparse distance matrix, multi-level layout, and negative sampling. The details are elaborated as follows.

#### 3.2.1 Sparse Distance Matrix

To reduce the computation and memory requirements of the node similarity, we propose to approximate the node similarity using a sparse representation without a significant effect on the layout quality. This scheme works due to the following observations. First, the node similarity of two nearby nodes with a small shortest path distance is relatively high according to the definition (Eq. 3). Second, the node similarity between widely separated nodes is almost infinitesimal. Therefore, a small shortest path distance has a significant contribution to the objective function. We utilize a sparse distance matrix to simplify the computation of pairwise node similarities by using the shortest path distance between a node and its neighbors (see Figure 2).

We define $k$-order nearest neighbors $NNG(v_i, k)$ of node $v_i$ as a set of nodes whose shortest path distances to $v_i$ are less than or equal to $k$:

$$NNG(v_i, k) = \{v | SPD(v_i, v) \leq k, v \neq v_i, v \in V\}.$$  

(8)

For instance, first-order nearest neighbors $NNG(v_i, 1)$ is the set of nodes connected to $v_i$. We can compute a sparse distance matrix where $GD_{ij} = SPD(v_i, v_j)$ if $v_j \in NNG(v_i, k)$. Eq. 3 is reformulated as:

$$NS_{ji} = \begin{cases} \frac{\exp(-GD_{ij}^2/2\sigma^2)}{\sum_{v_j \in NNG(v_i, k)} \exp(-GD_{ji}^2/2\sigma^2)}, & \text{if } v_j \in NNG(v_i, k) \\ 0, & \text{otherwise} \end{cases}$$

(9)

The worst case of finding a node’s neighborhoods is exploring all edges in $O(|E|)$. To find the $k$-order nearest neighbors, a breadth-first search must access $\min(|E|, (|E|/|V|)^k)$ nodes, where $|E|/|V|$ denotes the average degree [52]. Therefore, we can generate a sparse graph distance matrix $GD$ by finding the $k$-order nearest neighbor set in $O(D|V|)$. $D = \min(|E|, (|E|/|V|)^k)$. Measuring and storing node similarity with a large nearest neighbor set for graphs with millions of nodes is infeasible due to the memory limitation. In most instances, first-order nearest neighbors are sufficient to capture neighborhood information and produce pleasing graph layouts. We can generate $GD$ in $O(|E|)$ time by leveraging the first-order nearest neighbor set.

For node similarity $NS$, the value of the graph distances $GD_{ij}$ ranges from 1 to $k$. We can pre-compute the Gaussian distribution for $k$ different values and measure $NS$ with $O(k|V|)$ calculations. Thus, computing the node similarity using the sparse graph distance matrix has a $O(k|V|)$ computational complexity. In particular, the node similarity can be measured in $O(|V|)$ if we employ the first-order nearest neighbors.

#### 3.2.2 Negative Sampling

We employ the negative sampling technique [43] to approximate the gradient using a small set of nodes. We sample one positive node and $M$ negative nodes for each gradient computation. We use a logistic regression to separate one positive node $y_i$ and $M$ negative nodes $y_{jm}, m = 1, \ldots, M$. The likelihood function is defined as follows:

$$\log LP_{ij} + \sum_{m=1}^{M} Y_i \log (1 - LP_{ij})$$

(10)
in which $\gamma$ is a weight assigned to the negative samples. In this way, the gradient of each node needs $M + 1$ Euclidean distances. We randomly sample the positive node $y_j$ on the basis of the edge probability $NS_{ij}$. We identify the negative sample $y_{in}$ according to the node weight $\sum_i NS_{ij}$. We reformulate the optimization problem as follows:

$$D(NS|LP) = \sum_{(i,j) \sim NS_i} \log LP_{ij} + \sum_{m=1}^{M} \gamma \log(1 - LP_{im})$$  \hspace{1cm} (11)$$

The partial derivative of the objective function (Eq. (11)) is derived as:

$$\frac{\partial D}{\partial y_i} = -\frac{2b(y_i - y_j)^{2b-1}}{1 + LD(y_i)} + \sum_{m=1}^{M} \frac{2b(y_i - y_{in})^{2b-1}}{1 + LD(y_{in})}$$ \hspace{1cm} (12)$$


The gradient shows that each node receives one attractive force and $M$ repulsive forces. During optimization, we randomly select a node and compute the gradient of the node. Each gradient computation takes $O(M)$ time, where $M$ is the number of negative samples. Consequently, the negative sample technique reduces distance calculation from $O(|V|)$ to $O(M)$ for the gradient computation of each node. We define an iteration as computing gradient $|V|$ times. In practice, we find that the number of iterations is usually constant. The computational complexity of the optimization is $O(TM|V|)$, where $T$ denotes the iteration number. Therefore, the objective function can be effectively optimized by the stochastic gradient descent algorithm in linear time.

### 3.2.3 Multi-level Layout Scheme

The multi-level approach has been used widely in many graph layout methods [17, 22]. It starts from a coarse graph layout and iteratively optimizes to a refined layout. We design a multi-level scheme to generate a multi-level representation in linear time. Our scheme comprises three steps: coarsening, layout of the coarsest graph, and refinement.

**Coarsening.** The coarsening step generates a series of coarse graphs $G^0, G^1, G^2, ..., G^L$ with decreasing sizes, where $G^0$ is the original graph and $G^L$ is the coarsest graph. Given a graph $G^l = (V, E^l)$, we generate a coarser graph $G^{l+1}$ as follows. First, we randomly select a node $v^l$ (red nodes in Figure 3). Then, we assign $v^l$ and its first-order neighbors (nodes in blue regions) into a new node in $G^{l+1}$. Third, we delete edge $(v^l, v^j) \in E^l$ in the graph $G^l$, if $v^l$ and $v^j$ are assigned into the same node in $G^{l+1}$. This process is repeated until no nodes can be assigned. The coarsening step reduces the number of nodes at each level. However, when the size of $G^{l+1}$ is very close to $G^l$, the cost of the multi-level algorithm significantly increases [22] and the size of the coarsest graphs cannot be further reduced. Therefore, it is sufficient to cease the coarsening process when $|V^{l+1}| > \rho |V^l|$. We choose $\rho$ to be 0.8 to achieve the balance between the computational efficiency and the global structure extraction [26].

**Refinement.** Once we generate the graph layout of a coarse graph $G^{l+1}$, the initial layout of the finer graph $G^l$ is derived from $G^{l+1}$. We set the position of a node $v^l$ in $G^l$ to be the position of node $v^{l+1}$ in $G^{l+1}$ if $v^l$ is assigned to $v^{l+1}$ in the graph coarsening step. Then, we recursively refine the layout until we complete the finest graph $G^0$.

Conventional multi-level techniques require computing new node similarities for each level if we draw each graph level individually. We optimize the graph layout jointly by sharing the gradient through the multi-level representation. We pre-compute the node similarity between nodes of $G^l$ just once before the optimization. For the layout of $G^l$, we select node $v^l$ from $G^l$ and compute the gradient of node $v^l$. We forward the computed gradient of node $v^l$ to node $v^j$ in $G^l$ if $v^l$ is assigned to $v^j$ in the graph coarsening step.

