Taurus: Towards a Unified Force Representation and Universal Solver for Graph Layout

Mingliang Xue, Zhi Wang, Fahai Zhong, Yong Wang
Oliver Deussen, and Yunhai Wang

Abstract—Over the past few decades, a large number of graph layout techniques have been proposed for visualizing graphs from various domains. In this paper, we present a general framework, Taurus, for unifying popular techniques such as the spring-electrical model, stress model, and maxent-stress model. It is based on a unified force representation, which formulates most existing techniques as a combination of quotient-based forces that combine power functions of graph-theoretical and Euclidean distances. This representation enables us to compare the strengths and weaknesses of existing techniques, while facilitating the development of new methods. Based on this, we propose a new balanced stress model (BSM) that is able to layout graphs in superior quality. In addition, we introduce a universal augmented stochastic gradient descent (SGD) optimizer that efficiently finds proper solutions for all layout techniques. To demonstrate the power of our framework, we conduct a comprehensive evaluation of existing techniques on a large number of synthetic and real graphs. We release an open-source package, which facilitates easy comparison of different graph layout methods for any graph input as well as effectively creating customized graph layout techniques.

Index Terms—Graph Layout, Gradient Descent, Framework

1 Introduction

Graphs are commonly used for modeling complex data in many domains such as social media, finance and biology. The most commonly used graph visualization technique, node-link diagrams, depict nodes as points in a plane and edges as lines connecting these points. In past decades, various graph layout methods [25, 36] have been developed for producing aesthetically-pleasing drawings, while maintaining the underlying graph structures.

Rather than directly optimizing aesthetic criteria [34] (e.g., even node distribution and minimal edge crossing), most methods simulate one of two kinds of physical systems as a basis for layouting graphs: the spring-electrical model or the stress model. The spring-electrical model [8, 10] regards edges as springs that use attractive forces to pull connected nodes close to each other, at the same time treating nodes as electrically-charged particles that repel each other with repulsive forces. Based on this model, many variants of force-directed placement (FDP) algorithms have been developed for better revealing different structures and features of graphs. For example, FM^3 [18] and SFDP [20] use a multilevel scheme for overcoming local minima, the extended models of LinLog [31] and ForceAtlas2 [22] allow to better reveal clusters and local structures, respectively. While the spring-electrical model produces good layouts for many graphs, it does not encode the target (data-space) edge lengths between every pair of nodes.

This is the focus of stress models [14, 23, 40], which assume a spring between each pair of nodes with an ideal length equal to the graph-theoretical distance among the nodes. By minimizing the stress energy of the spring system, a layout is obtained. For efficiently solving such models, which involve considerably more interactions between the nodes, a few optimization strategies have been incorporated, such as stress majorization [14], and stochastic gradient descent (SGD) [42].

• Mingliang Xue, Zhi Wang, Fahai Zhong, and Yunhai Wang are with the Department of Computer Science, Shandong University, China. E-mail: {xml95007, wangzizi2020, zhongfahai, cloudseawang}@gmail.com
• Yong Wang is with School of Computing and Information Systems, Singapore Management University, Singapore. E-mail: yongwang@smu.edu.sg
• 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

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

To alleviate the involved computational costs, sparse stress models [13, 27, 32] have been proposed, which only impose springs for a subset of node pairs.

Because of the divergent mechanisms, these models have different characteristics when creating graph layouts. For example, FDP performs better in preserving neighborhood structures for many graphs, while the stress model tends to maintain the overall structures, especially for mesh-like graphs. However, it is still unclear why the models have such differences and how they are connected conceptually. Moreover, it is difficult to make a fair quantitative comparison because different optimization strategies are used. This not only hinders researchers to develop new methods but also poses a challenge for practitioners to choose a proper method for visualizing their graphs.

In this paper, we present a general framework, we call Taurus, **T**owards **a** **U**nified force **R**epresentation and **U**niversal **S**olver for graph layout, that offers a unified view for understanding and comparing most of the popular graph layout algorithms. It relies on two novel components: a unified force representation and a universal solver. The uniform force representation allows us to show that all existing methods can be formulated as a combination of quotient-based forces, using a quotient between power functions of graph-theoretical and Euclidian distances. This unified representation enables us to compare the strengths and weaknesses of different methods. The universal solver combines the advantage of SGD [30] in escaping local minima and the effectiveness of the Barnes & Hut approximation [3] in reducing computational cost, which allows us to solve different existing layout methods with the same optimizer.

Moreover, our framework can also be used as a general platform for developing new graph layout methods. In particular, we propose a balanced stress model, which combines the advantages of spring-electrical and stress models. Specifically, it exerts attractive and repulsive forces to all node pairs, where the attractive force is reciprocal and the repulsive force is proportional to the graph-theoretical distances. In doing so, the model avoids extremely large repulsive and attractive forces for nearby nodes, while pulling neighboring nodes close to each other.

We implement Taurus as a graph visualization package in C++, which allows users to define their own attractive and repulsive forces. To demonstrate its effectiveness, we comprehensively evaluate it by comparing various spring and stress layout methods on a large number of synthesized graphs with different structures such as lattices, trees and clusters. The evaluation includes two parts: verifying whether Taurus can produce similar results to the original implementations of existing methods, and examining how different methods behave on graphs with different characteristics. The results show that our solver enables all methods to perform as well as or even better than the original implementations, while our proposed balanced stress model makes a good trade-off in distance preservation and maintaining neighborhoods as well as cluster structures. In addition, we show that our Taurus allows users to flexibly customize the graph layout methods for meeting specific requirements.

The main contributions of this paper can be summarized as follows:

- We propose a general framework for graph visualization based on a novel quotient based force representation and an augmented SGD optimizer, which offers a unified view for understanding and comparing existing graph layout methods;

- We present a new graph layout method based on our framework and conduct a systematic analysis and extensive evaluation for our framework on different graph datasets through quantitative comparisons; and

- We release a library with the proposed general framework that enables rapid implementation and design of graph layout methods for any graph input.

2 Related Work

Related works can be categorized into three parts: graph layout methods, graph layout solvers and graph layout packages.

2.1 Graph Layout

Various graph layout methods have been proposed to visualize network data as node-link diagrams. Among them, the most common methods often use virtual physical models to represent the relationships between objects. By referring to the taxonomy by Gansner et al. [13], we classify such methods into three types: spring-electrical models, stress models and hybrid models.

Spring-electrical models [8, 10] regard nodes as electrically-charged particles that push nodes away from each other and edges as springs that pull nodes close to each other, often referred to as repulsive and attractive forces. A graph layout result is achieved when attractive and repulsive forces strike a balance. For a complete review of the graph layout methods developed from this model, please refer to Kobourov [28] and Gouvêa et al. [16]. Here, we briefly review some widely used models. Hu et al. [20] improve the repulsive force designed by Fruchterman and Reingold [10] and use a repulsive force that decays rapidly, avoiding edge-length distortion at the periphery of a layout. Noack et al. [31] introduce the LinLog model that employed a constant attractive force and set the repulsive force to the inverse of the distance. As a result, this model can generate graph layouts with clearly-separated node clusters. Kermarrec and Moin [26] further extend the LinLog model for revealing cluster structures at different levels. Inspired by these studies, the attractive and repulsive force of ForceAtlas2 [22] were designed to be proportional and inversely proportional to the distance between nodes, obtaining graph layouts with a good preservation of local structures and cluster separation.

Stress models [14, 23] also use a spring analogy but assume that there are springs connecting every pair of nodes in the graph. Spring forces are defined to create a layout with distances of nodes as close as possible to the graph-theoretical distances. Many variants of the stress model aim to improve its efficiency through sparse approximations. For example, progressive multidimensional scaling [5] and low-rank stress majorization [27] have been used to approximate the shortest path distances of all node pairs in a graph. The sparse stress model [32] speeds up the stress model by aggregating the terms of the objective function. Wang et al. [40] improved the stress model by imposing constraints on edge vectors and edge lengths, further enhancing the expressiveness of the stress model.

Hybrid models combine both models for overcoming their drawbacks. For example, Hu and Koren [21] resolve the warping effect of spring-electrical models by integrating attractive forces into the stress model. To reduce the cost for computing graph-theoretical distances, the Maxent-stress model (Maxent) [13] imposes stress constraints on pairs of neighboring nodes and entropy-based constraints on the remaining node pairs, the latter ones can be regarded as repulsive forces between all node pairs.

Noack [31] shows that energy-based layout methods like LinLog can be formulated as force representations. Similarly, Gansner et al. [13] represent the repulsive force as an entropy term and incorporate it into the stress-based energy model. However, there is still a lack of an inherent representation for unifying existing layout methods. In this work, we demonstrate that almost all methods from spring-electrical and stress models can be formulated as a combination of our proposed quotient-based forces. Moreover, we show that this unified view not only facilitates the understanding and comparison of different methods but also allows the development of new methods.

