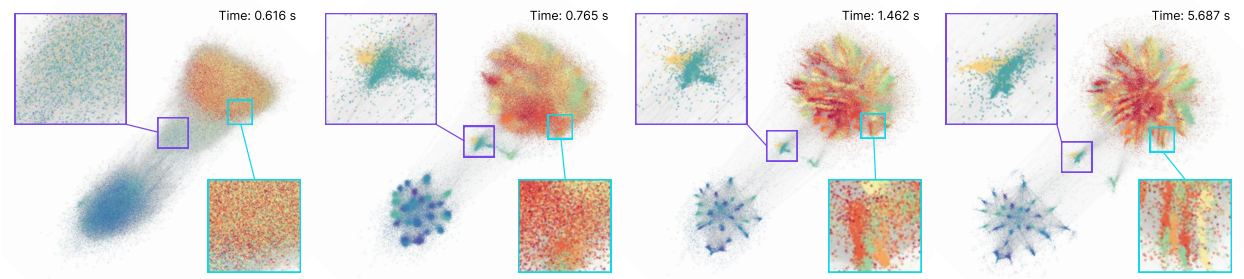
SNAP-tFDP: Massively Scalable Graph Layouts via Sparse Negative Sampling

Xin Chen, Shuowei Hou, Yifan Wang, Mingliang Xue, Zezheng Feng, Oliver Deussen, Weidong Huang, and Yunhai Wang



(a) Initial layout (b) One epoch (c) 5 epochs (d) 30 epochs

Fig. 1: Iterative layouts of communities with more than 800 users in the Orkut online social network [44] (a graph with 259,019 nodes and 6,893,156 edges) generated using our negative sampling-based graph layout algorithm using t-forces [47], which runs in $O(|E|)$ time. Node colors indicate community membership.The initial node positions are obtained using Pivot MDS (PMDS) [5] (a), followed by layouts after 1 (b), 5 (c), and 30 (d) epochs of the algorithm. The entire layout process required 5.69 seconds with 16 CPU threads (120 MByte RAM), whereas DRgraph [49] required 55.64 seconds on the same settings.

Abstract—Force-Directed Placement (FDP) is a widely used approach for network visualization, yet scaling it to massive graphs while preserving clear community structures remains a major computational and visual challenge. Existing approximation methods often rely on auxiliary data structures (e.g., spatial trees), which introduce substantial memory overhead; furthermore, traditional power-function-based forces frequently fail to separate dense clusters effectively. In this paper, we present a negative sampling-based algorithm that achieves $O(|E|)$ time complexity with a low memory footprint, without requiring complex multi-level representations. In a first step, we introduce a linearly normalized degree-weighting scheme, which, combined with short-range bounded t-distribution forces, effectively untangles dense structures and enhances visual cluster separation. To optimize for this formulation efficiently, we introduce an edge-centric negative sampling strategy that naturally reconstructs the global degree-weighted objective. Furthermore, we design a lock-free, bundle-based parallelization scheme that leverages the sparsity of stochastic updates to achieve significant speedups while mitigating access conflicts. Comprehensive evaluations on 12 large-scale graphs demonstrate that the proposed method outperforms state-of-the-art algorithms in neighborhood preservation and cluster separation. Compared to existing baselines, our method reduces memory consumption by 72% on average and leverages simple GPU parallelism to generate a high-quality layout for a graph with 4 million nodes and 34 million edges in below 10 seconds.

Index Terms—Graph Layout, Network Visualization, Negative Sampling

1 Introduction

Graphs are a fundamental data structure for representing relational information, in which entities are modeled as nodes and their relationships are encoded by edges. Node-link diagrams are one of the most widely used visualization techniques for graphs [24], as they support intuitive exploration and analysis of complex relational structures. Among algorithms for generating graph layouts in node-link diagrams, force-directed placement (FDP) methods have attracted substantial attention [12, 15, 22, 47]. Yet, with the rapid growth of data, traditional FDP algorithms become increasingly inadequate in terms of design objectives and computational efficiency.

From a design perspective, FDP methods primarily emphasize individual nodes and edges and optimize a set of aesthetic criteria, such as distribute nodes evenly, have uniform edge lengths or reduce edge crossings [11]. For large and complex graphs containing hundreds of thousands of nodes, however, node and edge overlaps are often unavoidable due to limited screen space (see Figure 1a). Moreover, the uniform distribution of nodes and edges enforced by most FDP methods can obscure clusters and community structures, thereby hindering cluster-oriented analysis tasks [28]. To improve the perceptual separation of clusters, LinLog [29] and ForceAtlas2 [21] apply *degree-based weighting* to repulsive forces, which increases separation between high-degree nodes while allowing low-degree nodes to remain closer to their associated high-degree nodes. Although this strategy improves visual grouping to some extent, their power-function-based forces inherently suffer from exerting extremely large repulsive forces and small attractive forces for short-ranges [47]. As a result, for graphs with massive numbers of nodes, layout stability and clustering quality can deteriorate. Therefore, enhancing cluster visibility in large-scale and complex graphs remains an open problem.

When scaling to massive graphs, a primary concern is not only perceptual cluster quality but, more critically, computational efficiency. This has motivated extensive research into approximation-based strate-

• *X. Chen, S. Hou and Y. Wang are with Renmin University of China. E-mail: {chenxin19961029, 2025104058, wang.yh}@ruc.edu.cn.*
• *Y. Wang is with Shandong University. E-mail: yfwang2001@gmail.com.*
• *M. Xue is with Shandong Provincial Institute of Educational Sciences. E-mail: xml95007@gmail.com.*
• *Z. Feng is with Northeastern University,. E-mail: fengzezheng@swc.neu.edu.cn.*
• *O. Deussen is with University of Konstanz. E-mail: oliver.deussen@uni-konstanz.de.*
• *W. Huang is with University of Technology Sydney. E-mail: weidong.huang@uts.edu.au.*
• *Yunhai Wang is the corresponding author.*

Manuscript received xx xxx. 202x; accepted xx xxx. 202x. Date of Publication xx xxx. 202x; date of current version xx xxx. 202x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.202x.xxxxxxx

gies for force-directed algorithms. The classical Barnes-Hut (BH) approximation [1] employs a spatial partitioning tree, which aggregates nodes to estimate long-range repulsive forces, reducing the time complexity from $O(|V|^2)$ to $O(|V|\log|V|)$. t-FDP [47] achieves $O(|V|)$ complexity by exploiting a low-rank kernel and projecting node interactions onto an interpolation grid, at the cost of increased space complexity. Building on tsNET [23], which employs t-SNE to capture local neighborhood structures, DRGraph [49] integrates a sparse distance matrix with a multilevel layout scheme and negative sampling to achieve a time complexity of $O(|V|+|E|)$. However, the combination of multiple complex components inhibits isolating the contribution of each component and limits applicability across broader scenarios. More recently, Omega [31] combines a low-rank resistance-distance embedding with random node-pair sampling, enabling stress optimization in $O(|E|)$ time without relying on pivot-based approximations. Nevertheless, all these methods require additional memory for auxiliary data structures, which constrains their scalability to very large graphs.

In this paper, we present SNAP-tFDP (**S**tochastic **N**egative-sampling **A**ccelerated **P**lacement **t-FDP**), which runs in $O(|E|)$ time without relying on auxiliary data structures. In contrast to prior degree-weighting approaches built on power-function-based forces, we employ the t-distribution-based repulsive force that is bounded at short ranges. This design maintains the stability of local adjacency relationships, leading to clustered rendering of communities and clearer separation between them. Building on this design, we compare multiple degree-weighting schemes and find that an additive degree weight normalized by the total number of nodes, $(d_i+d_j)/2|V|$, effectively guides cluster contraction and strengthens visual grouping.

To efficiently apply degree-weighting for repulsive forces, we introduce an edge-centric negative sampling strategy. It requires only two essential data structures: a list of all edges and a list of current node positions. In each iteration, the algorithm traverses all edges to compute attractive forces, and for each edge, k random samples are drawn uniformly from the node position list to compute repulsive forces. By randomly shuffling the edge order and updating node positions immediately after each force computation, the algorithm incorporates the principle of stochastic gradient descent (SGD), which speeds up convergence. The expected value of the repulsive forces produced by this process are equal to the full repulsive forces weighted by $k\cdot(d_i+d_j)/2(|V|-1)$, resulting in an optimization loss that closely matches the corresponding full-force objective.

In addition, we introduce a lock-free parallelization strategy for the proposed graph layout algorithm. Building on the sparsity of the stochastic updates, the method follows a HOGWILD!-style design [36] that avoids synchronization while maintaining stable convergence. To further reduce access conflicts in practice, we systematically group all out-edges from a common node i and pack them into a computational bundle. This entire bundle is then assigned to a single, dedicated thread. This design lowers the probability of write conflicts, improves cache locality, and preserves the convergence behavior of the serial algorithm while significantly improving efficiency.

We evaluated the effectiveness of our proposed method on 12 graph datasets of varying sizes, with node counts ranging from 7 thousand to 4 million and edge counts from 80 thousand to 117 million. In terms of layout quality, we use metrics such as the Silhouette Index [37] to show that SNAP-tFDP outperforms existing state-of-the-art methods in neighbor preservation, cluster separation, and visual grouping. In terms of efficiency, SNAP-tFDP reduces memory consumption by 72% compared with the most memory-efficient baseline and achieves a 150% speedup over the fastest existing method on average in the serial setting. Due to its simple design, SNAP-tFDP can be efficiently implemented on GPUs and achieves the *com-lj* graph's layout (4 million nodes and 34 million edges) in just 9.3 s using 0.8 GB GPU memory, whereas the state-of-the-art method DRGraph, which only provides a CPU implementation, requires 236 s with 16 CPU threads. In addition, we conduct a case study on the *com-friendster* graph (65.6 million nodes and 1.8 billion edges) to further demonstrate the applicability of SNAP-tFDP to ultra-large-scale graphs.

