Lines: 374

Graph Drawing via Gradient Descent, $(GD)^2$

Reyan Ahmed, Felice De Luca, Sabin Devkota, Stephen Kobourov, Mingwei Li

Department of Computer Science, University of Arizona, USA

Abstract. Readability criteria, such as distance or neighborhood preservation, are often used to optimize node-link representations of graphs to enable the comprehension of the underlying data. With few exceptions, graph drawing algorithms typically optimize one such criterion, usually at the expense of others. We propose a layout approach, Graph Drawing via Gradient Descent, $(GD)^2$, that can handle multiple readability criteria. $(GD)^2$ can optimize any criterion that can be described by a smooth function. If the criterion cannot be captured by a smooth function, a non-smooth function for the criterion is combined with another smooth function, or auto-differentiation tools are used for the optimization. Our approach is flexible and can be used to optimize several criteria that have already been considered earlier (e.g., obtaining ideal edge lengths, stress, neighborhood preservation) as well as other criteria which have not yet been explicitly optimized in such fashion (e.g., vertex resolution, angular resolution, aspect ratio). We provide quantitative and qualitative evidence of the effectiveness of $(GD)^2$ with experimental data and a functional prototype: http://hdc.cs.arizona.edu/~mwli/graph-drawing/.

1 Introduction

Graphs represent relationships between entities and visualization of this information is relevant in many domains. Several criteria have been proposed to evaluate the readability of graph drawings, including the number of edge crossings, distance preservation, and neighborhood preservation. Such criteria evaluate different aspects of the drawing and different layout algorithms optimize different criteria. It is challenging to optimize multiple readability criteria at once and there are few approaches that can support this. Examples of approaches that can handle a small number of related criteria include the stress majorization framework of Wang et al. [34], which optimizes distance preservation via stress as well as ideal edge length preservation. The Stress Plus X (SPX) framework of Devkota et al. [12] can minimize the number of crossings, or maximize the minimum angle of edge crossings. While these frameworks can handle a limited set of related criteria, it is not clear how to extend them to arbitrary optimization goals. The reason for this limitation is that these frameworks are dependent on a particular mathematical formulation. For example, the SPX framework was designed for crossing minimization, which can be easily modified to handle crossing angle maximization (by adding a cosine factor to the optimization function). This "trick" can be applied only to a limited set of criteria but not the majority of other criteria that are incompatible with the basic formulation.

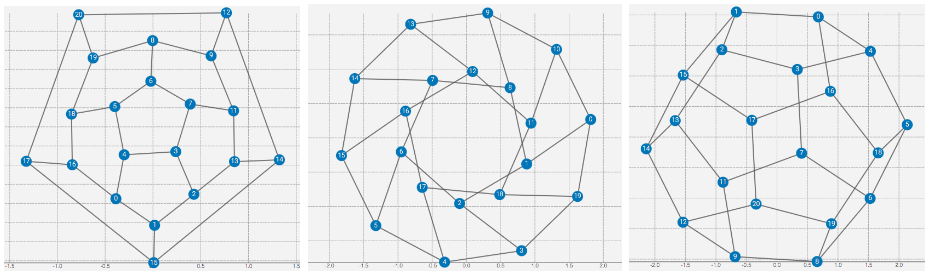

Fig. 1. Three $(GD)^2$ layouts of the dodecahedron: (a) optimizing the number of crossings, (b) optimizing uniform edge lengths, and (c) optimizing stress.

In this paper, we propose a general approach, Graph Drawing via Gradient Descent, $(GD)^2$, that can optimize a large set of drawing criteria, provided that the corresponding metrics that evaluate the criteria are smooth functions. If the function is not smooth, $(GD)^2$ either combines it with another smooth function and partially optimizes based on the desired criterion, or uses modern auto-differentiation tools to optimize. As a result, the proposed $(GD)^2$ framework is simple: it only requires a function that captures a desired drawing criterion. To demonstrate the flexibility of the approach, we consider an initial set of nine criteria: minimizing stress, maximizing vertex resolution, obtaining ideal edge lengths, maximizing neighborhood preservation, maximizing crossing angle, optimizing total angular resolution, minimizing aspect ratio, optimizing the Gabriel graph property, and minimizing edge crossings. A functional prototype is available on http://hdc.cs.arizona.edu/~mwli/graph-drawing/. This is an interactive system that allows vertices to be moved manually. Combinations of criteria can be optimized by selecting a weight for each; see Figure 1.

2 Related Work

Many criteria associated with the readability of graph drawings have been proposed [35]. Most of graph layout algorithms are designed to (explicitly or implicitly) optimize a single criterion. For instance, a classic layout criterion is stress minimization [24], where stress is defined by $\sum_{i<j} w_{ij}(|X_i - X_j| - d_{ij})^2$. Here, X is a $n \times 2$ matrix containing coordinates for the n nodes, d_{ij} is typically the graph-theoretical distance between two nodes i and j and $w_{ij} = d_{ij}^{-\alpha}$ is a normalization factor with α equal to $0, 1$ or 2. Thus reducing the stress in a layout corresponds to computing node positions so that the actual distance between pairs of nodes is proportional to the graph theoretic distance between them. Optimizing stress can be accomplished by stress minimization, or stress majorization, which can speed up the computation [20]. In this paper we only consider drawing in the Euclidean plane, however, stress can be also optimized in other spaces such as the torus [8].

Stress minimization corresponds to optimizing the global structure of the layout, as the stress metric takes into account all pairwise distances in the graph. The t-SNET algorithm of Kruiger et al. [25] directly optimizes neighborhood preservation, which captures the local structure of a graph, as the neighborhood preservation metric only considers distances between pairs of nodes that are close to each other. Optimizing local or global distance preservation can be seen as special cases of the more general dimensionality reduction approaches such as multi-dimensional scaling [26, 32].

Purchase et al. [28] showed that the readability of graphs increases if a layout has fewer edge crossings. The underlying optimization problem is NP-hard and several graph drawing contests have been organized with the objective of minimizing the number of crossings in the graph drawings [2,7]. Recently several algorithms that directly minimize crossings have been proposed [29, 31].

The negative impact on graph readability due to edge crossings can be mitigated if crossing pairs of edges have a large crossings angle [3,13,22,23]. Formally, the crossing angle of a straight-line drawing of a graph is the minimum angle between two crossing edges in the layout, and optimizing this property is also NP-hard. Recent graph drawing contests have been organized with the objective of maximizing the crossings angle in graph drawings and this has led to several heuristics for this problem [4, 10].