The running time of the multi-level scheme denotes the time of creating a series of coarse graphs: $\sum_{l=1}^{L} t_{create}(G^l)$. The worst case of creating $G^l$ from $G^{l-1}$ is accessing all nodes and edges in $O(|V^l| + |E^l|)$. Let us assume that $|V^l| \leq 0.8|V^{l-1}|$ and $|E^l| \leq 0.8|E^{l-1}|$ for all $l = 1, ..., L$, then $\sum_{l=1}^{L} t_{create}(G^l) = (|V^l| + |E^l|) (1 + \frac{1}{8} + \frac{1}{64} + \ldots + (\frac{1}{8})^L) \leq 5(|E| + |V|)$. The computational complexity is linear in $O(|V| + |E|)$.

The running time of the multi-level scheme denotes the time of creating a series of coarse graphs. $\sum_{l=1}^{L} t_{create}(G^l)$. The worst case of creating $G^l$ from $G^{l-1}$ is accessing all nodes and edges in $O(|V^l| + |E^l|)$. Let us assume that $|V^l| \leq 0.8|V^{l-1}|$ and $|E^l| \leq 0.8|E^{l-1}|$ for all $l = 1, ..., L$, then $\sum_{l=1}^{L} t_{create}(G^l) = (|V^l| + |E^l|) (1 + \frac{1}{8} + \frac{1}{64} + \ldots + (\frac{1}{8})^L) \leq 5(|E| + |V|)$. The computational complexity is linear in $O(|V| + |E|)$.

### 3.2.4 Complexity Analysis

**Computational complexity.** The computational complexity of our algorithm includes $k$-order nearest neighbor set searching $O(D|V|)$, node similarity computation $O(k|V|)$, coarse graph generation $O(|V| + |E|)$ and optimization $O(TM|V|)$. The total computational complexity of DRGraph is derived as:

$$c_{\text{DRGraph}}^{\text{computation}} = O(D|V| + k|V| + |E| + TM|V|),$$ \hspace{1cm} (13)$$

where $D = \min(|V|, (|E|/|V|)^k)$. DRGraph achieves a linear computational complexity of $O(|E| + |V| + TM|V|)$ if we employ the first-order nearest neighbor set ($k=1$).

**Memory consumption.** DRGraph requires $(D|V|)$ memory to store the similarity of sparse nodes, $O(|V| + |E|)$ memory to record all coarse graphs, and $O(|V|)$ memory to store the layout position of nodes. The total memory consumption of DRGraph is derived as:

$$c_{\text{DRGraph}}^{\text{memory}} = O(D|V| + |V| + |E|),$$ \hspace{1cm} (14)$$

We can reduce the memory complexity of DRGraph to $O(|E| + |V|)$ if we employ the first-order nearest neighbor set ($k=1$).

### 3.3 DRGraph versus tsNET

One limitation of tsNET is that its computational and memory complexities are quadratically proportional to the graph size. Contrastingly, DRGraph yields a linear computational complexity and only requires a linear memory consumption to store similarity and coarse graphs. DRGraph and tsNET are not guaranteed to converge to the global optimum due to the non-convexity objective function of t-SNE. This not only needs to modulate several parameters but also easily converges to local minima. In addition, a different random initialization may lead to a different graph layout. Though tsNET initializes the layout using PMDS, the result remains unpleasing if PMDS fails to maintain the global structure given a small number of pivots. DRGraph adopts the multi-level scheme to coarsen graphs and capture the global structure progressively. DRGraph can find the optimal initial layout using the coarsest graph. Besides, tsNET may distort the PMDS layout since PMDS preserves short and long shortest-path distances, which conflicts with the neighbor-preserving nature of tsNET. DRGraph successively refines graph layouts from the coarsest to the original one, resulting in no distortions between coarse graphs.
4 Results

In this section, we evaluate the efficiency and effectiveness of DRGraph. We conduct all experiments on a desktop PC with Intel(R) Core(TM) i7-6700 CPU, 64 GB memory, and Ubuntu 16.04 installed.

Datasets. We perform experiments on a broad range of datasets selected from the University of Florida Sparse Matrix Collection [10] and tsNET [34] (Table 1).

| Dataset          | #Nodes | #Edges | Description                |
|------------------|--------|--------|----------------------------|
| dwt72            | 72     | 75     | planar structure           |
| lespis           | 77     | 254    | collaboration network      |
| can_96           | 96     | 336    | mesh                       |
| rajal11          | 135    | 377    | miscellaneous network      |
| jazz             | 198    | 2,742  | collaboration network      |
| visbrazil        | 222    | 336    | tree-like network          |
| grid17           | 289    | 544    | grid                       |
| mesh3e1          | 289    | 800    | grid                       |
| netscience       | 379    | 914    | collaboration network      |
| dwt419           | 419    | 1,572  | planar structure           |
| price_1000       | 1,000  | 999    | tree-like network          |
| dwt1005          | 1,005  | 3,800  | planar structure           |
| cage8            | 1,015  | 4,994  | miscellaneous network      |
| bcstk09          | 1,083  | 8,677  | grid                       |
| block_2000       | 2,000  | 9,912  | clusters                   |
| sierpinski3d     | 2,050  | 6,144  | miscellaneous network      |
| CA-GrQc          | 4,158  | 13,422 | collaboration network      |
| EVA              | 4,475  | 4,652  | collaboration network      |
| 3elt             | 4,720  | 13,722 | 3D mesh                    |
| us_powergrid     | 4,941  | 6,594  | miscellaneous network      |
| G65              | 8,000  | 16,000 | 3D torus                   |
| fe_3elt2         | 11,143 | 32,818 | 3D mesh                    |
| bcstk31          | 32,715 | 572,914| 3D automobile component   |
| venkat50         | 62,424 | 827,671| 3D mesh                    |
| ship_003         | 121,728| 1,827,654| 3D ship                   |
| troll            | 213,453| 5,885,829| 3D structure              |
| web-NoteDame     | 325,729| 1,469,679| web graph                 |
| Flan_1565        | 1,564,794| 56,300,289| 3D steel flange          |
| com-Orkut        | 3,072,441| 117,185,083| online social network    |
| com-LiveJournal  | 3,997,962| 34,681,189| online social network    |

Methods. We compare DRGraph with seven widely used graph layout algorithms. We choose spring-electrical approach (FR [16]), energy-based approaches (KK [31] and Stress Majorization [20]), multi-level methods (FM [22] and SFDP [26]), and landmark-based algorithm (PMDS [5]), because they are representatives of well-established approaches. tsNET [34] is the state-of-the-art graph layout approach that best preserves neighborhood information. The implementations of FR, KK, Stress Majorization (S.M.), FM, and PMDS are gathered from OGDF-2018-03-28 [8]. tsNET [34] is provided by the authors. We accelerate tsNET with a GPU-based t-SNE implementations [6]. We employ the SFDP implementation of the GraphViz library. We repeat the experiments five times to remove the random effects.

Parameters. After a preliminary evaluation, we set the number of negative samples to be 5, γ=0.1, and the total number of iterations to be 400. DRGraph approximates the node similarity using the first-order nearest neighbors. Due to the space limit, we discuss parameter sensitivity in the supplementary material. We use the pre-set parameters for other methods.

4.1 Evaluation Metrics

We employ neighborhood preservation (NP), stress, crosslessness and minimum angle metrics to evaluate the graph layout quantitatively.

