

Force-directed graph layouts revisited: a new force based on the t-Distribution

Fahai Zhong, Mingliang Xue, Jian Zhang, Fan Zhang, Rui Ban, Oliver Deussen and Yunhai Wang

Abstract—In this paper, we propose the t-FDP model, a force-directed placement method based on a novel bounded short-range force (t-force) defined by Student's t-distribution. Our formulation is flexible, exerts limited repulsive forces for nearby nodes and can be adapted separately in its short- and long-range effects. Using such forces in force-directed graph layouts yields better neighborhood preservation than current methods, while maintaining low stress errors. Our efficient implementation using a Fast Fourier Transform is one order of magnitude faster than state-of-the-art methods and two orders faster on the GPU, enabling us to perform parameter tuning by globally and locally adjusting the t-force in real-time for complex graphs. We demonstrate the quality of our approach by numerical evaluation against state-of-the-art approaches and extensions for interactive exploration.

Index Terms—Graph Layout, Force Directed Placement, Student's t-Distribution, FFT

1 INTRODUCTION

GRAPHS are a ubiquitous form of data, whenever objects and their relations have to be represented; lots of effort has been spent on finding good ways to visualize them. Arguably one of the most prevalent representations for graphs are node-link diagrams, which are intuitive and efficient in supporting various graph analysis tasks [3].

Despite the existence of many other techniques, force-directed placement (FDP) methods [4] have gained a lot of attention for automatically layouting node-link diagrams. Here, a graph is modeled as a physical system of bodies with forces acting between them, the layout is computed by minimizing the overall energy of the system. While some variants of forces have been proposed [4], the spring-electric model [1], [5] is the most popular method. It assigns spring-like attractive forces between connected nodes and repulsive electrical forces between all node pairs to keep them at a distance. For most graphs, the method creates layouts with compelling performance [6], especially when applying its multi-level version, the scalable force-directed placement (SFDP) [7]. Since it is intuitive and easy to implement, the method is a key component in many visualization systems (e.g. Gephi [8] and D3 [9]).

A typical problem of spring-electric models is that large repulsive electric forces are exerted on each nearby pair of connected nodes, but only small attractive spring forces. In a physical simulation this is reasonable since repelling contact forces for real bodies should be extremely

large. However, such forces might destroy graph structures by preventing connected nodes in the graph from being neighbors in the layout (see the bottom statistics in Fig. 1(a)). Although recently proposed graph layout methods such as tsNET [2] allow to better preserve 2-ring neighborhoods, they fail in efficiently maintaining low stress errors and 1-ring neighborhoods (see Fig. 1(b)). DRGraph [10] improves the scalability of tsNET but has a similar issue with yielding large stress errors.

In this paper, we revisit the spring-electric model and propose a new heavily-tailed force based on the t-distribution (referred to as t-force), for better maintaining local graph structures while keeping the intuitiveness and simplicity of the model. Deviating from traditional spring-electric models gives us the flexibility to freely explore the interplay between attractive and repulsive forces. This is achieved by dividing forces into their short- and long-range components and improving their short-range aspects. The t-force has an upper bound at short-range and behaves similar to the electrical repulsive force at long-ranges. The attractive force at short-range is kept unaltered. By using t-forces to define the repulsive force within FDP and combining t-forces with spring forces as the attractive forces, we are able to compose a new t-force-based FDP model (*t*-FDP for short), which allows us at the same time better preserving graph neighborhoods, distances of nodes and clusters (see Fig. 1(c)).

Being one of two aspects of the t-SNE gradient, the t-force is similar to the repulsive force of the t-SNE model [11], a widely used method for dimensionality reduction. This allows us to employ an interpolation-based acceleration strategy for t-SNE using the fast Fourier transformation [12] which can be implemented on the GPU. Due to the parallel nature of the FFT, our whole model can be efficiently implemented on GPUs and this allows us to perform interactive parameter optimization. To enable a wide range of users to use our method, we provide an implementation as a drop-in force for the “d3-force” library.

We evaluated our approach using 50 graphs with node numbers ranging from 72 to 4 million and edge numbers ranging from 75 to 100 million. We quantitatively measured

- Fahai Zhong, Mingliang Xue, Yunhai Wang are with the Department of Computer Science, Shandong University, China. E-mail: {zhongfahai, xml95007, cloudseawang}@gmail.com
- Jian Zhang is with Computer Network Information Center, Chinese Academy of Sciences, Beijing, China. E-mail: zhangjian@sccas.cn
- Fan Zhang is with School of Computer Science and Technology, SDTU, China. E-mail: zhangfan@sdtu.edu.cn
- Rui Ban is with Intelligent Network Design Institute, CITC, Beijing, China. E-mail: banrui1@chinaunicom.cn
- Oliver Deussen is with Computer and Information Science, University of Konstanz, Konstanz, Germany. E-mail: oliver.deussen@uni-konstanz.de
- Yunhai Wang is the corresponding author

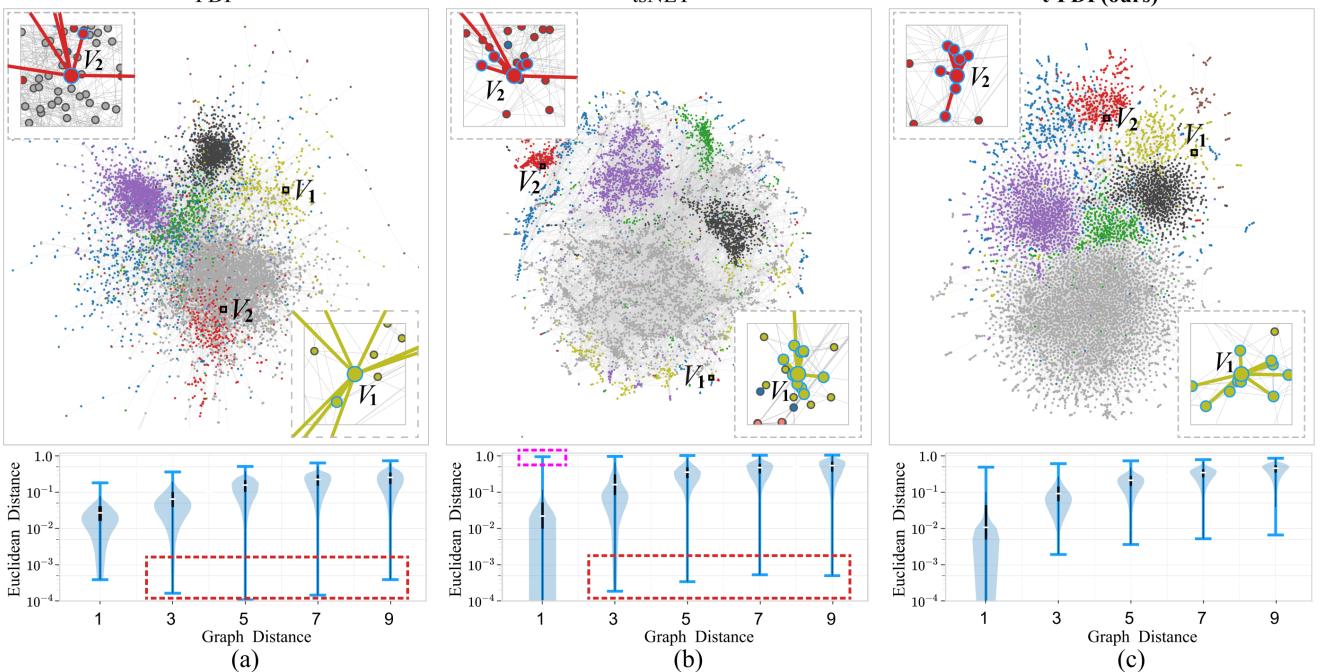


Fig. 1. Comparison of our technique (t-FDP) with standard force-directed placement (FDP) [1] and tsNET [2], a state-of-the-art neighborhood embedding-based approach on a labeled co-author network by summarizing the normalized Euclidean distances of graph nodes in layout space with violin plots shown in the bottom of (a,b,c). While FDP places nodes close by that have large graph distances (the red box on the violin plots in a), tsNET avoids this to some extent (the red box on the violin plots in b) but often keeps nodes with small graph distances far from each other (the pink box on the violin plots in b). t-FDP in (c) is able to reach both, short layout distances for nearby points in data space and large ones for distant data points. Therefore it separates all eight classes clearly, while the other methods mix them in part. Two example neighborhoods show this: in (a) the neighbours of V_1 do not have connections to V_1 and the graph neighbours of V_2 are mostly far away. tsNET is better in keeping the local neighborhood of V_1 , but the neighborhood of V_2 has large graph distances. Both cases have a better neighborhood preservation with t-FDP.

the quality of our results using various structural and readability metrics. For the tested data, our method performs similarly or even outperforms all existing methods in all metrics except the stress error, where our method is the second-best behind the stress model. Regarding performance, the CPU version of t-FDP is one order of magnitude faster than SFDP and DRGraph on a desktop computer with Intel i7-8700 CPU, while reducing the memory size considerably. Our GPU-based CUDA implementation is able to further improve performance by another order of magnitude.

In addition, we present two extensions of t-FDP for better exploring graph structures. First, we show how we reveal structures at different scales by re-applying our force to existing layouts with repulsive forces of different magnitudes. Second, we allow users to explore nodes of interest with fisheye-like visualizations by interactively changing force parameters. Since our method is very fast, interactivity can be maintained even for larger graphs.

In summary, our main contributions are as follows:

- we revisit the spring-electric model and show that it does not properly capture local neighborhoods and cluster structures in graphs;
- we propose a new short-range force based on the t-distribution and use it to compose a new FDP model that overcomes drawbacks of existing FDP models in neighborhood preservation; and
- we provide an FFT-based fast implementation¹ and an interactive demo² of our method, and demonstrate its effectiveness through a quantitative evaluation and extensions.

2 RELATED WORK

Related work falls into two categories: force-directed graph layouts and the t-distribution and its applications in graph layout.

2.1 Force-directed Graph Layouts

Since their first proposition by Tutte [13], many variants of force-directed layouts have been developed [4], [14]. Among them, spring-electric and stress models are two most popular categories.

Spring-electric model. By representing nodes as ring-like joints and edges as springs, the spring-electric model [1], [5] uses attractive forces to pull adjacent nodes together and repulsive forces between all nodes to repel them. When the energy of the system reaches a minimum, it finds the optimal layout. To avoid strong long-range forces, Eades [5] introduced logarithmic spring forces and repulsive forces inverse to the squared distance. To produce layouts with uniform edge lengths, Fruchterman and Reingold [1] (FR) redefine attractive forces proportional to the squared distance and repulsive forces reciprocal to the distance. Hu et al. [7] model repulsive forces as power functions of the form $f(d) = d^{-p}$ with p being any positive integer and find that forces inversely related to the squared distance work well for most graphs. Noack [15] generalizes this idea to a LinLog energy model which uses constant attractive forces and repulsive forces reciprocal to the distance, yielding well-separated clusters for their tested graphs.