The algorithms above are very effective at optimizing the specific readability criterion they are designed for, but they cannot be directly used to optimize additional criteria. This is a desirable goal, since optimizing one criterion often leads to poor layouts with respect to one or more other criteria: for example, algorithms that optimize the crossing angle tend to create drawings with high stress and no neighborhood preservation [12].

Recently, several approaches have been proposed to simultaneously improve multiple layout criteria. Wang et al. [34] propose a revised formulation of stress that can be used to specify ideal edge direction in addition to ideal edge lengths in a graph drawing. Devkota et al. [12] also use a stress-based approach to minimize edge crossings and maximize crossing angles. Eades et al. [17] provided a technique to draw large graphs while optimizing different geometric criteria, including the Gabriel graph property. Although the approaches above are designed to optimize multiple criteria, they cannot be naturally extended to handle other optimization goals.

Constraint-based layout algorithms such as COLA [15, 16], can be used to enforce separation constraints on pairs of nodes to support properties such as customized node ordering or downward pointing edges. The coordinates of two nodes are related by inequalities in the form of $x_i \geq x_j + gap$ for a node pair (i, j). These kinds of constraints are known as hard constraints and are different from the soft constrains in our $(GD)^2$ framework.

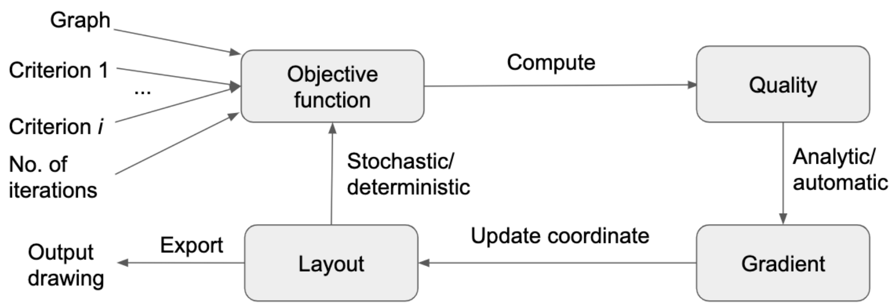

Fig. 2. The $(GD)^2$ framework: Given a graph and a set of criteria (with weights), formulate an objective function based on the selected set of criteria and weights. Then compute the quality (value) of the objective function of the current layout of the graph. Next, generate the gradient (analytically or automatically). Using the gradient information, update the coordinates of the layout. Finally, update the objective function based on the layout via regular or stochastic gradient descent. This process is repeated for a fixed number of iterations.

3 The $(GD)^2$ Framework

The $(GD)^2$ framework is a general optimization approach to generate a layout with any desired set of aesthetic metrics, provided that they can be expressed by a smooth function. The basic principles underlying this framework are simple. The first step is to select a set of layout readability criteria and a loss functions that measures them. Then we define the function to optimize as a linear combination of the loss functions for each individual criterion. Finally, we iterate the gradient descent steps, from which we obtain a slightly better drawing at each iteration. Figure 2 depicts the framework of $(GD)^2$: Given any graph G and readability criterion Q, we find a loss function $L_{Q,G}$ which maps from the current layout X (i.e. a $n \times 2$ matrix containing the positions of nodes in the drawing) to a real value that quantifies the current drawing. Note that some of the readability criteria naturally correspond to functions that should be minimized (e.g., stress, crossings), while others to functions that should be maximized (e.g., neighborhood preservation, angular resolution). Given a loss function $L_{Q,G}$ of X where a lower value is always desirable, at each iteration, a slightly better layout can be found by taking a small (ϵ) step along the (negative) gradient direction: $X^{(new)} = X - \epsilon \cdot \nabla_X \, L_{Q,G}$.

To optimize multiple quality measures simultaneously, we take a weighted sum of their loss functions and update the layout by the gradient of the sum.

3.1 Gradient Descent Optimization

There are different kinds of gradient descent algorithms. The standard method considers all vertices, computes the gradient of the objective function, and updates vertex coordinates based on the gradient. For some objectives, we need to consider all the vertices in every step. For example, the basic stress formulation [24] falls in this category. On the other hand, there are some problems where the objective can be optimized only using a subset of vertices. For example, consider stress minimization again. If we select a set of vertices randomly and minimize the stress of the induced graph, the stress of the whole graph is also minimized [36]. This type of gradient descent is called stochastic gradient descent. However, not all objective functions are smooth and we cannot compute the gradient of a non-smooth function. In that scenario, we can compute the subgradient, and update the objective based on the subgradient. Hence, as long as the function is continuously defined on a connected component in the domain, we can apply the subgradient descent algorithm. In table 3, we give a list of loss functions we used to optimize 9 graph drawing properties with gradient descent variants. In section 4, we specify the loss functions we used in detail.

When a function is not defined in a connected domain, we can introduce a surrogate loss function to 'connect the pieces'. For example, when optimizing neighborhood preservation we maximize the Jaccard similarity between graph neighbors and nearest neighbors in graph layout. However, Jaccard similarity is only defined between two binary vectors. To solve this problem we extend Jaccard similarity to all real vectors by its Lovász extension [5] and apply that to optimize neighborhood preservation. An essential part of gradient descent based algorithms is to compute the gradient/subgradient of the objective function. In practice, it is always not necessary to write down the gradient analytically as it can be computed automatically via automatic differentiation [21]. Deep learning packages such as Tensorflow [1] and PyTorch [27] apply automatic differentiation to compute the gradient of complicated functions.

When optimizing multiple criteria simultaneously, we combine them via a weighted sum. However, choosing a proper weight for each criterion can be tricky. Consider, for example, maximizing crossing angles and minimize stress simultaneously with a fixed pair of weights. At the very early stage, the initial drawing may have many crossings and stress minimization often removes most of the early crossings. As a result, maximizing crossing angles in those early stages can be harmful as moves nodes in direction that contradict those that come from stress minimization. Therefore, a well-tailored *weight scheduling* is needed for a successful outcome. Continuing with the same example, a better outcome can be achieved by first optimizing stress until it converges, and later adding weights for the crossing angle maximization. To explore different ways of scheduling, we provide an interface that allows manual tuning of the weights.

3.2 Implementation

We implemented the $(GD)^2$ framework in JavaScript. In particular we used the automatic differentiation tools in tensorflow.js [33] and the drawing library d3.js [6]. The prototype is available at http://hdc.cs.arizona.edu/~mwli/graph-drawing/.