2.2 Graph Layout Solvers

Most graph layout methods need an optimization solver to create desirable drawings. Solving a spring-electrical model has a time complexity of $O(n^2)$ at each iteration, where n is the number of nodes in the graph. To improve the computational efficiency of such models, several multilevel methods [3, 11, 12, 18–20, 39] have been proposed. Among them, the Barnes-Hut (BH) approximation [3] is the most commonly-used acceleration method. It uses hyper nodes to approximate repulsive forces, resulting in a time complexity of $O(n \log n)$. The method has been used by different spring-electric model algorithms, such as [20, 22]. Another method is to use random vertex sampling (RVS) [17] to accelerate the computation of repulsive forces. This method generates layouts similar to Barnes-Hut.

There are also many algorithms to optimize solutions for the stress model. The earliest stress model [23] employs gradient descent to find the optimal graph layout; however, it is often trapped into a local minimum. Gansner et al. [14] adapt stress majorization to the stress model, which is rooted in solving multidimensional scaling. Ensuring a monotonic decrease of the stress, the method has advantages over the original implementation. Recently, stochastic gradient descent (SGD), a powerful optimization solver widely used in machine learning, has also been applied to graph drawing [42]. It converges fast and achieves layouts with a lower stress error. Ahmed et al. [1] further proposed a SGD-based graph drawing approach $(SGD)^2$ that can handle multiple readability criteria of graph drawing simultaneously. We propose an augmented SGD solver for finding optimal layouts at minimal computation speed.

2.3 Graph Layout Packages

A number of open-source packages facilitate an easy implementation of different graph layout techniques. For example, Graphviz [15], Tulip [2] and OGDF [6] are C++ libraries that implement customized graph data structures and many graph drawing techniques. Data-Driven Documentation (D3) [4], the most popular web-based visualization toolkit, incorporates some graph drawing techniques (e.g., the spring-electrical model [10]). All packages allow users to directly use different graph layout methods without implementing them from scratch. Because of the underlying models, however, these packages often expose different APIs and parameters for different methods, resulting in cumbersome parameter tuning for the user and the need for understanding different approaches. Building upon our unified force representation and universal solver, our graph drawing package is much more generic and easier to use. Different solutions can be compared and the right method for the wanted layout can be selected.

3 Proposed Framework

As mentioned above, our general framework aims to unify existing graph layout methods. It consists of a *quotient-based force model* to describe the relationship among nodes, and a *universal optimization solver* to achieve optimal graph layouts. In this section, we first show how the proposed framework originates from the observations of prior graph layout approaches. Then, we present our quotient-based force model as well as the guidelines for using it. Finally, we introduce our proposed balanced stress model.

3.1 Revisiting Existing Graph Layout Methods

For a graph $G(V,E)$ with V^2 representing the set of node pairs, graph layout methods aim to map the graph nodes V to coordinates in 2D or 3D space and often require a model to represent the relationship between them. Depending on the underlying mechanism of building the model, Hu et al. [13,21] classified layout methods into two types: *spring-electrical models* and *stress models*. They propose to use *hybrid models*, which integrate spring-electrical and stress models. Spring-electrical models often use force modeling, while stress and hybrid models are built on energy modeling to specify the graph layout. Since the force on an object is the negative derivative of the energy with respect to the distance [41], we re-write all energy-based layout methods into the form of a force modeling for establishing a unified representation. In the following, we take one representative method of each model type as an example.

Force-Directed Placement. As a typical instance of the spring-electrical model, FDP [10] aims to meet the principles that connected nodes should be drawn near each other and all nodes should not be drawn too close to each other. It computes the position of each node $\mathbf{x}_i$ by exerting the attractive force $F^a_{i,j}$ and repulsive forces $F^r_{i,j}$ between the node and its neighbours and all other nodes, respectively.

$$\mathbf{e}_{i,j} = \frac{\mathbf{x}_j - \mathbf{x}_i}{||\mathbf{x}_i - \mathbf{x}_j||}, \tag{1}$$

$$F^a_{i,j} = ||\mathbf{x}_i - \mathbf{x}_j||^2 * \mathbf{e}_{i,j}, \ \forall\{i,j\} \in E, \tag{2}$$

$$F^r_{i,j} = -\frac{1}{||\mathbf{x}_i - \mathbf{x}_j||} * \mathbf{e}_{i,j}, \ \forall\{i,j\} \in V^2, \tag{3}$$

where $\mathbf{e}_{i,j}$ is a unit vector. By successively moving each node along the resultant force F_i,

$$F_i = \sum_{\{i,j\}\in E} F^a_{i,j} + \sum_{\{i,j\}\in V^2} F^r_{i,j},$$

the final layout is obtained when the force system reaches an equilibrium. To meet given layout principles, the unit vector $\mathbf{e}_{i,j}$ in Eqs. 2 and 3 can also be computed in terms of some constraints [7] (e.g., node non-overlapping and minimal edge crossing).

Stress Model. Unlike the spring-electric model, this model aims to preserve predefined edge lengths in the visualization. It assumes that there is a spring between every pair of nodes with an ideal spring length, which is equal to their graph-theoretical distance being one unit. Hence, it obtains an optimal layout by minimizing the energy function:

$$U = \sum_{\{i,j\}\in V^2} \frac{(||\mathbf{x}_i - \mathbf{x}_j|| - d_{ij})^2}{d^2_{ij}}, \tag{4}$$

where d_{ij} denotes the graph-theoretical distance between nodes i and j. The original model is solved by using the gradient descent method with the gradient:

$$\frac{\partial U}{\partial x_i} = -\sum_{\{i,j\}\in V^2} \frac{2(||\mathbf{x}_i - \mathbf{x}_j|| - d_{ij})}{d^2_{ij}} * \mathbf{e}_{i,j}.$$

Since the negative gradient of the energy here is regarded as the acting force, we write Eq. 4 as forms of attractive and reclusive forces:

$$F^a_{i,j\in V^2} = \frac{2||\mathbf{x}_i - \mathbf{x}_j||}{d^2_{ij}} * \mathbf{e}_{i,j}, \quad F^r_{i,j\in V^2} = -\frac{2}{d_{ij}} * \mathbf{e}_{i,j}. \tag{5}$$

For escaping local minima, stress majorization [14], a widely used method for MDS solutions, has been adapted for solving this model. To reduce the computation cost, a few extended stress models choose a subset of node pairs to compute the stress energy. For example, the low-rank based stress model, Mars [27] and sparse stress model (SSM) [32] both define the force range based on a set of pivot nodes, while SSM further incorporates the edge information.

Table 1. Quotient based force functions and their corresponding parameters of different layout methods: ω is the weight, α and β are the exponents of the graph-theoretical distance and the Euclidean distance between two nodes, respectively, and Ω is the force range. P is a set of pivot nodes [27], k_{fa} is defined as $-(deg(i)+1)(deg(j)+1)$ [22] with the node degree $deg(i)$. V^2 refers to all node pairs, E to node pairs connected by an edge, S to a k-ring neighborhood graph.

Method	Attractive Force	$\{\omega_1, \alpha_1, \beta_1, \Omega_1\}$	Repulsive Forces	$\{\omega_2, \alpha_2, \beta_2, \Omega_2\}$
FDP [10]	$\sum_{(i,j)\in E} \|\|\mathbf{x}_i - \mathbf{x}_j\|\|^2 \mathbf{e}_{ij}$	$\{1,2,0,E\}$	$\sum_{\{i,j\}\in V^2} \frac{-1}{\|\|\mathbf{x}_i - \mathbf{x}_j\|\|} \mathbf{e}_{ij}$	{-1,-1,0,V^2}
FA2 [22]	$\sum_{(i,j)\in E} \|\|\mathbf{x}_i - \mathbf{x}_j\|\| \mathbf{e}_{ij}$	$\{1,2,0,E\}$	$\sum_{\{i,j\}\in V^2} \frac{k_{fa}}{\|\|\mathbf{x}_i - \mathbf{x}_j\|\|} \mathbf{e}_{ij}$	$\{k_{fa}, -1, 0, V^2\}$
LinLog [31]	$\sum_{(i,j)\in E} 1 * \mathbf{e}_{ij}$	$\{1,1,0,E\}$	$\sum_{\{i,j\}\in V^2} \frac{-1}{\|\|\mathbf{x}_i - \mathbf{x}_j\|\|} \mathbf{e}_{ij}$	$\{-1,-1,0,V^2\}$
SM [14]	$\sum_{\{i,j\}\in V^2} \frac{2\|\|\mathbf{x}_i - \mathbf{x}_j\|\|}{d^2_{ij}} \mathbf{e}_{ij}$	$\{2,1,2,V^2\}$	$\sum_{\{i,j\}\in V^2} \frac{-2}{d_{ij}} \mathbf{e}_{ij}$	$\{-2,0,1,V^2\}$
MARS [27]	$\sum_{(i,j)\in P\times V} \frac{2\|\|\mathbf{x}_i - \mathbf{x}_j\|\|}{d_{ij}} \mathbf{e}_{ij}$	$\{2,1,1,P\times V\}$	$\sum_{(i,j)\in P\times V} -2\mathbf{e}_{ij}$	$\{-2,0,0,P\times V\}$
SSM [32]	$\sum_{(i,j)\in P\times V \cup E} \frac{2\|\|\mathbf{x}_i - \mathbf{x}_j\|\|}{d^2_{ij}} \mathbf{e}_{ij}$	$\{2,1,2,P\times V\cup E\}$	$\sum_{(i,j)\in P\times V \cup E} \frac{-2}{d_{ij}} \mathbf{e}_{ij}$	$\{-2,0,1,P\times V\cup E\}$
Maxent [13]	$\sum_{\{i,j\}\in S} \frac{2\|\|\mathbf{x}_i - \mathbf{x}_j\|\|}{d^2_{ij}} \mathbf{e}_{ij}$	$\{2,1,2,S\}$	$(\sum_{\{i,j\}\in S} \frac{-2}{d_{ij}} + \sum_{\{i,j\}\in V^2} \frac{-\alpha sgn(q)}{\|\|\mathbf{x}_i - \mathbf{x}_j\|\|^q})\mathbf{e}_{ij}$	$\{-2,0,1,S\}$, $\{-\alpha sgn(q), -q, 0, V^2\}$