As a compromise between LinLog and FR model, ForceAtlas2 [16] sets attractive forces proportional and repulsive forces reciprocal to the distance for better showing

1. <https://github.com/Ideas-Laboratory/t-fdp>

2. <https://t-fdp.github.io>

local neighborhoods and cluster structures. However, as pointed out by Fruchterman and Reingold [1], such a configuration might work poorly for complex graphs, because strong short-range repulsive forces often let the corresponding optimization be trapped in local minima. To alleviate this issue, Kumar *et al.* [17] introduce a heuristic method that imposes an upper bound on the repulsive forces as a stopping condition for visualizing directed acyclic graphs. In contrast, short-range t-forces in our t-FDP model allow us to largely alleviate this issue for general graphs (see Section 3.3).

To visualize large graphs, various multilevel approaches have been employed for improving scalability. For example, the Barnes-Hut technique [18] and the fast multipole method [19] are often used for approximating long-range repulsive forces [7], having an overall time complexity of $O(n \log n)$ for n nodes. Since the Barnes-Hut approximation is easy to implement, it is adopted by many FDP algorithms such as SFDP [7] and ForceAtlas2 [16]. Random vertex sampling was proposed by Gove [20] for computing repulsive forces. It generates similar layouts as Barnes-Hut and FMM, but with a time complexity of $O(n)$. Based on FFT acceleration, our t-FDP model also shows a runtime of $O(n)$, but the resulting layouts maintain lower stress errors.

Stress model. Alternatively, stress models, originally proposed by Kamada and Kawai [21] associate a spring between each pair of nodes with an ideal length proportional to the length of the shortest path between them. In doing so, this model can yield a layout with a much better global structure than a spring-electric model. However, stress models pay more attention to distant node pairs, resulting in typically poor preservation of local structures.

To preserve such local structures better, local versions of stress models [6], [22]–[24] have been proposed. The maxent-stress model [6] applies stress only on edges within a specified length, but electric repulsion forces to all nodes. This reduces the computational costs for the shortest paths of all node pairs. In contrast to this, t-FDP aims at improving spring-electric models by better balancing the representation of local and global structures. Our experimental results show that it performs better than the maxent-stress model for representing global structures while being significantly faster.

2.2 t-Distribution and its Applications for Graph Layout

The t-distribution [25] is a symmetric bell-shaped distribution with heavier tails compared with the Gaussian distribution. The function is defined by:

$$f(x) = \left(1 + \frac{x^2}{2v-1}\right)^{-v},$$

where v is the degree of freedom. With increasing v the distribution becomes closer to the Gaussian distribution.

By employing such a distribution to model the similarity between two points in the embedding space, the neighborhood embedding t-SNE [11] not only greatly alleviates the crowding problem of dimensionality reduction (DR) but also efficiently preserves local structures.

A number of t-distribution based DR methods have been proposed, such as LargeVis [26], UMAP [27] and TriMap [28] for preserving more global structures. On

the other hand, acceleration strategies were developed for improving the scalability of t-SNE, such as the Barnes-Hut approximation [29], GPU-based texture splatting [30] and interpolation-based FFT [12]. Among them, the interpolation-based FFT method can achieve a near-linear complexity by using the fast Fourier transform to approximate the repulsive force, which is one term of the negative gradient.

Recently, t-distribution based DR methods have been leveraged for graph layout. A representative approach is tsNET [2], which utilizes a customized t-SNE for capturing local structures. Compared to the stress model based on multidimensional scaling MDS [31], tsNET can generate high-quality layouts for a wider variety of graphs. To handle large graphs, Zhu et al. propose DRGraph [10] that further improves the scalability of tsNET by using negative sampling for gradient estimation.

In a similar spirit, we use the t-distribution to define the bounded short-range force in our t-FDP model. Furthermore, this formation enables us to employ the FFT [12] for improving the scalability, which is about one order of magnitude faster than DRGraph on the CPU and two orders faster on the GPU.

3 METHOD

Given an undirected graph with n nodes and m edges, the main goal of graph layout is to compute a low-dimensional position for each node that meets a set of structural and aesthetic criteria including evenly distributed nodes, uniform edge lengths, and reflecting symmetry. However, force-directed layout algorithms and especially spring-electric models do not explicitly strive for these goals but are solely based on two design principles proposed by Fruchterman and Reingold [1]:

P1: Nodes connected by an edge should be drawn close to each other; and

P2: Nodes should not be drawn too close to each other in general.

These two principles, however, cannot ensure that connected nodes become nearest neighbors in the layout for graphs with more than two nodes (see Fig. 1(a)). This results in losing important local graph structures. To address this issue, we propose a third design principle:

P3: Nodes connected by an edge should be drawn closer to each other than unconnected nodes.

This will enhance local structures and represent local connection patterns more faithfully.

3.1 Revisiting Spring-electric Models

To meet **P1** and **P2**, the spring-electric model employs attractive spring forces f^a to pull connected nodes together and electric repulsive forces f^r to repel nodes from each other. So far, different functions have been used for defining attraction and repulsion forces [7], [16] and many of them can be written in the form of power functions:

$$F^a(i, j) = \alpha \|\mathbf{x}_i - \mathbf{x}_j\|^p, \quad i \leftrightarrow j, \quad (1)$$

$$F^r(i, j) = -\|\mathbf{x}_i - \mathbf{x}_j\|^{-q}, \quad i \neq j, \quad (2)$$

where \mathbf{x}_i is the 2D position of the graph node i in the layout space, α is the weight, and p and q have to be non-negative.

The larger p , the stronger the long-range attractive force, while the larger q , the weaker the long-range repulsive force. At each time step, the resultant force will move the node until convergence is reached.

The layout quality is mainly influenced by p and q . By using the exponents $(2, 1)$ for p and q , respectively, the FR model [1] yields layouts with uniform edge lengths. The LinLog model [15] uses the exponents $(0, 1)$ for better revealing clusters in the graph. Recently, Jacomy et al. [16] suggest to use the exponents $(1, 1)$ as an intermediate model between these two models, these values are also set as default in many visualization libraries or systems such as GraphViz [32] or D3 [9].

Drawbacks. Attractive spring and repulsive electric forces allow to meet **P1** and **P2** for small and simple graphs. However, for large graphs, such forces based on power functions might not be able to adequately represent graph neighborhoods and cluster structures. In the following, we describe the drawbacks of classical FDP detailing the two problematic cases shown in Fig. 2.

First, as mentioned above, repulsive forces tend to become extremely large when the Euclidean distance between two nodes approaches zero. As a result, closely connected nodes might be pushed away from each other (see V_1 in Fig. 2(a)). In doing so, the resulting Euclidean distances often do not correlate with graph-theoretical distances (see Fig. 1(a)). This is counter-intuitive for graph exploration.

Second, long-range attraction forces that are too strong might break a cluster into multiple smaller pieces. As shown in Fig. 2(b), node V_2 is affected by a strong long-range attraction force from an outlier and moved out of the main cluster. Reducing long-range attraction forces or increasing short-range attraction might reduce this issue. However, adjusting the corresponding coefficients in the spring-electric model will always change both, long- and short-range forces at the same time.

Although the above two issues can be alleviated by assigning different coefficients to the forces at different levels as possible in multilevel methods [7], [33], [34], power-function-based forces inherently suffer from exerting extremely large repulsive forces and small attractive forces for short-ranges. On the other hand, small repulsive and large attractive forces for long-ranges help to reduce long edges, resulting in a layout with reasonably global structures. Thus, we propose our new model that decouples these effects and allows for improving repulsive and attractive forces at short-ranges, while keeping the characteristics of the traditional FDP models at long-range.

3.2 The t-Force

One straightforward way to avoid extremely large repulsive forces would be to define a constant force value when two nodes approach each other. However, only manipulating repulsive forces with a constant cannot meet **P3**, because it is hard to find a good balance to the spring-like attractive forces. Therefore, it is better to increase attraction and reduce repulsion at short-ranges so that neighbors are able to move closer together. To do so, we propose the t-force as a new short-range force that serves as two roles: a new repulsive force and a component of the attractive force for short-ranges.

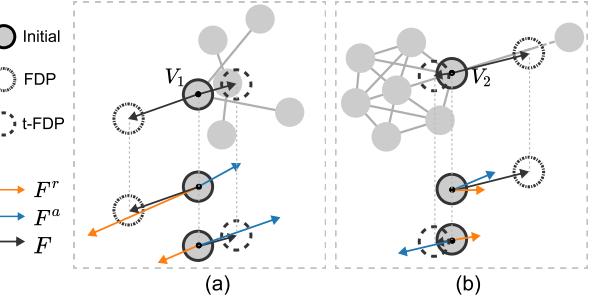


Fig. 2. Two problematic cases of traditional FDP: (a) three nearby non-neighbouring nodes exert large repulsive forces to push V_1 out of its cluster; (b) a strong long-range attractive force pulls V_2 out of a local cluster. t-FDP avoids both cases.

As a repulsive force, it should be short-range in nature, and thus we require it to be weak at long ranges similar to power-function-based repulsive forces. Moreover, it should have an upper bound at short ranges to avoid extremely large contact forces and improve optimization stability. Finally, to be used as an additional component of attractive forces at short ranges, we further require it to have similar behavior as a linear spring force, which is quite strong compared to other forces as shown by Eades [5]. Suppose the Euclidean distance between two graph nodes is $d = \|\mathbf{x}_i - \mathbf{x}_j\|$, the force $f(d)$ should satisfy the following three requirements:

- **R1:** $\exists \varsigma > 0$ s.t. $0 < f(d) \leq \varsigma, \forall d > 0$;
- **R2:** $f(d) \sim d^{-q}$ as $d \rightarrow \infty$, where $q > 0$; and
- **R3:** $f(d) \sim d$ as $d \rightarrow 0$.

In other words, such a force has an upper bound ς for short distances between two nodes and smoothly reduces to zero as the distance decreases with different possible decay rates.

After investigating a number of functions, we found the following t-distribution based force to meet our requirements. This does not mean that no other possible functions exist, but this one seems to be very simple and provides all required aspects:

$$f(d) = \frac{2\tau\tilde{\varphi}d}{(1 + \tau d^2)^{\tilde{\varphi}+1}}, \quad (3)$$

where τ and $\tilde{\varphi}$ are constants not less than zero. For simplicity, we set τ to 1 and simplify the notation of Eq. 3 to the following function:

$$f(d) = \frac{d}{(1 + d^2)^\varphi} \quad (4)$$

with exponent $\varphi \geq 1$.

The function $f(d)$ approaches 1 when $|d|$ is close to zero and gradually decreases to zero as $|d|$ increases. We refer to the heavy-tailed force defined by Eq. 4 as *t-force*. Fig. 3(a) shows $f(d)$ for three different values of φ . The maximum is always smaller than 1 and located in the d -range of $[0,1]$, $f(d)$ is heavily influenced by φ . The smaller φ , the larger the maximum and the heavier its tail. No matter what φ is, the function has an upper bound of ς .

3.3 The t-FDP model

Using the t-force, we now define the repulsive and attractive forces of our force-based graph layout as:

$$F^r(i, j) = -\frac{\|\mathbf{x}_i - \mathbf{x}_j\|}{(1 + \|\mathbf{x}_i - \mathbf{x}_j\|^2)^\gamma} \frac{\mathbf{x}_i - \mathbf{x}_j}{\|\mathbf{x}_i - \mathbf{x}_j\|}, \quad i \neq j, \quad (5)$$

$$F^a(i, j) = \left(\|\mathbf{x}_i - \mathbf{x}_j\| + \frac{\beta \|\mathbf{x}_i - \mathbf{x}_j\|}{1 + \|\mathbf{x}_i - \mathbf{x}_j\|^2} \right) \frac{\mathbf{x}_i - \mathbf{x}_j}{\|\mathbf{x}_i - \mathbf{x}_j\|}, \quad i \leftrightarrow j, \quad (6)$$

where β is the weight for the attractive t-force. Following the suggestion of ForceAtlas2 [16], we use a linear attractive spring force, and an attractive t-force with $\varphi = 1$.

