Balancing Between the Local and Global Structures (LGS) in Graph Embedding

Jacob Miller(B), Vahan Huroyan, and Stephen Kobourov

University of Arizona, Tucson, USA
jacobmiller1@arizona.edu, vahanhuroyan@math.arizona.edu,
kobourov@cs.arizona.edu

Abstract. We present a method for balancing between the Local and Global Structures (LGS) in graph embedding, via a tunable parameter. Some embedding methods aim to capture global structures, while others attempt to preserve local neighborhoods. Few methods attempt to do both, and it is not always possible to capture well both local and global information in two dimensions, which is where most graph drawing live. The choice of using a local or a global embedding for visualization depends not only on the task but also on the structure of the underly-ing data, which may not be known in advance. For a given graph, LGS aims to find a good balance between the local and global structure to preserve. We evaluate the performance of LGS with synthetic and real-world datasets and our results indicate that it is competitive with the state-of-the-art methods, using established quality metrics such as stress and neighborhood preservation. We introduce a novel quality metric, cluster distance preservation, to assess intermediate structure capture. All source-code, datasets, experiments and analysis are available online.

Keywords: Graph embedding · Graph Visualization · Local and global structures · Dimensionality Reduction · Multi-dimensional Scaling

1 Introduction

Graphs and networks are a powerful tool to encode relationships between objects. Graph embeddings, which map the vertices of a graph to a set of low dimensional vectors (real valued coordinates), are often used in the context of data visualization to produce node-link diagrams. While many layout methods exist [27], dimension reduction (DR) techniques have had success in providing desirable layouts, by capturing graph structure in reasonable computation times. DR methods are used to project high-dimensional data into low-dimensional space and some of these methods only rely on the relationships between the datapoints, rather than datapoint coordinates in higher dimension. These techniques are applicable for both graph embeddings and visualization. Further, local DR algorithms attempt to preserve the local neighborhoods, while global DR algorithms attempt to retain all pairwise distances.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
M. A. Bekos and M. Chimani (Eds.): GD 2023, LNCS 14465, pp. 263–279, 2023.
https://doi.org/10.1007/978-3-031-49272-3_18

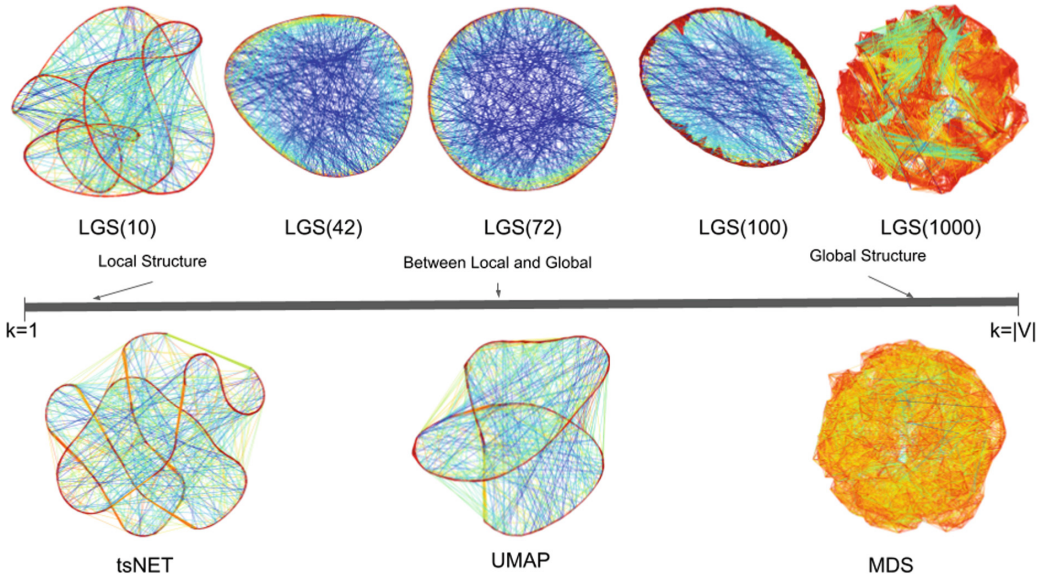


Fig. 1. Embeddings of the connected_watts_1000 graph; see Sect. 4. The top row shows LGS embeddings – from local to global – with varying neighborhood sizes (k). The LGS(72) layout captures the correct underlying model. The bottom row shows tsNET [15], UMAP [18], and MDS [30] embedding of the same graph.

Two popular techniques that are adapted in graph visualization are (metric) Multi-Dimensional Scaling (MDS) [5,16] and t-distributed stochastic neighbor embedding (t-SNE) [17]. The goals of these two algorithms are somewhat orthogonal: MDS focuses on preserving all pairwise distances, while t-SNE aims to preserve the likelihood of points being close in the embedding if they were close in the original space. MDS is said to preserve global structure, while t-SNE is said to preserve local neighborhoods [7]. These ideas are directly applicable to graph visualization, where we can define the distances as the graph theoretic distances, e.g., via all-pairs shortest paths (APSP) computation. In the graph layout literature, MDS is often referred to as stress minimization [11,30], and t-SNE has been adapted to graph layout in an algorithm known as tsNET [15] and later DRGraph [31]. Choosing the "best" graph embedding algorithm depends on the graph structure and the task. MDS is effective for structured/mesh-like graphs, while t-SNE works better for clustered/dense graphs. This phenomenon also applies to local and global force-directed layouts as well [15].

Automating the selection of the "best" embedding algorithm is challenging due to its dependency on graph structure. We introduce the Local-to-Global Structures (LGS) algorithm which provides a parameter-tuneable framework that can produces embeddings that span the spectrum from local optimization to global optimization.

Smaller values of the LGS parameter prioritize local structure, while larger values emphasize global structure. LGS enables exploration of the trade-off, revealing meaningful middle ground solutions. We introduce a new metric called

cluster distance to measure how well this intermediate structure is preserved. Everything described in this paper is available on Github: https://github.com/Mickey253/L2G. We provide a video and additional layouts and analysis in [19].

2 Background

Dimensionality Reduction (DR) refers to a large family of algorithms that map a set of high-dimensional datapoints in lower-dimensionsal space. Different DR algorithms aim to preserve various properties of the dataset, such as total variance, global distances, local distances, etc. In visualization contexts, the dataset is typically projected onto 2D or 3D Euclidean space. DR algorithms generally accept input of two types: sample or distance. Sample-based algorithms, such as Principal Component Analysis (PCA) [8,13] project the high dimensional data down to the embedding space. For distance-based inputs, the algorithms directly work with distance metrics. In the case of graph embeddings, the graph-theoretic distance is used, often all-pairs shortest path (APSP).

Popular techniques in the local category include t-SNE [17], UMAP [18], LLE [25], IsoMap [28], etc. For global structure, methods such as PCA [8] and MDS [5,16] are used. MDS has variants, but here we mean metric MDS which minimizes stress [26]. Few techniques attempt to capture both global and local structure. Chen and Buja [3] adapt MDS to capture local structure by selectively preserving distances between a subset of pairs using kNN. The underlying idea is similar to ours , but it does not provide a framework to cover the spectrum from local to global as our method does. While t-SNE's perplexity parameter aims to imitate the size of neighborhood to be preserved, in general increasing its value does not lead to a global structure preservation [29]. Anchor-t-SNE improves the global structure preservation by anchoring a set of points to use as a skeleton for the rest of the embedding [9], however, it does not provide a framework to cover the spectrum from local to global. UMAP [12,18] also aims to preserve the local structures of a dataset. While UMAP claims to preserve the global structures better than t-SNE, we show that this is not universally true for graph data in Sect. 5.