Maxent-Stress Model. Instead of specifying springs for all node pairs, the maxent-stress model [13] is a hybrid model that defines a stress model constraint on a subset of node pairs (typically, the set of graph edges E), while imposing an entropy-based constraint to the rest of the node pairs. Hence, the energy function is defined as follows:

$$U = \begin{cases} \sum_{(i,j)\in S} \frac{(||\mathbf{x}_i - \mathbf{x}_j|| - d_{ij})^2}{d^2_{ij}} + \alpha \sum_{(i,j)\notin S} \frac{sgn(q)}{||\mathbf{x}_i - \mathbf{x}_j||^q}, \text{ if } q \neq 0 \\ \sum_{(i,j)\in S} \frac{(||\mathbf{x}_i - \mathbf{x}_j|| - d_{ij})^2}{d^2_{ij}} + \alpha \sum_{(i,j)\notin S} \ln ||\mathbf{x}_i - \mathbf{x}_j||, \text{ if } q = 0 \end{cases}$$

where the default S is E but can also be the k-neighborhood graph, $\alpha > 0$ and $q > -2$. When q is not zero, the gradient of this model is:

$$\frac{\partial U}{\partial x_i} = -\Big(\sum_{(i,j)\in S} \frac{2(||\mathbf{x}_i - \mathbf{x}_j|| - d_{ij})}{d^2_{ij}} - \alpha \sum_{(i,j)\notin S} \frac{\mathrm{sgn}(q)}{q||\mathbf{x}_i - \mathbf{x}_j||^{q+1}}\Big) * \mathbf{e}_{i,j},$$

otherwise the gradient is:

$$\frac{\partial U}{\partial x_i} = -\left(\sum_{(i,j)\in S} \frac{2(||\mathbf{x}_i - \mathbf{x}_j|| - d_{ij})}{d^2_{ij}} - \alpha \sum_{(i,j)\notin S} \frac{1}{||\mathbf{x}_i - \mathbf{x}_j||}\right) * \mathbf{e}_{i,j}.$$

We can see that its attractive force has the same form but is only exerted to a subset of node pairs, while two repulsive forces are exerted in different ranges (as defined by S)

$$F^a_{i,j\in S} = \frac{2||\mathbf{x}_i - \mathbf{x}_j||}{d^2_{ij}} * \mathbf{e}_{i,j}, \quad F^{r1}_{(i,j)\in S} = -\frac{2}{d_{ij}} * \mathbf{e}_{i,j}, \tag{6}$$

$$F^{r2}_{(i,j)\notin S} = -\alpha \sum_{(i,j)\notin S} \frac{\mathrm{sgn}(q)}{q||\mathbf{x}_i - \mathbf{x}_j||^{q+1}} * \mathbf{e}_{i,j}. \tag{7}$$

When $q = 0$, the second repulsive force is:

$$F^{r2}_{(i,j)\notin S} = -\alpha \sum_{(i,j)\notin S} \frac{1}{||\mathbf{x}_i - \mathbf{x}_j||} * \mathbf{e}_{i,j}. \tag{8}$$

Besides these three methods, Table 1 lists the attractive and repulsive forces of a few other methods such as FA2 [22], Linlog [31], and MARS [27].

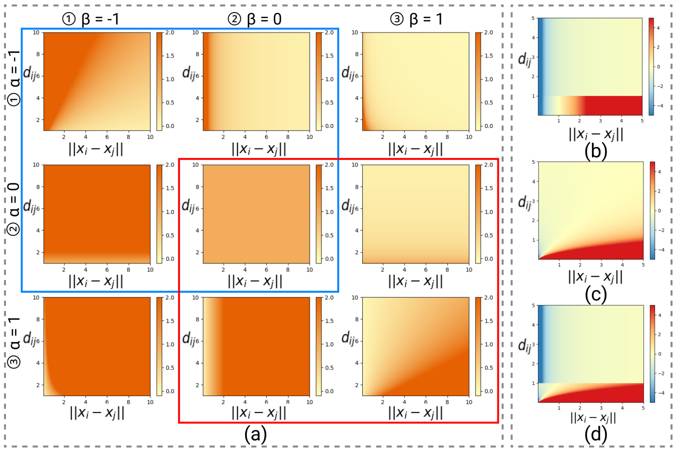


Fig. 1. (a) Influence of the parameters $\{\alpha, \beta\}$ on the force. Each plot shows the force magnitude as a function of the graph-theoretical distance between two nodes in the graph and the pairwise Euclidian distance in the layout for the given combination of α and β. The yellow color represents a force magnitude close to zero and the orange color a large force magnitude. The red and blue boxes cover the parameter settings satisfying the criteria **G1** and **G2**, respectively. (b) shows the resultant forces for FDP, (c) for the stress model and (d) for the maxent stress model.

3.2 Quotient based Force Function

After systematically comparing and analyzing various forces used in different graph layout methods (see Table 1), in the following we identify common components that appear in most methods, and further propose a quotient-based representation to unify them. Given l forces, the resultant force F_i exerted on a node i is:

$$F_{i,k} = \sum_{(i,j)\in\Omega_k} \omega_k * \frac{||\mathbf{x}_i - \mathbf{x}_j||^{\alpha_k}}{d_{ij}^{\beta_k}} * \mathbf{e}_{k,ij}, \quad F_i = \sum_k^l F_{i,k} \tag{9}$$

where Ω_k is the *force range* specifying the nodes that exert forces towards node i, ω_k is a weight with a sign deciding the *force type* (attractive force vs. repulsive force), α_k is the exponent of the graph-theoretical distance, β_k of the Euclidean distance between nodes j and i, which decide the *force magnitude*. To illustrate how this representation unifies existing graph layout methods, we formulate some popular methods in this representation, see Table 1. Taking the maxent-stress model as an example, its attractive and first repulsive force have the same force range $\{i, j\} \in S$, and their corresponding $\{\omega, \alpha, \beta\}$ are $\{2, 1, 2\}$ and $\{-2, 0, 1\}$, respectively. The range of the other repulsive force Ω is $\{i, j\} \notin S$, the parameters $\{\omega, \alpha, \beta\}$ are $\{-q\,\text{sgn}(q), -q, 0\}$. Note that low-rank stress majorization (MARS) [5] and the sparse stress model (SSM) [32] exert forces for ranges defined by the user specified pivot nodes P.

Parameter Effect. A closer look at the examples in Table 1 tells us that there should be at least two forces ($l \geq 2$) with different signs for weights ω_k to ensure convergence. Common force ranges are E, V^2 or the ones defined by the user-specified node sets P. In contrast, there are many choices of $\{\alpha_k, \beta_k\}$, each of them might create different behaviors. In the following, we show how existing methods set these parameters.

To investigate the effect of $\{\alpha_k, \beta_k\}$, we depict the force magnitude between two nodes i and j as a function of their pairwise Euclidean distance $||\mathbf{x}_i - \mathbf{x}_j||$ and the graph-theoretical distance d_{ij}. Fig. 1(a) shows the force magnitudes under different combinations of a few commonly used α and β values $\{-1, 0, 1\}$. From the plots, we can see that there are three kinds of the relationship:

1. If $\alpha\beta$ is zero, the force magnitude is purely determined by $||\mathbf{x}_i - \mathbf{x}_j||$ or d_{ij};
2. If $\alpha\beta$ is smaller than zero , the factors d_{ij} and $||\mathbf{x}_i - \mathbf{x}_j||$ have the same positive or negative effect on the force magnitude; and
3. If $\alpha\beta$ is larger than zero, the factors d_{ij} and $||\mathbf{x}_i - \mathbf{x}_j||$ have the opposite effect on the force magnitude.

Case 1 corresponds to the forces defined in FDP (see Eqs. 2 and 3) and the repulsive forces used in the stress model (see Eq. 5). In contrast, there is only one example (see the attractive force in Eq. 5) for Case 3 in Table 1 and no example for Case 2. In the following, we provide the guidelines for selecting proper α and β.

3.3 Guidelines for the Selection of Exponents

To faithfully maintain the relationship between nodes, we generalize a core principle of the FDP method about connected nodes to all node pairs. In FDP connected nodes should always be closer to each other than to other nodes.

- Nodes with small graph-theoretical distances should be drawn closer to each other than nodes with large distances.