Neighborhood preservation. NP is defined as the Jaccard similarity coefficient between the graph space and the layout space:

\[
NP = \frac{1}{|V|} \sum_{i \in V} \frac{|NNG(v_i, k_{eval}) \cap NNL(y_i, k_{eval})|}{|NNG(v_i, k_{eval}) \cup NNL(y_i, k_{eval})|},
\]

where \(NNG(v_i, k_{eval})\) denotes the \(k_{eval}\)-order nearest neighborhoods of node \(v_i\) in the graph space and \(NNL(y_i, k_{eval})\) is the \(k_{eval}\)-nearest neighbors (\(k_i = |NNG(v_i, k_{eval})|\)) of node \(y_i\) in the layout space. We evaluate the accuracy of neighborhood preservation with \(k_{eval} = 2\) [34].

Stress. The normalized stress measures how the graph layout fits theoretical distributions. For fair comparisons, we find a scalar \(\alpha\) to minimize the full stress:

\[
stress = \min \frac{1}{|V|^2} \sum_{0 \leq i < j \leq |V|} w_{ij}(\alpha |y_i - y_j| - SPD(v_i, v_j))^2. \tag{16}
\]

We use the conventional weighting factor of \(w_{ij} = 1/SPD(v_i, v_j)^2\).

Crosslessness. The crosslessness aesthetic metric [49] encourages graph layout methods to minimize the number of edge crosses. Inspired by it, we define the crosslessness as:

\[
crosslessness = \begin{cases} 
1 - \frac{c}{c_{max}}, & \text{if } c_{max} > 0 \\
1, & \text{otherwise}
\end{cases} \tag{17}
\]

where \(c\) is the number of crossings and \(c_{max}\) is the approximated upper bound on the number of edge crosses.

Minimum angle. The minimum angle metric quantifies the average deviation of the actual minimum angle from the ideal angle [49]:

\[
\min_{\text{angle}} = 1 - \frac{1}{|V|} \sum_{v \in V} \theta(v) - \theta_{\text{min}}(v) |\theta(v) - \frac{360}{\text{degree}(v)}| \tag{19}
\]

where \(\theta_{\text{min}}(v)\) is the actual minimum angle at node \(v\).

4.2 Selection of Parameters

The size of \(k\)-order nearest neighbors. Higher-order nearest neighbors contain many dissimilar nodes, which are treated as positive nodes by the negative sampling technique. Thus, as shown in Figure 4, DRGraph places dissimilar nodes close to one another resulting in a low graph layout quality. In addition, higher-order nearest neighbors cost large memory consumptions for keeping graph distances. Therefore, we choose first-order nearest neighbors, which is sufficient to provide locality properties, accelerate the computation of the node similarity, and meanwhile reduce the memory requirement.

![Figure 4. The effect of the size of \(k\)-order nearest neighbors.](image_url)
Table 2. Time consumptions (second) of different graph layout algorithms.

| Datasets      | FR  | KK  | S.M. | FM  | SFDP | PMDS | tsNET | DRGraph |
|---------------|-----|-----|------|-----|------|------|-------|---------|
| dwt,72        | 0.006 | 0.003 | 0.011 | 0.018 | 0.006 | 0.001 | 1.727 | 0.007 |
| lemsis        | 0.008 | 0.003 | 0.010 | 0.009 | 0.009 | 0.001 | 1.776 | 0.007 |
| can,26        | 0.010 | 0.005 | 0.016 | 0.012 | 0.010 | 0.001 | 1.767 | 0.009 |
| rajat11       | 0.016 | 0.010 | 0.025 | 0.021 | 0.015 | 0.002 | 1.772 | 0.015 |
| jazz          | 0.053 | 0.026 | 0.064 | 0.063 | 0.038 | 0.010 | 1.800 | 0.019 |
| visbrazil     | 0.035 | 0.026 | 0.077 | 0.046 | 0.052 | 0.008 | 1.753 | 0.026 |
| grid17        | 0.059 | 0.046 | 0.121 | 0.058 | 0.030 | 0.014 | 1.818 | 0.085 |
| meshk3el      | 0.062 | 0.046 | 0.127 | 0.065 | 0.031 | 0.012 | 1.793 | 0.079 |
| netscience    | 0.100 | 0.081 | 0.214 | 0.102 | 0.095 | 0.015 | 1.844 | 0.039 |
| dwt,419       | 0.128 | 0.102 | 0.262 | 0.101 | 0.054 | 0.018 | 1.793 | 0.043 |
| price,1000    | 0.627 | 0.665 | 1.480 | 0.204 | 0.283 | 0.040 | 1.941 | 0.121 |
| dwt,1000      | 0.663 | 0.686 | 1.500 | 0.141 | 0.152 | 0.048 | 1.955 | 0.111 |
| cage8         | 0.687 | 0.728 | 1.522 | 0.161 | 0.162 | 0.058 | 1.992 | 0.111 |
| bcustk9       | 0.836 | 0.850 | 1.718 | 0.166 | 0.175 | 0.061 | 2.013 | 0.282 |
| block,2000    | 2.663 | 3.050 | 6.174 | 0.390 | 0.382 | 0.131 | 2.217 | 0.224 |
| sierpinsk3d   | 2.682 | 3.007 | 6.309 | 0.317 | 0.312 | 0.092 | 2.078 | 0.235 |
| CA-GgQc       | 10.18 | 13.44 | 26.24 | 1.015 | 0.953 | 0.222 | 3.440 | 0.335 |
| EVA           | 12.45 | 15.11 | 30.01 | 0.781 | 1.330 | 0.191 | 3.218 | 0.493 |
| jelt          | 14.03 | 16.89 | 33.74 | 0.849 | 0.858 | 0.235 | 2.843 | 0.563 |
| us,powergrid  | 15.07 | 18.21 | 36.54 | 1.168 | 0.979 | 0.237 | 3.071 | 0.702 |
| G65           | 40.01 | 53.59 | 92.75 | 1.477 | 1.427 | 0.374 | 3.481 | 1.079 |
| fe,Aelt2      | 77.88 | 136.6 | 183.3 | 1.975 | 2.243 | 0.563 | 5.533 | 1.482 |
| bcustk31      | 792.6 | 3916 | 2358 | 8.171 | 13.08 | 5.446 | 8.928 | 2.674 |
| venkat50      | 2395 | 6848 | 14.92 | 22.55 | 8.213 | 17.29 |
| ship,003      | 9023 | 156 | 36.86 | 56.17 | 22.86 | 36.79 |
| troll         | (-) | (-) | (-) | 63.14 | 122.6 | (-) | 66.35 |
| Web-NoteDame  | (-) | (-) | (-) | (-) | 35.74 | (-) | 111.9 |
| Flam,1565     | (-) | (-) | (-) | 623.8 | 1395 | 490.7 | 823.5 |
| com-Orkut     | (-) | (-) | (-) | (-) | 4444 | (-) | 1994 |
| com-LiveJournal | (-) | (-) | 3068 | 7269 | 1644 | (-) | 2943 |

Figure 5. The effect of the \( b \) parameter.

The weight of negative samples \( \gamma \). \( \gamma \) controls the value of repulsive forces of the gradient. A small value of \( \gamma \) generates small repulsive forces, whereas natural clusters in the graph data tend to form groups. Thus, it is easier for similar nodes to move to one another in the early optimization process. DRGraph employs the early exaggeration technique to find a better solution. We set \( b = 0.01 \) for the coarsest graph to decrease the repulsive forces and form separated clusters. Then, we increase \( b \) for other coarse graphs and place nodes evenly for preserving visualization. NP, crosslessness, and minimum angle metrics slightly drop when \( \gamma \) is small. A large \( \gamma \) leads to a bad stress quality. We choose a medium value of 0.1 for finer graphs.

The iteration number \( T \). The layout quality becomes stable when the iteration number \( T \) is large adequately. We choose the iteration number \( T = 400 \) to generate comparable results.