4 Properties and Measures

In this section we specify the aesthetic goals, definitions, quality measures and loss functions for each of the 9 graph drawing properties we optimized: stress, vertex resolution, edge uniformity, neighborhood preservation, crossing angle, aspect ratio, total angular resolution, Gabriel graph property, and crossing number. In the following discussion, since only one (arbitrary) graph is considered, we omit the subscript G in our definitions of loss function $L_{Q,G}$ and write L_Q for short. Other standard graph notation is summarized in Table 1.

Notation	Description
G	Graph
V	The set of nodes in G, indexed by i, j or k
E	The set of edges in G, indexed by a pair of nodes (i, j) in V
$n = \|V\|$	Number of nodes in G
$\|E\|$	Number of edges in G
$Adj_{n\times n}$ and $A_{i,j}$	Adjacency matrix of G and its (i, j)-th entry
$D_{n\times n}$ and d_{ij}	Graph-theoretic distances between pairs of nodes and the (i, j)-th entry
$X_{n\times 2}$	2D-coordinates of nodes in the drawing
$\|\|X_i - X_j\|\|$	The Euclidean distance between nodes i and j in the drawing
θ_i	i^{th} crossing angle
φ_{ijk}	Angle between incident edges (i, j) and (j, k)

Table 1. Graph notation used in this paper.

4.1 Stress

We use stress minimization to draw a graph such that the Euclidean distance between pairs of nodes is proportional to their graph theoretic distance. Following the ordinary definition of stress [24], we minimize

$$L_{ST} = \sum_{i<j} w_{ij}(|X_i - X_j|_2 - d_{ij})^2 \tag{1}$$

Where d_{ij} is the graph-theoretical distance between nodes i and j, X_i and X_j are the 2D coordinates of nodes i and j in the layout. The normalization factor, $w_{ij} = d_{ij}^{-2}$, balances the influence of short and long distances: the longer the graph theoretic distance, the more tolerance we give to the discrepancy between two distances. When comparing two drawings of the same graph with respect to stress, a smaller value (lower bounded by 0) corresponds to a better drawing.

4.2 Ideal Edge Length

When given a set of ideal edge lengths $\{l_{ij} : (i,j) \in E\}$ we minimize the average deviation from the ideal lengths:

$$L_{IL} = \sqrt{\frac{1}{|E|} \sum_{(i,j)\in E} \left(\frac{||X_i - X_j|| - l_{ij}}{l_{ij}}\right)^2} \tag{2}$$

For unweighted graphs, by default we take the average edge length in the current drawing as the ideal edge length for all edges. $l_{ij} = l_{avg} = \frac{1}{|E|} \sum_{(i,j)\in E} ||X_i - X_j||$ for all $(i,j) \in E$. The quality measure $Q_{IL} = L_{IL}$ is lower bounded by 0 and a lower score yields a better layout.

4.3 Neighborhood Preservation

Neighborhood preservation aims to keep adjacent nodes close to each other in the layout. Similar to Kruiger et al. [25], the idea is to have the k-nearest (Euclidean) neighbors (k-NN) of node i in the drawing to align with the k nearest nodes (in terms of graph distance from i). A natural quality measure for the alignment is the Jaccard index between the two pieces of information. Let, $Q_{NP} = JaccardIndex(K, Adj) = \frac{|\{(i,j):K_{ij}=1 \text{ and } A_{ij}=1\}|}{|\{(i,j):K_{ij}=1 \text{ or } A_{ij}=1\}|}$, where Adj denotes the adjacency matrix and the i-th row in K denotes the k-nearest neighborhood information of i: $K_{ij} = 1$ if j is one of the k-nearest neighbors of i and $K_{ij} = 0$ otherwise.

To express the Jaccard index as a differentiable minimization problem, first, we express the neighborhood information in the drawing as a smooth function of node positions X_i and store it in a matrix $\hat{K}$. In $\hat{K}$, a positive entry $\hat{K}_{i,j}$ means node j is one of the k-nearest neighbors of i, otherwise the entry is negative. Next, we take a differentiable surrogate function of the Jaccard index, the Lovász hinge loss (LHL) [5], to make the Jaccard loss optimizable via gradient descent. We minimize

$$L_{NP} = LHL(\hat{K}, Adj) \tag{3}$$

where LHL is given by Berman et al. [5], $\hat{K}$ denotes the k-nearest neighbor prediction:

$$\hat{K}_{i,j} = \begin{cases} -(||X_i - X_j|| - \frac{d_{i,\pi_k} + d_{i,\pi_{k+1}}}{2}) & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases} \tag{4}$$

where d_{i,π_k} is the Euclidean distance between node i and its k^{th} nearest neighbor and Adj denotes the adjacency matrix. Note that $\hat{K}_{i,j}$ is positive if j is a k-NN of i, otherwise it is negative, as is required by LHL [5].

4.4 Crossing Number

Reducing the number of edge crossings is one of the classic optimization goals in graph drawing, known to affect readability [28]. Following Shabbeer et al. [31], we employ an expectation-maximization (EM)-like algorithm to minimize the number of crossings. Two edges do not cross if and only if there exists a line that separate their extreme points. With this in mind, we want to separate every pair of edges (the M step) and use the decision boundaries to guide the movement of nodes in the drawing (the E step). Formally, given any two edges $e_1 = (i,j), e_2 = (k,l)$ that do not share any nodes (i.e., i, j, k and l are all distinct), they do not intersect in a drawing (where nodes are drawn at $X_i = (x_i, y_i)$, a row vector) if and only if there exists a decision boundary $w = w_{(e_1,e_2)}$ (a 2-by-1 column vector) together with a bias $b = b_{(e_1,e_2)}$ (a scalar) such that: $L_{CN,(e_1,e_2)} = \sum_{\alpha=i,j,k \text{ or } l} ReLU(1 - t_\alpha \cdot (X_\alpha w + b)) = 0$.

Here we use (e_1, e_2) to denote the subgraph of G which only has two edges e_1 and e_2, $t_i = t_j = 1$ and $t_k = t_l = -1$. The loss reaches its minimum at 0 when the SVM classifier $f_{w,b} : x \mapsto xw + b$ predicts node i and j to be greater than 1 and node k and l to be less than -1. The total loss for the crossing number is therefore the sum over all possible pairs of edges. Similar to (soft) margin SVM, we add a term $|w_{(e_1,e_2)}|^2$ to maximize the margin of the decision boundary: $L_{CN} = \sum_{\substack{e_1=(i,j),\ e_2=(k,l)\in E \\ i,\ j,\ k \text{ and } l \text{ all distinct}}} L_{CN,(e_1,e_2)} + |w_{(e_1,e_2)}|^2$. For the E and M steps, we used the same loss function L_{CN} to update the boundaries $w_{(e_1,e_2)}, b_{(e_1,e_2)}$ and node positions X:

$$
\begin{aligned}
w^{(new)} &= w - \epsilon \nabla_w L_{CN} && \text{(M step 1)} \\
b^{(new)} &= b - \epsilon \nabla_b L_{CN} && \text{(M step 2)} \\
X^{(new)} &= X - \epsilon \nabla_X L_{CN}(X;\ w^{(new)}, b^{(new)}) && \text{(E step)}
\end{aligned}
$$

To evaluate the quality we simply count the number of crossings.