For simplicity, we divide all k forces within the graph into attractive and repulsive forces. To meet the above principle, two nodes with a larger graph-theoretical distance should be exerted a larger repulsive force ($\beta < 0$) and a smaller attractive force ($\beta > 0$). To prevent the layout from diverging to infinity or collapsing into a point, for two nodes with fixed graph-theoretical distances, the repulsive force should decrease as the Euclidian distance between the two nodes increases ($\alpha < 0$), and the attractive force should decrease as the Euclidian distance between two nodes decreases ($\alpha > 0$). For yielding a clustering (dispersing) effect, we can also use a constant repulsive (attractive) force with a large attractive (repulsive) force by setting $\alpha = 0$ ($\beta = 0$). Therefore, we identify the following two guidelines for choosing α and β:

- **G1:** For the attractive force, the exponent parameters are suggested to satisfy: $\alpha \geq 0, \beta \geq 0$; and
- **G2:** For the repulsive force, the exponent parameters are suggested to satisfy: $\alpha \leq 0, \beta \leq 0$.

The examples in Fig. 1 enclosed by the red box correspond to parameters meeting **G1** and blue box to parameters meeting **G2**.

Looking again at Table 1, we see that the attractive forces of all methods satisfy **G1**, whereas the repulsive force of some existing graph layout methods violate **G2**. For example, the repulsive forces of SM [14], MARS [27] and SSM [32] is $\frac{-2}{d_{ij}}$ are not in the blue box, but in the red box of Fig. 1. Since their magnitude decreases with increasing graph-theoretical distance, the repulsive force might not be able to repel node pairs with large graph-theoretical distances far from each other, resulting in false neighborhoods.

Reflections. To explore how different choices of α and β influence the final layout, we compute the resultant force for the two nodes with varying d_{ij} and Euclidean distance $||\mathbf{x}_i - \mathbf{x}_j||$. Figs. 1 (b,c,d) show the results for three methods: FDP, stress model and maxent-stress model, where a positive value indicates that the attractive force is larger than the repulsive force and vice versa for a negative value.

In Fig. 1(b), we can see that FDP exerts large repulsive forces on nodes with small Euclidean distance, but only applies attractive forces to connected nodes. In other words, it treats all nodes with d_{ij} being larger than 1 equally, resulting in a poor overall distance preservation.

Fig. 1(c) allows two observations about SM: i) the resultant force is close to zero in most places (see the yellow region), except the ones for nodes with small graph-theoretical distances; and ii) the force exerted on nearby nodes with large graph-theoretical distances is close to zero (see top-left corner). In other words, the original stress model might not be able to efficiently preserve local neighborhood structures and large graph-theoretical distances.

For the maxent-stress model, we set S and q to E and zero and show the resultant forces for two nodes in Fig. 1(d). When d_{ij} is larger than one, its resultant force is equal to the one of FDP, as shown in Fig. 1(b), otherwise it is the same as the one of the stress model shown in Fig. 1(c). Hence, we conclude that this model can be regarded as the combination of the spring-electrical and the stress model.

3.4 A Balanced Stress Model (BSM)

The above considerations and the possibility to formulate different methods with a unified formula help us to formulate a balanced stress model that would have an "ideal" behavior. It is surprisingly simple and in contrast to many methods we would consider to be "ad hoc" in their selection of weights and exponents its behavior is directly derived from the nature of the underlying problem.

As mentioned in Section 3.3, it is desirable to exert large repulsive force to the node pairs with large graph-theoretical distances d_{ij}, a fact that the stress model does not satisfy. To address this issue, we propose the following model:

$$F_i = \sum_{i,j \in V^2} \left(\frac{||\mathbf{x}_i - \mathbf{x}_j||}{d_{ij}} - \frac{d_{ij}}{||\mathbf{x}_i - \mathbf{x}_j||} \right) * \mathbf{e}_{ij}, \tag{10}$$

where the second term can be taken as a weighted repulsive force from the graph-theoretical distance as used in FDP (see Eq. 3), its reciprocal corresponds to the term of the attractive force. Doing so, nodes with large graph-theoretical distances are repelled far from each other. On the other hand, a model after Eq. (10) is able to effectively preserve graph-theoretical distances, since the resultant force is zero when the graph-theoretical distance between two nodes is equal to their Euclidean distance. To our understanding, this is a meaningful setup that builds on the intuition that a layout should reflect graph-theoretical distances as good as possible in its projected distances.

As shown in Fig. 2 (a), the resultant forces exerted on two nodes forms a skew-symmetric matrix and all forces are zero at the diagonal of the matrix. For us, an ideal behavior.

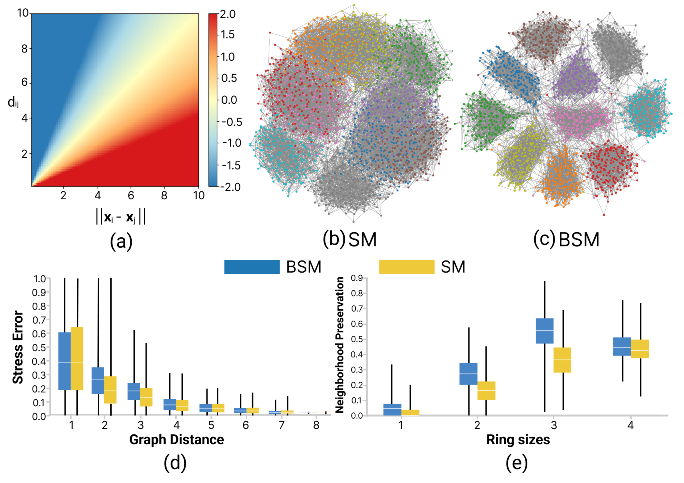


Fig. 2. (a) Resultant force for two nodes with varying graph-theoretical and Euclidian distances generated by our balanced SM. (b) Layouts generated by SM and balanced SM (c) on the graph *cluster_2000*. Our method clearly separates different clusters. (d,e) The boxplots summarize the value ranges (same for all boxplots in this paper) of the stress error (d) in terms of different graph distances and neighborhood preservation (e) degrees for different ring sizes.

Figs. 2(b,c) compare SM with our balanced model BSM using *cluster_2000*. BSM clearly reveals the cluster structures, while SM shows much less separation. To further investigate the differences between the methods, we calculate the stress error for different graph-theoretical distances and the neighborhood preservation degrees of different ring sizes. As shown in Figs. 2(d,e), BSM performs slightly worse than SM for smaller graph-theoretical distances and similar or even better for large distances, whereas it outperforms SM in neighborhood preservation no matter what the ring size is. These results are consistent with our reflections about SM in Section 3.3 and hence we speculate that our BSM performs better in balancing distance preservation and neighborhood preservation.

4 Augmented SGD Solver

Before we evaluate the balanced stress model in Section 5, we outline our second contribution that allows us to compare all the mentioned models. Recently, Zheng et al. [42] adapt the stochastic gradient descent technique, a powerful optimization solver widely used for training deep neural networks, to solve the stress model. They show that SGD can reach lower stress errors faster than stress majorization [14], while not requiring a good initialization. However, the method is originally designed for minimizing energy functions and thus has not been used for solving force-based layout methods.

Since energy is the negative integral of a force, we can formulate each of the quotient-based force models (see Eq. 9) as the following energy:

$$U_k = \begin{cases} \sum_{(i,j)\in\Omega_k} \frac{\omega_k}{\alpha_k + 1} * \frac{||\mathbf{x}_i - \mathbf{x}_j||^{\alpha_k+1}}{d_{ij}^{\beta_k}} & \alpha_k \neq -1 \\ \sum_{(i,j)\in\Omega_k} \omega_k \frac{\ln ||\mathbf{x}_i - \mathbf{x}_j||}{d_{ij}^{\beta_k}} & \text{otherwise.} \end{cases} \tag{11}$$

To minimize this energy, SGD repeatedly randomly picks a pair of nodes $\mathbf{x}_i$ and $\mathbf{x}_j$ for moving it along the force direction at a time:

$$\mathbf{r} = \frac{\omega_k}{2} * \frac{||\mathbf{x}_i - \mathbf{x}_j||^{\alpha_k}}{d_{ij}^{\beta_k}} * \mathbf{e}_{k,ij} \tag{12}$$

$$\mathbf{x}_i = \mathbf{x}_i - \eta_s \mathbf{r}, \quad \mathbf{x}_j = \mathbf{x}_j + \eta_s \mathbf{r}, \tag{13}$$

where η_s is the step size. With a carefully chosen step size, this method quickly converges to a reasonable layout. Unlike Zheng et al. [42], our SGD solver updates each node with two steps, since the gradient of the stress model in Eq. 4 is separated into attractive and repulsive forces. This facilitates us to apply SGD to solve force-based methods. However, directly applying SGD to FDP-based methods might be too expensive because of its $O(n^2)$ time complexity. To alleviate this issue, we suggest to combine it with the Barnes-Hut (BH) [3] method, which approximates the repulsive force between nodes by using a quadtree structure with a time complexity of $O(n \log n)$. Note that this approximation only holds for forces with parameters $\omega_k < 0$ and $\beta_k = 0$. With this augmented SGD(ASGD) solver, we first compute the repulsive force using the BH solver to move nodes and then use the SGD solver to update nodes at each iteration, as outlined in Algorithm 1.