The effect of \( b \). Figure 5 (a) illustrates the sum of the attractive and the repulsive forces (i.e., the gradient defined in Eq. 12) with respect to \( b \). \( b \) controls the value of the sum force without altering the ideal distance between nodes. Generally, a small \( b \) value (e.g., \( b = 1 \)) tends to place nodes close to others and generates localized clusters (Figure 5 (b)). A large \( b \) value (e.g., \( b = 3 \)) forces all edge lengths to be ideal but obscures the global structure (Figure 5 (d)). For 3D meshes (e.g., G65) and large social networks (e.g., com-Orkut), preserving all edge lengths of a manifold into the 2D space is intractable. Therefore, we set \( b = 1 \) to preserve the neighborhood identity for these graphs. We choose \( b = 3 \) when the input graph is a grid graph (e.g., grid17), in which all edges have the same length. For other graphs, we choose \( b = 2 \), which works well in preserving local and global structures.

4.3 Performance

Running time. Table 2 reports the running time of graph visualization process. For all approaches, the running time only includes the layout time without data process steps. We employ the single-thread version of DRGraph for a fair comparison. Unfilled items indicate the incapability of the corresponding algorithm caused by the huge memory consumption or computational cost. For small datasets, most graph layout methods perform comparably to each other. Especially, the single-thread version of DRGraph is faster than GPU-accelerated tsNET. For large datasets, FM\(^3\), DRGraph, and PMDS are much more efficient than others. PMDS is the fastest method due to the number of pivots used. Our DRGraph runs faster than PMDS on the com-Orkut dataset. The performance of PMDS is severely affected by the number of edges. For the Flam,1565 and com-LiveJournal datasets, DRGraph, FM\(^3\), SFDP and PMDS are comparable in terms of the running time.
FM3 and SFDP fail to visualize the com-Orkut dataset due to their huge memory consumptions. Though multilevel-based graph layout method FM3 achieves comparable performance on large-scale datasets, DRGraph requires less memory consumption and generates results with better NP than FM3 (see Section 4.4).

The parallel implementation of optimization enables further acceleration on a multi-core platform. Figure 6 plots the speedups in terms of the number of threads. Generally, the speedups increase with data sizes and the number of threads. The largest overall speedup (4.84×) is obtained by eight threads on the Flan_1565 dataset. DRGraph reduces the running time of Flan_1565 from 823.5 seconds to 171.1 seconds using eight threads. For the com-Orkut dataset, DRGraph spends much more time on graph coarsening, leading to a slightly small speedup (3.74×).

**Memory consumption.** Table 3 compares the memory usages. The memory usage denotes the maximum usage of the process during its lifetime. Energy models such as KK and S.M. are huge consumers of memory because they require quadratic memory complexity to store pairwise graph distances. DRGraph only consumes 7 GB memory to visualize Flan_1565 with 1,564,794 nodes and 56,300,289 edges. Contrarily, FM3 requires approximately 44 GB. Fundamentally, DRGraph achieves a linear complexity of memory consumption O(|E| + |V|) and scales up to large graphs with millions of nodes.

### 4.4 Graph Layout Quality

Tables 4, 5, 6, and 7 compare NP, stress, crossness, and minimum angle metrics of different graph layout algorithms. FR and KK produce a poor layout quality on large graphs, because they easily converge to
local minima and can hardly preserve the graph structure. DRGraph and tsNET are superior to other methods in terms of NP due to the local structure preservation nature of t-SNE. DRGraph performs slightly better than tsNET on graphs with regular structures (e.g., grid17 and 3elt). DRGraph obtains a worse layout quality than tsNET on e.g.8 and CA-GRQc. This is because the negative sampling technique cannot easily identify local and global structures of these irregular graphs. The gap of NP metric between tsNET and DRGraph is small indicating that our method achieves a comparable layout quality. In addition, DRGraph can achieve a better stress quality compared to tsNET. The stress metric of DRGraph and FM$^3$ is better on small graphs than the results obtained by PMDS. FM$^3$ reaches the best stress quality on large datasets. This is not surprising because our algorithm does not optimize the stress. DRGraph has a better NP quality than FM$^3$ and SFDP on almost all graphs. Moreover, DRGraph and SFDP achieve the best performance in terms of the crosslessness and the minimum angle aesthetic metrics. Ultimately, DRGraph achieves a comparable layout quality to FM$^3$, SFDP and tsNET.

4.5 Visualization Results

Figures 7 and 8 show representative graph layouts. Due to the space limit, more examples are given in supplementary material. We draw all graphs in Python using the Matplotlib library [27]. We compute the edge length from the layout space and use a red-to-green-to-blue color map to visualize the distribution of edge lengths. The shortest edge is in red, and the longest edge is in blue. Other edges are colored according to the scale. The node-link diagram suffers from the limited screen space and possible visual clutter for visualizing large-scale datasets. Thus, we randomly sample a subset of the edge set to reduce the visual clutter for graphs with more than 600,000 edges. Generally, DRGraph achieves visually readable layouts (see Figure 8). Force-directed methods, such as FR and KK cannot generate layouts clearly for large datasets (e.g., G65 and troll). We can see that tsNET and DRGraph achieve aesthetically pleasing results with clear structures. FM$^3$, SFDP, PMDS, and DRGraph usually produce better layouts than other methods on large datasets because they employ a multi-level or landmark approach to compute a better initial layout. Results of DRGraph exhibit clearer clustering structures compared with those by FM$^3$, SFDP, and PMDS on large social networks (e.g., com-Orkut). In Figure 7, we visualize users of the com-Orkut dataset by leveraging DRGraph. Different colors encode different ground-truth communities. We filter communities that have less than 800 users. Roughly speaking, there are four visible groups. Users from the same community tend to form tight clusters around the group center. However, communities located in group A are closely connected to other users without labels. Therefore, communities in the center of group A are visually indistinguishable from each other.

5 Discussions

Scalability. Methods such as KK, Stress Majorization (S.M.), and tsNET require $O(|V|^2)$ time to compute all pairwise Euclidean distances and $O(|V|^2)$ memory to store distances. They are not applicable to large-scale data. Contrarily, DRGraph achieves a linear complexity for the computation and memory consumption, and can be applied to datasets with millions of nodes.

Robustness. Many graph layout methods are only appropriate for limited types of graphs. DRGraph generates satisfying graph layouts on almost all datasets with appropriate parameters. We analyze the performance of DRGraph in planar, hierarchical, social, and tree-like graphs. DRGraph can easily achieve good results with a similar configuration of parameters without a significant parameter modification.

Generalizability. Graph layout methods can be unified as an optimization problem [65]. The difference between graph layout methods lies in the selection of node similarity, layout proximity, and distribution distance functions. Various configurations yield distinctive graph layout approaches. For instance, DRGraph chooses the same functions used in tsNET to keep the neighbor-preserving nature of t-SNE. DRGraph distinguishes itself from others in that it utilizes a sparse distance matrix, the negative sampling technique, and a multi-level approach to accelerate the computation. In addition, DRGraph is equivalent to a force-directed based approach, if the gradient is split into one attractive force and M repulsive forces. Therefore, it is feasible to employ conventional methods, such as simulated annealing [15] to accelerate the convergence. The simulated annealing technique randomly replaces system state (graph layout) into a new one with a high system energy to escape from local minima. Meanwhile, constraints for dynamic graph layout (e.g., temporal coherence constraint) can be considered cost terms in the objective functions. Thus, the unified formulation provides the opportunity of applying various functions or graph-related constraints into the formulation.