Graph Embedding is a problem to assign vectors to graph vertices, capturing the graph structure. More formally, given a graph $G = (V, E)$, find a d-dimensional vector representation of V that optimally preserves properties [2] (e.g., pairwise distances in MDS [5,16]). We restrict ourselves to 2D node-link visualization with edges represented by straight-line segments, so the problem is reduced to finding a 2D embedding for the vertices. Aesthetic criteria are often used to evaluate the quality of a graph embedding: the number of edge crossings, average edge length, overall symmetry, etc. [24]. Aesthetic criteria enhance readability and task facilitation, but information faithfulness is equally important. It ensures that the embedding accurately represents all underlying data, regardless of the task [20,21] and graph embeddings provide a nice benefit by directly optimizing graph structure preservation. Graph structure is a nebulous term; referring to inherent properties of the underlying graph such as local/global

distances. Global distance preservation methods capture the graph's topological structure by closely aligning embedded distances with graph-theoretic distances. This approach is ideal for connectivity-based tasks and offers insights into the global scale and shape of the data. Local structure preservation methods preserve the immediate neighborhood of each vertex, effectively capturing clusters or densely connected subgraphs. While nearby vertices in the embedding can be considered similar, distant vertices may have irrelevant distances. This can be observed in the presence of long edges in the local embedding column in Fig. 2.

Graph Embedding by Dimensionality Reduction: In a good embedding, the drawn distance should closely match the graph-theoretic distance between vertices [14]. This observation led to the use of stress function, which MDS aims to optimize, to obtain a graph embedding [11]. Stress can be minimized by majorization [11], stochastic gradient descent (SGD) [30], etc. The MDS approach suffers from an APSP computation, which usually relies on Floyd-Warshall's $O(|V|^3)$, or on Johnson's $O(|V|^2 \log |V| + |E||V|)$ algorithms. The maximum entropy model (MaxEnt) [10] adds a negative entropy between vertices in the graph. The motivation for MaxEnt is to improve the asymptotic complexity. The MaxEnt model places neighbor nodes closer while maximizing the distance between all vertices. This is conceptually similar to the LMDS of Chen and Buja [3]. Our approach differs from the MaxEnt model in motivation: Our LGS captures local structure, global structure, or balances between the two, whereas MaxEnt is primarily concerned with speed. We cannot avoid an APSP computation, and make use of SGD to optimize our objective function in lieu of majorization.

Optimizing stress creates effective layouts, but may neglect local structures; see Fig. 2. tsNET [15] captures local structure by also adding a repulsive force between vertices to achieve cluster separation. tsNET has been sped up by making use of negative sampling and sparse approximation to avoid the APSP computation [31]. Nocaj et al. [22] achieve effects similar to tsNET by weighting edges based on "edge embeddedness" and perform MDS on the weighted graph.

3 The Local-to-Global Structures (LGS) Algorithm

Local methods (e.g., t-SNE) preserve local neighborhoods, while global methods (e.g., MDS) capture all pair-wise distances. We propose the Local-to-Global Structures (LGS) algorithm that achieves the following 3 goals:

G1 A single parameter controlling local-global embedding balance
G2 When this parameter is small, the embedding preserves local neighborhoods
G3 When this parameter is large, the embedding preserves the global structure

By "local neighborhood" of a vertex we refer to the immediate neighbors of the vertex being considered. If the nearest neighbors of each vertex in an embedding match well with the nearest neighbors in the actual graph, then the embedding accurately preserves the local structures. By "global structure" we refer to the

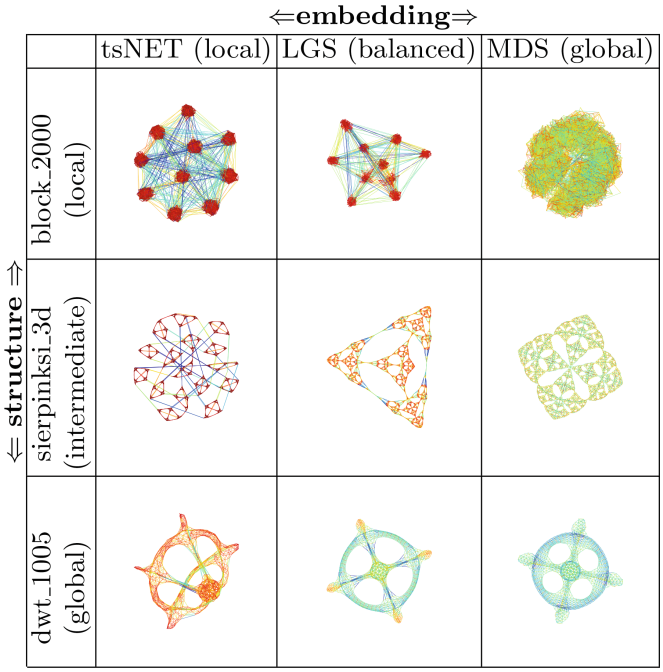


Fig. 2. Local embedding methods perform well on graphs with distinct local structure (block_2000), but they can distort the global shape of the graph (dwt_1005). Global methods capture the overall shape (e.g., dwt_1005), but may miss important local structures (block_2000). LGS(100) performs well for graphs with both local and global structure, such as sierpinski_3d, allowing us to see its fractal nature.

preservation of all pairwise graph distances (including long ones) in the embedding. Finally, "intermediate structure" refers to capturing both local neighbors and global structure. Figure 2 shows graphs exemplifying local, intermediate, and global structures and Sect. 3.2 defines formal embedding measures: neighborhood error, cluster distance, and stress. In Sect. 3.1 we explain the selection process for the balance parameter k and the objective function to ensure that the solution aligns with the stated goals. For G1, we modify MDS to preserve distances in a neighborhood defined by a parameter k). Thus, preserving distances for large neighborhoods satisfies G3. This leaves a question for G2: Does applying distance preservation to a subset of pairs result in locally faithful embeddings?

3.1 Adapting Stress Minimization for Local Preservation

We define a parameter, k, that represents the size of a neighborhood surrounding each vertex. A straightforward approach would involve simply selecting the k-nearest vertices for every given vertex (as in [3]). However, the graph-theoretic distance in an undirected graph is a discrete measure, which can create complications. For example, consider the local structure graph (top row) in Fig. 2. Although the within-cluster density is high, there are many edges between different clusters. Unfortunately, there is no simple way to test if an edge is within clus-

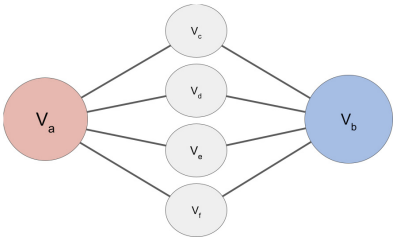


Fig. 3. An example of how we may skip over immediate neighbors when selecting neighborhoods to preserve. In this case, $c = 2$. There is only one unique walk of length ≤ 2 from v_a to v_c, v_d, v_e, v_f, but there are 4 such walks from v_a to v_b. In this case, v_b would be the first vertex added to v_a's most connected neighborhood.

ter or out-of-cluster. In order to produce tsNET-like embeddings, which should pay more attention to local structures, we must avoid preserving out-of-cluster edges.