Algorithm 1 Pseudocode for our augmented SGD solver

```
1:  Input: graph $G = (V, E)$,
2:  $\mathbf{X}$ = $RandomMatrix(|V|, 2)$
3:  for i = 1 to $iters$ do
4:      for k = 1 to $l$ do
5:          if $\omega_k < 0$ and $\beta_k = 0$ then
6:              solve_BH_forces($\mathbf{X}, \alpha_k, \beta_k, \Omega_k$)
7:          else
8:              solve_SGD_forces($\mathbf{X}, \alpha_k, \beta_k, \Omega_k$)
9:          end if
10:     end for
11: end for
12: return X
```

Fig. 3(a) shows the convergence curves of different solvers for computing the FDP layout of the *btree9* graph. Our solver performs similarly to the SGD solver, while both converge to smaller energy values than the BH solver. On the other hand, our solver is faster at each iteration and its overall time is lower than the SGD solver (see Section 5). Moreover, it is surprising that our solver even produces better layouts than the SGD solver. As shown in Figs. 3(b,c,d), only our solver is able to maintain the tree structure of the data. We speculate that this is due to the separation of attractive and repulsive forces for node movement.

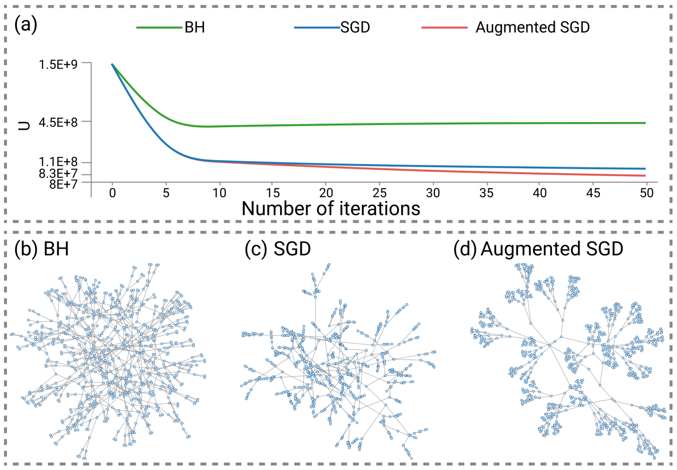


Fig. 3. Comparing three solvers (BH, SGD, and our augmented SGD) for the FDP method to layout the *btree9* graph. (a) The plots of $U(\mathbf{X})$ versus the number of iterations showing the convergence of the three solvers. Our augmented SGD solver performs similarly to SGD. (b,c,d) The layout results generated by three solvers, the one produced by our augmented SGD solver preserves the tree structure well.

5 Evaluation

We implemented our framework in C++ and warped it into an open-source library[1]. To customize a graph layout method, users only need to configure and assemble different forces together in terms of the design guidelines from Section 3.3. Based on this library, we evaluate Taurus from four perspectives. First, we validate our augmented SGD solver by comparing it with BH and SGD on a set of graphs. Second, we check to what extent the implementations of existing graph layout methods under Taurus generate similar results as the original ones. Third, we compare our proposed balanced stress model with the existing graph layout methods. Last, we present a usage scenario to illustrate the flexibility of Taurus for supporting the customization of graph layouts. All the experiments were done on a Windows desktop computer with an Intel Core i7-9700K processor with 32GB memory.

5.1 Experimental Design

Methods. To evaluate the capability of Taurus in expressing existing graph layout techniques, we selected five well-known methods as baselines: force-directed placement (FDP) [10], LinLog [31], stress model (SM) [14], Maxent [13], and FM^3 [18]. They are chosen by considering two factors: First, they cover three major categories of existing graph layout methods (spring-electrical models: FDP, LinLog and FM^3; stress models: SM; hybrid models: Maxent), as introduced in Section 2.1. Second, they have widely-used implementations. Four of these methods have C++ implementations: FDP, FM^3 and SM within the *OGDF* [6] package, and Maxent within the *graphviz* [9] package. For LinLog, we use the author's Java implementation [31]. We re-implement these graph layout methods under Taurus, and compared our results with the existing implementations.

Datasets. To evaluate our general framework, we generated three types of commonly-used graph datasets (i.e., grids, binary trees, and clustered graphs) and 15 real graphs of different applications.

- *Grids* are graphs with a regular tiling. The ideal layout results will be a uniform grid consisting of squares of a uniform size. Grids can be used to assess whether a graph layout method is able to preserve the regular graph structures. We generate both 2D and 3D grids of different sizes.

- *Binary trees* have been widely used to evaluate graph layout methods [13, 27]. Binary trees are often symmetric. Using them, we can evaluate whether a method is able to preserve the symmetric structures of a graph.

[1]`https://github.com/Ideas-Laboratory/Taurus`

- *Clustered graphs* refer to networks with clear community structures, which can be used to evaluate whether a graph layout method preserves such structures in the layout. We use the *Stochastic Block model* [24] implemented in the graph-tool [33] to generate such graphs. The number of communities in the generated graphs ranges from five to fifteen.

- *Real graphs* are networks publicly available on the Internet used by prior studies [29, 35, 43]. They come from different fields such as biology, social sciences and environmental structures, and have different sizes.

We generated fifteen example graphs for each of the three types of synthetic graphs and collected fifteen real graphs. To ensure that our solver can efficiently find optimal solutions for our layout methods, we did not test large graphs but used exemplars with 100 to 5000 nodes and 128 to 19016 edges. **Parameters.** There are three parameters of the augmented SGD solver, which influence the graph layout quality and speed, i.e., the maximum number of iterations, the step size η, and the decay rate of the Barnes-Hut algorithm λ. For η, we follow the suggestion of Zhang et al. [42] that gradually decreases from 1 to 0.01, while setting the maximum number of iterations to 200. For SM and BSM, we found that usually 30 iterations are enough for convergence. To remove the influence of the initialization, we use the same randomized initial layout for all methods on each graph.

Measures. Seven measures are used to evaluate the similarity of different implementations for the same graph layout method and to compare their performances. These measures were chosen to evaluate the capability for different graph structure preservation or graph readability.

- *Normalized stress error* (SE) [13] is used to measure the overall preservation of the graph-theoretical distances in the graph layouts. A small value indicates that graph-theoretical distances between nodes are well maintained.

- *Neighborhood preservation* (NP) checks whether the neighborhood around each node in the graph structure is also the neighborhood in the layout. We use the neig hborhood preservation measure introduced in [37], and define the neighborhood nodes of a graph node as the nodes with a maximum of two edges from it. A larger value of NP is preferred.

- *Crosslessness* (CL) quantifies the number of non-crossing edge pairs in a graph layout [34]. A larger score indicates fewer edge crossings.

- *Minimum Angle* (MA) measures the average deviation of the minimum angle from the ideal angle for each node in a layout. [34] A small value is preferred.

- *Runtime* measures the average time for computing a layout. We assess the average runtime of each graph layout method on one graph by calculating the average over 5 runs.

- *Cluster Extraction* (CE) [38] delineates the average distance of nodes within the same cluster. A small value of this measure indicates compact clusters, which is helpful for identifying graph communities and is thus preferred.

- *Cluster Distance* (CD) [38] measures the separation of different clusters in the layout. We calculate the minimum distance between nodes of two different clusters here. A large value indicates different clusters are well separated and is thus preferred.

Since CE and CD both require the cluster information, we only apply them to the clustered graphs.

To consistently compare different methods, we evaluate the relative difference of the measure M between the target implementation M_t and the reference implementation M_r:

$$\delta M = \frac{M_t - M_r}{M_r}, \tag{14}$$

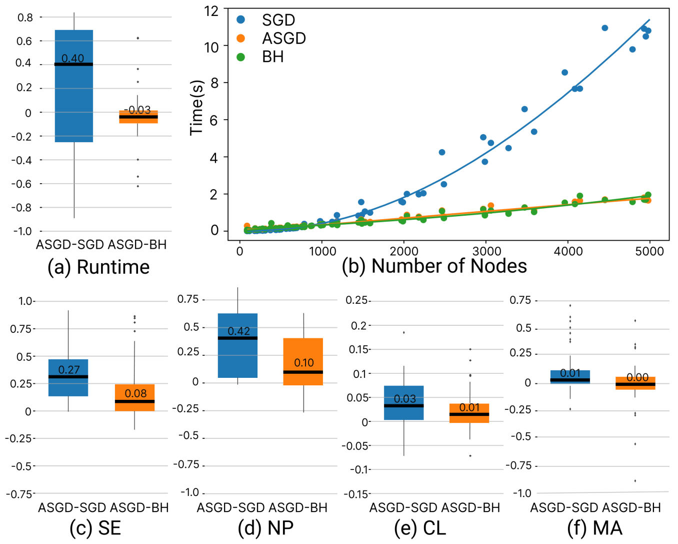


Fig. 4. (a) The boxplots of the runtime for different solvers, in which black lines represent the median value. (b) the plots of runtime versus number of nodes for each dataset. (c,d,e,f) The boxplots of four measures SE(c), NP(d), CL(e) and MA(f) for the layouts generated by using ASGD vs. BH and SGD. A large value is better in all boxplots.

where a value around zero indicates a specific implementation is similar to the reference implementation. Among all measures, positive differences of NP, CL and CD indicate a better performance of the target implementation, while negative differences of SE, MA, CE and runtime mean that the reference implementation performs better. To consistently show larger values as being better, we take the final value for SE, MA, CE and runtime as $1 - \delta M$.