Limitations. The sparse distance matrix and the negative sampling technique emphasize on preserving the local neighborhood structure while neglecting the global data structure. Therefore, we employ a multi-level layout scheme to maintain the global structure. There is no guarantee that all nodes are coarsened precisely because we generate coarse graphs randomly. There are some edges with very large length (see the web-NotreDame dataset). We aim to improve this in the future.

When the graph data size increases, it is not easy to achieve a good global layout. Because moving nodes when there are full of nodes in the layout space is difficult. tsNET* adopts the result of PMDS as the initial position of nodes. DRGraph employs a multi-level scheme to find a good initialization with coarse graphs.

The data size influences the scalability of generating the node-link diagram. When visualizing a graph data with millions of edges, the efficiency of visual exploration suffers from the limited screen space and possible visual clutter [63]. We leverage the graph sampling technique [45] that randomly selects a subset of the edge set to capture the overall structure. Other possible solutions are density-based visualization through splatting technique [58] and edge bundling [2, 12].

6 Conclusions

In this paper, we present an efficient graph layout algorithm by enhancing the nonlinear dimensionality reduction method with several new techniques. Our new method is feasible within $O(|V|+|E|+TM|V|)$ computational complexity and requires only $O(|V|+|E|)$ memory complexity. Experimental results demonstrate that DRGraph achieves a significant acceleration and generates graph layouts of comparable quality to tsNET. There are many future research directions. We plan to implement DRGraph in GPU by exploiting its parallelism and extend DRGraph for weighted graphs and dynamic graphs. We expect to integrate DRGraph into a visual analysis system for large-scale graphs.

Acknowledgments

This work is supported by National Natural Science Foundation of China (61772456, 61761136020).
| Dataset       | FR   | KK   | S.M. | FM$^3$ | SFDP | PMDS | tsNET | DRGraph |
|--------------|------|------|------|--------|------|------|-------|---------|
| dwt_72       | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] |
| can_96       | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] |
| grid17       | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] |
| price_1000   | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] |
| dwt_1005     | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] |
| block_2000   | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] |
| sierpinski3d | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] |
| EVA          | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] |
| 3elt         | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] |
| G65          | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] |
| troll        | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] |
| Flan_1565    | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] |
| web-NotreDame| ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] |
| com-Orkut    | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] |
| com-LiveJournal | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] | ![Image] |

Figure 8. Visualizations of selected graph datasets using FR, KK, S.M., FM$^3$, SFDP, PMDS, tsNET and DRGraph.
REFERENCES

[1] A. Arleo, W. Didimo, G. Liotta, and F. Montecchiani. A Distributed Multilevel Force-Directed Algorithm. IEEE Transactions on Parallel and Distributed Systems, 30(4):754–765, 2018.

[2] B. Buch, N. H. Riche, C. Hutter, K. Marriott, and T. Dwyer. Towards Unambiguous Edge Bundling: Investigating Confluent Drawings for Network Visualization. IEEE Transactions on Visualization and Computer Graphics, 23(1):541–550, Jan 2017.

[3] G. D. Battista, P. Eades, R. Tamassia, and I. G. Tollis. Graph Drawing: Algorithms for the Visualization of Graphs. Prentice Hall PTR, 1998.

[4] U. Brandes and M. Mader. A Quantitative Comparison of Stress-Minimization Approaches for Offline Dynamic Graph Drawing. In International Symposium on Graph Drawing, pp. 99–110. Springer, 2011.

[5] U. Brandes and C. Pich. Eigensolver Methods for Progressive Multi-dimensional Scaling of Large Data. In International Symposium on Graph Drawing, pp. 42–53. Springer, 2006.

[6] D. M. Chan, R. Rao, F. Huang, and J. F. Canny. t-SNE-CUDA: GPU-Accelerated t-SNE and its Applications to Modern Data. In International Symposium on Computer Architecture and High Performance Computing, pp. 330–338. IEEE, 2018.

[7] D. M. Chan, R. Rao, F. Huang, and J. F. Canny. GPU accelerated t-SNE and its applications to modern data. In Proceedings of the 24th ACM International Conference on Multimedia, pp. 325–328. ACM, 2016.

[8] T. A. Davis and Y. Hu. The University of Florida Sparse Matrix Collection. ACM Transactions on Mathematical Software, 38(1):1–25, Dec. 2011.

[9] T. Dwyer, Y. Koren, and K. Marriott. Stress Majorization with Orthogonal Ordering Constraints. In International Symposium on Graph Drawing, pp. 141–152. Springer, 2006.

[10] T. Dwyer, N. H. Riche, K. Marriott, and C. Mears. Edge Compression Techniques for Visualization of Dense Directed Graphs. IEEE Transactions on Visualization and Computer Graphics, 19(12):2596–2605, 2013.

[11] P. Eades. A Heuristic for Graph Drawing. Congressus Numerantium, 42:149–160, 1984.

[12] R. A. Fisher. The use of multiple measurements in taxonomic problems. Annals of Human Genetics, 7(2):179–188, 1936.

[13] A. Frick, A. Ludwig, and H. Mehlhau. A Fast Adaptive Layout Algorithm for Undirected Graphs. In Proceedings of the DIMACS International Workshop on Graph Drawing, pp. 388–403. Springer, 1994.

[14] T. M. Fruchterman and E. M. Reingold. Graph Drawing by Force-directed Placement. Software: Practice and Experience, 21(11):1129–1164, 1991.

[15] P. Gajer and S. G. Kobourov. GRIP: Graph Drawing with Intelligent Placement. In International Symposium on Graph Drawing, pp. 222–228. Springer, 2000.

[16] E. R. Gansner, Y. Hu, and S. North. A Maxent-Stress Model for Graph Layout. IEEE Transactions on Visualization and Computer Graphics, 19(6):927–940, 2012.

[17] E. R. Gansner, Y. Hu, S. North, and C. Scheidegger. Multilevel Agglomerative Edge Bundling for Visualizing Large Graphs. In IEEE Pacific Visualization Symposium, pp. 187–194, March 2011.

[18] E. R. Gansner, Y. Koren, and S. North. Graph Drawing by Stress Majorization. In Proceedings of the 12th International Conference on Graph Drawing, pp. 239–250. Springer, 2004.

[19] H. Gibson, J. Faith, and P. Vickers. A Survey of Two-Dimensional Graph Layout Techniques for Information Visualisation. Information Visualization, 12(3–4):324–357, 2013.

[20] S. Hackl and M. Jünger. Drawing Large Graphs with a Potential-Field-Based Multilevel Algorithm. In Proceedings of the 12th International Conference on Graph Drawing, pp. 285–295. Springer, 2004.

[21] D. Harel and Y. Koren. Graph Drawing by High-Dimensional Embedding. In The 10th International Symposium on Graph Drawing, pp. 207–219. Springer, 2002.

[22] G. E. Hinton and S. T. Roweis. Stochastic Neighbor Embedding. In Advances in Neural Information Processing Systems, pp. 857–864, 2003.

[23] J. Hoffswell, A. Borning, and J. Heer. Setoca: high-level constraints for graph layout. In Computer Graphics Forum, pp. 537–548, 2018.

[24] Y. Hu. Efficient, High-Quality Force-Directed Graph Drawing. Mathematica Journal, 10(1):37–71, 2005.

[25] J. D. Hunter. Matplotlib: A 2D graphics environment. Computing in Science & Engineering, 9(3):90–95, 2007.