Instead of considering distances directly, we find the top k most connected vertices for each vertex based on the hypothesis that more possible walks between vertices indicate greater similarity; see Fig. 3. Despite v_A and v_B not sharing an edge, they have the same set of neighbors. When v_A and v_B are both neighbors to a set of vertices, we can confidently state their similarity, confirmed by their shared proximity to vc, v_d, etc. [6]. The c-th power of an adjacency matrix $(A^c)_{i,j_G}$ encodes the number of c-length walks from vertex i to vertex j. To find the top k "most connected" vertices for each vertex, follow this procedure: Given an adjacency matrix of an undirected graph, A_G, raise it to the c-th power, take the sum of all powers $A^{\square} = \sum_{1 \leq j \leq c} A^i_G$, to obtain a matrix whose (i, j)-th element shows the number of walks from i to j of length less or equal than c. Since each row in $A^{\square}$ corresponds to a vertex, we find the k largest values in row i (by sorting). We define these top k vertices to be the "most connected" neighborhood $N_k(v_a)$ of vertex v_a; see Fig. 3. We further weight the power of the matrix with a decaying weight factor, s, $0 < s < 1$, such that $A^{\square} = \sum_{1 \leq i \leq c} s^i A^i_G$. We investigate a range of values for s, and set $s = 0.1$; see [19]. We propose a procedure to reduce the number of matrix multiplications which we used in our experiments; see [19].

Objective Function. We remark, that only preserving distances of a subset of pairs will result in poor embeddings: e.g., two vertices that cannot "see" each other can be placed arbitrarily close with no penalty. A second term is needed in the objective function to prevent this, and we add an entropy repulsion term as in [3, 10], to force pairs of vertices away from each other. For a given pairwise distance matrix $[d_{ij}]^n_{i,j=1}$ we define the following generalized stress function as an objective function:

$$\sigma(X) = \sum_{(i,j) \in N_k} (\|X_i - X_j\| - d_{ij})^2 - \alpha \sum_{(i,j) \notin N_k} \log \|X_i - X_j\|, \qquad (1)$$

where X_i is the embedded point in R^d, α is a fixed constant parameter that controls the weight of the logarithmic term, and N_k corresponds to the neighborhood that we aim to preserve in the embedded space. This objective function ensures that distances are preserved between the most-connected neighborhoods, while maximizing entropy. We use the negative logarithm of the distance between points, so that the repulsive force is relatively strong at small distances, but quickly decays (so that distant points are not forced to be too distant from each other). While similar to LMDS [3] and MaxEnt [10], the proposed objective function in Eq. 1 differs in (1) how the set N_k is selected (LMDS uses a kNN search

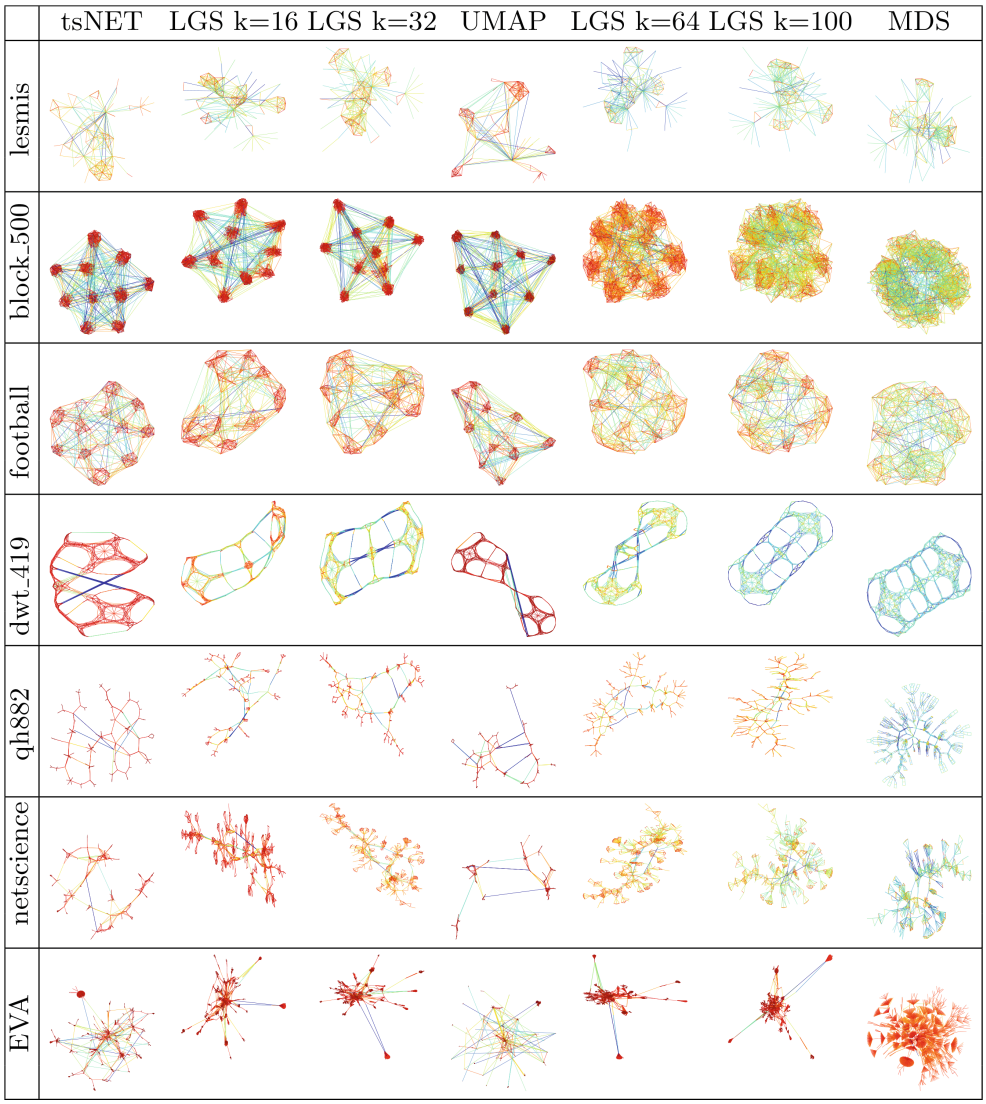


Fig. 4. Example embeddings. The first and last columns show the two extremes tsNET (local) and MDS (global); the middle column shows UMAP. The remaining columns show a gradual increase of LGS's k parameter, moving from local to global distance preservation (left to right). Note LGS outputs are vertically higher in each row.

and MaxEnt preserves distances between two vertices if and only if they share an edge) and (2) LMDS and MaxEnt cannot be easily parameterized to balance local and global structure preservation. We minimize the objective function by SGD which works well for stress minimization [1,30]. The parameter space of the algorithm is discussed in the supplemental material [19].