In summary, the main contributions of this paper are as follows:

- Based on t-distributed forces, we compare multiple degree-weighting schemes and reveal the key role of linear normalized degree-weighting in rendering cluster contraction and strengthening visual grouping.
- We propose a negative sampling-based graph layout method (SNAP-tFDP) that uses an edge-centric sampling strategy, which matches the expected value of the sampled repulsive force equivalent to the full weighted repulsive force while reducing the time complexity to $O(|E|)$ and lowering memory overhead.
- We reveal the sparsity of negative sampling-based graph layout and accordingly design a lock-free, bundle-based parallelization strategy, improving parallel efficiency while introducing only limited approximation error.

2 Related Work

Related work can be broadly categorized into two areas: graph layout methods, approximation techniques, and parallelization strategies.

2.1 Graph Layout Methods

Graph layout methods aim to visualize network data as node-link diagrams, positioning nodes and edges to reveal the underlying structural information of the graph. The most common approaches fall into three primary categories: spring-electrical models, stress models, and dimensionality reduction-based techniques.

Spring-electrical models treat a graph as a physical system in which nodes repel each other like charged particles, while edges pull connected nodes together like springs. The optimal layout is achieved when these forces reach equilibrium. Early approaches, such as the Fruchterman-Reingold model [15], define attractive forces that increase with distance and repulsive forces that decrease with distance, producing layouts with relatively uniform edge lengths. Later methods, including LinLog [29] and ForceAtlas2 (FA2) [21], incorporate degree-aware repulsive forces to enhance the visual separation of clusters. More recently, the t-FDP model [47] replaces the traditional power-function repulsive force with a bounded short-range force derived from the Student's t-distribution. This formulation limits the magnitude of repulsive forces between nearby nodes while maintaining similar long-range behavior, which improves the preservation of local neighborhood relationships in the resulting layouts. Building on t-FDP, this paper further introduces linear normalized degree-weighting and edge-centric negative sampling, enabling efficient untangling of dense structures and revelation of community patterns without extra overhead.

Stress models [22] traditionally associate a spring with every pair of nodes, where the ideal spring length is proportional to the graph-theoretic shortest path distance. By minimizing the discrepancy between Euclidean distances in the layout and graph-theoretic distances, stress models typically preserve global graph structure better than spring-electric models. However, this global objective often weakens local structures and incurs high computational cost due to all-pairs shortest path computation. To alleviate these issues, several local and hybrid variants have been proposed. For example, the Taurus framework [43] further combines stress and spring-electrical models to support multi-perspective graph visualization. Despite these improvements, stress models often produce dense "hairball" layouts, making community structures difficult to identify. The most recent advance, Omega [31], addresses the "hairball" problem by introducing a stress model based on low-rank resistance distances derived from the graph Laplacian, achieving $O(|E|)$ complexity; however, it incurs extra preprocessing time and relies on auxiliary data structures, which SNAP-tFDP avoids.

Dimensionality reduction-based techniques formulate graph layout as a low-dimensional embedding problem, typically seeking to preserve graph-theoretic distances or neighborhood similarities, while recent studies have explored the reverse direction by adapting graph drawing methods to Dimensionality Reduction (DR) [18,33,35]. Representative examples include tsNET [23] and DRGraph [49], which adopt the force models of t-SNE [39] and UMAP [27], respectively. Böhm et al. [3] further showed that t-SNE, UMAP, FA2, and Laplacian Eigenmaps (LE) can be expressed within a common attraction-repulsion framework. Rather than adapting existing DR force models or unifying

existing methods, this paper identifies the implicit degree weighting induced by negative sampling and shows that it naturally complements t-distributed forces, leading to improved cluster visibility.

2.2 Approximation Techniques for Graph Layout

The primary challenge of traditional FDP algorithms is their computational cost. In a standard FDP iteration, computing exact repulsive forces between all pairs of nodes requires $O(|V|^2)$ time, which becomes a severe bottleneck as the graph size grows. To improve scalability for large networks, various approximation techniques have been developed to reduce the cost of computing repulsive forces.

Multi-level Techniques address this computational bottleneck by approximating the influence of distant nodes through hierarchical aggregation. The Barnes-Hut (BH) method [1] organizes nodes into a spatial tree structure (e.g., quadtree or octree) and approximates the repulsive force from a distant group of nodes using their center of mass, reducing the complexity to $O(|V|\log|V|)$. Graph coarsening techniques provide another form of multi-level acceleration. Walshaw [41] constructs a hierarchy of progressively coarsened graphs using edge contraction, enabling faster layout computation while preserving global topology. Building on this idea, Hu [19] proposed the SFDP algorithm, which combines multi-level graph coarsening with the Barnes-Hut approximation to efficiently generate layouts for large graphs. Although effective, these methods require maintaining spatial trees or graph hierarchies, leading to additional preprocessing and memory costs. Consequently, achieving scalable and responsive performance for graphs with millions of nodes and edges remains challenging.

Sampling-based Approximations provide an alternative strategy to improve scalability by either reducing the graph size or approximating the force computations directly. Traditional graph sampling approaches extract a proxy graph by removing a portion of the edges based on specific metrics, such as spectral characteristics [2] or connectivity [48]. However, modifying the original topology can remove subtle but important connections, leading to incomplete visual representations. To maintain the full topology, recent methods integrate sampling directly into the force computation. For example, the Random Vertex Sampling (RVS) algorithm [17] updates node positions using repulsive forces calculated from a small, randomly selected subset of nodes, achieving $O(|V|)$ time complexity. Random Edge Sampling (RES) [16] similarly reduces attractive force computations. Although these random sampling methods are faster than multi-level techniques, the resulting layouts often suffer from degraded visual quality.

Contrastive Learning and Negative Sampling provide a different optimization paradigm that aligns well with force-directed layouts. Contrastive learning methods learn representations by pulling similar (positive) pairs closer and pushing dissimilar (negative) pairs apart in an embedding space [7]. Objectives such as the InfoNCE loss [32] are typically optimized using SGD with negative sampling, which avoids the cost of evaluating all possible pairs. This idea has also been adopted in dimensionality reduction techniques such as UMAP [27], which implicitly optimize contrastive objectives to preserve local neighborhood structures. For graph layout, DRGraph [49] employs negative sampling to estimate gradients of its objective function, combined with a sparse distance matrix and a multi-level scheme to reduce the computational complexity to $O(|V|+|E|)$.

Building on this idea, we reformulate the FDP layout process as a contrastive learning problem. By introducing an edge-centric negative sampling strategy, our method naturally incorporates normalized degree weighting without requiring auxiliary multi-level data structures. As a result, the proposed algorithm achieves $O(|E|)$ time complexity while maintaining stable community structures with low memory overhead, making it suitable for large-scale graph visualization.

2.3 Parallelization Strategies for Graph Layout

Another important approach for scaling graph layout algorithms is parallelization. In graph embedding, Force2Vec [35] reformulates force-directed computations as sparse matrix operations and exploits thread-level and SIMD parallelism, achieving substantial speedups over random-walk-based methods. However, it targets high-dimensional embeddings rather than 2D layouts. Rahman et al. [34] proposed Batch-Layout, which processes vertices in minibatches and improves memory locality through cache blocking. Its Barnes-Hut variant achieves near-linear speedups on multicore CPUs but still relies on tree-based force approximations. DRGraph [49] combines multithreading with negative sampling and multilevel coarsening. However, its coarse-to-fine refinement process introduces dependencies that limit parallel scalability and complicate GPU implementation. In contrast, SNAP-tFDP employs a lock-free, bundle-based parallelization scheme inspired by HOGWILD! [36], avoiding auxiliary data structures and synchronization overhead. This design enables efficient execution on both multicore CPUs and GPUs while maintaining a low memory footprint.

Table 1: Force formulations in different spring-electrical models. $\mathbf{y}_i$ denotes the two-dimensional position of vertex i in the layout space, and $\mathbf{e}_{ij}=\frac{\mathbf{y}_i-\mathbf{y}_j}{\|\mathbf{y}_i-\mathbf{y}_j\|}$ is the unit vector indicating the force direction.

Method	Attractive Force	Repulsive Force
FR [15]	$\|\|\mathbf{y}_i-\mathbf{y}_j\|\|^2\mathbf{e}_{ij}$	$\frac{-1}{\|\|\mathbf{y}_i-\mathbf{y}_j\|\|}\mathbf{e}_{ij}$
LinLog [29]	$1*\mathbf{e}_{ij}$	$\frac{-d_i\cdot d_j}{\|\|\mathbf{y}_i-\mathbf{y}_j\|\|}\mathbf{e}_{ij}$
FA2 [21]	$\|\|\mathbf{y}_i-\mathbf{y}_j\|\|\mathbf{e}_{ij}$	$\frac{-(d_i+1)\cdot(d_j+1)}{\|\|\mathbf{y}_i-\mathbf{y}_j\|\|}\mathbf{e}_{ij}$
t-FDP [47]	$\alpha(\|\|\mathbf{y}_i-\mathbf{y}_j\|\|+\frac{\beta\|\|\mathbf{y}_i-\mathbf{y}_j\|\|}{(1+\|\|\mathbf{y}_i-\mathbf{y}_j\|\|^2)})\mathbf{e}_{ij}$	$-\frac{\|\|\mathbf{y}_i-\mathbf{y}_j\|\|}{(1+\|\|\mathbf{y}_i-\mathbf{y}_j\|\|^2)^\gamma}\mathbf{e}_{ij}$

3 Method

To improve cluster separability and computational efficiency in large-scale undirected graph layout, our proposed SNAP-tFDP algorithm adopts a memory-efficient negative sampling scheme that implicitly induces degree-based weighting. In combination with t-distribution-based forces, this formulation mitigates cluster overlap and more faithfully preserves group-level proximity. This section first revisits spring-electrical models and degree-weighting strategies, then examines their behavior under t-forces, subsequently introduces an edge-centric negative sampling formulation with a consistent optimization loss, and concludes with a lock-free parallel implementation of SNAP-tFDP.