5.2 Comparison between Different Solvers

Since our ASGD is equivalent to SGD for stress model based methods, we only compare its efficiency with SGD and BH for spring based methods. Here, we choose the classic FDP method as an example method for comparison in terms of runtime and layout quality. To ensure a fair comparison, we use the same convergence condition for three solvers, while running each solver for a graph five times and calculating the average measures. For all measures, we apply Eq. 14 to normalize the results by taking the implementation based on our ASGD as M_t and the two others as the references.

Due to space limits, we only show the summarized runtime and four measures in Fig. 4, the complete scores of all measures can be found in the supplemental material, as well as runtimes and visualizations for additional ten large graphs. The boxplots in Fig. 4(a) provide a statistical summary of the runtime of three methods on all tested graphs, where our ASGD is faster than SGD by 40 percent and slightly slower than BH. To learn how fast these solvers are, we plot the relationship between runtime and the number of graph nodes in Fig. 4(b). We can see that all solvers perform similarly for graphs with a number of nodes being smaller than 1500, while ASGD and BH have significant advantages over SGD for large graphs. With increasing number of graph nodes, the runtime of SGD increases quadratically, while the runtime of ASGD and BH increases logarithmically. For the largest graph with 5000 nodes, ASGD and BH are six times faster than SGD (1.8s vs. 10.5s).

Figs. 4(c-f) provide a summary of four layout quality measures in terms of relative differences. For SE and NP, ASGD demonstrates significant benefits over SGD and BH, where the median of the relative increase is at least 8%, while it is similar for the other measures. In all, ASGD is significantly faster than SGD and similar to BH, while generating the best layout.

5.3 Comparison between Different Implementations

To demonstrate the effectiveness of our quotient-based representation and ASGD solver, we compare the implementation of each layout method under our framework as M_t with its original or existing implementation as M_r and compute δM in terms of the above measures.

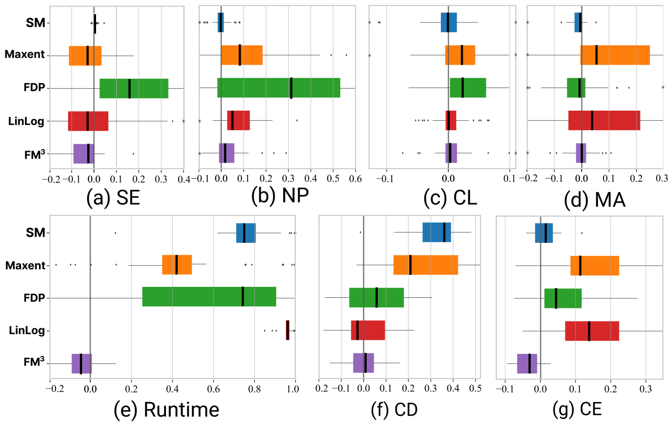


Fig. 5. Relative differences of the measures SE(a), NP(b), CL(c), MA(d) and runtime(e) for the layouts of all tested graphs and CD(f) and CE(g) for the clustered graphs by our implementation vs. existing ones. A larger value for all measures is better.

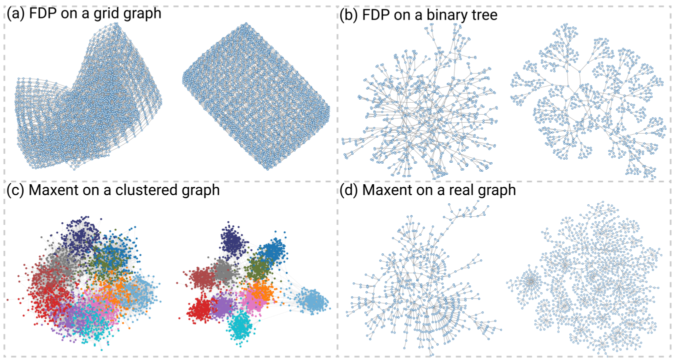


Fig. 6. Comparison of layout results on the same data generated by the same methods using the existing implementation (left) and our implementation (right). (a,b) Results generated by FDP; (c,d) results generated by Maxent.

Quantitative Results. The boxplots in Fig. 5 summarize the relative differences of seven measures computed from all tested datasets. For SE, the median values of the relative differences on all methods are either around zero (i.e., SM, Maxent, LinLog and FM^3 with a difference less than 3%), or larger than zero (FDP with a difference around 15%) as shown in Fig. 5(a). Similarly, the median value of the relative differences of NP on SM is around zero, the ones of Maxent, LinLog and FM^3 are larger than zero.

These results indicate that our implementation enables FDP to better preserve stress and neighborhood, while showing similar performances for the other methods. We have similar observations for the two readability measures in Figs. 5(c,d). Our implementations result in less edge crossing (CL) and smaller minimum angle (MA) for Maxent, FDP LinLog, and FM^3, while maintaining the quality of SM.

Furthermore, it largely reduces the runtime of SM, Maxent, FDP and LinLog by around 75%, 45%, 75% and 90%, respectively (see Fig. 5(e)). For FM^3, it leads to a slightly longer runtime (less than 5%) for most graphs. We speculate that the reason or this is the fact that our ASGD solver is not inherently designed for solving multi-level graph layout whereas the BH method is. Note that the SM implementation in the OGDF library is solved by stress majorization rather than SGD, FDP in the OGDF library is the exact implementation, while the other methods in the existing implementations are based on BH methods.

Figs. 5(f,g) show the results of the measures CD and CE for clustered graphs. We can see that the relative differences of all methods are positive or close to zero for CE and CD, indicating that our implementation has a higher capability in revealing cluster structures.

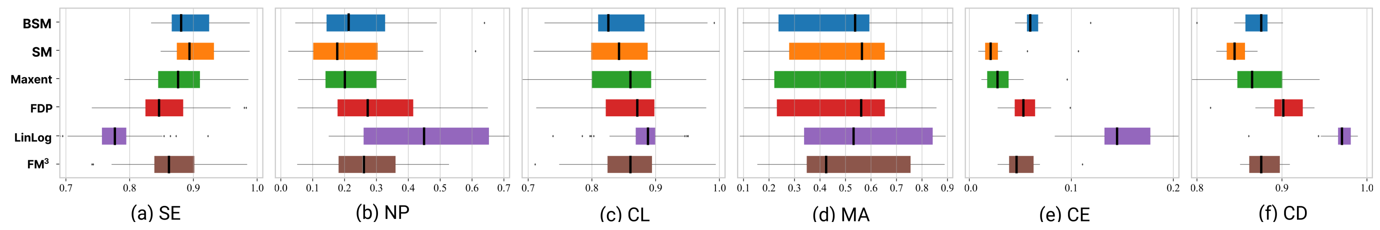


Fig. 7. Six measures over all datasets for six layout methods. A large value is better for all measures.

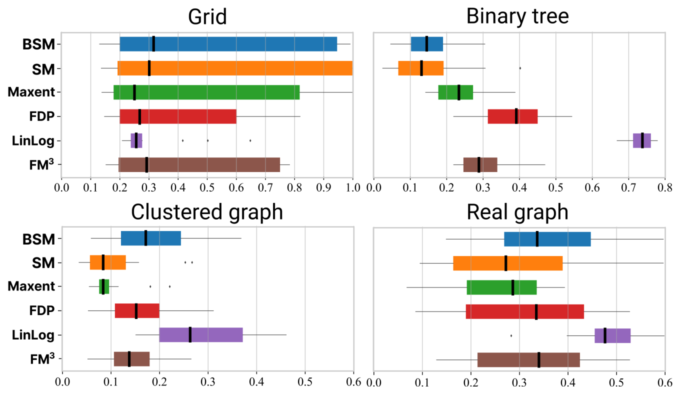


Fig. 8. Bxplots summarizing the NP measure of each type of graphs for six different layout methods. A larger value is better.

Qualitative Results. The above analysis shows that our implementation produces similar results as the baseline for stress model and LinLog, while largely improving the layout results for FDP and Maxent. Fig. 6 compares the results produced by different implementations of FDP and Maxent on four typical graphs.

Our FDP implementation is able to effectively maintain the grid structure for the graph shown in Fig. 6(a), whereas the grid is highly deformed with folds by the baseline implementation. We have similar observations for the tree structure shown in Fig. 6(b), our result on the right has fewer edge crossings and reveals a clearer hierarchy of clusters than the original one on the left. These results are consistent with the ones shown in Fig. 3, demonstrating the efficiency of our solver.

Figs. 6(c,d) show results generated by two implementations of Maxent. Our result is significantly better in terms of cluster preservation and reveals all different communities (Fig. 6(c)), while the original mixes clusters. This also holds for the collaboration network (Fig. 6(d)). Compared to the baseline, our result distributes all local clusters in 2D plane evenly with fewer edge crossing, while maintaining the overall structure. Yet, the baseline implementation better reveals the multi-ring structures, but has strong node overlaps and edge crossings.