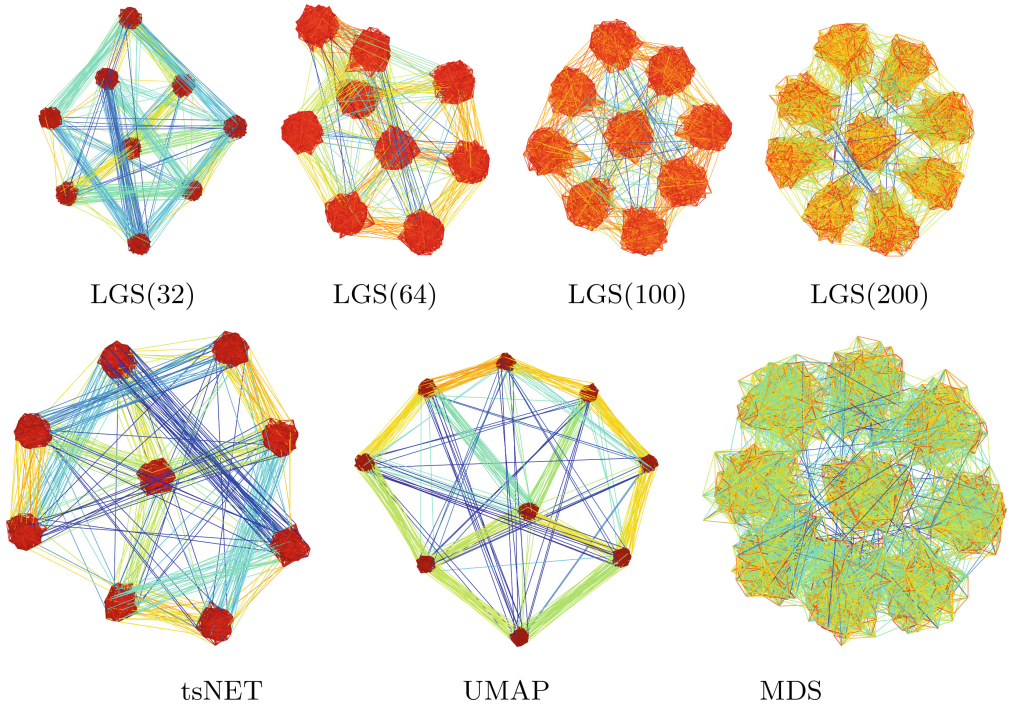


Fig. 5. The grid_cluster graph is generated so that each cluster has many out-of-cluster edges to its neighbors in a 3×3 lattice, providing a recognizable intermediate structure. tsNET and UMAP do not place clusters on a grid, MDS mixes the clusters; LGS(100) captures the 3× 3 grid and shows distinct clusters.

3.2 Evaluation Metrics

We discuss the evaluation metrics for embedding algorithms: local neighborhood error (NE) score, intermediate structure (CD), and global distances (Stress).

NE Metric: Neighborhood hits (NH) measures how well an embedding preserves local structures [3,7]. NH is the average Jaccard similarity of the neighbors in the high-dimensional and low-dimensional embedding. Let Y be an n × d dimensional dataset, X be its n × 2-dimensional embedding, and a radius r defines the size of the neighborhood one intends to measure. NH is defined as:

$$NH(Y, X, r) = \frac{1}{n}\sum_{i=1}^{n} \frac{|N_Y(p_i, r) \cap N_X(p_i, r)|}{|N_Y(p_i, r) \cup N_X(p_i, r)|} \quad (2)$$

where $N_Y(p_i, r)$ denotes the r nearest points to point p_i in Y and $N_X(p_i, r)$ the r nearest points to point p_i in X. For graph embeddings, this notion is called neighborhood preservation (NP) [10,15,31], with the main difference being that the radius r now refers to graph-theoretic distance: all vertices with shortest path distance $\leq r$ from vertex v_i. Specifically, NP measures the average Jaccard similarity of a vertex's graph-theoretic neighborhood of radius r and an equally sized neighborhood of that vertex's closest embedded neighbors. Since NH and NP measure accuracy, it is desirable to maximize these values. To facilitate comparison with the other two metrics (where lower scores mean better embeddings), we use Jaccard dissimilarity instead and refer to it as Neighborhood Error (NE).

Cluster Distance Metric: We introduce a new metric to measure how well intermediate structures are captured in an embedding. Since the distances between clusters in t-SNE cannot be interpreted as actual distances [29], while clusters in MDS embeddings are often poorly separated, we measure how faithful the relative distances between cluster centers are represented in the embedding. When cluster labels are given as part of the input (e.g., labels, classes), we can use them to define distances between the clusters. When cluster information is not given, we use k-means clustering in the high-dimensional data case, and modularity clustering in the graph case. The distances between clusters in the high-dimensional case is given by the Euclidean distance between the cluster centers. For graphs, we measure the distance between clusters by first taking the normalized count of edges between them, then subtracting the normalized count to convert similarity into dissimilarity. This produces a cluster-distance matrix, δ. Let $C_1, \ldots, C_n$ be the set of vertices belonging to cluster $1, \ldots, n$, then

$$\delta_{i,j} = 1 - \frac{1}{|E|} \sum_{u \in C_i, v \in C_j} 1(u, v \in E)$$

where 1 is the indicator function (1 if (u, v) is an edge and 0 otherwise). Once δ is computed, we compute the geometric center of each embedded cluster and compute the cluster-level stress between the graph-level-cluster and realized-cluster distances. This measure is small when similar clusters are placed closer and dissimilar clusters are placed far apart. The cluster distance (CD) is:

$$CD(\delta, \chi) = \sum_{i,j} \frac{\delta_{i,j} - ||\chi_i - \chi_j||}{\delta^{i,j}}^2 \quad (3)$$

where $\delta_{i,j}$ is the dissimilarity measure between cluster i and cluster j and χ_i is the geometric center of cluster i in the embedding. Although there are several existing metrics to measure cluster accuracy, such as silhouette distance and between/within-cluster sum of squares, they are not well suited to measure the quality of intermediate embeddings. Ideally, we would need a measure that checks how well the clusters are preserved and also verifies that the relative placement of the clusters is meaningful. The CD metric verifies meaningful cluster placements by measuring all pairwise distances between cluster centers. We remark that the

CD metric works best when the clusters have convex shapes (or shapes similar to spheres). For arbitrary non-convex shapes, such as half-moons or donuts, the CD metric might not provide meaningful insights.

Stress Metric: Stress has been used in many graph embedding evaluations.

$$\text{stress}(d, X) = \sum_{i,j} \left(\frac{d_{i,j} - ||X_i - X_j||}{d^{i,j}} \right)^2 \quad (4)$$

where d is the given distance matrix and X is the embedding. Embeddings are scaled to ensure fair comparisons in computing stress [10,15,31].