3.1 Revisiting Existing Spring-Electrical Models

Given an undirected graph $G=(V,E)$, where $V=\{1,\dots,n\}$ denotes the set of vertices and $E\subseteq V\times V$ denotes the set of edges, and d_i denotes the degree of vertex i, graph layout refers to assigning each vertex $i\in V$ a two-dimensional position $\mathbf{Y}=\{\mathbf{y}_1,\dots,\mathbf{y}_n\}\in\mathbb{R}^{n\times 2}$.

Due to their intuitive physical analogy and effective visual outcomes, *spring-electrical models* are widely used for graph layout. They model layout as a physical system governed by two principles: adjacent vertices are drawn together, while all vertices repel to prevent overlap. The net force on vertex i, denoted as $F(i)$, is the sum of attractive forces $F^a(i,j)$ and repulsive forces $F^r(i,j)$:

$$F(i)=\sum_{(i,j)\in E}F^a(i,j)+k\sum_{j\neq i}F^r(i,j), \tag{1}$$

where k is a coefficient for controlling the relative strength of repulsive forces. Table 1 summarizes representative force formulations.

Force-directed layouts are usually based on the principle of potential energy minimization. The entire system corresponds to an energy function $U(\mathbf{Y})$, and the force acting on a vertex can be expressed as the negative gradient of this energy function with respect to the vertex position [45]:

$$F(\mathbf{y}_i)=-\nabla_{\mathbf{y}_i}U(\mathbf{Y}). \tag{2}$$

The layout is obtained via iterative energy minimization until equilibrium, where the relative vertex positions encode the graph's structural properties for analysis.

Classical Models. Because of their simple form and intuitive physical interpretation, most early spring-electrical models define attractive and repulsive forces using power functions:

$$F^a(i,j)=\|\mathbf{y}_i-\mathbf{y}_j\|^p,\qquad F^r(i,j)=-\|\mathbf{y}_i-\mathbf{y}_j\|^{-q}. \tag{3}$$

In this formulation, a larger exponent p results in stronger attraction at long distances, while a larger exponent q leads to faster decay of the

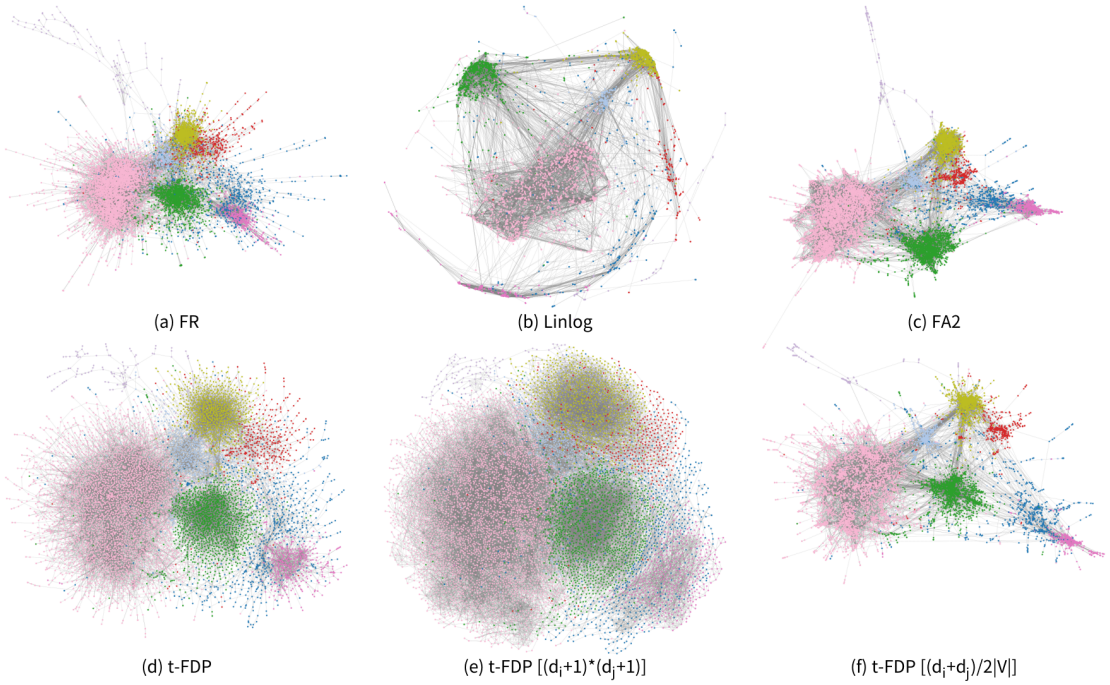


Fig. 2: Comparison of classical spring-electrical models and t-FDP models with different weighting schemes.

repulsive force at long distances.

The Fruchterman-Reingold (FR) model [15] with $p = 2, q = 1$ yields near-uniform edge lengths but weak community discrimination, often producing a "hairball" layout (see Figure 2). The LinLog model [29] adopts $p = 0, q = 1$ and degree-based weighting for repulsion $d_i \cdot d_j$, attracting low-degree vertices to high-degree centers and scaling inter-cluster repulsion by the connection strength between communities, which improves cluster separation. However, the product-based repulsion induces excessive intra-cluster expansion and reduces layout compactness.

Forceatlas2 (FA2) [21] adopts $p = 1, q = 1$ and replaces the repulsive weight with $(d_i + 1) \cdot (d_j + 1)$, yielding layouts intermediate between the above two extremes: improved cluster separation with stronger attraction to limit dispersion. Nevertheless, it exhibits cluster overlap similar to FR (e.g., the red cluster in Figure 2c), reflecting limitations of power-law force formulations.

t-FDP Model. Despite their widespread use, power-function force models exhibit unbounded short-range repulsion, causing instability and potential cluster distortion. To address this issue, tsNET [23] and DRGraph [49] adopt t-distribution-based force models, while t-FDP [47] further formalizes the resulting *t-force* (see Table 1) and analyzes its properties within the spring-electrical framework. The t-force decouples short- and long-range interactions: it remains bounded and strengthens attraction while limiting excessive repulsion at short ranges, and preserves near-linear attraction with weak repulsion at long ranges. Consequently, it maintains global structures while improving local adjacency, yielding more distinct, less overlapping clusters.

Figure 2d shows the layout produced by t-FDP using the parameter settings recommended by the authors, $\alpha = 0.1$, $\beta = 8$, and $\gamma = 2$ (see Table 1). Compared to the FR model, which also lacks degree weighting, inter-class separation is more apparent (e.g., light blue, yellow, red classes). However, without explicit degree weighting, clusters remain loose and inter-cluster distances insufficient, leading to ambiguous inter-class topology and hindering tasks such as group adjacency analysis [38]. For example, it is unclear whether the upper-left light blue class is more strongly connected to the yellow or the pink class.

3.2 Degree-Weighted t-FDP Model

As degree-based weighting pulls low-degree nodes toward connected high-degree nodes and pushes apart unconnected high-degree nodes, it complements the clustering properties of the t-force model. We

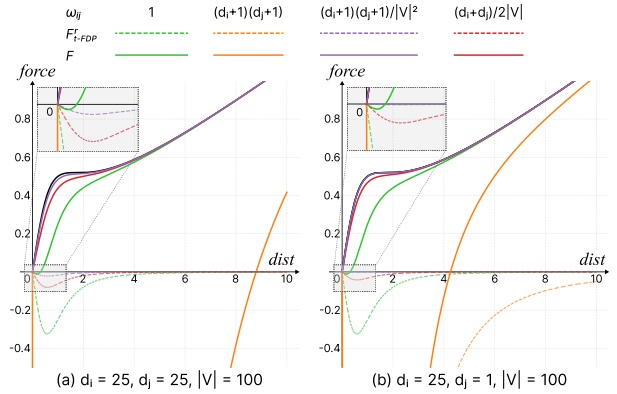


Fig. 3: Comparison of different degree-weighting schemes for (a) two high-degree nodes and (b) a high-degree node and a low-degree node. The attractive force $F^a_{t\text{-}FDP}$ is identical in all cases and is shown as a black solid line. Product-based weighting (orange) produces overly strong relative repulsion and its normalized variant (purple) nearly eliminates repulsion for low-degree nodes, while linearly normalized degree weighting (red) provides effective balance.

therefore combine them into a unified degree-weighted t-FDP model:

$$F(i) = \sum_{(i,j)\in E} F^a_{t\text{-}FDP}(i,j) + k \sum_{j\neq i} \omega_{ij} F^r_{t\text{-}FDP}(i,j), \qquad (4)$$

where ω_{ij} denotes a degree-dependent weighting factor applied to the repulsive force. A straightforward choice is the validated weighting scheme, $\omega_{ij} = (d_i + 1)(d_j + 1)$, used in FA2 [21]. However, as shown in Figure 2e, combining it with the t-force does not improve clustering; it over-expands clusters, blurring inter-class boundaries and adjacency.

This limitation stems from an imbalance in short-range forces. As shown in Figure 3, the unweighted t-FDP model (green solid line) has a very small equilibrium distance: below it, repulsion dominates and prevents overlap, whereas at group level, the same repulsion induces cluster expansion and looseness. Applying the product weighting $(d_i + 1)(d_j + 1)$ not only fails to alleviate this issue but amplifies it by greatly increasing the equilibrium distance. As shown by the solid orange curve, the inflated weight causes repulsion to dominate over a wider range, shifting the equilibrium point outward. Consequently, even adjacent nodes are overly separated, leading to problematic cluster expansion. This effect also affects low-degree pairs, where the weighting unnecessarily enlarges equilibrium distances, opposing the goal of

pulling highly connected nodes to form compact clusters.

To mitigate the force imbalance and cluster expansion caused by excessively large weight coefficients, we could introduce a normalization factor that constrains the magnitude of the repulsive force. Since the scale of $d_i \cdot d_j$ is comparable to $|V|^2$, a mathematically appealing modified weight would use this as a normalization constant:

$$\omega_{ij} = \frac{(d_i + 1)(d_j + 1)}{|V|^2}.$$

As shown by the purple curve in Figure 3, this normalization suppresses repulsive growth, allowing attraction to dominate at both short and long ranges; for connected pairs, the equilibrium distance approaches zero, yielding tighter clusters. However, the quadratic normalization overattenuates repulsion: for typical low-degree nodes, the normalized repulsive force becomes negligible across distances (purple dashed line in Figure 3b). Consequently, even unconnected nodes are drawn together via indirect interactions, causing clusters to collapse into dense singularities that obscure topology and internal structure.