4.5 Crossing Angle Maximization

When edge crossings are unavoidable, the graph drawing can still be easier to read when edges cross at angles close to 90 degrees [35]. Heuristics such as those by Demel et al. [10] and Bekos et al. [4] have been proposed and have been successful in graph drawing challenges [11]. We use an approach similar to the force-directed algorithm given by Eades et al. [18] and minimize the squared cosine of crossing angles: $L_{CAM} = \sum_{\substack{\text{all crossed edge pairs} \\ (i,j),(k,l)\in E}} (\frac{\langle X_i - X_j, X_k - X_l \rangle}{|X_i - X_j| \cdot |X_k - X_l|})^2$. We evaluate quality by measuring the worst (normalized) absolute discrepancy between each crossing angle θ and the target crossing angle (i.e. 90 degrees): $Q_{CAM} = \max_\theta |\theta - \frac{\pi}{2}| / \frac{\pi}{2}$.

4.6 Aspect Ratio

Good use of drawing area is often measured by the aspect ratio [14] of the bounding box of the drawing, with 1 : 1 as the optimum. We consider multiple rotations of the current drawing and optimize their bounding boxes simultaneously. Let $AR = \min_\theta \frac{\min(w_\theta, h_\theta)}{\max(w_\theta, h_\theta)}$, where w_θ and h_θ denote the width and height of the bounding box when the drawing is rotated by θ degrees. A naive approach to optimize aspect ratio, which scales the x and y coordinates of the drawing by certain factors, may worsen other criteria we wish to optimize and is therefore not suitable for our purposes. To make aspect ratio differentiable and compatible with other objectives, we approximate aspect ratio based on 4 (soft) boundaries (top, bottom, left and right) of the drawing. Next, we turn this approximation and the target (1 : 1) into a loss function using cross entropy loss. We minimize

$$L_{AR} = \sum_{\theta \in \{\frac{2\pi k}{N}, \text{ for } k=0,\cdots(N-1)\}} crossEntropy([\frac{w_\theta}{w_\theta + h_\theta}, \frac{h_\theta}{w_\theta + h_\theta}], [0.5, 0.5]) \tag{5}$$

where N is the number of rotations sampled (e.g., $N = 7$), and w_θ, h_θ are the (approximate) width and height of the bounding box when rotating the drawing around its center by an angle θ. For any given θ-rotated drawing, w_θ is defined to be the difference between the current (soft) right and left boundaries, $w_\theta = \text{right} - \text{left} = \langle \text{softmax}(x_\theta), x_\theta \rangle - \langle \text{softmax}(-x_\theta), x_\theta \rangle$, where x_θ is a collection of the x coordinates of all nodes in the θ-rotated drawing, and softmax returns a vector of weights $(\dots w_k, \dots)$ given by $\text{softmax}(x) = (\dots w_k, \dots) = \frac{e^{x_k}}{\sum_i e^{x_i}}$. Note that the approximate right boundary is a weighted sum of the x coordinates of all nodes and it is designed to be close to the x coordinate of the right-most node, while keeping other nodes involved. Optimizing aspect ratio with the softened boundaries will stretch all nodes instead of moving the extreme points. Similarly, $h_\theta = \text{top} - \text{bottom} = \langle \text{softmax}(y_\theta), y_\theta \rangle - \langle \text{softmax}(-y_\theta), y_\theta \rangle$ Finally, we evaluate the drawing quality by measuring the worst aspect ratio on a finite set of rotations. The quality score ranges from 0 to 1 (where 1 is optimal): $Q_{AR} = \min_{\theta \in \{\frac{2\pi k}{N}, \text{ for } k=0,\cdots(N-1)\}} \frac{\min(w_\theta, h_\theta)}{\max(w_\theta, h_\theta)}$

4.7 Angular Resolution

Distributing edges adjacent to a node makes it easier to perceive the information presented in a node-link diagram [23]. Angular resolution [3], defined as the minimum angle between incident edges, is one way to quantify this goal. Formally, $ANR = \min_{j \in V} \min_{(i,j),(j,k) \in E} \varphi_{ijk}$, where φ_{ijk} is the angle formed by between edges (i, j) and (j, k). Note that for any given graph, an upper bound of this quantity is $\frac{2\pi}{d_{max}}$ where d_{max} is the maximum degree of nodes in the graph. Therefore in the evaluation, we will use this upper bound to normalize our quality measure to $[0, 1]$, i.e. $Q_{ANR} = \frac{ANR}{2\pi/d_{max}}$. To achieve a better drawing quality via gradient descent, we define the angular energy of an angle φ to be $e^{-s\cdot\varphi}$, where s is a constant controlling the sensitivity of angular energy with respect to the angle (by default $s = 1$), and minimize the total angular energy over all incident edges:

$$L_{ANR} = \sum_{(i,j),(j,k)\in E} e^{-s\cdot\varphi_{ijk}} \tag{6}$$

4.8 Vertex Resolution

Good vertex resolution is associated with the ability to distinguish different vertices by preventing nodes from occluding each other. Vertex resolution is typically defined as the minimum Euclidean distance between two vertices in the drawing [9,30]. However, in order to align with the units in other objectives such as stress, we normalize the minimum Euclidean distance with respect to a reference value. Hence we define the vertex resolution to be the ratio between the shortest and longest distances between pairs of nodes in the drawing, $VR = \frac{\min_{i\neq j}||X_i - X_j||}{d_{max}}$, where $d_{max} = \max_{k,l}||X_k - X_l||$. To achieve a certain target resolution $r \in [0,1]$ by minimizing a loss function, we minimize

$$L_{VR} = \sum_{i,j\in V, i\neq j} ReLU(1 - \frac{||X_i - X_j||}{r \cdot d_{max}})^{\,2} \tag{7}$$

In practice, we set the target resolution to be $r = \frac{1}{\sqrt{|V|}}$, where $|V|$ is the number of vertices in the graph. In this way, an optimal drawing will distribute nodes uniformly in the drawing area. In the evaluation, we report, as a quality measure, the ratio between the actual and target resolution and cap its value between 0 (worst) and 1 (best).