4 LGS Embedding of Graphs

We start with a visual analysis and discussion of layouts produced by LGS. Following the convention, several embeddings of the same graph are displayed side-by-side with increasing the value of k from left to right, going from local to global. Underneath the LGS embeddings, we place t-SNE, UMAP, and MDS embeddings of the same graph. For all graph embeddings provided in this paper, we use the jet color scheme to encode edge length. An edge length of 1 (ideal for unweighted graphs) is drawn in green, while red indicates that edge has been compressed (length < 1) and blue indicates the edge is stretched (length > 1). This makes clusters easy to spot as bundles of red edges, and global structure preservation apparent when most edges are green. Similar to tsNET, low values of k capture local neighborhoods well, by allowing some longer edges. As a result, clusters tend to be well separated. Note that tsNET allows even longer edges in an embedding, occasionally breaking the topology; see Fig. 1. Higher k values make LGS similar to MDS, with more uniform edge lengths. This reveals global structures (e.g., mesh, grid, lattice) but may overlook clusters.

Grid_cluster is a synthetic example with 900 vertices (9 clusters of size 100 each) and 10108 edges, created by stochastic block model (SBM) to illustrate the notion of cluster distance preservation. Within cluster edges are created with probability 0.8. We distinguish between two types of out-of-cluster edges. Clusters are first placed on a lattice. Out-of-cluster edges are created with probability 0.01 if they are adjacent in the lattice (no diagonals) and 0.001 otherwise. The layouts of this graph are in Fig. 5. Note the visual similarities between LGS(32) and tsNET; both seperate each cluster into dense sub-regions and place them seemingly randomly in the plane. Also note the similarities between LGS(200) and MDS, where both methods tend to miss the clusters. UMAP also fails to capture the intermediate structure built into this network: while there is a single cluster placed in the middle of the other eight, the surrounding shape is not a square. LGS(100) accurately places each cluster in the appropriate position, making it the only one that clearly shows the 3×3 underlying lattice. Although less dense and separable the clusters are more faithful in terms of placement.

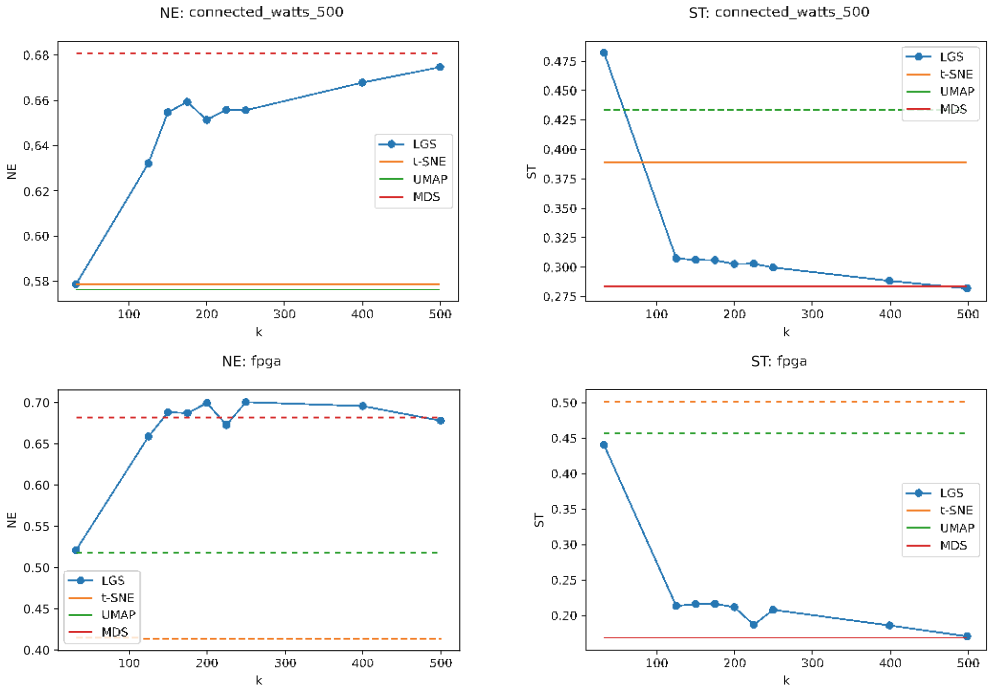


Fig. 6. Behavior of NE and stress: as k increases NE gets worse and stress gets better (LGS transitions from preserving local to global structure); tsNET, UMAP, and MDS values are shown as dotted lines for comparison. Note that in general, we expect to see an upward trend in NE, a downward trend for stress, and a parabola shape for CD.

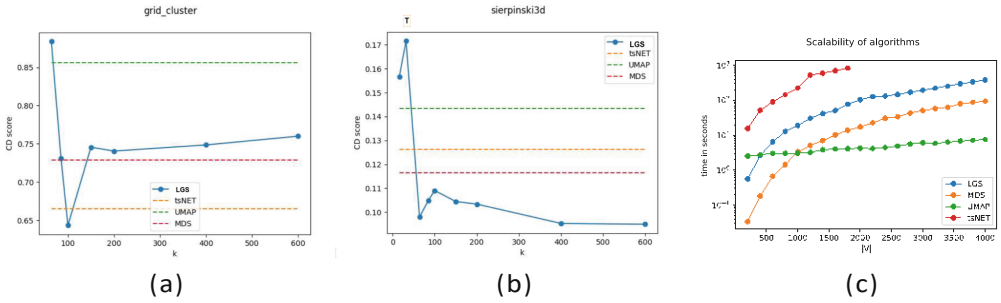


Fig. 7. (a–b) CD metric on the grid_cluster and sierpinkski3d graphs. Note that in these examples there are values of k which outperform competing algorithms. (d) Running time of each tested algorithm.

Connected_watts_1000 is a Watts-Strogatz random graph on a 1000 vertices and 11000 edges. It first assigns the vertices evenly spaced around a cycle with the nearest (7) vertices connected by an edge. Then, with low probability, some random 'chords' of the cycle are added by rewiring some of the local edges to other random vertices. This type of graph models the small-world phenomenon seen in real-world examples, such as social networks. The embeddings of connected_watts_1000, obtained by LGS, tsNET, UMAP, and MDS are in Fig. 1. We

observe that tsNET and UMAP embeddings accurately capture the existence of a one-dimensional structure, but twist and break the circle to varying degrees. Meanwhile, MDS overcrowds the space, forming a classic 'hairball' where there is no discernible structure. For intermediate values of k in LGS, the circular structure in the data and the numerous chord connections become clearly visible.

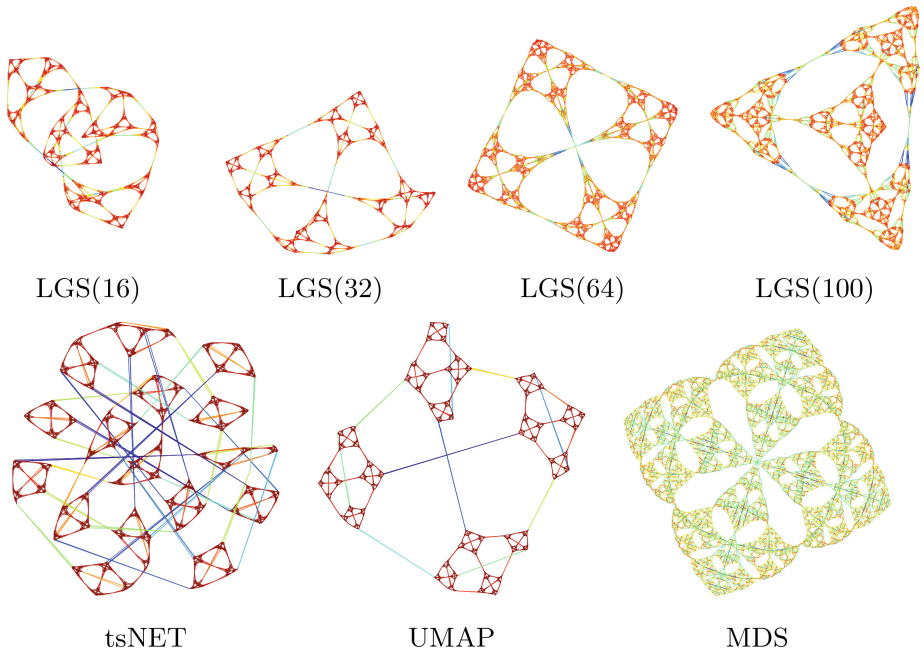


Fig. 8. Sierpinksi3d graph is a fractal with regular local and global structure. LGS manages to capture the recursive nature of the underlying structure. tsNET, UMAP miss the global placement of pyramids and MDS stretches them.