Therefore, we adopt a linearly normalized degree weight to balance cluster compactness and structural readability:

$$\omega_{ij} = \frac{(d_i + d_j)}{2|V|}. \tag{5}$$

The red curve in Figure 3 shows that linear normalization maintains strong contraction, with the equilibrium near zero, pulling connected nodes together tightly to form compact clusters. Unlike quadratic normalization, it preserves a weak but non-zero repulsion (red dashed line), preventing cluster collapse into singularities and retaining internal structure. Relative to the unweighted t-FDP model, it also suppresses excessive repulsion at group level, allowing attraction from sparse inter-cluster edges to dominate; inter-cluster distances thus reflect connectivity, yielding clearer separation and more faithful group adjacency.

Figure 2f illustrates the layout with linearly normalized degree weights. It retains the low-overlap clustering of t-FDP while improving inter-community separation. Compared to Figure 2d and Figure 2e, previously loose clusters (e.g., the red class) become more compact, yielding sharper boundaries. Relative cluster positions align with the underlying topological connectivity: the light blue class lies closer to the pink than to the yellow class, consistent with 236 versus 132 inter-class edges. This indicates that linear normalization effectively maps inter-community connectivity to spatial proximity, producing layouts that better reflect group-level structure. Due to space limitations, we defer the additional ablation study and discussion of alternative degree-normalization schemes to the supplemental material.

3.3 Edge-Centric Negative Sampling

As indicated by Equation 1, spring-electrical models have per-epoch time complexity $O(|V|^2)$ due to the computation of repulsive forces between all vertex pairs. Acceleration methods such as BH [1] and RVS [17] approximate these forces via auxiliary structures; while reducing cost, they incur additional memory overhead and can degrade layout quality.

Inspired by UMAP [9], we introduce linear degree weighting (Equation 5) via an edge-centric negative sampling scheme: iterate over edges for attraction and draw k uniform negatives per edge for repulsion. For expectation analysis, let X_{ij} indicate whether $(i, j) \in E$, and let the random variable $Y_{ij,s}$ count how often node s is sampled as a negative when drawing k negative samples for (i, j). Over one epoch, the total force on node i decomposes into attractive and repulsive terms:

$$F^a_{SNAP}(i) = \sum_{j=1}^{|V|} \big(X_{ij} F^a(i, j) + X_{ji} F^a(j, i)\big),$$

$$F^r_{SNAP}(i) = \sum_{j=1}^{|V|} \Big(X_{ij} \sum_{s=1}^{|V|} Y_{ij,s} F^r(i, s) + \sum_{p=1}^{|V|} X_{jp} Y_{jp,i} F^r(j, i)\Big),$$

where F^a and F^r denote the forces in t-FDP. The edge-centric traversal leaves the sum of attractive forces unchanged. For repulsion, we consider $\mathbb{E}[F^r_{SNAP}(i)]$. Since each edge draws k negatives uniformly from the remaining $|V| - 1$ nodes, $\mathbb{E}[Y_{ij,s}] = \frac{k}{|V|-1}$ for any $s \neq i$.

Algorithm 1 Edge-Centric Negative Sampling Graph Layout

Input: Edge list E, initial positions $\mathbf{Y}$, number of epochs T, number of negative samples k, learning rate η
Output: Final node positions $\mathbf{Y}$

```
Randomly shuffle edge list E
for t = 1 to T do
    for edge (i, j) in E do
        y_i = y_i + η F^a_{t-FDP}(i, j)
        y_j = y_j − η F^a_{t-FDP}(i, j)
        for j = 1 to k do
            Sample y_s ∼ Uniform(Y \ {y_i})
            y_i = y_i + η F^r_{t-FDP}(i, s)
            y_s = y_s − η F^r_{t-FDP}(i, s)
        end for
    end for
    Update η
end for
```

By linearity of expectation, the expected repulsive force on node i decomposes into two terms, reflecting its roles during edge traversal:

1. The first term represents the repulsion node i experiences from negative samples s when processing its own outgoing edges (i, j). Taking the expected value allows us to substitute $Y_{ij,s}$ with $\frac{k}{|V|-1}$. Rearranging summation isolates $\sum_{j=1}^{|V|} X_{ij}$, which is exactly the degree d_i of node i, yielding $\sum_{s \neq i} d_i \frac{k}{|V|-1} F^r(i, s)$.
2. The second term represents the repulsion node i experiences when it is selected as a negative sample during the processing of another node j's outgoing edge (j, p). Again taking the expectation allows us to substitute $Y_{jp,i}$ with $\frac{k}{|V|-1}$. Rearranging the summation to isolate $\sum_{p=1}^{|V|} X_{jp}$ gives the degree d_j of the source node j. This yields an expected contribution of $\sum_{j \neq i} d_j \frac{k}{|V|-1} F^r(j, i)$.

To combine the two terms, we relabel the summation index s to j and use the symmetry of the repulsive force ($F^r(i, j) = F^r(j, i)$). This results in the following expected total repulsive force on node i in one full epoch:

$$\mathbb{E}[F^r_{SNAP}(i)] = \frac{k}{|V| - 1} \sum_{j \neq i} (d_i + d_j) F^r(i, j).$$

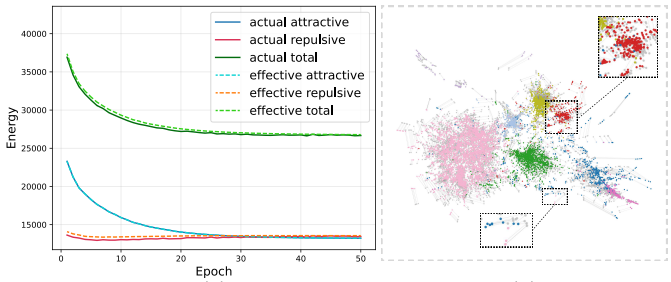
Combining it with the attractive force term and accounting for the double counting of undirected edges, the expected resultant force becomes

$$\mathbb{E}[F_{SNAP}(i)] = 2\Big(\sum_{(i,j) \in E} F^a(i, j) + k \sum_{j \neq i} \frac{d_i + d_j}{2(|V| - 1)} F^r(i, j)\Big). \tag{6}$$

The objective implicitly optimized by SNAP-tFDP is therefore equivalent to Equation 4. For large graphs where $|V| \gg 1$, the coefficient $\frac{d_i + d_j}{2(|V|-1)}$ closely matches the designed linear normalized weight from Equation 5. Thus, the proposed algorithm is an accurate approximation of the global degree-weighted objective.

Algorithm 1 outlines an implementation of our method. It requires only two data structures: the edge list and the current node positions. To ensure symmetry with node-centric models and satisfy the derivation above, each undirected edge is duplicated as two directed ones, (i, j) and (j, i), allowing for consistent force application at both endpoints. Learning rate η follows standard practice, controlling update magnitude and decaying over epochs. Despite its simplicity, SNAP-tFDP combines implicit weighting via negative sampling with stochastic gradient descent (SGD)-based optimization. Unlike traditional FDP methods that accumulate forces before updating, it applies immediate updates: for each directed edge, it computes attraction and updates both endpoints; for each negative sample, it computes repulsion and updates the source and sampled node. This avoids storing a resultant force array and introduces stochastic noise that aids escape from local minima and accelerates convergence [46].

To validate our theoretical derivation and implementation, we analyze the dataset used in Figure 2 by comparing actual and effective energy during optimization. The actual energy is accumulated from



(a) (b)

Fig. 4: Validation of the edge-centric negative sampling strategy. (a) Energy loss curves showing that the actual energy of SNAP-tFDP converges consistently with the effective energy. (b) Superimposition of layouts by SNAP-tFDP (colored) and the full computation (gray).

the negative sample pairs drawn in each epoch, whereas the effective energy is computed from the degree-weighted objective in Equation 4 using the global node positions after each epoch (our ground truth). As shown in Figure 4a, minor discrepancies appear in early epochs, but the two curves quickly converge. This confirms that the stochastic gradient estimation of SNAP-tFDP is asymptotically unbiased. To assess structural fidelity, we further compare the final SNAP-tFDP layout with the ground truth layout produced by full $O(|V|^2)$ computation. As shown in Figure 4b, while some peripheral low-degree nodes exhibit small deviations due to weaker constraints, the core cluster structures and group adjacency relations are well preserved. In particular, the light blue cluster is positioned closer to the pink cluster than to the yellow cluster, accurately reflecting their relative inter-cluster adjacency.

Complexity Analysis. SNAP-tFDP processes the edge list once per epoch. For each edge, it computes one attractive force in constant time and performs k negative sampling steps, each involving a constant-time repulsive force computation. Therefore, the time complexity per epoch is $O(|E|(1+k))$, which reduces to $O(|E|)$ since k is a small constant (typically $k \ll |V|$). In terms of space complexity, SNAP-tFDP stores only the edge list (with $2|E|$ directed edges for an undirected graph) and the node position matrix ($2|V|$ values for two-dimensional coordinates). The total space complexity is thus $O(|V|+|E|)$, which supports scalability to large graphs.

3.4 Lock-Free Parallelization

To efficiently process extremely large graphs, parallelization that fully exploits hardware resources is necessary. However, edges processed simultaneously by different threads may share endpoints. Concurrent updates to the same node position can therefore cause overwrites and conflicts, which may affect layout quality. Although lock-based synchronization [50] can prevent such conflicts, it introduces substantial overhead and may reduce the benefit of parallel execution. The HOGWILD! model proposed by Recht et al. [36] shows that lock-free parallel updates are effective when gradient updates are sparse, meaning that each update modifies only a small subset of parameters. In this section, we will first demonstrate that the SNAP-tFDP algorithm satisfies this sparsity condition. Building on this, we further design a parallel scheme based on computational bundles, aiming to reduce access conflicts without locks.