Combining both forces on the i -th node yields the following resultant force:

$$F(i) = \sum_{i \neq j} -\frac{\|\mathbf{x}_i - \mathbf{x}_j\|}{(1 + \|\mathbf{x}_i - \mathbf{x}_j\|^2)^\gamma} \frac{\mathbf{x}_i - \mathbf{x}_j}{\|\mathbf{x}_i - \mathbf{x}_j\|} + \alpha \sum_{i \leftrightarrow j} \left(\|\mathbf{x}_i - \mathbf{x}_j\| + \frac{\beta \|\mathbf{x}_i - \mathbf{x}_j\|}{1 + \|\mathbf{x}_i - \mathbf{x}_j\|^2} \right) \frac{\mathbf{x}_i - \mathbf{x}_j}{\|\mathbf{x}_i - \mathbf{x}_j\|}, \quad (7)$$

where γ is the exponent for the repulsive t-force component, and α is the weight for the attractive force. This model has the same attractive spring forces as the original model, but enhances it with an attractive short-range t-force and replaces its repulsive force with a repulsive short-range t-force. The attractive short-range t-force has a similar form as the one in tsNET (see the supplemental material), which is weighted by an extra term that is derived from graph theoretical distances. In contrast to tsNET, our attractive short-range t-force is only exerted on given edges, and is combined with long-range spring forces for better maintaining global structures (see Section 4.2).

Parameter Analysis. The parameter γ specifies the extent and magnitude of the repulsive t-force that controls the longest distance of neighbors in the layout, while α and β tune the weight of the attractive long-range and short-range t-forces, respectively. To meet the above three principles, they have to be set in a reasonable way. However, analyzing the relationship between them for large graphs is hard because each node is influenced by potential forces from many other nodes. For simplicity, we first investigate their valid ranges from the extreme case with two connected nodes and then provide guidelines for general graphs.

For specifying α and β , we formally represent **P2** as the relationship between repulsive and attractive force for two connected nodes i and j :

$$\lim_{\|\mathbf{x}_i - \mathbf{x}_j\| \rightarrow 0} \frac{F^a(i, j)}{F^r(i, j)} < 1. \quad (8)$$

Substituting Eq. 5 and Eq. 6 into Eq. 8 yields:

$$\alpha(1 + \beta) < 1. \quad (9)$$

We can see that α and β cannot be both large, while they cannot be both small so as to balance attractive and repulsive forces. The smaller α , the weaker the long-range attraction and the more likely it is to produce long edges, resulting in the layout with uniformly distributed nodes (see Fig.4(a) which uses $\alpha = 0.01$). Otherwise, long-edges are increased and the layout tends to split into many sub-clusters (see Fig.4(a) with $\alpha = 0.50$). Increasing β results in the stronger the short-range attraction and the more likely it is to group connected nodes together, forming more sub-clusters (see Fig.4(b) with $\beta = 50$). For a fixed β , a smaller α lets $\alpha\beta$ decrease and the proportion of repulsive forces increase. In this case the layout tends to distribute nodes more uniformly (see Fig.4(a) which uses $\alpha = 0.01$). We can see that there are similar trends in Figs.4(a,b).

For a fixed $\alpha\beta$, decreasing α increases the proportion of short-range attractive and repulsive forces relative to the overall forces, resulting in a better preservation of graph

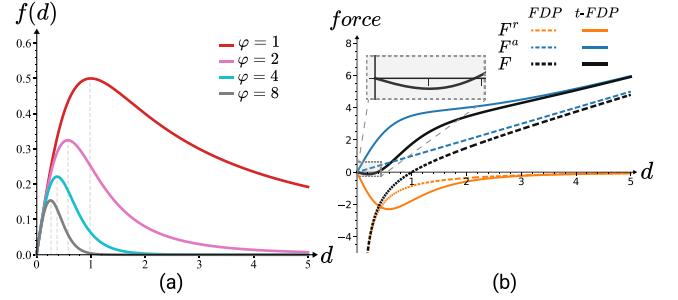


Fig. 3. (a) Function graphs of the t-force for different φ ; (b) comparison of repulsive (orange), attractive (blue) and resultant (black) forces exerted on two connected nodes between FDP and t-FDP shown as the dotted and solid lines, respectively. The major difference are the short-range forces, which are bounded in t-FDP.

neighborhoods (see the amount of the 1-ring neighborhood preservation in the left of Fig.4(b)). Yet, distance preservation will be worse (see stress error in Fig.4(b)) because smaller long-range attractive forces lead to many long edges.

For specifying γ , we formally represent **P3** which requires the attractive force between two connected nodes to be larger than the repulsive force for forming reliable clusters except when nodes approach each other. Hence, F^a and F^r satisfy the following condition:

$$F^a(i, j) > F^r(i, j) \quad \text{if } \|\mathbf{x}_i - \mathbf{x}_j\| > \epsilon$$

where ϵ is a small value but larger than zero. Since the t-force is a short-range force, the attractive force is certainly larger at long ranges than the repulsive force because of its spring force component. However, for short distances this condition might not hold. As shown in Fig. 3(a), the larger the exponent γ , the smaller the force magnitude. To prevent the attractive t-force weighted by $\alpha\beta$ from being smaller than the repulsive force, we require:

$$\gamma > 1. \quad (10)$$

As shown in Fig.3(a), larger exponents γ result in repulsive forces with shorter range. In relative terms, the resultant force creates an attraction at closer distances, which leads to better representing local cluster structures (see Fig.5(a)) but does not efficiently preserve neighborhoods and distances (Fig.5(b)).

Overall, γ specifies the extent and magnitude of the repulsive t-force that controls the longest distance of neighbors in the layout, while α and β tune the weights of the attractive long-range and short-range t-forces, respectively. In our experiments, we find that a configuration of $\alpha = 0.1$, $\beta = 8$ and $\gamma = 2$ works well and preserves local as well as global structures. Fig. 3(b) compares attractive, repulsive, and resultant forces exerted on a node using the standard FDP and our t-FDP model. The repulsive force is larger than the attractive force when the distance between two nodes is smaller than a certain value; otherwise, the attractive force is larger. This is well aligned with the three requirements **R1-R3**.

3.4 Approximate Calculation of Repulsive Forces

Computing repulsive forces in a graph involves the interaction between each node and all other nodes, resulting in a computational effort of $O(n^2)$ (we refer to this as the *exact* method). As proposed in [29] for t-SNE, the computation can be accelerated by using tree-based

approximation methods such as Barnes-Hut (BH) [18] to truncate the t-forces. However, this still requires $O(n \log n)$ computations, which is infeasible for large graphs. Random vertex sampling [20] runs in $O(n)$ time, but the resulting layouts for large graphs are often worse than the ones generated by the other methods. To address this issue, we introduce an interpolation-based Fast Fourier Transform (ibFFT) algorithm, which was originally designed for accelerating t-SNE [12]. It yields similar layouts as the BH method, but is about ten times faster for most large graphs.

To do so, we first represent the repulsive t-force as multiple sums of the weighted kernel functions:

$$\begin{aligned} F^r(i) &= \sum_{j=1, j \neq i}^n \frac{\|\mathbf{x}_i - \mathbf{x}_j\|}{(1 + \|\mathbf{x}_i - \mathbf{x}_j\|^2)^\gamma} \frac{\mathbf{x}_i - \mathbf{x}_j}{\|\mathbf{x}_i - \mathbf{x}_j\|} \\ &= \sum_{j=1, j \neq i}^n \frac{\mathbf{x}_i - \mathbf{x}_j}{(1 + \|\mathbf{x}_i - \mathbf{x}_j\|^2)^\gamma} \\ &= \mathbf{x}_i \sum_{j=1}^n \mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) - \sum_{j=1}^n \mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) \mathbf{x}_j \end{aligned} \quad (11)$$

where the kernel $\mathbf{K}(\mathbf{x}_i, \mathbf{x}_j)$ is:

$$\mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{(1 + \|\mathbf{x}_i - \mathbf{x}_j\|^2)^\gamma}. \quad (12)$$

Since \mathbf{x}_j is a 2D point, each dimension can be calculated independently:

$$F^r(i)_{(1)} = \mathbf{x}_{i(1)} \sum_{j=1}^n \mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) - \sum_{j=1}^n \mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) \mathbf{x}_{j(1)} \quad (13)$$

$$F^r(i)_{(2)} = \mathbf{x}_{i(2)} \sum_{j=1}^n \mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) - \sum_{j=1}^n \mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) \mathbf{x}_{j(2)}, \quad (14)$$

where $\mathbf{x}_{j(1)}$ and $\mathbf{x}_{j(2)}$ are the x-coordinate and y-coordinate of \mathbf{x}_j , respectively. Hence, $F^r(i)$ is determined by three sums of products between the kernel $\mathbf{K}(\mathbf{x}_i, \mathbf{x}_j)$ and the weight v_j :

$$\psi(\mathbf{x}_i) = \sum_{j=1}^n \mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) v_j. \quad (15)$$

where v_j is 1, $\mathbf{x}_{j(1)}$, or $\mathbf{x}_{j(2)}$. For example, $F^r(i)_{(1)}$ is computed by two sums of products in Eq. 13 with the weights v_j being 1 and $\mathbf{x}_{j(1)}$. Using the naive way to compute such a sum takes n^2 time, which is prohibitive for a large number of nodes n .

Alternatively, Linderman et al. [12] exploited the low-rank nature of the kernel \mathbf{K} to approximate Eq. 15 with a small number ($k \times k$) of equi-spaced 2D grid nodes. This is done in three steps:

- projecting all data points \mathbf{x}_i onto the grid by using Lagrange polynomials with a time complexity $O(k^2 n)$;
- computing the interaction of the grid nodes, which can be accelerated by FFT with a complexity $O(2k^2 \log k)$; and
- back-projecting the interaction of all grid nodes to the original points with time complexity $O(k^2 n)$.

We can see that the overall time complexity is $O(k^2 n)$ where k is a small number.

However, the above polynomial interpolation scheme suffers from the Runge phenomenon [35] which reduces the

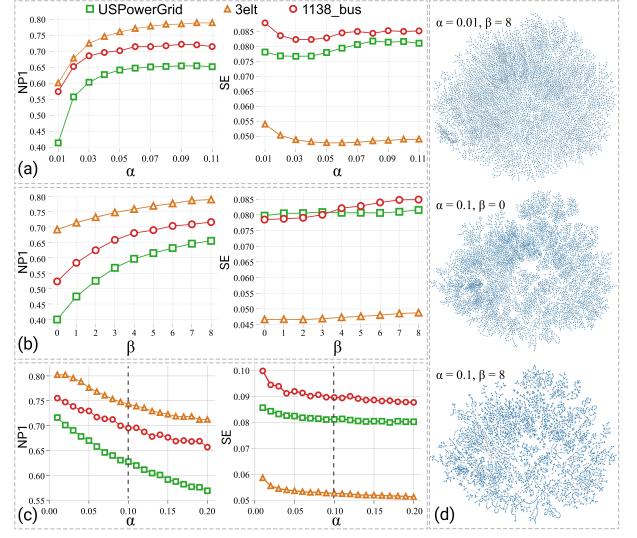


Fig. 4. Influence of α and β for graph layouts: (a) For a fixed $\beta = 8$, a small α distributes nodes in the layout of the *USpowerGrid* graph more evenly; (b) For a fixed $\alpha = 0.1$, a large β results in more sub-clusters; (c) Different combinations of α and β for $\alpha\beta = 0.8$ create different 1-ring neighborhood preservation values (higher is better) and stress errors (lower is better) for the three given graphs. $\alpha = 0.1$ creates a good balance between optimizing both aspects; (d) Three layout results of the *USpowerGrid* graph generated by different combinations of α and β .