Sierpinksi_3d models the Sierpinski pyramid with 2050 vertices and 6144 edges – a finite fractal object with recursively smaller recurring patterns (the pyramid itself is built out of smaller pyramids). These fractal properties are ideal for showcasing the LGS algorithm at work, as small local structures build upon each other to create a global shape; see Fig. 8. We observe that tsNET captures the smallest structures well but places them arbitrarily in the embedding space. UMAP does better at placing the local structures in context but still creates long edges and twists not present in the data. While MDS visually captures the fractal motifs, it 'squishes' local structures. LGS can be used to balance these extremes.

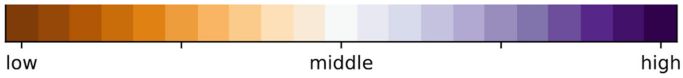
We demonstrate more examples in Table 4, with additional embeddings available in the supplemental material [19]. Note that for lower values of k the embedding obtained by LGS visually resembles the output of tsNET, while for larger values of k the obtained embedding is more similar to the outputs of MDS. We see this reflected numerically in many graphs; see Fig. 6 and Table 1. For

intermediate values of k, LGS often outperforms tsNET, UMap and MDS with respect to the intermediate structure preservation, measured by cluster distance (CD).

5 Evaluation

We test LGS on a selection of real-world and synthetic graphs from [4, 15, 31]; A full list can be found in [19].

Table 1. NE scores on LGS for varying values of k (left) and on competing algorithms (right). The colormap is normalized by row with dark orange representing the lowest score (best) and dark purple representing the highest (worst). Bold text indicates the lowest score in that row.



	k=32	k=64	k=85	k=100	k=150	k=200		tsnet	umap	mds
lesmis	0.3761	0.3169	0.3126	0.3174	0.3166	0.3125		**0.3058**	0.3952	0.3141
can_96	0.4393	0.3980	0.4605	0.4650	0.4661	0.4658		0.3523	**0.3417**	0.4653
football	0.4390	0.4469	0.4452	0.4377	0.4371	**0.4369**		0.4636	0.4693	0.4380
rajat11	0.4347	0.4374	0.4377	0.3868	0.3641	0.3615		0.3942	**0.3331**	0.3608
mesh3e1	0.1841	0.1415	0.0933	0.0971	0.1444	**0.0**		0.1210	0.1241	0.0003
connected_watts_300	0.1995	0.3990	0.5311	0.5386	0.5302	0.5365		0.1805	**0.1778**	0.5453
block_model_300	0.6474	0.6767	0.6748	0.6794	0.6874	0.6741		**0.5538**	0.5675	0.6785
powerlaw300	0.5070	0.5075	0.5126	0.5044	0.5233	0.5062		**0.3563**	0.3849	0.5075
netscience	0.4556	0.4597	0.4751	0.4860	0.5017	0.5118		0.4319	**0.3207**	0.5123
dwt_419	0.3016	0.3151	**0.2404**	0.4085	0.2666	0.3203		0.2796	0.2892	0.2421
powerlaw500	0.5942	0.5630	0.5433	0.5712	0.5593	0.5432		**0.4237**	0.4683	0.5604
block_model_500	0.5396	0.6551	0.6832	0.6945	0.7024	0.7115		**0.4377**	0.4430	0.7151
connected_watts_500	0.5788	0.5731	**0.5676**	0.5852	0.6547	0.6513		0.5789	0.5764	0.6808
grid_cluster	0.3767	0.3831	0.3767	**0.3239**	0.3456	0.3863		0.8487	0.3491	0.4742
price_1000	0.7103	0.7108	0.7228	0.7313	0.7400	0.7468		0.6015	**0.5297**	0.7943
connected_watts_1000	0.6217	0.6712	0.7870	0.8199	0.8563	0.8472		0.6283	**0.6090**	0.8390
powerlaw1000	0.6555	0.6362	0.6409	0.6382	0.6241	0.6532		**0.3956**	0.4206	0.6384
block_model_1000	0.5753	0.5552	0.5593	0.6179	0.7549	0.7777		**0.4703**	0.4880	0.7822
dwt_1005	0.4999	0.4517	0.4417	0.4586	0.4499	0.4826		**0.3692**	0.4581	0.4372
btree9	0.7337	0.8082	0.8321	0.8625	0.8812	0.8811		0.6960	**0.5876**	0.9241
CSphd	0.5816	0.5797	0.5711	0.5783	0.6015	0.6128		0.6510	**0.4378**	0.6051
fpga	0.5199	0.5875	0.6043	0.6521	0.6884	0.6998		**0.4136**	0.5179	0.6818
sierpinski3d	0.6225	0.5729	0.5313	0.5439	0.5192	**0.4252**		0.4504	0.4554	0.4600
EVA	0.4544	0.4690	0.4822	0.4797	0.5070	0.5123		0.8763	**0.2295**	0.5629

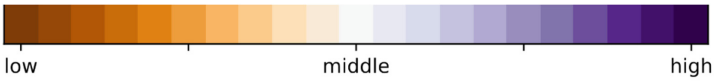
We compare LGS against state-of-the-art techniques for local and global embeddings: tsNET from the repository linked in [15], UMAP from the umap python library written by the authors of [18], and MDS via the python bindings from [30] with default parameters. Our implementation of LGS is available online. The experiments were performed on an Intel® Core™ i7-3770 machine (CPU @ 3.40 GHz × 8 with 32 GB of RAM) running Ubuntu 20.04.3 LTS.

NE, CD and Stress Values and Trends. To evaluate how well LGS preserves local neighborhoods, we compare the average NE scores over several runs and present our results in Table 1. We can see a general trend: although, tsNET

performs better with respect to NE values, LGS has consistently lower NE values than MDS. Additionally, as we increase the size of the neighborhood parameter, the NE values tend to increase by bringing the layouts closer to those of MDS. Interestingly, UMAP also tends to fall somewhere between tsNET and MDS on this metric. As expected, in many cases, LGS 'transitions' from tsNET to UMAP then finally to MDS as one goes from left to right, increasing k.