Following [36], sparsity is characterized by three statistics: the maximum number of variables involved in a single update (Ω), the maximum frequency with which a variable is accessed (Δ), and the maximum probability that two updates access the same variable (ρ). In Algorithm 1, the node positions $\mathbf{Y}$ define the optimization variables, and each inner-loop epoch—corresponding to the processing of a single edge (i,j) together with its k negative samples—constitutes one stochastic gradient update. Because the number of negative samples k is a small constant, each update modifies at most $k+2$ nodes, giving $\Omega = k+2 \ll |V|$, indicating strong sparsity in the parameter updates.

To estimate Δ and ρ, consider one epoch that processes all $|E|$ edges. Accesses to a node v arise from two cases: topological interactions and negative sampling. As an endpoint of edges, v participates in d_v inner-loop epochs. In addition, it may be selected as a negative sample; the expected number of such selections is $\frac{k|E|}{|V|-1}$. Let d_{max} denote the maximum degree in the graph. The proportion of updates involving the most frequently accessed node (Δ) and the upper bound on the probability of conflicts between two such nodes (ρ) can be approximated as

$$\Delta \approx \frac{d_{max} + \frac{k|E|}{|V|-1}}{|E|} = \frac{d_{max}}{|E|} + \frac{k}{|V|-1},$$

$$\rho \leq (k+2)\Delta = (k+2)\left(\frac{d_{max}}{|E|} + \frac{k}{|V|-1}\right).$$

In both expressions, the negative sampling term $\frac{k}{|V|-1}$ approaches zero as $|V|$ increases and has little impact on sparsity. The topological interaction term $\frac{d_{max}}{|E|}$ has the same order as the case analyzed in [36] for graph-cut problems, where Δ *is the maximum degree divided by* $|E|$ *and* ρ *is at most* 2Δ. Therefore, the update process satisfies the conditions of a sparse optimization problem.

In our implementation, we designed an improved parallel computation scheme to mitigate conflicts caused by a small number of very high-degree nodes in graphs with skewed degree distributions, such as scale-free networks [30]. Instead of processing edges independently, after an initial random shuffle, the algorithm groups all node pairs (i,j) sharing the same source node i into a computational bundle and assign the entire bundle to a single thread. This design assigns all updates of source node i handled by the same thread, matching the algorithm's asymmetric pattern: each stochastic step updates the source $(k+1)$ times, whereas destination and negative nodes are updated only once. Avoiding contention on these hot source nodes is thus critical, and bundling improves cache locality. Combined with sparsity, this scheme yields strong multi-threaded speedup with stable convergence. An empirical comparison of conflict rates of the original HOGWILD! scheme and our bundle-based scheme is provided in the supplemental material.

4 Evaluation

In this section, we evaluate the effectiveness and efficiency of SNAP-tFDP. Our implementation in C++ is warped into an open-source library[1]. All experiments are conducted on a high-performance workstation equipped with an AMD Ryzen Threadripper 3990X processor with 64 physical cores, 220 GB main memory, and NVIDIA GeForce RTX 3090 GPU (24 GB GPU memory) running Ubuntu 20.04 LTS.

4.1 Experiment Setup

Datasets. To provide a comprehensive evaluation, we collected 12 large-scale graphs with diverse data distributions and sizes, containing up to 4 M nodes and 117 M edges. These datasets were obtained from the Florida Sparse Matrix Collection [10], the SNAP network collection [25], and the benchmark sets used by tsNET [23] and DRGraph [49]. They span a wide range of graph properties, including varying average degrees, different numbers of node labels, and diverse application domains. As a result, the evaluation examines both runtime and visual quality across balanced graphs, graphs with skewed label distributions, and high-density networks. For SNAP community datasets (com-*) with too many labels, all nodes were used for performing layout algorithms, while only the top 5,000 communities by label quality were retained for visual quality metrics, following [44].

Methods. We compare SNAP-tFDP with eleven existing graph layout algorithms: two stress-based methods (PMDS [5] and Omega [31]), six spring-electrical methods (FR [15], SFDP [19], LinLog [29], FA2 [21], BatchLayout [34], and t-FDP [47]), and three dimensionality reduction-based methods (tsNET [23], DRGraph [49], and Force2Vec [35]). The implementations of PMDS, FR, and SFDP are obtained from OGDF [8] and GraphViz [13], whereas all other methods use the implementations provided by their respective authors. tsNET is accelerated using a GPU-based implementation [6], while t-FDP and BatchLayout are accelerated by the interpolation-based Fast Fourier Transform (ibFFT) [26] and the BH approximation, respectively. We also include the GPU implementation of FA2 from cuGraph [14].

[1] https://github.com/AnonymousUser202604/SNAP-tFDP

Table 2: Test datasets

Name	$\|V\|$	$\|E\|$	$\|E\|/\|V\|$	Labels
APH	7,487	119,043	15.90	8
aircraft	7,517	20,267	2.70	5
co_author	8,319	32,655	3.93	8
socfb-Yale4	8,561	405,440	47.36	36
ACO	13,381	245,778	18.37	10
socfb-UF21	35,111	1,465,654	41.74	34
soc-Flickr-ASU	80,513	5,899,882	73.28	195
com-dblp	317,080	1,049,866	3.31	13,477
com-amazon	334,863	925,872	2.76	75,149
com-youtube	1,134,890	2,987,624	2.63	8,385
com-orkut	3,072,441	117,185,083	38.14	6,288,363
com-lj	3,997,962	34,681,189	8.67	287,512

For fair comparison, we use the same PMDS layout as initialization for all methods except SFDP and DRGraph, which are initialized by a multi-level scheme. Each method is executed with the default parameters specified in the corresponding paper or implementation. Unless otherwise specified, the parallel version of all methods utilizes 16 threads. Since SFDP, DRGraph, and SNAP-tFDP involve stochastic operations, each experiment is repeated 5 times, and the reported results correspond to the average performance over these runs for each graph.

Metrics. As SNAP-tFDP aims to improve cluster visibility and intra-cluster neighborhood structures in large-scale graphs, we adopt three complementary cluster-level metrics. Metrics such as stress and edge crossings are not considered because they primarily evaluate distance preservation or edge readability rather than our target objectives.

- Neighborhood Preservation [40] (**NP**) measures how well the layout preserves local neighborhood structure: $NP = \frac{1}{|V|}\sum_{i=1}^{|V|}\frac{|N_G(i,r)\cap N_L(\mathbf{y}_i,k_i)|}{|N_G(i,r)\cup N_L(\mathbf{y}_i,k_i)|}$, where $N_G(i,r)$ denotes the set of r-ring neighbors of node i in the input graph G, $k_i = |N_G(i,r)|$, and $N_L(\mathbf{y}_i,k_i)$ denotes the set of k_i-nearest neighbors of $\mathbf{y}_i$ in the layout space. In our experiments, we set $r = 2$ to capture neighborhood structure at the cluster level, following DRGraph [49].
- Silhouette Index [37] (**SI**) measures the compactness and separation of clustering structures: $SI = \frac{1}{|V|}\sum_{i=1}^{|V|}\frac{b(i)-a(i)}{\max\{a(i),b(i)\}}$, where $a(i)$ is the average distance from node i to other nodes in the same cluster, and $b(i)$ is the average distance from node i to nodes in the nearest neighboring cluster.
- Clustering Quality [28] (**CQ**) evaluates how well node positions align with the ground-truth clustering: $CQ = ARI(L, \text{KMeans}(\mathbf{Y}))$, where L denotes the ground-truth labels, $\mathbf{Y}$ represents the node positions in the layout, and ARI is the Adjusted Rand Index [20].

For all the above metrics, higher values indicate better layout quality. In addition to the visual quality metrics, we also measure the *runtime* and *peak memory usage* to evaluate computational performance.

4.2 Selection of Parameters

To improve reproducibility and reduce parameter tuning, SNAP-tFDP inherits the t-FDP parameters (α, β, and γ) recommended in [47]. Consequently, only two parameters require selection: k and T.

The number of negative samples k. This parameter controls both the computational cost of repulsive forces and their relative strength. A small k leads to lower runtime but may cause excessive contraction, distorting intra-cluster structures. Increasing k helps to reveal intra-cluster details; however, this improvement quickly saturates while the runtime increases proportionally. As shown in Figure 5a, the green cluster in the lower-left corner is overly contracted at $k = 1$, but becomes clearly separated at $k = 3$, with only marginal changes for larger k. Figure 5b reports the three quality metrics, with individual dataset scores shown in gray and their mean values shown in red. These metrics may increase or decrease with k, but overall show limited variation. Based on the observed trade-off between visual quality and runtime, $k = 3$ is selected as the default setting for all experiments.

The number of epochs T. Parameter T has a strong impact on layout quality at small values, but the improvement saturates around $T = 50$, as shown in Figure 6a. This trend is also confirmed visually in Figure 6b, where structural changes become very small after epoch 50. In addition, the serial SNAP-tFDP algorithm and its two lock-free parallel variants share similar trajectories, and the differences are smaller than the variability introduced by negative sampling. The complete results for all metrics and graph scales (small, medium, and large) are provided in the supplemental material. Considering the diminishing marginal returns, we set $T = 50$ as the default.

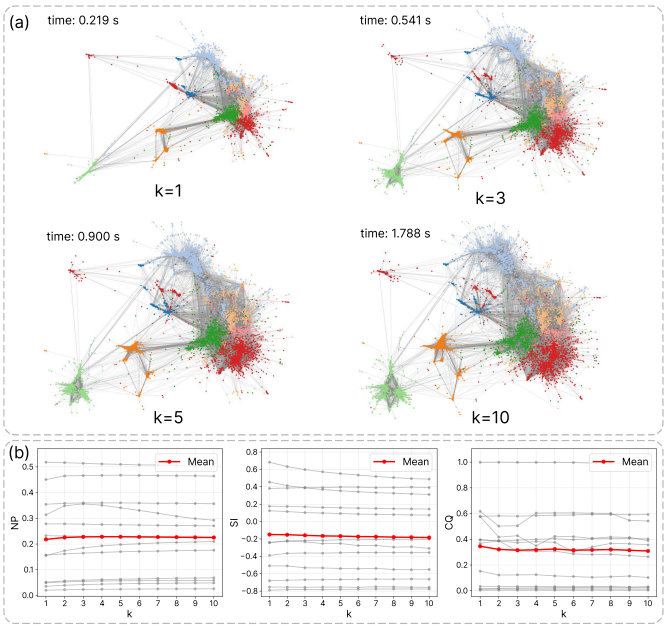


Fig. 5: Effect of the number of negative samples k. (a) Layout results and corresponding runtime of the serial SNAP-tFDP algorithm for different k on the *APH* dataset. (b) **NP**, **SI**, and **CQ** scores under varying k. Results for individual datasets are shown in gray, with the average across datasets highlighted in red.