$$Q_{VR} = \min(1.0, \frac{\min_{i,j}||X_i - X_j||}{r \cdot d_{max}}) \tag{8}$$

4.9 Gabriel Graph Property

A graph is a Gabriel graph if it can be drawn in such a way that any disk formed by using an edge in the graph as its diameter contains no other nodes. Not all graphs are Gabriel graphs, but drawing a graph so that as many of these edge-based disks are empty of other nodes has been associated with good readability [17]. This property can be enforced by a repulsive force around the midpoints of edges. Formally, we establish a repulsive field with radius r_{ij} equal to half of the edge length, around the midpoint c_{ij} of each edge $(i,j) \in E$, and we minimize the total potential energy:

$$L_{GA} = \sum_{\substack{(i,j)\in E,\\ k\in V\setminus\{i,j\}}} ReLU(r_{ij} - |X_k - c_{ij}|)^{\,2} \tag{9}$$

where $c_{ij} = \frac{X_i+X_j}{2}$ and $r_{ij} = \frac{|X_i-X_j|}{2}$. We use the (normalized) minimum distance from nodes to centers to characterize the quality of a drawing with respect to Gabriel graph property: $Q_{GA} = \min_{(i,j)\in E, k\in V} \frac{|X_k-c_{ij}|}{r_{ij}}$.

5 Experimental Evaluation

In this section, we describe the experiment we conducted on 10 graphs to assess the effectiveness and limitations of our approach. The graphs used are depicted in Figure 3 along with information about each graph. The graphs have been chosen to represent a variety of graph classes such as trees, cycles, grids, bipartite graphs, cubic graphs, and symmetric graphs.

In our experiment we compare $(GD)^2$ with neato [19] and sfdp [19], which are classical implementations of a stress-minimization layout and scalable force-directed layout. In particular, we focus on 9 readability criteria: stress (`ST`), vertex resolution (`VR`), ideal edge lengths (`IL`), neighbor preservation (`NP`), crossing angle (`CA`), angular resolution (`ANR`), aspect ratio (`AR`), Gabriel graph properties (`GG`), and crossings (`CR`). We provide the values of the nine criteria corresponding to the 10 graphs for the layouts computed by by neato, sfdp, random, and 3 runs of $(GD)^2$ initialized with neato, sfdp, and random layouts in Table 2. Bold values are the best. Green cells show an improvement, yellow cells show a tie, with respect to the initial values.

In this experiment, we focused on optimizing a single metric. In some applications, it is desirable to optimize multiple criteria. We can use a similar technique i.e., take a weighted sum of the metrics and optimize the sum of scores. In the prototype (http://hdc.cs.arizona.edu/~mwli/graph-drawing/), there is a slider for each criterion, making it possible to combine different criteria.

6 Limitations

Although $(GD)^2$ is a flexible framework that can optimize a wide range of criteria, it cannot handle the class of constraints where the node coordinates are related by some inequalities, i.e., the framework does not support hard constraints. Similarly, this framework does not naturally support shape-based drawing constraints such as those in [15, 16, 34]. $(GD)^2$ takes under a minute for the small graphs considered in this paper. We have not experimented with larger graphs as the implementation has not been optimized for speed.

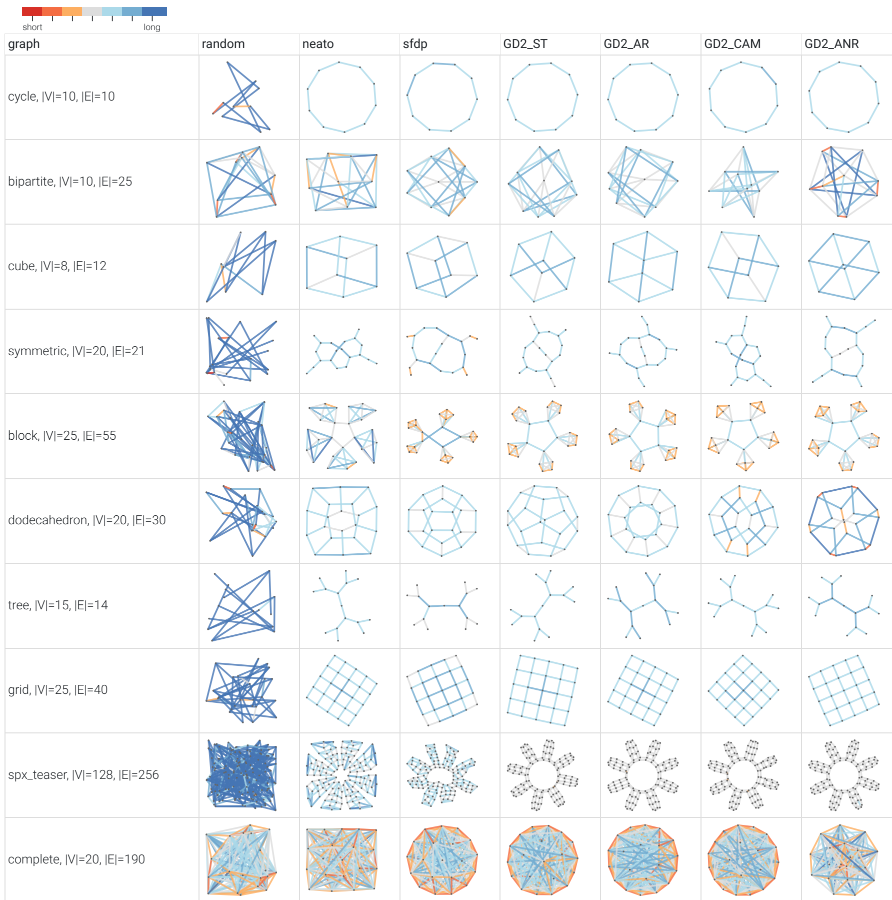

Fig. 3. Drawings from different algorithms: neato, sfdp and $(GD)^2$ with stress (ST), aspect ratio (AR), crossing angle maximization (CAM) and angular resolution (ANR) optimization on a set of 10 graphs. Edge color is determined by the discrepancy between actual and ideal edge length (here all ideal edge lengths are 1); informally, short edges are red and long edges are blue.

7 Conclusions and Future Work

We introduced the graph drawing framework $(GD)^2$ and showed how this approach can be used to optimize different graph drawing criteria and combinations thereof. The framework is flexible and natural directions for future work include adding further drawing criteria and better ways to combine them. To compute the layout of large graphs, a multi-level algorithmic model might be needed.