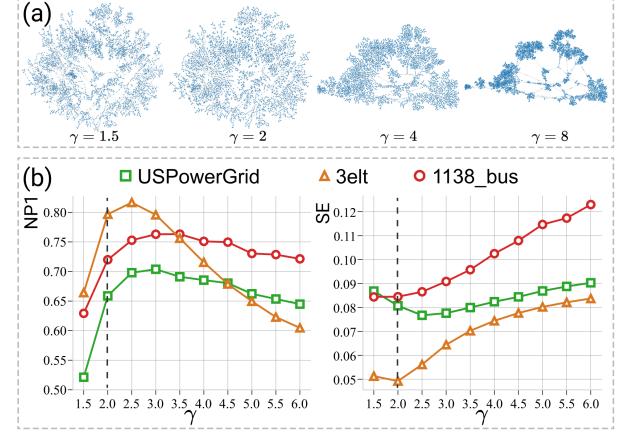


Fig. 5. Influence of γ for graph layouts: (a) Layouts of the *USpowerGrid* graph for $\gamma = 2, 4, 8$; (b) 1-ring neighborhood preservation(left) and stress error(right) for varying γ for three graphs. $\gamma = 2$ creates a good balance between optimizing both aspects.

accuracy as k increases. To overcome this issue, Linderman et al. [12] proposed to divide the interval of the whole layout space $[\mathbf{x}_{\min}, \mathbf{x}_{\max}] \times [\mathbf{x}_{\min}, \mathbf{x}_{\max}]$ into a collection of $N_{\text{int}} \times N_{\text{int}}$ equal sized intervals and then apply polynomial interpolation in each interval with $k \times k$ equi-spaced nodes. Specifically, the computation is done by first projecting each data point \mathbf{x}_i into the corresponding grid square, then computing the interaction of the $k \times k$ equi-spaced grid nodes within each grid square; and finally back-projecting the interaction of all grid nodes within each grid square to the original points. The first and last steps still requires $O(k^2 n)$ operations, while the second one can be done in $O(2(N_{\text{int}} k)^2 \log(N_{\text{int}} k))$ operations with the FFT acceleration. Hence, the total computational complexity becomes $O(nk^2 + (N_{\text{int}} k)^2 \log(N_{\text{int}} k))$.

Choices of k . Linderman et al. [12] suggested to set $k = 3$

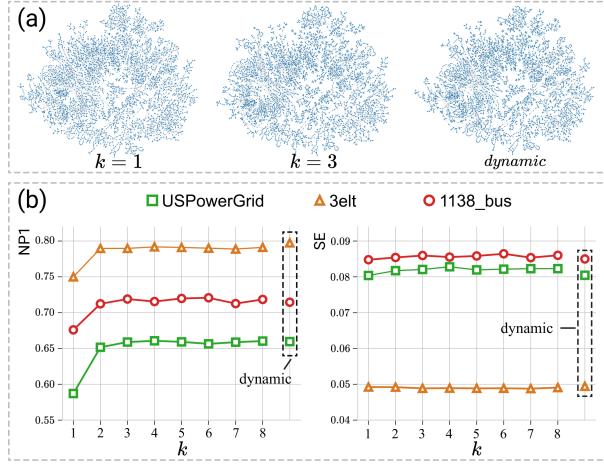


Fig. 6. Influence of different k values on the layout. (a) Layouts of the *UsPowerGrid* graph generated by using $k = 1$, $k = 3$, and our dynamic strategy; (b) 1-ring neighborhood preservation(left) and stress error(right) vary with k on three graphs. The values of our dynamic strategy are in the dotted boxes.

and $N_{\text{int}} = \max(50, [y_{\text{max}} - y_{\text{min}}])$ for t-SNE when visualizing high dimensional data. We found that the layouts generated by using $k = 3$ are similar to the ones produced by using $k = 1$ for displaying global structures (see Fig. 6(a)) but better in neighborhood preservation (left of Fig. 6(b)). However, the computational complexity quadratically increases with k .

To find a good trade-off, we suggest to vary the values of $k = 1, 2, 3$ during the optimization: we start with $k = 1$ for 90% of the iterations to obtain a reasonable layout and then use $k = 2$ for 5% of the iterations and finally $k = 3$ for the remaining iterations. An example is shown in the right of Fig. 6(a), which has almost the same structure as the one generated by directly using $k = 3$ (see the middle in Fig. 6(a)). We tested multiple datasets and found that such dynamic changes in k values generates high-quality layouts (see values for the dynamic strategy in Fig. 6(b)) with the least computation time. The computation time of the dynamic strategy is close to that of the setting $k = 1$.

Based on such choices of k , our t-FDP can run in $O(n)$ time, especially for large graphs with N_{int} being much smaller than n .

4 EVALUATION

Following UMAP [27], we implemented t-FDP in *Python3* and used the *numba* library [36] compiler to translate the Python code to fast machine code with a similar speed as plain C code. To accelerate it using the GPU, we used the open-source matrix library *cupy* to speed up matrix computations with NVIDIA CUDA. To accommodate a wide range of users, we further provide a Javascript implementation as a drop-in force for the “d3-force” library, which allows for accelerating it with WebGL if a GPU is available. In the following, we refer to the interpolation-based Fast Fourier Transform (ibFFT) implementation of t-FDP by *ibFFT*.

To demonstrate its effectiveness, we evaluate t-FDP in two ways. First, we validate our FFT-based approximation by comparing it with the exact method and other approximations. Second, we compare it with a few state-of-the-art layout methods on a set of synthetic and real-world

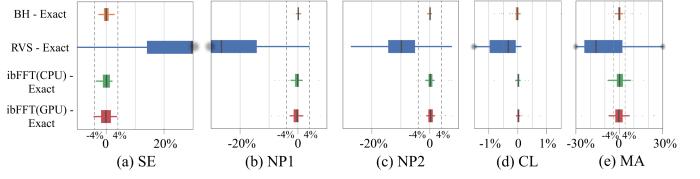


Fig. 7. Five relative error metrics (abbreviations see text) for layouts generated by comparing four approximation methods with the exact method of the t-FDP model

graphs of various sizes. All results were measured on a desktop machine with Intel i7-8700 processor, 32 GB CPU memory, and NVIDIA GeForce RTX 2080 GPU (8GB GPU memory).

Datasets. For a comprehensive evaluation, we collected 50 graphs with varying numbers of nodes and edges (see Table 1 in the supplemental material). Most of them are selected from the Florida Collection [37], the SNAP network collection [38] and the collections used by tsNET [2] and DRGraph [10]. As for FDP methods, our t-FDP model can inherently handle multi-component graphs and hence we include 7 graphs with more than one component. In addition, we synthesized three graphs with various cluster structures by using the graph-tool python library [39].

Methods. The comparison includes ten graph layout methods: force-directed placement by Fruchterman-Reingold (FR) [1], force-directed layout by random vertex sampling (FR-RVS) [20], SFDP [7], ForceAtlas2 (FA2) [16], Linlog [15], PivotMDS (PMDS) [40], stress model (SM) [41], the maxent-stress model (Maxent) [6], tsNET [2] and DRGraph [10]. Including our methods, we can categorize all methods into three groups: force-based methods (FR, SFDP, FA2, LinLog, and t-FDP), stress-based methods (PMDS, SM, Maxent) and neighboring embedding methods (tsNET and DRGraph). We use the public C++ libraries OGDF [42] and GraphViz [32] for performing FR, SM, and SFDP while taking the implementations of FA2, Linlog, Maxent, tsNET and DRGraph from the authors of the original papers. We implemented PMDS by ourselves instead of using the method provided by OGDF, since it was not stable and did not allow to efficiently handle large data. tsNET was accelerated with a GPU-based t-SNE implementation [43]. For FR-RVS, the public Javascript implementation cannot handle large graphs and we re-implemented it with Python and used the Numba [36] library for acceleration. We ran each method with the default parameters provided by the original paper or implementation. For example, we set the perplexity of tsNET to 40 and choose the first-order nearest neighbor for DRGraph.

For fair comparisons, we use the same PMDS layout as initialization for all methods except SFDP and DRGraph, which are initialized by a multi-level scheme. Since SFDP and DRGraph are inherently random, we assess the average quality at runtime of each method on one graph by calculating the average over 5 runs. We provide the results of a random initialization in the supplemental material, they show that most of the methods can be significantly improved by using the PMDS layout as initialization, which is consistent with the previous finding in t-SNE and UMAP [44].

Metrics. Following the evaluations of tsNET [2] and DRGraph [10], we evaluate graph layouts with four metrics: normalized stress error, neighborhood preservation degree, crosslessness, and minimum angle. The first two metrics characterize the graph structure in preserving distances and neighborhoods, while the latter two reflect graph readability in terms of edge crossing and minimum angle.

- *Normalized stress error (SE)* [6] To assess the distance-preservation of a layout, we use the normalized stress error defined as:

$$SE = \frac{2}{n(n-1)} \sum_{i < j} \frac{(\bar{s}\|\mathbf{x}_i - \mathbf{x}_j\| - d_{ij})^2}{d_{ij}^2},$$

where d_{ij} is the shortest path distance between the nodes i and j and \bar{s} is a scaling factor to uniformly scale the layout. When the nodes i and j are unreachable in the graphs with multiple components, we follow DRGraph [10] that defines d_{ij} as infinity and only computes stress errors for the node pairs with finite distances. To make a fair comparison, the optimal scaling factor \bar{s} is obtained by minimizing the normalized stress error. For more details, please refer to Gansner et al. [6].

- *Neighborhood Preservation Degree (NP)*. To measure layout quality in preserving neighborhood, the neighborhood preservation degree is defined as the Jaccard similarity between the input graph and k_i -nearest neighborhood graph defined in the layout:

$$NP = \frac{1}{n} \sum_i \frac{|N_G(i, r) \cap N_L(\mathbf{x}_i, k_i)|}{|N_G(i, r) \cup N_L(\mathbf{x}_i, k_i)|},$$

where $N_G(i, r)$ are r -ring neighbors of node i in graph space, k_i is $|N_G(i, r)|$, and $N_L(\mathbf{x}_i, k_i)$ is a set of nodes corresponding to the k_i -nearest-neighbors of \mathbf{x}_i in the layout space. In our study, we compute the neighborhood preservation degrees with 1-ring and 2-ring graph neighborhoods and denote them as $NP1$ and $NP2$, respectively.