[26] J. Macomy, T. Venturini, S. Heymann, and M. Bastian. ForceAtlas2, A Continuous Graph Layout Algorithm for Handy Network Visualization. PloS one, 9(6):e98679, 2014.

[27] P. Joia, D. Coimbra, J. A. Cuminato, F. V. Paulovich, and L. G. Nonato. Local affine multidimensional projection. IEEE Transactions on Visualization and Computer Graphics, 17(12):2563–2571, 2011.

[28] I. T. Jolliffe. Principal Component Analysis and Factor Analysis, pp. 115–128, 1966.

[29] T. Kamada, S. Kawai, et al. An Algorithm for Drawing General Undirected Graphs. Information Processing Letters, 31(1):7–15, 1989.

[30] S. Kiefert, T. Dwyer, K. Marriott, and M. Wybrow. HOL: Human-like Orthogonal Network Layout. IEEE Transactions on Visualization and Computer Graphics, 22(1):349–358, 2016.

[31] Y. Koren. Graph Drawing by Subspace Optimization. In Proceedings of the Sixth Joint Eurographics-IEEE TCVG conference on Visualization, pp. 65–74. Eurographics Association, 2004.

[32] J. Krueger, P. Rauber, R. M. Martins, A. Kerren, S. Kobourov, and A. C. Telea. Graph Layouts by t-SNE. Computer Graphics Forum, 36(3):283–294, 2017.

[33] J. B. Kuuskul. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. Psychometrika, 29(1):1–27, 1964.

[34] O.-H. Kwon, T. Cnrsrnin, and K.-L. Ma. What Would a Graph Look Like in This Layout? A Machine Learning Approach to Large Graph Visualization. IEEE Transactions on Visualization and Computer Graphics, 24(1):478–488, 2018.

[35] O.-H. Kwon and K.-L. Ma. A Deep Generative Model for Graph Layout. IEEE Transactions on Visualization and Computer Graphics, 26(6):665–675, 2019.

[36] S. Liu, W. Cui, Y. Wu, and M. Liu. A survey on information visualization: recent advances and challenges. The Visual Computer, 30(12):1373–1393, 2014.

[37] Y. Lu, Z. Yang, and J. Corander. Doubly Stochastic Neighbor Embedding on Spheres. arXiv preprint arXiv:1609.01977, 2016.

[38] L. v. d. Maaten and G. Hinton. Visualizing Data using t-SNE. Journal of Machine Learning Research, 9(Nov):2579–2605, 2008.

[39] C. McGrath, J. Blythe, and D. Krackhardt. Seeing Groups in Graph Representations of Words and Phrases and their Compositionality. In Advances in Neural Information Processing Systems, pp. 3111–3119, 2017.

[40] A. Nguyen and S.-H. Hong. K-core based Multi-level Graph Visualization for Scale-free Networks. In IEEE Pacific Visualization Symposium, pp. 21–25, 2017.

[41] Q. H. Nguyen, S. H. Hong, P. Eades, and A. Meidiana. Proxy Graph: Visual Quality Metrics of Big Graph Sampling. IEEE Transactions on Visualization and Computer Graphics, 23(6):1600–1611, 2017.

[42] P. Pezzotti, T. Höltt, B. Lelieveldt, E. Eisemann, and A. Vilanova. Hierarchical stochastic neighbor embedding. Computer Graphics Forum, 35(3):21–30, 2016.

[43] P. Pezzotti, B. P. F. Lelieveldt, L. v. d. Maaten, T. Höltt, E. Eisemann, and A. Vilanova. Approximated and User Steerable tSNE for Progressive Visual Analytics. IEEE Transactions on Visualization and Computer Graphics, 23(7):1739–1752, July 2017.

[44] H. C. Purchase. Metrics for graph drawing aesthetics. Journal of Visual Languages & Computing, 13(5):501–516, 2002.

[45] J. Ren, J. Schneider, M. Osvjanikov, and P. Wonka. Joint Graph Layouts for Visualizing Collections of Segmented Meshes. IEEE Transactions on Visualization and Computer Graphics, 24(9):2546–2558, 2018.

[46] U. Rüegg, S. Kiefert, T. Dwyer, K. Marriott, and M. Wybrow. Stress-Minimizing Orthogonal Layout of Data Flow Diagrams with Ports. In International Symposium on Graph Drawing, pp. 319–330, 2014.

[47] S. J. Russell and P. Norvig. Artificial intelligence: a modern approach.
[53] J. W. Sammon. A Nonlinear Mapping for Data Structure Analysis. *IEEE Transactions on Computers*, C-18(5):401–409, May 1969.

[54] R. Tamassia. *Handbook of Graph Drawing and Visualization*. Chapman & Hall/CRC, 2016.

[55] J. Tang, J. Liu, M. Zhang, and Q. Mei. Visualizing Large-scale and High-dimensional Data. In *Proceedings of the International Conference on World Wide Web*, pp. 287–297, 2016.

[56] J. B. Tenenbaum, V. De Silva, and J. C. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(5500):2319–2323, 2000.

[57] L. Van Der Maaten. Accelerating t-SNE Using Tree-based Algorithms. *Journal of Machine Learning Research*, 15(1):3221–3245, 2014.

[58] R. Van Liere and W. De Leeuw. GraphSplatting: Visualizing graphs as continuous fields. *IEEE Transactions on Visualization and Computer Graphics*, 9(2):206–212, 2003.

[59] X. Wang, X. He, Y. Cao, M. Liu, and T. Chua. KGAT: Knowledge Graph Attention Network for Recommendation. In *Proceedings of the ACM International Conference on Knowledge Discovery & Data Mining*, pp. 950–958, 2019.

[60] Y. Wang, Z. Jin, Q. Wang, W. Cui, T. Ma, and H. Qu. DeepDrawing: A Deep Learning Approach to Graph Drawing. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):676–686, 2019.

[61] Y. Wang, Y. Wang, Y. Sun, L. Zhu, K. Lu, C.-W. Fu, M. Sedlmair, O. Deussen, and B. Chen. Revisiting Stress Majorization as a Unified Framework for Interactive Constrained Graph Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):489–499, 2018.

[62] K. Wongsuphasawat, D. Smilkov, J. Wexler, J. Wilson, D. Man, D. Fritz, D. Krishnan, F. B. Vigas, and M. Wattenberg. Visualizing Dataflow Graphs of Deep Learning Models in TensorFlow. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):1–12, 2018.

[63] Y. Wu, N. Cao, D. Archambault, Q. Shen, H. Qu, and W. Cui. Evaluation of Graph Sampling: A Visualization Perspective. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):401–410, 2017.

[64] Y. Wu, N. Pitipornvivat, J. Zhao, S. Yang, G. Huang, and H. Qu. egoSlider: Visual Analysis of Egocentric Network Evolution. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):260–269, 2016.

[65] Z. Yang, J. Peltonen, and S. Kaski. Optimization Equivalence of Divergences Improves Neighbor Embedding. In *Proceedings of the International Conference on Machine Learning*, pp. 460–468, 2014.

[66] V. Yoghourdjian, T. Dwyer, G. Gange, S. Kieffer, K. Klein, and K. Marriott. High-Quality Ultra-Compact Grid Layout of Grouped Networks. *IEEE Transactions on Visualization and Computer Graphics*, 22:339–348, 2016.

[67] L. Zaorálek, T. Buríánek, and V. Snášel. Dimension reduction methods in Graph Drawing Problem. In *International Conference on Intelligent Systems Design and Applications*, pp. 13–18, 2014.