Next, we report the average CD scores for the graphs in our benchmark in Table 2. Unlike the NE values which increase as we increase k and the stress values which decrease as we increase k, the best CD values are obtained for intermediate values of k. This confirms that a balance between local and global optimization is needed to capture intermediate structures.

Table 2. CD scores following the same scheme as Table 1. NA indicates no clusters present in the data.



	k=32	k=64	k=85	k=100	k=150	k=200		tsnet	umap	mds
lesmis	0.7308	**0.6745**	0.7116	0.7210	0.7135	0.7260		0.6949	0.6904	0.7195
can_96	0.7088	0.6701	0.6669	0.6666	0.6666	0.6666		**0.5174**	0.6747	0.6666
football	0.7747	0.7580	0.7911	0.7933	0.7983	0.8002		0.7804	**0.7090**	0.7979
rajat11	0.4092	0.4045	**0.3672**	0.3862	0.4083	0.4075		0.4607	0.5093	0.4080
mesh3e1	0.4087	0.3951	0.3824	0.3879	0.3865	0.3674		0.3948	0.3739	**0.3673**
connected_watts_300	0.3044	0.2445	0.2098	0.1989	0.1966	0.1968		0.2853	0.3028	**0.1929**
block_model_300	0.7454	0.7726	0.7994	0.7646	0.7727	0.7980		0.7767	**0.7307**	0.7901
powerlaw300	N/A	N/A	N/A	N/A	N/A	N/A		N/A	N/A	N/A
netscience	0.2716	0.2720	0.2717	0.2698	0.2649	0.2689		0.3222	0.3775	**0.2439**
dwt_419	0.1938	0.1899	0.1898	0.1962	0.1925	0.1943		**0.1657**	0.2173	0.1880
powerlaw500	0.4923	0.4332	0.4174	0.4336	0.4066	0.4249		**0.3607**	0.4261	0.4271
block_model_500	**0.6414**	0.7065	0.6626	0.6837	0.6819	0.6855		0.6551	0.7569	0.6975
connected_watts_500	0.2480	0.2268	0.2321	0.2378	0.2235	0.2191		0.2417	0.2294	**0.2120**
grid_cluster	0.9131	0.7905	0.8347	0.8040	0.7488	0.7415		0.7786	0.7454	**0.7335**
price_1000	N/A	N/A	N/A	N/A	N/A	N/A		N/A	N/A	N/A
connected_watts_1000	0.2734	0.2304	0.2946	0.2802	0.3648	0.2894		0.2333	0.2323	**0.2184**
powerlaw1000	0.3917	0.3671	0.3754	0.3640	**0.3590**	0.3775		0.4362	0.4210	0.3683
block_model_1000	0.6308	**0.5358**	0.5447	0.5435	0.5853	0.5419		0.5632	0.5977	0.5991
dwt_1005	0.1597	0.1414	0.1413	0.1466	0.1414	0.1415		0.1438	0.1456	**0.1386**
btree9	N/A	N/A	N/A	N/A	N/A	N/A		N/A	N/A	N/A
CSphd	N/A	N/A	N/A	N/A	N/A	N/A		N/A	N/A	N/A
fpga	0.1692	0.1511	0.1493	0.1601	0.1497	0.1506		0.1770	0.1966	**0.1305**
sierpinski3d	0.1252	0.1156	0.1050	0.1038	0.1021	0.1016		0.1244	0.1321	**0.0974**
EVA	N/A	N/A	N/A	N/A	N/A	N/A		N/A	N/A	N/A

We compute and report the averaged stress scores in Table 3. MDS is consistently good at minimizing the stress, but we see a salient trade-off between the stress scores of LGS's tsNET-like embeddings with low k values and LGS's MDS-like embeddings with high k values. When we look at small neighborhoods such as k = 16, we tend to see high stress values, however, the values decrease as we expand the neighborhoods. UMAP does not seem to capture global structure well for these graphs, often having the highest stress values.

Table 3. Stress scores following the same scheme as Table 1

	k=32	k=64	k=85	k=100	k=150	k=200		tsnet	umap	mds
lesmis	0.1988	0.1702	0.1673	0.1670	0.1665	**0.1656**		0.2290	0.4058	0.1661
can_96	0.1782	0.1526	0.1400	**0.1390**	0.1391	0.1391		0.2315	0.2237	0.1393
football	0.2719	0.2622	0.2578	0.2549	0.2544	**0.2544**		0.3521	0.3888	0.2548
rajat11	0.1636	0.1685	0.1774	0.1474	0.1250	**0.1246**		0.3167	0.4076	0.1251
mesh3e1	0.1125	0.0655	0.0328	0.0482	0.0613	0.0050		0.0858	0.0251	**0.0049**
connected_watts_300	0.4984	0.2359	0.2353	0.2267	0.2008	0.2081		0.4218	0.4702	**0.1896**
block_model_300	0.3592	0.3132	0.3087	0.3051	0.3052	0.2929		0.3889	0.4437	**0.2863**
powerlaw300	0.1975	0.1848	0.1741	0.1668	0.1644	0.1532		0.2619	0.3731	**0.1505**
netscience	0.1914	0.1539	0.1604	0.1627	0.1493	0.1472		0.3296	0.5198	**0.1131**
dwt_419	0.0688	0.0638	0.0360	0.1206	0.0521	0.0759		0.1417	0.1646	**0.0312**
powerlaw500	0.5465	0.2172	0.2023	0.2133	0.2022	0.1893		0.2643	0.3413	**0.1772**
block_model_500	0.3360	0.3224	0.3160	0.3108	0.3047	0.3016		0.4047	0.4510	**0.2855**
connected_watts_500	0.4816	0.3380	0.3367	0.3279	0.3055	0.3019		0.3884	0.4332	**0.2826**
grid_cluster	0.5494	0.5335	0.5814	0.3049	0.2553	0.2478		0.4195	0.3760	**0.2371**
price_1000	0.3543	0.2428	0.2320	0.2347	0.2127	0.2011		0.3654	0.5721	**0.1848**
connected_watts_1000	0.5578	0.3867	1.0963	0.9221	1.6998	0.8549		0.4169	0.4460	**0.3166**
powerlaw1000	0.3012	0.2418	0.2344	0.2410	0.2129	0.2196		0.2907	0.3615	**0.1815**
block_model_1000	0.4389	0.3456	0.3444	0.3409	0.3292	0.3212		0.4078	0.4379	**0.2921**
dwt_1005	0.1858	0.1206	0.0939	0.1165	0.0891	0.1355		0.3556	0.0669	**0.0424**
btree9	0.4973	0.3090	0.3510	0.2740	0.2818	0.2563		0.3539	0.3603	**0.2307**
CSphd	0.2642	0.2483	0.2218	0.1925	0.2051	0.1883		0.2943	0.4243	**0.1446**
fpga	0.4414	0.3139	0.2888	0.2498	0.2157	0.2113		0.5018	0.4575	**0.1679**
sierpinski3d	0.5063	0.3015	0.2097	0.2304	0.1769	0.1298		0.5194	0.2525	**0.1252**
EVA	0.6843	1.0466	0.7862	0.6387	0.4662	0.6394		0.3535	0.4913	**0.1907**

Effect of k on Evaluation Metrics: To visually explain LGS behavior, we plot examples NE, CD, and stress with respect to k. In Fig. 6, we demonstrate two separate plots for each graph. It can be seen that we often fall in between the values of NE and stress that tsNET and MDS reach. These plots show what we expect to see: as k increases NE increases and stress decreases. In Fig. 7(a-b), we plot the CD values of our layout with tsNET, UMAP, MDS for comparison. In many layouts LGS indeed has the lowest CD score. Values of k were chosen to be representative of the local-global tradeoff.