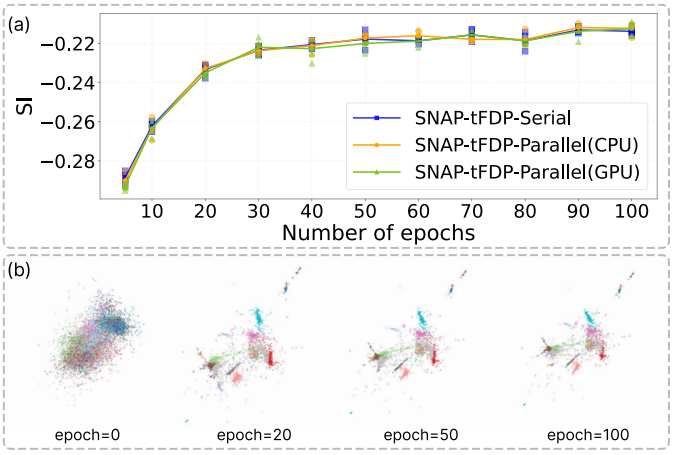


Fig. 6: (a) Convergence of the silhouette index (**SI**) for the serial SNAP-tFDP algorithm and its two lock-free parallel variants on the largest *com-lj* graph, where semi-transparent points indicate the results of five independent runs. (b) Layouts at four different epochs; for clarity, only the top 20 largest communities are shown.

4.3 Visual Quality

The heatmap in Figure 7 presents the **NP**, **SI**, and **CQ** scores of layouts generated by eleven baselines and the serial SNAP-tFDP. The full results, including parallel variants, are provided in the supplemental material. As the absolute values of these metrics are strongly influenced by dataset characteristics, the last row reports the mean performance computed only over methods that successfully produce layouts for all datasets. Under this setting, SNAP-tFDP achieves the best performance across all three metrics.

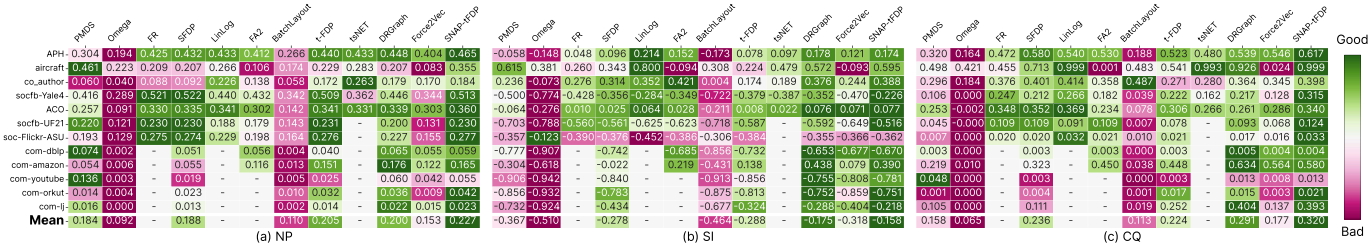


Fig. 7: Heatmaps employing a pink-to-green colormap illustrate the scores of **NP** (a), **SI** (b), and **CQ** (c) for layouts generated by nine methods across all datasets. Empty cells indicate that the graph was too large to be processed by the corresponding method. Each row corresponds to a dataset, and each column to a layout method. Colors are scaled row-wise based on the best and worst within each dataset.

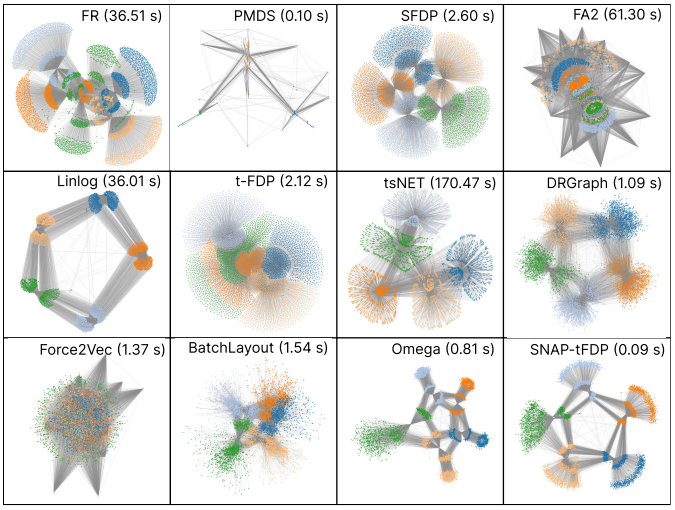


Fig. 8: Layouts and corresponding runtimes of twelve serial methods for the *aircraft* dataset. SNAP-tFDP (lower right) combines degree weighting with t-forces, yielding clear cluster structures with the lowest runtime, as fast as pure Pivot MDS.

In terms of **NP**, SNAP-tFDP achieves five first-place and three second-place rankings across the 12 datasets. Omega exhibits the lowest average score and ranks last on eleven datasets, as its low-rank resistance distance embedding discards necessary high-frequency components to differentiate densely connected local clusters. PMDS also underperforms representative methods from other categories, such as SFDP and DRGraph, although it performs relatively well on datasets where many nodes share similar topological relationships (e.g., *aircraft*). Among spring-electrical methods, BatchLayout performs the worst due to the BH approximation, consistent with the results reported in the original paper. The degree-weighted LinLog and FA2 achieve only moderate performance. FR and SFDP perform well on datasets with fewer than 100,000 nodes; however, FR cannot scale to larger datasets, while SFDP degrades substantially, indicating that power-function-based forces are less effective at preserving neighborhood structures. Overall, t-FDP outperforms other spring-electrical methods and ranks second in average performance, but falls behind SNAP-tFDP on 11 datasets. We believe this is due to the better contraction of clusters in SNAP-tFDP. Among dimensionality reduction-based methods, Force2Vec performs the worst, as it is designed for high-dimensional embedding rather than 2D graph layout. For the datasets on which tsNET can be applied, its average score is lower than that of SNAP-tFDP, while DRGraph is inferior to SNAP-tFDP on 9 of the 12 datasets.

Regarding **SI** and **CQ**, Omega and PMDS again rank among the worst-performing methods, suggesting stress-based methods are less effective at producing well-separated clusters. BatchLayout also performs poorly, followed by Force2Vec, for reasons similar to those discussed for **NP**. t-FDP and tsNET achieve better results but still lag behind the top-performing methods, as they tend to produce relatively loose clusters with less distinct boundaries (see Figure 8). FR and SFDP obtain moderate scores but are consistently inferior to the degree-weighted FA2 and LinLog, indicating that strong attractive forces reduce inter-cluster distances and make cluster separation difficult. DRGraph and

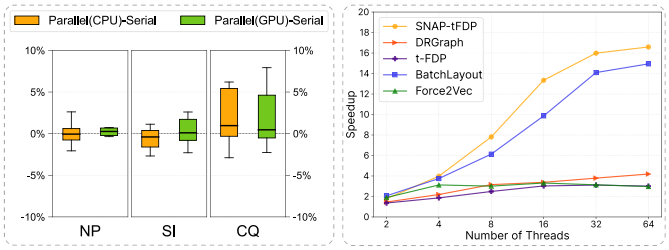


Fig. 9: (a) Differences in visual quality scores between the two parallel variants and the serial SNAP-tFDP across all datasets. (b) Speedup comparison of SNAP-tFDP and the parallel implementations of existing methods as a function of the number of CPU threads on the largest *com-lj* dataset.

SNAP-tFDP achieve the best performance, as both adopt negative sampling, which can be interpreted as normalized degree-weighted repulsion (Section 3.3), leading to more compact clusters and clearer grouping. In particular, on the **CQ** metric, the ordering DRGraph < LinLog < SNAP-tFDP indicates that SNAP-tFDP combines the cluster separation effect of degree weighting with the overlap reduction provided by t-force.

Figure 8 shows visual results of applying these methods to the *aircraft* dataset. Although Linlog and PMDS achieve higher scores in the **NP** and **SI** metrics, SNAP-tFDP exhibits clearer visual features. Nodes with low degrees are pushed to the periphery, forming loose clusters, while high-degree nodes clearly form cohesive small clusters and further establish a clear two-layer pentagonal inter-cluster connection structure. More visualizations of the corresponding layouts are provided in the supplemental material.

Furthermore, both parallel variants (CPU and GPU) show strong consistency with the serial SNAP-tFDP, as summarized in Figure 9a. The variations of **NP** and **SI** lie within [-5%, 5%], while most **CQ** values also fall within this range, with some cases even improving the median **CQ**. This shows that for large graphs, the proposed lock-free parallelization yields high-quality layouts similar to the serial solution.