- *Crosslessness (CL)*. To indicate the degree of edge crossing, the crosslessness metric [45] is defined as a normalized value of the number of edge crossings:

$$CL = \begin{cases} 1 - \sqrt{c/c_{max}}, & \text{if } c_{max} > 0 \\ 1, & \text{otherwise.} \end{cases}$$

Here, c is the number of edge crossings, and c_{max} is the theoretical upper bound of this number in each graph. A larger value indicates a better layout with edge crossing.

- *Minimum Angle (MA)*. The minimum angle metric [45] computes the mean deviation between the actual minimum angle and the ideal minimum angle between edges:

$$MA = 1 - \frac{1}{n} \sum_i \frac{|\theta(i) - \theta_{min}(i)|}{\theta(i)}, \quad \theta(i) = \frac{2\pi}{\text{degree}(i)}$$

where $\theta_{min}(i)$ is actual minimum angle at the node i . MA is in the range $[0,1]$ and reaches the maximum when all the nodes have equal angles between all incident edges.

For comparing different methods, we calculated the relative values for the above-given metrics computed from

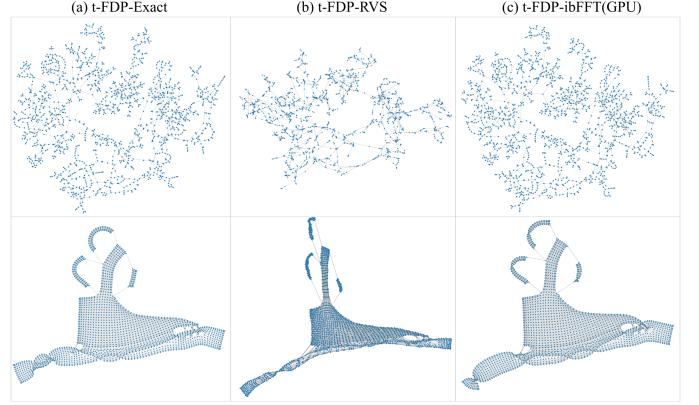


Fig. 8. Visual results for *bcsppwr07* and *walshaw-1* generated by the exact t-FDP model (a) and two approximation methods: t-FDP-RVS (b) and t-FDP-ibFFT(GPU) (c).

layouts generated by a source s and a target method t . Taking the *SE* metric as an example, the relative error between two layouts \mathbf{X}_s and \mathbf{X}_t of one graph is:

$$\overline{SE} = \frac{SE(\mathbf{X}_s) - SE(\mathbf{X}_t)}{SE(\mathbf{X}_t)},$$

for any $SE(\mathbf{X}_t)$ larger than zero. Since a smaller stress error indicates better distance preservation, a negative relative error reflects that the source method performs better. For the other metrics, positive relative errors indicate that the source method performs better.

4.1 Comparison of Approximation Methods

The goal of this experiment was to measure the efficiency of the CPU and GPU versions of our ibFFT algorithm. It is done by comparing four approximation methods (BH [18], RVS [20], and the CPU and GPU versions of our ibFFT algorithm) with the exact method in terms of the quality statistics, convergence rate and runtime performance. For simplicity, we refer five methods as *t-FDP-BH*, *t-FDP-RVS*, *t-FDP-ibFFT(CPU)*, *t-FDP-ibFFT(GPU)* and *t-FDP-exact*, respectively.

Quality Statistics. The boxplots in Fig. 7 summarize the relative values of five layout quality metrics between four pairs of counterparts.

Except the comparison to t-FDP-RVS, we can see that the relative values of all metrics are in the range [-4%, 4%] with a mean value very close to zero. Compared to t-FDP-exact, t-FDP-BH yields the smallest interquartile ranges, followed by two versions of our methods, while t-FDP-RVS performs the worst. Our methods result in slightly more negative relative values for SE and positive relative values for NP2 and CL, indicating that their resulting layouts are similar or even slightly better than the ones of the exact method for some graphs. In contrast, t-FDP-RVS results relative values for SE that are around 30% and the ones for NP1 are smaller than -20%, indicating that the resulting layouts are considerably worse than the ones of the exact method. We speculate such lower NP1 values are due to our short-range repulsive forces, where a subset of repulsive forces randomly sampled by RVS might not be enough to repel non-neighborhood nodes. Note

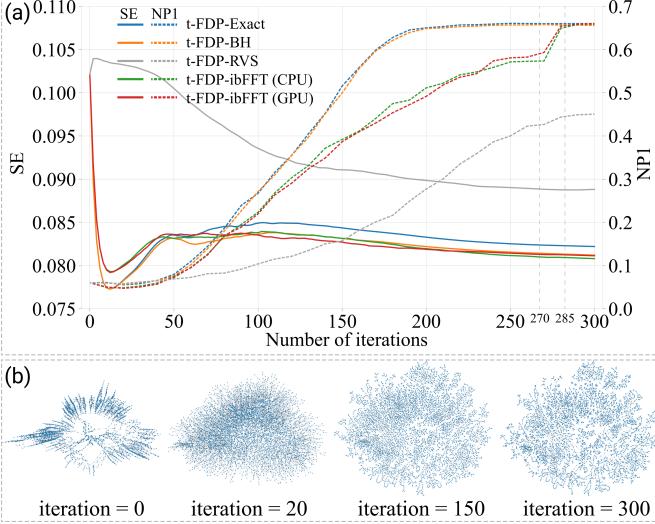


Fig. 9. (a) Convergence of stress error (SE, solid lines) and neighborhood preservation degree (NP1, dashed lines) for applying the exact method and four different approximation methods of the t-FDP model to the *UspowerGrid* graph; (b) the layout results generated at four different iterations.

that the difference between the two versions of t-FDP-ibFFT is caused by floating point finite precision on GPUs.

Fig. 8 shows the results generated by applying the t-FDP-exact, t-FDP-RVS and t-FDP-ibFFT (GPU) on two graphs, where t-FDP-ibFFT (GPU) produces almost the same results as the exact method and performs better than t-FDP-RVS in revealing the tree- and grid-like structures of the two graphs. We speculate that computing our bounded short-range repulsive forces using randomly sampled nodes might not be enough to characterize the graph structures.

Convergence Rate. To further inspect the differences between these methods, we investigate the convergence rate of SE and NP1 on the USPowerGrid graph for five different methods. Since these methods have different runtime per iteration, we explore convergence per iterations (see Fig. 9) and found that all methods show good convergence with increasing number of iterations. The convergence per unit of time of these methods can be found in the supplemental material. With regard to SE, t-FDP-BH and t-FDP-ibFFT behave similarly and perform slightly better than t-FDP-exact. Regarding NP1, two versions of t-FDP-ibFFT perform worse than the others for the first 270 iterations and then quickly almost reach their results. This is due to using only a single interpolation point in the first 90% iterations and then using two and three interpolation points in the remaining iterations (see Section 3.4). In all, our methods yield a similar neighborhood preservation but a slightly better distance preservation than the exact method and its BH approximation. We speculate that this is because of the random noise induced by our approximation methods, which might improve the optimization quality [46]. The supplemental material shows all results generated by these methods, which have highly similar structures. Conversely, t-FDP-RVS behaves quite different from the other methods and yields larger stress errors and lower NP1 values, which is consistent with the result shown in Fig. 7.

Runtime Performance. Regarding runtime performance, the

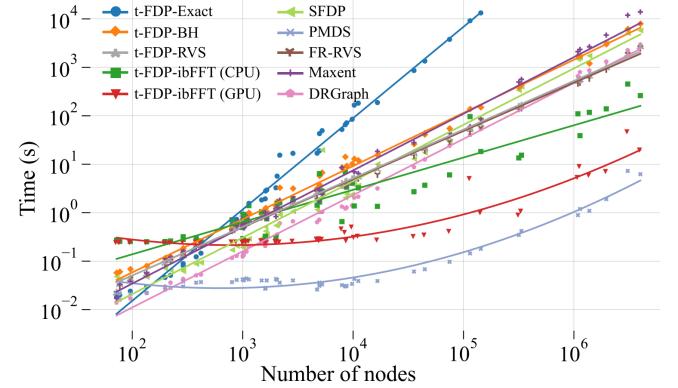


Fig. 10. Computation time of five different implementations of our t-FDP model in comparison to five other methods, which can process all datasets.

curves in Fig. 10 present the relationship between runtime and number of graph nodes. When the number of nodes is smaller than 500, both versions of t-FDP-ibFFT take a runtime of 300ms, which is tolerable but slower than t-FDP-exact. Beyond this, the benefit of ibFFT is clearly demonstrated. We can see that t-FDP-ibFFT (GPU) is one order of magnitude faster than t-FDP-ibFFT (CPU) for large graphs, which is also one order of magnitude faster than the t-FDP-BH. In contrast to that, the exact method is the slowest and takes more than 3 hours to solve a graph with 100K nodes, whereas our GPU version can generate the layout in less than 10 seconds for most graphs with millions of nodes.

Overall, t-FDP-ibFFT (GPU) generates similar layouts as methods like t-FDP-BH, while being much faster for large graphs. Hence, we use this implementation to compare with other layout methods on all datasets.

4.2 Comparison of Layout Methods

Screenshots of the layouts generated by all methods on various graphs with complete scores can be found in the supplemental material. In the following, we compare the layout methods in terms of layout quality statistics, visual results and runtime performance.

Quality statistics. The heatmaps in Fig. 11 present the SE, NP1, and NP2 values generated by eleven different layout methods for 50 graphs, the other metrics can be found in the supplemental material. Each row corresponds to a graph, sorted by their number of nodes, and each column corresponds to a graph layout method. Each cell shows a numerical value with the background color encoding the relative metric on the same row and the empty one indicates that the graph is too large to be processed by the corresponding layout method. We can see that only six methods (FR-RVS, SFDP, PMDS, Maxent, DRGraph and our t-FDP) can handle all graphs. Our t-FDP performs similarly as FR and SFDP in stress errors for most data and performs similarly as tsNET in neighborhood preservation when the number of nodes is less than 500. With the increasing number of nodes, its advantage over the other methods becomes more evident.