6 Discussion and Limitations

We described LGS: an adaptable algorithmic framework for embeddings that can prioritize local neighborhoods, global structure, or a balance between the two. LGS provides flexible structure preservation choices with comparable embedding quality to previous single-purpose methods (local or global), while also outperforming state-of-the-art methods in preserving intermediate structures.

There are several limitations: Our results are based on a small number of graphs. Additional systematic experimentation would further support the usefulness of LGS Some experiments for high-dimensional datasets are in [19]. LGS modifies MDS's objective function to accommodate varying neighborhood sizes, similarly, one could adapt the KL divergence cost function of t-SNE. Note that t-SNE's perplexity parameter ostensibly controls the size of a neighborhood, but high perplexity values do not result in global structure preservation [29].

LGS algorithm has several hyperparameters, including c, α, and k. We provide default values for c and α based on experiments, and leave k as a true hyperparameter. Our intention is for a visualization designer to adjust k as

needed; to generate a spectrum of embeddings to get a sense of both local and global properties of a dataset. An interactive LGS version is not yet available. While LGS runs in seconds for graphs with a few thousand vertices, the running time can become untenable for larger instances, due to the $O(|V|^2)$ optimization per epoch, and pre-processing with APSP. While LGS's runtime is comparable with those of tsNET and MDS (see Fig. 7(c)) both can be sped up through the use of approximations [9,23,31], Speeding up LGS is a potential future work.

References

1. Börsig, K., Brandes, U., Pasztor, B.: Stochastic gradient descent works really well for stress minimization. In: GD 2020. LNCS, vol. 12590, pp. 18–25. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-68766-3_2
2. Cai, H., Zheng, V.W., Chang, K.C.: A comprehensive survey of graph embedding: problems, techniques, and applications. IEEE Trans. Knowl. Data Eng. 30(9), 1616–1637 (2018)
3. Chen, L., Buja, A.: Local multidimensional scaling for nonlinear dimension reduction, graph drawing, and proximity analysis. J. Am. Stat. Assoc. 104(485), 209–219 (2009)
4. Davis, T.A., Hu, Y.: The University of Florida sparse matrix collection. ACM Trans. Math. Softw. 38(1), 1:1–1:25 (2011)
5. De Leeuw, J.: Applications of convex analysis to multidimensional scaling (2005)
6. Ertoz, L., Steinbach, M., Kumar, V.: A new shared nearest neighbor clustering algorithm and its applications. In: Workshop on Clustering High Dimensional Data and Its Applications at 2nd SIAM International Conference on Data Mining, vol. 8 (2002)
7. Espadoto, M., Martins, R.M., Kerren, A., Hirata, N.S.T., Telea, A.C.: Toward a quantitative survey of dimension reduction techniques. IEEE Trans. Vis. Comput. Graph. 27(3), 2153–2173 (2021)
8. Frey, D., Pimentel, R.: Principal component analysis and factor analysis (1978)
9. Fu, C., Zhang, Y., Cai, D., Ren, X.: AtSNE: efficient and robust visualization on GPU through hierarchical optimization. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 176–186 (2019)
10. Gansner, E.R., Hu, Y., North, S.C.: A maxent-stress model for graph layout. IEEE Trans. Vis. Comput. Graph. 19(6), 927–940 (2013)
11. Gansner, Emden R.., Koren, Yehuda, North, Stephen: Graph drawing by stress majorization. In: Pach, J́anos. (ed.) GD 2004. LNCS, vol. 3383, pp. 239–250. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-31843-9_25
12. Ghojogh, B., Ghodsi, A., Karray, F., Crowley, M.: Uniform manifold approximation and projection (UMAP) and its variants: Tutorial and survey. CoRR abs/2109.02508 (2021)
13. Jolliffe, I.T.: Principal Component Analysis. Springer Series in Statistics, Springer (1986)
14. Kamada, T., Kawai, S.: An algorithm for drawing general undirected graphs. Inf. Process. Lett. 31(1), 7–15 (1989)
15. Kruiger, J.F., Rauber, P.E., Martins, R.M., Kerren, A., Kobourov, S.G., Telea, A.C.: Graph layouts by t-SNE. Comput. Graph. Forum 36(3), 283–294 (2017)

16. Kruskal, J.B.: Multidimensional scaling by optimizing goodness of fit to a non-metric hypothesis. Psychometrika 29(1), 1–27 (1964)
17. Van der Maaten, L., Hinton, G.: Visualizing data using t-SNE. J. Mach. Learn. Res. 9(11) (2008)
18. McInnes, L., Healy, J.: UMAP: uniform manifold approximation and projection for dimension reduction. CoRR abs/1802.03426 (2018)
19. Miller, J., Huroyan, V., Kobourov, S.: Balancing between the local and global structures (LGS) in graph embedding (2023). https://arxiv.org/abs/2308.16403
20. Munzner, T.: Visualization Analysis and Design. CRC Press, Boca Raton (2014)
21. Nguyen, Q.H., Eades, P., Hong, S.: On the faithfulness of graph visualizations. In: Carpendale, S., Chen, W., Hong, S. (eds.) IEEE Pacific Visualization Symposium, PacificVis 2013, 27 February 2013 – 1 March 2013, Sydney, NSW, Australia, pp. 209–216. IEEE Computer Society (2013)
22. Nocaj, A., Ortmann, M., Brandes, U.: Untangling the hairballs of multi-centered, small-world online social media networks. J. Graph Algorithms Appl. 19(2), 595–618 (2015)
23. Ortmann, M., Klimenta, M., Brandes, U.: A sparse stress model. J. Graph Algorithms Appl. 21(5), 791–821 (2017)
24. Purchase, H.C.: Metrics for graph drawing aesthetics. J. Vis. Lang. Comput. 13(5), 501–516 (2002)
25. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. Science 290(5500), 2323–2326 (2000)
26. Shepard, R.N.: The analysis of proximities: multidimensional scaling with an unknown distance function. I. Psychometrika 27(2), 125–140 (1962)
27. Tamassia, R.: Handbook of Graph Drawing and Visualization. CRC Press, Boca Raton (2013)
28. Tenenbaum, J.B., Silva, V.D., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science 290(5500), 2319–2323 (2000)
29. Wattenberg, M., Viégas, F., Johnson, I.: How to use t-SNE effectively. Distill (2016). https://doi.org/10.23915/distill.00002
30. Zheng, J.X., Pawar, S., Goodman, D.F.M.: Graph drawing by stochastic gradient descent. IEEE Trans. Vis. Comput. Graph. 25(9), 2738–2748 (2019)
31. Zhu, M., Chen, W., Hu, Y., Hou, Y., Liu, L., Zhang, K.: DRGraph: an efficient graph layout algorithm for large-scale graphs by dimensionality reduction. IEEE Trans. Vis. Comput. Graph. 27(2), 1666–1676 (2021)