Crossings	neato	sdfp	rnd	$(GD)^2_n$	$(GD)^2_s$	$(GD)^2_r$
dodec.	**6.0**	**6.0**	79.0	**6.0**	**6.0**	10.0
cycle	**0.0**	**0.0**	11.0	**0.0**	**0.0**	**0.0**
tree	**0.0**	**0.0**	31.0	**0.0**	**0.0**	**0.0**
block	23.0	**16.0**	297.0	23.0	**16.0**	25.0
compl.	**3454**	3571	3572	**3454**	3571	3572
cube	**2.0**	**2.0**	18.0	**2.0**	**2.0**	**2.0**
symme.	1.0	**0.0**	77.0	1.0	**0.0**	**0.0**
bipar.	**40.0**	52.0	**40.0**	**40.0**	**40.0**	**40.0**
grid	**0.0**	**0.0**	190.0	**0.0**	**0.0**	**0.0**
spx t.	73.0	**71.0**	7254.0	73.0	**71.0**	76.0

Ideal edge length	neato	sdfp	rnd	$(GD)^2_n$	$(GD)^2_s$	$(GD)^2_r$
dodec.	0.14	0.15	0.53	0.1	0.15	**0.08**
cycle	**0.0**	**0.0**	0.42	**0.0**	**0.0**	**0.0**
tree	**0.03**	0.13	0.31	**0.03**	0.04	0.09
block	0.31	0.43	0.5	**0.25**	0.33	0.31
compl.	0.42	**0.41**	0.45	**0.41**	**0.41**	**0.41**
cube	0.08	0.12	0.29	0.03	**0.0**	0.12
symme.	0.08	0.19	0.46	0.07	0.05	**0.04**
bipar.	0.31	0.26	0.44	0.16	0.13	**0.1**
grid	0.01	0.09	0.41	**0.0**	**0.0**	0.01
spx t.	0.4	0.32	0.45	0.3	**0.2**	0.32

Stress	neato	sdfp	rnd	$(GD)^2_n$	$(GD)^2_s$	$(GD)^2_r$
dodec.	21.4	17.58	111.05	**17.45**	17.58	17.6
cycle	**0.77**	**0.77**	30.24	**0.77**	**0.77**	**0.77**
tree	**2.11**	2.7	98.49	**2.11**	2.62	5.5
block	26.79	28.22	203.31	12.72	23.71	**11.2**
compl.	33.54	31.58	37.87	31.53	31.49	**31.47**
cube	2.75	2.71	11.69	2.66	2.69	**2.65**
symme.	9.88	5.38	180.48	9.88	**3.36**	3.97
bipar.	9.25	**8.5**	12.48	8.52	**8.5**	9.6
grid	**6.77**	7.38	221.66	**6.77**	6.78	**6.77**
spx t.	674.8	418.4	9794	**227.1**	235.3	227.2

Angular resolution	neato	sdfp	rnd	$(GD)^2_n$	$(GD)^2_s$	$(GD)^2_r$
dodec.	0.39	0.39	0.01	**0.6**	0.39	**0.6**
cycle	**0.8**	**0.8**	0.05	**0.8**	**0.8**	**0.8**
tree	0.61	0.56	0.04	0.78	0.83	**0.88**
block	0.05	0.01	0.0	**0.36**	0.02	0.29
compl.	0.0	**0.01**	0.0	0.0	**0.01**	0.0
cube	0.28	0.3	0.01	**0.46**	0.44	0.4
symme.	0.66	0.6	0.03	0.68	0.76	**0.77**
bipar.	0.01	0.03	0.01	0.02	0.04	**0.11**
grid	0.52	**0.54**	0.0	0.52	**0.54**	0.52
spx t.	0.02	0.0	0.0	**0.03**	0.0	0.0

Neighbor preservation	neato	sdfp	rnd	$(GD)^2_n$	$(GD)^2_s$	$(GD)^2_r$
dodec.	0.32	0.3	0.1	**0.5**	0.3	**0.5**
cycle	**1.0**	**1.0**	0.08	**1.0**	**1.0**	**1.0**
tree	**1.0**	**1.0**	0.02	**1.0**	**1.0**	**1.0**
block	0.57	0.93	0.12	0.83	0.93	**1.0**
compl.	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
cube	**0.5**	**0.5**	0.12	**0.5**	**0.5**	**0.5**
symme.	0.75	0.95	0.05	0.75	**1.0**	**1.0**
bipar.	**0.47**	**0.47**	0.43	**0.47**	**0.47**	0.43
grid	**1.0**	**1.0**	0.05	**1.0**	**1.0**	**1.0**
spx t.	0.36	0.44	0.03	0.49	0.46	**0.53**

Gabriel graph property	neato	sdfp	rnd	$(GD)^2_n$	$(GD)^2_s$	$(GD)^2_r$
dodec.	0.16	**0.64**	0.07	0.32	**0.64**	0.32
cycle	**1.0**	**1.0**	0.29	**1.0**	**1.0**	**1.0**
tree	**1.0**	**1.0**	0.05	**1.0**	**1.0**	**1.0**
block	0.16	0.03	0.04	0.57	0.14	**0.59**
compl.	0.0	0.01	0.02	0.04	0.01	**0.07**
cube	0.43	0.51	0.01	0.75	**0.8**	0.71
symme.	0.54	**1.0**	0.15	0.7	**1.0**	**1.0**
bipar.	0.08	0.11	0.25	0.48	0.64	**0.74**
grid	**1.0**	**1.0**	0.03	**1.0**	**1.0**	**1.0**
spx t.	0.04	0.0	0.02	0.06	**0.08**	**0.08**

Vertex resolution	neato	sdfp	rnd	$(GD)^2_n$	$(GD)^2_s$	$(GD)^2_r$
dodec.	0.52	0.54	0.07	0.7	**0.81**	0.68
cycle	**0.98**	**0.98**	0.32	**0.98**	**0.98**	**0.98**
tree	0.68	0.57	0.23	**0.69**	0.68	0.68
block	0.66	0.38	0.1	**0.72**	0.59	0.51
compl.	0.8	**1.0**	0.18	0.84	**1.0**	0.91
cube	0.66	**0.82**	0.11	0.66	**0.82**	0.67
symme.	0.35	0.43	0.06	0.38	0.51	**0.6**
bipar.	0.83	**0.87**	0.21	0.83	**0.87**	0.35
grid	0.87	0.8	0.08	**0.88**	**0.88**	**0.88**
spx t.	0.47	**0.48**	0.05	0.47	**0.48**	0.32