Fig. 11(a) shows that Maxent and SFDP work well for most graphs but yield large stress errors for some (e.g., *add32*), whereas tsNET and DRGraph generate large stress

	FR	FR-RVS	SFDP	Linlog	FA2	PMDS	Maxent	SM	tsNET	DRGraph	t-FDP	FR	FR-RVS	SFDP	Linlog	FA2	PMDS	Maxent	SM	tsNET	DRGraph	t-FDP	FR	FR-RVS	SFDP	Linlog	FA2	PMDS	Maxent	SM	tsNET	DRGraph	t-FDP
dwt_72	0.037	0.044	0.047	0.172	0.057	0.065	0.034	0.028	0.043	0.095	0.059	0.991	0.966	0.962	0.600	0.931	0.854	0.934	0.981	0.981	0.817	0.968	0.883	0.906	0.876	0.626	0.845	0.681	0.899	0.928	0.882	0.851	0.905
lesmis	0.126	0.225	0.122	0.213	0.149	0.155	0.126	0.081	0.126	0.173	0.130	0.428	0.312	0.399	0.528	0.513	0.261	0.167	0.312	0.567	0.570	0.544	0.723	0.626	0.716	0.627	0.629	0.747	0.572	0.688	0.697	0.656	0.680
can_96	0.074	0.091	0.074	0.083	0.075	0.086	0.149	0.073	0.083	0.087	0.082	0.360	0.594	0.319	0.258	0.498	0.225	0.237	0.504	0.715	0.510	0.688	0.536	0.609	0.528	0.524	0.568	0.474	0.411	0.553	0.657	0.595	0.641
raja11	0.075	0.092	0.073	0.179	0.183	0.117	0.120	0.063	0.107	0.154	0.100	0.531	0.500	0.564	0.688	0.693	0.409	0.146	0.418	0.671	0.610	0.750	0.640	0.629	0.677	0.563	0.626	0.615	0.375	0.623	0.724	0.681	0.678
jazz	0.146	0.156	0.147	0.250	0.175	0.169	0.131	0.101	0.161	0.188	0.151	0.419	0.328	0.411	0.457	0.401	0.319	0.226	0.356	0.530	0.403	0.506	0.841	0.775	0.840	0.776	0.773	0.83	0.766	0.829	0.774	0.722	0.814
visbrazil	0.082	0.123	0.086	0.186	0.119	0.194	0.086	0.060	0.097	0.128	0.083	0.139	0.142	0.142	0.480	0.397	0.028	0.036	0.070	0.310	0.337	0.545	0.464	0.407	0.467	0.495	0.506	0.369	0.323	0.387	0.546	0.517	0.518
grid17	0.018	0.019	0.019	0.171	0.025	0.027	0.014	0.014	0.020	0.025	0.017	0.998	0.958	0.985	0.342	0.763	0.717	1.000	1.000	0.961	0.649	0.983	0.780	0.796	0.770	0.392	0.783	1.000	1.000	0.769	0.711	0.809	
mesh3d	0.008	0.030	0.016	0.065	0.017	0.042	0.003	0.002	0.012	0.016	0.006	0.965	0.894	0.889	0.566	0.844	0.810	1.000	0.997	0.933	0.773	0.983	0.917	0.776	0.834	0.629	0.828	0.633	1.000	1.000	0.902	0.809	0.927
netscience	0.087	0.143	0.111	0.175	0.118	0.145	0.082	0.056	0.107	0.167	0.092	0.450	0.449	0.474	0.642	0.582	0.307	0.081	0.256	0.622	0.574	0.752	0.602	0.553	0.605	0.622	0.645	0.438	0.309	0.501	0.729	0.690	0.689
dwt_419	0.022	0.069	0.035	0.180	0.048	0.033	0.026	0.016	0.028	0.044	0.022	0.555	0.489	0.534	0.367	0.514	0.486	0.439	0.563	0.526	0.495	0.591	0.723	0.632	0.720	0.474	0.693	0.626	0.759	0.735	0.694	0.727	
cluster	0.186	0.210	0.187	0.225	0.199	0.239	0.230	0.182	0.212	0.208	0.198	0.065	0.040	0.068	0.171	0.107	0.053	0.001	0.041	0.199	0.118	0.074	0.378	0.331	0.377	0.368	0.378	0.346	0.288	0.378	0.372	0.379	
qb882	0.066	0.117	0.080	0.244	0.092	0.095	0.048	0.095	0.098	0.066	0.343	0.357	0.359	0.483	0.497	0.128	0.065	0.245	0.443	0.432	0.585	0.505	0.489	0.516	0.416	0.422	0.560	0.259	0.288	0.413	0.606	0.579	0.622
price_1000	0.137	0.261	0.127	0.198	0.188	0.279	0.112	0.092	0.156	0.179	0.125	0.085	0.128	0.155	0.559	0.457	0.037	0.007	0.077	0.312	0.309	0.677	0.452	0.416	0.482	0.558	0.596	0.345	0.205	0.206	0.624	0.601	0.563
dwt_1005	0.026	0.045	0.030	0.225	0.033	0.030	0.020	0.01	0.021	0.038	0.041	0.028	0.356	0.432	0.303	0.286	0.301	0.296	0.345	0.456	0.368	0.549	0.516	0.559	0.498	0.406	0.483	0.469	0.506	0.562	0.642	0.512	0.616
cage8	0.142	0.174	0.139	0.227	0.151	0.159	0.156	0.118	0.186	0.169	0.143	0.119	0.091	0.110	0.299	0.184	0.090	0.008	0.087	0.321	0.234	0.303	0.225	0.196	0.223	0.318	0.262	0.178	0.117	0.199	0.425	0.375	0.327
btree9	0.138	0.314	0.144	0.199	0.159	0.301	0.123	0.155	0.150	0.161	0.151	0.092	0.097	0.047	0.427	0.709	0.312	0.005	0.002	0.21	0.204	0.448	0.375	0.251	0.500	0.743	0.560	0.115	0.109	0.085	0.345	0.588	0.802
bscs109	0.022	0.018	0.023	0.152	0.032	0.041	0.017	0.015	0.020	0.026	0.014	0.443	0.410	0.471	0.425	0.234	0.383	0.417	0.478	0.484	0.452	0.482	0.842	0.866	0.826	0.475	0.767	0.683	0.891	0.968	0.881	0.788	0.856
1138_bus	0.082	0.156	0.088	0.218	0.103	0.133	0.071	0.062	0.129	0.111	0.085	0.201	0.284	0.362	0.579	0.558	0.108	0.029	0.098	0.450	0.502	0.717	0.360	0.386	0.427	0.464	0.515	0.232	0.200	0.261	0.566	0.536	0.590
qh1484	0.075	0.130	0.069	0.300	0.068	0.103	0.056	0.035	0.100	0.089	0.056	0.296	0.293	0.337	0.429	0.439	0.098	0.060	0.250	0.369	0.401	0.552	0.441	0.413	0.461	0.402	0.529	0.255	0.276	0.405	0.559	0.543	0.608
clusterHier	0.164	0.161	0.160	0.222	0.188	0.190	0.160	0.138	0.175	0.205	0.163	0.091	0.080	0.089	0.210	0.133	0.040	0.002	0.042	0.210	0.142	0.217	0.460	0.390	0.460	0.491	0.496	0.320	0.261	0.337	0.497	0.499	0.489
bcspw07	0.065	0.125	0.086	0.220	0.091	0.090	0.060	0.046	0.095	0.100	0.051	0.185	0.270	0.314	0.503	0.489	0.081	0.035	0.111	0.387	0.440	0.677	0.343	0.397	0.431	0.419	0.490	0.229	0.202	0.273	0.525	0.507	0.578
fidapex6	0.031	0.051	0.041	0.109	0.067	0.083	0.025	0.017	0.067	0.041	0.022	0.721	0.512	0.632	0.400	0.232	0.323	0.435	0.719	0.677	0.586	0.795	0.609	0.729	0.534	0.642	0.332	0.702	0.858	0.725	0.738	0.850	
block_2000	0.163	0.169	0.164	0.222	0.188	0.181	0.165	0.140	0.187	0.212	0.162	0.029	0.031	0.058	0.167	0.053	0.022	0.000	0.017	0.108	0.063	0.156	0.278	0.231	0.276	0.332	0.296	0.141	0.120	0.160	0.360	0.309	0.330
sierpinsk33	0.083	0.109	0.079	0.309	0.092	0.087	0.065	0.063	0.080	0.085	0.082	0.488	0.461	0.509	0.505	0.532	0.211	0.307	0.582	0.550	0.542	0.637	0.483	0.475	0.510	0.411	0.495	0.308	0.428	0.539	0.558	0.525	0.599
lp_ship04l	0.173	0.273	0.174	0.236	0.211	0.177	0.184	0.139	0.174	0.201	0.158	0.008	0.005	0.008	0.044	0.051	0.006	0.000	0.003	0.046	0.041	0.056	0.261	0.214	0.311	0.193	0.319	0.181	0.223	0.303	0.307	0.341	0.455
data	0.044	0.057	0.027	0.357	0.054	0.026	0.022	0.010	0.102	0.104	0.020	0.594	0.328	0.508	0.269	0.451	0.407	0.578	0.685	0.593	0.515	0.731	0.331	0.379	0.358	0.304	0.320	0.229	0.205	0.206	0.525	0.457	0.382
3elt	0.049	0.105	0.066	0.301	0.062	0.064	0.053	0.038	0.069	0.095	0.045	0.564	0.450	0.662	0.202	0.538	0.307	0.412	0.510	0.620	0.573	0.795	0.554	0.464	0.627	0.226	0.534	0.351	0.469	0.512	0.613	0.593	0.723
crack	0.012	0.026	0.010	0.334	0.017	0.025	0.005	0.004	0.017	0.010	0.006	0.747	0.737	0.865	0.184	0.526	0.908	0.681	0.782	0.727	0.795	0.672	0.754	0.704	0.816	0.180	0.729	0.180	0.727	0.744	0.780	0.880	
fe_4elt	0.052	0.091	0.055	0.420	0.064	0.071	0.049	0.042	0.093	0.070	0.071	0.331	0.227	0.393	0.171	0.398	0.434	0.566	0.614	0.304	0.270	0.395	0.375	0.281	0.422	0.160	0.423	0.249	0.329	0.430	0.461	0.598	0.568
fidapex11	0.070	0.120	0.062	0.186	0.072	0.083	0.029	0.021	0.132	0.055	0.038	0.193	0.171	0.184	0.203	0.195	0.160	0.176	0.205	0.205	0.199	0.274	0.345	0.229	0.342	0.291	0.327	0.350	0.367	0.331	0.346	0.439	
bscs32	0.046	0.085	0.072	0.342	0.065	0.061	0.049	0.033	0.078	0.050	0.028	0.260	0.231	0.277	0.240	0.263	0.165	0.154	0.216	0.278	0.266	0.394	0.345	0.221	0.307	0.221	0.307	0.227	0.271	0.325	-	0.371	0.441
bscs32t	0.099	0.179	0.100	0.349	0.166	0.133	0.080	0.046	0.204	0.																							