[68] M. Zinsmaier, U. Brandes, O. Deussen, and H. Strobelt. Interactive Level-of-Detail Rendering of Large Graphs. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2486–2495, 2012.
Tables 1, 2, 3, 4, and 5 show the parameter sensitivity. Experiments on the com-Orkut and com-LiveJournal datasets are not feasible due to the enormous computational cost. Some items are unfilled due to the memory limitation and the computational cost.

The size of $k$-order nearest neighbors. Table 1 shows the neighborhood preservation accuracy (NP), the normalized stress, crosslessness and minimum angle metrics with respect to the size of $k$-order nearest neighbors. We find that NP drops when $k$ becomes large on almost all datasets. The reason is that higher-order nearest neighbors contain dissimilar nodes, which are treated as positive nodes. Thus, DRGraph places dissimilar nodes close to each other, resulting in a low layout quality. When $k$ is small, DRGraph usually achieves higher results of crosslessness and minimum angle. Besides, higher-order nearest neighbors cost large memory consumptions for keeping graph distances. Thus, we choose the first-order nearest neighbors, which reduces the computational complexity to $O(|E| + |V| + TM|V|)$ and the memory complexity to $O(|E| + |V|)$. 1-order nearest neighbors is sufficient to provide locality properties, accelerate the computation of the node similarity, and meanwhile reduce the memory requirement.

The number of negative samples $M$. Table 2 reports the layout quality with respect to the number of negative samples $M$. We find that the performance slightly rises when the number of negative samples increases. With a large subset of nodes, we can approximate the gradient accurately. However, the computation complexity is linear with the number of negative samples. To keep the balance between quality and efficiency, we choose the number of negative samples $M=5$.

The weight of negative samples $\gamma$. Table 3 lists the layout quality with a varied $\gamma$. $\gamma$ controls the value of repulsive forces of the gradient. A small value of $\gamma$ generates small repulsive forces, while nodes in the graph data are too close to each other and form local clusters. We can see that the neighborhood preservation accuracy drops when $\gamma$ is very small. A large value of $\gamma$ forces dissimilar nodes far away. The edges are longer than the ideal length in the layout space, leading to bad stress quality. The crosslessness and minimum angle metrics are stable for different choices of $\gamma$. We choose a medium value of 0.1, which has a high neighborhood preservation accuracy and small stress. DRGraph employs the early exaggeration technique to find a better initialization. $\gamma$ is set to be 0.01 for the coarsest graph and $\gamma = 0.1$ for the rest.

The iteration number $T$. Table 4 presents the layout quality with respect to the iteration number $T$. The accuracy of the neighborhood preservation becomes stable when the iteration number $T$ is large adequately. Besides, the iteration number has little effect to the crosslessness and minimum angle metrics. In practice, we choose the iteration number $T = 400$ to accelerate the convergence and generate aesthetically pleasing results.

The effect of $b$. Figure 1 (a) illustrates the sum of the attractive force and the repulsive force (i.e., the gradient) with respect to $b$. $b$ controls the value of the sum force without altering the ideal distance between nodes. In general, a small $b$ value (e.g., $b = 1$) tends to put similar nodes close to others (see Figure 1 (b)). As shown in Table 5, a small value of $b$ would increase neighborhood preservation accuracy. However, when $b$ is small, the layout distance varies greatly, resulting in a bad stress quality. A very large $b$ value (e.g., $b = 3$) forces all edge lengths to be ideal but obfuscates the global structure (see Figure 1 (f)). DRGraph achieves good stress and minimum angle quality but a low neighborhood preservation accuracy (see Table 5) when $b$ is large. In our implementation, we choose $b = 3$ when the input graph is a grid graph (e.g., grid17), in which all edges have the same length. For 3D meshes (e.g., G65) (e.g., G65) and large graphs (e.g., web-NotreDame), it is intractable to preserve all edge lengths of a manifold in the two-dimensional layout space. Therefore, we set $b = 1$ to preserve the neighborhood identity. For other graphs, we choose $b = 2$, which works well in preserving both local and global structures.

![Force](https://via.placeholder.com/150.png)

**Figure 1.** The effect of the $b$ parameter, which forces the edge to the ideal length.

- M. Zhu, W. Chen, Y. Hu, Y. Hou, L. Liu and K. Zhang are with State Key Lab of CAD&CG, Zhejiang University. E-mail: {minfeng.zhu, cadhyz, houyuxuan, liuliangjun}@zju.edu.cn, chenwei@cad.zju.edu.cn, zhangkaiyuan20@gmail.com.
- Wei Chen is the corresponding author.

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx
Table 1. The layout quality with respect to the \( k \)-order nearest neighbors.

| \( k \) | 1   | 2   | 3   | 4   | 5   |
|-------|-----|-----|-----|-----|-----|
| NP   |     |     |     |     |     |
| stress |     |     |     |     |     |
| crosslessness |     |     |     |     |     |
| min_angle |     |     |     |     |     |
| dwt_72  | 8811 | .9151 | .8751 | .8663 | .8271 |
| lemsis  | 5963 | .6852 | .5633 | .5111 | .5074 |
| can_96  | 6490 | .6197 | .5826 | .5607 | .5511 |
| raja11  | 7010 | .6538 | .5933 | .5665 | .5515 |
| jazz    | 7738 | .7538 | .6663 | .6394 | .6707 |
| visbrazil | 4802 | .4561 | .5091 | .5157 | .2946 |
| grid17  | 8505 | .7672 | .7331 | .7110 | .6985 |
| mesh3el | 9941 | .9035 | .8184 | .7839 | .7606 |
| ntsclimate | 6665 | .6453 | .5211 | .4423 | .4502 |
| dwt_419 | 7550 | .7256 | .6885 | .6665 | .6544 |
| price_1000 | 5567 | .4214 | .2796 | .1842 | .1389 |
| dwt_1005 | 5949 | .5629 | .5203 | .4919 | .4729 |
| cage8   | 2963 | .2505 | .2106 | .1800 | .1629 |
| bcsstk09 | 8865 | .8251 | .7580 | .7340 | .7171 |
| block2000 | 3024 | .2755 | .2463 | .2000 | .1621 |
| sierpinsk3d  | 5768 | .5996 | .5137 | .5098 | .4685 |
| CA-GrQc | 1735 | .1543 | .1039 | .0712 | .0456 |
| EVA     | 7145 | .5866 | .4665 | .4216 | .3663 |
| 3elt    | 6441 | .5872 | .5177 | .5966 | .5504 |
| us_powergrid | 4616 | .4245 | .3622 | .3248 | .3008 |
| G65     | 2692 | .2518 | .2351 | .2156 | .1926 |
| fe_delt2 | 5702 | .5350 | .5199 | .4764 | .4608 |
| bcsstk1 | 3660 | .3953 | .3557 | .3246 | .3143 |
| venkat50 | 6506 | .6906 | .6679 | .5448 | .5120 |
| ship_003 | 2002 | .2003 | .1894 | .1789 | .1638 |
| troll   | 2539 | .2579 | .2408 | .2242 | .2175 |
| NotreDame | 6471 | .4192 | .2181 |     |     |
| Flan_1565 | 2067 | .2004 | .1898 |     |     |

Table 2. The layout quality with respect to the size of negative samples.