Aspect ratio	neato	sdfp	rnd	$(GD)^2_n$	$(GD)^2_s$	$(GD)^2_r$
dodec.	0.92	0.91	0.88	**0.96**	**0.96**	**0.96**
cycle	**0.96**	0.95	0.67	**0.96**	0.95	**0.96**
tree	0.73	0.67	**0.88**	0.86	0.76	**0.88**
block	0.9	0.74	0.7	**0.96**	0.9	**0.96**
compl.	0.89	0.97	0.91	**0.98**	**0.98**	**0.98**
cube	0.76	0.79	0.57	0.87	0.79	**0.88**
symme.	0.58	0.67	**0.89**	0.6	0.67	**0.89**
bipar.	0.82	0.9	**0.91**	0.82	0.9	**0.91**
grid	**1.0**	**1.0**	0.82	**1.0**	**1.0**	**1.0**
spx t.	0.98	0.86	0.88	**0.99**	**0.99**	**0.99**

Crossing angle	neato	sdfp	rnd	$(GD)^2_n$	$(GD)^2_s$	$(GD)^2_r$
dodec.	**0.06**	0.12	0.24	**0.06**	0.09	0.15
cycle	**0.0**	**0.0**	0.19	**0.0**	**0.0**	**0.0**
tree	**0.0**	**0.0**	0.23	**0.0**	**0.0**	**0.0**
block	0.11	0.1	0.24	**0.05**	0.06	0.09
compl.	0.25	**0.24**	**0.24**	**0.24**	**0.24**	**0.24**
cube	**0.03**	**0.03**	0.21	**0.03**	**0.03**	0.04
symme.	0.03	**0.0**	0.24	0.03	**0.0**	**0.0**
bipar.	**0.16**	0.17	0.23	**0.16**	0.17	0.19
grid	**0.0**	**0.0**	0.23	**0.0**	**0.0**	**0.0**
spx t.	0.16	0.22	0.25	0.16	**0.15**	0.21

Table 2. The values of the nine criteria corresponding to the 10 graphs for the layouts computed by neato, sfdp, random, and 3 runs of $(GD)^2$ initialized with neato, sfdp, and random layouts. Bold values are the best. Green cells show an improvement, yellow cells show a tie, with respect to the initial values.

References

1. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al.: Tensorflow: A system for large-scale machine learning. In: 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI'16). pp. 265–283 (2016)
2. Ábrego, B.M., Fernández-Merchant, S., Salazar, G.: The rectilinear crossing number of k_n: Closing in (or are we?). Thirty Essays on Geometric Graph Theory (2012)
3. Argyriou, E.N., Bekos, M.A., Symvonis, A.: Maximizing the total resolution of graphs. In: Proceedings of the 18th International Conference on Graph Drawing. pp. 62–67. Springer (2011)
4. Bekos, M.A., Förster, H., Geckeler, C., Holländer, L., Kaufmann, M., Spallek, A.M., Splett, J.: A heuristic approach towards drawings of graphs with high crossing resolution. In: Proceedings of the 26th International Symposium on Graph Drawing and Network Visualization. pp. 271–285. Springer (2018)
5. Berman, M., Rannen Triki, A., Blaschko, M.B.: The lovász-softmax loss: a tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4413–4421 (2018)
6. Bostock, M., Ogievetsky, V., Heer, J.: D3: Data-driven documents. IEEE transactions on visualization and computer graphics **17**(12), 2301–2309 (2011)
7. Buchheim, C., Chimani, M., Gutwenger, C., Jünger, M., Mutzel, P.: Crossings and planarization. Handbook of Graph Drawing and Visualization pp. 43–85 (2013)
8. Chen, K.T., Dwyer, T., Marriott, K., Bach, B.: Doughnets: Visualising networks using torus wrapping. In: Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems. pp. 1–11 (2020)
9. Chrobak, M., Goodrich, M.T., Tamassia, R.: Convex drawings of graphs in two and three dimensions. In: Proceedings of the 12th annual symposium on Computational geometry. pp. 319–328 (1996)
10. Demel, A., Dürrschnabel, D., Mchedlidze, T., Radermacher, M., Wulf, L.: A greedy heuristic for crossing-angle maximization. In: Proceedings of the 26th International Symposium on Graph Drawing and Network Visualization. pp. 286–299. Springer (2018)
11. Devanny, W., Kindermann, P., Löffler, M., Rutter, I.: Graph drawing contest report. In: Proceedings of the 25th International Symposium on Graph Drawing and Network Visualization. pp. 575–582. Springer (2017)
12. Devkota, S., Ahmed, R., De Luca, F., Isaacs, K.E., Kobourov, S.: Stress-plus-x (spx) graph layout. In: Proceedings of the 27th International Symposium on Graph Drawing and Network Visualization. pp. 291–304. Springer (2019)
13. Didimo, W., Liotta, G.: The crossing-angle resolution in graph drawing. Thirty Essays on Geometric Graph Theory (2014)
14. Duncan, C.A., Goodrich, M.T., Kobourov, S.G.: Balanced aspect ratio trees and their use for drawing very large graphs. In: Proceedings of the 6th International Symposium on Graph Drawing. pp. 111–124. Springer (1998)
15. Dwyer, T.: Scalable, versatile and simple constrained graph layout. Comput. Graph. Forum **28**, 991–998 (2009)
16. Dwyer, T., Koren, Y., Marriott, K.: Ipsep-cola: An incremental procedure for separation constraint layout of graphs. IEEE transactions on visualization and computer graphics **12**, 821–8 (2006)