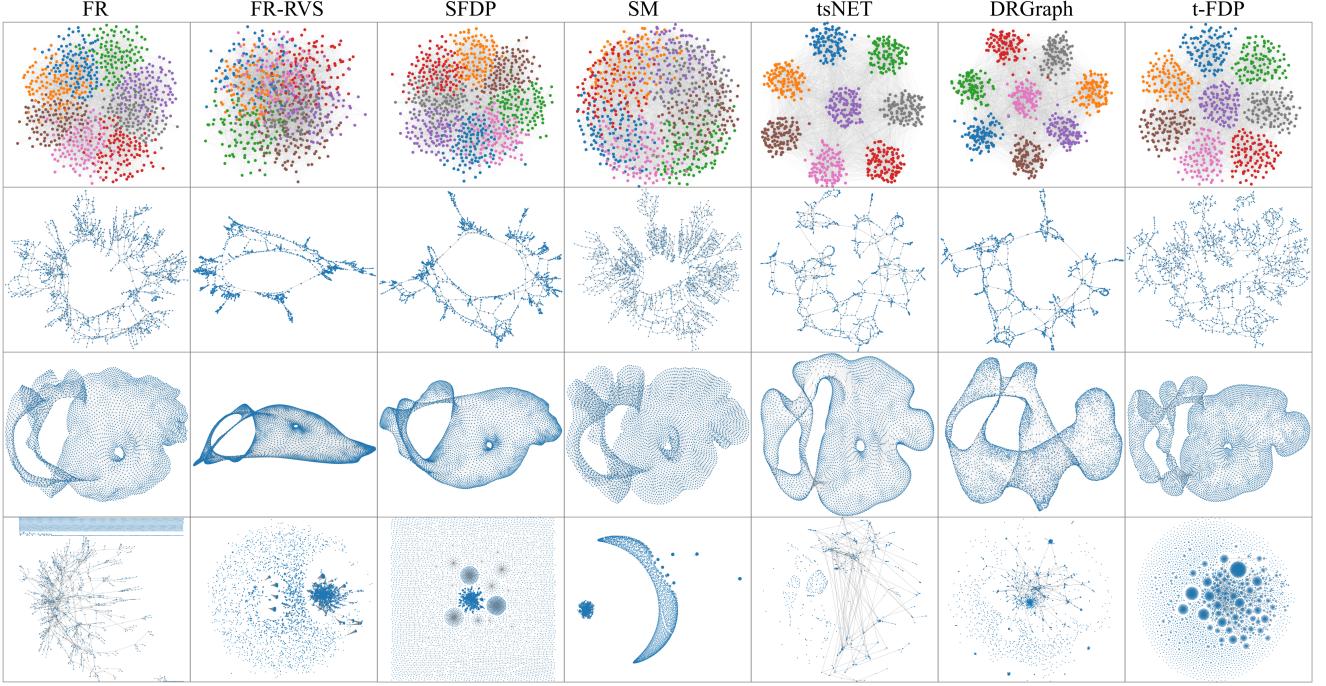


Fig. 13. Layouts by seven methods for the four data sets: *cluster* (top row), *bcsppwr07* (second row), *3elt* (third row), and *eva* (bottom row). t-FDP shows a good ability to highlight clusters, at the same time a good mixture between local and global structures for *bcsppwr07* and a good unfolding for *3elt*.

performs only slightly worse than t-FDP for the stress error, it performs the worst for NP1 and NP2. Similarly, PMDS is the second-worst for stress error, NP1, and NP2. After carefully examining the layouts generated by Maxent and PMDS, we found that neither of them can efficiently handle graphs whose numbers of edges are larger than the numbers of nodes. On the other hand, it is not surprising that SM is the best and LinLog is the worst for the stress error, since SM is designed for preserving distances while LinLog is for revealing clusters.

The boxplots in Fig. 12 summarize the EC and MA scores over 37 graphs that can be handled by all methods. Fig. 12(a) shows that t-FDP performs similarly to the other methods w.r.t. EC, while PMDS and Maxent are slightly worse. Fig. 12(b) shows that Maxent performs the best in MA, followed by SM and t-FDP, while PMDS is the worst. From these results, we conclude that the readability of our t-FDP produced layouts is comparable and even superior to the existing methods.

Note that FR-RVS is significantly worse than FR w.r.t. SE and NP2, which contradicts the results from Gove [20]. After carefully checking our results, we found that the differences are due to initialization, where Gove [20] used an initialization of randomly distributed nodes on a uniformly spaced disc. By using similar random initializations, FR-RVS and FR perform similarly but the resulting layouts are worse than the ones generated by PMDS initialization. The full evaluation can be found in the supplemental material, where we also include the visual layouts generated by FR-RVS and t-FDP-RVS for comparison.

Visual Results. Fig. 13 shows the visual results of the six methods applied to four data sets: *cluster*, *bcsppwr07*, *3elt* and *eva*. We can see that t-FDP can characterize clusters as tsNET and DRGraph for the clustered graphs (see the top row),

clearly revealing the global structures as FR and SM while maintaining the neighborhood structures (see the second row), a good unfolding for mesh-like structures as FR (see the third row) and faithfully depicting multi-component structures (see the last row). Although SM maintains a global structure by preserving pairwise distances, local structure preservation and clustering ability are the worst. FR and SFDP behave similarly to SM. tsNET and DRGraph seem to form clusters even when there are no cluster structures in the original graph, they also have difficulties in maintaining the global structure. For example, they squeeze the branches of the tree-like graph *bcsppwr07* into a few small sub-clusters, which are overlapping (see the second row of Fig. 13). Note that we did not employ any packing algorithm to arrange multiple components, whereas FR and SFDP tightly pack them in layout space.

Runtime Performance. Zhu et al. [10] show in their experiments that only SFDP, PMDS, and DRGraph can handle graphs with millions of nodes. Besides these methods, we include Maxent for making a comprehensive comparison with our t-FDP with regard to runtime performance. For all methods, the runtime includes only the layout time without data processing steps such as data loading and layout initialization. Fig. 10 reports the runtime for each graph.

In contrast to what was reported by Zhu et al. [10], our implementation of PMDS is the fastest, because of its linear computational complexity. For small graphs with less than 1K nodes, all methods can be processed in less than a second, while the performances vary significantly for larger graphs. SFDP, Maxent, and DRGraph have similar performances for graphs with more than 100K nodes, because of their similar computational complexity $O(n \log n)$ for computing repulsive forces. The GPU version of the ibFFT based t-FDP is two orders of magnitude faster than these methods due

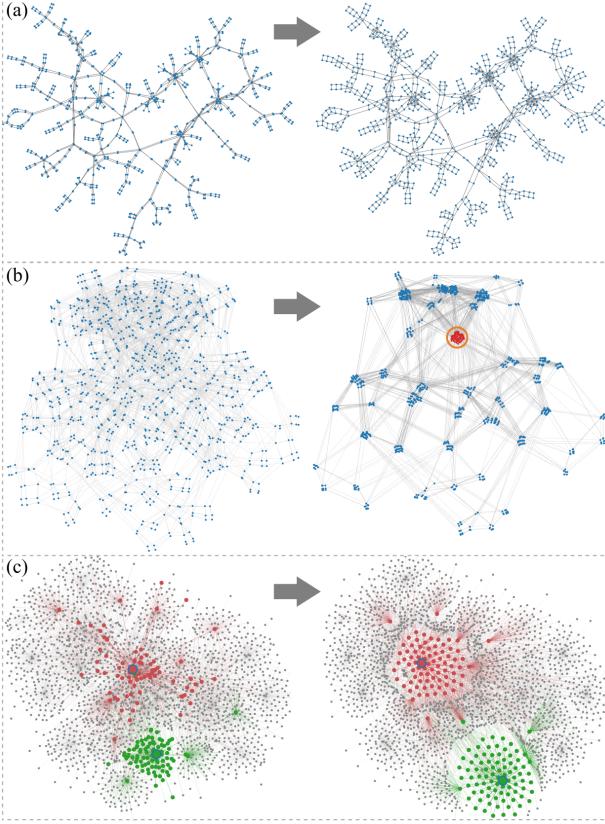


Fig. 14. Refining t-FDP graph layouts by applying a large repulsive force (a), and a repulsive force with shorter range (b), and locally changing attractive and repulsive forces (c). These result in uniform distributed local neighborhoods (a), major clusters (b), and fisheye views (c).

to its ibFFT approximation. For example, the GPU version of t-FDP model takes 30s for the *com-LiveJournal* graph (4 millions nodes and 35 million edges), while DRGraph, SFDP and Maxent require 2958s, 5745s and 11954s, respectively.

Based on the results of the five metrics and runtime performances, we can conclude that our t-FDP model is able to generate high-quality layouts for most graphs while being extremely fast to compute.

5 EXTENSIONS

Since our t-FDP model retains the flexibility and simplicity of traditional FDP, it inherits all possible extensions of these models, such as multi-level layout methods [7] or constrained layouts [47]. Besides that, our t-FDP model can be further extended for better supporting the interactive exploration of graphs by globally and locally adjusting the repulsive t-force. Since our model is fast enough, these refinement extensions can be done with real-time interaction.

Global Refinement. Users often want to explore different graph structures, such as detailed local neighborhoods or skeleton-like structures. Taking a t-FDP layout as initialization, we can achieve such visualizations by re-applying t-FDP with repulsive t-forces of different values. For example, using a large repulsive t-force will distribute nearby nodes evenly, resulting in detailed local neighborhoods. On the other hand, a small repulsive t-force will move connected nodes closer together and reveal major structures.

Fig. 14(a) shows an example of applying a large repulsive force to the layout of the graph *qh882* (on the left), distributing the nodes at the tip of the branches evenly (right) and improving the NP1 score from 0.585 to 0.643. In both results, the warping effect [48] –that lets nodes in the periphery tend to be closer– is greatly alleviated while unfolding all parts of the graph. The example in Fig. 14(b) is obtained in a reverse way: applying a repulsive force with a shorter range (larger γ) allows to display a clear skeleton structure of the graph *cage8*.

Local Refinement. During exploration, one major task is to find and examine the neighborhood of certain nodes. Although our t-FDP performs well in neighborhood preservation, it cannot ensure that all neighborhoods are always well preserved. The left in Fig. 14(c) shows an example of *lp_ship04l* graph, where some neighboring nodes (see the red and green ones) cannot be placed properly because of their own local clusters. To alleviate this issue, we enhance the attractive forces between the focal nodes and their neighbors for pulling them together, while exerting repulsive forces to highlight them. Meanwhile, we exert large repulsive forces between other nodes for compressing the surrounding area. In doing so, a fisheye-like visualization is generated. The right in Fig. 14(c) shows the result, where most neighborhoods of the red and green nodes are clearly revealed. Note that a few nodes are still not pulled together because of their own local clusters.

6 CONCLUSION AND FUTURE WORK

We present t-FDP, a novel FDP model for graph placement. It is based on the observation that existing FDP models cannot properly capture local neighborhoods, especially due to the large contact forces when two nodes overlap.

Therefore, we devise a new short-range force based on the t-distribution: t-force. It has a defined upper bound, behaves similar to existing power functions based attractive forces at long-range and repulsive forces at short-range. Furthermore, we adapt the FFT based approximation strategy used for t-SNE to accelerate the computation of the repulsive force. We quantitatively compare our t-FDP model with different state-of-the-art layout methods, showing that the t-FDP based on the FFT approximation outperforms them in most cases and is one magnitude faster on CPU and two orders faster using a GPU. Lastly, we demonstrate the usefulness of t-FDP in exploring different graph structures.

Our approach still has certain limitations, which we would like to address in the future: First, t-FDP performs slightly worse than stress models in distance preservation, although it shows better results than tsNET and DRGraph. We therefore plan to explore the possibility of improving its long-range forces. Second, t-FDP replaces a single parameter for balancing attractive and repulsive forces acting on each node by three parameters α , β and γ , which might result in additional complications for users. On the other hand, these parameters provide freedom for adapting the method to different tasks [3], therefore, we like to explore automated parameter tuning methods. Finally, we plan to investigate other possible forms of short-range forces to further improve layout quality and explore their applications in dimensionality reduction.

ACKNOWLEDGMENTS

The authors like to thank the anonymous reviewers for their valuable input, this work was supported by the grants of the National Key Research & Development Plan of China (2019YFB1704201) and NSFC (62132017, 62141217), as well as by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2117 - 422037984.