Overall, the results of all methods implemented in Taurus are *similar to* or *even better than* the original implementations, with a largely reduced runtime. In particular, our versions of FDP and Maxent largely improve layout quality.

5.4 Comparison between Different Methods

Here, we compare our proposed layout method BSM with five existing methods (SM, FDP, Maxent, LinLog and FM^3). To show the effectiveness of this model, we implemented all these methods by Taurus with the same solver and use the same initial layout for each tested graph.

Quantitative Results. Fig. 7 summarizes the values of six measures overall tested graphs using boxplots. For the complete results presented in table, please refer to the supplemental material In terms of the stress error, BSM is slightly worse than SM but outperforms the other methods, while LinLog is the worst. In contrast, BSM is better than SM and Maxent in neighborhood preservation, close to FDP and FM^3, worse than LinLog. Regarding readability measures, BSM performs similarly to the other methods but is worse than LinLog for CL. Similarly, BSM is worse than LinLog and similar as FDP and FM^3 with regard to CE and CD on clustered graphs. This is reasonable, since LinLog is designed for revealing clusters, while BSM is for distance preservation.

While LinLog performs the worst in terms of the stress error, it is the best for preserving neighborhoods. This is interesting, since we often assume that LinLog is good in revealing clusters and FDP and FM^3 performs well in neighborhood preservation. After examining the statistics of each type of graph in Fig. 8, we found that FM^3 works well for all graphs and LinLog performs poorly on grids but works well for other graphs, whereas SM and our BSM perform best for such graphs. For tree structured graphs, BSM performs better than SM but worse than the others, because maintaining the overall distance is not helpful here. In contrast, BSM performs similar or even better than FDP and FM^3 on clustered and real graphs, although it is designed for preserving graph-theoretical distances. For each of the other measures, the distribution of the results does not show any significant dependency on the data type.

In summary, BSM achieves a balanced capability in preserving overall distances (SE), neighborhoods (NP) and clusters (CE and CD), especially for clustered and real graphs. In addition, it maintains similar graph readability values as the other methods and therefore matches our design goal.

Qualitative Results. Fig. 9 shows the visual results of four typical example graphs. The dataset in the first row of Fig. 9 is a graph with a 3D-like grid, where only BSM and SM preserve the grid structure, while the other methods heavily deform it.

BSM is able to clearly visualize the major branches and the symmetry of tree graph shown in the second row of Fig. 9. Other methods (except SM) do not preserve the symmetry but show more minor branches. For this graph, Maxent, FDP and LinLog have a higher score in neighborhood preservation, whereas they do not show major branches. Hence, we speculate that BSM might perform even better in visualizing large tree structures than these methods. Although LinLog allows to inherently reveal cluster structures, its results are too tight to explore any details. FDP and Maxent alleviate this issue, but might create overlaps between clusters. In contrast, BSM better balances intra- and inter-cluster separation well, where the maroon and yellow green clusters are at the center and are surrounded by the other clusters. This makes sense, since these two clusters have the most inter-cluster connections with the others in the ground truth data. In contrast, FM^3 does not reflect this ground truth. For the real graphs, the results of BSM are similar to or even slightly better than SM in terms of preserving radial graph structures and local clusters, while both of them perform better than the others.

Overall, the visual results by BSM align well with the observations in Section 5.4: BSM is able to visualize the structures of different types of graphs well.

5.5 Usage Scenario

Lastly, we present a usage scenario to showcase how Taurus can facilitate graph visualizations in a *unified* manner. Suppose Bob, a data scientist who often needs to explore networks in his daily work and has a basic understanding of different graph layout techniques. He wants to visualize a network that he wants to explore the potential clusters. With Taurus, he can easily try different graph layout methods within the same framework.

First, he uses Taurus to quickly implement BSM by simply configuring the four parameters $\{\Omega, \omega, \alpha, \beta\}$ for attractive forces with

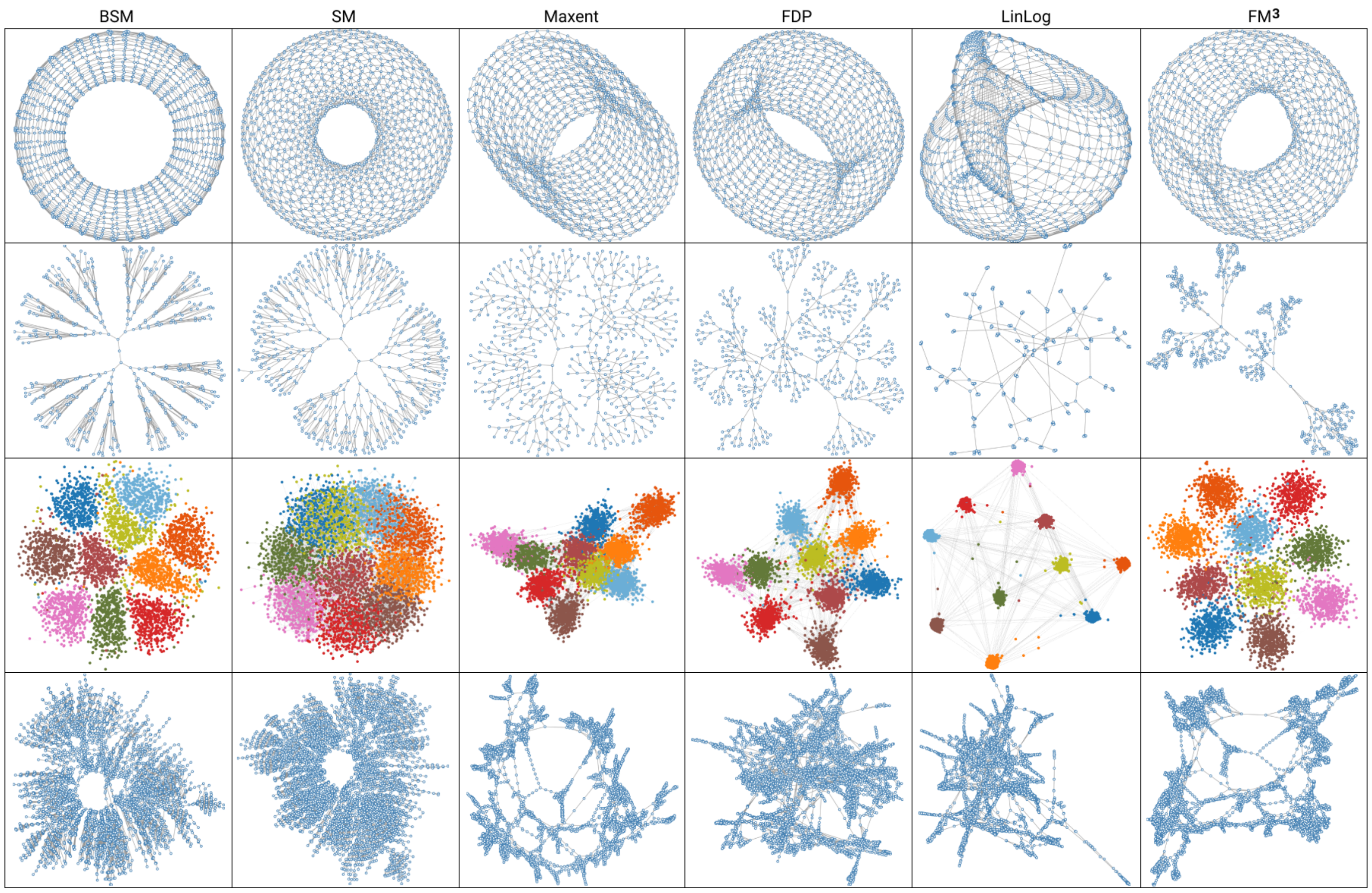
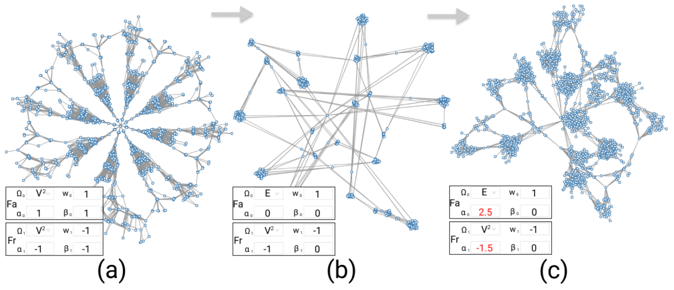


Fig. 9. BSM results of four graphs (top down: *grid_1000*, *btree_513*, *cluster_4463*, and *US_powergrid*) with different structures in comparison to baseline methods (SM, FDP, LinLog, Maxent and FM^3) implemented by Taurus.


(a)
Fa: Ω_0 V^2, w_0 1, α_0 1, β_0 1
Fr: Ω_1 V^2, w_1 -1, α_1 -1, β_1 -1

(b)
Fa: Ω_0 E, w_0 1, α_0 0, β_0 0
Fr: Ω_1 V^2, w_1 -1, α_1 -1, β_1 0

(c)
Fa: Ω_0 E, w_0 1, α_0 2.5, β_0 0
Fr: Ω_1 V^2, w_1 -1, α_1 -1.5, β_1 0

Fig. 10. Utilizing Taurus to obtain good graph layout results. By entering the corresponding parameters, the user quickly gets layout results for (a) BSM, (b) LinLog and (c) a customized layout method.