17. Eades, P., Hong, S.H., Klein, K., Nguyen, A.: Shape-based quality metrics for large graph visualization. In: Proceedings of the 23rd International Conference on Graph Drawing and Network Visualization. pp. 502–514. Springer (2015)
18. Eades, P., Huang, W., Hong, S.H.: A force-directed method for large crossing angle graph drawing. arXiv preprint arXiv:1012.4559 (2010)
19. Ellson, J., Gansner, E., Koutsofios, L., North, S.C., Woodhull, G.: Graphviz—open source graph drawing tools. In: Proceedings of the 9th International Symposium on Graph Drawing. pp. 483–484. Springer (2001)
20. Gansner, E.R., Koren, Y., North, S.: Graph drawing by stress majorization. In: International Symposium on Graph Drawing. pp. 239–250. Springer (2004)
21. Griewank, A., Walther, A.: Evaluating derivatives: principles and techniques of algorithmic differentiation, vol. 105. SIAM (2008)
22. Huang, W., Eades, P., Hong, S.H.: Larger crossing angles make graphs easier to read. Journal of Visual Languages & Computing **25**(4), 452–465 (2014)
23. Huang, W., Eades, P., Hong, S.H., Lin, C.C.: Improving multiple aesthetics produces better graph drawings. Journal of Visual Languages & Computing **24**(4), 262 – 272 (2013)
24. Kamada, T., Kawai, S.: An algorithm for drawing general undirected graphs. Information Processing Letters **31**(1), 7 – 15 (1989)
25. Kruiger, J.F., Rauber, P.E., Martins, R.M., Kerren, A., Kobourov, S., Telea, A.C.: Graph layouts by t-sne. Comput. Graph. Forum **36**(3), 283–294 (2017)
26. Kruskal, J.B.: Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. Psychometrika **29**(1), 1–27 (1964)
27. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems. pp. 8024–8035 (2019)
28. Purchase, H.: Which aesthetic has the greatest effect on human understanding? In: Proceedings of the 5th International Symposium on Graph Drawing. pp. 248–261. Springer (1997)
29. Radermacher, M., Reichard, K., Rutter, I., Wagner, D.: A geometric heuristic for rectilinear crossing minimization. In: The 20th Workshop on Algorithm Engineering and Experiments. p. 129–138 (2018)
30. Schulz, A.: Drawing 3-polytopes with good vertex resolution. J. Graph Algorithms Appl. **15**(1), 33–52 (2011)
31. Shabbeer, A., Ozcaglar, C., Gonzalez, M., Bennett, K.P.: Optimal embedding of heterogeneous graph data with edge crossing constraints. In: NIPS Workshop on Challenges of Data Visualization (2010)
32. Shepard, R.N.: The analysis of proximities: multidimensional scaling with an unknown distance function. Psychometrika **27**(2), 125–140 (1962)
33. Smilkov, D., Thorat, N., Assogba, Y., Nicholson, C., Kreeger, N., Yu, P., Cai, S., Nielsen, E., Soegel, D., Bileschi, S., Terry, M., Yuan, A., Zhang, K., Gupta, S., Sirajuddin, S., Sculley, D., Monga, R., Corrado, G., Viegas, F., Wattenberg, M.M.: Tensorflow.js: Machine learning for the web and beyond. In: Proceedings of Machine Learning and Systems 2019, pp. 309–321 (2019)
34. Wang, Y., Wang, Y., Sun, Y., Zhu, L., Lu, K., Fu, C.W., Sedlmair, M., Deussen, O., Chen, B.: Revisiting stress majorization as a unified framework for interactive constrained graph visualization. IEEE transactions on visualization and computer graphics **24**(1), 489–499 (2017)
35. Ware, C., Purchase, H., Colpoys, L., McGill, M.: Cognitive measurements of graph aesthetics. Information visualization **1**(2), 103–110 (2002)

36. Zheng, J.X., Pawar, S., Goodman, D.F.: Graph drawing by stochastic gradient descent. IEEE transactions on visualization and computer graphics **25**(9), 2738–2748 (2018)

8 Appendix

The following table summarizes the objective functions used to optimize the nine drawing criteria via different optimization methods.

Property	Gradient Descent	Subgradient Descent	Stochastic Gradient Descent
Stress	$\sum\limits_{i<j} w_{ij}(\lvert X_i - X_j\rvert_2 - d_{ij})^2$	$\sum\limits_{i<j} w_{ij}(\lvert X_i - X_j\rvert_2 - d_{ij})^2$	$w_{ij}(\lvert X_i - X_j\rvert_2 - d_{ij})^2$ for a random pair of nodes $i, j \in V$
Ideal Edge Length	$\sqrt{\frac{1}{\lvert E\rvert} \sum\limits_{(i,j)\in E} (\frac{\lvert\lvert X_i - X_j\rvert\rvert - l_{ij}}{l_{ij}})^2}$ (Eq. 2)	$\frac{1}{\lvert E\rvert} \sum\limits_{(i,j)\in E} \lvert \frac{\lvert\lvert X_i - X_j\rvert\rvert - l_{ij}}{l_{ij}} \rvert$	$\lvert \frac{\lvert\lvert X_i - X_j\rvert\rvert - l_{ij}}{l_{ij}} \rvert$ for a random edge $(i, j) \in E$
Crossing Angle	$\sum\limits_i cos(\theta_i)^2$	$\sum\limits_i \lvert cos(\theta_i)\rvert$	$\lvert cos(\theta_i)\rvert$ for a random crossing i
Neighborhood Preservation	Lovász **softmax** [5] between neighborhood prediction (Eq.4) and adjacency matrix Adj	Lovász **hinge** [5] between neighborhood prediction (Eq.4) and adjacency matrix Adj	Lovász **softmax** or **hinge** [5] on a random node. (i.e. Jaccard loss between a random *row* of K in Eq. 4 and the corresponding row in the adjacency matrix Adj)
Crossing Number	Shabbeer et al. [31]	Shabbeer et al. [31]	Shabbeer et al. [31]
Angular Resolution	$\sum\limits_{(i,j),(j,k)\in E} e^{-\varphi_{ijk}}$	$\sum\limits_{v\in E} e^{-\varphi_{ijk}}$	$e^{-\varphi_{ijk}}$ for random $(i, j), (j, k) \in E$
Vertex Resolution	$\sum_{i,j\in V, i\neq j} ReLU(1 - \frac{\lvert\lvert X_i - X_j\rvert\rvert}{d_{max}\cdot r})^2$ (Eq. 7)	$\sum_{i,j\in V, i\neq j} ReLU(1 - \frac{\lvert\lvert X_i - X_j\rvert\rvert}{d_{max}\cdot r})$	$ReLU(1 - \frac{\lvert\lvert X_i - X_j\rvert\rvert}{d_{max}\cdot r})$ for random $i, j \in V, i \neq j$
Gabriel Graph	$\sum_{(i,j)\in E, k\in V\setminus\{i,j\}} ReLU(r_{ij} - \lvert X_k - c_{ij}\rvert)^2$ (Eq. 9)	$\sum_{(i,j)\in E, k\in V\setminus\{i,j\}} ReLU(r_{ij} - \lvert X_k - c_{ij}\rvert)$	$ReLU(r_{ij} - \lvert X_k - c_{ij}\rvert)$ for random $(i, j) \in E$ and $k \in V \setminus \{i, j\}$
Aspect Ratio	Eq. 5	Eq. 5	Eq. 5

Table 3. Summary of the objective functions via different optimization methods.