4.4 Performance

Runtime. Figure 10 illustrates how runtime scales with the number of edges. For clarity, we show only the methods applicable to all datasets; omitted methods are substantially slower than our method. The complete results are available in the supplemental material. Among single-threaded CPU implementations, the serial SNAP-tFDP shows near-linear scaling and is the fastest among the eight methods, as indicated by the least-squares fitted line. PMDS is the second fastest, but is on average 150% slower than SNAP-tFDP across datasets, measured as the mean relative slowdown. For datasets with a very high $|E|/|V|$ ratio (e.g., *com-orkut*, the dataset with the highest number of edges), the ibFFT implementation of t-FDP gains a slight advantage due to its $O(|V|)$ time complexity, whereas SNAP-tFDP scales as $O(|E|)$. Nevertheless, t-FDP remains about 500% slower on average. Although Omega has the same $O(|E|)$ complexity as SNAP-tFDP, it is 1449% slower in practice. Force2Vec, DRGraph, SFDP, and BatchLayout are even slower, with average slowdowns of 841%, 911%, 2300%, and 3292%, respectively. For GPU implementations, SNAP-tFDP reduces the average slowdown by 39% relative to FA2 and by 366% relative to

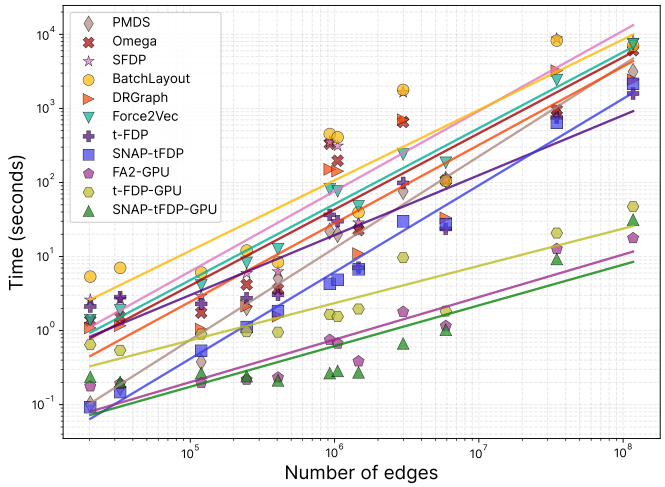


Fig. 10: Runtime of eight layout methods, together with the GPU implementations of FA2, t-FDP, and SNAP-tFDP, across all datasets.

Table 3: Comparison of memory consumption (MB). The last three columns report GPU memory. The final row shows the mean relative reduction (MRR), computed as the average of per-dataset ratios $(x - x_{\text{SNAP-tFDP}})/x$.

Dataset	PMDS	Omega	SFDP	BatchLayout	DRGraph	Force2Vec	t-FDP	Ours	FA2-GPU	t-FDP-GPU	Ours
APH	60	139	109	17	21	13	702	**7**	404	421	**261**
aircraft	20	68	48	11	13	7	695	**5**	398	421	**261**
co_author	25	80	55	11	15	8	747	**6**	398	445	**261**
socfb-Yale4	203	293	304	44	60	30	719	**11**	422	425	**269**
ACO	120	217	206	30	40	21	716	**9**	410	423	**265**
socfb-UF21	751	931	1,070	127	217	96	780	**28**	482	553	**285**
soc-Flickr-ASU	2,990	3,477	4,140	549	858	359	1,029	**97**	966	705	**353**
com-dblp	723	2,164	1,469	149	335	77	1,050	**31**	488	1,985	**281**
com-amazon	701	2,125	1,439	145	322	70	1,139	**30**	474	1,983	**279**
com-youtube	2,333	7,674	4,914	436	1,134	201	1,561	**89**	932	4,015	**315**
com-orkut	55,630	74,991	85,047	11,245	18,393	7,197	10,146	**1,898**	11,762	9,451	**2,073**
com-lj	20,358	30,737	33,267	4,119	7,391	2,175	5,487	**671**	5,500	8,905	**821**
MRR (vs. Ours)	91.58%	96.45%	97.82%	72.14%	81.19%	72.31%	94.94%	–	49.78%	60.23%	–

t-FDP. Due to additional preprocessing overhead, SNAP-tFDP-GPU is slower than the serial SNAP-tFDP on datasets with fewer than 100k edges; however, its runtime remains below 300 ms in these cases.

Figure 9b plots speedup versus thread count for CPU-parallel methods. Owing to its simple parallel design, SNAP-tFDP achieves the highest speedup among all methods. While BatchLayout scales near-linearly up to 16 threads, SNAP-tFDP achieves higher speedups throughout the entire range, reaching 16.0× at 64 threads versus 14.9× for BatchLayout. In contrast, DRGraph, t-FDP, and Force2Vec saturate much earlier, reaching only 4.2×, 3.0×, and 3.3×, respectively, indicating limited scalability beyond a small number of threads.

Memory Consumption. A key advantage of SNAP-tFDP is memory efficiency (Table 3). Compared with the three strongest baselines—Force2Vec, BatchLayout, and DRGraph—it reduces memory consumption by 72%, 72%, and 81% on average, respectively. t-FDP incurs higher overhead on small datasets due to its screen-dependent interpolation grid; although it is the most efficient among the remaining methods on *com-orkut*, SNAP-tFDP still achieves a 95% average reduction. In contrast, PMDS, Omega, and SFDP require 54 GB, 73 GB, and 83 GB on *com-orkut*, exceeding typical hardware limits.

For GPU memory usage, SNAP-tFDP also achieves the smallest memory footprint, as it requires no auxiliary data structures. In contrast, FA2 and t-FDP rely on a quadtree and an interpolation grid, respectively, resulting in consistently higher memory consumption across all datasets. Especially, on *com-lj*, SNAP-tFDP uses 0.82 GB versus 8.70 GB for t-FDP (≈10x less); on *com-orkut*, 2 GB versus 9.20 GB (≈4.5x less). The smaller ratio on the latter dataset is partly due to additional host-device transfers introduced by *cupy*, which also explains why SNAP-tFDP-GPU is faster despite slower serial performance. This efficiency enables processing large graphs on consumer-grade GPUs without exhausting memory.

5 Case Study

To demonstrate the capability of SNAP-tFDP for ultra-large-scale graphs, we conduct a case study on the *com-friendster* dataset [25],

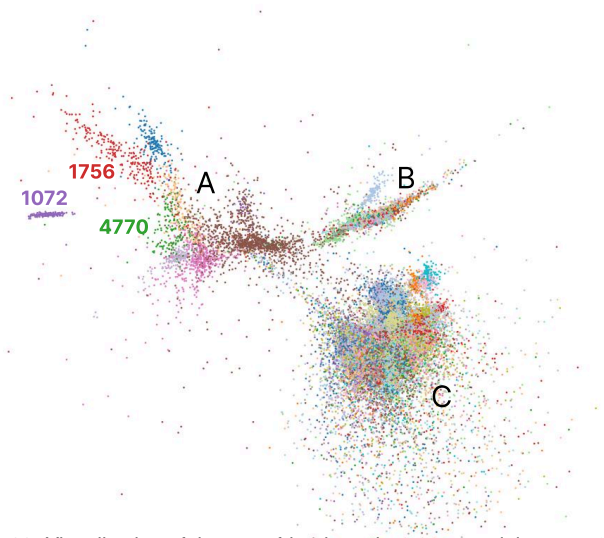


Fig. 11: Visualization of the *com-friendster* dataset, containing over 65.6 million nodes and 1.8 billion edges.

which contains over 65.6 million nodes and 1.8 billion edges. The raw data alone occupies 30 GB, which is 15 times larger than the *com-orkut* dataset. At this scale, most existing graph layout algorithms become infeasible due to time and memory limitations. In contrast, using default settings with a 16-thread CPU parallel implementation, SNAP-tFDP generates the layout in 4287.3 seconds (∼1.2 hours) with a peak memory usage of only 29.48 GB. This result shows that visualization of such large graphs is feasible on a standard single-node workstation.

To evaluate layout quality and structural fidelity, we isolated and visualized the top 200 highest-quality communities (Figure 11). Edges were omitted for clarity, and colors indicate community labels. The resulting layout reveals three structural observations about the Friendster network: First, the 200 communities self-organize into three distinct macro-clusters (labeled A, B, and C), suggesting that individual communities aggregate into higher-level social structures. Second, the separation within these macro-clusters reflects the density of inter-community connections. Macro-cluster A shows clear boundaries between communities, whereas B and C appear more entangled and overlapping. This pattern is consistent with the underlying topology: the average numbers of inter-community edges for macro-clusters A, B, and C are 2,950, 5,076, and 6,820, respectively. Finally, communities with fewer inter-community connections tend to be positioned more independently. For example, in macro-cluster A, the green, red, and purple communities extend progressively further away from the dense center. This spatial arrangement aligns with their decreasing inter-community edge counts of 4,770, 1,760, and 1,080, respectively.

6 Conclusion

We presented an efficient graph layout algorithm that achieves clear cluster separation. A linearly normalized degree-weighting scheme was first introduced for t-FDP, together with an analysis of its effect on cluster separation. Based on this formulation, we propose an edge-centric negative sampling strategy to implicitly realize the weighting with $O(|E|)$ time complexity and $O(|V| + |E|)$ space complexity. In addition, a lock-free, bundle-based parallelization scheme was developed to support efficient execution. Quantitative evaluations on layout quality and efficiency demonstrated the advantages of our method, and a case study further showed its applicability to ultra-large-scale graphs.

Several directions remain for future work. First, it is of interest to study whether the proposed framework remains applicable to alternative force models beyond the t-distribution, such as Gaussian-based forces [4]. Second, uniform negative sampling is not well suited to layout objectives such as preserving uniform edge lengths or improving the visibility of low-degree leaf nodes. More flexible weighting and negative sampling schemes may help address these objectives. Finally, integrating the method into open-source frameworks such as AutoFDP [42] is a promising direction.

Acknowledgments

The authors like to thank the anonymous reviewers for their valuable input. This work is supported by the grants of the NSFC (No.62402284, No.U2436209), the Beijing Natural Science Foundation (L247027), NSF of Shandong province (ZR2024QF212), the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University of China.