| M     | 1   | 2   | 3   | 4   | 5   |
|-------|-----|-----|-----|-----|-----|
| NP    |     |     |     |     |     |
| stress |     |     |     |     |     |
| crosslessness |     |     |     |     |     |
| min_angle |     |     |     |     |     |
| dwt_72  | 8889 | .8625 | .8473 | .8449 | .8357 |
| lemsis  | 5692 | .6481 | .6426 | .6349 | .6427 |
| can_96  | 5964 | .6321 | .6354 | .6280 | .6446 |
| raja11  | 7014 | .6945 | .6948 | .6965 | .6864 |
| jazz    | 7780 | .7759 | .7722 | .7718 | .7637 |
| visbrazil | 4468 | .4710 | .5137 | .5057 | .5413 |
| grid17  | 7408 | .8031 | .8424 | .8949 | .8979 |
| mesh3el | 8991 | .9023 | .9947 | .9962 | .9966 |
| ntsclimate | 6277 | .6575 | .6467 | .6466 | .6603 |
| dwt_419 | 7464 | .7295 | .7543 | .7555 | .7550 |
| price_1000 | 4764 | .5343 | .5498 | .5549 | .5574 |
| dwt_1005 | 5425 | .5887 | .6094 | .6008 | .6149 |
| cage8   | 2303 | .2756 | .2916 | .3174 | .3343 |
| bcsstk09 | 8361 | .8752 | .8866 | .8922 | .8909 |
| block2000 | 2767 | .2931 | .3028 | .3075 | .3151 |
| sierpinsk3d  | 5008 | .5853 | .6576 | .5475 | .5793 |
| CA-GrQc | 1306 | .1951 | .1730 | .1854 | .2003 |
| EVA     | 6569 | .7010 | .7188 | .7250 | .7365 |
| 3elt    | 5776 | .5924 | .6348 | .6430 | .6596 |
| us_powergrid | 4152 | .4488 | .4642 | .4732 | .4833 |
| G65     | 2422 | .2587 | .2615 | .2625 | .2659 |
| fe_delt2 | 4379 | .5771 | .5867 | .5969 | .5952 |
| bcsstk31 | 3580 | .3780 | .3920 | .3846 | .3911 |
| venkat50 | 5364 | .6040 | .6382 | .6473 | .6408 |
| ship_003 | 1818 | .1944 | .2000 | .2031 | .2084 |
| troll   | 2419 | .2488 | .2523 | .2585 | .2624 |
| toros3  | 1135 | .1416 | .1481 | .1413 | .1477 |
| NotreDame | 4536 | .4569 | .4650 | .4779 | .4786 |
| Flan_1565 | 1812 | .1999 | .2040 | .2102 | .2107 |
Table 3. The layout quality with respect to $\gamma$.

| $\gamma$ | 0.01 | 0.05 | 0.1  | 0.2  | 1.0  |
|---------|------|------|-----|-----|-----|
| NP      | stress | crosslness | min_angle |
| dwt_72  | 0.7951 | 0.8217 | 0.8343 | 0.8919 | 0.9158 |
| lesmis  | 0.6689 | 0.6534 | 0.6552 | 0.6411 | 0.6361 |
| can_06  | 0.6151 | 0.6172 | 0.6348 | 0.6378 | 0.6515 |
| 0.8917 | 0.8217 | 0.8343 | 0.8919 | 0.9158 |

Table 4. The layout quality with respect to the iteration number T.

| T       | 100 | 200 | 300 | 400 | 500 |
|---------|-----|-----|-----|-----|-----|
| NP      | stress | crosslness | min_angle |
| dwt_72  | 0.6705 | 0.7446 | 0.8388 | 0.8808 | 0.9098 |
| lesmis  | 0.6156 | 0.6355 | 0.6513 | 0.6538 | 0.6468 |
| can_06  | 0.5988 | 0.6479 | 0.6577 | 0.6551 | 0.6509 |
| 0.8917 | 0.8217 | 0.8343 | 0.8919 | 0.9158 |

- Table 3. The layout quality with respect to $\gamma$.
- Table 4. The layout quality with respect to the iteration number T.
| b    | NP | stress | crosslessness | min_angle |
|------|----|--------|---------------|-----------|
|倦    | 7668 | .7825  | .8773 | .8385 | .7613 | 1.000  |
| lesmis | .6512 | .6626  | .6564 | .6482 | .6245 | 1.000  |
| can_96 | .6055 | .6312  | .6496 | .6421 | .6289 | .72  |
| rajat11 | .6870 | .7012  | .7026 | .6735 | .6633 | .107  |
| jazz | .7843 | .7755  | .7744 | .7658 | .7564 | .162  |
| visbrazil | .5180 | .4935  | .4806 | .4711 | .4599 | .107  |
| grid17 | .7049 | .7947  | .8472 | .8544 | .8540 | .107  |
| mesh3el | .8012 | .9387  | .9905 | .9947 | .9966 | .107  |
| netscience | .6892 | .6859  | .6603 | .6345 | .6161 | .107  |
| dwt_419 | .7033 | .7263  | .7501 | .7523 | .7195 | .107  |
| price_1000 | .6036 | .5893  | .5484 | .5080 | .4674 | .107  |
| dwt_1005 | .5080 | .5750  | .5823 | .5809 | .5939 | .107  |
| cage8 | .3564 | .3252  | .2967 | .2824 | .2686 | .107  |
| bcstk09 | .7833 | .8567  | .8833 | .8880 | .8857 | .107  |
| block_2000 | .3207 | .3098  | .3020 | .2975 | .2907 | .107  |
| sierpinski3d | .5257 | .5684  | .5746 | .5438 | .5354 | .107  |
| CA-GrQc | .3081 | .2174  | .1736 | .1537 | .1407 | .107  |
| EVA | .7957 | .7646  | .7118 | .6648 | .6231 | .107  |
| 3elt | .6432 | .6530  | .5504 | .5116 | .4885 | .107  |
| us_powergrid | .4629 | .4227  | .3502 | .2827 | .2491 | .107  |
| G65 | .2511 | .2694  | .2615 | .2451 | .2188 | .107  |
| fe_4elt2 | .5836 | .5784  | .4979 | .4260 | .3756 | .107  |
| bcstk31 | .3893 | .3944  | .3788 | .3418 | .3416 | .107  |
| venkat50 | .6431 | .6084  | .5222 | .4309 | .3802 | .107  |
| ship_003 | .1976 | .1979  | .1947 | .1877 | .1743 | .107  |
| troll | .2548 | .2505  | .2365 | .2181 | .2022 | .107  |
| torso3 | .1450 | .1376  | .1207 | .1074 | .1074 | .107  |
| web-NotreDame | .4752 | .3458  | .2835 | .2533 | .2358 | .107  |
| Flan_1565 | .2077 | .1968  | .1575 | .1258 | .0976 | .107  |
| Graph Dataset      | FR | KK | S.M. | FM$^3$ | SFDP | PMDS | tsNET | DRGraph |
|-------------------|----|----|------|--------|------|------|-------|---------|
| lesmis            | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) |
| rajat11           | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) |
| jazz              | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) |
| visbrazil         | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) |
| mesh3e1           | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) |
| netscience        | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) |
| dwt_419           | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) |
| cage8             | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) |
| bcsstk09          | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) |
| CA-GrQc           | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) |
| us_power grid     | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) |
| fe_4elt2          | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) |
| bcsstk31          | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) |
| venkat50          | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) |
| ship_003          | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) | ![Visualization](image) |

Figure 2. Visualizations of selected graph datasets using FR, KK, Stress Majorization (S.M.), FM$^3$, SFDP, PMDS, tsNET and DRGraph.