$\{V^2,1,1,1\}$ and for repulsive forces with $\{V^2,-1,-1,-1\}$. The layout result (Fig. 10 (a)) shows that the network seems to have a clustering structure. To display clusters more clearly, Bob updates the parameters to $\{E,1,0,0\}$ and $\{V^2,-1,-1,0\}$ for attractive and repulsive forces, i.e. applying the LinLog method. He obtains a more compact clustering result as shown in Fig. 10 (b). However, the clusters in this graph layout are too tight to examine the relationship between nodes within the same cluster. To achieve the optimal graph layout, Bob further adjusts α and β of the attractive force to 2.5 and -1.5, and finds that the result in Fig. 10 (c) shows the detail of each cluster more clearly, making it easier for Bob to know which clusters contain more nodes. With Taurus, Bob can easily customize graph layout methods to explore different graph structures, which facilitates him to perform a deeper analysis.

6 Conclusion and Future Work

In this paper, we propose a general framework, Taurus, to unify popular graph layout methods. It consists of two major components: a unified quotient-based force representation to model repulsive and attractive forces of different graph layout techniques, and a universal augmented stochastic gradient descent (ASGD) solver to find the optimal graph layout results. We systematically analyze our general framework and provide guidelines for designing effective graph layout methods. We also release a graph layout library based on Taurus that facilitates convenient implementation of graph visualizations in a unified manner.

In the future, we would like to extend Taurus along the following directions. First, apart from the current open-source package in C++, we plan to extend Taurus to other programming languages like Python and JavaScript. Second, Taurus provides a clear design space for graph layout techniques, but it might require users to try multiple different parameters for a specific graph analysis task (see Section 5.5). Therefore, we like to explore automated parameter tuning methods for automatically fining proper parameters to generate desired visualizations. Third, it will be interesting to further extend our framework to unify graph layout techniques with special model designs (e.g., tsNET [29] and DRGragh [43]). Last, we would like to conduct a large user study to compare the different layout methods in terms of layout principles and then use the findings to further improve our framework.

Acknowledgments

This work was supported by the grants of the National Key Research and Development Program of China (2018AAA0101100), and NSFC (62132017, 62141217). Oliver Deussen was funded by the German Research Foundation (DFG) - Project-ID 2,51654672 - TRR 161.

References

[1] R. Ahmed, F. De Luca, S. Devkota, S. Kobourov, and M. Li. Multicriteria scalable graph drawing via stochastic gradient descent, $(SGD)^2$. *IEEE Transactions on Visualization and Computer Graphics*, 28(6):2388–2399, 2022. doi: 10.1109/TVCG.2022.3155564

[2] D. Auber, D. Archambault, R. Bourqui, M. Delest, J. Dubois, A. Lambert, P. Mary, M. Mathiaut, G. Melançon, B. Pinaud, B. Renoust, and J. Vallet. TULIP 5. In *Encyclopedia of Social Network Analysis and Mining*, pp. 1–28. Springer, 2017. doi: 10.1007/978-1-4614-7163-9_315-1

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

[4] M. Bostock, V. Ogievetsky, and J. Heer. D^3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, 2011. doi: 10.1109/tvcg.2011.185

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

[6] 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*, 2011:543–569, 2013.

[7] T. Dwyer. Scalable, versatile and simple constrained graph layout. *Computer Graphics Forum*, 28(3):991–998, 2009. doi: 10.1111/j.1467-8659.2009.01449.x

[8] P. Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42:149–160, 1984.

[9] J. Ellson, E. Gansner, L. Koutsofios, S. C. North, and G. Woodhull. Graphviz—open source graph drawing tools. In *International Symposium on Graph Drawing*, pp. 483–484. Springer, Feb. 2001. doi: 10.1007/3-540-45848-4_57

[10] T. M. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software: Practice and Experience*, 21(11):1129–1164, 1991. doi: 10.1007/978-3-658-21742-6_49

[11] P. Gajer, M. T. Goodrich, and S. G. Kobourov. A multi-dimensional approach to force-directed layouts of large graphs. *Computational Geometry*, 29(1):3–18, 2004. doi: 10.1016/j.comgeo.2004.03.014

[12] P. Gajer and S. G. Kobourov. Grip: Graph drawing with intelligent placement. In *International Symposium on Graph Drawing*, pp. 222–228. Springer, May 2000. doi: 10.1142/9789812796608_0011

[13] E. R. Gansner, Y. Hu, and S. North. A maxent-stress model for graph layout. *IEEE Transactions on Visualization and Computer Graphics*, 19(6):927–940, 2012. doi: 10.1109/TVCG.2012.299

[14] E. R. Gansner, Y. Koren, and S. North. Graph drawing by stress majorization. In *International Symposium on Graph Drawing*, pp. 239–250. Springer, 2004. doi: 10.1007/978-3-540-31843-9-25

[15] E. R. Gansner and S. C. North. An open graph visualization system and its applications to software engineering. *Software: Practice and Experience*, 30(11):1203–1233, 2000.

[16] A. M. Gouvêa, T. S. da Silva, E. E. Macau, and M. G. Quiles. Force-directed algorithms as a tool to support community detection. *The European Physical Journal Special Topics*, 230(14):2745–2763, 2021. doi: 10.1140/epjs/s11734-021-00167-0

[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

[18] S. Hachul and M. Jünger. Drawing large graphs with a potential-field-based multilevel algorithm. In *International Symposium on Graph Drawing*, pp. 285–295. Springer, 2004. doi: 10.1007/978-3-540-31843-9_29

[19] S. Hachul and M. Jünger. Large-graph layout with the fast multipole multilevel method. *Spring, V (December)*, pp. 1–27, 2005.

[20] Y. Hu. Efficient high-quality force-directed graph drawing. *Mathematica Journal*, 10(1):37–71, 2005.

[21] Y. Hu and Y. Koren. Extending the spring-electrical model to overcome warping effects. In *IEEE Pacific Visualization Symposium*, pp. 129–136. IEEE, May 2009. doi: 10.1109/PACIFICVIS.2009.4906847

[22] 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

[23] T. Kamada, S. Kawai, et al. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31(1):7–15, 1989. doi: 10.1142/9789814434478_0005

[24] B. Karrer and M. E. Newman. Stochastic blockmodels and community structure in networks. *Physical Review E*, 83(1):016107, 2011. doi: 10.1103/PhysRevE.83.016107

[25] M. Kaufmann and D. Wagner. *Drawing graphs: methods and models*. Springer, 2003.

[26] A.-M. Kermarrec and A. Moin. FlexGD: A flexible force-directed model for graph drawing. In *IEEE Pacific Visualization Symposium*, pp. 217–224. IEEE, 2013. doi: 10.1109/PacificVis.2013.6596148

[27] M. Khoury, Y. Hu, S. Krishnan, and C. Scheidegger. Drawing large graphs by low-rank stress majorization. *Computer Graphics Forum*, 31(3pt1):975–984, 2012. doi: 10.1111/j.1467-8659.2012.03090.x

[28] S. G. Kobourov. Force-directed drawing algorithms. 2004.

[29] 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

[30] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. doi: 10.1038/nature14539

[31] A. Noack. An energy model for visual graph clustering. In *International symposium on graph drawing*, pp. 425–436. Springer, 2003. doi: 10.1007/978-3-540-24595-7_40

[32] M. Ortmann, M. Klimenta, and U. Brandes. A sparse stress model. In *International Symposium on Graph Drawing and Network Visualization*, pp. 18–32. Springer, 2016. doi: 10.1007/978-3-319-50106-2_2

[33] T. P. Peixoto. The graph-tool python library. *figshare*, 2014. doi: 10.6084/m9.figshare.1164194

[34] H. C. Purchase. Metrics for graph drawing aesthetics. *Journal of Visual Languages & Computing*, 13(5):501–516, 2002. doi: 10.1006/jvlc.2002.0232

[35] R. A. Rossi and N. K. Ahmed. The network data repository with interactive graph analytics and visualization. *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 29(1):4292–4293, 2015. doi: 10.1609/aaai.v29i1.9277

[36] R. Tamassia. *Handbook of Graph Drawing and Visualization*. CRC press, 2013.

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

[38] F. Van Ham and B. Rogowitz. Perceptual organization in user-generated graph layouts. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1333–1339, 2008. doi: 10.1109/TVCG.2008.155

[39] C. Walshaw. A multilevel algorithm for force-directed graph drawing. In *International Symposium on Graph Drawing*, pp. 171–182. Springer, 2000. doi: 10.1142/9789812773296_0012

[40] Y. Wang, Y. Wang, Y. Sun, L. Zhu, K. Lu, C.-W. Fu, M. Sedlmair, O. Deussen, and B. Chen. Revisiting stress majorization as a unified framework for interactive constrained graph visualization. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):489–499, 2017. doi: 10.1109/TVCG.2017.2745919

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

[42] J. X. Zheng, S. Pawar, and D. F. Goodman. Graph drawing by stochastic gradient descent. *IEEE Transactions on Visualization and Computer Graphics*, 25(9):2738–2748, 2018. doi: 10.1109/TVCG.2018.2859997

[43] 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