REFERENCES

- [1] T. M. J. Fruchterman and E. M. Reingold, "Graph drawing by force-directed placement," *Software: Practice and Experience*, vol. 21, no. 11, pp. 1129–1164, 1991. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.4380211102>
- [2] 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*, vol. 36, no. 3, pp. 283–294, 2017. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13187>
- [3] B. Lee, C. Plaisant, C. S. Parr, J.-D. Fekete, and N. Henry, "Task taxonomy for graph visualization," in *Proceedings of the 2006 AVI workshop on BEyond time and errors: novel evaluation methods for information visualization*, 2006, pp. 1–5.
- [4] S. G. Kobourov, *Force-Directed Drawing Algorithms*, R. Tamassia, Ed. Oxford: Chapman and Hall/CRC, 2013.
- [5] T. Biedl and G. Kant, "A better heuristic for orthogonal graph drawings," *Computational Geometry*, vol. 9, no. 3, pp. 159–180, 1998. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0925772197000266>
- [6] E. R. Gansner, Y. Hu, and S. North, "A maxent-stress model for graph layout," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 6, pp. 927–940, June 2013.
- [7] Y. Hu, "Efficient, high-quality force-directed graph drawing," *Mathematica journal*, vol. 10, no. 1, pp. 37–71, 2005.
- [8] M. Bastian, S. Heymann, and M. Jacomy, "Gephi: an open source software for exploring and manipulating networks," in *Proceedings of the international AAAI conference on web and social media*, vol. 3, no. 1, 2009, pp. 361–362.
- [9] M. Bostock, V. Ogievetsky, and J. Heer, "D³ data-driven documents," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2301–2309, Dec 2011.
- [10] 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*, vol. 27, no. 2, pp. 1666–1676, Feb 2021.
- [11] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008. [Online]. Available: <http://jmlr.org/papers/v9/vandermaaten08a.html>
- [12] 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*, vol. 16, no. 3, pp. 243–245, Mar 2019. [Online]. Available: <https://doi.org/10.1038/s41592-018-0308-4>
- [13] W. T. Tutte, "How to draw a graph," *Proceedings of the London Mathematical Society*, vol. s3-13, no. 1, pp. 743–767, 1963. [Online]. Available: <https://londmathsoc.onlinelibrary.wiley.com/doi/abs/10.1112/plms/s3-13.1.743>
- [14] Y. Chen, Z. Guan, R. Zhang, X. Du, and Y. Wang, "A survey on visualization approaches for exploring association relationships in graph data," *Journal of Visualization*, vol. 22, no. 3, pp. 625–639, 2019.
- [15] A. Noack, "Energy models for graph clustering," *Journal of Graph Algorithms and Applications*, vol. 11, no. 2, pp. 453–480, 2007.
- [16] 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*, vol. 9, no. 6, pp. 1–12, 06 2014. [Online]. Available: <https://doi.org/10.1371/journal.pone.0098679>
- [17] P. Kumar and K. Zhang, "Visualization of clustered directed acyclic graphs with node interleaving," in *Proceedings of the 2009 ACM Symposium on Applied Computing*. New York, NY, USA: Association for Computing Machinery, 2009, pp. 1800–1805. [Online]. Available: <https://doi.org/10.1145/1529282.1529685>
- [18] J. Barnes and P. Hut, "A hierarchical o(n log n) force-calculation algorithm," *Nature*, vol. 324, no. 6096, pp. 446–449, Dec 1986. [Online]. Available: <https://doi.org/10.1038/324446a0>
- [19] L. F. Greengard, "The rapid evaluation of potential fields in particle systems," Ph.D. dissertation, USA, 1987, aAI8727216. [Online]. Available: <https://dl.acm.org/doi/10.5555/913529>
- [20] R. Gove, "A random sampling o(n) force-calculation algorithm for graph layouts," *Computer Graphics Forum*, vol. 38, no. 3, pp. 739–751, 2019. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13724>
- [21] T. Kamada and S. Kawai, "An algorithm for drawing general undirected graphs," *Inf. Process. Lett.*, vol. 31, no. 1, p. 7–15, apr 1989. [Online]. Available: [https://doi.org/10.1016/0020-0190\(89\)90102-6](https://doi.org/10.1016/0020-0190(89)90102-6)
- [22] Y. Koren and A. Civril, "The binary stress model for graph drawing," in *International Symposium on Graph Drawing*. Springer, 2008, pp. 193–205.
- [23] L. Chen and A. Buja, "Local multidimensional scaling for nonlinear dimension reduction, graph drawing, and proximity analysis," *Journal of the American Statistical Association*, vol. 104, no. 485, pp. 209–219, Mar 2009. [Online]. Available: <https://doi.org/10.1198/jasa.2009.0111>
- [24] Y. Wang, Y. Wang, Y. Sun, L. Zhu, K. Lu, C.-W. Fu, M. Sedlmair, O. Deussen, and B. Chen, "Revisiting stress majorization as a unified framework for interactive constrained graph visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 489–499, 2018.
- [25] K. Pearson, "X. Contributions to the mathematical theory of evolution.—II. Skew variation in homogeneous material," *Philosophical Transactions of the Royal Society of London.(A.)*, vol. 186, pp. 343–414, 1895.
- [26] J. Tang, J. Liu, M. Zhang, and Q. Mei, "Visualizing large-scale and high-dimensional data," in *Proceedings of the 25th International Conference on World Wide Web*. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2016, pp. 287–297. [Online]. Available: <https://doi.org/10.1145/2872427.2883041>
- [27] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.
- [28] E. Amid and M. K. Warmuth, "Trimap: Large-scale dimensionality reduction using triplets," *arXiv preprint arXiv:1910.00204*, 2019. [Online]. Available: <https://doi.org/10.48550/arXiv.1910.00204>
- [29] L. Van Der Maaten, "Accelerating t-SNE using tree-based algorithms," *Journal of Machine Learning Research*, vol. 15, no. 1, p. 3221–3245, jan 2014. [Online]. Available: <https://dl.acm.org/doi/10.5555/2627435.2697068>
- [30] N. Pezzotti, J. Thijssen, A. Mordvintsev, T. Höllt, B. Van Lew, B. P. Lelieveldt, E. Eisemann, and A. Vilanova, "GPGPU linear complexity t-SNE optimization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 1172–1181, Jan 2020.
- [31] J. Douglas Carroll and P. Arabie, *Multidimensional Scaling*, ser. Handbook of Perception and Cognition (Second Edition). San Diego: Academic Press, Jan 1998. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/B9780120999750500051>
- [32] J. Ellson, E. Gansner, L. Koutsofios, S. C. North, and G. Woodhull, "Graphviz—open source graph drawing tools," in *Graph Drawing*, P. Mutzel, M. Jünger, and S. Leipert, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 483–484.
- [33] C. Walshaw, "A multilevel algorithm for force-directed graph drawing," in *Proceedings of the 8th International Symposium on Graph Drawing*, ser. GD '00. Berlin, Heidelberg: Springer-Verlag, 2000, p. 171–182. [Online]. Available: <https://dl.acm.org/doi/10.5555/647552.729414>
- [34] S. Hachul and M. Jünger, "Drawing large graphs with a potential-field-based multilevel algorithm," in *International Symposium on Graph Drawing*. Springer, 2004, pp. 285–295.
- [35] J. F. Epperson, "On the runge example," *The American Mathematical Monthly*, vol. 94, no. 4, pp. 329–341, 2022/03/11/1987, full publication date: Apr., 1987. [Online]. Available: <https://doi.org/10.2307/2323093>
- [36] S. K. Lam, A. Pitrou, and S. Seibert, "Numba: A llvm-based python jit compiler," in *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, 2015, pp. 1–6.
- [37] T. A. Davis and Y. Hu, "The university of florida sparse matrix collection," *ACM Trans. Math. Softw.*, vol. 38, no. 1, dec 2011. [Online]. Available: <https://doi.org/10.1145/2049662.2049663>

- [38] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection," <http://snap.stanford.edu/data/>, Jun. 2014. [Online]. Available: <http://snap.stanford.edu/>
- [39] T. P. Peixoto, "The graph-tool python library," 2014. [Online]. Available: http://figshare.com/articles/graph_tool/1164194
- [40] U. Brandes and C. Pich, "Eigensolver methods for progressive multidimensional scaling of large data," in *Graph Drawing*, M. Kaufmann and D. Wagner, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 42–53.
- [41] M. Khoury, Y. Hu, S. Krishnan, and C. Scheidegger, "Drawing large graphs by low-rank stress majorization," *Computer Graphics Forum*, vol. 31, no. 3pt1, pp. 975–984, 2012. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2012.03090.x>
- [42] M. Chimani, C. Gutwenger, M. Jünger, G. W. Klau, K. Klein, and P. Mutzel, "The open graph drawing framework (ogdf)." *Handbook of graph drawing and visualization*, vol. 2011, pp. 543–569, 2013.
- [43] 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 *30th International Symposium on Computer Architecture and High Performance Computing*, Sep. 2018, pp. 330–338.
- [44] D. Kobak and G. C. Linderman, "Initialization is critical for preserving global data structure in both t-sne and umap," *Nature Biotechnology*, vol. 39, no. 2, pp. 156–157, Feb 2021. [Online]. Available: <https://doi.org/10.1038/s41587-020-00809-z>
- [45] H. C. Purchase, "Metrics for graph drawing aesthetics," *Journal of Visual Languages & Computing*, vol. 13, no. 5, pp. 501–516, Oct 2002. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1045926X02902326>
- [46] J. Schneider and S. Kirkpatrick, *Stochastic Optimization*. Springer Science & Business Media, 2006.
- [47] T. Dwyer, "Scalable, versatile and simple constrained graph layout," *Comput. Graph. Forum*, vol. 28, no. 3, pp. 991–998, 2009. [Online]. Available: <https://doi.org/10.1111/j.1467-8659.2009.01449.x>
- [48] Y. Hu and Y. Koren, "Extending the spring-electrical model to overcome warping effects," in *2009 IEEE Pacific Visualization Symposium*, April 2009, pp. 129–136.



Jian Zhang received his PhD degree in Applied Mathematics from the University of Minnesota in 2005. After a postdoc at Pennsylvania State University, he is now a professor in the Computer Network Information Center, Chinese Academy of Sciences (CAS). His current research interests include scientific computing and scientific visualization.



Fan Zhang received the B.Eng. and Ph.D. degrees from the School of Computer Science and Technology, Shandong University, China, in 2009 and 2015, respectively. He is currently an associate professor at the School of Computer Science and Technology, Shandong Technology and Business University, China. His research interests include computer graphics, computer vision, and artificial intelligence.



Rui Ban is a senior engineer at Intelligent Network Design Institute of China Information Technology Designing Consulting Institute Co., Ltd. His research interests include big data visualization and visual analytics.



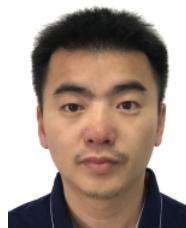
Fahai Zhong is a third-year Master student in the School of Computer Science and Technology, at Shandong University. His research interests include graph visualization and dimensional reduction.



Oliver Deussen graduated at Karlsruhe Institute of Technology and is professor at University of Konstanz (Germany) and visiting professor at the Chinese Academy of Science in Shenzhen. He served as Co-Editor in Chief of Computer Graphics Forum and was President of the Eurographics Association. His areas of interest are modeling and rendering of complex biological systems, non-photorealistic rendering as well as Information Visualization.



Mingliang Xue is a fifth-year Ph.D. student in the School of Computer Science and Technology, Shandong University. His research interests include graph visualization and dimensional reduction.



Yunhai Wang is professor in School of Computer Science and Technology at Shandong University. He serves as the associate editor of Computer Graphics Forum. His interests include scientific visualization, information visualization and computer graphics.