References

[1] J. Barnes and P. Hut. A hierarchical o (n log n) force-calculation algorithm. *Nature*, 324(6096):446–449, 1986. doi: 10.1038/324446a0 2, 3, 5

[2] J. Batson, D. A. Spielman, N. Srivastava, and S.-H. Teng. Spectral sparsification of graphs: theory and algorithms. *Communications of the ACM*, 56(8):87–94, 2013. doi: 10.1145/2492007.2492029 3

[3] J. N. Böhm, P. Berens, and D. Kobak. Attraction-repulsion spectrum in neighbor embeddings. *Journal of Machine Learning Research*, 23(95):1–32, 2022. 2

[4] C. Both, N. Dehmamy, R. Yu, and A.-L. Barabási. Accelerating network layouts using graph neural networks. *Nature Communications*, 14(1):1560, Mar. 2023. doi: 10.1038/s41467-023-37189-2 9

[5] U. Brandes and C. Pich. Eigensolver methods for progressive multidimensional scaling of large data. In *Proceedings of the 14th International Symposium on Graph Drawing*, pp. 42–53. Springer, 2006. doi: 10.1007/978-3-540-70904-6_6 1, 6

[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 *Proceedings of the 30th International Symposium on Computer Architecture and High Performance Computing*, pp. 330–338. IEEE, Campinas, Brazil, 2018. doi: 10.1109/CAHPC.2018.8645912 6

[7] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pp. 1597–1607. PMLR, 2020. 3

[8] M. Chimani, C. Gutwenger, M. Jünger, G. W. Klau, K. Klein, and P. Mutzel. The open graph drawing framework (OGDF). In R. Tamassia, ed., *Handbook of Graph Drawing and Visualization*, pp. 543–569. Chapman and Hall/CRC, Boca Raton, FL, 2013. 6

[9] S. Damrich and F. A. Hamprecht. On umap's true loss function. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, eds., *Advances in Neural Information Processing Systems*, vol. 34, pp. 5798–5809. Curran Associates, Inc., 2021. 5

[10] T. A. Davis and Y. Hu. The university of florida sparse matrix collection. *ACM Transactions on Mathematical Software*, 38(1), art. no. 1, 25 pages, Dec. 2011. doi: 10.1145/2049662.2049663 6

[11] S. Di Bartolomeo, T. Crnovrsanin, D. Saffo, E. Puerta, C. Wilson, and C. Dunne. Evaluating graph layout algorithms: A systematic review of methods and best practices. *Computer Graphics Forum*, 43(6):e15073, 2024. doi: 10.1111/cgf.15073 1

[12] P. Eades. A heuristic for graph drawing. *Congressus numerantium*, 42(11):149–160, 1984. 1

[13] J. Ellson, E. Gansner, L. Koutsofios, S. C. North, and G. Woodhull. Graphviz–open source graph drawing tools. In *Proceedings of the 9th International Symposium on Graph Drawing*, pp. 483–484. Springer, 2001. 6

[14] A. Fender, B. Rees, and J. Eaton. Rapids cugraph. In *Massive Graph Analytics*, pp. 483–493. Chapman and Hall/CRC, 2022. 6

[15] T. M. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software: Practice and experience*, 21(11):1129–1164, 1991. doi: 10.1002/spe.4380211102 1, 2, 3, 4, 6

[16] R. Gove. Force-directed graph layouts by edge sampling. In *Proceedings of the IEEE Symposium on Large Data Analysis and Visualization (LDAV)*, pp. 1–5. IEEE, 2019. doi: 10.1109/LDAV48142.2019.8944364 3

[17] R. Gove. A random sampling $o(n)$ force-calculation algorithm for graph layouts. *Computer Graphics Forum*, 38(3):739–751, 2019. doi: 10.1111/cgf.13724 3, 5

[18] D. Hangan, S. Kobourov, and J. Miller. Bridging graph drawing and dimensionality reduction with stochastic stress optimization. *arXiv preprint arXiv:2605.00641*, 2026. 2

[19] Y. F. Hu. Efficient and high quality force-directed graph drawing. *The Mathematica Journal*, 10:37–71, 2005. 3, 6

[20] L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985. doi: 10.1007/BF01908075 7

[21] M. Jacomy, T. Venturini, S. Heymann, and M. Bastian. Forceatlas2, a continuous graph layout algorithm for handy network visualization designed for the gephi software. *PloS one*, 9(6):e98679, 2014. doi: 10.1371/journal.pone.0098679 1, 2, 3, 4, 6

[22] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31(1):7–15, 1989. doi: 10.1016/0020-0190(89)90102-6 1, 2

[23] J. F. Kruiger, P. E. 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. doi: 10.1111/cgf.13187 2, 4, 6

[24] B. Lee, C. Plaisant, C. S. Parr, J.-D. Fekete, and N. Henry. Task taxonomy for graph visualization. In *Proceedings of the AVI Workshop on BEyond Time and Errors: Novel Evaluation Methods for Information Visualization (BELIV)*, pp. 1–5, 2006. doi: 10.1145/1168149.1168168 1

[25] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014. 6, 9

[26] G. C. Linderman, M. Rachh, J. G. Hoskins, S. Steinerberger, and Y. Kluger. Fast interpolation-based t-sne for improved visualization of single-cell rna-seq data. *Nature Methods*, 16(3):243–245, 2019. doi: 10.1038/s41592-018-0308-4 6

[27] L. McInnes, J. Healy, and J. Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2020. doi: 10.48550/arXiv.1802.03426 2, 3

[28] A. Meidiana, S.-H. Hong, P. Eades, and D. Keim. A quality metric for visualization of clusters in graphs. In *Proceedings of the 27th International Symposium on Graph Drawing and Network Visualization (GD)*, pp. 125–138. Springer, 2019. doi: 10.1007/978-3-030-35802-0_10 1, 7

[29] A. Noack. Energy models for graph clustering. *Journal of Graph Algorithms and Applications*, 11(2):453–480, 2007. doi: 10.7155/jgaa.00154 1, 2, 3, 4, 6

[30] J.-P. Onnela, J. Saramäki, J. Hyvönen, G. Szabó, D. Lazer, K. Kaski et al. Structure and tie strengths in mobile communication networks. *Proceedings of the National Academy of Sciences*, 104(18):7332–7336, 2007. doi: 10.1073/pnas.0610245104 6

[31] Y. Onoue. Graph drawing stress model with resistance distances. *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–10, 2026. doi: 10.1109/TVCG.2026.3694437 2, 6

[32] A. v. d. Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. doi: 10.48550/arXiv.1807.03748 3

[33] F. V. Paulovich, A. Arleo, and S. van den Elzen. When dimensionality reduction meets graph (drawing) theory: Introducing a common framework, challenges and opportunities. *Computer Graphics Forum*, 44(3):e70105, 2025. doi: 10.1111/cgf.70105 2

[34] M. K. Rahman, M. H. Sujon, and A. Azad. Batchlayout: A batch-parallel force-directed graph layout algorithm in shared memory. In *Proceedings of the 2020 IEEE Pacific Visualization Symposium (PacificVis)*, pp. 16–25. IEEE, 2020. doi: 10.1109/PacificVis48177.2020.3756 3, 6

[35] M. K. Rahman, M. H. Sujon, and A. Azad. Force2vec: Parallel force-directed graph embedding. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pp. 442–451. IEEE, 2020. doi: 10.1109/ICDM50108.2020.00053 2, 3, 6

[36] B. Recht, C. Re, S. Wright, and F. Niu. HOGWILD!: A lock-free approach to parallelizing stochastic gradient descent. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, eds., *Advances in Neural Information Processing Systems*, vol. 24. Curran Associates, Inc., 2011. 2, 3, 6

[37] P. J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987. doi: 10.1016/0377-0427(87)90125-7 2, 7

[38] B. Saket, P. Simonetto, and S. Kobourov. Group-level graph visualization taxonomy, 2014. doi: 10.48550/arXiv.1403.7421 4

[39] L. Van Der Maaten. Accelerating t-sne using tree-based algorithms. *The journal of machine learning research*, 15(1):3221–3245, Jan. 2014. 2

[40] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. 7

[41] C. Walshaw. A multilevel algorithm for force-directed graph drawing. In *Proceedings of the 8th International Symposium on Graph Drawing*, pp. 171–182. Springer, 2000. doi: 10.1007/3-540-44541-2_17 3

[42] M. Xue, Y. Wang, Z. Wang, L. Zhu, L. Cui, Y. Chen et al. AutoFDP: Automatic force-based model selection for multicriteria graph drawing. *IEEE Transactions on Visualization and Computer Graphics*, 32(2):1554–1568, 2026. doi: 10.1109/TVCG.2025.3631659 9

[43] M. Xue, Z. Wang, F. Zhong, Y. Wang, M. Xu, O. Deussen et al. Taurus: Towards a unified force representation and universal solver for graph layout. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):886–895, 2023. doi: 10.1109/TVCG.2022.3209371 2

[44] J. Yang and J. Leskovec. Defining and evaluating network communities based on ground-truth. In *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*, MDS '12, art. no. 3, 8 pages. Association for Computing Machinery, New York, NY, USA, 2012. doi: 10.1145/2350190.2350193 1, 6

[45] H. D. Young, R. A. Freedman, T. Sandin, and A. L. Ford. *University Physics*, vol. 9. Addison-Wesley Reading, MA, 1996. 3

[46] J. X. Zheng, S. Pawar, and D. F. M. Goodman. Graph drawing by stochastic gradient descent. *IEEE Transactions on Visualization & Computer Graphics*, 25(09):2738–2748, Sept. 2019. doi: 10.1109/TVCG.2018.2859997 5

[47] F. Zhong, M. Xue, J. Zhang, F. Zhang, R. Ban, O. Deussen et al. Force-directed graph layouts revisited: a new force based on the t-distribution. *IEEE Transactions on Visualization and Computer Graphics*, 30(7):3650–3663, 2023. doi: 10.1109/TVCG.2023.3238821 1, 2, 3, 4, 6, 7

[48] F. Zhou, S. Malher, and H. Toivonen. Network simplification with minimal loss of connectivity. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pp. 659–668. IEEE, 2010. doi: 10.1109/ICDM.2010.133 3

[49] M. Zhu, W. Chen, Y. Hu, Y. Hou, L. Liu, and K. Zhang. DRGraph: An efficient graph layout algorithm for large-scale graphs by dimensionality reduction. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1666–1676, 2020. doi: 10.1109/TVCG.2020.3030447 1, 2, 3, 4, 6, 7

[50] M. Zinkevich, M. Weimer, L. Li, and A. Smola. Parallelized stochastic gradient descent. *Advances in Neural Information Processing Systems*, 23, 2010. 6